Institut für Informatik
der Technischen Universität München

Integration formaler Fehlereinflussanalyse in die Funktionsentwicklung bei der Automobilindustrie

Markus Pister

Institut für Informatik
der Technischen Universität München

Integration formaler Fehlereinflussanalyse in die Funktionsentwicklung bei der Automobilindustrie

Markus Pister

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h. c. Manfred Broy
2.	Univ.-Prof. Dr. Bernd Heißing

Die Dissertation wurde am 26.06.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 15.12.2008 angenommen.

Danksagung

An dieser Stelle möchte ich mich bei all den Menschen bedanken, die mich mit ihrer Zeit, ihren Anregungen und der Bereitstellung günstiger organisatorischer Rahmenbedingungen unterstützt und somit diese Dissertation ermöglicht haben.

Allen voran gilt mein Dank Herrn Prof. Dr. Dr. h. c. Manfred Broy für seine Betreuung und seinen vielfältigen Beistand. Die Anstellung an seinem Lehrstuhl eröffnete mir das Themenumfeld für diese Publikation. In angenehmer Erinnerung sind mir die hilfreichen Gespräche geblieben, die der konkreten Themenfindung gedient haben. Letztlich gab der gewährte zeitliche Freiraum wertvolle Entlastung in der finalen Phase.

Mein besonderer Dank gilt Dr. Jürgen Schuller. Aufgrund seiner Vermittlung konnte ich praxisnahe Erkenntnisse zum Thema der Erstellung von Funktionssicherheitsanalysen in der Industrie gewinnen. Seine kontinuierliche inhaltliche Begleitung und seine nützlichen Anregungen halfen maßgeblich bei der Gestaltung der Arbeit.

Des Weiteren möchte ich nicht versäumen, mich bei meinen Kollegen für die zahlreichen Erörterungen meiner Entwürfe und den daraus gewonnenen Erkenntnissen zu bedanken. Stellvertretend möchte ich hier Thomas Reiß, Dr. Leonid Kof, Christian Pfaller und David Cruz erwähnen.

Ich möchte mich hier weiter bei Prof. Dr. Bernd Heißing bedanken, der sofort zugesagt hat, die Dissertation als Zweitgutachter zu betreuen.

Mein Dank gilt auch Dr. Peter Felix Tropschuh, Dr. Uwe Koser und Dr. Ralf Schwarz. Sie gaben mir mit meiner Aufnahme beim INI.TUM interdisziplinäre Einblicke in diverse Entwicklungsthemen der Automobilindustrie und gewährten mir eine starke Unterstützung bei organisatorischen Aufgaben wie der Veröffentlichung dieser Arbeit.

Abschließen möchte ich die Danksagung mit meiner Familie, die mir von privater Seite her den Rücken gestärkt und mir so die Motivation und Zuversicht zur Vollendung dieser Arbeit gegeben hat.

Zusammenfassung

Hohe Anforderungen an die Funktionssicherheit eines Automobils fordern von der Entwicklung ein Vorgehen, bei dem die Einhaltung der Funktionssicherheit überprüft werden kann. In der Praxis werden zur Verifikation der Funktionssicherheit die Fehler-Möglichkeits- und Einfluss-Analyse (FMEA) und die Fehlerbaumanalyse (FTA) eingesetzt. Der steigende Anteil präziser formaler Modelle in der Entwicklung ermöglicht den steigenden Ansprüchen aus Normen und der Automobilindustrie hinsichtlich der Funktionssicherheit gerecht zu werden. Bei geeigneten formalen Modellen kann weiter die Verifikation teilweise automatisiert und so die Qualität der Entwicklung auf einen konstant hohen Stand gebracht werden.

Der Schwerpunkt der Arbeit ist der Entwurf formaler Modelle und Modellierungstechniken mit denen die FMEA und die FTA formal durchgeführt werden können. Die Modelle und Modellierungstechniken beschreiben das Verhalten der Systeme oder Beziehungen zwischen Systemverhalten. Sie sind für eine Integration mit bestehenden Artefakten der Entwicklung geeignet. Die Verhaltensmodellierung ist an die in der Entwicklung verwendeten Modellierungswerkzeuge, wie Simulink™, und an die verwendeten Dokumente der Entwicklung angepasst. Konsistent zur Verhaltensmodellierung werden Modellierungstechniken für Fehlverhalten definiert. Fehlverhalten werden als Modifikationen des Sollverhaltens ausgedrückt. Um die möglichen Fehlverhalten eines Systems zu erfassen, werden potentielle Fehler, die Fehlverhalten verursachen können, vorgegeben. Zu den jeweiligen Modellierungstechniken für Fehler wird allgemein der Begriff des Fehlerzusammenhangs formal definiert. Weiter werden spezifische in den Methoden FMEA und FTA verwendete Zusammenhänge formalisiert, um eine Automatisierung zu ermöglichen. Abschließend zeigt die Arbeit Möglichkeiten auf, die Durchführung der Analysen zu automatisieren.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

Formale Modelle

Mechatronische Systeme im Automobil sind heutzutage meist komplex, müssen strenge Rahmenbedingungen erfüllen und insbesondere in Hinsicht auf Funktionssicherheit hohen Qualitätsanforderungen entsprechen. Dies fordert von der Entwicklung ein Vorgehen, mit dem ein für Normen wie IEC61508 [DIN04] ausreichend funktionssicheres Produkt erstellt werden kann und bei dem die Einhaltung der Anforderungen an die Funktionssicherheit des Produktes überprüft werden können. Diesen Ansprüchen kann mit dem Einsatz formaler Methoden begegnet werden, da formale Methoden eine hohe Präzision bei der Beschreibung und somit eine den Normen angemessene Verifikation der Eigenschaften des Systems erlauben. Die Verifikation kann bei einer geeigneten formalen Modellierung teilweise automatisiert werden, womit die Qualität der Entwicklung auf einen konstant hohen Stand gebracht werden kann.

Informelle Beschreibungen

Nicht alle Beschreibungen eines Systems sind mit angemessenem Aufwand detailliert formalisierbar. Besonders auf höheren Abstraktionsebenen gibt es eine Reihe von Beschreibungen, die zwar für den Menschen gut lesbar und verständlich sind, aber vage und informell formuliert sind. Diese Beschreibungen haben häufig den Vorteil, in einem interdisziplinären Umfeld gut kommunizierbar zu sein. Sie können jedoch nicht formal verifiziert werden. Heutzutage findet deshalb in der Automobilindustrie die Entwicklung eines Produkts mit einer Mischung aus formalen und informellen Beschreibungen statt.

Sicherheitsanalyse

Bei der Entwicklung einer funktionalen Spezifikation eines mechatronischen Systems sind als ein wesentlicher Aspekt die Anforderungen an die Funktionssicherheit des Systems zu beachten. Die Implementierung eines mechatronischen Systems kann potenzielle Fehler aufweisen, die bereits bei der Erstellung entstanden sind (z.B. Programmierfehler), wie auch Fehler, die erst während der Laufzeit entstehen (z.B. Materialermüdung). Um Aussagen über die Funktionssicherheit eines Systems machen zu können, sind Fehler als ein unerwünschter, aber unvermeidbarer Teil eines

Systems explizit zu berücksichtigen. Eine Aktivität bei der Entwicklung ist deshalb die Durchführung von Sicherheitsanalysen, in denen potentielle Fehler identifiziert und deren Wirkungen analysiert werden. Zwei gängige Sicherheitsanalyse-Methoden sind die Fehler-Möglichkeit- und Einfluss-Analyse (FMEA[1]) und die Fehlerbaumanalyse (FTA[2]). Anhand der Ergebnisse dieser beiden Analysen können Anforderungen an die Funktionssicherheit verifiziert werden. Die Ergebnisse können als Rückkopplung in die Entwicklung fließen, wie auch als Nachweise der Einhaltung von Sicherheitsanforderungen verwendet werden.

Systeme

Die Systeme, welche analysiert werden, sind groß (über 10^4 Modellelemente) und heterogen. Es sind Regelsysteme einerseits für physikalische kontinuierliche Größen und andererseits für diskrete Schaltlogiken. Sie sind eine Mischung aus Datenflusssystemen und reaktiven Systemen. Die Verhaltensmodelle zu diesen Systemen sind Komponentenmodelle, welche mit Prädikaten, Automaten, Gleichungen und direkt in Programmiersprachen spezifiziert werden. Für diese verschiedenen Charakteristika und Spezifikationstechniken gibt es spezifische Lösungsmöglichkeiten zur formalen Sicherheitsanalyse. Offen ist hierbei ein methodischer Ansatz, der diese Lösungsmöglichkeiten in die bestehende Entwicklung heterogener Systeme integriert.

1.2 Beitrag der Arbeit

Integration

In dieser Arbeit werden die Sicherheitsanalyse-Methoden Fehler-Möglichkeit- und Einfluss-Analyse und Fehlerbaumanalyse mit einer weitgehend strukturierten Entwicklung mechatronischer Systeme integriert. Die Sicherheitsanalyse-Methoden werden in einem ganzheitlichen Ansatz an die formale Verhaltensmodellierung des Systems angepasst, wie auch die informellen Beschreibungen berücksichtigt.

formaler Anteil

Der Schwerpunkt der Arbeit ist die Integration der Sicherheitsanalysetechniken in die formalen Modellierungstechniken, mit denen das Verhalten mechatronischer Systeme und derer Umgebung modelliert werden kann. Hierzu gehören unter anderem Referenzmodelle die allgemein den Aufbau der Systeme beschreiben, wie auch Modellierungstechniken zur konkreten Verhaltensbeschreibung der Systeme. Der in dieser Arbeit verwendete formale Teil der Modellierungstechniken ist an FOCUS [BS01] und Simulink™ [Ang07] angelehnt. Passend zu den Verhaltensmodellen und den jeweiligen Modellierungstechniken werden spezifische Fehlverhalten definiert. Diese Fehlerverhalten können als Modifikationen des Sollverhaltens in die formalen Verhaltensmodelle induziert werden. Um eine möglichst vollständige Analyse aller potentiellen Fehlverhalten für ein konkretes System zu erfassen, werden allgemeine Sammlungen möglicher potentieller Fehler für entsprechende Komponenten vorgegeben, welche die Fehlverhalten verursachen können. Zu den Fehlverhalten wird der Begriff des Fehlerzusammenhangs formal definiert, um eine automatisierte Analyse

[1] Failure Mode and Effect Analysis
[2] Fault Tree Analysis

der Wirkung von Fehlern prinzipiell zu ermöglichen. Schließlich werden Möglichkeiten aufgezeigt, die Ursache-Wirkungs-Beziehungen weitgehend automatisch zu ermitteln.

informeller Anteil

Der verbleibende informelle Teil der Spezifikation und auch die Sicherheitsanalyse-Methoden, die für informelle Beschreibungen entworfen sind, werden mit dem formalen Teil verknüpft. Zu den informellen Anteilen gehören abstrakte Verhaltensanforderungen (z.B. „Gegenlenken bei Untersteuern“), wie auch vom Verhalten unabhängige Anforderungen. Informelle Fehlerbeschreibungen lassen keine automatische Verarbeitung zu und werden deßhalb manuell mit den formalen Fehlerbeschreibungen und Modellen verknüpft und manuell auf ihre Wirkung hin untersucht.

Schnittstellen

Diese Arbeit konzentriert sich auf Verhaltensmodelle. Die Schnittstelle zu anderen physikalischen Modellen, welche nicht im Sollverhalten modelliert werden (wie z.B. dem räumlichen Aufbau eines Systems), wird umrissen und diskutiert. Des Weiteren fokussiert die Arbeit die funktionale Konstruktion der Systeme. Es wird so lediglich die Modellierung von Fehlern, welche Abweichungen von dem spezifizierten Verhalten sind, und die Analyse der Wirkung dieser Fehler innerhalb der Funktionsarchitektur analysiert. Schnittstellen zur Weiterverfolgung der Wirkungen und Ursachen der Fehler in die Fertigung, zum Entwicklungsprozess und zur Nutzung werden informell vorgestellt.

Bewertung des Ansatzes

Die Anwendbarkeit des in dieser Arbeit dargestellten Ansatzes wird anhand eines Beispieles verdeutlicht. Um Aussagen über die Wirtschaftlichkeit und Anwendbarkeit des Ansatzes treffen zu können, werden die Methoden auf zwei weitere reale Projekte der Automobilindustrie angewendet, und mit dem dort üblichen Vorgehen verglichen. Des Weiteren wird der vorgestellte Ansatz mit anderen aktuellen Ansätzen aus der Literatur verglichen und im Gebiet der Sicherheitsanalyse eingeordnet.

Mit dieser Arbeit wird durch die Vorstellung eines Ansatzes zur formalen teilautomatisierten funktionalen Sicherheitsanalyse, ein Beitrag zur effizienten Entwicklung funktionssicherer mechatronischer Systeme geleistet.

1.3 Begleitende Fallstudien

Lenksystem

Die in dieser Arbeit vorgestellten Modelle, Techniken und Methoden werden anhand zweier begleitender Fallstudien verdeutlicht. Gegenstand der ersten Fallstudie ist ein *Lenksystem* eines Fahrzeugs mit einer *Überlagerungslenkung*. Eine Überlagerungslenkung berechnet abhängig von der Fahrsituation einen Überlagerungswinkel und addiert diesen zum Lenkradwinkel. Sie hat zwei Teilfunktionen: die *geschwindigkeitsabhängige Lenkübersetzung* und die *Stabilisierungsfunktion* (Fahrer unterstützende Lenkkorrektur bei instabiler Fahrsituation). Diese Fallstudie ist ähnlich den Beschreibungen aus [BSOB04] (S. 104 ff.).

Differenzial Die zweite Fallstudie ist die Applikationssoftware eines Fahrwerkregelsystems. Die Software berechnet abhängig von der Fahrsituation ein *Differenzenmoment* zwischen den Rädern, welches dann umgesetzt wird. Das System ist eine Stabilisierungsfunktion, mit der bremsende Eingriffe des ESP, welche den Fahrkomfort und die Sportlichkeit senken, reduziert werden können.

Verwendung Beide Systeme kommen aus dem Fahrwerk-Regelbereich und haben ähnliche Eigenschaften. Sie erfüllen die in Abschnitt 1.1 aufgeführte Heterogenität hinsichtlich Datenfluss-Systemen und reaktiven Systemen, sowie der Modellierung mit C-Code, Automaten und Gleichungen. Die Fallstudien werden in dieser Arbeit begleitend in den Kapiteln zur Verdeutlichung der Ansätze und zur Bewertung der Beiträge der Arbeit verwendet.

1.4 Related Work

Fehlermodelle Eine ausführliche Beschreibung der verwandten Arbeiten findet jeweils in den Kapiteln mit den Beiträgen statt. Prinzipiell gibt es verschiedene Modelle für Fehler, Zusammenhänge zwischen Fehlern und verschiedene spezifische Techniken zur Analyse. Ein allgemeines denotationelles Modell ist in [Bre01] zu finden. Für Automaten ist es das Hinzufügen von Zuständen und Transitionen und die Erreichbarkeit von unerwünschten Zuständen bzw. Ereignissen [Thu04, ORS05, SSL+95]. Für Datenflusssysteme ist es die Abweichung von den spezifizierten Sollwerten der Ausgaben [Str04, MP01, BSN07].

Automaten Für die formale Sicherheitsanalyse gibt es eine Menge von Ansätzen zu spezifischen Systemen. Am weitesten fortgeschritten ist die Analyse für Systeme, welche komplett in endlichen Automaten modelliert sind. Mit Diagnoseautomaten aus [SSL+95],[SSL+96] kann die Wirkung einzelner Fehler im Sinne unerwünscht ausgeführter Ereignisse ermittelt werden. In [OR04],[ORS05] wird ein Ansatz vorgestellt, mit dem die Dokumente der Sicherheitsanalyse hinsichtlich der Zusammenhänge von Automaten zu formalen Anforderungen (temporale Logik [CG01, Lam02]) voll generiert werden können. Die endlichen Automaten können um den Faktor Zeit erweitert werden. [Thu04] stellt die Analyse für Systeme, die in Interval Temporal Logic (ITL)[CMZ02] beschrieben sind, vor.

Datenfluss Liegt der Schwerpunkt mehr auf dem Datenfluss, so ist in [Str04], [Str06], [PCB+55] ein Ansatz zu finden, mit dem Zusammenhangsmodelle zu Abweichungen generiert werden, welche mit einem Constraint-Solver verifiziert werden können. In diesem Bereich gibt es zahlreiche Ansätze [BSN07], welche Tests verwenden, um auf Zusammenhänge zu schließen. Weitere Arbeiten, wie [Sac01], konzentrieren sich auch auf eine geeignete Abstraktion der Systeme, um die großen Wertebereiche der Variablen auf eine zur Analyse geeignete kleinere Menge zu reduzieren.

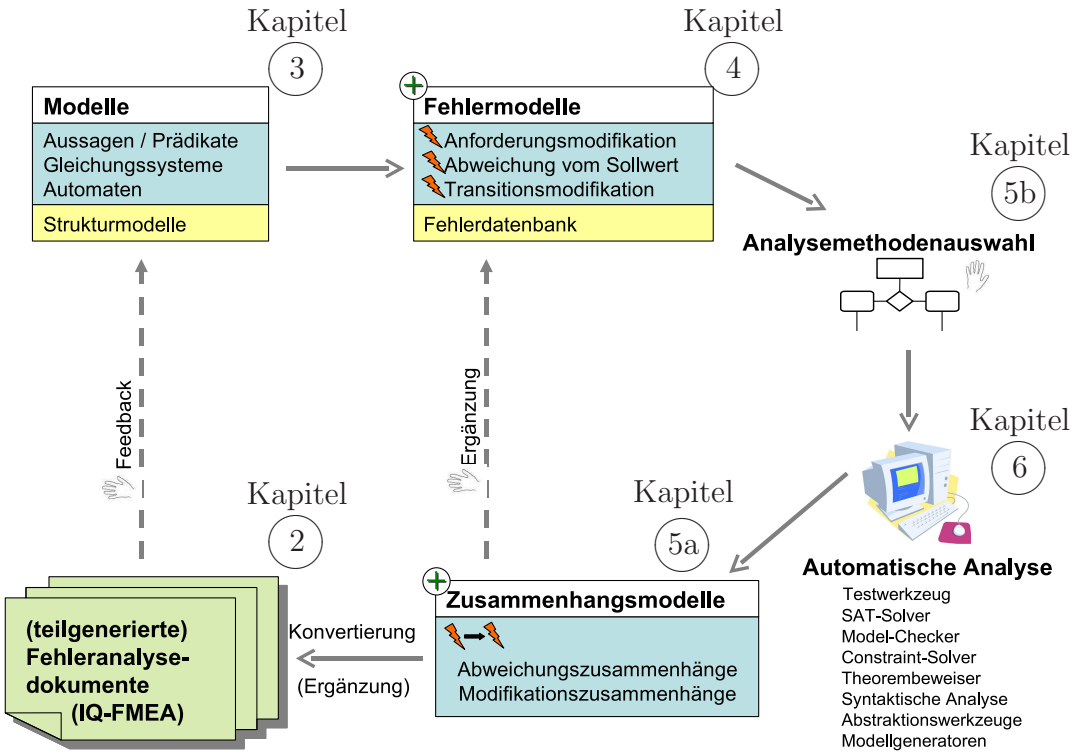


Abb. 1.1: Überblick über die Struktur der Arbeit

1.5 Kapitelübersicht

Struktur

Die Kapitel der Arbeit können verschiedenen Stationen einer teilweise automatisierten Fehlerauswirkungsanalyse zugeordnet werden. Abbildung 1.1 zeigt den Kreislauf, der ausgehend von System- und Fehlermodellen über eine teilweise automatisierte Analyse Zusammenhangsmodelle zwischen Anforderungen und Fehlern erstellt und diese geeignet dokumentiert. Die Reihenfolge der Kapitel sieht zuerst in Kapitel 2 eine Einführung in die informelle Fehleranalyse vor, in deren Dokumente die Ergebnisse der automatisierten Analyse integriert werden. Kapitel 3 beschreibt Modelle, welche aus der Entwicklung kommen und als Grundlage für die Fehleranalyse dienen. Kapitel 4 beschreibt ein formales Fehlermodell welches eine Grundlage für Methoden und mögliche Automatisierungen der Sicherheitsanalyse bildet. Kapitel 5 widmet sich zwei Themen. Es werden zum Einen formale Modelle von Zusammenhängen zwischen Fehlern beschrieben und zum Anderen spezifische Methoden zur Ermittlung der Zusammenhänge vorgestellt, welche allerdings die Automatisierung noch nicht berücksichtigen. Kapitel 6 gibt einen Überblick über die Möglichkeiten verschiedener Techniken zur automatischen Durchführung der Analysen. Im Detail ist die Gliederung folgende:

Sicherheitsanalyse

Kapitel 2 enthält eine Einführung in die Grundlagen einer Sicherheitsanalyse und stellt zwei in der Industrie angewandte Verfahren vor. Dazu werden allgemein die grundlegenden Begriffe, die bei einer Sicherheitsanalyse auftreten, informell defi-

niert. Im Wesentlichen werden die Begriffe *Fehler* und *Funktionssicherheit* definiert. Neben der Terminologie wird für die Sicherheitsanalyse und den Entwicklungsprozess aufgezeigt, wo Schnittstellen bestehen. Schließlich werden zwei konkrete Methoden (*Fehlermöglichkeits- und Einflussanalyse* und *Fehlerbaumanalyse*) und deren Dokumente vorgestellt, wie sie momentan in der Industrie Anwendung finden. Auf Basis dieses Kapitels betrachten die folgenden Kapiteln der Arbeit, in wie fern die Methoden formalisiert, deren Ergebnisse automatisch generiert und in den Entwicklungsprozess integriert werden können.

Systemmodell

Kapitel 3 stellt die Systemmodellierung vor. Es wird anhand informeller Referenzmodelle über den Aufbau und die Eigenschaften eingebetteter Systeme die in dieser Arbeit gegebene Systemvorstellung aufgebaut, in welche die formalen Modelle eingeordnet werden können. Es folgt eine Beschreibung der formalen Modelle des Verhaltens und der Struktur der Artefakte, auf denen das Verhalten implementiert ist. Die Verhaltensmodelle können mit Modellierungstechniken, die spezifisch auf die Beschreibung bestimmter Aspekte der Modelle zugeschnitten sind, beschrieben werden. Hierzu gehören Prädikate und Automaten (so wie deren Sonderformen 'aufgelöste' Gleichungen und Betriebsmodi-Automaten). Schließlich werden Abhängigkeiten zwischen Modellen gezeigt, die durch Überführungen in präzisere Modelle und durch Zusammenführungen aus Modellen von Teilsystemen entstehen. Ein Beitrag dieses Kapitels ist auch die Herausstellung von Ansatzpunkten für die Fehlermodellierung und -ermittlung. Auf dieser Basis können in den folgenden Kapiteln Beschreibungen zu Fehlern definiert werden.

Fehlermodell

Kapitel 4 zeigt Fehlermodelle und Modellierungstechniken für diese Fehlermodelle, welche auf die Systemmodellierung des letzten Kapitels zugeschnitten sind. Basis dafür ist das Fehlermodell von [Bre01] für reaktive Systeme. In diesem kann allgemein für stromgebundene Black-Box-Spezifikationen, wie auch für Automaten, eine Modifikation getrennt von der Soll-Spezifikation durch Hinzufügen und Entfernen von Tupeln der Relation, die das Verhalten beschreibt, modelliert werden. Der Beitrag dieses Kapitels ist die Erweiterung dieser Modellierungstechniken für zustandsgebundene Prädikate, sowie für deren Sonderformen, der zu Ausgaben aufgelösten Gleichungen und der Betriebsmodi, deren Zustand nicht direkt abgelesen werden kann. Insbesondere wird die Kombinierbarkeit der Fehler detaillierter untersucht. Abgerundet mit einem Datenmodell und einer Sammlung von typischen Fehlern der Implementierungsebene entsteht so ein umfassendes Fehlermodell für die Funktionssicherheitsanalyse der Software eingebetteter Systeme.

Fehlerzusammenhänge

Kapitel 5 beschreibt verschiedene Arten von Fehlerzusammenhängen und definiert diese formal. Es wird allgemein für Verhaltensrelationen definiert, was ein Fehlerzusammenhang in einem zusammengesetzten System ist. Des Weiteren wird zur Ermittlung dieser Fehlerzusammenhänge der Begriff Fehlerauswirkung definiert. Fehlerauswirkungen sind Abweichungen von Kanalwerten aufgrund von Fehlverhalten. In beiden Fällen kann häufig nicht die exakte Folge angegeben werden. Hierzu wurde eine Abbildung definiert, anhand derer die Folgefehler eingegrenzt werden können. Auf dieser Basis wird eine Menge von Fehlerzusammenhangsmodellen vor-

gestellt, welche spezifisch für die Modellierungstechniken aus Kapitel 3 zugeschnitten sind (vgl. Abb. 1.1, Punkt 5a). Die komplexen Zusammenhänge zwischen Zeit und Anforderungen können vereinfacht modelliert werden und spezifisch an Methoden, Möglichkeiten, zu untersuchende Systeme und Anforderungen an die Analyse angepasst werden. Zu den jeweiligen spezifischen Zusammenhangsmodellen werden Methoden angegeben, um eine Funktionssicherheitsanalyse eines Modells systematisch durchzuführen (vgl. Abb. 1.1, Punkt 5b). Diese teils aus der Literatur entnommenen Methoden werden an die Fehlermodelle aus Kapitel 4 angepasst bzw. geeignet erweitert. Es wird mit diesen formalen Modellen und Methoden eine Basis für die systematische, softwaregestützte Ermittlung von Fehlerabhängigkeiten geschaffen.

Automatisierung

Kapitel 6 beschreibt eine Menge von Analysewerkzeugen, mit denen Teile der Fehlerzusammenhänge in einem Modell automatisch ermittelt werden können. Hierzu werden in einem ersten Schritt die Vor- und Nachteile hinsichtlich Effizienz und Mächtigkeit der Analysen gelistet. Zu den Analysewerkzeugen gehören: Testgeneratoren, boolesche Verifikationswerkzeuge (SAT-Solver, Modelchecker und Theorembeweiser) mit angebundenen Constraint-Solvern, syntaktische Analysewerkzeuge, Modellgeneratoren und Abstraktionswerkzeuge. In einem zweiten Schritt werden Anfragen an die Werkzeuge vorgestellt, mit denen die Zusammenhänge ermittelt werden können.

Schließlich werden in Kapitel 7 die Ergebnisse der Arbeit zusammengefasst.

Kapitel 2

Funktionssicherheitsanalyse

Begriff

Während der Erstellung und des Einsatzes eines Produktes können Fehler entstehen. Es können zum Beispiel während der Erstellung Softwarekomponenten falsch implementiert oder falsch spezifiziert werden. Im Einsatz können Datenkabel brechen oder Daten durch elektromagnetische Störungen verfälscht werden. Diese Fehler können unter Umständen gravierende Folgen wie z.B. Unfälle mit Personenschaden haben, wenn sie beim Einsatz des Produktes auftreten. Selbst geringere Folgen können noch einen negativen Einfluss auf den Umsatz und Gewinn eines Unternehmens haben ([Voe02], S. 2, Absatz 2). Zur Vermeidung von Fehlern ist deshalb die Analyse hinsichtlich der potentiellen Fehler eines Produktes eine wesentliche Aufgabe der Entwicklung ([Jon00]).[1] Eine Aktivität ist hierbei die *Funktionssicherheitsanalyse*.[2] Sie befasst sich mit den potentiellen Fehlern, die ein Produkt direkt nach seiner Fertigstellung haben kann oder die beim Einsatz eines Produktes entstehen können. Die in dieser Arbeit betrachteten Funktionssicherheitsanalysen sind in Bezug auf die Einsatzzeit der Produkte statisch, also die Analysen sind nach Fertigstellung des Produktes abgeschlossen und die Ergebnisse ändern sich während des Einsatzes nicht mehr.

Zweck

Die Ergebnisse der Funktionssicherheitsanalyse werden in Dokumenten festgehalten. Diese Ergebnisse unterstützen drei Aufgaben, die bei der Entwicklung sicherheitsrelevanter Produkte durchzuführen sind. Die Aufgaben und der spezifische Beitrag der Analyse sind folgende (siehe [Lev95], S. 289; [MK05], S. 282):

- *Entwicklung*: Die Betrachtung potentieller Fehler eines Systems, um diese oder deren Folgen zu eliminieren oder zu kontrollieren.
- *Operationelles Management*: Die Betrachtung potentieller Fehler eines Systems, in Hinblick auf eine Verbesserung der Richtlinien, der Formulierung

[1] Die Betrachtung von zwei Projekten im Automobil-Bereich hat gezeigt, dass der Aufwand für Funktionssicherheit mindestens ein Viertel der Entwicklungszeit in Anspruch nimmt.

[2] Der mehrdeutige Begriff Sicherheit wird in dieser Arbeit nicht als Angriffssicherheit (engl. Security) verstanden, sondern als Funktionssicherheit (engl. Safety).

Produkt *(Konstruktion)*	**Prozess** *(Nutzung)*	**Produktion** *(Fertigung)*
- Hardware	- Wartung	- Aufstellung
· Elektrik	· Konfiguration	- Chemie
· Mechanik	· Dokumentation	- Bearbeitung
· Schnittstellen	· Training	- Training
- Software	- Anwendung	
· SW-Funktionen	· Betriebsmodi	
· Hardwareschnittstellen	· Training	
- Funktionen	· Dokumentation	
· System	· Überlastung	
· Subsystem	· Nutzungsschnittstelle	

Tabelle 2.1: Unterteilung der Betrachtung eines Produktes (siehe [Eri05], S. 242)

von Sicherheitsvorschriften bzw. -hinweisen und der Steigerung der Motivation hinsichtlich der Effizienz und Zuverlässigkeit der Tätigkeiten beim Umgang mit dem Produkt

- *Zertifizierung*: Das Aufzeigen bzw der Nachweis der Funktionssicherheit eines Systems zur Akzeptanz der Öffentlichkeit oder anderen Institutionen.

Produktorientierung

Funktionssicherheitsanalysen können in drei sich ergänzende Kategorien unterteilt werden, die in Tabelle 2.1 aufgeführt sind (siehe [Eri05], S. 242). Die mit dem Präfix *Produkt* gekennzeichnete Kategorie befasst sich mit dem Aufbau des Produktes, also der Architektur und der Gestaltung der Implementierung. Sie betrachtet in einer White-Box-Sicht, welche Fehler in Produktteilen entstehen können und wie diese Fehler die Funktionalität des Produktes und somit die Umwelt beeinflussen können. Die Kategorie *Prozess* befasst sich mit den Aspekten der Nutzung und Wartung des Produktes. Das Produkt wird hierbei als eine Black-Box betrachtet, dessen Nutzungsschnittstelle im Vordergrund steht. Die Kategorie *Produktion* betrachtet Fehler der Maschinen, der Mitarbeiter und des Produktionsprozesses selbst.

✍ Diese Arbeit fokussiert sich auf die Produkt-Funktionssicherheitsanalyse. Im weiteren Verlauf des Kapitels wird deshalb der Begriff *Sicherheitsanalyse* als Synonym des Begriffs Produkt-Funktionssicherheitsanalyse verwendet. Dies heißt insbesondere, dass die Fehler, die sich auf die Nutzung und die Fertigung beziehen, nur dann betrachtet werden, wenn es zur Bewertung eines Fehlers notwendig ist.

Kapitelübersicht

Dieses Kapitel beschreibt Grundlagen der Sicherheitsanalyse sowie zwei der in der Industrie angewandten Verfahren. Abschnitt 2.1 definiert allgemein die grundlegenden Begriffe, die bei einer Sicherheitsanalyse auftreten. Dabei werden die Begriffe *Fehler* und *Funktionssicherheit* definiert. Abschnitt 2.2 beschreibt die Zusammenhänge der Sicherheitsanalysen mit dem Entwicklungsprozess. Schließlich stellen die Abschnitte 2.3 und 2.4 die Methoden *Fehler-Möglichkeit- und Einfluss-Analyse* und *Fehlerbau-*

manalyse vor, wie sie momentan in der Industrie Anwendung finden und in der Literatur beschrieben werden.

2.1 Definitionen (Terminologie)

Die Begriffe, die in dem Feld der Sicherheitsanalyse vorkommen, werden in diesem Abschnitt allgemein definiert. Im weiteren Verlauf der Arbeit (Kapitel 4) folgen formale Definitionen, die auf spezifische Sichten und Methoden hin spezialisiert sind. Die Definitionen der Begriffe zum Thema Sicherheitsanalyse sind nicht einheitlich (siehe [Lev95], [Eri05], [Bre01], [Ech90], [Thu04], [DIN04]). Ziel dieses Abschnitts ist die Hinführung zur Begriffswelt der Sicherheitsanalyse und die Erklärung, wie die Begriffe in dieser Arbeit verstanden werden. Im Wesentlichen orientieren sich die Definitionen an der IEC 61508 ([DIN04]). In Abschnitt 2.1.1 wird der Fehlerbegriff (Fehler, Versagen, Abweichung) definiert. Abschnitt 2.1.2 widmet sich dem Zusammenspiel von Fehlern als Kombination und in einer Ursache-Wirkungs-Beziehung. Abschnitt 2.1.3 definiert Begriffe zur Bewertung der Fehlerwirkung (Vorfall, Gefahr, Risiko) und definiert schließlich die Funktionssicherheit, welche für ein System bei der Sicherheitsanalyse geprüft wird.

2.1.1 Fehler

Fehler

Der grundlegende Begriff für eine Sicherheitsanalyse ist der des Fehlers. Auf dessen Definition wird die weitere Begriffswelt aufgebaut. Die Betrachtung des Fehlerbegriffs kann sowohl aus der Ereignis-Sicht wie aus der Zustands-Sicht erfolgen. Die Definition wird aus drei Teilen aufgebaut, welche separat definiert werden.[3]

Definition 2.1.1 (Fehler)
Ein *Fehler* ist ein Versagen (Def. 2.1.2), eine Abweichung (Def. 2.1.3) oder ein Fehlzustand (Def. 2.1.4). ⌟

Versagen

Ereignisse und deren kausale Zusammenhänge beschreiben den dynamischen Anteil eines Systems ([BS01]; [Lev95]; [Bal01]). In einem System treten Ereignisse auf, die Folgeereignisse implizieren. Das Soll-Verhalten einer Komponente ist dadurch definiert, dass sie auf eintretende Ereignisfolgen unter bestimmten Rahmenbedingungen entsprechende Ereignisfolgen liefert. Ein Versagen ist eine Abweichung vom Soll-Verhalten:

[3] In der Literatur findet man alternativ auch eine Auftrennung des Begriffs Fehler in falsches menschliches Handeln und Fehler von Maschinen ([Mus99],S.85). Fehler von Maschinen werden in nach Außen hin wahrnehmbare Fehler und in Defekte, die eine Ursache für die wahrnehmbaren Fehler sind, eingeteilt ([Mus99], S.41).

Definition 2.1.2 (Versagen)
Ein *Versagen* ist ein beobachtbares Verhalten eines Systems, welches nicht das spezifizierte Soll-Verhalten erfüllt.[4] ⌟

Beobachtbarkeit

Ein Versagen zeichnet sich dadurch aus, dass sich die Menge der Ereignisse und der kausalen Zusammenhänge des Soll-Verhaltens geändert hat. Es können nicht mehr nur die geplanten Ereignisabläufe im System ablaufen, sondern auch Abläufe stattfinden, in denen Fehlerereignisse vorkommen. Beobachtet man ein Versagen, so ist dieses daran zu erkennen, dass die Ausgaben eines Systems von den Soll-Ausgaben abweichen. Bezogen auf die Eingaben und Ausgaben eines Systems ist ein Fehler eine Abweichung:

Definition 2.1.3 (Abweichung)
Eine *Abweichung* ist eine Nichtübereinstimmung der Eingaben bzw. Ausgaben eines Systems mit den spezifizierten Soll-Eingaben bzw. Soll-Ausgaben.[5] ⌟

Die Definition einer Abweichung wird am Beispiel eines Steuergerätes einer Fahrwerks-Regelung verdeutlicht:

Beispiel 2.1.1 (Abweichung)
Ein Steuergerät hat unter anderem die Soll-Funktion, bei untersteuern des Fahrzeuges das passende Gegenlenken zu veranlassen (siehe Ausgabe (I)). Die Abweichung in Ausgabe (II) ist eine unerwünschte Abweichung, bei der die falsche Gegenlenkfunktion ausgewählt wird. Dies ist im Soll-Verhalten nicht vorgesehen. Ausgabe (III) ist ein Beispiel für eine Abweichung, welches das Ausbleiben einer Soll-Ausgabe ist. Hier leistet das Steuergerät seine Funktion nicht.

(I) (untersteuern →) Ausgabe: Kommando zum Gegenlenken (Modus Untersteuern)
(II) (untersteuern →) Ausgabe: *Kommando zum Gegenlenken (Modus Übersteuern)*
(III) (untersteuern →) Ausgabe: *(kein Kommando zum Gegenlenken)* ⌟

Zustands-Sicht

In der *Zustands-Sicht* wird das Verhalten eines Systems mit Zuständen und Zustandsübergängen (Transitionen) beschrieben ([Bal01]; [Lev95]). Ein System ist in einem Zustand und geht entsprechend einer ausgewählten Transition in einen anderen Zustand über. Das Soll-Verhalten einer Komponente wird definiert, indem für die Zustände (inklusive der Zustände der Eingabevariablen) die das System haben kann, jeweils nur bestimmte mögliche Folgezustände gegeben sind. Die zustandsorientierte Sicht lässt sich mit der ereignisorientierten Sicht kombinieren, in dem die

[4] [DIN04]: Beendigung der Fähigkeit einer Funktionseinheit, eine geforderte Funktion auszuführen.

[5] [DIN04]: Nichtübereinstimmung zwischen Rechenergebnissen, beobachteten oder gemessenen Werten oder Beschaffenheiten, und den betreffenden wahren, spezifizierten oder theoretisch richtigen Werten oder Beschaffenheiten.

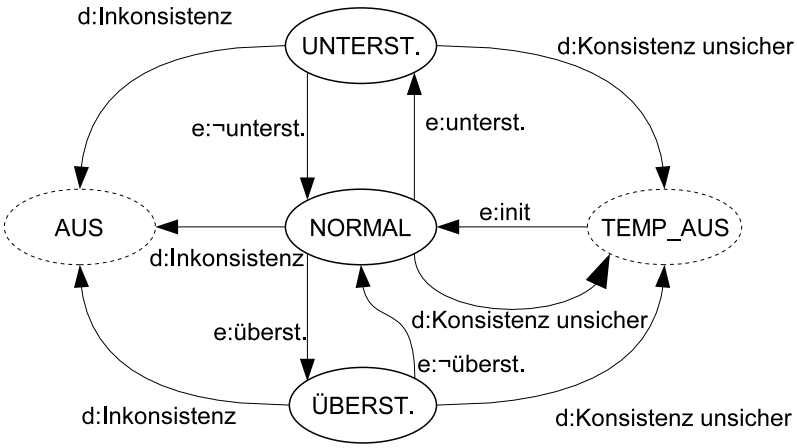


Abb. 2.1: Beispiel eines Automaten mit Fehlerzuständen

Zustandsübergänge mit Ereignissen verbunden werden (siehe [Bro98]). Die zustandsorientierte Sicht konzentriert sich auf den Zustand eines Systems, der ein Versagen eines Systems oder Abweichungen der Eingaben und Ausgaben verursacht. Es gilt folgende Definition:

Definition 2.1.4 (Fehlzustand)
Ein *Fehlzustand* ist ein interner Zustand, der ein Versagen verursachen kann.[6] ⌟

Beispiel 2.1.2 (Fehlzustand)
Abbildung 2.1 veranschaulicht die Definition des Begriffs Fehlzustand anhand des Steuergerätes aus Beispiel 2.1.1. Dieses schaltet abhängig von der Erkennung e der Fahrsituation in den jeweiligen Modus zur Stabilisierung der Fahrsituation. Die Erkennung der Fahrsituation wird von einer Diagnose d überwacht. Erkennt diese Unstimmigkeiten, so schaltet sie das System ab. In diesen Zustand sollte das System aber nie kommen, da die Stabilisierung dann nicht verfügbar ist (Fall III in Bsp. 2.1.1). Dieser unerwünschte Zustand ist ein Fehlzustand. Abhängig von der Diagnose kann er wieder verlassen werden (TEMP_AUS) oder nicht (AUS). Ebenso kann der unerwünschte Zustand eintreten, in dem das System, wie in Fall II in den Zustand Übersteuern geht, obwohl es eigentlich im Zustand Untersteuern sein sollte. ⌟

HW-Ausfall

Die Fehlzustände eines Systems können feiner unterschieden werden, um die Herkunft der Fehler besser zu verstehen und damit auch die Maßnahmen passend ableiten zu können. Eine Gruppe von Fehlern sind die physikalischen technischen Fehler der Hardware eines Systems. Diese treten zur Laufzeit auf und müssen meist mit passenden physikalischen Maßnahmen, wie der physikalischen Architekturgestaltung oder geeigneten Wartungsmaßnahmen vermieden werden.

[6] [DIN04]: nicht normale Bedingung, die eine Verminderung oder den Verlust der Fähigkeit einer Funktionseinheit verursachen kann, eine geforderte Funktion auszuführen.

Definition 2.1.5 (Hardwareausfall)
Ein *Hardwareausfall* ist ein Fehlzustand, der aus physikalischen Mechanismen in der Hardware resultiert und zum Versagen eines Systems führt.[7] ⌟

Erstellung

Die bisher beschriebenen Fehlzustände waren Zustände, die erst während Laufzeit des Systems aufgetreten sind. Als Komplement gibt es eine Menge von Fehlzuständen, die bereits bei der Inbetriebnahme gültig sind. Diese Fehlzustände kommen aus einer fehlerhaften Erstellung des Produktes. Hier handelt es sich also nicht um einen Systemzustand im Sinne einer Variablenbelegung, sondern um eine falsch definierte bzw. implementierte Funktionalität, die im Produkt fest verankert ist. Bei einer Fokussierung auf eine Produkt-Sicherheitsanalyse interessieren nur die Fehler in der Entwicklung des Produktes, die zur Nichterfüllung einer Spezifikation führen.

Definition 2.1.6 (systematischer Ausfall)
Ein *systematischer Ausfall* ist ein Fehlzustand, dessen Ursache im Entwurf oder der Fertigstellung eines Systems liegt.[8] ⌟

Wahrscheinlichkeit

Jeder Fehlzustand hat Zeiten, zu denen er gültig ist (siehe [MK05], S. 289 ff.). Häufig ist es die Aufgabe der Analyse, eine Vorhersage zu treffen, wie wahrscheinlich es ist, dass ein Fehler in einem Betrachtungszeitraum gültig ist. Der Betrachtungszeitraum kann je nach Bedarf spezifisch definiert sein. Es kann z.B. die absolute Zeit ab Entstehung des Produktes sein, aber auch die Ausführungszeit ([Mus99]). Diese Einschätzung für die Erscheinung eines Fehlers ist ein wesentlicher Faktor bei der Entscheidung entsprechende Sicherheitsmaßnahmen zu ergreifen.

Definition 2.1.7 (Fehlerwahrscheinlichkeit)
Die *Fehlerwahrscheinlichkeit* ist die Wahrscheinlichkeit, dass ein Fehlzustand in einem definierten Zeitraum gültig ist. (siehe [MK05], S. 289) ⌟

2.1.2 Zusammenspiel von Fehlern

Kombination

Bei der Sicherheitsanalyse können auch Kombinationen von Fehlern betrachtet werden. Je nach Ziel der Analyse macht es Sinn, die einzelnen Teilfehler der Kombination zu fokussieren, oder die Kombination als ein Ganzes zu nehmen. Für eine Ermittlung der Wirkung spielt z.B. die Gesamterscheinung des Fehlers eine Rolle, wobei die Teilfehler meist nicht interessant sind. Umgekehrt sind bei der Suche nach Maßnahmen zur Vermeidung einer Kombination die verschiedenen Teilfehler relevant. Es ergibt sich somit aus der Betrachtung heraus die Unterscheidung in Einfach- und Mehrfach-Fehler:

[7] [DIN04]: Ausfall, der zu einem zufälligen Zeitpunkt auftritt und der aus einem oder mehreren möglichen Mechanismen in der Hardware resultiert, die zu einer Verschlechterung der Bauteile führen.

[8] [DIN04]: Ausfall, bei dem eindeutig auf eine Ursache geschlossen werden kann, die nur durch eine Modifikation des Entwurfs oder des Fertigungsprozesses, der Art und Weise des Betreibens, der Bedienungsanleitung oder anderer Einflussfaktoren beseitigt werden kann.

Definition 2.1.8 (Einfachfehler)
Ein *Einfachfehler* ist ein Fehler, der als atomarer, nicht aus Teilfehlern kombinierter, Fehler betrachtet wird. ⌟

Definition 2.1.9 (Mehrfachfehler)
Ein *Mehrfachfehler* ist ein aus einer Menge von Einfachfehlern zusammengesetzter Fehler. ⌟

Ursache und Wirkung

Ein wesentlicher Bestandteil der Analyse eines Fehlers ist es, ihn mit anderen Fehlern in Bezug zu bringen. Zum Einen interessiert die Wirkung des Fehlers. Ein Fehler kann einen anderen Fehler zur Folge haben. Zum Anderen kann ein Fehler einen anderen Fehler als Ursache haben. Auf diese Weise lässt sich eine Kette von Fehlern beschreiben, die in einer Ursache-Wirkungs-Beziehung stehen. Diese Kette wird *Ursache-Wirkungs-Kette* genannt. Beschreibt man diese Beziehung als eine Relation, so stehen auf der linken Seite der Tupel die Ursachen und auf der rechten Seite die Folgefehler. Diese Relation kann im Detail verschieden ausgeprägt sein. Formale Beschreibungen der Ausprägungen sind in [Bre01] und [Thu04] zu finden. Eine auf diese Arbeit hin zugeschnittene Beschreibung wird in Kapitel 5 gegeben.

Definition 2.1.10 (Folgefehler (FF))
Ein *Folgefehler* (FF) ist ein Fehler, der unter bestimmten Bedingungen, die in einem System erfüllt sind, unausweichlich als Folge eines anderen Fehlers entsteht. (vgl. [Lev95]) ⌟

Definition 2.1.11 (Fehlerursache (FU))
Eine *Fehlerursache* (FU) ist ein Fehler, der unter bestimmten Bedingungen, die in einem System erfüllt sind, zu einem anderen Fehler geführt hat. (vgl. [Lev95] und [Kre06], S. 16) ⌟

Begrenzung der U-W-Ketten

Die Ursache-Wirkungs-Ketten können theoretisch beliebig lang gesponnen werden. Es ist sinnvoll, die Betrachtung der Ketten an bestimmten Punkten zu beenden. In Richtung der Wirkung werden die Ketten meist an einer gegebenen Systemgrenze oder bei besonderen Folgeereignissen wie einem Unfall (siehe Abschnitt 2.1.3) beendet. In Richtung der Ursachen werden die Ketten mit primären Fehlern beendet. Die Ursachen dieser primären Fehler werden nicht in der aktuellen Analyse betrachtet, sondern in anderen ergänzenden Analysen, wie der Nutzungsanalyse, der Entwickungsprozessanalyse und der Fertigungsanalyse. Bei der Analyse der Kette werden die Wahrscheinlichkeiten für die primären Fehler statistisch erhoben, wohingegen die Wahrscheinlichkeiten der Folgefehler aus denen der primären Fehler ermittelt werden können.

Definition 2.1.12 (Primärfehler (PF))
Ein *Primärfehler* (PF) ist ein Fehler, für den im gegebenen Betrachtungsraum keine Fehler als Ursache gegeben sind. (vgl. [Kre06], S. 18) ⌟

U-W-Ketten Die Ursache-Wirkungs-Ketten, können in zwei Richtungen aufgespannt werden. Die Wirkung kann erstens entlang der Systemhierarchie zur jeweils umgebenden Komponente verfolgt werden. Zweitens kann die Wirkung auf der gleichen Ebene der Systemhierarchie weiterverfolgt werden. Die in dieser Arbeit betrachteten Ketten sind folgende:

$$\text{Hardwareausfall} \xrightarrow{\textit{ist ein}} \text{Fehlzustand} \xrightarrow{PF \quad FF} \underset{\textit{Ebene i}}{\text{Versagen}} \xrightarrow{FU \quad FF} \underset{\textit{Ebene i-1}}{\text{Versagen}} \xrightarrow{FU \quad FF} \ldots$$

$$\text{systematischer Ausfall} \xrightarrow{\textit{ist ein}} \text{Fehlzustand} \xrightarrow{PF \quad FF} \underset{\textit{Ebene i}}{\text{Versagen}} \xrightarrow{FU \quad FF} \underset{\textit{Ebene i-1}}{\text{Versagen}} \xrightarrow{FU \quad FF}$$

$$\ldots$$

$$\underset{\textit{Ebene i}}{\text{Versagen}} \xrightarrow{PF \quad FF} \underset{\textit{Ebene i}}{\text{Abweichung}} \xrightarrow{FU \quad FF} \underset{\textit{Ebene i}}{\text{Abweichung}} \xrightarrow{FU \quad FF} \underset{\textit{Ebene i}}{\text{Abweichung}} \xrightarrow{FU \quad FF} \ldots$$

Fehlerklasse Für allgemeinere Aussagen, können Fehler in *Fehlerklassen* zusammengefasst werden. Eine Klasse ist hierbei über die gemeinsamen Eigenschaften aller in ihr enthaltenen konkreten Fehler bestimmt. Es kann zum Beispiel eine Klasse "Wert zu groß" alle die Fehler umfassen, bei denen eine Funktion zu große Werte liefert, aber die Latenzzeit verschieden lang ist. Anhand dieser Zusammenfassung ist es z.B. einfacher, die Wahrscheinlichkeit für einen Fehler der Klasse abzuschätzen, statt jeweils die Wahrscheinlichkeiten der konkreten Fehler abzuschätzen.

2.1.3 Bewertung der Fehler

Bewertung Ziel der Sicherheitsanalyse ist die Bewertung der Funktionssicherheit. Auf Basis der Begriffe des vorherigen Abschnitts, die sich auf die Beschreibung der Fehler konzentriert haben, werden nun die Begriffe hinsichtlich der Bewertung der Fehler definiert. Aufbauend auf den Bewertungen von Fehlern mittels Schaden und Risiko wird der Begriff Funktionssicherheit definiert.

Schaden als Folge Die Wirkung eines Fehlers kann in einer Ursache-Wirkungs-Kette angegeben werden. Um die Wirkung bewerten zu können, werden die Ketten bis hin zu speziellen aussagekräftigen Folgen untersucht. Ein aussagekräftiges Maß ist der Schaden. Entsprechend können für die Bewertung Folgen betrachtet werden, die einen Schaden zur Folge haben. Hierbei spricht man von gefährlichen Vorfällen.

Definition 2.1.13 (gefährlicher Vorfall)
Ein *gefährlicher Vorfall* ist ein unerwünschtes Ereignis, das einen Schaden zur Folge hat.[9] ⌟

Schadensbewertung In der obigen Definition ist offen gelassen, wie ein Schaden bewertet wird. Die Angabe der Stärke des Schadens hängt sowohl von dem Betrachter, wie auch von dem Bewertungsaspekt ab. Der Schaden wird in einer Metrik angegeben, die absolut

[9] [DIN04]: Gefährdungssituation, die zu einem Schaden führt.

oder relativ sein kann. Es kann zum Beispiel aus Sicht des Fahrers die Stärke der Verletzung bzw. aus Sicht des Unternehmens der finanzielle Schaden als Metrik angegeben werden. Üblicherweise existieren für die verschiedenen Domänen spezifische Kataloge in denen Metriken für die Stärken der Schäden definiert werden.

Zustand der Umwelt

Möchte man allgemein die Wirkung der Fehler einer Fehlerklasse bewerten, so ist zu berücksichtigen, dass nicht immer ein Schaden entsteht. Ein Folgefehler tritt nur unter bestimmten Bedingungen auf, die im System und der Umwelt gelten. Entsprechend muss nicht jeder Fehler zu einem Unfall führen. So kann z.B. das Versagen der Lenkung dazu führen, dass das Fahrzeug über die Mittelspur einer Straße fährt. Kommt gerade kein Gegenverkehr und der Fahrer bringt das Fahrzeug wieder in die Spur, so entsteht für ihn kein Schaden. Allerdings ist nur kein Unfall zustande gekommen, da in der Umgebung bestimmte Umstände erfüllt waren. Diese Zustände, bei denen potentielle Unfallfolgen möglich sind, werden als Gefahrenzustände bezeichnet. Bei einer Sicherheitsanalyse werden die Fehler an der Nutzungsschnittstelle als potentielle Gefahrenzustände betrachtet. Die Bewertung der Wirkung der Gefahrenzustände auf die Umwelt übernimmt hierbei die Prozessanalyse (Anwendung des Produktes in der Umwelt) und die Untersuchung der Ursachen dafür die Produktanalyse (Fehler des Produktes) (siehe Abbildung 2.2).

Definition 2.1.14 (Gefahrenzustand)
Ein *Gefahrenzustand* ist ein Zustand eines Systems, der unter bestimmten, in der Umwelt erfüllten, Bedingungen zu einem gefährlichen Vorfall führt.[10] ⌟

Zustands-bewertung

Den Umweltbedingungen, die gültig sein müssen, damit ein Gefahrenzustand einen gefährlichen Vorfall zur Folge hat, kann eine Wahrscheinlichkeit zugeordnet werden. Betrachtet man für alle potentiellen Vorfälle deren Wahrscheinlichkeit und deren Ausmaß des Schadens, so bekommt man ein Maß, wie kritisch ein Gefahrenzustand ist.

Definition 2.1.15 (Gefahrenschwere)
Die *Gefahrenschwere* ist die Wahrscheinlichkeit, mit der ein Gefahrenzustand zu einem gefährlichen Vorfall führt, kombiniert mit dem Ausmaß des Schadens. (vgl. [Lev95], S. 179) ⌟

Risiko

Mit der Gefahrenschwere ist eine Abschätzung für die Folgen eines Gefahrenzustands gegeben. Ein System befindet sich jedoch nur für eine bestimmte Zeit in einem Gefahrenzustand. Um die Gefahrenzustände bewerten zu können, ist es zusätzlich nötig, die Wahrscheinlichkeit zu ermitteln, mit der sich das System in dem Gefahrenzustand befindet. Dies kann über eine Ermittlung der Entstehungswahrscheinlichkeiten der Ursachen bis hin zu den gegebenen primären Fehlen geschehen.

[10] [DIN04]: Umstand, durch den eine Person einer Gefährdung ausgesetzt ist.

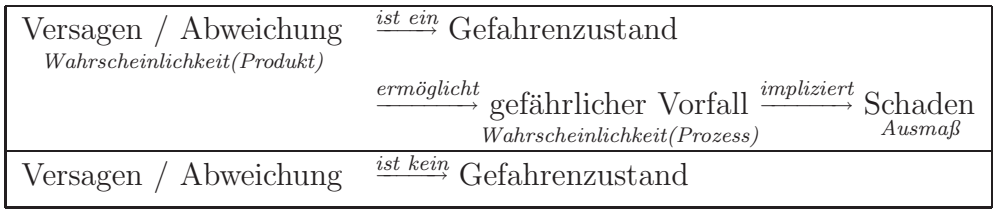


Abb. 2.2: Bewertung der Fehlerwirkung

Definition 2.1.16 (Risiko)
Das *Risiko* ist die Wahrscheinlichkeit, mit der ein Gefahrenzustand gültig ist, kombiniert mit der zugehörigen Gefahrenschwere.[11] ⌟

Funktionssicherheit

Ist das Risiko bei einer Funktion Null, so ist diese funktionssicher. Der Begriff der Funktionssicherheit (s.u.) wird allerdings meist nur zu einem gewissen Grad erreicht. Dies liegt insbesondere daran, dass sich eine Produktentwicklung immer in einem Kompromiss des Dreiecks Kosten - Zeit - Qualität ([Kor05]) bewegt. Es soll ein vertretbar sicheres System erstellt werden. Entsprechend wird der Begriff Funktionssicherheit nicht absolut, sondern als das Erreichen eines vertretbaren Grads der Sicherheit gesehen (siehe [MK05], S. 286). Hieraus ergibt sich folgende Definition:

Definition 2.1.17 (Funktionssicherheit)
Die *Funktionssicherheit* ist der Zustand eines Systems, frei von unvertretbaren Risiken zu sein.[12]
Idealvorstellung: Das System ist frei von Gefahrenzuständen, die vom Betrieb des Systems ausgehen. (vgl. [Lev95], S. 181) ⌟

Zusammenfassend ergibt sich für die Sicherheitsanalyse das Ziel, zu bestimmen, in wie fern die Funktionssicherheit eines Produktes erfüllt ist. Dazu werden zum einen die Gefahrenzustände hinsichtlich ihrer Wirkung, und zum anderen hinsichtlich ihrer Wahrscheinlichkeit bewertet.

[11] [DIN04]: Kombination aus der Wahrscheinlichkeit, mit der ein Schaden auftritt, und dem Ausmaß dieses Schadens.

[12] [DIN04]: *vertretbares Risiko*: Risiko, das basierend auf den aktuellen gesellschaftlichen Wertvorstellungen in einem gegebenen Zusammenhang tragbar ist. — *Sicherheit*: Freiheit von unvertretbaren Risiken.

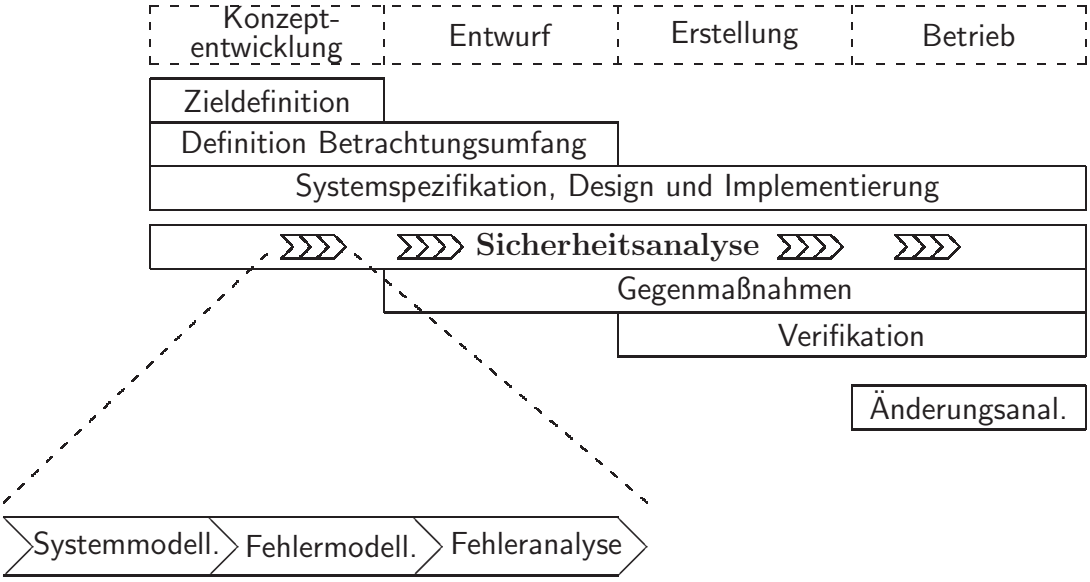


Abb. 2.3: Sicherheitsanalyse im Lebenszyklus (vgl. [Lev95], S. 293)

2.2 Sicherheitsanalyse in der Systementwicklung

Schnittstellen

Ergänzend zu der Definition des Begriffs der Sicherheitsanalyse beschäftigt sich dieser Abschnitt mit den Aktivitäten, die mit ihr in Zusammenhang stehen (vgl. [Lev95], S. 293). Neben den Aktivitäten der Sicherheitsanalyse wird auf die Aktivitäten, deren Ergebnisse in die Sicherheitsanalyse einfließen, bzw. zu denen die Sicherheitsanalyse etwas beiträgt, eingegangen. Schwerpunkt ist hierbei auch die Vorstellung von potentiellen Schnittstellen und Synergien zu Aktivitäten der Systementwicklung.

iterativer Zyklus

Abbildung 2.3 skizziert die Einordnung der Sicherheitsanalyse im Prozess. Die Aktivitäten der Sicherheitsanalyse werden iterativ zu projektspezifischen Meilensteinen der Entwicklung ausgeführt. Die gemeinsamen Schritte jeder Methode sind hierbei die *Systemmodellierung*, die *Fehlermodellierung* und die *Fehleranalyse* (siehe Abbildung 2.3 links unten). Unterschiede bestehen im Grad der Detaillierung, und der Intensität mit der die Aktivitäten unterstützt werden und in den Phasen des Entwicklungsprozesses, in denen sie einsetzbar sind. Bei der Systemmodellierung wird das System für die anzuwendende Methode geeignet dargestellt. Bei der Fehlermodellierung werden potentielle Fehler, die in einem System entstehen können, und Daten zu diesen Fehlern gesammelt, die jeweils in der Methode betrachtet werden sollen (z.B.: historische Daten, relevante Standards, praktische Erfahrungen, wissenschaftliche Untersuchungen und experimentelle Ergebnisse). Bei der Fehleranalyse werden die Fehler qualitativ und quantitativ analysiert. Eine qualitative Analyse ist eine Betrachtung der kausalen Zusammenhänge (z.B. Ursache-Wirkungs-Ketten). Eine quantitative Analyse ist eine Betrachtung der Fehler mit Metriken (z.B. Wahrscheinlichkeiten).

Entwicklungsphasen

Die Teilaufgaben der Sicherheitsanalyse sowie die mit der Sicherheitsanalyse verbundenen Aktivitäten finden in verschiedenen Phasen der Entwicklung eines Produktes statt. Abbildung 2.3 skizziert in der oberen Leiste eine Möglichkeit, ein Projekt in Phasen zu untergliedern. Die Phasen müssen nicht immer sequentiell nacheinander abgearbeitet werden, sondern können auch fließend ineinander übergehen.[13] Anhand dieser Phasen kann einsortiert werden, wann die Sicherheitsanalyse mit anderen Aktivitäten in Berührung kommt. Die erste Phase ist die *Konzeptentwicklung*, in der allgemein die Ziele, die das Produkt erfüllen soll, festgelegt werden, die technische Machbarkeit geprüft wird und ein grober Plan zur Realisierung erstellt wird. Es folgt der *Entwurf*, in dem das System spezifiziert wird. Es wird eine Architektur erstellt und geprüft ob das Produkt so erstellt werden kann, dass es der Spezifikation genügt. Als drittes folgt die *Erstellung* des Produktes. Zu den Aufgaben gehören die Erstellung der endgültigen Architektur, die Implementierung der Komponenten sowie das Überprüfen der Korrektheit. Letztlich folgt der *Betrieb*, in dem das Produkt angewendet wird. Ein wesentlicher Zeitpunkt ist hierbei der Übergang von der Erstellung zum Betrieb, da ab diesem Zeitpunkt an dem Produkt kaum etwas geändert werden kann und die Sicherheitsanalyse eine Voraussetzung für die Betriebszulassung ist.

Methoden der Sicherheitsanalyse

In den frühen Phasen ist die Spezifikation noch sehr grob und unterliegt häufigen Änderungen. Entsprechend eignen sich Methoden, die schnell durchführbar sind und keine detaillierte Spezifikation benötigen. Da die Fehleranalyse tendenziell eine detailliertere Spezifikation benötigt, legen die Methoden für die frühen Phasen mehr Gewicht auf die vorhandenen Daten zur Systemmodellierung und die Fehlermodellierung. Geeignete Methoden sind hier die z.B. die Preliminary Hazard List, die Preliminary Hazard Analysis ([Eri05], S.55, S.73) sowie die Gefahren- und Risikoanalyse (IEC61508 [DIN04]). Zu späteren Phasen sind die Spezifikation und die Architektur eher stabil und präzise. Entsprechend können hier detailliertere Methoden, die zwar einen hohen Zeitaufwand erfordern, dafür aber genauere Ergebnisse erzielen, angewendet werden. Zu diesen Methoden gehören die Fehler-Möglichkeit- und Einfluss-Analyse (FMEA) (siehe Abschnitt 2.3) und die Fehlerbaumanalyse (FTA) (siehe Abschnitt 2.4). Diese Methoden legen den Schwerpunkt auf die Zusammenhangsanalyse, um die Funktionssicherheit des Produktes detaillierter zu bestimmen. Sie werden sowohl zur Rückkopplung in die Entwicklung, wie auch zur Zertifizierung eines Produktes genutzt. Im Folgenden werden die in Abbildung 2.3 gegebenen Aktivitäten hinsichtlich einer Sicherheitsanalyse mittels einer FMEA und FTA betrachtet.

Ziele und Umfang

Die Aktivität der *Definition der Zielsetzung und des Betrachtungsumfangs* (erster und zweiter Balken in Abbildung 2.3) findet in der Konzeptentwicklung und im Entwurf statt. Die folgenden Rahmenbedingungen, welche teilweise auch aus Sicherheitsanalysen früher Phasen resultieren, werden für die (weitere) Sicherheitsanalyse festgesetzt ([MK05], S. 283):

[13] Abhängig von der Domäne gibt es verschiedene detailliertere Ausprägungen der Phasen (siehe [BRS00], [EM07]). Diese lassen sich auf die in dieser Arbeit skizzierten Phasen abbilden.

- Zweck, der mit der Sicherheitsanalyse erfüllt werden soll (z.B.: TÜV-Abnahme und Zertifizierung, Feedback in die Entwicklung, Nachweis über Stand der Technik).

- Höhe des Sicherheits-Levels anhand des geplanten Betriebs des Produkts (z.B. ASIL-D[14])

- Umfang des Betrachtungszeitraum (Betrieb, Gesamtzeit) und des zu betrachtenden Aspektes (Produkt, Prozess, Herstellung)

- Bestimmung der zu betrachtenden Schäden (Personenschaden, Kosten, Image, ...)

- Zu verwendende Dokumente der Entwicklung mit den zur Verfügung stehenden Informationen (Funktions-Modelle, Lastenhefte, Test-Dokumente, ...) und Richtlinien für diese Dokumente. Die Richtlinien sollten bereits hier so gestaltet sein, dass sie sowohl für die Sicherheitsanalyse, wie auch für die Entwicklung entsprechend geeignet sind. Ein Beispiel für eine Richtlinie ist die Vorgabe der Verwendung einer Modell-Bibliothek, mit der auch effizient Fehler modelliert werden können. Eine solche Beschreibung ist in Kapitel 3 zu finden.

Maßnahmen, Verifikation, Feedback

Aus der Sicherheitsanalyse heraus ergeben sich zum einen identifizierte Gefahren, für die *Gegenmaßnahmen* ergriffen werden sollen und zum anderen Anforderungen an die *Verifikation*, bei denen überprüft wird, ob Fehler präsent sind. Umgekehrt gilt, dass die tatsächlich implementierten Maßnahmen und Schritte wiederum eine Wirkung auf die Wahrscheinlichkeit der Fehler haben, und sich dadurch die Ergebnisse der Sicherheitsanalyse ändern. So kann z.B. die Einführung einer redundanten Architektur dazu führen, dass eine Gefahr wesentlich unwahrscheinlicher wird. Für einige Sicherheitsanalysen wird deshalb auch vorgeschlagen[15], die Sicherheitsanalysen einmal mit und einmal ohne Berücksichtigung der Gegenmaßnahmen durchzuführen, um deren Wirkung zu bewerten.

Änderungsanalyse

Die Resultate der Sicherheitsanalyse fließen in die *Änderungsanalyse* ein, in der die nicht beseitigten Gefahren und Maßnahmen so weit quantifiziert werden, um über die Durchführung von Änderungen entscheiden zu können. Es werden Wahrscheinlichkeiten, die wirtschaftlichen Auswirkungen, die potentiellen Schäden, die Kosten für Vorbeugungs- oder Korrekturmaßnahmen und das verbleibende Risiko ermittelt und abgewägt.

Spezifikation

Die meisten Informationen für eine Sicherheitsanalyse werden bei der *Systemspezifikation*, dem *Design* und der *Implementierung* erstellt. Hier wird das System beschrieben, indem die Systemgrenze festgelegt wird, die Eigenschaften und Funktionen, die es zur Verfügung stellt, definiert werden, sowie die Architektur und Implementierung erstellt wird. Um diese Informationen in der Sicherheitsanalyse ohne

[14] Automotive Safety Integrity Level D
[15] Vorschlag von Verband der Automobilindustrie für FMEA

großen Mehraufwand wiederverwenden zu können, ist es notwendig, die Modelle sowohl für die Entwicklung wie auch für die Sicherheitsanalyse nutzbar zu machen. In dieser Arbeit spiegelt sich das hohe Gewicht der Systemspezifikation für die Sicherheitsanalyse in Kapitel 3 wider, in dem die Modelle für die Systemdefinition beschrieben werden, um später auf diesen die zusätzlichen Modelle für die Fehleranalyse aufzubauen.

Die beiden Methoden FMEA und FTA bieten aufgrund des hohen Aufwandes zu ihrer Erstellung und der detaillierten Systemdefinition als Grundlage ein hohes Optimierungspotential durch Automatisierung. Sie werden in den folgenden Abschnitten näher beschrieben.

2.3 Fehler-Möglichkeit- und Einfluss-Analyse

Übersicht

Eine Methode der Sicherheitsanalyse, die in der Industrie eingesetzt wird, ist die *Fehler-Möglichkeit- und Einfluss-Analyse (FMEA)* (siehe auch [MK05]). Dieser Abschnitt skizziert zuerst die Ziele und das Ergebnis dieser Methode. Danach folgt die Beschreibung der Schritte, mit denen die Ergebnisse erarbeitet werden.

Ziele

Das Kernziel der FMEA ist die Bewertung der Funktionssicherheit eines Systems anhand der Betrachtung der Einfachfehler, die in dem System entstehen können. Zusätzlich werden mit der FMEA eine Menge von Neben- und Teilzielen verfolgt, die sich aus dem Vorgehen der FMEA ergeben (siehe [Kre06], S. 53; [MK05], S. 300):

a) Strukturierung und Beschreibung des betrachteten Systems (Modellbildung)

b) Identifikation der potentiellen Einfachfehler des Systems

c) Erstellung von Anregungen zur Verbesserung des Systems (Vermeidungsmaßnahmen)

d) Ableitung von Anforderungen zur Fehlererkennung in der Entwicklungsphase

e) Ableitung von Anforderungen zur Fehlererkennung zur Laufzeit (Kundenbetrieb)

f) Unterstützung der Spezifikation (Lastenhefterstellung, Pflichtenhefterstellung)

Diese zusätzlichen Ziele ergeben sich aus den Schritten, die zur Durchführung einer FMEA gehören. Der Beitrag der Schritte zu diesen Zielen und die Übereinstimmung mit den Zielen der Systementwicklung sind bei der Erklärung der Schritte angegeben.

Ergebnis-Dokument

Das Ergebnis einer FMEA kann in Tabellen festgehalten werden, welche die im Schema der Tabelle 2.2 gegebenen Spalten enthalten (siehe [PCB+55]; [Eri05], S. 247). Es sind die Funktionen und Eigenschaften der Komponenten des Systems und die dazu gehörenden potentiellen Fehler notiert (siehe Spalte *Funktion* und *Fehler* der Tabelle 2.2). Zu jedem Fehler ist eine Wirkung an der Systemgrenze (Folgefehler) und

Funktion	*Fehler*	*Wirkung*	*B*	*Ursachen*	*Vermeidungs-Maßnahme*	*A*	*Entdeckungs-Maßnahme*	*E*
Erkennung: Erkennen untersteuern	Situation nicht erkannt	kein Gegensteuern	6	Sensorauswertung zu ungenau	feinere Sensorauswertung	5	Test im Fahrversuch (Testspec. Nr. 0815)	5
	Situation fälschlich erkannt	falscher Eingriff	9	Fehler in Software	Monitoring mit Abschaltung	2	SW-Test (Testspec. Nr. 0911)	3
				Sensor versagt	Kopplung mit ESP	2	Signalplausibilisierung	2
Regelung: Berechnen des Zusatzlenkwinkels	zu kleiner Winkel	zu schwacher Eingriff	3	ungenauer Sensor	redundante Sensoren	6	Test im Fahrversuch (Testspec. Nr. 4711)	6

Tabelle 2.2: Beispiel für eine FMEA-Tabelle

eine Bewertung der dazu gehörenden Gefahrenschwere angegeben. Den potentiellen Fehlern ist eine Menge möglicher Ursachen zugeordnet, die jeweils die Einfachfehler auslösen können. Jedem Fehler ist eine Menge möglicher Maßnahmen zu dessen Vermeidung und Entdeckung zugeordnet, sowie Kennzahlen zum verbleibenden Risiko nach Anwendung der Maßnahmen. Eine detaillierte Beschreibung der Einträge der Spalten erfolgt im Rahmen der Betrachtung der FMEA-Aktivitäten.

Schritte

Die hauptsächlichen, systematisch und vollständig auszuführenden Aufgaben einer FMEA sind das Finden der potentiellen Fehler und die Bestimmung ihrer Wirkung. Eine Möglichkeit, diese Aufgabe zu erfüllen, ist ein aus 7 Teilaufgaben bestehendes Vorgehen (siehe [MK05], S. 300), welches auch in den Fallstudien angewendet wurde. Diese Schritte sind:

(I) Systemmodellierung

  (a) Erstellen des Systemstrukturbaums

  (b) Zuordnen von Anforderungen zu den Komponenten

  (c) Vernetzung der Anforderungen entlang des Systemstrukturbaums

(II) Zuordnung von Fehlern zu den Anforderungen (Fehlermodellierung)

(III) Fehleranalyse

  (a) Vernetzen der Fehler entlang des Anforderungsnetzes

  (b) Zuordnen von Kennzahlen zu den primären Fehlern

  (c) Angabe von Maßnahmen zur Reduktion des Risikos

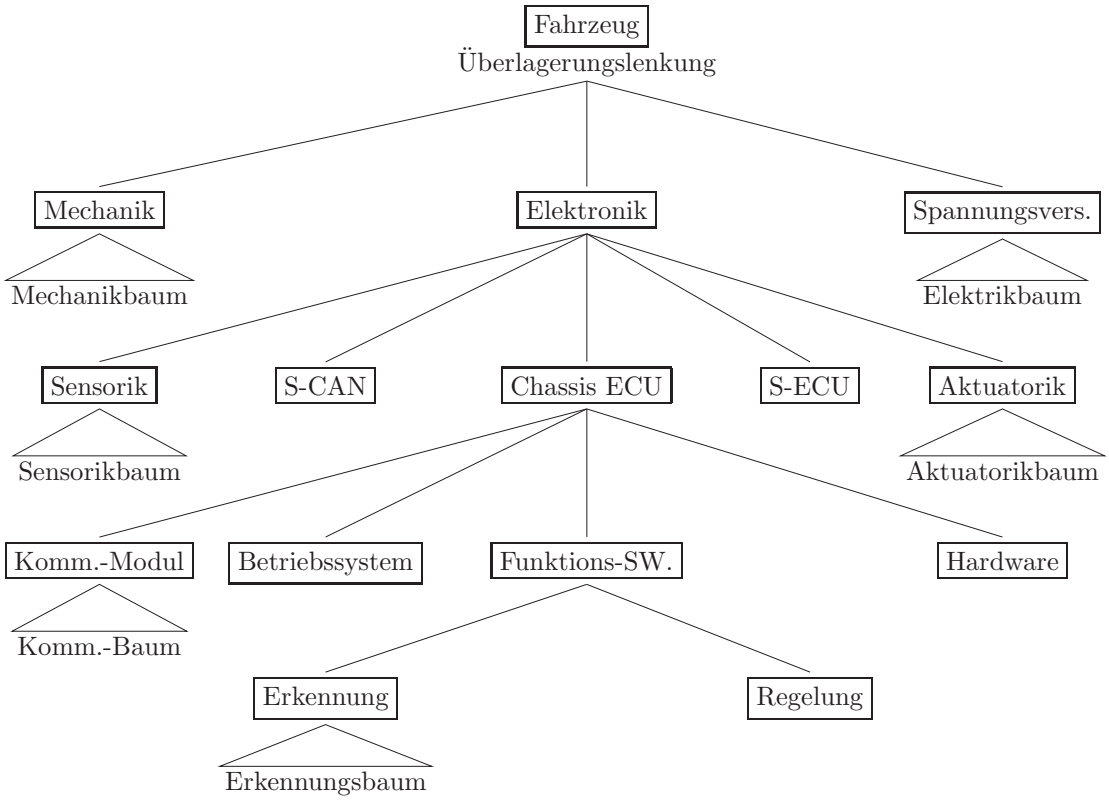


Abb. 2.4: Beispiel eines Systemstrukturbaums

Elemente

(I a) Erstellen des Systemstrukturbaums Die Komponentenstruktur des Systems wird in einem Baum dargestellt. Hierbei ist die Wurzel des Baumes das zu betrachtende System, die Knoten darunter dessen Teilkomponenten und schließlich die Blätter des Baumes atomare Komponenten, die für die Betrachtung nicht weiter zerlegt werden. Die Entscheidung, ob und wie eine Komponente weiter zerteilt wird, gibt die Architekturbeschreibung des Systems vor. Die Komponenten können sowohl logische Komponenten, wie auch physikalische Komponenten sein. Das System in Abbildung 2.4 wird zum Beispiel zuerst nach physikalischen Komponenten zerlegt, dann aber bei den Steuergeräten (ECUs) ein immaterieller Anteil Applikationslogik (Funktions-Software) abgespalten. Von diesem Knoten ab entspricht der Baum der logischen Architektur, die durch die Applikationslogik gegeben ist.

Definition 2.3.1 (Systemstrukturbaum)
Die Menge der (hierarchischen) Komponenten-Identifikatoren sei $\texttt{COMP}$.
Ein *Systemstrukturbaum* ist eine Relation $\texttt{TREE\_COMP} \subseteq \texttt{COMP} \times \texttt{COMP}$ mit der Baumeigenschaft:

$$\#\texttt{TREE\_COMP} = \#\texttt{COMP} - 1 \text{ und } \forall\{(r_1, r_2), (s_1, s_2)\} \subseteq \texttt{TREE\_COMP} : r_2 \neq s_2 \qquad \lrcorner$$

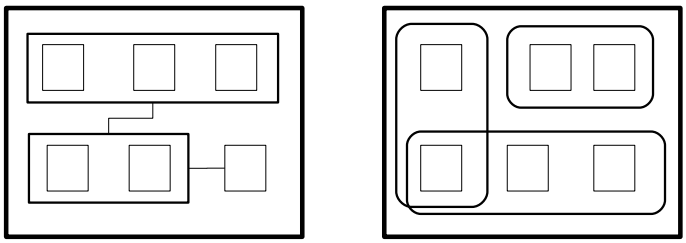


Abb. 2.5: Komposition (links) und Kombination (rechts) eines Systems

Systemgrenze

Die in dieser Arbeit gegebenen Methoden der Sicherheitsanalyse orientieren sich an dieser Komponentenstruktur eines Systems. Das Fahrzeug wird in Abbildung 2.4 in Mechanik, Elektronik und Elektrik unterteilt und diese dann entsprechend der Architektur weiter bis zu den Blättern des Baums zerlegt. Abbildung 2.5 skizziert auf der linken Seite diese Komposition schemenhaft, in dem sich das Gesamtsystem (großes Rechteck) aus Subsystemen (darin enthaltene Rechtecke) zusammensetzt. Das Gesamtsystem im Automobilbereich ist das Fahrzeug. Um nicht eine Sicherheitsanalyse des gesamten Fahrzeugs machen zu müssen, wird dessen Nutzungsschnittstelle so zerlegt, dass die Wirkung der Teile der Schnittstelle thematisch geschlossen beurteilt werden kann. Der in Abbildung 2.4 zu betrachtende Schnittstellenteil ist z.B. die Überlagerungslenkung. Ein Schnittstellenteil kann sich auf Teile von zusammengesetzten Komponenten beziehen und eine Aufteilung der Schnittstelle des Systems muss nicht zu disjunkten Mengen von zugehörigen Komponenten führen. Abbildung 2.5 skizziert dies auf der rechten Seite, in dem die zu den Schnittstellenteilen gehörenden Komponenten mit abgerundeten Kreisen gruppiert sind. Das Gesamtsystem erhält man entsprechend durch Kombination der zu den Schnittstellenteilen gehörenden Komponentenmengen. Eine Überführung der Sichten findet auf der Ebene atomarer Komponenten statt, in denen die Zergliederung zu übereinstimmenden Teilkomponenten führt. In Abbildung 2.4 wird z.B. die Überlagerungslenkung eines Systems betrachtet. Diese betrifft auch allgemeine Komponenten, wie z.B. die Spannungsversorgung, die auch für andere Schnittstellenteile, wie z.B. die Beleuchtung wichtig ist. Bei der Sicherheitsanalyse für einen Schnittstellenteil werden dann nur die Komponenten (-teile) betrachtet, die betroffen sind, und entsprechende Schnittstellen zu Sicherheitsanalysen anderer Teile definiert.

Synergien

Die hierarchische Struktur eines Systems wird auch in der Systemdefinition und Beschreibung festgehalten. Die dort gegebene Information kann, insofern sie formal gegeben ist, durch Extraktion der jeweiligen Hierarchiemodelle automatisch in einen Systemstrukturbaum transformiert werden (Ein Beispiel ist in Anhang A).

Anforderungsarten

(I b) Zuordnen von Anforderungen zu den Komponenten Den Komponenten werden (auf allen Ebenen des Systemstrukturbaums) Anforderungen zugeordnet, die sie erfüllen müssen. Hierbei kann es sich sowohl um Verhaltensbedingungen wie auch um Qualitätseigenschaften (siehe [ISO01]) handeln. Die Definition dieser Anforderungen findet üblicherweise bei der Spezifikation statt und kann aus den

dortigen Dokumenten übernommen werden (siehe Anhang A). Die Anforderungen lassen sich in drei Arten einteilen:

- Angabe der *Verhaltensbedingungen*: Hier werden Anforderungen an die Ausgaben in Abhängigkeit der Eingaben des Systems gestellt, die zur Laufzeit erfüllt sein müssen.
- Beschreibungen der *Ausgabenbedeutung* des Systems: Sie beschreiben informell die Bedeutung bzw. Berechnung der Ausgaben des Systems und sind so eine Sonderform der Verhaltensbedingungen
- Angabe der *Qualitätsanforderungen*: Zu den Komponenten gibt es eine Menge von Anforderungen, die sich nicht direkt im Verhalten des Systems beobachten lassen, bzw. gar kein Verhalten haben. Diese Anforderungen betrachten unter anderem die Aspekte Änderbarkeit, Übertragbarkeit und Benutzbarkeit.

Definition 2.3.2 (Anforderungszuweisung)
Die Menge der Komponenten-Identifikatoren sei $\texttt{COMP}$.

Die Menge der Verhaltensbedingungen sei: $\texttt{REQ}^{\texttt{constraint}}$
Die Menge der Ausgabenbedeutungen sei: $\texttt{REQ}^{\texttt{output}}$
Die Menge der Qualitätsanforderungen sei: $\texttt{REQ}^{\texttt{quality}}$

Die Menge der Anforderungen sei $\texttt{REQ} = \texttt{REQ}^{\texttt{output}} \dot{\cup} \texttt{REQ}^{\texttt{constraint}} \dot{\cup} \texttt{REQ}^{\texttt{quality}}$.

Die *Anforderungszuweisung* ist eine Relation $\texttt{COMP\_REQ} \subseteq \texttt{COMP} \times \texttt{REQ}$. ⌟

Verhaltensmuster

Es gibt Beschreibungen der Ausgaben, die in verschiedene Verhaltensmuster aufgeteilt werden können. Ein Verhaltensmuster ist dann unter entsprechenden Umweltbedingungen gültig. Hierbei ist darauf zu achten, dass die Aufteilung der Umweltbedingungen vollständig ist. Abbildung 2.6 zeigt Beispielanforderungen für eine Komponente „Funktions-Software“. In diesen wird die Beschreibung der Ausgabe des Stellwinkels in 4 verschiedene Verhaltensmuster entsprechend der Bedingungen Übersteuern, Untersteuern, μ-Split (MS) und Stabil aufgeteilt. Eine formale Definition eines Verhaltensmusters befindet sich in Abschnitt 3.3.4.

(I c) Vernetzung der Anforderungen entlang des Systemstrukturbaums

Rahmen für Folgefehler

Die Abhängigkeiten zwischen den Anforderungen werden angegeben, um einen Rahmen für die Analyse der Ursache-Wirkungs-Beziehungen zwischen Fehlern zu bekommen. Ein einer Anforderung zugeordneter Fehler kann nur dann einen einer anderen Anforderung zugeordneten Fehler als Folge haben, wenn eine Abhängigkeit zwischen den Anforderungen besteht. Die Menge der Abhängigkeiten zwischen den Anforderungen heißt *Anforderungsnetz* bzw. bei Ausgabenbedeutungen *Funktionsnetz*.

Ausrichtung am Baum

Um die zu modellierenden Abhängigkeitsbeziehungen auf eine handhabbare Menge zu reduzieren, werden nur Zusammenhänge entlang der Äste des Systemstrukturbaums betrachtet. Es wird also die Wirkung auf die jeweils umfassendere Komponente modelliert. Abbildung 2.7 stellt die Schritte zur Ermittlung des zu modellierenden

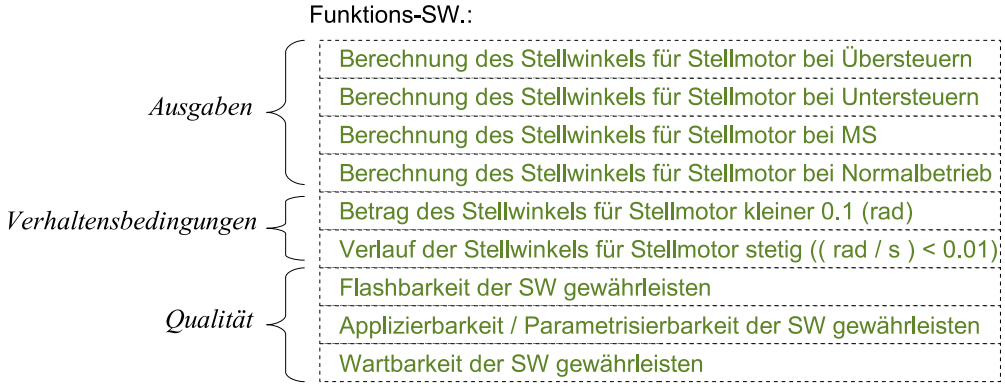


Abb. 2.6: Beispiel einer Anforderungszuweisung zur Komponente Funktions-SW.

Funktionsnetzes anhand eines Beispiels logischer Komponenten dar. Ausgehend von der Systemarchitektur wird in Schritt (I) die Relation ermittelt, die angibt, wie die verschiedenen Ausgaben (in diesem Falle mathematische Funktionen) unmittelbar voneinander abhängen. In Schritt (II) wird die transitive Hülle der Relation gebildet, in der explizit alle Abhängigkeiten angegeben sind. In Schritt (III) werden von der transitiven Hülle der Abhängigkeiten nur diese dargestellt, die von der aktuellen Ebene des Systemstrukturbaums zu der darüber liegenden Ebene reichen. Damit sind die Zusammenhänge auf gleicher Ebene abstrahiert und nur die Abhängigkeiten zur umfassenden Komponente modelliert. Der Vorteil des so modellierten Funktionsnetzes ist, dass es kreisfrei ist (Abhängigkeit zwischen F_2 und G_2 ist abstrahiert), und die Menge der zu modellierenden Abhängigkeiten gesunken ist (Reduktion von 9 auf 6 Elemente). Dieses Anforderungsnetz entlang des Systemstrukturbaums ist wie folgend definiert:

Definition 2.3.3 (Anforderungsnetz)
Das *Anforderungsnetz* ist eine Relation $\texttt{REQ\_REQ} \subseteq \texttt{COMP\_REQ} \times \texttt{COMP\_REQ}$
bei der gilt: $\forall((c_1, r_1), (c_2, r_2)) \in \texttt{REQ\_REQ} : (c_1, c_2) \in \texttt{TREE\_COMP}$. ⌟

Anforderungsarten

Die Abhängigkeiten der Anforderungen werden zumindest für alle drei Arten (Ausgaben, Verhaltensbedingungen, Qualitätsanforderungen) innerhalb der jeweiligen Art angegeben. Eine Vernetzung zwischen den Anforderungsarten ist prinzipiell möglich, der Nutzen sollte aber hier spezifisch geprüft werden. Abbildung 2.8 zeigt ein Beispiel eines Anforderungsnetzes.

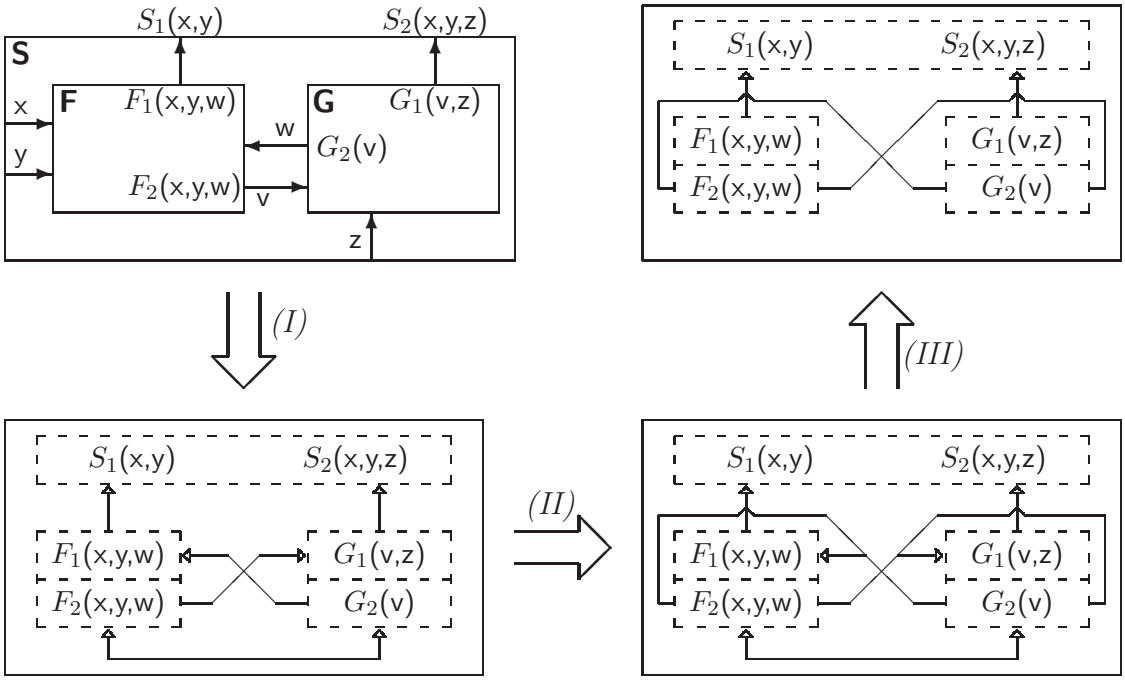


Abb. 2.7: Angabe des Funktionsnetzes

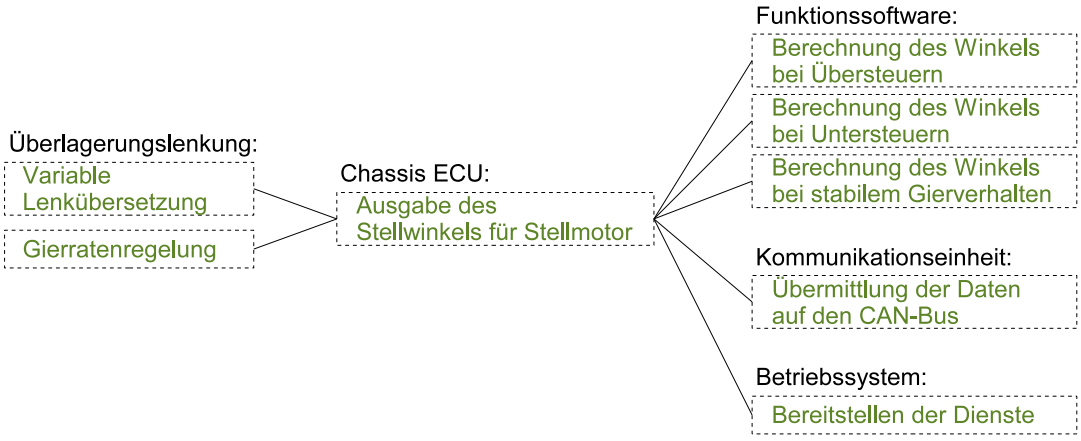


Abb. 2.8: Beispiel eines Anforderungsnetzes

(II) Zuordnung von Fehlern zu den Anforderungen Den Anforderungen der Komponenten werden Fehler zugeordnet. Je nach Art der Anforderungen können entsprechende Fehler angegeben werden, die sich teilweise systematisch aus der Beschreibung ableiten lassen. *Ableitung von Anforderungen*

Definition 2.3.4 (Fehlerzuweisung)
Die Menge der Fehler sei `FAIL`.
Die *Fehlerzuweisung* ist eine Relation $\texttt{REQ\_FAIL} \subseteq \texttt{COMP\_REQ} \times \texttt{FAIL}$. ⌟

- *Ausgaben*: Bei dieser Anforderungsart ist ein Fehler eine Abweichung von der Soll-Ausgabe. Diese Abweichung kann je nach gewünschter Präzision von einer allgemeinen Fehlerbeschreibung (z.B.: „falsche Ausgabe“) bis hin zur Beschreibung eines spezifischen Fehlverhaltens reichen (z.B.: „Wert true statt false nach dritter Anfrage“). Vorschläge zu einer geeigneten Auswahl der Fehlerbeschreibungen in Abhängigkeit der Funktions-Charakteristika (datenflussorientiert, zustandsbehaftet, algorithmisch, ...) und des Datentyps der Ausgabe sind in Abschnitt 4.2 zu finden. Abbildung 2.9 zeigt ein Beispiel für Ausgabefehler zu der Funktion „Gierratenregelung“, welche datenflussorientiert ist und einen reellen Wert liefert.

- *Verhaltensbedingungen*: Bei Verhaltensbedingungen sind Fehler die Negation der Bedingung. In Fällen, in denen die Verhaltensbedingung semantisch eindeutig in Abhängigkeit von booleschen Variablen definiert ist, ist die Negation als einzige Angabe einer Abweichung ausreichend. Bei Anforderungen, die nicht eindeutig definiert sind, oder die sich auf Variablen beziehen, die eine Metrik darstellen, ist es zusätzlich möglich, die Abweichung stufenweise darzustellen (siehe Abbildung 2.9 unten).

- *Qualitätsanforderungen*: Analog zu den Verhaltensbedingungen zeichnen sich die Abweichungen von den Qualitätsanforderungen ebenfalls als Negation der Soll-Eigenschaft aus. Auch hier kann bei Bedarf eine Abstufung der Abweichungen angegeben werden, die aber meist nicht objektiv anhand einer Metrik gemessen werden kann, sondern anhand von Fallstudien ermittelt wird. (z.B.: „komfortable Fahrt gewährleisten“, „Komfort reduziert“, „keine komfortable Fahrt“).

Unabhängig von den Anforderungen können die Fehler in drei verschiedene Kategorien eingeteilt werden: (a) primäre Fehler, (b) Folgefehler und (c) ungeprüfte Folgefehler[16]. Beschreibt ein Knoten im Systemstrukturbaum eine implementierte Komponente, so kann es dazu kommen, dass diese Komponente Implementierungs- oder Laufzeitfehler aufweist. Diese Fehler werden als primäre Fehler bezeichnet, da sie an dieser Stelle im Systemstrukturbaum mit einer Wahrscheinlichkeit unabhängig *Fehler-Einstufung*

[16]Es wurde bewusst nicht der Begriff „potentieller Folgefehler“ gewählt, da alle in einer FMEA vorkommenden Fehler potentiell sind.

Abb. 2.9: Beispiel einer Fehlerzuweisung

von anderen Fehlern entstehen können. Neben den primären Fehlern können einer Funktion einer Komponente auch Folgefehler zugeordnet werden, also Fehlererscheinungen, die ihren Ursprung in einer inneren Komponente haben. Da bei einer zusammengesetzten Komponente das Verhalten nach außen analysiert und bewertet werden soll, ist es oft sinnvoll, eine Menge ungeprüfter Folgefehler anzugeben, bei denen man erst nach der Fehlervernetzung erkennt, ob diese Fehler tatsächlich Folgefehler der identifizierten primären Fehler sind. Mit der Angabe dieser ungeprüften Folgefehler kann eine Analyse auf Folgefehler stattfinden, bei welcher die Entwickler explizit Folgefehler ausschließen müssen und somit die Wahrscheinlichkeit einer übersehenen Wirkung eines Fehlers sinkt.

Beispiel 2.3.1 (ungeprüfte Folgefehler)
Für die Funktion „Lenkwinkel berechnen“ gibt es unter anderen die ungeprüften Folgefehler „zu großer Betrag“, „zu kleiner Betrag“, „zu großer Wert“ und „zu kleiner Wert“. Bei einer Untersuchung von Fehlern innerer Komponenten hin auf diese ungeprüften Folgefehler zeigt sich in der Fallstudie, dass die ersten Fehler geeignete Folgefehler sind und die letzten Fehler nicht zur Beschreibung der Folge der identifizierten primären Fehler geeignet sind. Trotzdem wurden die letzten Fehler untersucht und erst dann zur Beschreibung der Folge ausgeschlossen. ⌟

Netzstruktur

(III a) Vernetzen der Fehler Die Fehler der Komponenten werden mit denen derer Teilkomponenten vernetzt. Die Vernetzung verläuft wie bei der Vernetzung der Anforderungen in Schritt III. Auch hier ist es aus pragmatischer Sicht wichtig, die Fehlerfolgen auf gleicher Ebene, die z.B. auch Zyklen enthalten können, zu abstrahieren und nur den Teil der transitiven Hülle (wie in Abbildung 2.7 gezeigt) zu betrachten, der entlang des Anforderungsnetzes geht. Es entsteht so ein gerichteter Graph, der oft auch als *Fehlernetz* bezeichnet wird.

Definition 2.3.5 (Fehlernetz für FMEA)
Das *Fehlernetz für FMEA* ist eine Relation $\texttt{FAIL\_FAIL} \subseteq \texttt{REQ\_FAIL} \times \texttt{REQ\_FAIL}$
bei der die Ausrichtung am Anforderungsnetz gilt:
$\forall((cr_1, f_1), (cr_2, f_2)) \in \texttt{FAIL\_FAIL} : (cr_1, cr_2) \in \texttt{REQ\_REQ}$. ⌟

Abhängigkeit

Ein Fehler ist genau dann von einem anderen Fehler abhängig, wenn er durch Entstehung des anderen Fehlers selbst als Folge entstehen kann. Eine Formale Beschreibung des Abhängigkeit-Begriffs ist in Kapitel 5 zu finden. Mit der Vernetzung der Fehler zur jeweils darüber liegenden Ebene wird so inkrementell die Wirkung der primären Fehler bis hin zur Systemgrenze modelliert.

Kennzahlen

(III b) Zuordnen von Kennzahlen zu den primären Fehlern Die primären Fehler werden mit Kennzahlen versehen, welche die Stärke des Handlungsbedarfs anzeigen. Die Kennzahlen werden aus dem Expertenwissen der Entwickler und Analysten heraus gebildet und sind somit ein Abbild der pragmatischen Schätzungen. Statistisch steigt die Qualität dieser Kennzahlen mit der Menge der beteiligten Experten.

Skalen

Für die jeweiligen Kennzahlen werden Werte auf Skalen von 1 bis 10 vergeben. Die Bedeutung der Werte auf den Skalen wird spezifisch für die jeweiligen Projekte in Bewertungskatalogen konkretisiert. So kann z.B. der Wert 5 bei der Entdeckungswahrscheinlichkeit bedeuten, dass Tests und ein Review durchgeführt wurden, der Wert 7 hingegen, dass nur eines von beidem durchgeführt wurde. Falls über einen Wert keine Aussage gemacht werden kann, so wird der Maximalwert 10 gewählt. Damit wird dem Grundsatz entsprochen, im Zweifelsfall immer den schlimmsten Fall anzunehmen. Im Folgenden werden die zu ermittelnden Kennzahlen erläutert:

- *A-Wert:* Dieser Wert ist eine Abschätzung der Auftrittswahrscheinlichkeit, mit der ein primärer Fehler innerhalb eines Zeitraumes oder bis zu einem Zeitpunkt hin auftreten[17] wird. Je größer der Wert ist, desto größer ist die Auftrittswahrscheinlichkeit. Der Wert kann aus Zuverlässigkeitsberechnungen und Erfahrungswerten ermittelt werden.
- *E-Wert:* Dieser Wert ist eine Abschätzung der Entdeckungswahrscheinlichkeit, mit der innerhalb eines Zeitraumes oder bis zu einem Zeitpunkt hin entdeckt wird, ob ein potentieller primärer Fehler tatsächlich vorhanden ist. Je kleiner der Wert ist, desto größer ist die Entdeckungswahrscheinlichkeit. Sowohl Erfahrungswerte, wie auch statistische Berechnungen können zur Erhebung der Angabe genutzt werden.
- *B-Wert:* Dieser Wert ist eine Abschätzung des Grads des Schadens ("Bedeutung") den ein Fehler potentiell zur Folge hat. Ein höherer Wert gibt einen größeren Grad an. Der B-Wert wird den Folgefehlern an der Systemgrenze zugeordnet.

[17] Der Begriff *auftreten* wird in dieser Arbeit als Synonym des Begriffs *entstehen* verwendet.

Risiko Diese drei Kennzahlen gehen in die Berechnung der Risikoprioritätszahl (RPZ) ein. Diese Zahl wird zu jedem primären Fehler angegeben und indiziert in Verbindung mit den A-, E-, B-Werten den Handlungsbedarf für Maßnahmen. Umgekehrt werden die Zahlen durch ergriffene Maßnahmen reduziert. Meist wird in den Bewertungskatalogen festgehalten, ab welchem Wert ein Fehler als kritisch einzustufen ist. Verschiedene Beispiele und Methoden zu den Skalen und der Berechnung der RPZ sind in [IK00] zu finden. Im Allgemeinen ist die RPZ das Produkt der Kennzahlen, also:

$$RPZ = A \cdot E \cdot B$$

Maßnahmen **(III c) Angabe von Maßnahmen zur Reduktion des Risikos** Anhand der Kennzahlen wird entschieden, bei welchen Fehlern in welchem Ausmaß Maßnahmen getroffen werden. Ziel dieser Maßnahmen ist es, den E-Wert und den A-Wert entsprechend zu senken. Wurden die Maßnahmen getroffen, so kann für die von der Maßnahme betroffenen Fehler eine erneute Ermittlung der Kennzahlen stattfinden. Am Ende der Erstellung eines Produktes sollten alle Fehler akzeptable Kennzahlen haben. Entsprechend der beiden zu senkenden Werte gibt es auch zwei Arten von Maßnahmen:

- *Vermeidungsmaßnahme:* Sie senkt die Auftrittswahrscheinlichkeit eines Fehlers. Dies kann zum Beispiel durch Designänderungen, Einsatz von bestimmten Richtlinien und Methoden bei der Entwicklung oder dem Einsatz von qualitativ hochwertigeren Bauteilen erreicht werden.
- *Entdeckungsmaßnahme:* Sie erhöht die Wahrscheinlichkeit zu erkennen, ob ein potentieller Fehler tatsächlich vorhanden ist. Die Entdeckung ist dabei so zeitig, dass entweder noch Maßnahmen getroffen werden können, die den Fehler verhindern oder zumindest die Wirkung des Fehlers auf ein akzeptables Maß gesenkt werden kann. Dies kann zum Beispiel durch Tests während der Entwicklung oder durch Monitoring (in Kombination mit Vermeidungsmaßnahmen) im Fahrzeug erreicht werden.

Mögliche Maßnahmen und deren Wirkung auf die Werte in Bezug auf die Kosten werden in dieser Arbeit nicht näher betrachtet. Informationen hierzu sind in [Wag07] zu finden.

2.4 Fehlerbaumanalyse

Ziel Die *Fehlerbaumanalyse* dient der Ermittlung und Bewertung der Ursachen eines Fehlers, wobei hier auch Mehrfachfehler als Ursachen betrachtet werden (siehe [Thu04]). Die betrachteten Ursachen eines Fehlers können also andere Fehler (als Einfachfehler) oder Kombinationen von Fehlern (als Mehrfachfehler) sein. Die FTA ist eine Ergänzung zur FMEA, in der Einfachfehler und deren Wirkung betrachtet werden.

Schritte

Die Gemeinsamkeiten und Unterschiede der FTA und der FMEA lassen sich anhand der Beschreibung des Vorgehens verdeutlichen. Eine FTA besteht aus vier Teilaufgaben ([Lev95], S. 317): (i) der Systemdefinition, (ii) dem Aufbau des Fehlerbaumes, (iii) der Ermittlung der minimalen Schnittmengen primärer Fehler und (iv) Auswertung der Wahrscheinlichkeiten. Die Systemdefinition ist identisch mit der Aufgabe (I) der FMEA. Unterschiede sind in den folgenden Aufgaben zu erkennen. Der Aufbau des Fehlerbaumes entspricht in etwa den Aufgaben (II) und (III a) der FMEA, allerdings werden hier auch Mehrfachfehler als Ursachen betrachtet. Schritt (iii) entspricht einer neuen Aufgabe, in der Schnittmengen primärer Fehler ermittelt werden, die als Ursache für einen bestimmten Fehler in Frage kommen. Die Auswertung der Wahrscheinlichkeiten findet hier mit tatsächlichen Wahrscheinlichkeiten statt, und nicht wie bei der FMEA mit relativen Werten. In den folgenden Absätzen wird auf die Aufgaben der FTA eingegangen.

Aufbau des Fehlerbaumes

Die Aufgabe des Aufbaus des Fehlerbaumes entspricht der Erstellung eines Fehlernetzes, welches die Abhängigkeiten zu den Fehlerursachen beschreibt. Bei der ursprünglichen Fehlerbaumanalyse wird dieses Netz für nur einen zu untersuchenden Folgefehler aufgebaut und ist ein Baum, der so genannte *Fehlerbaum* (siehe [Int90]). Um nicht für jeden zu untersuchenden Folgefehler einen eigenen Baum zu erstellen, gibt es Ansätze, statt den Fehlerbäumen Fehlernetze zu verwenden (siehe [MP01]). Der Aufbau als Baum oder Netz steht bei dieser Arbeit nicht im Vordergrund. Deshalb wird im Folgenden der Begriff Fehlerbaum als Stellvertreter für beides, Bäume und Netze, verwendet. Die Beschreibung der Abhängigkeiten zwischen den Fehlern ist eine spezifische Eigenschaft der Fehlerbaumanalyse. Abbildung 2.10 stellt die wesentlichen Elemente zur Beschreibung dieser Abhängigkeiten dar. Folgefehler werden als Rechtecke dargestellt. In der obersten Ebene des Fehlerbaumes sind die Fehler, für welche die Ursachen dargestellt werden. Handelt es sich bei einer Ursache um einen Mehrfachfehler, so werden dessen Teil-Fehler anhand einer UND-Verknüpfung explizit aufgeführt. Kommen für einen Fehler verschiedene Fehlerursachen in Frage, so werden diese mit einer ODER-Verknüpfung aufgeführt. Es gibt Situationen, in denen ein Fehler oder eine Kombination von Fehlern nur zu einem Folgefehler führen, wenn Bedingungen im System erfüllt sind. Dies wird mit dem Bedingungssymbol dargestellt, welches sich entweder direkt nach Fehlern oder direkt nach UND-Verknüpfungen befinden kann. Wie bei dem Fehlernetz der FMEA werden auch im Fehlerbaum nur Zusammenhänge dargestellt, die von einer inneren Komponente zu einer äußeren reichen. Entsprechend können aus dem Fehlernetz der FMEA die Zusammenhänge zu Einfachfehlern übertragen werden.

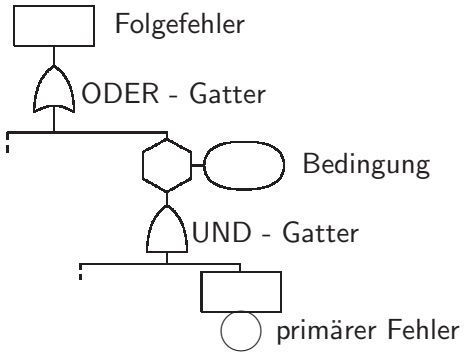


Abb. 2.10: Fehlerbaumanalyse-Symbole

Definition 2.4.1 (Fehlernetz für FTA)
Die Menge der Bedingungen sei COND.

Das *Fehlernetz für FTA* ist eine Relation
$\texttt{FAIL\_FAIL} \subseteq \texttt{REQ\_FAIL} \times \mathcal{P}(\texttt{REQ\_FAIL}) \times \texttt{COND}$
bei der gilt: $\forall((cr_1, f_1), \bigcup_{i\in[2..n]} \{(cr_i, f_i)\}, c) \in \texttt{FAIL\_FAIL} : (cr_1, cr_i) \in \texttt{REQ\_REQ}$. ⌟

minimale Schnitt-mengen

Die möglichen Kombinationen primärer Fehler, die einen Folgefehler auf oberster Ebene des Baumes haben, werden in so genannten Schnittmengen zusammengefasst. Teilaufgabe einer Fehlerbaumanalyse ist es, jeweils die minimalen Schnittmengen an primären Fehlern zu ermitteln, die zu dem betrachteten Folgefehler führen. Eine Schnittmenge ist minimal, wenn es keine andere Schnittmenge gibt, die eine echte Teilmenge dieser Schnittmenge ist. Die Ermittlung dieser Schnittmengen dient mit als Indiz, um den Effekt der Vermeidung primärer Fehler zu erkennen, denn kann ein Fehler aus einer minimalen Schnittmenge ausgeschlossen werden, so kann diese Schnittmenge keine Ursache für den zu untersuchenden Folgefehler mehr sein. Die Ermittlung der minimalen Schnittmengen geschieht, in dem der Baum in die disjunktive Normalform gebracht wird. Es werden alle Zwischenebenen entfernt, so dass ein Baum entsteht, der unter der Wurzel ein ODER-Gatter hat und darunter die durch ein UND-Gatter verbundenen minimalen Schnittmengen, die zur Auslösung des Fehlers genügen.

quantitative Auswertung

Bei der Fehlerbaumanalyse werden die Wahrscheinlichkeiten der primären Fehler zur Berechnung der Wahrscheinlichkeit des zu untersuchenden Folgefehlers verwendet. An dieser Stelle wird eine Berechnung vorgestellt, die eine stochastische Unabhängigkeit der primären Fehler annimmt und die Bedingungen bei den Ästen im Baum ignoriert. Damit findet eine obere Abschätzung der Fehlerwahrscheinlichkeit statt. Genauere Berechnungsmethoden sind in [VGRH81] zu finden. Die Wahrscheinlichkeit, dass eine Schnittmenge von Fehlern $s_i = \{f_{1_i}, f_{2_i}, \ldots f_{n_i}\}$ eintritt, ist das Produkt der Wahrscheinlichkeiten der Fehler. Die Wahrscheinlichkeit des zu untersuchenden Folgefehlers $P(f)$ ist dann das Produkt der Wahrscheinlichkeiten, dass die Schnittmengen s_i mit $i \in \{1..m\}$ jeweils nicht auftreten. Es gilt:

$$P(f) = 1 - \prod_{i=1}^{m}(1 - \prod_{j=1}^{n_i} P(f_{j_i}))$$

Beispiel 2.4.1 (Fehlerbaum)
Gegeben sei eine Funktion zur Stabilisierung der Drehrichtung eines Fahrzeugs. Diese Funktion hat unter anderem die Aufgabe, zu verhindern, dass das Fahrzeug untersteuert. Der zu untersuchende Fehler ist der, bei dem die Funktion nicht ausgeführt wird. Die Funktion kann von zwei Systemen im Fahrzeug erbracht werden: der Überlagerungslenkung und der Regelung der Drehzahl der einzelnen Räder. Die Überlagerungslenkung bekommt ihre Daten von einem Raddrehzahlsensor, der die Geschwindigkeit der einzelnen Räder misst, und einem Giersensor, welcher die Drehgeschwindigkeit des Fahrzeugs um die senkrechte Achse misst.

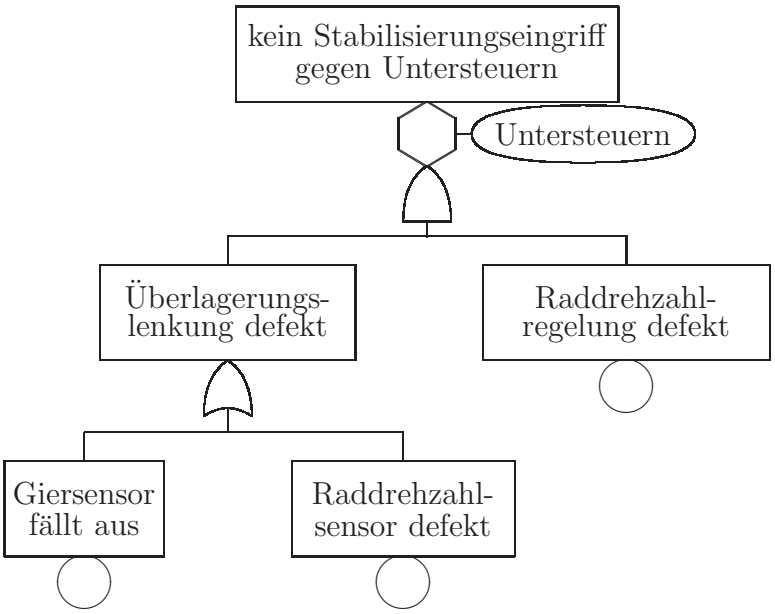


Abb. 2.11: Beispiel eines Fehlerbaums

Gegeben seien folgende Fehler mit deren Wahrscheinlichkeiten (relativ zur Betriebszeit des Fahrzeuges):

f_1 = kein Stabilisierungseingriff gegen Untersteuern	
f_2 = Überlagerungslenkung defekt	
f_3 = Raddrehzahlregelung defekt	$P(f_3) = 10^{-4}$
f_4 = Raddrehzahlsensor defekt	$P(f_4) = 10^{-4}$
f_5 = Giersensor fällt aus	$P(f_5) = 10^{-4}$

Der Fehler f_1 kann nur dann auftreten, wenn das Fahrzeug gerade untersteuert. Im Sinne einer Worst-Case-Abschätzung wird davon ausgegangen, dass ein sehr sportlicher Fahrer maximal 1/10 der Betriebszeit untersteuert. Abbildung 2.11 zeigt einen vereinfachten Fehlerbaum zu diesem System.

Die minimalen Schnittmengen zu diesem Baum sind:

$$\{\{f_3, f_4\}, \{f_3, f_5\}\}$$

Die Wahrscheinlichkeit des Fehlers f_1 ist entsprechend:

$$\begin{aligned} P(f_1) &= 1 - (1 - P(f_3) \cdot P(f_4)) \cdot (1 - P(f_3) \cdot P(f_5)) \\ &= 1 - (1 - 10^{-4} \cdot 10^{-4}) \cdot (1 - 10^{-4} \cdot 10^{-4}) \\ &= 1 - (1 - 10^{-8})^2 \\ &\approx 2 \cdot 10^{-8} \end{aligned}$$

Würde man bei dieser Rechnung noch die Bedingung des Zustands Untersteuern berücksichtigen, so sinkt die Wahrscheinlichkeit auf $2 \cdot 10^{-9}$. ⌟

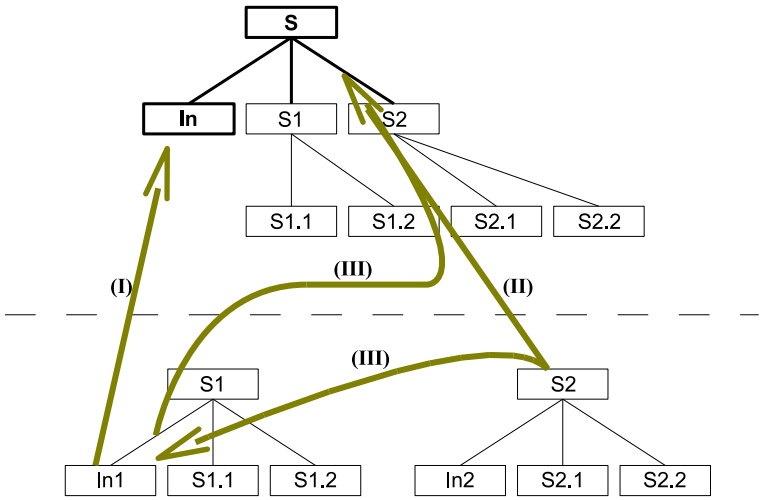


Abb. 2.12: Zusammenführen zweier Analysen

2.5 Zusammenführung von Analysen

Teilsysteme

Die Gesamtprodukte sind oft komplex und werden deshalb in Teilsysteme aufgeteilt. Die Entwicklung und Analyse der Teilsysteme findet simultan statt. Analog zur Entwicklung finden auch die Sicherheitsanalysen simultan statt. Die FTA bzw. FMEA der Teilsysteme werden für das Gesamtprodukt zusammengeführt. Dieser Abschnitt zeigt, wie die Schnittstellen definiert werden, um diese Analysen zusammenzuführen.

Schnittstellenbetrachtung

Werden die Sicherheitsanalysen von Teilsystemen zusammengefügt, so ist die neue zu betrachtende Schnittstelle für die Fehlerfolgen die des zusammengefügten Produktes. Die Sicherheitsanalysen für die Teilsysteme sind, wie in Abschnitt 2.3 und 2.4 beschrieben, gegeben. Die Ausgaben der Teilsysteme sind nicht immer direkt mit den Ausgaben des zusammengefügten Systems verbunden. Teile der Ausgaben des einen Produkts können die Eingaben eines anderen Produkts sein und umgekehrt. Ähnlich wie beim Aufbau des Funktionsnetzes wird eine transitive Hülle gebildet, welche die Wirkung eines Fehlers durch die anderen Teilkomponenten hindurch nach außen hin betrachtet. Um die transitive Hülle bilden zu können, muss bei den Fehleranalysen zusätzlich angegeben werden, wie sich Fehler von den Eingaben eines Systems hin zu den Ausgaben ausbreiten.

Systemstrukturbaum

Die Wurzel des Systemstrukturbaums ist die Systemgrenze. Eine Möglichkeit, die Wirkung von Fehlern an der Eingabe hin zu der Ausgabe eines Systems auszudrücken, ist die Erstellung einer künstlichen Teilkomponente, welche die Eingabeschnittstelle ins System repräsentiert. Die Systemstrukturbäume haben dann, wie in Abbildung 2.12 zu sehen, jeweils direkt unter der Wurzel die Eingabekomponente, welche die Anforderungen und Fehler der Umwelt enthält. Für den neu zusammengesetzten Systemstrukturbaum ist die Eingabekomponente wieder direkt unter der Wurzel. Die Eingabekomponenten der Teilsystemstrukturbäume müssen also entsprechend überführt werden.

Zusammenfügen

Das Zusammenfügen der Komponenten $S1$ und $S2$ beinhaltet folgende Teilaufgaben: Die beiden Wurzeln werden unter der neuen Hauptwurzel eingehängt. Die Verknüpfung der Fehler und der Anforderungen kann in drei Kategorien eingeteilt werden. Beziehen sich die Anforderungen in den Eingabekomponenten an die Umgebung auch im zusammengesetzten System an die Umgebung, so können sie direkt in die Eingabekomponente des zusammengesetzten Systems übernommen werden (Abbildung 2.12, (I)). Die restlichen Anforderungen der Eingabekomponenten beziehen sich auf die anderen Komponenten im System und werden entsprechend implizit in der Funktions- / Fehlervernetzung festgehalten. Beim Aufbau der Fehler- / Anforderungsnetze kommen die restlichen zwei Kategorien zum tragen. Sind die Anforderungen der Wurzeln die gleichen, wie die der Wurzeln der Teilkomponenten bzw. der Eingabekomponenten, so können diese direkt übernommen und verbunden werden (Abbildung 2.12, (II)). Sind die Anforderungen der Teilsysteme gleichzeitig die Anforderungen der Eingabekomponenten anderer Teilsysteme, so werden die Zusammenhänge durch die anderen Teilsystemen hindurch weiter verfolgt, bis sie an Anforderungen des zusammengesetzten Systems gelangen (Abbildung 2.12, (III)).

Komplexität

Die Ermittlung der Zusammenhänge zwischen den Fehlern im Fall (III), in dem durch andere Teilsysteme hindurch die Folgen betrachtet werden, ist spezifisch für die jeweiligen Sicherheitsanalysetechniken. Grundlegend ist hier ein Kompromiss zwischen dem Aufwand für die Analyse und der Menge der exakt gefundenen Fehlerabhängigkeiten zu schließen. Dieser Kompromiss besteht darin, spezifische Annahmen über die Systeme zu treffen bzw. die zu betrachtenden Fehlerzusammenhänge entsprechend abstrakt darzustellen oder auszuschließen. So kann man zum Beispiel in einer Variante der FMEA nur das System ohne Sicherheitsfunktionen betrachten. Hier gilt z.B. die Annahme, dass die Betrachtung von Einfachfehlern genügt, da kein Aufhebungsmechanismus (z.B. durch Redundanz usw.) im System vorhanden ist. Diese Mechanismen werden dann erst in den Maßnahmen berücksichtigt. Verschiedene Arten dieser Kompromisse werden in Kapitel 5 definiert und formal beschrieben.

2.6 Zusammenfassung

Dieses Kapitel beschreibt zwei produktorientierte Verfahren der Funktionssicherheitsanalyse, die in der Industrie angewandt werden. Dabei werden Ansatzpunkte herausgestellt, an denen die Durchführung der Verfahren automatisiert werden kann. Dieses Kapitel dient so zur Motivation und Positionierung weiterer Arbeiten im Bereich der Sicherheitsanalyse.

Das Kapitel ist in drei Teile gegliedert. Der erste Teil definiert informell die Begriffswelt der Funktionssicherheit. Diese wird beginnend mit dem Fehlerbegriff über den Unfall hin zur Funktionssicherheit aufgebaut. Die definierten Begriffe orientieren sich an deutschsprachigen Standards, da im englischsprachigen Raum die Semantiken der Begriffe häufig mehrdeutig sind.

Der zweite Teil ordnet die Aufgabe der Funktionssicherheitsanalyse in den Entwicklungsprozess ein. Es werden Schnittstellen zu parallel laufenden Aktivitäten identifiziert, wie auch die Möglichkeit der Anwendung der Analysen hinsichtlich des Fortschritts der Entwicklung. Damit werden Rahmenbedingungen an eine spätere Automatisierung und Modellierung der Analysen geschaffen.

Der dritte Teil beschreibt die Durchführung zweier Analysen, der FMEA und der FTA. Die einzelnen Schritte zur Erstellung der Dokumente dieser Analysen werden erklärt. Sie beginnen mit der Erstellung eines Systemstrukturbaums und gehen über die Fehlernetzerstellung hin zur Bewertung der Fehler. Zu den einzelnen Schritten werden Ansatzpunkte zur formalen Modellierung, Interpretation und Automatisierung gegeben.

Kapitel 3

Grundlegende Modelle und Modellierungstechniken

Dieses Kapitel beschreibt Modelle und Modellierungstechniken zur Spezifikation eingebetteter Systeme[1] und derer Umgebung im Fahrzeug. Ziel ist es, anhand von Referenzmodellen und formalen Beschreibungen der Modellierungstechniken eine Basis zur systematischen Fehlermodellierung und damit zur Funktionssicherheitsanalyse (siehe [Lev95]) zu schaffen. Eine Funktionssicherheitsanalyse betrachtet nur die Fehler, die sie identifizieren und modellieren kann. Die Beschreibung der Fehler ist immer relativ zu der korrespondierenden Systemspezifikation und damit abhängig von den darin enthaltenen Modellen. Die Modelle sind somit ausschlaggebend für die Gestaltung einer Funktionssicherheitsanalyse. Kern ist hierbei der Begriff Modell, der in dieser Arbeit wie folgt definiert ist:

Definition 3.0.1 (Modell)
Ein *Modell* ist eine Abstraktion eines Sachverhalts, bei dem die für einen bestimmten Zweck wesentlichen Eigenschaften erhalten bleiben. (siehe auch [Min65]) ⌟

Erläuterung Modell

In dieser Arbeit dienen Modelle der verständlichen Darstellung von Annahmen über Sachverhalte (als Referenzmodelle) und zur Ermöglichung eines systematischen theoretischen Umgangs mit Teilaspekten komplexer Sachverhalte (hinsichtlich der Spezifikation von Systemen). Bei Modellen findet immer eine Abstraktion der Sachverhalte statt, die nicht dem Zweck des Modells dienlich sind. Der Inhalt eines Modells ist ein so weit wie möglich vereinfachter Sachverhalt, ohne dass wesentliche Gegebenheiten verloren gehen. Die Inhalte, der Grad der Formalität und die Darstellung der Modelle beeinflussen deren Nutzen. So eignen sich formale bzw. mathematische Modelle besonders zur Verarbeitung mit Software, da sie präzise interpretierbar sind. Informelle Modelle können hingegen einem besseren Überblick bzw. Verständnis dienen. In dieser Arbeit sind die Inhalte der Modelle (angelehnt an [Lev95], S. 306):

[1] Eingebettete Systeme sind Rechnersysteme, die in mechatronische Gesamtsysteme eingebaut sind. [FGP04]

- *Struktur*: die (statischen) Zwischenbeziehungen der Elemente entlang einiger Dimensionen (wie Raum, Zeit, relative Wichtigkeit oder logischen bzw. entscheidenden Eigenschaften).
- *Unterscheidende Qualitäten*: eine qualitative Beschreibung bestimmter Variablen, Parametern und Eigenschaften, die das System charakterisieren und ähnliche Strukturen unterscheiden (z.B. Qualitätsanforderungen).
- *Verhalten / Dynamik*: eine Beschreibung der Parameter und Variablen in Verbindung mit den unterscheidenden Qualitäten bezüglich des Wertebereichs, des Zeitverlaufs und dem Grad der Gewissheit in relevanten Situationen, also entlang der Dimensionen Wertebereich, Wahrscheinlichkeit und Zeit.

Kapitelübersicht

In Abschnitt 3.1 werden mit informellen Referenzmodellen die in dieser Arbeit getroffenen Annahmen über den Aufbau (Struktur) und die Eigenschaften (unterscheidende Qualitäten) eingebetteter Systeme beschrieben. Ziel ist es, eine Systemvorstellung aufzubauen, in welche die in diesem Kapitel vorgestellten Modelle und Modellierungstechniken eingeordnet werden können. Auch werden anhand dieser Referenzmodelle die Schwerpunkte der zu behandelnden Inhalte bei der System- und Fehlermodellierung abgeleitet.

Abschnitt 3.2 beschreibt, wie Modelle des Verhaltens eingebetteter Systeme und deren Umgebung aussehen können und wie Modelle der Artefakte aussehen können, auf denen das Verhalten implementiert ist.

Die Verhaltensmodelle können mit den Modellierungstechniken aus Abschnitt 3.3 spezifiziert werden. Im Wesentlichen werden verschiedene Darstellungen der Verhaltensmodelle aufgeführt, die spezifisch auf die Beschreibung bestimmter Aspekte der Modelle zugeschnitten sind. Hierzu gehören Prädikate und Automaten, sowie deren Sonderformen aufgelöste Gleichungen und Betriebsmodi-Automaten. Es wird skizziert, wie diese Modellierungstechniken als Grundlage zur Beschreibung von Fehlverhalten herangezogen werden können.

Die Modelle werden in einem Entwicklungsprozess systematisch und schrittweise erstellt. Bei diesem Vorgehen werden Modelle in präzisere Modelle überführt und Modelle von Teilen des Systems zusammengeführt. Die Abhängigkeiten zwischen diesen Modellen werden in Abschnitt 3.4 beschrieben. Anhand dieser Abhängigkeiten werden Ansatzpunkte gegeben, um potentielle Fehler zu identifizieren, die aus Abweichungen dieser Abhängigkeiten entstehen.

Abschnitt 3.5 fasst schließlich die Ansatzpunkte zur Fehleridentifikation und -modellierung, die sich aus den vorhergehenden Abschnitten ergeben, zusammen.

3.1 Referenzmodelle

In diesem Abschnitt werden Systemvorstellungen als Referenzmodelle präsentiert, die einen Überblick über die Modellierung eingebetteter Systeme geben. Sie stellen

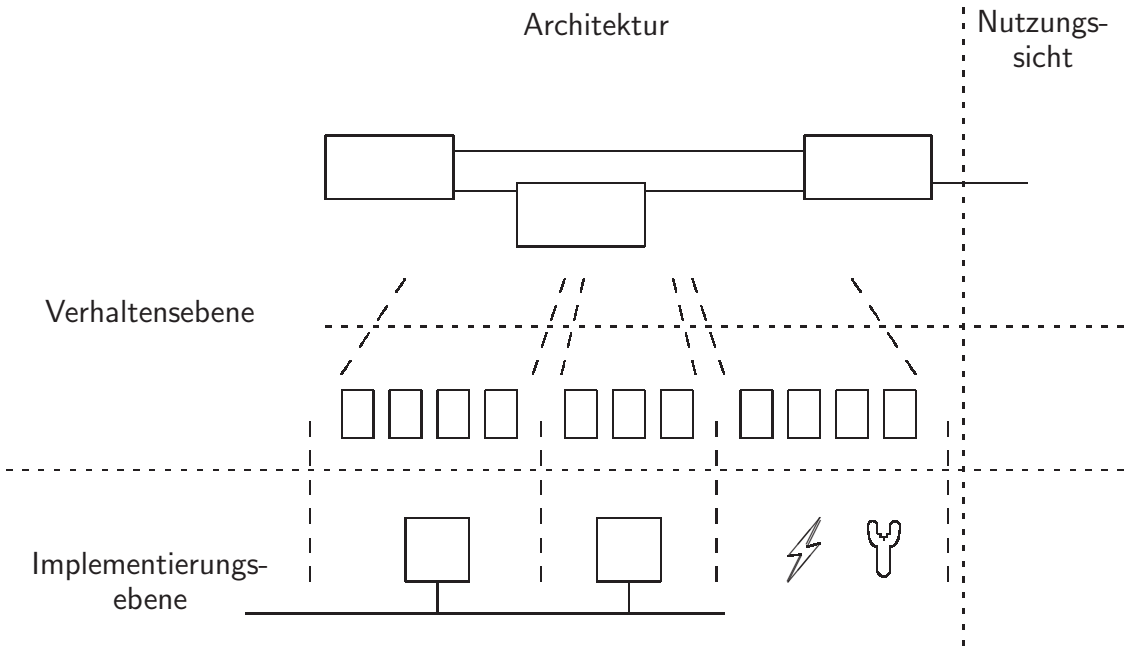


Abb. 3.1: Abstraktionsschichten bei mechatronischen Systemen

dar, welche Eigenschaften abstrahiert werden und welche Auswirkungen die Abstraktion auf die Modellierung hat.

Verhaltensebene

Ein *mechatronisches System* setzt sich aus mechanischen, elektrischen[2] und elektronischen[3] Anteilen zusammen. Die Anteile werden zu einem gemeinsamen Ganzen zusammengefügt und erfüllen gemeinsam eine Aufgabe. Der Schwerpunkt der Modellierung und der Funktionssicherheitsanalyse liegt auf den funktionalen Zusammenhängen zwischen den Anteilen des Systems. Unter funktional wird hierbei das Verhalten des Systems verstanden. Bei der Modellierung der Systeme gibt es entsprechend eine Abstraktionsebene, bei der eine Verhaltensextraktion stattfindet und die restlichen Aspekte des Systems nicht betrachtet werden. Diese Ebene wird *Verhaltensebene* genannt. Abbildung 3.1 stellt diese funktionale Abstraktionsebene dar, in der die Verhaltensweisen der Bestandteile eines mechatronischen Systems modelliert und für das gesamte System in Zusammenhang gebracht werden können. Mit dem steigenden Anteil an elektronischen Systemen in Fahrzeugen, die immer komplexere Steuerungsaufgaben bewältigen müssen, nimmt die Modellierung der Verhaltensebene einen immer wichtigeren Platz in der Entwicklung ein (siehe [AUT06]).

Domäneneigenschaften

Die Eigenschaften der Funktionen, die das Verhalten der Teile eines Systems beschreiben, sind, entsprechend der Domäne, der sie zugeordnet sind, verschieden. Die Domäne *Mechanik* (dargestellt in Abbildung 3.1 als Zange) enthält Funktionen, die zeitkontinuierlich sind. Die mathematische Beschreibung der Funktionen ist hierbei durch die physikalischen Eigenschaften der Elemente vorgegeben und

[2]Der Begriff *Elektrik* bezieht sich auf die Versorgung mit Energie.

[3]Der Begriff *Elektronik* bezieht sich hier auf den Signal-/Informationscharakter von elektrischem Strom.

enthält meist keine komplizierten Algorithmen über die Zeit. Viele Größen sind dort wertkontinuierlich. Einige Größen sind zusätzlich stetig. Zum Beispiel kann sich die Position eines trägen Teils nicht sprunghaft ändern. Auch die der Domäne *Elektrik* (dargestellt als Blitz) zugeordneten Funktionen sind zeitkontinuierlich. Allerdings finden bedingt durch Relais oder andere Schaltelemente wesentlich öfter diskrete Wertsprünge statt. Bei den der Domäne *Elektronik* zugeordneten Funktionen stehen Signale bzw. Informationen im Vordergrund. Diese sind heutzutage nur noch in den seltensten Fällen wertkontinuierlich. Sie werden durch den Einsatz der Digitaltechnik meist mit diskreten Werten dargestellt. Die diskreten Werte der Signale sind innerhalb der Elektronik nur zu bestimmten Zeitpunkten relevant. Die Funktionen lassen sich so über Komponenten mit diskreten Berechnungszeitpunkten beschreiben (siehe [AD94]). Viele der Systeme sind zusätzlich getaktet, was innerhalb einer Taktung eine zeitdiskrete Beschreibung der Funktionalität mit einheitlichen Zeitabständen ermöglicht.

Verhaltensabstraktionsebenen

Die Elektronik eines mechatronischen Systems ist meist in einer Menge von Steuergeräten untergebracht, die über Bussysteme miteinander kommunizieren. Die Komplexität in den Steuergeräten ist meist so hoch, dass diese wie allgemeine Rechner organisiert sind. Sie haben ein Betriebssystem, eine Kommunikationsschnittstelle und die eigentlichen (Funktions-)Applikationen (siehe [AUT06]). Diese Rechnersysteme mit der Aufgabe der Steuerung physikalischer Prozesse nennt man *eingebettete Systeme* (siehe Abbildung 3.2). Aufgrund des Aufbaus der Steuergeräte bietet es sich an, verschiedene Abstraktionsebenen für die Funktionalität, also *Verhaltensabstraktionsebenen*, zu definieren. Diese Abstraktionsebenen sind, ähnlich wie in dem ISO/OSI-Schichtenmodell (siehe [ISO94]), durch Zwischenschichten realisiert. Auf der untersten Abstraktionsebene wird (ähnlich der OSI-Schicht-1) die tatsächliche Belegung der einzelnen Signale auf der Hardware betrachtet. Auf dieser Ebene wird zwar alles „naturgetreu“ nachgebildet, die Modelle werden aber schnell unübersichtlich und komplex. Je nach Bedarf lassen sich (analog zum OSI-Referenzmodell) eine Reihe von Zwischenebenen einziehen, bis hin zur Verhaltensebene der Applikation (analog OSI-Schicht-7). Auf dieser Ebene wird die gewünschte Funktionalität modelliert, die das beobachtbare Verhalten des Systems nach außen maßgeblich bestimmt. Das genaue Verhalten auf dieser obersten Ebene hängt von der Realisierung der darunter liegenden Ebenen ab. Es kann aber trotz der Realisierungsabhängigkeit eine angemessene Modellierung des Verhaltens des Systems stattfinden, wenn man davon ausgeht, dass die darunter liegenden Realisierungen der Ebenen das Verhalten nur gering oder nicht beeinflussen (siehe [WFH+06]).

Zusammenhang zwischen Ebenen

Die jeweils höher liegenden Abstraktionsebenen beinhalten implizit die darunter liegenden Ebenen. Auch wenn das Verhalten bzw. die Funktionalität der darunter liegenden Ebene nicht direkt modelliert wird, können die Elemente / Funktionen der jeweils darunter liegenden Ebenen versagen, also ihren Dienst nicht erbringen und auf diese Weise Verhaltensfehler in die jeweils darüber liegende Ebene induzieren. Hat man sich in einem Modell für eine Abstraktionsebene entschieden, so ist es notwendig, die Schnittstelle bzw. Einbindung der darunter liegenden Ebenen, so wie die möglichen Fehler zu kennen, die an den Schnittstellen bzw. durch die abstrahierten

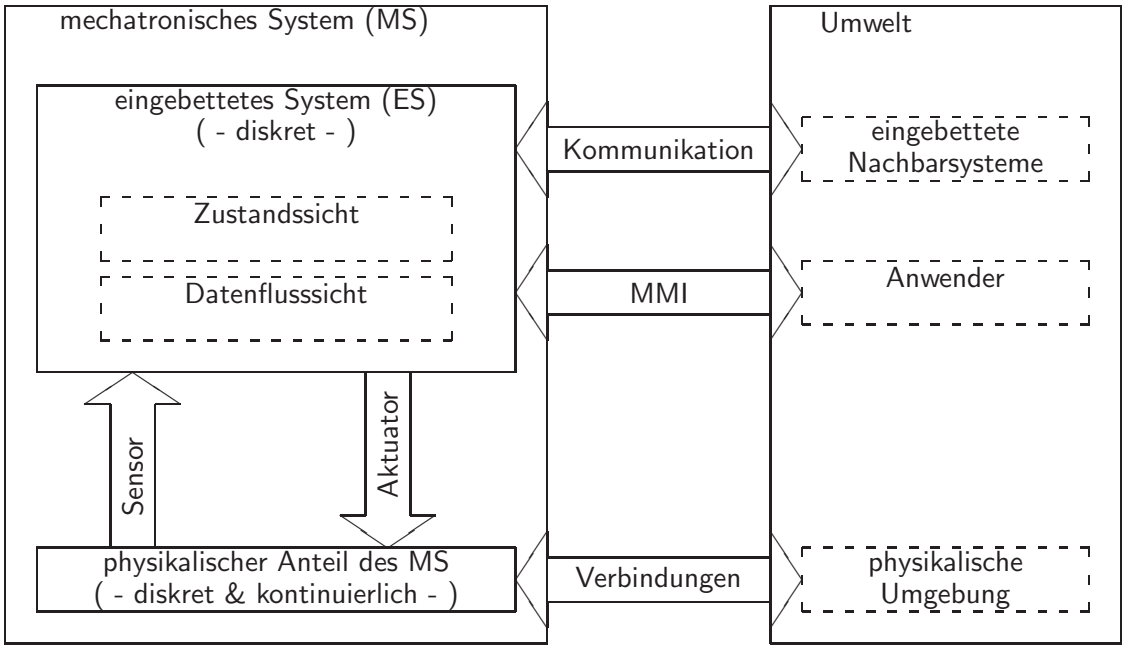


Abb. 3.2: System und Umgebung (angelehnt an [FGP04] und [VDI04])

Eigenschaften auftreten können. Die Einbindung der tiefer liegenden Abstraktionsebenen kann in einem Modell explizit modelliert werden. Durch die Angabe der Abstraktionsebene kann aber auch implizit eine Abbildungs-/Zugriffsvorschrift angenommen werden, von der auf potenzielle Fehler geschlossen werden kann (siehe Abschnitt 4.5.2). Abstrahiert man zum Beispiel von der Verteilung der Funktionen auf Steuergeräte, so kommen bei der Übertragung der Signale zwischen den Systemen die Standard-CAN-Übertragungsfehler als potenzielle Fehler in Frage.

Nutzungsschnittstelle

Die Beschreibung eines mechatronischen bzw. eingebetteten Systems enthält als wesentlichen Bestandteil die Interaktion mit der Umwelt. Diese Interaktion wird durch das Verhalten des Systems an dessen Systemgrenze bestimmt. Die Grenze zwischen dem System und der Umwelt wird auch als *Nutzungsschnittstelle* bezeichnet. Im Gegensatz zu dem Verhalten des Systems nach außen steht die Architektursicht, die den Aufbau des Systems und das Zusammenwirken der Bestandteile des Systems beschreibt. Wie in Abbildung 3.1 (rechts senkrecht dargestellt) zu sehen, ist die Nutzungsschnittstelle unabhängig von den Abstraktionsebenen. Beschreiben die Modelle des Systems auf verschiedenen Abstraktionsschichten das gleiche Verhalten des Gesamtsystems, so ist auch die Nutzungsschnittstelle auf beiden Ebenen gleich. Im Verlauf der Entwicklung und mit der Modellierung bzw. Festlegung der Realisierung tiefer liegender Abstraktionsebenen kann eine Verfeinerung oder gar eine Änderung der Nutzungsschnittstelle einhergehen. Ziel ist es, im Laufe der Entwicklung Nutzungsschnittstellenbeschreibungen zu erhalten, die in einer Verfeinerungsbeziehung stehen, die in Richtung der detailliertesten Abstraktionsebene immer feiner wird. Auch die Fehler an der Nutzungsschnittstelle können nach den verschiedenen

Abstraktionsebenen gegliedert sein. Jeder Fehler, der an der Nutzungsschnittstelle auftritt, ist Teil einer Funktionssicherheitsanalyse, da hier die Umwelt beeinflusst wird (siehe [Eri05]).

Regelungs- versus Interaktionssicht

Betrachtet man die Elektronik als Kernstück der Funktionalität eines mechatronischen Systems, so wird mit dessen Modellierung der Großteil der Modellierung des Systems abgedeckt. Die Logik dieser eingebetteten Rechnersysteme hat die Aufgabe, zwei verschiedenartige Sichten zu verbinden. Aus Sicht der Regelungstechnik dienen die Systeme dazu, physikalische Prozesse zu steuern. Diese Steuerung findet über Funktionen statt, die abhängig von den Eingabewerten einen Ausgabewert berechnen. Sie sind im wesentlichen Diskretisierungen kontinuierlicher Funktionen. Die meisten dieser Funktionen haben, abgesehen von Integralwerten, keine große Menge an Zuständen, die das Verhalten über einen längeren Zeitraum beeinflussen. Die steigende Vernetzung mit anderen Systemen und die steigende Interaktion mit dem Menschen fordern auch die Sicht der ereignisbasierten Systeme, wie z.B. Automaten. Diese Systeme zeichnen sich dadurch aus, auf konkrete Ereignisse von außen zu reagieren und mit einer Menge verschiedener Zustände verschiedene Verhaltensweisen über längere Zeiträume zu zeigen, oder komplizierte algorithmische Aufgaben bewältigen zu können. Der Fokus bei diesen Systemen liegt in der Interaktion bzw. den Abläufen und nicht auf der Berechnung eines Funktionswertes. Diese Kombination von Domänen stellt Herausforderungen an die formale Verifikation der Systeme. Diese Herausforderungen werden in Abschnitt 6.1 genauer beschrieben.

Schnittstellen der Systeme

Abbildung 3.2 stellt die Arten von Systemen dar, die ein eingebettetes System umgeben. In der Regelungssicht werden die Systeme von Sensoren mit Werten beliefert, die physikalische Größen abgreifen. Die Sensoren sind die eingehende Schnittstelle von der kontinuierlichen Sicht zu der diskreten Rechnerwelt. Umgekehrt liefern die eingebetteten Systeme Werte an Aktuatoren, um physikalische Vorgänge zu beeinflussen. Sie sind die ausgehende Schnittstelle zur physikalischen Welt. Die Fehler, die an diesen Schnittstellen auftreten können sind über die Bauart und den physikalischen Wert bzw. dem mechanischen Verhalten des Restsystems bestimmt. An der Mensch-Maschinen-Schnittstelle können die Systeme Nachrichten vom Nutzer bekommen oder Nachrichten an den Nutzer senden. Von den Schnittstellen können systematisch potenzielle Fehler abgeleitet werden. So sind z.B. an der Mensch-Maschine-Schnittstelle (MMI) falsche Eingaben relevant. Bei der Interaktion mit anderen Systemen können zusätzlich spezifische Fehler zur Reihenfolge der Nachrichten identifiziert werden. Die Einordnung der Fehler anhand der Schnittstelle ist in Abschnitt 3.5 zu finden. Folgendes Beispiel skizziert die Schnittstellen eines Systems und daraus ableitbare Fehler:

Beispiel 3.1.1 (Schnittstellen als Indiz für potenzielle Fehler)
Gegeben sei ein *Lenksystem* eines Fahrzeugs mit einer *Überlagerungslenkung*. Eine Überlagerungslenkung berechnet abhängig von der Fahrsituation einen Überlagerungswinkel und addiert diesen zu dem Lenkradwinkel. Sie hat zwei Teilfunktionen: die *geschwindigkeitsabhängige Lenkübersetzung* und die *Stabilisierungsfunktion* (Fahrer unterstützende Lenkkorrektur bei instabiler Fahrsituation).

Über Sensoren bekommt die Überlagerungslenkung die aktuellen Werte der Gierrate[4], der Querbeschleunigung, des Lenkradwinkels und der Radgeschwindigkeiten. Mögliche ableitbare Fehler von dieser Schnittstelle sind *Sensorfehler* (z.B. Justierfehler, Sensorausfall).

Die Überlagerungslenkung (UL) und die Raddrehzahlsteuerung (RDS[5]) interagieren miteinander. Deren Stabilisierungsfunktion (Bremsen der Räder bei instabiler Fahrsituation) kann mit Hilfe der Überlagerungslenkung teilweise später eingreifen. Der Fahrkomfort wird so erhöht, da ein Bremseingriff weniger komfortabel ist, als ein Lenkeingriff. Hierzu ist ein Austausch von Nachrichten notwendig, bei dem beide Systeme jeweils den Zustand des anderen Systems kennen müssen, um richtig zu handeln. Mögliche ableitbare Fehler von dieser Schnittstelle sind *Nachrichtenfehler* (z.B. falsche Nachricht, verspätete Nachricht).

Des Weiteren kann der Fahrer die Stabilisierungsfunktion der Überlagerungslenkung (und der Raddrehzahlsteuerung) mit einem Taster aktivieren, bzw. deaktivieren. Mögliche ableitbare Fehler sind *diskrete Sensorfehler* (z.B. Signal des Tasters ohne Berührung, keine Reaktion des Tasters bei Berührung). ⌟

3.2 Modellierung der Systeme

Eigenschaften der Modelle

Eine Grundlage einer Funktionssicherheitsanalyse ist die Systemdefinition (siehe Kapitel 2). Um eine automatische Verarbeitung dieser Systemdefinition zu ermöglichen, findet die Definition mit Modellen statt, die symbolisch und statisch sind. Es werden in dieser Arbeit Modelle eines Systems betrachtet, die für eine Funktionssicherheitsanalyse relevant sind. Die Modelle unterscheiden sich abhängig von der Art der Abstraktion, der Art des beschriebenen Systems und dem Zustand des beschriebenen Objekts. Besonders der Grad des Determinismus, wie auch die Kontinuität sind unterscheidende Eigenschaften (bzw. Qualitäten) der Modellelemente. Die Eigenschaften (nach [Lev95], S. 305) der verwendeten Modelle werden im Folgenden erläutert:

- *symbolische Modelle*: Modelle, welche die strukturellen Eigenschaften eines Systems mit Eigenschaftsbeschreibungen und logischen Aussagen repräsentieren. Sie nutzen mathematische oder logische Operationen um das Verhalten des Originalsystems zu beschreiben.

- *statische Modelle*: Modelle, die ihre Merkmale zeitunabhängig beibehalten und nicht ändern.

- *deterministische Modelle*: Modelle, die auf gleiche Eingabefolgen oder Reize bei gleichen Randbedingungen immer das gleiche Ergebnis liefern.

[4]Die Gierrate ist die Drehung des Fahrzeugs um die Hochachse.
[5]Wird häufig auch als elektronisches Stabilitäts-Programm (ESP) bezeichnet.

- *unscharfe Modelle*: Modelle, die sich bei diskreten Funktionen deterministisch verhalten, bei kontinuierlichen Funktionen jedoch Abweichungen in einem Toleranzbereich zulassen.

Grad des Determinismus

Die meisten der Modelle eingebetteter Systeme sind deterministisch. Bei der Beschreibung physikalischer Vorgänge kann aufgrund der Realisierung (Bauteilungenauigkeit, nicht modellierte Umwelteinflüsse, ...) eine Unschärfe angegeben werden, innerhalb derer der tatsächliche Funktionswert von dem ideal beschriebenen Wert abweichen kann. Diese Unschärfe kann zu Nichtdeterminismus an den Schnittstellen der diskreten Systeme zu den kontinuierlichen Systemen führen. Die Umwelt hingegen wird meist mit nichtdeterministischen Modellen dargestellt, da hier oft beliebige Verhaltensweisen angenommen werden müssen. Ebenso kann bei der Beschreibung der Fehler die Verwendung nichtdeterministischer Modelle notwendig sein, da z.B. ein lockerer Kontakt zu einer beliebigen Übertragung oder Nicht-Übertragung eines Signals führen kann.

Abschnittsübersicht

Abschnitt 3.2.1 beschreibt, wie das Verhalten (bzw. die Funktionalität) eines Systems modelliert werden kann und Abschnitt 3.2.2 die Modellierung der Artefakte, auf denen das Verhalten abgelegt ist (z.B.: Software, Sensor, ...). Die Verhaltensmodelle sind hierbei mathematische Modelle, die aus Relationen zwischen Eingaben und Ausgaben bestehen. Die Implementierungsmodelle haben ihren Schwerpunkt auf der Struktur und werden als Instanzen von UML-Klassen-Diagrammen modelliert.

3.2.1 Modellierung der Verhaltensebene

Dieser Abschnitt beschreibt ein mathematisches Konzept zur Verhaltensmodellierung von Systemen. Es wird allgemein beschrieben, wie mit Relationen das Verhalten modelliert werden kann. Dieses Konzept ist die Grundlage für Modellierungstechniken, die in Abschnitt 3.3 zur Spezifikation von Systemen genutzt werden können.

Systeme

Systeme werden auf funktionaler Ebene über ihre Schnittstelle und über ihr Verhalten definiert. Die *Schnittstelle*, über die ein System S mit seiner Umwelt kommuniziert, wird über die (Namen der) Kanäle, deren Datentypen und deren Charakterisierung als Eingabe- oder Ausgabekanal definiert.[6] Die *exemplarische Kommunikation* wird an jedem Kanal f über eine Funktion $f_i(t)$ dargestellt, welche den Wert, der an einem Kanal zu einer bestimmten Zeit t anliegt, angibt (Der Bezeichner f ist hierbei durch den Kanalnamen zu ersetzen und der Wert i identifiziert den exemplarischen Ablauf). Die Menge aller möglichen Funktionen an einem Kanal wird $\mathcal{F}_f = \bigcup f_i(t)$ genannt. Die Menge $\mathcal{F}_f$ stellt den *dynamischen Typ* eines Kanals dar (siehe [LSV95]). Die Definitionsmenge D_f der kanalwertbeschreibenden Funktionen ist bei kontinuierlichen Systemen die Menge der positiven reellen Zahlen einschließlich der Null $\mathbb{R}_0^+$. Die Zeit schreitet also konstant ab einem bestimmten Startzeitpunkt fort. Die Wertemenge aller möglichen exemplarischen Funktionen eines Kanals ist $W_{\mathcal{F}_f} = \bigcup W_{f_i(t)}$,

[6] In einigen Arbeiten wie z.B. [LSV01] werden diese Kanäle auch als externe (Kommunikations-) Variablen bezeichnet.

also die Vereinigung der Wertemengen der einzelnen exemplarischen Funktionen. Diese Wertemenge wird auch als (statischer) *Datentyp* des Kanals bezeichnet, der die möglichen Belegungen des Kanals zeitunabhängig darstellt. Das *Verhalten* eines Systems wird über eine Relation $R_S \subseteq \mathcal{F}_f \times \mathcal{F}_g \times \ldots$ dargestellt, in der alle möglichen Kombinationen exemplarischer Eingabe- und Ausgabekommunikationen in Beziehung gebracht werden.

Kommunikation

Ein System kann entweder atomar sein oder aus mehreren Untersystemen (auch Komponenten genannt[7]) zusammengesetzt sein. Die Systeme werden über *Kommunikationskanäle* verbunden, das heißt, die Werte eines Ausgabekanals eines Systems werden als Eingaben eines anderen Systems genutzt. Kanäle, die nicht mit anderen Systemen verbunden sind, treten als Kanäle des übergeordneten Systems auf. Die *Kommunikation* zwischen Systemen findet ausschließlich über den Austausch von Funktionswerten über die Kanäle statt.

Zeit-Diskretisierung

Die Modellierung mechatronischer Systeme fordert die Berücksichtigung zweier verschiedener Aspekte. Zum Einen wird die Mechanik und Elektrik modelliert. Hier stellen die Ein- und Ausgaben der Systeme Werte physikalischer Größen dar, die zu jedem Zeitpunkt einen bestimmten Wert haben. Zum Anderen betrachtet man eingebettete Systeme, die auf diskreten Steuergeräten abgelegt sind. Die Werte an den Kanälen sind für die Logik dieser Systeme nur an diskreten Zeitpunkten relevant. Die physikalische Funktionalität der Steuergeräte wird hierbei abstrahiert, in dem für die kanalbeschreibenden Funktionen Konstanten verwendet werden, die sich zu den jeweiligen Taktzyklen ändern können (siehe [LSV01]). Auf der Verhaltensebene (siehe Abschnitt 3.1) ist für diese Systeme die Definitionsmenge D_f der Funktionen, welche die Kanalwerte darstellen, diskret. Sie entspricht der Vielfachenmenge $\mathbb{V}_0^+(t_s)$, wobei t_s eine Taktlänge ist (siehe Abbildung 3.3). Die Folge der Kanalwerte zu den diskreten Zeitpunkten, also die Funktion f_i kann als ein Strom x von Werten dargestellt werden (siehe Definition 3.2.1 aus [Bre01], [BS01]). Auf der Verhaltensebene haben in dieser Arbeit alle diskreten Systeme eine systemweit einheitliche (schnellste) Taktung mit einer systemweit einheitlichen Zeit. Eine langsamere Taktung muss für die Systeme explizit modelliert werden. Auf das Zusammenspiel zwischen kontinuierlichen Funktionen und zeitdiskreten Funktionen wird in Abschnitt 3.3.5 weiter eingegangen.

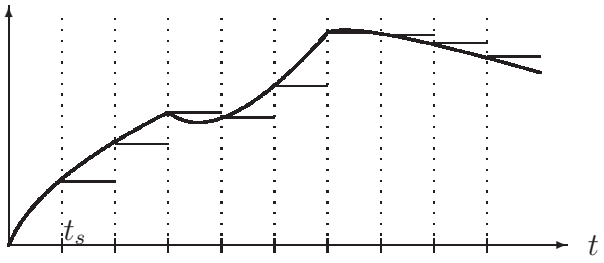


Abb. 3.3: Beispiel-Diskretisierung einer Funktion

[7] Der Begriff *System* wird in dieser Arbeit als Synonym für den Begriff *Komponente* verwendet. Im Sprachgebrauch sind Komponenten meist Teile eines Gesamtsystems.

Definition 3.2.1 (Strom)
Sei M eine Menge von Werten. Ein *Strom* x über M ist eine endliche oder unendliche Sequenz von Elementen aus M.

Die Menge der endlichen Ströme wird mit M^*, die der unendlichen Ströme mit M^∞ bezeichnet. Die Menge aller Ströme über M ist $M^\omega = M^* \cup M^\infty$.

Der Ausdruck $\langle d_1, d_2, \ldots, d_n \rangle$ bezeichnet einen endlichen Strom der Länge n mit $d_1, d_2, \ldots, d_n \in M$
$\langle\rangle$ ist der leere Strom. ⌟

Stromverarbeitung

Die diskreten Zeit- und Werteigenschaften der Kanäle und die damit verbundene Interpretation der Belegungen als Nachrichten betonen besonders den Aspekt der Interaktion zwischen den Systemen und den Aspekt der Verarbeitung der Nachrichten. Es bieten sich zusätzlich zu den mathematischen Operatoren $+, -, \cdot, /, ...$, welche auf Elemente von Strömen angewendet werden, eine Menge stromverarbeitender Operationen an, um mit den Nachrichten umzugehen (siehe [Bre01], [BS01]).

Definition 3.2.2 (Stromverarbeitende Operatoren)
$\#x$ bezeichnet die Länge des Stroms x. Für unendliche Ströme x gilt: $\#x = \infty$.

$x.t$ bezeichnet den t-ten Wert eines Stroms x falls $\#x \geq t$.

$x \frown y$ bezeichnet die Konkatenation von Strömen, d.h. $x = \langle x_1, \ldots, x_n \rangle, \quad y = \langle y_1, \ldots, y_m \rangle$,
$x \frown y = \langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle$.
Für $\#x = \infty$ gilt: $x \frown y = x$.

Bezogen auf eine exemplarische Kanalfunktion kann f_i mit einem Strom x beschrieben werden, wobei der Wert $f_i(t)$ dem Wert $x.t$ entspricht. ⌟

In [BDD+92] finden sich weitere ausführliche Beschreibungen von Operatoren und Definitionen. Da die Beschreibung der diskreten Systeme nicht im Fokus dieser Arbeit steht, wird hier auf die Quelle [BS01] verwiesen. Die Darstellung der hier genannten Operationen dient der Verdeutlichung des Charakters der diskreten Kommunikation der Systeme.

Formeln

Die Definition von Operatoren zeigt bereits, dass die Relationen, die das Verhalten eines Systems anhand der zugehörigen Ein- und Ausgabefunktionen repräsentieren, mit Formeln realisiert werden. Hierzu ist es notwendig, die dort verwendeten Begriffe zu definieren:

Definition 3.2.3 (Formel & Term)
Die Menge aller Variablen wird mit *VAR* benannt. Jeder Variable $v \in VAR$ wird ein Typ zugewiesen, der den maximalen Wertebereich beschreibt.

Eine *Belegung* α weist jeder Variablen einen Wert $\alpha.v$ zu. *VAL* bezeichnet die Menge aller Belegungen.

Zwei Belegungen stimmen für eine Variablenmenge $V \subset VAR$ überein, wenn sie allen Variablen der Menge V den gleichen Wert geben:

$$\alpha \overset{V}{=} \beta \overset{def}{=} \forall v \in V : \alpha.v = \beta.v$$

Ein Term τ ist eine Abbildung, die sich mit einer Belegung α zu einem Wert aus der Wertemenge W_τ des Terms auswerten lässt.

Eine Formel Φ ist eine Abbildung (und Sonderfall eines Terms), die sich mit einer Belegung α zu einem booleschen Wert $\{true, false\}$ auswerten lässt.

Die Auswertung einer Formel Φ mit einer Belegung α zu $true$ wird als $\alpha \vDash \Phi$ notiert.

$\Phi[w/v]$ beschreibt die Formel Φ, in der die freie Variable $v \in VAR$ durch die Variable w ersetzt wird.

$\Phi[\Phi_j/\Phi_i]$ beschreibt für eine Formel der Form $\Phi = (\Phi_1 \wedge \Phi_2 \wedge ...)$, die Formel, in der die Teilformel Φ_i durch die Teilformel Φ_j ersetzt wird.

$free(\Phi)$ bezeichnet die Menge der freien Variablen in Φ. ⌟

Bindung

Mit Hilfe dieser Definitionen lässt sich eine Formel Φ an ein System S binden in dem in der Formel die freien Variablen v_i durch die Kanäle $f, g, \ldots$ ersetzt werden, also $\Phi[f/v_1][g/v_2]\ldots$ gilt. Als Belegungen der Variablen können dann entweder die Wertverläufe oder die aktuellen Werte der Kanäle des Systems verwendet werden. Eine genaue Beschreibung der Bindung der Formeln an ein System ist in Abschnitt 3.3.1 zu finden.

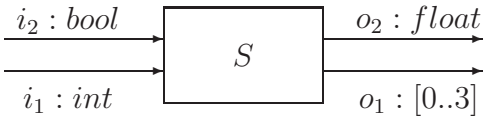
Definition 3.2.4 (Schnittstelle)
Eine Schnittstelle eines Systems S ist ein Tupel (I_S, O_S), wobei die endliche Menge $I_S \subseteq VAR$ die Eingabe- und die endliche Menge $O_S \subseteq VAR$ die Ausgabekanäle des Systems S sind. ⌟

Die in dieser Arbeit betrachteten Schnittstellen eines Systems sind statisch, d.h. eine Schnittstelle ändert sich zur Laufzeit nicht. Systeme mit Schnittstellen, die sich zur Laufzeit ändern, werden u.a. in [GSB97] betrachtet. Eine Darstellung von Schnittstellen ist in in Abbildung 3.4 gegeben.

Definition 3.2.5 (Verhaltensrelation)
Die Verhaltensrelation ist eine Relation R_S eines Systems S, welche n-Tupel zwischen den exemplarischen Eingabe- und Ausgabekommunikationen enthält. Es gilt:

$$R_S \subseteq \mathcal{F}_{I_S} \times \mathcal{F}_{O_S}$$

Hierbei entspricht $\mathcal{F}_{I_S}$ dem Tupel der dynamischen Datentypen der Eingabekanäle, also
$\mathcal{F}_{I_S} = \mathcal{F}_f \times \mathcal{F}_g \times \ldots$ mit $f, g, \ldots \in I_S$ und $\mathcal{F}_{O_S}$ dem der Ausgabekanäle.



Name	DatenTyp	Art
i_1	int	input
i_2	bool	input
o_1	[0 .. 3]	output
o_2	float	output

Abb. 3.4: Darstellung der Schnittstelle eines Systems

Die Menge aller Verhalten zu einem System S ist:

$$REL_S = \mathcal{P}(\mathcal{F}_{I_S} \times \mathcal{F}_{O_S}).$$

$dom.R_S$ („Domain R“) ist die Menge der Eingabe-Tupel, die in der Relation R_S enthalten sind.
$rng.R_S$ („Range R“) ist die Menge der Ausgabe-Tupel, die in der Relation R_S enthalten sind.
(Definitions- und Wertebereich von R_S). ⌟

Das Verhalten eines Systems S wird durch eine Verhaltensrelation R_S definiert. Die Relation R_S ist im unendlichen und im kontinuierlichen Fall nur durch Hilfsmittel wie Formeln beschreibbar. Unabhängig von der Beschreibbarkeit ist die Nutzung der Formeln üblich, da die Beschreibungen deutlich effizienter und lesbarer durchgeführt werden können. Die Beschreibung der Relation R_S mit Formeln ist in Abschnitt 3.3 erklärt.

Realisierbarkeit

Prinzipiell lassen sich mit diesen Relationen auch Verhaltensweisen beschreiben, die von Systemen nicht realisierbar sind. So können z.B. Verhalten beschrieben werden, bei denen die Systeme auf Eingaben reagieren, die in der Zukunft stattfinden. Ein reales System kann aber nur auf Eingaben reagieren, die bereits stattgefunden haben. In [BS01] wird gezeigt, dass für ein System die Realisierbarkeit im Wesentlichen nur durch eine fehlende Kausalität und bzw. oder durch eine Verletzung der Linkstotalität nicht sichergestellt werden kann. Ein linkstotales System zeigt für jede Eingabe der Umgebung eine Reaktion. Bei einem kausal monotonen System kann ein ausgegebener Wert nicht mehr zurückgenommen werden. Bei diskreten Systemen haben zwei Eingabeströme mit dem selben Präfix auch zwei Ausgabeströme mit dem selben Präfix zur Folge.

Definition 3.2.6 (kausale Monotonie)
Ein System S ist *kausal monoton*, wenn eine Verlängerung der Eingabe i zu i' immer zur Folge hat, dass die Ausgabe entweder unverändert bleibt, oder sich auch verlängert. Es gilt:

$$\forall i, i' \in \mathcal{F}_{I_S}, o, o' \in \mathcal{F}_{O_S} : ((i, o) \in R_S \wedge i \sqsubseteq i' \wedge (i', o') \in R_S) \Rightarrow o \sqsubseteq o' \quad ⌟$$

Definition 3.2.7 (Linkstotalität)
Ein System S ist *linkstotal*, wenn die verhaltensbeschreibende Relation R_S für jede mögliche Eingabe i eine Ausgabe o aufweisen kann. Es gilt:
$\forall i \in \mathcal{F}_{I_S} : \exists o \in \mathcal{F}_{O_S} : (i, o) \in R_S$ oder kurz: $dom.R_S = \mathcal{F}_{I_S}$ ⌟

Diese Eigenschaften von Systemen sind in vielen Beschreibungstechniken implizit vorhanden oder werden von Werkzeugen explizit gefordert. So ist z. B. in dem Programm Simulink, wie auch im Programm Stateflow diese Eigenschaft implizit vorhanden, denn die Ausgaben können nur unabhängig von den Eingaben, oder in Abhängigkeit der bereits vorhandenen Eingaben stattfinden.

Laufzeitbedingungen

Die Beschreibung des Verhaltens eines Systems ist von der Umwelt unabhängig. Ein System wird in eine Systemlandschaft integriert und es treten in den meisten Fällen nicht alle Eingabekombinationen auf, die für das System über die Schnittstellen möglich wären. Ein Hilfsmittel bei der Spezifikation eines Systems für einen konkreten Einsatz in einer Systemumgebung ist die Angabe von Bedingungen, die im Einsatz erfüllt sein müssen. Mit dieser Angabe von Bedingungen wird die Menge der Elemente der verhaltensbeschreibenden Relation so reduziert, dass das System nur noch für die in dem konkreten Einsatz vorkommenden Pfade linkstotal ist. Diese Bedingungen werden *Laufzeitbedingungen* genannt. Dieses bedingte und somit tatsächlich genutzte Verhalten wird als R_S^{assert} bezeichnet, mit $R_S^{assert} \subseteq R_S$. Ein Beispiel zu Laufzeitbedingungen wird in Abschnitt 3.3.1 gezeigt.

3.2.2 Modellierung der Implementierungsstruktur

Zweck

Dieser Abschnitt beschreibt, wie Modelle der Artefaktestruktur aussehen können, in denen das im vorherigen Abschnitt 3.2.1 spezifizierte Verhalten implementiert ist. Diese Modelle dienen nicht der Modellierung des Verhaltens. Sie dienen der Zuordnung spezifischer Eigenschaften der Implementierungselemente zu den Verhaltens-Spezifikationen. Schwerpunkte bei diesen Modellen sind die Struktur zwischen den Elementen und die Attribute der Elemente. Von diesen Strukturen und Attributen können potenzielle Fehler und Fehlverhalten systematisch abgeleitet werden. So kann zum Beispiel ein CAN-Bus bis zu einem gewissen Grad eine Abschirmung gegen elektromagnetische Strahlungen haben. Ein davon abgeleiteter Fehler wäre die elektromagnetische Störung, die keine Übertragung zulässt.

Klassifizierung

Die Implementierungsmodelle sind entsprechend dem Design der Systeme verschieden, können aber anhand gemeinsamer Eigenschaften klassifiziert werden. Die gemeinsamen Eigenschaften der Klassen werden in dieser Arbeit mit Klassendiagrammen der Unified Modelling Language (UML) definiert (siehe [Fow03], [ISO05]). Ein Implementierungsmodell ist dann eine Instanz (UML-Objektmodell) des jeweiligen Klassendiagramms. Hierbei ist $\mathbb{C}$ die Menge der Klassen, $\mathbb{I}$ die Menge der Objekte (bzw. Instanzen), $CI \subseteq \mathbb{C} \times \mathbb{I}$ die Relation zwischen den Klassen und den Instanzen und $II \subseteq \mathbb{I} \times \mathbb{I}$ die Relation zwischen den Instanzen bzw. den miteinander verbundenen Objekten.

Faktoren für die Klassifizierung

Die Klassen der Implementierungsmodelle sind spezifisch auf die jeweiligen Verhaltensmodelle ausgerichtet. Sie hängen von der Abstraktionsebene des Verhaltensmodells ab, von der Domäne, die modelliert wird, von dem Grad der Zerlegung der Funktionsarchitektur, von dem Architekturkonzept und von den konkreten physikalischen Eigenschaften. Im Folgenden werden die beeinflussenden Faktoren anhand von Beispielen verdeutlicht:

- *Abstraktionsebene des Verhaltensmodells:* Ein Verhaltensmodell, welches die Datentypen abstrahiert, hat ein Implementierungsmodell, bei dem eine verhaltenstreue Umsetzung der Datentypen eine Eigenschaft ist. Ist hingegen in dem Verhaltensmodell der Datentyp bereits exakt vorgegeben, so ist diese Eigenschaft im Implementierungsmodell nicht gegeben.
- *modellierte Domäne:* Das Verhalten der Domäne Elektronik wird in elektronischen Bauteilen implementiert. Entsprechend wird das Verhalten anderer Domänen in Bauteilen anderer Domänen implementiert. Die Domäne beeinflusst so die Artefakte, die modelliert werden.
- *Granularität der Komponenten:* Entsprechen die kleinsten Systeme des Verhaltensmodells der Elektronik, so ist es nicht notwendig, im Implementierungsmodell die Struktur der Steuergeräte und Busse darzustellen. Sind hingegen die kleinsten Systeme einzelne Steuergeräte, so ist deren Verbindung über Busse relevant.
- *Architekturkonzept:* Das Verhalten der Systeme kann mit verschiedenen Architekturkonzepten implementiert werden. Ein Beispiel ist eine Implementierung der Elektronik nach dem AUTOSAR-Konzept [AUT06] oder nach dem in Abbildung 3.5 aus einem Projekt erhobenen Modell.
- *konkrete physikalische Eigenschaften:* Findet die Übertragung von Daten in einem Bus-System elektrisch statt, so spielt bei den Leitungen die Empfindlichkeit gegenüber elektromagnetischer Strahlung eine Rolle. Bei optischer Übertragung hingegen hat die elektromagnetische Strahlung keinen Einfluss auf die Leitungen.

Abbildung Verhalten zur Implementierung

Die Elemente der Verhaltensmodelle werden denen der Implementierungselemente zugeordnet. Sei $\mathbb{S}$ die Menge der Systeme der Verhaltensmodelle und $\mathbb{I}$ die Menge der Implementierungselemente. Die Zuordnung der Systeme der Verhaltensmodelle zu den Implementierungselementen ist eine Relation $SI \subseteq \mathbb{S} \times \mathbb{I}$. Ein Implementierungsmodell ist für ein Verhaltensmodell geeignet, wenn jedes Blatt der Systemhierarchie einem Implementierungselement zugewiesen werden kann. Jedem System kann so eine Menge potentieller Eigenschaften und somit potentieller Fehler zugewiesen werden.

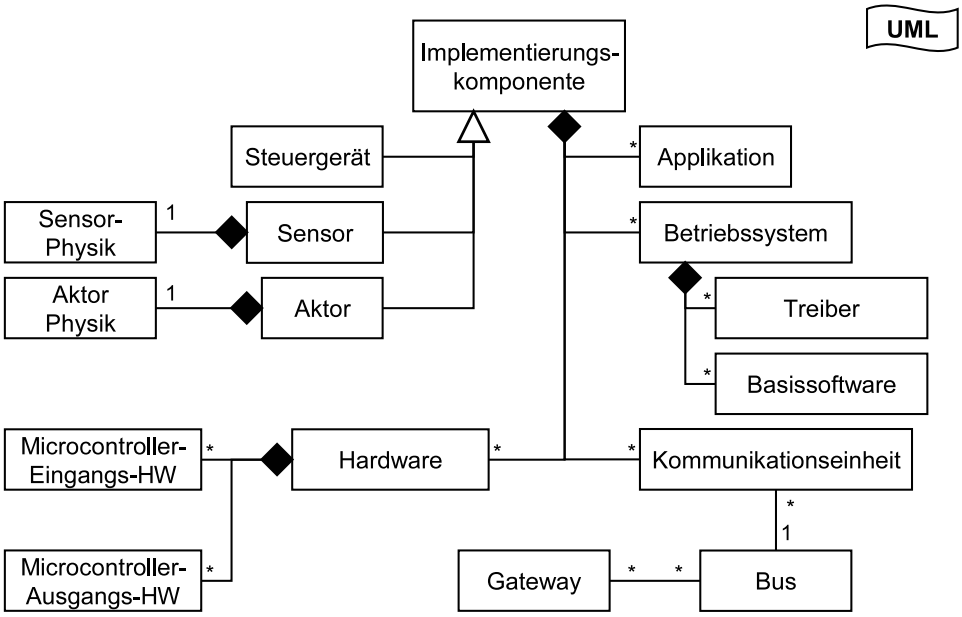


Abb. 3.5: Beispiel eines Implementierungs-Klassendiagramms (vgl. [BSN07])

Eigenschaften der Elemente

Den Implementierungselementen werden Eigenschaften zugewiesen, anhand derer potentielle Fehler abgeleitet werden können. Diese Eigenschaften sind Qualitätseigenschaften. Sie umfassen im Groben: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. Eine detailliertere Referenzauflistung dieser Eigenschaften ist in der Norm ISO 9126 ([ISO01]) zu finden. Betrachtet man zum Beispiel ein Steuergerät, so gehört zur Zuverlässigkeit unter anderem eine gewisse Robustheit gegen thermische Belastung, elektromagnetische Strahlung und mechanische Erschütterungen.

Zusammenfassung

Zusammenfassend enthalten die Implementierungsmodelle die Struktur zwischen den Implementierungselementen und die qualitativen Eigenschaften dieser Elemente. Auf eine genauere Beschreibung der Ausprägungen der Implementierungsmodelle wird hier verzichtet, da der Schwerpunkt dieser Arbeit auf der Analyse des Verhaltens liegt. Grundlegend können von Implementierungsmodellen potenzielle Fehler anhand der qualitativen Eigenschaften abgeleitet werden, deren Wirkung entlang der Struktur ermittelt werden und schließlich deren Erscheinen im Verhaltensmodell über die Abbildung *SI* verfolgt werden.

3.3 Modellierungstechniken

Spezifikation des Verhaltens

Abschnitt 3.2.1 hat gezeigt, wie mit Relationen zwischen kanalbeschreibenden Funktionen das Verhalten von Systemen modelliert werden kann. Aufbauend darauf werden hier Modellierungstechniken angegeben, mit denen diese Relationen spezifiziert

werden können. Die Techniken sind teilweise auf die zu modellierenden Aspekte zugeschnitten. Ausgehend von den allgemeinen Spezifikationen mit Formeln werden darauf aufbauend Sonderformen der Formeln betrachtet. Für die Regelungssicht werden zu den Ausgabekanälen aufgelöste Gleichungen vorgeschlagen. Für die Interaktionssicht werden Automaten vorgeschlagen. Diese beiden Sichten können mit Betriebsmodi-Automaten kombiniert werden. Letztlich wird diskutiert, ob und wie mit diesen Modellierungstechniken die komplexe physikalische Umgebung der Systeme geeignet abstrahiert werden kann.

3.3.1 Blackbox Spezifikationen

Prädikate

Eine *Black-Box Spezifikation* beschreibt das Verhalten eines Systems S mit einem Prädikat. Ein *Prädikat* ist eine Formel Φ, deren freien Variablen Eingabe- und Ausgabekanäle des Systems S sind und genau die Tupel exemplarischer Eingaben und Ausgaben von S in R_S sind, für die Φ wahr wird.

Bindung der Variablen

Die Prädikate nutzen Variablen zur Charakterisierung der Tupel. Hier gibt es zwei Möglichkeiten bei der Bindung der Variablen an die Kanäle. Die erste Möglichkeit ist, die Variablen der Formeln mit exemplarischen Kommunikationen an den jeweiligen Kanälen zu belegen. Diese Art der Blackbox-Spezifikation mit stromgebundenen Formeln ist in [Bre01] ausgeführt. Es gibt verschiedene Formeln, die das Verhalten R_S beschreiben. $PRED_S^{Stream}$ ist die Menge aller Formeln, die das Verhalten eines Systems S mit exemplarischen Kommunikationen beschreiben.

Definition 3.3.1 (stromgebundenes Prädikat)
Ein *stromgebundenes Prädikat* ist eine Formel $[\![S]\!] \in PRED_S^{Stream}$ aus der Menge:
$PRED_S^{Stream} \stackrel{def}{=} \{\Phi | (free(\Phi) \subseteq (I_S \cup O_S)) \wedge (\forall \alpha \in \mathcal{F}_{I_S} \times \mathcal{F}_{O_S} : (\alpha \vDash \Phi) \Leftrightarrow (\alpha \in R_S))\}$
Die Prädikatenmengen-Menge für die Relationen R_S von S heißt $PRED_{REL_S}^{Stream}$. ⌟

operative Bindung

Die zweite Möglichkeit der Variablenbindung ist, die freien Variablen der Formeln mit Werten der Kanäle in Bezug zu einem Zeitpunkt t zu belegen. In diesem Fall erfüllt eine Kombination von Strömen $x_1, x_2, ...$ an den jeweiligen Kanälen das Prädikat, wenn zu jedem Zeitpunkt t die Formel für die Werte $x_1.t, x_2.t, ...$ erfüllt ist. Die breitere Schrift in der folgenden Formel, weist darauf hin, dass hier keine Auswertung stattfindet, sondern das Ergebnis der breiteren Schrift ein Term oder eine Formel ist.

Definition 3.3.2 (operationelles Prädikat)
Eine Formel $[\![S]\!]^{op} \in PRED_S^{op}$ aus der Menge:
$PRED_S^{op} \stackrel{def}{=}$
$\{\Phi | (\boldsymbol{\forall t \geq 0 : \Phi[\Phi^t_{i_1}/\Phi_{i_1}] \ldots [\Phi^t_{i_n}/\Phi_{i_n}][\Phi^t_{o_1}/\Phi_{o_1}] \ldots [\Phi^t_{o_m}/\Phi_{o_m}]}) \in PRED_S^{Stream})\}$
mit $\{i_1, ...i_n\} = I_S$ und $\{o_1, ...o_m\} = O_S$
und $\forall 1 \leq a \leq n : \Phi_{i_a} = (\boldsymbol{i_a}) \wedge \Phi^t_{i_a} = (\boldsymbol{i_a.t})$
und $\forall 1 \leq a \leq m : \Phi_{o_a} = (\boldsymbol{o_a}) \wedge \Phi^t_{o_a} = (\boldsymbol{o_a.t})$ ⌟

Definition 3.3.3 (Operationalisierung)
Eine Abbildung von einem operationellen Prädikat Φ auf ein strombasiertes Prädikat:
$\Phi[\uparrow \mathbb{T} \uparrow] : PRED_S^{op} \times \mathcal{P}(\mathbb{N}) \rightarrow PRED_S^{stream}$
$\Phi[\uparrow \mathbb{T} \uparrow] \stackrel{def}{=} (\forall \boldsymbol{t} \in \mathbb{T} : \boldsymbol{\Phi[\Phi^t_{i_1}/\Phi_{i_1}]\ldots[\Phi^t_{i_n}/\Phi_{i_n}][\Phi^t_{o_1}/\Phi_{o_1}]\ldots[\Phi^t_{o_m}/\Phi_{o_m}]})$
mit $\{i_1, ...i_n\} = I_S$ und $\{o_1, ...o_m\} = O_S$
und $\forall 1 \leq a \leq n : \Phi_{i_a} = (\boldsymbol{i_a}) \wedge \Phi^t_{i_a} = (\boldsymbol{i_a.t})$
und $\forall 1 \leq a \leq m : \Phi_{o_a} = (\boldsymbol{o_a}) \wedge \Phi^t_{o_a} = (\boldsymbol{o_a.t})$ ⌟

Teilformeln

Ist ein System S mit einer Blackbox-Spezifikation definiert, so kann das gegebene Prädikat $[\![S]\!]$ die *konjunktive Form*

$$[\![S]\!] = \bigwedge_{i \in [1..n]} (\Phi_i)$$

haben. Die Menge aller Prädikate mit konjunktiver Form zu einem System S ist: $PRED(\bigwedge)^{Stream}_{REL_S}$. Die einzelnen Formeln Φ_i entsprechen Verhaltensbedingungen (siehe Abschnitt 2.3) oder anderen Systembeschreibungen. Sie können als Teilprädikate verstanden werden, die das System aus Sicht der jeweiligen Formel beschreiben. Diese Zerlegung kann die Verständlichkeit der Spezifikation erhöhen, in dem Eigenschaften explizit hervorgehoben werden. Teilformeln sind unter anderem dazu geeignet, um sich nur auf eine Teilmenge der Eingabe- und Ausgabekanäle zu beziehen. Eine besondere Form sind Formeln, die sich nur auf gewisse Eingabebedingungen beziehen, also partiell gültig sind. Diese Prädikate werden auch *Assumption / Guarantee-Spezifikationen* genannt und haben die Form

$$\Phi^{Assumption} \Rightarrow \Phi^{Guarantee}.$$

Das folgende Beispiel zeigt, wie ein System mit Blackbox-Spezifikationen im Asumption-Guarantee-Stil definiert werden kann.

Beispiel 3.3.1 (Black-Box-Spezifikation eines Systems)
Bei abgeschalteter Überlagerungslenkung verhält sich das Lenksystem S wie eine herkömmliche Lenkung. Es gilt dann ein Prädikat $[\![S]\!]^{op} = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3)$ mit folgenden Teilformeln:

Die Übersetzung vom Lenkradwinkel (Eingabe i_{lrw}) zum Vorderradwinkel (Ausgabe o_{vrw}) hat im Gültigkeitsbereich $[-12rad..12rad]$ den konstanten Wert 0.05:

$$\Phi_3 = ((-12rad \leq i_{lrw} \leq 12rad) \Rightarrow (o_{vrw} = 0.05 * i_{lrw}))$$

Der Betrag des Vorderradwinkels ist bei allen Lenkradwinkeln, deren Betrag größer als $12rad$ im Bogenmaß ist, kleiner gleich $0.6rad$, da das Rad sonst an den Radkasten stößt:

$$\Phi_1 = ((i_{lrw} < -12rad) \Rightarrow (o_{vrw} \leq -0.6rad))$$
$$\Phi_2 = ((i_{lrw} > 12rad) \Rightarrow (o_{vrw} \leq 0.6rad))$$ ⌟

Laufzeitbedingungen

Die Laufzeitbedingungen, die in einem System gelten, werden ebenfalls mit Formeln angegeben. Da sie nicht das Verhalten des Systems beschreiben, sondern Bedingungen an dessen Einsatz, werden sie gesondert betrachtet. Die Formeln $[\![S]\!]_{Assertion}$

werden nur dann betrachtet, wenn für das System eine Konsistenzprüfung in seiner Systemlandschaft gemacht wird.

Beispiel 3.3.2 (Einsatz von Laufzeitbedingungen)
Das in vorherigem Beispiel spezifizierte Lenksystem S mit abgeschalteter Überlagerungslenkung kann wesentlich einfacher spezifiziert werden, wenn man annimmt, dass die Eingaben der Umwelt eingegrenzt sind. Es gelten folgende Aussagen:

Die Übersetzung vom Lenkradwinkel zum Vorderradwinkel hat den konstanten Wert 0.05:

$$\Phi_1 = (o_{vrw} = 0.05 * i_{lrw})$$

Alle Ausführungen münden in einem Vorderradwinkel kleiner / gleich $0.6rad$:

$$\Phi_2 = (|o_{vrw}| \leq 0.6rad)$$

Für ein System S gilt dann $[\![S]\!]^{op} = \Phi_1$
und die Laufzeitbedingung $[\![S]\!]^{op}_{Assertion} = \Phi_2$. ⌟

Bezug zur Zeit

Abschließend wird in diesem Abschnitt ein Operator eingeführt, um mit einem operationellen Prädikat algorithmische Aufgaben definieren zu können. Ein System greift auf die aktuelle Belegung zu einem Zeitpunkt t, aber auch auf Belegungen der Vergangenheit zu Zeitpunkten $(t-a)$ zurück, wenn es sich diese gemerkt hat. Bei einer Beschreibung eines diskreten Systems S mit einem Prädikat $[\![S]\!]^{op}$ muss dementsprechend die Möglichkeit bestehen, sich auf vergangene Werte zu beziehen. Die Realisierung dieses Zugriffs geschieht üblicherweise über einen delay()-Operator. Dieser Operator verhält sich wie ein Puffer (FIFO-Stack), der die eingehenden Werte speichert und nach einer angegebenen Zeit wieder freigibt. Viele Modellierungssprachen bieten diesen Operator nur für die Zeit eines Berechnungsschrittes an, wobei durch Verschachtelung der Operatoren längere Verzögerungen formulierbar sind. Weiter fordern sie die Angabe eines Initialwertes, der die Ausgabe des Operators beim ersten Rechenschritt bestimmt[8]. Formal ist der Operator wie folgend definiert:

Definition 3.3.4 (Delay-Operator)
Der *Delay-Operator* zur Referenzierung vergangener Werte ist eine Konkatenation des Initialstroms $\langle a_1, ..., a_n \rangle$ mit dem zu betrachtenden Strom x:

$$\mathsf{delay}_{\langle a_1,...,a_n \rangle}(x.t) \stackrel{def}{=} (\langle a_1, ..., a_n \rangle \frown x).t$$ ⌟

Möchte man verschiedene Prädikate zu bestimmten Systemzeiten gelten lassen, so lässt sich dies mit einem künstlich angelegten Zählerkanal realisieren, der in jedem Rechenschritt inkrementiert wird.

[8] Die Sprache Lustre bietet einen *pre*-Operator, der den Wert eines Ausdrucks im vorhergehenden Berechnungsschritt angibt und einen „→“-Operator zur Initialisierung im ersten Schritt. (siehe [HCRP91])

3.3.2 Spezifikation mit zu Ausgabekanälen aufgelösten Gleichungen

mathematische Formeln

Die Beschreibung mechanischer bzw. elektrischer Systeme, sowie ein Großteil der Beschreibung der physikalischen Prozesse, die von eingebetteten Systemen gesteuert werden, entsprechen typischerweise den Eigenschaften der Regelungssicht (Abschnitt 3.1) mit dem Schwerpunkt auf dem Datenfluss. Die datenflussorientierte Beschreibung eines Verhaltens dieser Systeme kann mit mathematischen Gleichungen realisiert werden, die zu den jeweiligen ausgabekanalbeschreibenden Variablen aufgelöst sind. Für eine Ausgabekanalmenge $o_j \subseteq O_S$ entsteht so eine Formel der Form:

Definition 3.3.5 (aufgelöste Gleichung)
Ein Verhalten für eine Ausgabe $o_j \subseteq O_S$ in einem Prädikat $[\![S]\!]^{op}$ ist gegeben durch eine zu o_j aufgelöste Gleichung der Form:

$$\Phi_{o_j} \stackrel{def}{=} (\boldsymbol{o_j = \tau_{o_j}})$$

mit $o_j \subseteq O_S$
und $\mathit{free}(\tau_{o_j}) \cap o_j = \emptyset$
und mit *free* und τ gemäß Definition 3.2.3

Die Menge aller zu einer Ausgabe o_j hin aufgelösten Gleichungen Φ_{o_j} in einem System S ist:

$$PRED^{op}_{REL\_o_j}.$$ ⌟

Beispiel 3.3.3 (Spezifikation mit aufgelösten Gleichungen)
Das Verhalten der geschwindigkeitsabhängigen Lenkübersetzung des Lenksystems S aus Beispiel 3.1.1 wird folgendermaßen datenflussorientiert festgelegt:

Als Ausgabe liefert es den Wert des Vorderradwinkels o_{vrw}, welcher von der Geschwindigkeit i_v sowie dem Lenkradwinkel i_{lrw} abhängt. Das Verhalten wird über ein Ausgabeprädikat definiert, also $[\![S]\!] = \Phi_{o_{vrw}}$. Nach Definition 3.3.5 hat die Formel damit die Form $\Phi_{o_{vrw}} = (o_{vrw} = \tau_{o_{vrw}})$.

Der Term, der die Funktion zur Berechnung des Ausgabewertes beschreibt, ist:

$$\tau_{o_{vrw}} = (i_{lrw}/(16 \cdot \mathsf{max}(1, (|i_v|/100km/h)))$$ ⌟

Anwendbarkeit

Eine vollständige Beschreibung eines Systems anhand aufgelöster Gleichungen erhält man durch eine Konjunktion der jeweils dem Ausgabekanal zugeordneten Formeln, also $[\![S]\!]^{op} = \Phi_{o_1} \wedge \Phi_{o_2} \wedge ...\Phi_{o_n}$. Die Verwendung aufgelöster Gleichungen ist keine Garantie für ein ausführbares System. Die möglichen Fehlformulierungen können aber auf ein überschaubares Maß reduziert werden (z.B. Division durch Null). Die Gleichungen können in Textform, aber auch wie in Simulink (siehe Abbildung 3.6) graphisch dargestellt werden. Ebenso kann eine Untergliederung in Teilfunktionen stattfinden, um Redundanzen bei der Beschreibung zu vermindern.

Präzision

Die mathematischen Gleichungen liefern präzise Ergebnisse mit einer unendlichen Genauigkeit. In der Realität finden jedoch immer Abweichungen statt, die in einem Toleranzbereich liegen. Teilweise möchte man gewisse Ungenauigkeiten in den Formeln zulassen, um weiteren Gestaltungsfreiraum bei der Implementierung zu geben. Oft wird eine gewisse Ungenauigkeit der Beschreibungen implizit angenommen. So findet z.B. bei Simulink die Simulation über eine Diskretisierung der Zeit statt, die entsprechende Abweichungen von der Beschreibung mit Formeln mit sich bringt. Mathematische Gleichungen wie in Simulink können also ein ideales Verhalten beschreiben, sind aber für eine Spezifikation zu präzise und müssen aufgeweicht werden. Ein System kann in diesen Fällen mit „unscharfen“ Formeln beschrieben werden, in denen die Ausgaben mit einem Toleranzbereich versehen werden. Für eine Gleichung kann bei der Spezifikation eine obere und untere Schranke angegeben werden, welche die Toleranzgrenzen explizit beschreiben. Diese Angabe der Grenzen erlaubt es, den Toleranzbereich flexibel zu gestalten. Eine Formel Φ_x wird so zu einer Formel $\Phi_x^{upper} \wedge \Phi_x^{lower}$. Eine weitere Möglichkeit besteht in der Angabe eines Toleranzbereiches, in dem eine relative oder absolute Abweichung a akzeptabel ist. Aus Fallstudien haben sich folgende für diese Arbeit relevanten Toleranzangaben ergeben:

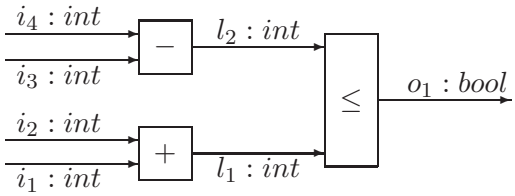


Abb. 3.6: Darstellung einer Funktion

Definition 3.3.6 (unscharfe Beschreibungen)
Unscharfe Beschreibungen einer Ausgabe o_j in einem Prädikat $[\![S]\!]^{op}$ sind gegeben durch:

$$\Phi_{o_j}^{\pm a} \stackrel{def}{=} (o_j \stackrel{<\pm a>}{=} \tau_{o_j}) \stackrel{def}{=} (\tau_{o_j} - a \leq o_j \leq \tau_{o_j} + a)$$

$$\Phi_{o_j}^{\div a} \stackrel{def}{=} (o_j \stackrel{<\div a>}{=} \tau_{o_j}) \stackrel{def}{=} (|\tau_{o_j} - o_j| \leq |\tau_{o_j} \cdot a|)$$

$$\Phi_{o_j}^{\div a \pm b} \stackrel{def}{=} \Phi_{o_j}^{\div a} \vee \Phi_{o_j}^{\pm b}$$

⌟

Ergänzung der Gleichungen

Die unscharfe Beschreibung des Systems kann um zusätzliche Prädikate ergänzt werden, die eine Anwendbarkeit der konkreten Funktion sicherstellen. Eine Forderung an eine Funktion ist z.B. die Stetigkeit, die für fast alle Steuerungen für Mechanik notwendig ist. Ebenso kann hinsichtlich der Steigung der Funktion angegeben werden, ob diese differenzierbar sein soll. Ein weiterer Punkt ist die Angabe der Monotonie, ob eine Funktion monoton steigend oder fallend sein soll, bzw. ob die Steigung einer Funktion monoton steigend oder fallend sein soll.

Beispiel 3.3.4 (Spezifikation mit unscharfen Gleichungen)
Das Lenksystem S mit geschwindigkeitsabhängiger Lenkübersetzung soll nicht zu stark von einem herkömmlichen Lenksystem mit konstanter Lenkübersetzung abweichen. Für das Übersetzungsverhältnis zwischen dem Lenkradwinkel (i_{lrw}) und

dem Vorderradwinkel (o_{vrw}) möchte man offen lassen, wie eine konkrete Implementierung der Lenkübersetzung aussieht (damit z.B. jeder Anbieter noch sein spezielles Wissen einbringen kann). Für das Lenksystem S gilt dann mit *free*, τ gemäß Definition 3.2.3 und delay gemäß Definition 3.3.4:

$$[\![S]\!]^{op}_{Assertion} = (|o_{vrw}| \leq 0.6) \text{ und } [\![S]\!]^{op} = (\Phi^{\pm 0.2}_{o_{vrw}} \wedge \Phi_1 \wedge \Phi_2)$$

mit:

Verhalten: $\tau_{o_{vrw}} = (0.05 \cdot i_{lrw})$

Differenzbegrenzung: $\Phi_1 = (|o_{vrw} - \mathsf{delay}_{\langle o_{vrw}.0 \rangle}(o_{vrw})| < 0.0002)$

Monotonie: $\Phi_2 = (\mathsf{sign}(i_{lrw} - \mathsf{delay}_{\langle i_{lrw}.0 \rangle}(i_{lrw})) = \mathsf{sign}(o_{vrw} - \mathsf{delay}_{\langle o_{vrw}.0 \rangle}(o_{vrw})))$ ⌟

3.3.3 Spezifikation mit Zustandsautomaten

schrittweises Verhalten

Das Verhalten einiger Komponenten ist algorithmischer Natur. Das heißt, die Komponenten führen Schritt für Schritt ihre Berechnungen durch. Zustandsautomaten bieten die Möglichkeit, ein Systemverhalten schrittweise zu beschreiben. Ein Schritt besteht aus einem Zustandsübergang (inklusive dem identischen Zustandsübergang), der durch bestimmte Werte oder Wertfolgen an den Eingabekanälen und einem Trigger-Signal ausgelöst wird. Ebenso können mit einem Zustandsübergang die Werte an den Ausgabekanälen geändert werden. Weitere Beschreibungen von Zustandsautomaten findet man in [Bre01] und [BS01]. Der unten definierte Automat ist speziell auf getaktete Rechnersysteme abgestimmt, bei der die Belegungen der Variablen die Werte an den Kanälen zu einem Zeitpunkt t sind.

Definition 3.3.7 (Zustandsautomat)
Ein *Zustandsautomat* A wird durch das 6-Tupel $A_S = (I, O, L, S_0, \delta, \Phi^{trigger})$ bestimmt.

Die Mengen I und O enthalten die Eingabe- und Ausgabekanäle (Variablen) des Systems. Die Menge L enthält die lokalen Variablen.

Der *Zustand* des Automaten A ist eine Belegung α aller Variablen der Menge $I \cup O \cup L$ mit den an den Kanälen anliegenden Werten (im Sinne der Bindung $[\![.]\!]^{op}$). Die nicht-leere Menge $S_0 \subseteq VAL$ enthält die Startzustände.

Die Relation $\delta \subseteq VAL \times VAL$ ist die Menge der *Transitionen* $(\alpha, \beta) \in \delta$, wobei α den Zustand vor und β den Zustand nach dem Zustandsübergang beschreibt.

Automaten sind *linkstotal*. Es gilt:

$$\forall \alpha, \beta, \gamma \in VAL : ((\alpha, \beta) \in \delta \wedge \gamma \overset{L \cup O}{=} \beta) \Rightarrow (\alpha, \gamma) \in \delta$$

Ein *Kontrollzustand* $k = (V_k, \beta^k)$ ist ein abstrakter Zustand, der sich auf die Belegung β^k einer Teilmenge $V_k \subset I \cup L \cup O$ bezieht und somit eine Menge von Zuständen umfasst. Es gilt:

$$k \overset{def}{=} \{\alpha | \alpha \overset{V_k}{=} \beta^k\}.$$

Ein *Datenzustand* ist ein Kontrollzustand $d = (V_d, \beta^d)$, der sich auf die lokalen und die Ausgabevariablen bezieht, also $V_d = L \cup O$.

Ein *Datenkontrollzustand*(dkz) ist ein Kontrollzustand $l = (V_l, \beta^l)$, der sich auf einen Teil der lokalen und der Ausgabevariablen bezieht, also $V_l \subseteq L \cup O$.

Eine *linkstotale Transition* $(\alpha, d) \subseteq \delta$ mit der Belegung α und dem Datenzustand d ist eine Menge von Transitionen, die Transitionen von α zu jedem Zustand aus d umfasst, also

$$(\alpha, d) = \{(\beta, \beta') | \beta = \alpha \wedge \beta' \in d\}$$

Eine *Datenkontrollzustand-Transition* $(k, l') \subseteq k \times l'$ ist eine Menge linkstotaler Transitionen mit dem Kontrollzustand $k = (V_k, \beta^k)$ und dem Datenkontrollzustand $l' = (V_{l'}, \beta^{l'})$, wobei gilt $V_{l'} \subseteq V_k$.

Ein Zustandsübergang findet statt, wenn es in δ mindestens ein Tupel (α, β) gibt, für das die Belegung α erfüllt ist. Die Belegungen der Variablen aus $L \cup O$ ändern sich dabei nicht, wenn nicht gleichzeitig die Formel $\Phi^{trigger}$ erfüllt ist. ⌟

Formeln

Die Menge der Zustände, die mit einem Kontrollzustand k zusammengefasst wird, kann durch eine Formel Φ_k beschrieben werden, die bei einer Auswertung zu 'wahr' einen gültigen Zustand bezeichnet. Die Menge der Startzustände S_0 kann ebenfalls mit einer Formel Φ_{S_0} beschrieben werden.

Die Relation δ enthält oft viele Elemente und wird so meist nicht direkt als Menge von Tupeln angegeben, sondern wird durch eine Formel

$$[\![\delta]\!] = (\Phi_1^\delta \vee \Phi_2^\delta \vee ...)$$

beschrieben. Die Teilformeln beschreiben jeweils eine Datenkontrollzustand-Transition und zeichnen sich dadurch aus, dass sie als freie Bezeichner nur Variablen aus der Belegung vor dem Zustandsübergang und Variablen aus der dem Zustandsübergang folgenden Belegung enthalten[9]. Sie beschreiben des Weiteren nur Mengen linkstotaler Transitionen[10]. Die Relation δ ist dann definiert als [11]:

$$\begin{aligned} \delta \quad = \quad & \{(\alpha, \beta) | (\alpha, \beta \vDash [\![\delta]\!]) \wedge (\alpha \vDash \Phi^{trigger})\} \\ & \cup \{(\alpha, \beta) | (\alpha \overset{L \cup O}{=} \beta) \wedge ((\alpha \vDash \neg \Phi^{trigger}) \vee (\nexists (\alpha, \gamma) \vDash [\![\delta]\!]))\} \end{aligned}$$

Blackbox-Konvertierung

Die Beschreibung der Automaten A_S kann in eine Blackbox-Spezifikation $[\![S]\!]^{op}$ umgewandelt werden. Hierbei werden die Variablenreferenzen v und v' durch $\mathsf{delay(v)}$ und v ersetzt. Es gilt $[\![S]\!]^{op} = [\![\delta]\!][\mathsf{delay}_{S_0}(V)/V][V/V']$, wobei V der Menge der freien

[9] Üblicherweise werden die dem Zustandsübergang folgenden Variablenbelegungen mit dem Zeichen ' versehen (also x und x').

[10] Die Beschränkung auf linkstotale Transitionen ist gegeben, wenn Formeln keine Variablenbelegung x' für eine Eingabevariable $x \in I$ referenzieren.

[11] Es ist entweder ein Triggerzeitpunkt und der Zustand ist in der Transitionsmenge enthalten (dann schaltet die beschriebene Transition), oder der Zustand ist nicht enthalten, bzw. es ist kein Triggerzeitpunkt (dann ändern sich nur die Eingaben).

Variablen entspricht und S_0 den Startzuständen. Einige Begriffe werden im weiteren Verlauf der Arbeit wegen dieser Konvertierbarkeit nicht explizit für Automaten, sondern nur für Blackbox-Spezifikationen beschrieben.

Trigger

Automaten dienen der Beschreibung von getakteten Systemen. Da alle Modelle einer einheitlichen schnellsten Taktung unterliegen, kann mit der Angabe der Formel $\Phi^{trigger}$ eine langsamere spezielle Taktung angegeben werden. Diese langsamere Taktrate wird durch Überprüfen eines von außen kommenden *Trigger*- oder Takt-Signals realisiert, welches dann entsprechend ausgewertet wird. Findet in einem Takt keine Transition statt, so bleiben die Werte der lokalen Variablen und der Ausgabevariablen gleich (und somit die Werte der Kanäle des Systems).

Konsistenz

Mit Formeln beschriebene Automaten sind immer linkstotale und kausale Systeme. Die Linkstotalität ergibt sich daraus, dass die Formeln nur linkstotale Transitionen beschreiben und die fehlenden Transitionen, die ein Akzeptieren jeder Eingabe zulassen, automatisch ergänzt werden. Das System zeigt also für jede Eingabe ein Verhalten. Ebenso ist die Kausalität gegeben, da die einzelnen Transitionen immer nur den aktuellen Zustand und den Folgezustand betreffen. Eine weitere Eigenschaft bei Automaten ist die Schaltbereitschaft. Ein Automat A ist schaltbereit für eine Belegung α, wenn er von dieser Belegung zu einer anderen Belegung β eine Transition ausführen kann.

Darstellung

Automaten eignen sich besonders zur Modellierung algorithmischer Abläufe. Die Beschreibung des Verhaltens mit Handlungsschritten erlaubt die anschauliche Modellierung (siehe Abbildung 3.7) komplexer Interaktionen zwischen den Systemen. Des Weiteren können die Zustände und Transitionen der Automaten formal in einer zu Black-Box-Konvertierbaren Darstellung gefasst werden, wie in folgendem Beispiel gezeigt wird.

Beispiel 3.3.5 (Spezifikation mit Automaten)
Die Stabilisierungsfunktion der Raddrehzahlsteuerung (RDS) und die Stabilisierungsfunktion der Überlagerungslenkung (UL) kommunizieren miteinander. Die Zustände *active* bedeuten, dass die Systeme jeweils gerade eingreifen. Die Zustände *single* und *kombi* bedeuten, das die (eingeschaltete) RDS gerade die UL berücksichtigt (*kombi*) oder nicht (*single*). Die UL kann entweder eingeschaltet sein (*on*), abgeschaltet sein (*off*) oder auf die Bestätigung der RDS warten(*wait*), dass diese im Zustand *kombi* ist. Die Überlagerungslenkung soll nur dann eingeschaltet werden können, wenn die RDS im Kombi-Modus ist. Die Überlagerungslenkung kann mit dem Schalter *sw* ein bzw. ausgeschaltet werden.

Dieses in Abbildung 3.7 dargestellte Verhalten der Systeme wird in Formeln mit zwei Automaten dargestellt:
$A_{RDS} = (I_{RDS}, O_{RDS}, L_{RDS}, S_{RDS_0}, \delta_{RDS}, \Phi_{RDS}^{trigger})$ und
$A_{UL} = (I_{UL}, O_{UL}, L_{UL}, S_{UL_0}, \delta_{UL}, \Phi_{UL}^{trigger})$

Die einzelnen Elemente sind wie folgt definiert:
$I_{RDS} = \{req : bool, act : bool\}$
$O_{RDS} = \{conf : bool\}$

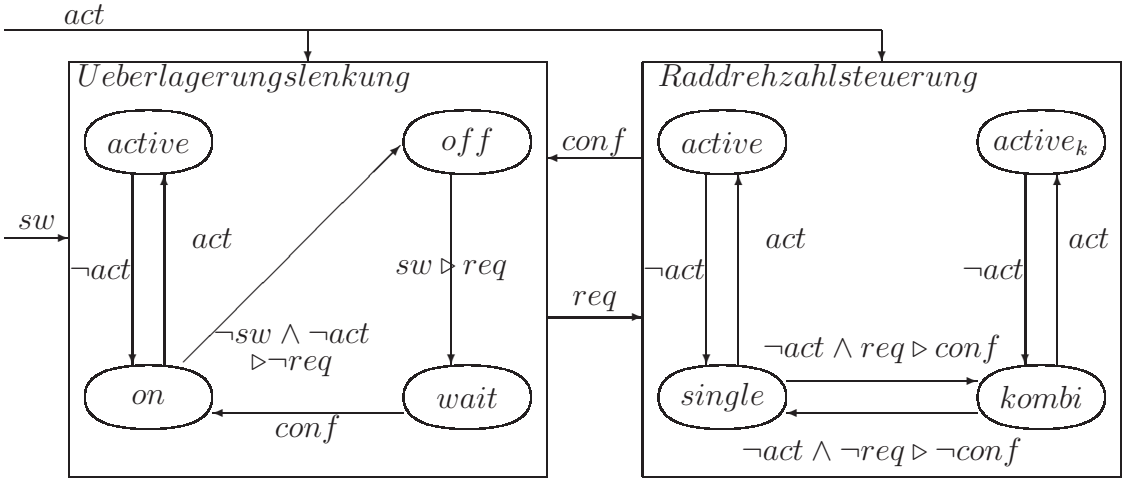


Abb. 3.7: Beispieldarstellung zweier Automaten (siehe Beispiel 3.3.5)

$L_{RDS} = \{state_{RDS} : \{active, kact, single, kombi\}\}$
$S_{RDS_0} = \{\alpha_{RDS} | state_{RDS} = single \wedge conf = false\}$
$\Phi_{RDS}^{trigger} = (true)$
$[\![\delta_{RDS}]\!] =$
$(act = true \wedge state_{RDS} = single \wedge state'_{RDS} = active \wedge conf' = conf)$
$\vee\ (act = false \wedge state_{RDS} = active \wedge state'_{RDS} = single \wedge conf' = conf)$
$\vee (act = false \wedge state_{RDS} = single \wedge req = true \wedge state'_{RDS} = kombi \wedge conf' = true)$
$\vee\ (act = false \wedge state_{RDS} = kombi \wedge req = false \wedge state'_{RDS} = single \wedge conf' = false)$
$\vee\ (act = true \wedge state_{RDS} = kombi \wedge state'_{RDS} = kact \wedge conf' = conf)$
$\vee\ (act = false \wedge state_{RDS} = kact \wedge state'_{RDS} = kombi \wedge conf' = conf)$

$I_{UL} = \{sw : bool, act : bool, conf : bool\}$
$O_{UL} = \{req : bool\}$
$L_{UL} = \{state_{UL} : \{active, on, off, wait\}\}$
$S_{UL_0} = \{\alpha_{UL} | state_{UL} = off \wedge req = false\}$
$\Phi_{UL}^{trigger} = (true)$
$[\![\delta_{UL}]\!] =$
$(act = true \wedge state_{UL} = on \wedge state'_{UL} = active \wedge req' = req)$
$\vee\ (act = false \wedge state_{UL} = active \wedge state'_{UL} = on \wedge req' = req)$
$\vee\ (act = false \wedge sw = false \wedge state_{UL} = on \wedge state'_{UL} = off \wedge req' = false)$
$\vee\ (sw = true \wedge state_{UL} = off \wedge state'_{UL} = wait \wedge req' = true)$
$\vee\ (conf = true \wedge state_{UL} = wait \wedge state'_{UL} = on \wedge req' = req)$ ⌟

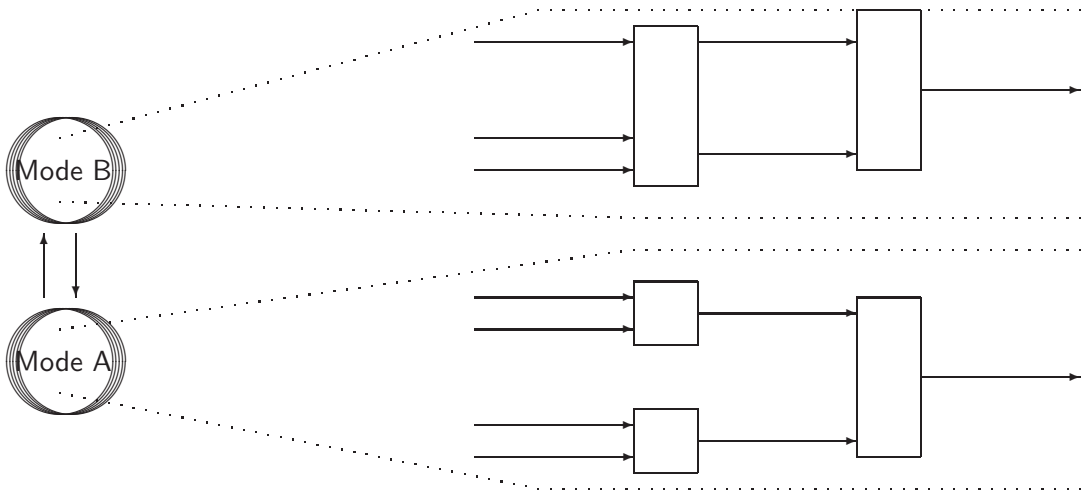


Abb. 3.8: Beispieldarstellung zweier Betriebsmodi

3.3.4 Spezifikation mit Betriebsmodi-Automaten

Betriebsmodus

Automaten bieten mit Kontrollzuständen die Möglichkeit, mehrere Zustände zusammenzufassen. Das Verhalten eines Systems kann mit Kontrollzuständen übersichtlicher beschrieben werden. Ein Sonderfall der Kontrollzustände sind die *Betriebsmodi*. Mit ihnen werden verschiedene Verhaltensmuster unterschieden, die auch mit verschiedenen Modellen dargestellt werden können. Abbildung 3.8 stellt den Begriff des Betriebsmodus graphisch dar (siehe auch [BBR+05]). Für ein System gilt in einem Betriebsmodus das Verhalten, das in einem Modellteil spezifiziert ist und in einem anderen Betriebsmodus das Verhalten, das in einem anderen Modellteil spezifiziert ist. Je nach dem, in welchem Betriebsmodus sich das System gerade befindet, werden die entsprechenden Ausgaben der Teilmodelle aus dem System weitergeleitet. Die Umschaltlogik zwischen den Betriebsmodi wird in einem expliziten Betriebsmodi-Automaten modelliert.

Kombination der Sichten

Der Einsatz dieser Betriebsmodi-Darstellung eignet sich zur Modellierung des Zusammenspiels der Regelungssicht und der Interaktionssicht. In den jeweiligen Teilmodellen der Betriebsmodi können regelungstechnische Funktionen untergebracht werden, die anhand der Zustände der Betriebsmodi-Automaten aktiviert werden. Das Zusammenspiel der Sichten wird explizit dargestellt und es können entsprechende Verifikations- und Modellierungsrichtlinien den jeweiligen repräsentativen Modellteilen zugeordnet werden. Die Betriebsmodi-Automaten selbst kommunizieren abgesehen von der Information über ihren aktuellen Zustand nicht nach außen. Die Interaktion mit anderen Komponenten wird in getrennten Zustandsautomaten modelliert, wodurch der Charakter der Betriebsmodi-Automaten als Sicht-Schnittstelle hervorgehoben wird.

Beispiel 3.3.6 (Betriebsmodi eines Systems)
Der Fahrer kann bei der geschwindigkeitsabhängigen Lenkübersetzung zwischen zwei Kennlinien (Sport und Komfort) für die Übersetzung wählen. Die Umschaltung erfolgt, falls angefordert, wenn die Räder gerade aus zeigen. Die Modelle für die Regelung der Lenkübersetzung (welche den Kennlinien der Modi entsprechen) und das Modell für die Umschaltlogik (Automat) können bei einer Modellierung als Betriebsmodi-Automat getrennt entwickelt und analysiert werden. ⌟

Aktivierung der Teilmodelle

Bei der Modellierung mit einem Automaten mit Kontrollzuständen umfasst der Zustand des Systems alle Variablenwerte der Komponenten. Wird eine Komponente mit Betriebsmodi modelliert, so gibt es verschiedene Möglichkeiten, die Teilmodelle miteinander zu kombinieren. Das Zusammenspiel zwischen den Teilmodellen wird zum einen bezüglich der Rechenleistung festgelegt und zum anderen bezüglich des Informationsaustausches. Bezüglich der Rechenleistung gilt:

- Die Teilmodelle werden parallel ausgeführt und gerechnet.
  Auch wenn ein Teilmodell gerade nicht seine Ausgaben aus der Komponente senden kann, führt es trotzdem Berechnungen aus. Diese Modellierung erfordert nicht die Berücksichtigung eines Abschaltkonzeptes, benötigt aber für jedes der Modelle Rechenleistung.
- Nur das aktive Teilmodell wird ausgeführt (Sequenzielle Ausführung).
  Bei dieser Art der Koordination werden die Teilmodelle sequentiell ausgeführt, wobei die Sequenz durch den Betriebsmodi-Automaten bestimmt wird. Hier ist die benötigte Rechenleistung geringer, da schlimmstenfalls nur die Leistung des rechenintensivsten Teilmodells benötigt wird. Hierbei muss ein Abschaltkonzept berücksichtigt werden.

Übergang zwischen den Modi

Bei Informationssystemen werden die Teilmodelle meist aus Effizienzgründen sequenziell geschaltet, also immer nur das aktive Teilmodel ausgeführt. Bei der Sequenziellschaltung werden in [BRS05] drei Arten des Zusammenspiels der Komponenten bezüglich der einzelnen Variablen bei einem Modus-Wechsel dargestellt. Es wird entweder bei einem Modus-Wechsel die Variable des aktivierten Teilmodells immer neu initialisiert, oder sie behält den Zustand, den sie bei der Deaktivierung des Teilmodells zuletzt gehabt hat, oder sie übernimmt den Wert von dem zuletzt aktivierten Teilmodell. Aus der Modellierungsidee der Funktionsmodellierung mit Simulink ist jedoch die Parallelkomposition der Teilmodelle mit herkömmlichen Mitteln leicht zu realisieren. Werden die Modelle parallel ausgeführt, so ergeben sich in der Praxis zwei Möglichkeiten, die eine relativ unabhängige Modellierung der Teilsysteme zulassen:

- Es findet keine Kommunikation zwischen den Teilmodellen statt.
  Die verschiedenen Teilmodelle werden parallel voneinander gerechnet und es wird je nach Betriebsmodus entschieden, welche Ausgabe des jeweiligen Modells ausgegeben wird. Bei einer Umschaltung zu einem anderen Betriebsmodus ist es egal, welche Ausgaben das andere Teilmodell gehabt hat.

- Die Teilmodelle können auf die Ausgabe der Gesamtkomponente zugreifen. Es findet eine unidirektionale Kommunikation von dem jeweils aktivierten Teilmodell zu den anderen Teilmodellen statt. Ein Modus entspricht so einem Kontrollzustand, dessen lokale Variablen in keinem anderen Zustand gelesen oder verändert werden.

formale Definition

Im Folgenden wird eine formale Definition eines Betriebsmodus gegeben, welche eine parallele Berechnung der Teilmodelle beinhaltet. Die parallele Ausführung wurde gewählt, da die gängigen Datenfluss-Beschreibungssprachen für reaktive Systeme (z.B. Simulink oder Lustre) nur die parallele Zusammensetzung von Teilmodellen kennen (siehe [MR98b]) und so eine Umsetzung mir diesen Sprachmitteln möglich ist. Zur Verwendung anderer Modi-Interpretationen sei auf die Arbeiten [MR98b], [BRS05] und [HHK03] verwiesen.

Definition 3.3.8 (Betriebsmodi-Komponente)
Eine *Betriebsmodi-Komponente* S^{BM} mit der Schnittstelle (I^{BM}, O^{BM}) und dem Verhalten $[\![S^{BM}]\!]^{op}$ wird durch das Tupel $S^{BM} = (A, T, O, \Phi^{mapping})$ bestimmt.

Gegeben sei eine Menge von Namen der Betriebsmodi der Komponente
$$N_{BM} = \{bm1, bm2, ...\}.$$

A ist ein Zustandsautomat mit der Einschränkung, dass dessen Ausgabe O_A aus einem Kanal $O_A = \{o_A\}$ besteht. Die Wertemenge des Kanals entspricht der Menge der Namen der Modi, also $W_{o_A} = N_{BM}$.

Die Menge T enthält die Komponenten, deren Ausgaben in den jeweiligen Betriebsmodi aus der Komponente ausgegeben werden, also $T = \bigcup_{x \in N_{BM}} \{S_x\}$.

Die Eingabekanäle der Komponente S^{BM} sind die Vereinigung der Eingabekanäle der Komponenten in T und des Automaten A, also $I^{BM} = I_A \cup (\bigcup_{x \in N_{BM}} I_{S_x})$.

Die Menge $O^{BM} = \{o_{BM.1}, o_{BM.2}, ..., o_{BM.n}\}$ ist die Menge der Ausgabekanäle der Komponente S^{BM}. Sie kann o_A enthalten.

Jede Teilkomponente S_x mit $x \in N_{BM}$ hat eine (interne) Ausgabe
$O_x = \{o_{x.1}, o_{x.2}, ..., o_{x.n}\}$.

Das Prädikat $\Phi^{mapping} = \bigwedge_{x \in N_{BM}, 1 \leq z \leq n} \Phi^{mapping}_{x.z}$ ist eine Konjunktion von Teilprädikaten für jeden Betriebsmodi und jeden Ausgabekanal der Komponente. Die Teilprädikate haben die Form
$$\Phi^{mapping}_{x.z} = ((o_A = x) \Rightarrow (o_{x.z} = o_{BM.z})).$$

Das Verhalten der Komponente ergibt sich aus dem Verhalten des Automaten, der Teilmodelle und des Mappings, also
$$[\![S^{BM}]\!]^{op} = \Phi^{mapping} \wedge [\![A]\!] \wedge [\![S_{bm1}]\!]^{op} \wedge [\![S_{bm2}]\!]^{op} ... \quad \lrcorner$$

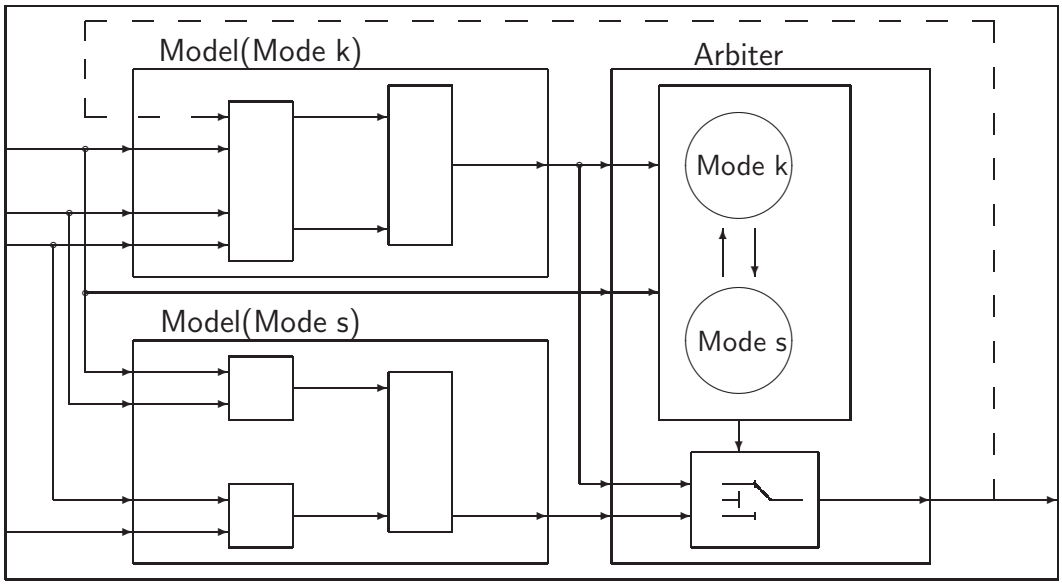


Abb. 3.9: Beispielimplementierung zweier Betriebsmodi (siehe Bsp. 3.3.7)

Implementierung

Die meisten Modellierungssprachen bieten nicht die Möglichkeit, Betriebsmodi explizit zu modellieren. Betriebsmodi lassen sich über ein Konstrukt mit herkömmlichen Mitteln relativ einfach erstellen. Die Umsetzung der Betriebsmodi in einer sequentiellen Programmiersprache geschieht durch Verwendung von if-Anweisungen und wird in [MR98a] erklärt. Bei einer graphischen Datenfluss-Modellierung werden die verschiedenen Teilmodelle in getrennten Teilkomponenten modelliert, wobei die Eingabekanäle der äußeren Komponente auch die Eingabekanäle der Teilkomponenten sind. Die Ausgaben der Teilkomponenten werden an einen so genannten *Schiedsrichter (engl. Arbiter)* gesendet. Dieser entscheidet, von welcher Teilkomponente die Ausgaben aus der äußeren Komponente herausgeleitet werden. Der Schiedsrichter besteht aus einem Automaten, der den aktuellen Zustand bestimmt und einem Schalter (engl. Switch) der entsprechend die Ausgaben weiterleitet. Abbildung 3.9 verdeutlicht diesen Aufbau. Je nach dem, welche der beiden oben gegebenen Interpretationen für Betriebsmodi gewählt ist, kann eine Rückkopplung der Ausgaben zu den Teilmodellen stattfinden (angedeutet mit gestrichelter Linie).

Beispiel 3.3.7 (Spezifikation mit Betriebsmodi-Komponenten)
Gegeben ist eine geschwindigkeitsabhängige Lenkübersetzung S_{BM}, deren Lenkübersetzung wahlweise zwei verschiedenen Kennlinien ($sport(s)$ und $komfort(k)$) entsprechen kann. Zur Vereinfachung findet die Umschaltung zwischen den Kennlinien sofort statt. Diese Komponente ist definiert als (siehe auch Abbildung 3.9) $S^{BM} = (A, T, O^{BM}, \Phi^{mapping})$ mit:

$T = (S_s, S_k)$

$I_{S_s} = \{i_{lrw} : float, i_v : float\}$
$O_{S_s} = \{o_{vrw\_s} : float\}$
$[\![S_s]\!]^{op} = (o_{vrw\_s} = i_{lrw}/(16 \cdot \mathsf{max}(1, (|i_v|/100km/h)))$

$I_{S_k} = \{i_{lrw} : float\}$
$O_{S_k} = \{o_{vrw\_k} : float\}$
$[\![S_k]\!]^{op} = (o_{vrw\_k} = i_{lrw}/(20 \cdot \mathsf{max}(1, (|i_v|/100km/h)))$

$O_{BM} = \{o_{vrw}\}$

$A = (\{taste : bool\},$
$\{modus : \{s, k\}\},$
$\{state : \{s, k, s_{pre}, k_{pre}\}\},$
$\{\alpha_{S_{BM}} | state = s \wedge modus = s\},$
$((taste = true \wedge state = s \wedge state' = k_{pre} \wedge modus' = k)$
$\vee(taste = false \wedge state = k_{pre} \wedge state' = k \wedge modus' = k)$
$\vee(taste = true \wedge state = k \wedge state' = s_{pre} \wedge modus' = s)$
$\vee(taste = false \wedge state = s_{pre} \wedge state' = s \wedge modus' = s)),$
$true\ )$

$\Phi^{mapping} = ((modus = s) \Rightarrow (o_{vrw} = o_{vrw\_s})) \wedge ((modus = k) \Rightarrow (o_{vrw} = o_{vrw\_k}))$ ⌟

3.3.5 Spezifikation der kontinuierlichen Anteile

Eigenschaften der Schnittstelle

Die Besonderheit mechatronischer Systeme liegt in dem Zusammenspiel der auf der einen Seite stehenden Logik und Hardware der eingebetteten Rechnersysteme und des auf der anderen Seite stehenden physikalischen Teils, der aus dem physikalischen Teil des mechatronischen Systems und dem der Umwelt besteht (siehe Abschnitt 3.1). Die *eingebetteten Systeme* sind getaktete Rechnersysteme, die mit diskreten Werten umgehen. Das Verhalten der physikalischen Systeme kann zusätzlich nichtdeterministisch, wertkontinuierlich und insbesondere zeitkontinuierlich sein. Die Schnittstellen des getakteten eingebetteten Systems zum physikalischen Teil werden als *Sensoren* (kontinuierlich zu diskret) und *Aktuatoren* (diskret zu kontinuierlich) bezeichnet. Dieser Abschnitt konzentriert sich auf die Modellierung des Verhaltens der Umgebung des eingebetteten Rechnersystems. Es werden keine neuen Modellierungstechniken eingeführt, sondern es wird darauf eingegangen, wie mit den bisher aufgeführten Modellierungstechniken die Physik hinsichtlich der Bewertung der Funktionssicherheit des eingebetteten Systems geeignet modelliert werden kann. Dazu werden die Eigenschaften der physikalischen Teile beschrieben und entsprechende Vereinfachungen bei der Beschreibung des Verhaltens genannt, um die Bewertung der Modelle zu ermöglichen.

Eigenschaften des physikalischen Teils

Die physikalische Umgebung des eingebetteten Systems hat zwei wesentliche Eigenschaften:

(1.) Das Verhalten des physikalischen Teils ist häufig von Faktoren beeinflusst, die nicht präzise erfasst werden können. So kann z.B. die Lenkung durch Unebenheiten der Straße beeinflusst werden. Besonders Systeme der Umwelt des mechatronischen Systems, in der auch der Faktor Mensch und andere natürliche Systeme Bestandteile sind, sind meist so komplex, dass sie nicht deterministisch modellierbar sind. Ein Modell des physikalischen Teils ist also meistens nichtdeterministisch.

(2.) Das Verhalten der Umgebung unterliegt nicht mehr der künstlich geschaffenen Logik, die programmiert wird, sondern ist durch die mechanische Bauweise oder physikalischen Umstände gegeben. Die Komplexität des Systems ist also ein fester Faktor. Dies zeigt sich z.B. darin, dass die Mechanik meistens kontinuierlich ist. Sowohl die Zeit, wie auch die Variablen-Werte sind kontinuierlich.

freie Modellierung

Die Beschreibung des Verhaltens der physikalischen Umwelt kann theoretisch mit beliebigen Blackbox-Spezifikationen (siehe Abschnitt 3.3.1 und 3.3.2) stattfinden. Die Spezifikationen können zum Beispiel Integrale und trigonometrische Funktionen enthalten. Der Übergang zwischen dem kontinuierlichen und dem diskreten Teil des Systems an den Aktuatoren und Sensoren kann real modelliert werden. Nichtdeterminismus kann entweder mit unscharfen Gleichungen (siehe Abschnitt 3.3.2), Ungleichungen oder logischen Aussagen modelliert werden. Beliebige kontinuierliche nichtdeterministische Modelle sind mit heutigen Mitteln der Mathematik nicht verifizierbar und nur schwer analysierbar ([Sch03], S. 58). Dies ist die Ursache für die im Rest des Abschnitts genannten Beschreibungsmechanismen.

Sampling

Eine Vereinfachung bei der Beschreibung der Umgebung des eingebetteten Systems ist, die Beschreibung zu diskretisieren. Die Zeit wird zu diskreten Zeitpunkten abstrahiert. Dabei können entweder flexible Zeitspannen zwischen den Zeitpunkten zugelassen werden ([Sch03], [Hen96]), oder feste Zeitspannen verwendet werden. Zur Analyse eingebetteter Systeme eignet sich die Verwendung fester Zeitspannen, welche die Umgebung zu den Taktzeitpunkten des eingebetteten Systems betrachten ([Sch03], [HCRP91]). Des Weiteren werden die kontinuierlichen (eventuell sogar unendlichen) Wertebereiche der Variablen der Umgebung auf endliche Zahlen abgebildet (z.B. als Fließkomma- oder Integer-Zahlen einer bestimmten Bitlänge). Diese Diskretisierung der Zeit und der Werte wird Sampling genannt. Alle Modelle, die einer derartigen Diskretisierung zugrundeliegen, haben einen endlichen Zustandsraum. Zumindest theoretisch sind diese Modelle simulierbar und verifizierbar. In der Praxis wächst jedoch auch bei endlichen Wertemengen der Variablen der Zustandsraum sehr schnell, so dass eine automatische Verifikation schnell mit der Größe des Zustandsraums sehr viel Speicherplatz und Berechnungszeit braucht. Um eine Anwendung in der Praxis zu erlangen, ist eine systematische starke Einschränkung der Menge der diskreten Zustände nötig.

ND-Automaten

Ist ein abstraktes Systemmodell diskret, kann der Nichtdeterminismus der Umgebung mit ND-Automaten modelliert werden. Von einem Zustand gehen dann entsprechend mehrere Transitionen aus, die bei gleichen Eingaben schaltbereit sind. Bei einer Modellierung mit Automaten müssen geeignete abstrakte Zustände gefunden werden, mit denen die wichtigen Eigenschaften des Systems weiterhin dargestellt werden können. Eine genaue Beschreibung der Modellierung der Physik mit Automaten findet man in [OR04] und [ORS05].

Linearisierung

Eine Darstellung der Umwelt mit Automaten mit entsprechend abstrakten Zuständen ist nicht immer möglich und die Modelle sind auch nicht immer intuitiv verständlich. Eine Alternative besteht darin, nicht den Zustandsraum weiter einzuschränken, sondern die Funktionen, die das Verhalten beschreiben, zu approximieren. Hier-

zu werden die Funktionen in eine verifizierbare Form gebracht. Die am häufigsten verwendete Vereinfachung ist die lineare Approximation der Funktionen. Auf diese Weise erhält man Funktionen, die in (Un-)Gleichungssysteme umgeformt werden können, die systematisch mit Mitteln der Mathematik gelöst werden können (siehe [Hen96], [LSV95], [Sch03]).

3.4 Abhängigkeiten zwischen Modellen

schrittweises Vorgehen

Die bisher beschriebenen Modellierungstechniken ermöglichen es, ein System zu beschreiben. In einem Entwicklungsprozess wird ein Modell jedoch nicht sofort detailliert beschrieben, sondern es wird schrittweise erarbeitet. Ein Hilfsmittel bei der schrittweisen Modellierung eines Systems ist die Zerlegung in einfacher zu modellierende Teile, die Modellierung der Teile und dann das Zusammenfügen der Teile zu einem Ganzen. Zwei Modelle, die zusammen ein Ganzes ergeben, stehen in einer Kompositionsbeziehung. Ein weiteres Mittel der schrittweisen Modellierung ist, zuerst eine grobe Beschreibung zu erstellen, und diese dann immer detaillierter zu machen. Ein Modell, welches eine detailliertere Beschreibung eines anderen Modells ist, steht in einer Verfeinerungsbeziehung zu diesem.

Abschnittsübersicht

Dieser Abschnitt beschreibt die Abhängigkeiten Komposition und Verfeinerung für die Modellierungstechniken in Abschnitt 3.3. Von diesen Abhängigkeiten können gezielt Fehler abgeleitet werden, die aus dem Umgang mit den Modellen entstehen.

3.4.1 Komposition

Systeme werden üblicherweise aus mehreren Teilkomponenten zusammengesetzt. Idealerweise können diese im Entwicklungsprozess voneinander unabhängig entwickelt werden. Allein die Strukturierung bringt oft bereits ein deutlich verständlicheres Bild des Systems. Dieser Abschnitt erläutert in Anlehnung an [Bre01] kurz die Voraussetzungen an eine *Komposition* von Systemen und deren Definition.

Kompositionsbegriff

Systeme agieren nur über ihre Ein- und Ausgabekanäle mit ihrer Umgebung. Ein komponiertes System entsteht so über die Verschachtelung der Kanäle, in dem angegeben wird, welche Ausgaben eines Systems als Eingaben eines anderen Systems dienen. Fügt man zwei Systeme zusammen, so dürfen ihre Ausgabekanäle keine gleichen Namen haben, um einen Identitätskonflikt zu vermeiden. Es gilt (angelehnt an [Bre01]):

Definition 3.4.1 (Komposition)
Gegeben sind zwei Systeme S und T mit den Schnittstellen (I_S, O_S) und (I_T, O_T) und dem Verhalten R_S und R_T mit $O_S \cap O_T = \emptyset$

Ein System U ist genau dann ein aus S und T zusammengesetztes System, wenn gilt: $U = S \otimes T$

Der Kompositionsoperator $\otimes$ für die Systeme S, T und U ist folgend definiert:
$\otimes : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$
$U = S \otimes T \overset{def}{\Leftrightarrow} ((I_U, O_U) = (I_S, O_S) \otimes (I_T, O_T)) \wedge (R_U = R_S \otimes R_T)$

Der Kompositionsoperator $\otimes$ für die Systeme (I_S, O_S), (I_S, O_S) und (I_U, O_U) ist folgend definiert:
$(I_U, O_U) = (I_S, O_S) \otimes (I_T, O_T) \overset{def}{\Leftrightarrow} (I_U = (I_S \cup I_T) \setminus (O_S \cup O_T)) \wedge (O_U = (O_S \cup O_T))$
Die Menge der Rückkopplungskanäle ist:
$L_U \overset{def}{=} (I_S \cup I_T) \cap (O_S \cup O_T)$ ⌟

Black-Box

Im Falle der Black-Box-Spezifikationen ist die Komposition von zwei Systemen wesentlich einfacher auszudrücken. Für die Schnittstelle gilt die obige Kompositionsregel. Da die Kanäle über ihre Namen identifiziert werden und somit in den Gleichungen durch eindeutige Variablennamen repräsentiert werden, gilt für das Verhalten (siehe [Bre01]):

Definition 3.4.2 (Komposition (mit Prädikaten))
Gegeben seien die Verhaltensweisen $[\![S]\!]$ und $[\![T]\!]$ zweier Systeme S und T. Das Verhalten der Komposition $[\![S \otimes T]\!] = [\![S]\!] \otimes [\![T]\!]$ ist gegeben durch
$$[\![S]\!] \wedge [\![T]\!]$$ ⌟

mehrere Systeme

Die Definition der Komposition gilt nur für zwei Systeme. Es kann durch eine iterative Anwendung der Komposition eine beliebig große Menge an Systemen zusammengefasst werden. Für den Kompositionsoperator gilt hierbei sowohl das Assoziativwie auch das Kommutativgesetz, d.h. die Reihenfolge des Zusammensetzens spielt keine Rolle. Wichtig ist jedoch, dass die zu verbindenden Kanäle in ihren Namen übereinstimmen. Hier kann mittels der Umbenennung, die einer Bijektion auf Kanalnamen entspricht, Abhilfe geschafft werden (siehe [Bre01]).

Serienschaltung

Ein Sonderfall der Komposition ist die *Serienschaltung*, bei der alle Ausgaben der jeweils vorhergehenden Komponente als Eingabe der folgenden Komponenten dienen. Bei dieser Art der Komposition ist somit eine rückkopplungsfreie Komponente beschrieben, die als Eingabe die Kanäle der ersten Teilkomponente und als Ausgabe die Kanäle der zweiten Teilkomponente hat. Beschrieben wird diese Art der Komposition für zwei Komponenten S und T mit $S \succ T$.

3.4.2 Verfeinerung

unterschiedliche Spezifikationen

Die Entwicklung eines Systems findet nicht immer in vollem Detaillierungsgrad statt. Es ist oft hilfreich, von dem tatsächlichen Verhalten zu abstrahieren. So wird z.B. bei der Entwicklung eines Systems zu Beginn eine grobe Beschreibung des Systems erstellt, die dann anhand von Entwurfsentscheidungen weiter und genauer beschrieben wird. Diese Entwurfsentscheidungen umfassen unter anderem die Aufteilung eines Systems in Teilsysteme und die Wahl einer Systemarchitektur, aber auch die präziseren Anforderungen an das Soll-Verhalten, das mit fortschreitendem Reifegrad immer besser verstanden wird. Es ist auch zur besseren Handhabbarkeit und Analyse nützlich, bestimmte Eigenschaften eines Systems in einer für den jeweiligen Betroffenen gut lesbaren Beschreibung darzustellen, ohne ihn mit Details zu überschütten.

Zusammenhang

Die in einem Entwicklungsprozess bestehenden Spezifikationen stellen immer das gleiche System dar und sind dementsprechend voneinander abhängig. Idealerweise stehen die Spezifikationen zu der jeweils detailliertesten Spezifikation in einer Verfeinerungsbeziehung. Eine Spezifikation ist eine *Verfeinerung* einer anderen Spezifikation, wenn sie im Wesentlichen das gleiche System aus einer gleichen Sicht beschreibt, aber zusätzliche Eigenschaften und Konkretisierungen enthält. Es gibt je nach konkretem Verwendungszweck eine Menge verschiedener Verfeinerungsarten (siehe [BS01], [Bre01], [Bro02]). In dieser Arbeit hat die Verfeinerung folgende konkrete Aufgaben (die auch in Kombination auftreten können) zu unterstützen:

- Das Auflösen von Nichtdeterminismus (Verhaltensverfeinerung)
- Das Einschränken der Menge der Eingabeströme (Bedingte Verfeinerung)
- Das Sicherstellen einer Mindestfunktionalität nach der Verfeinerung (Bedingte sicherstellende Verfeinerung)
- Das Abbilden der Kanäle auf andere Kanäle mit anderen Datentypen (Schnittstellenverfeinerung)

Verhaltensverfeinerung

Im Folgenden wird der formale Begriff der Verfeinerung von einem einfachen Verfeinerungsbegriff hin zu einem allgemeinen Verfeinerungsbegriff aufgebaut. Die erste Variante der Verfeinerung kommt nur der Aufgabe des Auflösens von Nichtdeterminismus entgegen, ohne die Menge der akzeptierten Eingabeströme zu reduzieren und ohne auf die Menge der Ausgabeströme zu achten. Diese Verfeinerung wird Verhaltensverfeinerung genannt und ist wie folgt definiert (siehe [Bre01], [BS01]):

Definition 3.4.3 (Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T.
Das System T ist eine *Verhaltensverfeinerung* des Systems S, notiert als:

$$S \rightsquigarrow T$$

wenn jedes Verhalten von T auch ein Verhalten von S ist. Es gilt also:

$R_T \subseteq R_S$ und $dom.R_S = dom.R_T$ mit dom gemäß Definition 3.2.5. ⌟

bedingte Verfeinerung

Mit der Verhaltensverfeinerung lässt sich zwar die Menge der Ausgabeströme reduzieren, es ist aber nicht möglich, die Menge der Eingabeströme zu reduzieren, da bei der Komponente nur Nichtdeterminismus aufgelöst wurde. Die Einschränkung der Eingabe, wenn man z.B. die Eingabe von einem unendlichen Datentypen auf einen endlichen Datentypen reduzieren möchte, geschieht über eine bedingte Verhaltensverfeinerung, bei der über eine Kondition angegeben wird, welche Werte nicht berücksichtigt werden müssen.

Definition 3.4.4 (bedingte Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T, sowie eine Bedingung C.
Das System T ist eine *bedingte Verhaltensverfeinerung* des Systems S

$$S \rightsquigarrow_C T$$

wenn jedes Verhalten von T auch ein Verhalten von S ist und die Eingabemenge reduziert werden kann. Es gilt also:

$$R_T \subseteq R_S \text{ und } rng.C = dom.R_T \text{ und } rng.C \subseteq dom.R_S$$ ⌟

Die Verwendung der Bedingung C bei der Beschreibung der bedingten Verhaltensverfeinerung kann theoretisch weggelassen werden, da eine Einschränkung des Eingabebereichs auch durch Auslassen der Bedingung $dom.R_S = dom.R_T$ bei der Verhaltensverfeinerung möglich wäre. Es ist jedoch für den Entwickler besser nachvollziehbar, wenn die Einschränkung explizit sichtbar ist. Der verkürzte Begriff der Verhaltensverfeinerung ist:

Bei Relationsspezifikation: $R_T \subseteq R_S$
Bei Blackboxspezifikation: $[\![T]\!] \Rightarrow [\![S]\!]$

Mindestfunktionalität

Bei der Spezifikation eines Systems liegen häufig auch Anforderungen vor, die eine gewünschte Mindestfunktionalität fordern, also eine minimale Menge an Ausgabeströmen. Der bisherige Begriff der Verfeinerung betrachtet nur, dass das Verhalten einer Komponente nicht erweitert wird. Über die Bedingung G kann bei der bedingten sicherstellenden Verfeinerung eine Mindestforderung an die verfeinerte Komponente weitergegeben werden.

Definition 3.4.5 (bedingte sicherstellende Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T, sowie die Bedingungen C und G.
Das System T ist eine *bedingte sicherstellende Verhaltensverfeinerung* des Systems S

$$S \rightsquigarrow_{(C,G)} T$$

wenn jedes Verhalten von T auch ein Verhalten von S ist und G berücksichtigt wird. Es gilt also:

$$S \rightsquigarrow_C T \text{ und } G \vDash R_T$$ ⌟

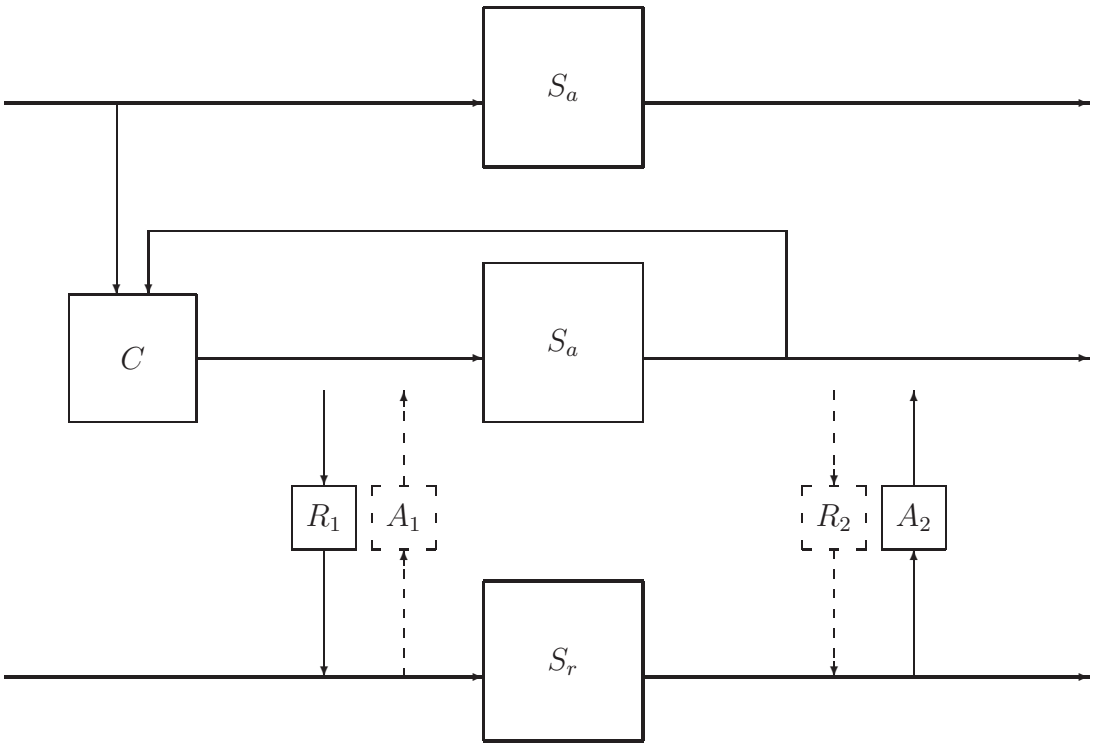


Abb. 3.10: allgemeine Verfeinerung einer Komponente

Nachrichtenverfeinerung

Neben der Verfeinerung des Verhaltens kann auch eine Verfeinerung der Schnittstelle mit einhergehen. Hier können sich Datentypen, die Anzahl der Nachrichtenkanäle sowie die Repräsentation der Nachrichten verändern. Dies kann z.B. notwendig sein, wenn eine Funktion an ein spezifisches Steuergerät angepasst wird. Hier ist also ein allgemeinerer Begriff der Verfeinerung notwendig, der eine abstraktere Spezifikation mit einer konkreteren Spezifikation in Beziehung setzt (siehe Abbildung 3.10). Die Nachrichten des abstrakten Systems werden über eine Repräsentationsrelation in Beziehung mit dem konkreten System gesetzt. Eine Abstraktions- / Repräsentationsrelation zwischen zwei Systemen ist wie folgt definiert (siehe auch [Bro02]):

Definition 3.4.6 (Abstraktionsrelation & Repräsentationsrelation)
Eine Verfeinerungsbeziehung $g \overset{(A,R)}{\rightsquigarrow} f$ zwischen zwei Nachrichtenkanälen f und g ist durch eine *Abstraktionsrelation* $A \subseteq \mathcal{F}_f \times \mathcal{F}_g$ und eine *Repräsentationsrelation* $R \subseteq \mathcal{F}_g \times \mathcal{F}_f$ gegeben. Dabei muss gelten, dass beide Relationen surjektiv und linkstotal sind und eine konkrete Nachricht immer wieder auf die ausgehende abstrakte Nachricht schließen lässt, also gilt: $R \succ A = ID$ ⌟

Schnittstellenverfeinerung

Auf Basis der Verfeinerungsbeziehung zwischen Nachrichtenkanälen ist es möglich, zwei Komponenten mit verschiedenen Schnittstellen in Beziehung zu setzen. Die jeweils verfeinerte Komponente hat das Verhalten der jeweils generelleren Komponente zu erfüllen. Die Definition einer entsprechenden korrekten Abstraktionsrelation und

Repräsentationsrelation ist der kreative Anteil, der bei dieser Art der Verfeinerung als korrekt angenommen wird. Soll der dynamische Datentyp der abstrakteren Beschreibung bei der Verfeinerung eingeschränkt werden, so lässt sich dies über die Angabe einer zusätzlichen Bedingung regeln, die angibt, für welche Eingabeströme die Bedingung gelten soll (siehe Komponente C in Abbildung 3.10). Die Verfeinerung der Schnittstelle ist wie folgt definiert (siehe [Bre01], [BS01]):

Definition 3.4.7 (Schnittstellenverfeinerung)
Gegeben seien zwei Systeme mit den Schnittstellen (I_S, O_S) und (I_T, O_T) und den verhaltensbeschreibenden Relationen S und T. Gegeben seien auch eine Repräsentationsrelation R_1 mit der Schnittstelle (I_S, I_T), eine Abstraktionsrelation A_2 mit der Schnittstelle (O_T, O_S) und eine Bedingung C mit der Schnittstelle (I_C, I_S). Das System T ist eine *Schnittstellenverfeinerung* des Systems S, notiert als:

$$S \overset{(R_1,A_2)}{\rightsquigarrow_{(C,G)}} T$$

wenn gilt:

$$S \rightsquigarrow_{(C,G)} (R_1 \succ T \succ A_2)$$ ⌟

Für diesen allgemeineren Verfeinerungsbegriff ist die Verhaltensverfeinerung der Sonderfall, in dem die Repräsentationsrelation und die Abstraktionsrelation der Identität entsprechen. Dieser erweiterte Verfeinerungsbegriff lässt auch Verfeinerungen zu, die nicht immer sinnvoll sind. Abgeänderte Definitionen und Kriterien zur Verfeinerung finden sich in [BS01] und [Bro98]. Der Begriff der Verfeinerung wird mit dem folgenden Beispiel verdeutlicht:

Beispiel 3.4.1 (Verfeinerung eines Systems)
Gegeben sei eine Spezifikation S einer geschwindigkeitsabhängigen Lenkübersetzung. Die Eingaben sind die Geschwindigkeit v ($W_{\mathcal{F}_v} = \mathbb{Z}$) und der Lenkradwinkel lrw ($W_{\mathcal{F}_{lrw}} = \mathbb{Z}$). Die Ausgabe ist der Vorderradwinkel vrw ($W_{\mathcal{F}_{vrw}} = \mathbb{Z}$). Es gelten folgende Aussagen:

Die Lenkung soll mit steigender Geschwindigkeit indirekter übersetzen:

$$\Phi_1 = ((|v| > |\mathsf{delay}_{\langle 0\rangle}(v)|) \Rightarrow (|(\mathsf{delay}_{\langle 0\rangle}(lrw)/\mathsf{delay}_{\langle 1\rangle}(vrw))| \leq |(lrw/vrw)|))$$

und der Vorderradwinkel soll auf Fahrerwunsch immer auf mindestens 15° eingeschlagen werden können:

$$\Phi_2 = (\forall v : \exists lrw : (vrw > 15°))$$

Gelten für ein System S beide Aussagen, also $[\![S]\!]^{op} = (\Phi_1 \wedge \Phi_2)$, so erfüllen alle Kennlinien diese Spezifikation, die sich an diese Bedingungen halten.

Verhaltensverfeinerung:

Die Spezifikation $[\![S]\!]^{op}$ wird in der Spezifikation $[\![S_1]\!]^{op}$ präzisiert, indem der Nichtdeterminismus beseitigt, also eine Kennlinie als Lösung ausgesucht wird. Es gilt:

$$[\![S_1]\!]^{op} = (vrw = \mathsf{round}(lrw/(16 \cdot \mathsf{max}(1, |v|/100km/h))))$$

Diese System erfüllt auch die Spezifikation $[\![S]\!]^{op}$. Es gilt also: $S \rightsquigarrow S_1$

bedingte Verhaltensverfeinerung:

Zusätzlich wird vorsehen, dass der Eingabewinkel einer Implementierung kleiner 720° ist. Größere Werte werden nicht betrachtet. Die Bedingung C lautet:

$$[\![C]\!]^{op} = (|lrw| < 720°)$$

Dies erlaubt eine Spezifikation S_2, die das gleiche Verhalten wie S_1 aufweist, der Datentyp des Eingabekanals lrw allerdings auf $[-719..719]$ reduziert ist. Es gilt:

$$S \rightsquigarrow_C S_2 \text{ und für } S_2 \text{ gilt zusätzlich: } W_{\mathcal{F}_{lrw}} = [-719..719]$$

bedingte sicherstellende Verhaltensverfeinerung:

Um die existentielle Eigenschaft des Mindestlenkwinkels sicherzustellen, wird die Bedingung $\Phi_G = (\exists lrw : (vrw > 15°))$ angegeben. Diese ist nicht bei jeder Verfeinerung $S \rightsquigarrow_C S_2$ erfüllt, da bei Geschwindigkeiten ab 300 km/h der Lenkradwinkel $< 720°$ nicht mehr ausreicht.

Eine Möglichkeit, diese Bedingung zu erfüllen, ist die Angabe der verschärften Bedingung $[\![C_3]\!]^{op}$ für ein System S_3, bei dem zusätzlich die Eingabe v auf die üblichen 250 km/h eingeschränkt wird.

$$[\![C_3]\!] = (|lrw| < 720° \wedge |v| < 250km/h)$$

Damit gilt bei angepasstem Datentyp v auf $[-249km/h..249km/h]$:

$$S \rightsquigarrow_{C_3} S_3 \text{ und } S \rightsquigarrow_{(C_3,G)} S_3.$$

Schnittstellenverfeinerung:

Letztlich soll der Datentyp der Kanäle v und lrw auf einen Kanal b mit einer 18-Bit-Integer-Darstellung zusammengefasst werden. Hierzu sind die jeweiligen Abstraktions- bzw. Repräsentationsrelationen $R_v, R_{lrw} und A_{vrw}$. Es gilt:

$$\begin{aligned}
[\![R_v]\!]^{op} = (\quad & \textbf{if } (-249 \leq v < 249) \\
& \textbf{then } (b \textbf{ div } 1024 == v) \\
& \textbf{else } (b \textbf{ div } 1024 \in [-256.. -250, 249..255]))
\end{aligned}$$

$$\begin{aligned}
[\![R_{lrw}]\!]^{op} = (\quad & \textbf{if } (-719 <= lrw < 719) \\
& \textbf{then } (b \textbf{ mod } 1024 == lrw) \\
& \textbf{else } (b \textbf{ mod } 1024 \in [-1024.. -720, 719..1023]))
\end{aligned}$$

$$A_{vrw} = ID$$

Ein System S_4 ist nur dann eine Verfeinerung, wenn es sich wie S_3 verhält und alle Abbildungen von 249 bzw. 719 gleich interpretiert. Es gilt also:

$$[\![S_5]\!]^{op} = (vrw = \mathsf{sig}(b \,\mathsf{mod} 1024) \cdot \mathsf{min}(|b \,\mathsf{mod} 1024|, 719)/(\mathsf{min}(|(b \,\mathsf{div} 1024)|, 249)/100 + 1))$$

$$\text{und } S \overset{(R_v, R_{lrw}, A_{vrw})}{\rightsquigarrow_{(C_4,G)}} S_5$$ ⌟

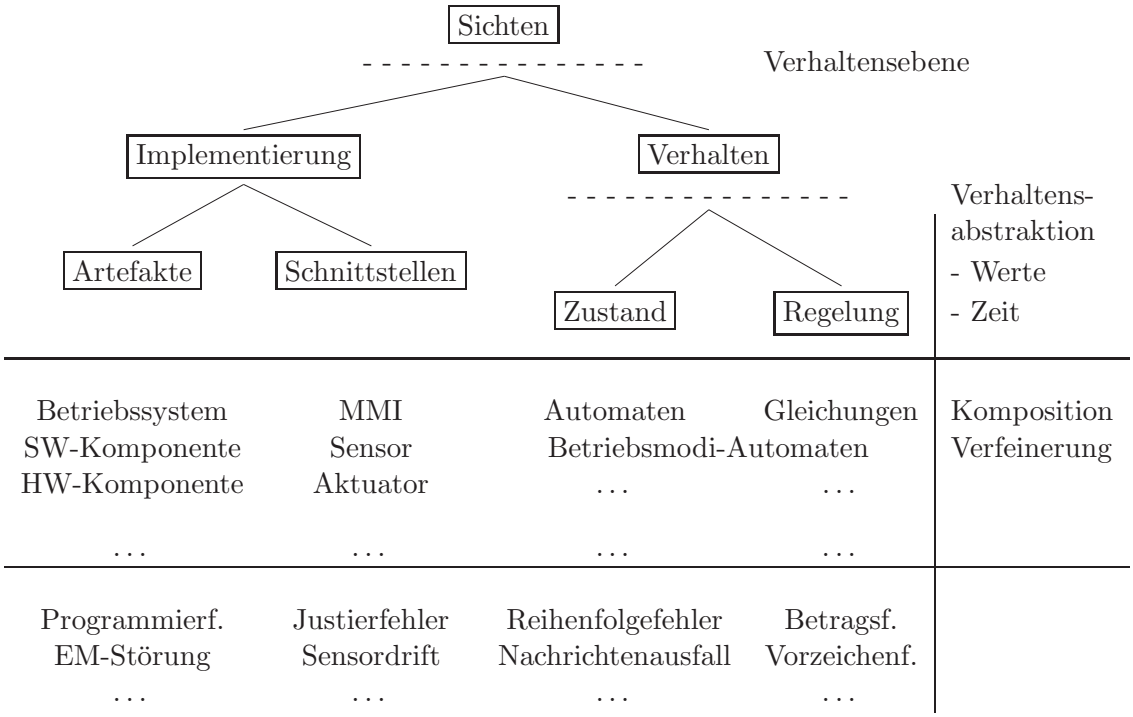


Abb. 3.11: Sichten auf ein System als Grundlage zur Fehleridentifikation

Abweichungen von diesen Verfeinerungsbeziehungen sind potenzielle Fehlerquellen, die bei einer Funktionssicherheitsanalyse berücksichtigt werden können. So kann es zum Beispiel passieren, dass Datentypen gewechselt werden, ohne dass dabei auf Rundungsfehler geachtet wird. Diese sind somit potenzielle Fehler.

3.5 Ansatzpunkte für Fehlerbeschreibungen

Sichten

Aus den Sichten auf die Systeme und aus den jeweiligen Modellierungstechniken lassen sich Ansatzpunkte für die Identifikation und Modellierung von Fehlern ableiten (siehe Abbildung 3.11). Die Sichten sind nicht disjunkt, sondern betrachten die gleichen Artefakte aus verschiedenen Blickwinkeln. Grundlegend gibt es die Verhaltenssicht und die Implementierungssicht. Die Implementierungssicht beschreibt die Artefakte, auf denen das Verhalten abgelegt ist (siehe Abschnitt 3.1). Die Implementierungssicht liefert im Wesentlichen Ansatzpunkte zur Identifikation der potentiellen Fehler eines Systems. In der Verhaltenssicht werden das Verhalten der potentiellen Fehler und das der Folgefehler in einem System betrachtet. Je nach dem, wie die Grenze zur Verhaltensebene gezogen wird, kommen in der Implementierungsstruktur entsprechende Fehler vor. Abstrahiert man zum Beispiel von der Verteilung der Funktionen auf Steuergeräte, so kommen bei der Übertragung der Signale zwischen den Systemen die Standard-CAN-Übertragungsfehler als potenzielle Fehler in Frage.

Implementierungssichten

Die Implementierungssicht (siehe Abschnitt 3.2.2) kann den Blickwinkel auf die Schnittstellen (siehe Beispiel in Abschnitt 3.1) und den auf die Artefakte selbst haben. Ansatzpunkte zur Identifikation potentieller Fehler sind die Arten der Artefakte (Betriebssystem, SW-Modul, ...) und die Qualitätseigenschaften aus [ISO01] (Zuverlässigkeit, Robustheit, ...). Bei elektronischen Bauteilen sind dies z.B. unter anderen elektomagnetische oder thermische Störungen. Auch die Spezifika der Schnittstellen der Systeme liefern Ansatzpunkte zur Fehleridentifikation. So lassen sich aus der Art der Wertübermittlung, anhand der Art der übermittelten Größe und dem Verhalten des Restsystems weitere potenzielle Fehler identifizieren. Wird z.B. die Beschleunigung des Fahrzeugs von einem Sensor gemessen, so kann es bei dem Sensor Justierfehler oder Sensordrift-Fehler geben. Bei der Kommunikation mit anderen Rechnersystemen können hingegen Nachrichtenfehler als potenzielle Fehler betrachtet werden. Den potentiellen Fehlern werden in der Implementierungssicht Wahrscheinlichkeiten zugewiesen. Diese können anhand von Statistiken, Erfahrungen und Schätzverfahren ermittelt werden.

Verhaltenssicht

Während in der Implementierungssicht den Fehlern ein Label und eine Wahrscheinlichkeit zugewiesen wird, steht in der Verhaltenssicht (siehe Abschnitt 3.2.1) der Einfluss der Fehler auf das Verhalten im Mittelpunkt. Der potenzielle Einfluss eines Fehlers auf das Verhalten hängt von der Modellierung bzw. der Sicht auf das Verhalten ab. Die Fehlverhalten können spezifisch für die Interaktionssicht und die Regelungssicht sein. In der Regelungssicht, welche z.B. mit Gleichungen (siehe Abschnitt 3.3.2) modelliert wird, sind Wertabweichungen relevant (z.B. Wert zu groß für 3 Sekunden). Zeit spielt hier eine untergeordnete Rolle. In der Interaktionssicht, welche z.B. mit Automaten (siehe Abschnitt 3.3.3) modelliert wird, sind Nachrichtenfehler relevant (z.B. Nachricht zu spät, Reihenfolgefehler, ...). Zeit und Zeitpunkte spielen hier eine wichtige Rolle. Schließlich sollten insbesondere auch die Zusammenhänge zu Betriebsmodi in einem Fehlermodell berücksichtigt werden können.

Abstraktion

Die Modelle können auf verschiedenen Verhaltensabstraktionsebenen stehen. So kann z.B. ein zeitkontinuierliches Verhalten eines Systems auf ein zeitdiskretes Verhalten abstrahiert werden. Verschiedene Modelle können so in Beziehung zueinander stehen (siehe Abschnitt 3.4). Aus dem Umgang mit Modellen lassen sich potenzielle Fehler identifizieren, welche durch Verletzungen der Verfeinerungs- und Kompositions-Beziehungen entstehen.

Analyse

Die Art der Modellierung und die Sichten auf ein System haben also direkten Einfluss auf die identifizierbaren potentiellen Fehler. Umgekehrt betrachtet ist die Beschreibung der Fehler immer relativ zu der korrespondierenden Sicht der Systemspezifikation beschrieben. Eine Funktionssicherheitsanalyse betrachtet entsprechend nur die Fehler, die sie identifizieren und modellieren kann. Ansatzpunkte für weitere Arbeiten sind zum einen Zuordnungen bestehender formaler Ansätze der Funktionssicherheitsanalyse zu den hier gegebenen Modellsichten, so wie die Betrachtung und der Entwurf von spezifischen Fehlermodellen und Funktionssicherheitsanalysemethoden.

3.6 Zusammenfassung

In diesem Kapitel werden mit Referenzmodellen eingebetteter Systeme und Modellierungstechniken die Grundlagen der Spezifikation dieser Systeme skizziert. Dabei werden Ansatzpunkte herausgestellt, an denen spezifische potenzielle Fehler identifiziert, wie auch Fehlerverhalten spezifisch beschrieben werden können. Diese Arbeit dient so als Grundlage für weitere Arbeiten der Fehlermodellierung und Funktionssicherheitsanalyse.

Die Arbeit ist in vier Kernteile gegliedert. Der erste Teil beschreibt Referenzmodelle eingebetteter Systeme und Referenzmodelle zu den Abstraktionsebenen und Eigenschaften bei der Modellierung der Systeme. Wesentlich sind die Schnittstellen eingebetteter Systeme (Sensoren, Aktuatoren, MMI, Kommunikationsschnittstelle), die Modellierung verschiedener Domänen (Mechanik, Elektrik, Elektronik) und die Abstraktion von Implementierungsdetails.

Der zweite Teil erklärt, wie mit Modellen das Verhalten beschrieben werden kann und wie mit Modellen die Artefakte definiert werden können, auf denen das Verhalten implementiert ist. Das Verhalten wird als eine Relation zwischen kanalbeschreibenden Funktionen modelliert. Komplementär stehen die Implementierungmodelle, welche die Struktur und qualitativen Eigenschaften der Implementierung enthalten. Diese qualitativen Eigenschaften sind ein Ansatzpunkt zur Identifizierung potentieller Fehler, deren Wirkung als Fehlverhalten im Verhaltensmodell dargestellt und in einer Funktionssicherheitsanalyse weiter analysiert werden kann.

Der dritte Teil zeigt Modellierungstechniken, mit denen die Verhaltensmodelle spezifiziert werden können. Hierzu gehören Prädikate und Automaten (‚so wie deren Sonderformen „aufgelöste“ Gleichungen und Betriebsmodi-Automaten). Die Modellierungtechniken sind auf spezifische Sichten hin ausgerichtet (Regelungssicht, Interaktionssicht, usw.). Diese spezifischen Modellierungstechniken bieten so eine Grundlage zu spezifischen Fehlerbeschreibungen.

Der vierte Teil zeigt zwei Abhängigkeiten zwischen Verhaltensmodellen, die bei der Modellierung regelmäßig auftreten. Eine Abhängigkeit ist die Komposition, mit der verschiedene Modelle zu einem Ganzen zusammengefügt werden. Die andere Abhängigkeit ist die Verfeinerung, in der ein Modell detaillierter wird. Anhand der Beschreibung dieser Abhängigkeiten können potenzielle Fehler identifiziert werden, welche durch Missachtung der Abhängigkeiten entstehen.

Die sich aus den Kernteilen ergebenden Ansatzpunkte zur Fehleridentifikation und -modellierung werden schließlich zusammengefasst.

Kapitel 4

Fehlermodelle, -modellierung und -ermittlung

Sichten auf Fehler

Fehler sind unerwünschte Zustände eines Systems oder Ereignisse, die nicht dem gewollten Verhalten eines Systems entsprechen (siehe Abschnitt 2.1.1). Ein unerwünschter Zustand im Bereich der Bahntechnik ist z.B., dass sich auf einem Bahnübergang sowohl ein Fahrzeug, als auch ein Zug befinden (siehe [OR04]). Ein Fehlerereignis ist z.B., dass sich eine Bahnschranke öffnet, obwohl der Zug den Übergang noch nicht passiert hat. Fehler haben immer einen Bezug zu einem System. Dieses wird in dieser Arbeit mit den Modellen aus Abschnitt 3.2 beschrieben. Die Modelle sind zum Einen Modelle des Verhaltens und zum Anderen Modelle der Implementierung, auf denen das Verhalten abgelegt ist. Die Beschreibung eines Fehlers bezieht sich so auf die Verhaltenssicht oder auf die Implementierungssicht.

Beitrag

Dieses Kapitel zeigt Fehlermodelle und Modellierungstechniken für diese Fehlermodelle, welche spezifisch auf die Anforderungen der Sicherheitsanalyse eingebetteter Systeme zugeschnitten sind. Zu den Modellierungstechniken werden anhand spezifischer Eigenschaften methodische Schritte angegeben, um die potentiellen Fehler zu ermitteln. Basis für die Modellierungstechniken ist das Fehlermodell von [Bre01] für reaktive Systeme. In diesem kann allgemein für stromgebundene Blackbox-Spezifikationen, wie auch für Automaten, eine Modifikation getrennt von der Soll-Verhaltensspezifikation durch Hinzufügen und Entfernen von Relationstupeln modelliert werden. Dieses Kapitel enthält im Wesentlichen zwei Beiträge. Der erste Beitrag ist die Erweiterung dieser Modellierungstechniken um zustandsgebundene Prädikate. Es werden auch Techniken zu den Sonderformen der Prädikate und Automaten angegeben. Diese Sonderformen sind die zu Ausgaben aufgelösten Gleichungen und die Betriebsmodi, deren Zustand nicht direkt abgelesen werden kann. Der zweite Beitrag ist die detaillierte Analyse der Kombinierbarkeit der Fehler. Abgerundet mit einem Datenmodell und einer Sammlung von typischen Fehlern der Implementierungsebene entsteht so ein ganzheitliches Fehlermodell für die Funktionssicherheitsanalyse eingebetteter Systeme.

Kapitel-übersicht

Abschnitt 4.1 beschreibt eine Modellierungsmöglichkeit, Fehler in Verhaltensmodellen reaktiver Systeme zu modellieren. Ein Fehler wird als eine Modifikation des Sollverhaltens modelliert. Diese können, entsprechend der Abhängigkeit zwischen den Modifikationen kombiniert werden.

Die Fehlermodelle können mit den Modellierungstechniken aus Abschnitt 4.2 spezifiziert werden. Es werden im Wesentlichen verschiedene Darstellungen der Fehlermodelle vorgestellt, welche auf spezifische Aspekte der Beschreibung von Fehlern zugeschnitten sind. Zu diesen Aspekten gehören die Modellierungstechniken aus Abschnitt 3.3, mit denen das Soll-Verhalten spezifiziert wurde, wie auch die Arten der Fehler, die passend zu den Sichten der Techniken modelliert werden. Es werden allgemein Fehlerbeschreibungen zu Blackbox-Spezifikationen und Automaten definiert. Die Sonderformen der Modifikationen für zu Ausgaben aufgelösten Gleichungen und für Betriebsmodi-Automaten werden eigens behandelt, da diese in den Fallstudien vorwiegend vorgefunden wurden. Für alle Fehlerarten wird zusätzlich betrachtet, wie diese geeignet ermittelt und miteinander kombiniert werden können, um Mehrfachfehler und zusammengesetzte Fehlerbeschreibungen zu modellieren.

Die Verhaltensmodifikationen basieren auf Fehlern, welche in der Implementierungsebene verursacht wurden. Abschnitt 4.3 präsentiert ein einfaches Datenmodell, um informelle Fehlerbeschreibungen der Implementierungsebene festzuhalten. Weiter werden Methoden der Fehlerermittlung auf Implementierungsebene skizziert. Abschnitt 4.4 skizziert Möglichkeiten, die Modifikationen mit Modifikationskomponenten in Verhaltensmodelle zu induzieren.

Inhalt des Abschnitts 4.5 ist eine kleine Sammlung von Fehlern, welche in der Literatur und den Fallstudien zu eingebetteten Systemen vorgefunden wurden und demonstriert so die Anwendbarkeit der in diesem Kapitel vorgestellten Modellierungstechniken.

4.1 Fehlermodellierung auf Verhaltensebene

Fehlverhalten

Grundlage für die Fehlermodellierung auf der Verhaltensebene ist die Beschreibung des Systems. Diese Beschreibung des Verhaltens eines System S ist durch die Relation R_S gegeben, welche aus Eingabe-Ausgabe-Tupeln besteht. Tritt in einem System ein Verhaltensfehler auf, so ändert sich entsprechend das Verhalten von S. Wann ein Fehler beobachtbar auftritt, kann spezifisch festgelegt werden. Er kann zu verschiedenen Zeitpunkten auftreten, sicher auftreten oder nur eventuell auftreten (siehe Abschnitt 2.1.1). Für ein System S ist ein Fehler die Abweichung vom Sollverhalten. Ein fehlerbehaftetes System S^F ist ein System S in dem Fehler, also die Abweichungen, potentiell oder sicher enthalten sind. Potentiell bedeutet, dass das Verhalten von S^F in eine Teilmenge von R_S und eine Menge R_S^F, in der die Fehler sicher auftreten, disjunkt aufgeteilt werden kann. Das *Fehlverhalten* R_{S^F} eines Systems ist das Verhalten eines Systems, bei dem Fehler (im Sinne einer Abweichung vom Sollverhalten) unter Berücksichtigung der möglichen Auftritts-Zeitpunkte integriert sind.

Ist ein Fehler nicht absolut sicher zu einem Zeitpunkt mit einem festen Verhalten beschrieben, so führt dies zu einem Nichtdeterminismus im fehlerbehafteten System. Es gilt:

Definition 4.1.1 (Fehlverhalten)
Das *Fehlverhalten* R_{S^F} eines Systems ist das verbleibende Sollverhalten $R_{S'}$ vereinigt mit den fehlerhaften Verhaltenstupeln R_S^F. Es gilt:

$$R_{S^F} = R_{S'} \cup R_S^F \text{ mit } R_{S'} \subseteq R_S \quad \lrcorner$$

Fehlverhalten als Modifikation

Statt dem Fehlverhalten R_{S^F} eines Systems kann eine *Modifikation* des Verhaltens eines Systems modelliert werden, welche die Änderung ist, die aus einem korrektem System S ein fehlerbehaftetes System S^F macht. Eine Modifikation $\mathcal{M}$ des Verhaltens ist die Hinzunahme und das Entfernen von Tupeln aus der Relation. Betrachtet man die obige Definition des Fehlverhaltens, so führt das Entfernen von Tupeln dazu, dass aus der Relation R_S die Relation R'_S wird. Die Hinzunahme von Tupeln ergibt die Menge R_S^F. Diese Betrachtung eines Fehlers ist eine Erweiterung zu den Fehlerdefinitionen, die nur die Hinzunahme von Fehlertupeln beachten (z.B. [SSL+95]). Ein modifiziertes System wird mit $S\Delta\mathcal{M}$ beschrieben. Nach einer Modifikation muss immer noch für jede mögliche Eingabe eine Ausgabe definiert sein. Die Menge der hinzugefügten Tupel wird mit F und die Menge der entnommenen Tupel mit E bezeichnet. Nach [Bre01] ist eine Modifikation wie folgt definiert:

Definition 4.1.2 (Modifikation & Modifikationsoperator)
R_S, E_S und F_S seien Relationen über der Schnittstelle (I_S, O_S),
also $R_S, E_S, F_S \in REL_S$. Die Menge aller Modifikationen ist:

$$MOD_S \stackrel{def}{=} REL_S \times REL_S$$

Eine *Modifikation* $\mathcal{M}_S$ ist ein Tupel aus einer Menge E_S und F_S:

$$\mathcal{M}_S \stackrel{def}{=} (E_S, F_S) \text{ mit } \mathcal{M}_S \in MOD_S$$

Sie modifiziert mit dem *Modifikationsoperator* Δ das Verhalten so, dass gilt:
$\Delta : REL_S \times MOD_S \to REL_S$

$$R_S\Delta\mathcal{M}_S \stackrel{def}{=} (R_S \setminus E_S) \cup F_S \quad \lrcorner$$

Wird mit $R_S\Delta\mathcal{M}_S$ ein Fehlverhalten definiert, so gilt: $R_{S^F} = R_S\Delta\mathcal{M}_S$.

Einschränkung der Beschreibung

Die Beschreibung eines Fehlverhaltens mittels einer Modifikation im Sinne der oben stehenden Definition schränkt die Möglichkeiten der Fehlerbeschreibungen ein. In kleinster Einheit kann ein Fehler ein falscher Zustand oder ein unerwünschtes Ereignis sein (siehe Abschnitt 2.1.1). Eine Modifikation wird auf das System angewendet, bevor die Ausführung beginnt. Entsprechend betrifft eine Modifikation die gesamte Ausführungszeit. Ein Fehler im Sinne eines einzelnen falschen Zustands oder fehlerhaften Ereignisses zu einem Zeitpunkt t kann nur dann beschrieben werden, wenn auch die unbedingten Folgen in der Ausführungszeit mit berücksichtigt werden. Abbildung 4.1 veranschaulicht diese Einschränkung. Gegeben sei eine Modifikation, die

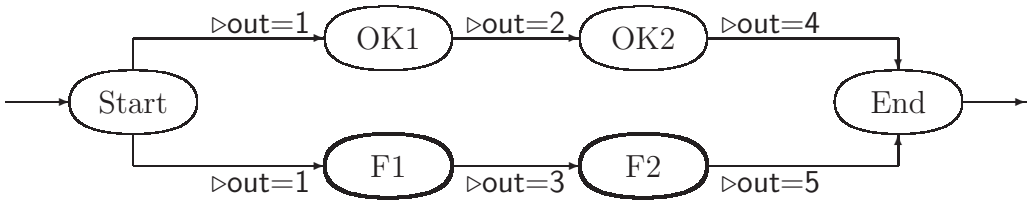


Abb. 4.1: Einschränkung bei der Beschreibung mit Modifikationen

die Zustandsfolge des Automaten auf $Start, F1, F2, End$ ändert. Würde man als kleinste Einheit eines Fehlers einen unerwünschten Zustand betrachten, so wäre ein Fehler der Zustand $F1$. Ein weiterer Fehler wäre der Zustand $F2$, welcher ein unbedingter Folgefehler des Zustandes $F1$ ist. Diese unbedingte Abhängigkeit zwischen $F1$ und $F2$ ist mit Modifikationen nur dann ausdrückbar, wenn sie in der gleichen Modifikation vorkommen. Die Beschreibung von Fehlern als Modifikationen ist also nicht geeignet, um unbedingte Ursache-Wirkungs-Zusammenhänge an der gleichen Schnittstelle eines Systems zu analysieren. Hinsichtlich einer Produkt-FMEA und -FTA, welche Folgefehler entlang der Systemhierarchie untersuchen (siehe Abschnitt 2.3), ist allerdings diese Art der Fehlerbeschreibung passend zugeschnitten: Der Fehlermodellierer muss unbedingte Wirkungen an der gleichen Schnittstelle berücksichtigen und die Fehlerbeschreibungen an den verschiedenen Schnittstellen entlang der Systemhierarchie können separat beschrieben werden. Auch von der Mächtigkeit ist diese Art der Fehlerbeschreibung ausreichend. Es kann sowohl mit dem neutralen Element $(\varnothing, \varnothing)$ keine Veränderung des Verhaltens stattfinden, als auch für eine bestehende Relation R_S mit (R_S, R_S^F) die komplette Relation ersetzt werden. Für die Betrachtung von Fehlern, die an der gleichen Schnittstelle hinsichtlich Ursache und Wirkung untersuchbar sein sollen, sei auf [Thu04] verwiesen.

einfache Kombination

Modifikationen können kombiniert werden. Obwohl die Kombination von Modifikationen formal immer gleich ist, gibt es methodisch zwei mögliche Interpretationen der Kombination von Modifikationen. Die Unterschiede zwischen den Möglichkeiten sind ähnlich denen der Verknüpfungen bei der Fehler-Möglichkeit- und Einfluss-Analyse (siehe Abschnitt 2.3) und der Fehlerbaumanalyse (siehe Abschnitt 2.4).

- Im Sinne der *Und-Verknüpfung* bedeutet die Kombination, dass in dem resultierenden Fehlverhalten zwei Fehler unabhängig voneinander und damit auch gleichzeitig auftreten können. Auf diese Weise sind Mehrfachfehler als eine Menge von Teilfehlern darstellbar, welche bei der Fehlerbaumanalyse als Menge hin auf ihre Wirkung untersucht werden.
- Im Sinne der *Oder-Verknüpfung* bedeutet die Kombination, dass entweder der eine oder der andere Fehler vorhanden ist, nicht aber beide. So lässt sich eine komplexe Modifikation aus verschiedenen Alternativen zusammenfügen, wodurch die Beschreibung eines Fehlers vereinfacht werden kann (siehe Abschnitt 4.2.2).

Prinzipiell werden zwei Modifikationen kombiniert, in dem auf ein System zuerst eine Modifikation und dann auf dieses bereits modifizierte System die zweite Modifikation angewendet wird.

Definition 4.1.3 (einfache Modifikationskombination)
Die *einfache Modifikationskombination* + ist die sequentielle Anwendung von Modifikationen.

$$+ : MOD_S \times MOD_S \rightarrow MOD_S$$
$$\mathcal{M}_S^{(1)} + \mathcal{M}_S^{(2)} \stackrel{def}{=} (E_S^{(1)} \cup E_S^{(2)}, (F_S^{(1)} \setminus E_S^{(2)}) \cup F_S^{(2)})$$

Nach [Bre01] gilt:
$R_S \Delta(\mathcal{M}_S^{(1)} + \mathcal{M}_S^{(2)}) = (R_S \Delta \mathcal{M}_S^{(1)}) \Delta \mathcal{M}_S^{(2)} = (((R_S \setminus E_S^{(1)}) \cup F_S^{(1)}) \setminus E_S^{(2)}) \cup F_S^{(2)}$ ⌟

Modifikations-unabhängigkeit

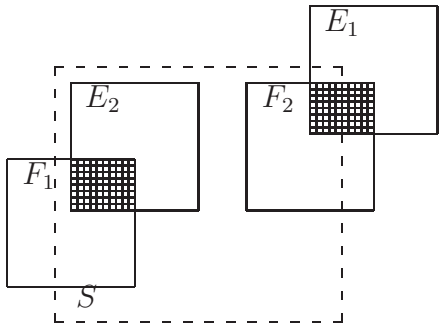


Abb. 4.2: Modifikations-Abhängigkeit

Abhängig von der Bedeutung (Oder- bzw. Und-Verknüpfung) ist eine beliebige Kombination von Modifikationen, die sich sowohl in dem Entfernen, wie auch in dem Hinzufügen von möglichen Ausführungspfad-Tupeln in dem Verhalten eines Systems zeigen, nur bedingt möglich. Die Modifikation eines bereits modifizierten Systems kann dazu führen, dass Effekte der vorhergehenden Modifikation wieder aufgehoben werden. Es können zum Beispiel Pfad-Tupel, die in der vorhergehenden Modifikation gelöscht wurden, in der folgenden Modifikation wieder hinzugefügt werden. Abbildung 4.2 zeigt die Mengen zweier Modifikationen, wobei die zweite Modifikation einige, in der ersten Modifikation entfernte Pfad-Tupel, wieder hinzufügt (siehe karierter Bereich). In den meisten Fällen ist eine Abhängigkeit zwischen Modifikationen nicht erwünscht, da mit ihr Probleme in der Reihenfolge der Anwendung entstehen. Es soll gelten, dass die hinzugefügten Pfade bestehen bleiben und die gelöschten Pfade entfernt bleiben. Nach [Bre01] ist die Modifikationsunabhängigkeit wie folgend definiert:

Definition 4.1.4 (Modifikationsunabhängigkeit)
Die *Modifikationsunabhängigkeit* $\stackrel{dep}{\#}$ zweier Modifikationen $\mathcal{M}_S^{(1)}$ und $\mathcal{M}_S^{(2)}$ ist gegeben, wenn gilt:

$$\stackrel{dep}{\#}: MOD_S \times MOD_S \rightarrow BOOL$$
$$\mathcal{M}_S^{(1)} \stackrel{dep}{\#} \mathcal{M}_S^{(2)} \stackrel{def}{=} \begin{array}{l} (\mathcal{M}_S^{(1)} = (E_S^{(1)}, F_S^{(1)}) \\ \wedge \mathcal{M}_S^{(2)} = (E_S^{(2)}, F_S^{(2)}) \\ \wedge E_S^{(1)} \cap F_S^{(2)} = \emptyset \\ \wedge E_S^{(2)} \cap F_S^{(1)} = \emptyset) \end{array}$$

Damit gilt: $(\mathcal{M}_S^{(1)} \stackrel{dep}{\#} \mathcal{M}_S^{(1)}) \Rightarrow ((\mathcal{M}_S^{(1)} + \mathcal{M}_S^{(2)}) = (E_S^{(1)} \cup E_S^{(2)}, F_S^{(1)} \cup F_S^{(2)}))$ ⌟

koordinierte Kombination

Die Abhängigkeit zwischen Modifikationen ist nicht immer vermeidbar. Die existierenden Ansätze umgehen dieses Problem in dem sie entweder die Unabhängigkeit fordern [Bre01], die Modifikationen auf das Hinzufügen von Tupeln (F-Mengen) beschränken [ORS05, SSL+96] oder die fehlerhaften Systeme bereits in modifizierter Form vorliegen müssen [Str06]. Die hier vorgeschlagene Alternative ist, den Anteil der Schnittmengen an der resultierenden Modifikation zu bestimmen und entsprechend eine Abschätzung des resultierenden Fehlverhaltens zu erzielen. Ein Beispiel ist die Kombination der Fehler-Basisklassen in Abschnitt 4.2.2. Der Verbleib des abhängigen Anteils kann anhand zweier Mechanismen gesteuert werden. Ein Mechanismus ist die Reihenfolge der Modifikationen, die das Ergebnis bestimmt. In der Reihenfolge weiter hinten stehende Modifikationen werden von weiter vorne stehenden Modifikationen nicht beeinflusst. Überschneiden sich sowohl $E_S^{(1)}$ und $F_S^{(2)}$, sowie $E_S^{(2)}$ und $F_S^{(1)}$ (siehe Abbildung 4.2), so gibt es mit der einfachen Kombination nicht die Möglichkeit, dass beide F-Mengen oder beide E-Mengen erhalten bleiben. Mit einer einfachen Kombination kann immer nur eine Schnittmenge hinzugenommen werden und die andere ausgeschlossen werden. In diesem Falle bietet der Mechanismus der koordinierten Kombination Abhilfe, in der angegeben werden kann, welche Mengen überwiegen. Damit kann für eine Modifikationskombination eine obere Schranke mit einem maximalen Verhalten und eine untere Schranke mit einem minimalen Verhalten bestimmt werden.

Definition 4.1.5 (koordinierte Modifikationskombination)
Die *koordinierte Modifikationskombination* ist die Anwendung von Modifikationen, bei der wahlweise die Mengen E oder F erhalten bleiben.

$$+_F : MOD_S \times MOD_S \to MOD_S$$
$$\mathcal{M}_S^{(1)} +_F \mathcal{M}_S^{(2)} \stackrel{def}{=} (E_S^{(1)} \cup E_S^{(2)}, F_S^{(1)} \cup F_S^{(2)})$$

$$+_E : MOD_S \times MOD_S \to MOD_S$$
$$\mathcal{M}_S^{(1)} +_E \mathcal{M}_S^{(2)} \stackrel{def}{=} (E_S^{(1)} \cup E_S^{(2)}, (F_S^{(1)} \setminus E_S^{(2)}) \cup (F_S^{(2)} \setminus E_S^{(1)}))$$

Für das Verhalten gilt:

$$R_S\Delta(\mathcal{M}_S^{(1)} +_F \mathcal{M}_S^{(2)}) = (((R_S \setminus E_S^{(1)}) \setminus E_S^{(2)}) \cup F_S^{(1)}) \cup F_S^{(2)}$$

$$R_S\Delta(\mathcal{M}_S^{(1)} +_E \mathcal{M}_S^{(2)}) = (((R_S \setminus E_S^{(1)}) \cup F_S^{(1)}) \setminus E_S^{(2)}) \cup (F_S^{(2)} \setminus E_S^{(1)}) \quad \lrcorner$$

minimale Modifikation

Die Mengen E_S und F_S einer Modifikation $\mathcal{M}_S$ können sich überschneiden. Da die Elemente der Schnittmenge zuerst entfernt und dann wieder hinzugefügt werden, hat die Schnittmenge die gleiche Wirkung, als wäre sie nur in der Menge F enthalten. Für alle Modifikationen $\mathcal{M}^{(1)}$ und $\mathcal{M}^{(2)}$, für die gilt, dass $E_S^{(1)} \setminus F_S^{(1)}$ gleich $E_S^{(2)} \setminus F_S^{(2)}$ ist und $F_S^{(1)}$ gleich $F_S^{(2)}$ ist, ist die Wirkung auf ein System identisch. Bei gleicher Wirkung sind die Mengen E und F minimal, wenn sie sich nicht überschneiden.

Definition 4.1.6 (Modifikationsminimalität)
Die *Modifikationsminimalität* $?_{min}$ einer Modifikation $\mathcal{M}_S$ ist gegeben, wenn gilt:

$?_{min} : MOD_S \rightarrow BOOL$

$$\mathcal{M}_S ?_{min} \overset{def}{\Leftrightarrow} E_S \cap F_S = \emptyset \text{ mit } (E_S, F_S) \in MOD_S$$

Eine Modifikation wird mit dem Minimierungsoperator $\downarrow_{min}$ wie folgt minimiert:

$\downarrow_{min}: MOD_S \rightarrow MOD_S$

$$\mathcal{M}_S \downarrow_{min} \overset{def}{=} (E_S \setminus F_S, F_S)$$ ⌟

Zusammenfassung

Dieser Abschnitt definiert somit die Möglichkeit, Fehlverhalten als Modifikationen zu beschreiben, und diese zu kombinieren. Die Modellierung der Mengen E und F der Modifikationen wird in Abschnitt 4.2 behandelt. Abbildung 4.2 zeigt eine Möglichkeit, wie sich Modifikationen überschneiden können. Weitere Arten der Überschneidung, die nicht für die Kombination von Modifikationen sondern zu deren Analyse dienen, werden in Kapitel 5 behandelt. Die in diesem Abschnitt gegebenen Abhängigkeiten der Überschneidung der Menge $E^{(1)}$ mit der Menge $F^{(2)}$ und bzw. oder der Menge $E^{(2)}$ mit der Menge $F^{(1)}$ genügen zur Beschreibung von Konflikten und damit zum im nächsten Abschnitt gegebenen Aufbau komplexer Fehlerbilder.

4.2 Modellierungstechniken für Fehler

Abschnittsübersicht

Im Abschnitt 4.1 wurde gezeigt, wie mit Relationen das Verhalten eines Systems geändert werden kann. Es wurde des Weiteren gezeigt, wie Modifikationen kombiniert werden können. Dieser Abschnitt widmet sich Modellierungstechniken, mit denen die Relationen der Modifikationen spezifiziert werden können. Die Techniken sind jeweils auf die Modellierungstechniken des Systemverhaltens zugeschnitten. Abbildung 4.3 veranschaulicht die vier Bezugspunkte für Modifikationen. Grundlegend beschreibt Abschnitt 4.2.1 die *Systemmodifikationen*, welche sich auf das Systemverhalten als Ganzes beziehen. Im Rahmen der anforderungsbasierten Entwicklung beschreibt dieser Abschnitt auch die *Anforderungsmodifikationen*, welche sich auf Teilprädikate beziehen, die Anforderungen widerspiegeln. Für die Datenfluss-Sicht bei aufgelösten Gleichungen sind in Abschnitt 4.2.2 die *Ausgabemodifikationen* zur Modellierung von Abweichungen der Sollausgaben definiert. Das algorithmische Verhalten kann in Abschnitt 4.2.3 mit *Transitionsmodifikationen* durch gezielte Manipulation der Transitionen geändert werden. Schließlich wird die Modifikation von Betriebsmodi-Automaten, welche beide Sichten verbinden, in Abschnitt 4.2.4 erörtert.

Abhängigkeit vom Sollverhalten

Die Beschreibungen von Fehlverhalten und Modifikationen können abhängig oder unabhängig von dem Verhalten des fehlerfreien Systems sein. Eine von dem fehlerfreien Verhalten R_S *unabhängige Modifikationsbeschreibung* definiert zwei Mengen E und F, die für beliebige R_S gleich sind. Eine von dem fehlerfreien Verhalten R_S *unabhängige Fehlverhaltensbeschreibung* definiert eine Menge $\mathcal{R}_S^F$ mit

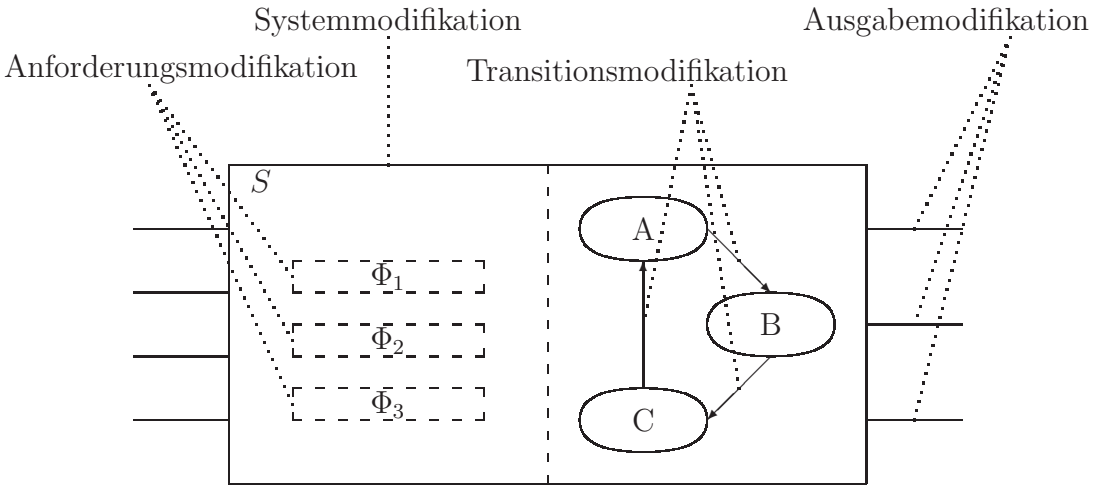


Abb. 4.3: Bezugspunkte zur Fehlerbeschreibung mit Modifikationen

$$R_S^F \subseteq \mathcal{R}_S^F \subseteq (R_S \cup R_S^F)$$

die alle fehlerhaften Tupel enthält, und für alle $R_S \subseteq (\mathcal{F}_{I_S} \times \mathcal{F}_{O_S}) \setminus \mathcal{R}_S^F$ gleich bleibt. Der in der Einleitung dieses Kapitels genannte Fehlerzustand, in dem sich sowohl Zug wie auch Fahrzeug auf dem Übergang befinden, ist unabhängig von dem fehlerfreien System beschrieben. Die Menge $\mathcal{R}_S^F$ enthält alle Tupel, in denen der beschriebene Fehlerzustand auftritt. Die einzige Abhängigkeit zum Sollverhalten ist die Invariante, dass die Schnittmenge aus dem Fehlverhalten und den fehlerfreien Verhalten leer ist, also $R_S \cap \mathcal{R}_S^F = \emptyset$. In einer FMEA (siehe Abschnitt 2.3) oder FTA (siehe Abschnitt 2.4) sind die Fehlverhaltensbeschreibungen zu Verhaltensbedingungen unabhängige Fehlverhaltensbeschreibungen, da hier die Negation der Bedingung die Menge $\mathcal{R}_S^F$ beschreibt, welche auch feiner unterteilt werden kann. Unabhängige Modifikations- und Fehlverhaltensbeschreibungen sind z.B. in [OR04] zu finden. Alternativ bietet sich die Beschreibung des Unterschieds vom fehlerbehafteten zum fehlerfreien System an, wie sie auch in [Str04] vorgestellt wird. Dies wird an dem Werteverlauf in Abbildung 4.4 verdeutlicht. Der Werteverlauf des fehlerfreien Systems ist als durchgehende Linie dargestellt. Der fehlerhafte Verlauf ist als gestrichelte Linie eingezeichnet. Dieser gestrichelte Verlauf gilt als fehlerhaft, wenn er unter den Bedingungen auftritt, in denen die durchgehende Linie auftreten sollte. Unter anderen Bedingungen ist er korrekt. Um nicht bei jedem Fehler diese Bedingungen beschreiben zu müssen, kann er relativ als eine Abweichung von dem fehlerfreien Werteverlauf definiert werden. Auf diese Weise lassen sich Fehlerklassen bilden, deren Beschreibung der Abweichung die gemeinsame Eigenschaft ist. In Abbildung 4.4 ist z. B. der Wert in der Zeit von 4 bis 7 um 3 zu hoch. Diese Charakteristik kann Fehler relativ zu beliebigen fehlerfreien Werteverläufen umfassen. In einer FMEA oder FTA sind die Beschreibungen der Abweichungen von den Ausgaben von dem Sollverhalten R_S eines Systems abhängig. Die Menge der abweichenden Tupel R_S^F hängt von dem Sollverhalten R_S ab.

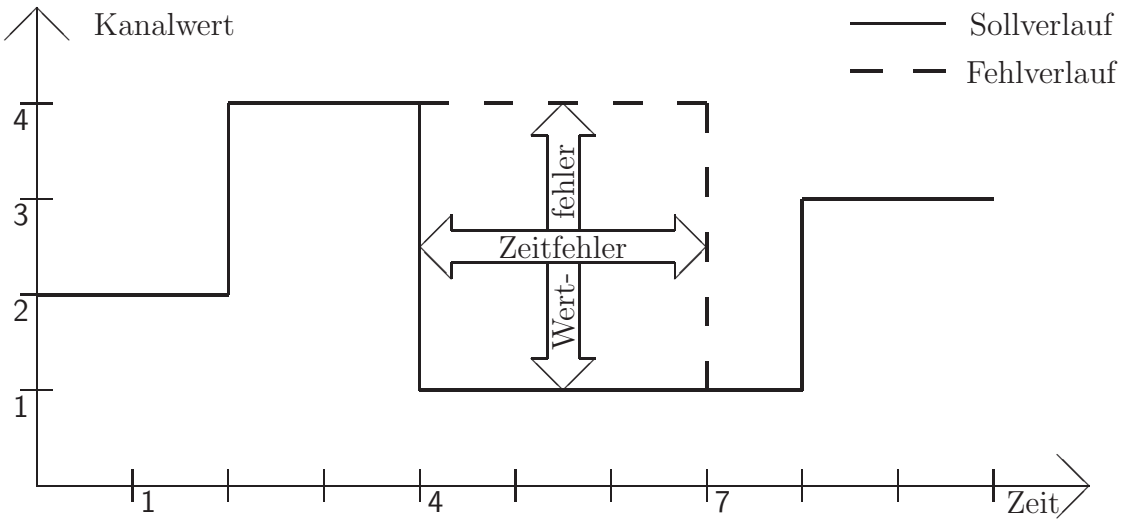


Abb. 4.4: Interpretation von Fehlern

Datenfluss- vs. Zustandssicht

Die Abweichung von fehlerhaften Systemen zu korrekten Systemen kann auf verschiedene Weisen beschrieben werden. Wesentlich ist hierbei die Sicht auf das System (siehe Abschnitt 3.1). Betrachtet man den Datenfluss zwischen Komponenten, so ist z.B. die Abweichung des Wertes eine relevante Eigenschaft des Fehlers. Die gesteuerte Größe wird in diesem Sinne für eine Zeit in die falsche Richtung gesteuert (siehe Abbildung 4.4). Betrachtet man den Werteverlauf aus der Zustandssicht, so ändert sich nicht die Wertfolge, sondern es verschiebt sich der Zeitpunkt des Übergangs von dem Wert 4 zu dem Wert 1. Der Zustandsübergang kommt in diesem Sinne zu spät. Die Art der Beschreibung eines Fehlers bezieht sich also auf die Sicht auf das System.

4.2.1 Modifikation von Black-Box-Spezifikationen

Wird ein System S mit einem Prädikat $[\![S]\!]$ beschrieben (Abschnitt 3.3.1), so kann die entsprechende Modifikation mit Prädikaten folgendermaßen aussehen: Zum einen kann $[\![S]\!]$ durch Hinzunahme von weiteren Teilprädikaten verschärft werden, also die Anzahl der Elemente in der Relation R_S verringert werden, und zum anderen kann $[\![S]\!]$ durch Ablösen bestehender Teilprädikate entschärft werden, also die Anzahl der Elemente in R_S erhöht werden. Gleichermaßen kann für ein Prädikat $[\![S]\!]^{op}$ durch Hinzufügen und Entfernen von Teilprädikaten die Menge der gültigen Wertkombinationen der Kanäle relativ zu den Zeitpunkten t und damit die Relation R_S vergrößert oder verkleinert werden. Angelehnt an [Bre01] gilt:

Definition 4.2.1 (Systemmodifikation, freier Modifikationsoperator)
$[\![S]\!]$, $[\![F_S]\!]$ und $[\![\overline{E}_S]\!] = \neg[\![E_S]\!]$ seien Prädikate für ein System S mit der Schnittstelle (I_S, O_S). Eine *Systemmodifikation* $\mathcal{M}_S = (E_S, F_S)$ entspricht bei Prädikaten einer Modifikation

$$[\![\mathcal{M}_S]\!] = ([\![\overline{E}_S]\!], [\![F_S]\!]).$$

Die Menge aller Modifikationen ist:

$$PRED^{Stream}_{MOD_S} \stackrel{def}{=} PRED^{Stream}_{REL_S} \times PRED^{Stream}_{REL_S}$$

Eine Modifikation $[\![\mathcal{M}_S]\!]$ modifiziert mit dem *freien Modifikationsoperator* Δ ein System, welches mit $[\![S]\!]$ beschrieben ist, so dass gilt:

$$\Delta : PRED^{Stream}_{REL_S} \times PRED^{Stream}_{MOD_S} \rightarrow PRED^{Stream}_{REL_S}$$
$$[\![S]\!]\Delta[\![\mathcal{M}_S]\!] \stackrel{def}{=} ([\![S]\!] \wedge [\![\overline{E}_S]\!]) \vee [\![F_S]\!]$$

Für operationelle Prädikate gilt:

$$\Delta : PRED^{op}_{REL_S} \times PRED^{op}_{MOD_S} \rightarrow PRED^{op}_{REL_S}$$
$$[\![S]\!]^{op}\Delta[\![\mathcal{M}_S]\!]^{op} \stackrel{def}{=} ([\![S]\!]^{op} \wedge [\![\overline{E}_S]\!]^{op}) \vee [\![F_S]\!]^{op}$$ ⌟

Bei dieser Definition einer Modifikation ist das neutrale Element die Modifikation $(true, false)$. Die Änderung aller Formeln ist mit der Modifikation $(\neg[\![S]\!], \Phi)$ bzw. $(\neg[\![S]\!]^{op}, \Phi)$ möglich.

Beispiel 4.2.1 (Freie Modifikation einer Spezifikation)
Gegeben sei ein System S, welches den Lenkradwinkel i_{lrw} als Eingabe bekommt, und als Ausgabe den gewünschten Vorderradwinkel o_{vrw} berechnet. Für das System S gelte folgendes Prädikat $[\![S]\!]^{op} = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3)$. Die Teilprädikate Φ_1, Φ_2 und Φ_3 sind hierbei Verhaltensbedingungen.

$$\Phi_1 = (|o_{vrw} - \mathsf{delay}_{\langle o_{vrw}.0\rangle}(o_{vrw})| \leq 0.01)$$
$$\Phi_2 = (o_{vrw} \geq -0.3)$$
$$\Phi_3 = (o_{vrw} \leq 0.3)$$

Ein denkbarer Fehler ist eine fehlende Stromversorgung, welche zum Senden des Wertes 0 führt. Die zugehörige Modifikation lautet:

$$[\![\mathcal{M}^{(1)}_S]\!]^{op} = (true, o_{vrw} = 0).$$

Angewendet auf das System S ergibt sich so das Fehlverhalten:

$$[\![S]\!]^{op}\Delta[\![\mathcal{M}^{(1)}_S]\!]^{op} = ((\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \wedge true) \vee (o_{vrw} = 0)$$

Ein weiterer denkbarer Fehler ist ein Kurzschluss, welcher zum Senden des Wertes 255 führt. Die zugehörige Modifikation lautet:

$$[\![\mathcal{M}^{(2)}_S]\!]^{op} = (true, o_{vrw} = 255).$$

Angewendet auf das System S ergibt sich so das Fehlverhalten:

$$[\![S]\!]^{op}\Delta[\![\mathcal{M}^{(2)}_S]\!]^{op} = ((\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \wedge true) \vee (o_{vrw} = 255)$$ ⌟

Das Auftreten eines Fehlers kann über die Zeit variieren. Ein permanenter Fehler (z.B. der Bruch eines mechanischen Teiles) kann andere Folgen zeigen, als ein temporär auftretender Fehler (z.B. eine EM-Störung). Hier können für die Fehlverhalten analog zu der Spezifikation des Systems Bedingungen beschrieben werden, die in Bezug auf die Zeit oder den vorherigen Zustand des Systems das weitere Auftreten des Fehlers bestimmen. Ein Beispiel, das Auftreten mittels Automaten zu beschreiben findet sich in [ORS05].

Auftreten der Fehler

Definition 4.2.2 (einfache Modifikationskombination)
Die *einfache Modifikationskombination* ist ein sequentielles Anwenden der einzelnen Modifikationen. Es gilt:

$$+ : PRED_{MOD_S}^{Stream} \times PRED_{MOD_S}^{Stream} \rightarrow PRED_{MOD_S}^{Stream}$$
$$[\![\mathcal{M}_S^{(1)}]\!] + [\![\mathcal{M}_S^{(2)}]\!] \stackrel{def}{=} ([\![\overline{E}_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!], ([\![F_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!]) \vee [\![F_S^{(2)}]\!])$$

Nach [Bre01] gilt:

$$\begin{aligned} [\![S]\!]\Delta([\![\mathcal{M}_S^{(1)}]\!] + [\![\mathcal{M}_S^{(2)}]\!]) &= ([\![S]\!]\Delta[\![\mathcal{M}_S^{(1)}]\!])\Delta[\![\mathcal{M}_S^{(2)}]\!] \\ &= ((([\![S]\!] \wedge [\![\overline{E}_S^{(1)}]\!]) \vee [\![F_S^{(1)}]\!]) \wedge [\![\overline{E}_S^{(2)}]\!]) \vee [\![F_S^{(2)}]\!] \end{aligned}$$

Gleiches gilt für operationelle Prädikate $[\![\ldots]\!]^{op}$ mit dem Operator $\oplus$:

$$\oplus : PRED_{MOD_S}^{op} \times PRED_{MOD_S}^{op} \rightarrow PRED_{MOD_S}^{op}$$
$$[\![\mathcal{M}_S^{(1)}]\!]^{op} \oplus [\![\mathcal{M}_S^{(2)}]\!]^{op} \stackrel{def}{=} ([\![\overline{E}_S^{(1)}]\!]^{op} \wedge [\![\overline{E}_S^{(2)}]\!]^{op}, ([\![F_S^{(1)}]\!]^{op} \wedge [\![\overline{E}_S^{(2)}]\!]^{op}) \vee [\![F_S^{(2)}]\!]^{op}) \quad \lrcorner$$

In der obigen Definition ist gegeben, dass die Kombination von operationell definierten Modifikationen genau wie die Kombination von auf Strömen definierten Modifikationen verläuft. Dies ist zwar so korrekt, allerdings sei hier darauf hingewiesen, dass die verschiedenen Notationen bei der Kombination zu verschiedenen Ergebnissen führen. Dieser Effekt hat Auswirkungen auf den Umgang mit den Modifikationen (z.B. bei der Abstraktion der Ursache-Wirkungs-Ermittlung in Abschnitt 5). Die Unterschiede werden im Folgenden verdeutlicht:

Semantik

Beispiel 4.2.2 (operativ versus strombasiert)
Gegeben sei ein boolesches System S mit dem operativen Prädikat

$$[\![S]\!]^{op} = (c = a \wedge b)$$

bzw. äquivalent mit dem strombasierten Prädikat

$$[\![S]\!] = (\forall t > 0 : c.t = a.t \wedge b.t).$$

Des Weiteren seien zwei Modifikationen $\mathcal{M}_S^{TsF}$ und $\mathcal{M}_S^{FsT}$ gegeben. Die operativen Beschreibungen hierzu seien:

$$[\![\mathcal{M}_S^{TsF}]\!]^{op} = (true, c = true) \text{ und } [\![\mathcal{M}_S^{FsT}]\!]^{op} = (true, c = false).$$

Strombasiert ist die Beschreibung dieser Modifikationen folgende:

$$[\![\mathcal{M}_S^{TsF}]\!] = (true, \forall t > 0 : c.t = (a.t \wedge b.t) \vee true) \text{ und}$$
$$[\![\mathcal{M}_S^{FsT}]\!] = (true, \forall t > 0 : c.t = (a.t \wedge b.t) \vee false)$$

Die strombasierte Kombination der beiden Modifikationen liefert:

$$\begin{aligned}&[\![\mathcal{M}_S^{TsF}]\!] + [\![\mathcal{M}_S^{FsT}]\!] \\ &\quad = (true, (\forall t > 0 : c.t = (a.t \wedge b.t) \vee true) \vee (\forall t > 0 : c.t = (a.t \wedge b.t) \vee false)) \\ &\quad = [\![\mathcal{M}_S^{TsF\_FsT}]\!]\end{aligned}$$

Die operative Kombination der Modifikationen ergibt hingegen die Bayessche Modifikation[1]:

$$\begin{aligned}[\![\mathcal{M}_S^{TsF}]\!]^{op} \oplus [\![\mathcal{M}_S^{FsT}]\!]^{op} &= (true, true) \\ &= [\![\mathcal{M}_S^{Bayes}]\!]^{op} \\ &\backsimeq [\![\mathcal{M}_S^{Bayes}]\!] = (true, true)\end{aligned}$$

Damit gilt:

$$([\![\mathcal{M}_S^{Tsf}]\!] + [\![\mathcal{M}_S^{FsT}]\!]) \not\backsimeq ([\![\mathcal{M}_S^{Tsf}]\!]^{op} \oplus [\![\mathcal{M}_S^{FsT}]\!]^{op})$$ ⌟

Stromtreue

Um auch bei operativ definierten Modifikationen die Kombination der Modifikationen im Sinne der Vereinigung der Strommengen E und F (siehe Definition 4.1.3) zu ermöglichen wird der Kombinationsoperator erweitert:

Theorem 4.2.1 (stromtreue operative Modifikationskombination)
Die *stromtreue operative Modifikationskombination* ist wie folgend (mit der Hilfsvariable l) definiert:

$$\begin{aligned}&+ : PRED_{MOD_S}^{op} \times PRED_{MOD_S}^{op} \rightarrow PRED_{MOD_S}^{op} \\ &\qquad [\![\mathcal{M}_S^{(1)}]\!]^{op} + [\![\mathcal{M}_S^{(2)}]\!]^{op} \stackrel{def}{=} \\ &\qquad\qquad (\mathsf{false}, \neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge (l = 1) \wedge [\![S]\!]^{op} \Delta [\![\mathcal{M}_S^{(1)}]\!]^{op} \\ &\qquad\qquad\quad \vee \neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge (l = 2) \wedge [\![S]\!]^{op} \Delta [\![\mathcal{M}_S^{(2)}]\!]^{op})\end{aligned}$$ ⌟

Methodische Festlegung

Möchte man in einer Sicherheitsanalyse einer Komponente eine Menge von Fehlern zuordnen, so ist, insbesondere für eine Betrachtung von Mehrfachfehlern zu überprüfen, wie diese kombiniert werden. Anhand dessen kann der Umgang mit den Fehlern angepasst werden. Verzichtet man auf die Stromtreue, so ist zu bedenken, dass bei jedem Rechenschritt ein Wechsel zwischen den Modifikationen möglich ist, und so bei unvorsichtiger Modellierung die Kombination einen größeren Fehler ergibt, als gewünscht. Umgekehrt ist zu beachten, dass bei einer stromtreuen Kombination alle Fehlerbilder explizit modelliert werden müssen, die über die Zeit einen Wechsel zwischen den einzelnen Fehlern beinhalten.

Unabhängigkeit

Um die Unabhängigkeit von Fehlern und somit die Effekte zwischen Fehlern auszuschließen, ist es notwendig, dass die verschärfenden Einschränkungen eines Fehlers nicht durch die entschärfenden Formeln des anderen Fehlverhaltens wieder aufgehoben werden. Eine (von der Reihenfolge) unabhängige Kombination der Fehler ist möglich, wenn gilt (siehe [Bre01], S. 104):

[1] Eine Bayessche Modifikation entspricht der Modifkation $[\![\mathcal{M}_S^{Bayes}]\!] = (true, true)$

Definition 4.2.3 (Modifikationsunabhängigkeit)
Die *Modifikationsunabhängigkeit* $\overset{dep}{\#}$ zweier Modifikationen $[\![\mathcal{M}_S^{(1)}]\!]$ und $[\![\mathcal{M}_S^{(2)}]\!]$ ist gegeben, wenn gilt:

$$\overset{dep}{\#}: PRED_{MOD_S}^{Stream} \times PRED_{MOD_S}^{Stream} \rightarrow BOOL$$

$$\begin{aligned}[\![\mathcal{M}_S^{(1)}]\!] \overset{dep}{\#} [\![\mathcal{M}_S^{(2)}]\!] \quad \overset{def}{=} \quad & ([\![\mathcal{M}_S^{(1)}]\!] = ([\![\overline{E}_S^{(1)}]\!], [\![F_S^{(1)}]\!]) \\ & \wedge [\![\mathcal{M}_S^{(2)}]\!] = ([\![\overline{E}_S^{(2)}]\!], [\![F_S^{(2)}]\!]) \\ & \wedge [\![F_S^{(1)}]\!] \Rightarrow [\![\overline{E}_S^{(2)}]\!] \\ & \wedge [\![F_S^{(2)}]\!] \Rightarrow [\![\overline{E}_S^{(1)}]\!])\end{aligned}$$

Damit gilt:

$$([\![\mathcal{M}_S^{(1)}]\!] \overset{dep}{\#} [\![\mathcal{M}_S^{(2)}]\!]) \Rightarrow (([\![\mathcal{M}_S^{(1)}]\!] + [\![\mathcal{M}_S^{(2)}]\!]) = ([\![\overline{E}_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!], [\![F_S^{(1)}]\!] \vee [\![F_S^{(2)}]\!]))$$

Gleiches gilt für operationelle Prädikate $[\![\ldots]\!]^{op}$. ⌟

Beispiel 4.2.3 (Einfache Kombination von Modifikationen)
Da beide Fehler $[\![\mathcal{M}_S^{(1)}]\!]$ und $[\![\mathcal{M}_S^{(2)}]\!]$ aus Beispiel 4.2.1 in dem System S auftreten können, werden sie kombiniert. Beide Modifikationen sind unabhängig, da die jeweiligen $[\![E_S^{(x)}]\!]$-Bedingungen *true* sind, und somit immer gilt $[\![F_S^{(x)}]\!] \Rightarrow [\![E_S^{(y)}]\!]$. Die kombinierte Modifikation ist so:

$$\begin{aligned}[\![\mathcal{M}_S^{(3)}]\!]^{op} &= ([\![\mathcal{M}_S^{(1)}]\!]^{op} + [\![\mathcal{M}_S^{(2)}]\!]^{op}) \\ &= (true, (o_{vrw} = 0) \vee (o_{vrw} = 255))\end{aligned}$$

⌟

Abschätzung

Wie schon in vorhergehendem Abschnitt erwähnt, kann die Unabhängigkeit zwischen Fehlern nicht immer gewährleist werden. In diesen Fällen muss explizit überdacht werden, wie mit der Kombination umgegangen wird. Hier bietet sich die Abschätzung einer oberen und unteren Grenze des resultierenden Verhaltens an, die mit der koordinierten Modifikationskombination erreicht werden kann.

Definition 4.2.4 (koordinierte Modifikationskombination)
Die *koordinierte Modifikationskombination* ist das Zusammenführen von Modifikationen, bei denen entweder die Formeln $[\![\overline{E}]\!]$ oder $[\![F]\!]$ überwiegen. Es gilt:

$$+_F : PRED_{MOD_S}^{Stream} \times PRED_{MOD_S}^{Stream} \rightarrow PRED_{MOD_S}^{Stream}$$

$$[\![\mathcal{M}_S^{(1)}]\!] +_F [\![\mathcal{M}_S^{(2)}]\!] \overset{def}{=} ([\![\overline{E}_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!], [\![F_S^{(1)}]\!] \vee [\![F_S^{(2)}]\!])$$

$$+_E : PRED_{MOD_S}^{Stream} \times PRED_{MOD_S}^{Stream} \rightarrow PRED_{MOD_S}^{Stream}$$

$$[\![\mathcal{M}_S^{(1)}]\!] +_E [\![\mathcal{M}_S^{(2)}]\!] \overset{def}{=} ([\![\overline{E}_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!], ([\![F_S^{(1)}]\!] \wedge [\![\overline{E}_S^{(2)}]\!]) \vee ([\![F_S^{(2)}]\!] \wedge [\![\overline{E}_S^{(1)}]\!]))$$

Gleiches gilt für operationelle Prädikate $[\![\ldots]\!]^{op}$. ⌟

Beispiel 4.2.4 (koordinierte Kombination von Modifikationen)
Für das System S aus Beispiel 4.2.1 gibt es Fehler bei der Trimmung. Sie führen jeweils dazu, dass sich der Wertebereich verschiebt, die Steigung der Funktion aber trotzdem begrenzt ist. Es gilt:

$$[\![\mathcal{M}_S^{(4)}]\!] = (o_{vrw} \geq -0.1, (o_{vrw} \leq 0.4) \wedge \Phi_1 \wedge (o_{vrw} \geq -0.1))$$

und

$$[\![\mathcal{M}_S^{(5)}]\!] = (o_{vrw} \leq 0.1, (o_{vrw} \geq -0.4) \wedge \Phi_1 \wedge (o_{vrw} \leq 0.1)).$$

Diese beiden Modifikationen sind voneinander abhängig, da gilt:

$$([\![F_S^{(4)}]\!] \Rightarrow [\![E_S^{(5)}]\!]) = false$$

und

$$([\![F_S^{(5)}]\!] \Rightarrow [\![E_S^{(4)}]\!]) = false$$

Es gibt 4 Kombinationsmöglichkeiten für diese beiden Modifikationen:

1.) Der Wertebereich des Vorderradwinkels sinkt. Die Einschränkungen von 0.3 auf 0.1 sollen überwiegen. Das Resultat der Analyse dieser Kombination ergibt, dass die gewünschte Funktionalität nur noch teilweise erhalten sein kann. Es gilt:

$$\begin{aligned}[\![\mathcal{M}_S^{(6e)}]\!]^{op} &= ([\![\mathcal{M}_S^{(4)}]\!]^{op} +_E [\![\mathcal{M}_S^{(5)}]\!]^{op}) \\ &= (-0.1 \leq o_{vrw} \leq 0.1, (-0.1 \leq o_{vrw} \leq 0.1) \wedge \Phi_1)\end{aligned}$$

2.) Der Wertebereich des Vorderradwinkels wächst. Die Erweiterungen von 0.3 auf 0.4 sollen überwiegen. Das Resultat der Analyse dieser Kombination ergibt, dass auch Werte ausgegeben werden, die außerhalb des spezifizierten Bereichs liegen. Es gilt:

$$\begin{aligned}[\![\mathcal{M}_S^{(6f)}]\!]^{op} &= ([\![\mathcal{M}_S^{(4)}]\!]^{op} +_F [\![\mathcal{M}_S^{(5)}]\!]^{op}) \\ &= (-0.1 \leq o_{vrw} \leq 0.1, (-0.4 \leq o_{vrw} \leq 0.4) \wedge \Phi_1)\end{aligned}$$

3.) Der Vollständigkeit halber werden auch die durch die Reihenfolge bestimmten Kombinationen aufgelistet, deren Aussagekraft allerdings in der Praxis schwer zu bewerten ist. Falls die Verschiebung nach oben überwiegt, gilt:

$$\begin{aligned}[\![\mathcal{M}_S^{(6b)}]\!]^{op} &= ([\![\mathcal{M}_S^{(5)}]\!]^{op} + [\![\mathcal{M}_S^{(4)}]\!]^{op}) \\ &= (-0.1 \leq o_{vrw} \leq 0.1, (-0.1 \leq o_{vrw} \leq 0.4) \wedge \Phi_1)\end{aligned}$$

4.) Die Verschiebung nach unten überwiegt. Dann gilt:

$$\begin{aligned}[\![\mathcal{M}_S^{(6a)}]\!]^{op} &= ([\![\mathcal{M}_S^{(4)}]\!]^{op} + [\![\mathcal{M}_S^{(5)}]\!]^{op}) \\ &= (-0.1 \leq o_{vrw} \leq 0.1, (-0.4 \leq o_{vrw} \leq 0.1) \wedge \Phi_1)\end{aligned}$$

⌟

Systemwechsel

Eine mit Formeln definierte Modifikation $[\![\mathcal{M}_S]\!]$ wird für die Kanalnamen eines Systems S definiert. Es gibt Eigenschaften, die für verschiedene Systeme gelten können und deren Modifikationen für die verschiedenen Systeme gleich aussehen. So ist zum Beispiel die Verschiebung der Trimmung für verschiedene Varianten von Lenksystemen eines Fahrzeugs anwendbar. Eine Modifikation $[\![\mathcal{M}_S]\!]$ kann für ein System T angepasst werden, indem die freien Variablen der Formeln durch die den neuen Kanälen zugehörigen Formeln ersetzt werden. Hierbei müssen weiterhin die Datentypen der Variablen übereinstimmen.

Definition 4.2.5 (Schnittstellenanpassung einer Modifikation)
$[\updownarrow \ldots] : PRED^{Stream}_{MOD_S} \times (I_S \cup O_S) \times (I_T \cup O_T) \rightarrow PRED^{Stream}_{MOD_T}$

$$[\![\mathcal{M}_S]\!][\updownarrow \pi_{(S,T)}] \stackrel{def}{=} \begin{array}{l}([\![E_S]\!][i_T^{(1)}/i_S^{(1)}][i_T^{(2)}/i_S^{(2)}]\ldots[o_T^{(1)}/o_S^{(1)}][o_T^{(2)}/o_S^{(2)}]\ldots \\ ,[\![F_S]\!][i_T^{(1)}/i_S^{(1)}][i_T^{(2)}/i_S^{(2)}]\ldots[o_T^{(1)}/o_S^{(1)}][o_T^{(2)}/o_S^{(2)}]\ldots)\end{array}$$

Mit $\pi_{(S,T)} = \{(i_S^{(1)}, i_T^{(1)}), (i_S^{(2)}, i_T^{(2)}), \ldots, (o_S^{(1)}, o_T^{(1)}), (o_S^{(2)}, o_T^{(2)}), \ldots\}$

Gleiches gilt für operationelle Prädikate $[\![\ldots]\!]^{op}$. ⌟

Beispiel 4.2.5 (Schnittstellenanpassung einer Modifikation)
Gegeben sei die Modifikation $[\![\mathcal{M}_S^{(5)}]\!]$ aus Beispiel 4.2.4, ein weiteres Lenksystem T mit der Schnittstelle $(\{i_{lrw\_t}\}, \{o_{vrw\_t}\})$ und eine Relation $\pi_{(S,T)} = \{(i_{lrw}, i_{lrw\_t}), (o_{vrw}, o_{vrw\_t})\}$.

Die auf das System T angepasste Modifikation $[\![\mathcal{M}_S^{(5)}]\!][\updownarrow \pi_{S,T}]$ ist:

$$[\![\mathcal{M}_S^{(5)}]\!][\updownarrow \pi_{S,T}] = \begin{array}{l}(o_{vrw\_t} \leq 0.1, \\ (o_{vrw\_t} \geq -0.4) \wedge (o_{vrw\_t} \leq 0.1) \wedge \Phi_1[i_{lrw}/i_{lrw\_t}][o_{vrw}/o_{vrw\_t}])\end{array}$$
⌟

Fehlerermittlung

Systemmodifikationen werden aus der Implementierungsebene motiviert. Um ein potentielles Fehlverhalten zu ermitteln, wird betrachtet, auf welchen Elementen der Implementierungsebene das Systemverhalten abgelegt ist, dann zu den Elementen die potentiellen Fehler gesucht und die damit verbundenen Modifikationen auf das Systemverhalten angewendet. Abschnitt 4.5.2 enthält eine Sammlung potentieller Fehler. Die methodischen Schritte zur Fehlerermittlung sind folgende:

`FMS1`: Ermittlung potentieller Fehler der zugehörigen Implementierungselemente und Sammlung der dazu gehörenden Modifikationen

`FMS2`: Anpassung der Modifikationen an die Schnittstelle des Systems

`FMS3`: Im Falle von Mehrfachfehlern Ermittlung der Abhängigkeit und im Konfliktfall Auswahl der Kombination (koordinierte Modifikation)

`FMS4`: Anwendung der resultierenden Modifikationen auf das Prädikat des Systems

Anforderungsmodifikation

Die bisher definierten Anwendungen von Modifikationen beziehen sich immer auf das gesamte System. Prädikate in konjunkter Form $[\![S]\!] \in PRED(\bigwedge)^{Stream}_{REL_S}$ bieten eine weitere Möglichkeit zur Anwendung von Modifikationen. In einer FMEA oder FTA (siehe Abschnitt 2.3) entsprechen die Verhaltensbedingungen und Ausgaben den Teilprädikaten. Hat ein Prädikat $[\![S]\!]$ diese konjunktive Form, so ist es aus pragmatischer Sicht einfacher, Modifikationen auf die einzelnen Teilprädikate Φ_i zu beziehen, ohne alle Teilprädikate berücksichtigen zu müssen, also die Modifikation auf das Prädikat $[\![S]\!]$ anzuwenden. Bei Anwendung dieser Anforderungsmodifikationen ist eine Anforderung bzw. Anforderungsmenge nicht erfüllt, aber die restlichen Anforderungen erfüllt. Im Fehlerfall werden eines oder mehrere der Teilprädikate $\{\Phi_x | \Phi_x \in X_\Phi\}$ mit $X_\Phi \subset \{\Phi_1, ..., \Phi_n\}$ jeweils zu Teilprädikaten der Form $\Phi_x \Delta \mathcal{M}_x$.

Dies führt dazu, dass alle Tupel, die zuvor ein Teilprädikat Φ_x erfüllten, nun alle ein Prädikat $\Phi_x \Delta \mathcal{M}_x$ erfüllen, das der Negation, oder zumindest einem Verhalten außerhalb des in der Anforderung definierten Verhaltens entspricht. Es gilt also für eine Menge $X_\Phi = \{\Phi_{x_1}, ..., \Phi_{x_m}\}$ eine Modifikation $\mathcal{M}$ mit einer Formel Φ_F der Form $[\![S]\!][\Phi_{x_1} \Delta \mathcal{M}_{x_1} / \Phi_{x_1}][\Phi_{x_2} \Delta \mathcal{M}_{x_2} / \Phi_{x_2}]$.... Da $[\![F]\!]$ das komplette neue Verhalten des Systems beinhaltet, kann $[\![E]\!] = false$ sein. Das Verhalten des modifizierten Systems ergibt sich so zu :

Definition 4.2.6 (konjunkt eingeschränkter Modifikationsoperator)
Eine Modifikation $[\![\mathcal{M}_S]\!]$ modifiziert ein System $[\![S]\!] \in PRED(\bigwedge)^{Stream}_{REL_S}$ mit dem *konjunkt eingeschränkten Modifikationsoperator* Δ^{Φ_j} mit $j \in [1..n]$ so, dass gilt:

$$\Delta^{\Phi_j} : PRED(\bigwedge)^{Stream}_{REL_S} \times PRED^{Stream}_{REL_S} \times PRED^{Stream}_{MOD_S} \rightarrow PRED(\bigwedge)^{Stream}_{REL_S}$$
$$[\![S]\!] \Delta^{\Phi_j} [\![\mathcal{M}_S]\!] \stackrel{def}{=} [\![S]\!][\Phi_j \Delta [\![\mathcal{M}_S]\!] / \Phi_j]$$

Die sequentielle Anwendung des konjunkt eingeschränkten Modifikationsoperators wird notiert als:

$$\Delta^J : PRED(\bigwedge)^{Stream}_{REL_S} \times (PRED^{Stream}_{REL_S})^* \times (PRED^{Stream}_{MOD_S})^* \rightarrow PRED(\bigwedge)^{Stream}_{REL_S}$$
$$[\![S]\!] \Delta^J M = ([\![S]\!][\Phi_a \Delta [\![\mathcal{M}_S^{(1)}]\!] / \Phi_a])[\Phi_b \Delta [\![\mathcal{M}_S^{(2)}]\!] / \Phi_b]...$$

mit $M = \langle [\![\mathcal{M}_S^{(1)}]\!], \ldots, [\![\mathcal{M}_S^{(m)}]\!] \rangle$

und $J = \langle \Phi_a, \Phi_b, ... \rangle$ und $\#J = \#M$

Gleiches gilt für operationelle Prädikate $[\![...]\!]^{op}$. ⌟

Beispiel 4.2.6 (Konjunktiv eingeschränkte Modifikation)
Die Modifikation $[\![\mathcal{M}_S^{(5)}]\!]$ aus Beispiel 4.2.4 enthält in dem Prädikat $[\![F_S^{(5)}]\!]$ zusätzlich die Bedingung Φ_1, welche ein Stellvertreter für die weiterhin gültigen Anforderungen ist.[2] Diese Modifikation ist allgemeiner formuliert, wenn die Bedingung Φ_1 nicht in dem Prädikat $[\![F_S^{(5)}]\!]$ aufgeführt ist. Dann gilt:

$$[\![\mathcal{M}_S^{(5')}]\!] = (o_{vrw} \leq 0.1, (o_{vrw} \geq -0.4) \wedge (o_{vrw} \leq 0.1)).$$

Angewandt auf das System S mit dem konjunkt eingeschränkten Modifikationsoperator ergibt sich:

$$\begin{aligned} [\![S]\!] \Delta^{\Phi_2} [\![\mathcal{M}_S^{(5')}]\!] &= [\![S]\!][\Phi_2 \Delta [\![\mathcal{M}_S]\!] / \Phi_2] \\ &= \Phi_1 \wedge \Phi_3 \wedge (-0.4 \leq o_{vrw} \leq 0.1) \\ &= \Phi_1 \wedge (-0.4 \leq o_{vrw} \leq 0.1) \end{aligned}$$

Die Bedingung Φ_1 bleibt erhalten, und muss nicht in die Modifikation mit aufgenommen werden. ⌟

[2] In zwei beobachteten Projekten enthielt die Menge der Anforderungen an eine Komponente durchschnittlich 20 Anforderungen.

Diese Definition steht nicht im Widerspruch zur Definition 4.2.1, ist jedoch bezüglich der Definition eines Systems als eine Konjunktion von Teilprädikaten einfacher zu handhaben, da die Beschreibung kürzer und lesbarer ist. Für die jeweils nicht zerlegbaren Teilprädikate gilt die obige Definition für eine Änderung, bei der das jeweilige Teilprädikat aufgeweicht bzw. verhärtet werden kann. Mit der freien Modifikation Δ kann die eingeschränkte Modifikation nur dann verbunden werden, wenn die eingeschränkten Modifikationen vor den freien Modifikationen angewendet werden. Grund hierfür ist, dass nach einer freien Modifikation ein Prädikat $[\![S]\!]$ nicht mehr in der konjunktiven Form ist. Die Kombination der Modifikationen in Bezug auf den konjunktiv eingeschränkten Modifikationsoperator ergibt sich aus der sequentiellen Anwendung der einzelnen Modifikationen. Ein Verhalten kann dann entweder dem hinzugefügten Fehlverhalten beider Modifikationen entsprechen, dem eingeschränkten Verhalten beider Modifikationen oder jeweils einer Kombination von eingeschränktem und hinzugefügtem Fehlverhalten der Modifikationen.

frei vs. gebunden

Theorem 4.2.2 (konjunkt eingeschränkte Modifikationskombination)
Gegeben sei ein System S beschrieben mit $[\![S]\!] = \Phi_1 \wedge \Phi_2 \wedge \Phi_3$ und $[\![S]\!] \in PRED(\bigwedge)^{Stream}_{REL_S}$. Zwei Modifikationen $[\![\mathcal{M}^{(1)}_S]\!]$ und $[\![\mathcal{M}^{(2)}_S]\!]$ werden mit dem konjunkt eingeschränkten Modifikationskombinationsoperator entsprechend der sequentiellen eingeschränkten Modifikation eines Systems S wie folgend kombiniert:

$+^{\Phi_a,\Phi_b} : PRED^{Stream}_{REL_S} \times PRED^{Stream}_{REL_S} \times PRED^{Stream}_{MOD_S} \times PRED^{Stream}_{MOD_S} \rightarrow PRED^{Stream}_{MOD_S}$

$[\![\mathcal{M}^{(1)}_S]\!] +^{\Phi_a,\Phi_b} [\![\mathcal{M}^{(2)}_S]\!] \stackrel{def}{=}$ — siehe Anhang D.6

$([\![\overline{E}^{(1)}_S]\!] \wedge [\![\overline{E}^{(2)}_S]\!], (([\![F^{(1)}_S]\!] \wedge [\![F^{(2)}_S]\!]) \vee (\Phi_a \wedge [\![\overline{E}^{(1)}_S]\!] \wedge [\![F^{(2)}_S]\!]) \vee (\Phi_b \wedge [\![\overline{E}^{(2)}_S]\!] \wedge [\![F^{(1)}_S]\!])))$

Werden mehrere Modifikationen kombiniert, so wird dies notiert als:

$\sum^J : PRED^{Stream}_{REL_S} * \times PRED^{Stream}_{MOD_S} * \rightarrow PRED^{Stream}_{MOD_S}$

$$\sum^J M \stackrel{def}{=} (([\![\mathcal{M}^{(1)}_S]\!] +^{\Phi_a,\Phi_b} [\![\mathcal{M}^{(2)}_S]\!]) +^{\Phi_a \wedge \Phi_b,\Phi_c} [\![\mathcal{M}^{(3)}_S]\!]) \ldots$$

mit $M = \langle [\![\mathcal{M}^{(1)}_S]\!], \ldots, [\![\mathcal{M}^{(m)}_S]\!] \rangle$

und $J = \langle \Phi_a, \Phi_b, ... \rangle$ und $\#J = \#M$

Gleiches gilt für operationelle Prädikate $[\![\ldots]\!]^{op}$. ⌟

Wie bei der allgemeinen Modifikation muss auch bei der Modifikation mit Prädikaten ein System nach der Modifikation für alle Eingaben kausal monoton und linkstotal sein (siehe Abschnitt 3.3.1). Das so modifizierte System zeigt für jedes Verhalten der Umgebung eine Reaktion und kann so beliebige Systeme mit gleicher Schnittstelle ersetzen. Auf diese Eigenschaft ist besonders bei der konjunktiv eingeschränkten Modifikation zu achten, da die Eigenschaften dieser Systeme in der Spezifikation häufig redundant vorhanden sind. Zum Beispiel ist eine Eigenschaft, welche als Verhaltensbedingung spezifiziert ist, auch in den Prädikaten enthalten, die die Ausgaben berechnen (also die Verfeinerungen der Verhaltensbedingungen sind). Ändert man ein Teilprädikat, welches eine redundante Eigenschaft enthält, so ergibt sich schnell ein überspezifiziertes System, welches nicht mehr realisierbar ist.

Konsistenz

Fehlerermittlung

Die Ermittlung von Fehlern zu einzelnen Verhaltensbedingungen findet meist bei einer anforderungsbasierten Sicherheitsanalyse statt. Diese wird unter Anderem bei komplexen Systemen wie Software angewendet, bei denen geprüft wird, welche Folgen eine inkorrekte Implementierung der Anforderung hat. Ein Verfahren, dieses kreative Suchen zu unterstützen, ist die Verwendung von Leitworten, die zum Hinterfragen von Anforderungen hinsichtlich potentieller Fehler verwendet werden können. Ein konkreter Ansatz hierzu ist 'HazOp' ([MK05]), welcher folgende Leitworte vorsieht:
- NEIN bzw. KEIN,
- MEHR, WENIGER,
- SOWOHL ... ALS AUCH, TEILWEISE,
- ANDERS ALS, *UMKEHRUNG*

Automatisiert ist bei vorliegen formaler Anforderungen die Negation der Verhaltensbedingung bzw. das Auslassen der Verhaltensbedingung durchführbar, da hier kein Wissen über den Inhalt der Anforderung notwendig ist. Je nach Bedeutung der Verhaltensbedingung können auch individuell Fehler definiert werden. Die methodischen Schritte sind hier:

`FMA1` Ermittlung der Verhaltensbedingungen und der zugehörigen Modifikationen

`FMA2` Im Falle von Mehrfachfehlern Prüfung der Abhängigkeiten und Auswahl der Kombination

`FMA3` Anwendung der resultierenden Modifikation auf das Prädikat

`FMA4` Prüfen der Konsistenz des resultierenden Fehlverhaltens

Zusammenfassung

Mit den in diesem Abschnitt genannten Definitionen ist eine Grundlage geschaffen, um die Fehler, die in einer FMEA oder FTA vorkommen, formal zu modellieren. Nach Definition 4.2.1 können die Fehler als Modifikationen mit Formeln für stromgebundene und operative Prädikate modelliert werden und auf Systeme angewendet werden. Diese Fehler können, wenn sie unabhängig sind (Definition 4.2.3), mit der einfachen Modifikationskombination (Definition 4.2.2) kombiniert werden. Hierbei ist bei operativen Prädikaten darauf zu achten, wie die Fehlerkombinationen interpretiert werden (Theorem 4.2.1), um ein einheitliches Verständnis des Umfangs der zu beschreibenden Fehler zu haben. Sind die Modifikationen nicht unabhängig, so kann die tatsächliche Wirkung abgeschätzt werden, in dem entweder die anhängigen Tupel der E-Mengen, oder die der F-Mengen überwiegen (Definition 4.2.4). Repräsentieren die Systeme das Verhalten physikalischer Komponenten, so sind die Modifikationen, welche sich aus Fehlern der Physik ergeben auf die gesamten Komponenten anzuwenden. Bei mechatronischen Systemen wird eine FMEA oder FTA häufig anforderungsbasiert durchgeführt (siehe Abschnitt 2.3). Um als Bezugspunkt für eine Modifikation eine Anforderung verwenden zu können, wurde der in [Bre01] definierte Modifikationsoperator um die Einschränkung des Bezugs auf Anforderungen erweitert (Definition 4.2.6). Auf Basis dieser Grundlage können nun Modifikationstechniken für spezifischere Modellierungstechniken (wie aufgelöste Gleichungen oder Automaten) definiert werden.

4.2.2 Modifikation von aufgelösten totalen Gleichungen

Fokus auf Gleichheitsoperator

Eine Sonderform der Blackbox-Spezifikationen sind Prädikate, welche für die Eingaben total definiert sind und von der Form her zu einem Ausgabekanal hin aufgelöst sind (siehe Abschnitt 3.3.2). Auf diese Prädikate können, genau wie auf die Blackbox-Spezifikationen, beliebige Modifikationen angewendet werden. Anhand der Form der Teilprädikate bietet sich eine Menge von speziellen Modifikationen an, die sich auf den Gleichheitsoperator zwischen der Ausgabe und dem Term beziehen. Die Relation zwischen dem Term und der Ausgabe wird entsprechend modifiziert und kann so als Abweichung vom Sollwert modelliert werden (siehe auch [Str04]). Diese Modifikationen werden Ausgabemodifikationen genannt und sind wie folgend definiert:

Definition 4.2.7 (Ausgabemodifikation)
Eine *Ausgabemodifikation* ist eine Modifikation $[\![\mathcal{M}_{o_j}]\!]^{op} \stackrel{def}{=} ([\![\overline{E}_{o_j}]\!]^{op}, [\![F_{o_j}]\!]^{op})$ mit $o_j \subseteq O_S$ für ein Teilpädikat $\Phi_{o_j} = (o_j = \tau_{o_j})$ bzw. $\Phi_{o_j} = (o_j \in \tau_{o_j}^{\{\}})$. Es gilt:

$$[\![\overline{E}_{o_j}]\!]^{op} \in \{(true), (\Phi_{condition\_o_j} \Rightarrow \neg\Phi_{o_j})\}$$

$$[\![F_{o_j}]\!]^{op} \stackrel{def}{=} (\Phi_{condition\_o_j} \Rightarrow \Phi_{fail\_o_j}) \wedge (\neg\Phi_{condition\_o_j} \Rightarrow \Phi_{o_j})$$

mit $free(\Phi_{condition\_o_j}), free(\Phi_{fail\_o_j}) \subseteq \{\tau_{o_j}, o_j\}$ und $\Phi_{fail\_o_j} \wedge \Phi_{o_j} = false$.[3]
Die Menge aller Ausgabemodifikationen zu einer Ausgabe o_j ist: $PRED^{op}_{MOD\_o_j}$. ⌟

Parameter

Das Fehlverhalten wird mit der Formel $\Phi_{fail\_o_j}$ ausgedrückt. Hier können o_j und τ_{o_j} beliebig in Relation gesetzt werden, aber auch o_j auf vom Sollwert unabhängige Werte gesetzt werden. Die herkömmlichen Beschreibungen von Abweichungen sind in Definition 4.2.7 um eine Bedingung $\Phi_{condition\_o_j}$ erweitert worden, mit der bestimmt werden kann, wann die Abweichungen auftreten. Diese Bedingung richtet sich nach Eigenschaften, die für die Ausgabe erfüllt sein müssen. Hierbei kann auch eine direkte Referenzierung der Zeit nach dem gleichen Schema wie bei den Systembeschreibungen in Abschnitt 3.3.1 erfolgen.

Auftreten der Fehler

Das Auftreten der Fehler kann entweder so angegeben werden, dass sie sicher auftreten, oder so, dass sie sporadisch auftreten. Dies wird über die Formel $[\![\overline{E}_{o_j}]\!]^{op}$ bestimmt. Soll ein Fehler immer dann, wenn es möglich ist auftreten, so wird die Formel $[\![\overline{E}_{o_j}]\!]^{op} = (\Phi_{condition\_o_j} \Rightarrow \neg\Phi_{o_j})$ verwendet. Das definierte Fehlverhalten $[\![F_{o_j}]\!]^{op}$ enthält nicht das Sollverhalten Φ_{o_j}, wenn $\Phi_{condition\_o_j}$ gilt, da $\Phi_{fail\_o_j} \wedge \Phi_{o_j} = false$ gilt. Ist die Bedingung für den Fehler erfüllt, bleibt somit nicht mehr die Option der fehlerfreien Belegung. Möchte man den Fehler sporadisch auftreten lassen, so wird die Formel $[\![\overline{E}_{o_j}]\!]^{op} = (true)$ verwendet, und somit steht frei, ob gerade die Formel für den Fehlerfall oder die als normal spezifizierte Formel gilt.

[3] Damit gilt: $\Phi_{fail\_o_j} \Rightarrow \neg\Phi_{o_j}$ und $\Phi_{o_j} \Rightarrow \neg\Phi_{fail\_o_j}$

Beispiel 4.2.7 (Modifikation einer Ausgabe)
Gegeben sei eine geschwindigkeitsabhängige Lenkübersetzung S (siehe Abschnitt 3.3.3), welche definiert ist als:

$$[\![S]\!]^{op} = \Phi_{o_{vrw}} \text{ mit } \tau_{o_{vrw}} = (i_{lrw}/(16 \cdot \mathsf{max}(1, (|i_v|/100km/h))).$$

Eine einfache Modifikation $[\![\mathcal{M}_{o_{vrw}}^{BetragZuKlein}]\!]^{op}$ dieses Systems ist, dass das System immer zu kleine Beträge liefert. Die Bedingung, die das Auftreten eines Fehlers bestimmt, ist, dass ein Betrag, der größer als Null ist, anliegt. Es gilt also:

$$\Phi_{condition\_o_{vrw}}^{BetragZuKlein} = (|\tau_{o_{vrw}}| > 0)$$

Das Fehlerbild ist die zu kleine Ausgabe, wobei das Vorzeichen der Ausgabe erhalten bleibt:

$$\Phi_{fail\_o_{vrw}}^{BetragZuKlein} = ((|\tau_{o_{vrw}}| > o_{vrw}) \wedge (\mathsf{sign}(\tau_{o_{vrw}}) = \mathsf{sign}(o_{vrw})))$$

Nach Definition 4.2.7 lassen sich nun die beiden Formeln
$[\![F_{o_{vrw}}^{BetragZuKlein}]\!]^{op}$ und $[\![E_{o_{vrw}}^{BetragZuKlein}]\!]^{op}$ ableiten.
Zur Beschreibung der Modifikation $[\![\mathcal{M}_{o_{vrw}}^{BetragZuKlein}]\!]^{op}$ genügt also die Angabe der Bedingung, der Abweichung und der Entscheidung, ob $[\![E_{o_{vrw}}^{BetragZuKlein}]\!]^{op}$ gleich *true* ist. ⌟

Fehlerermittlung

Die potentiellen Fehler, die für Ausgaben auftreten können werden auf verschiedene Arten ermittelt. Entscheidend ist hier der Zweck, den das Fehlerbild erfüllen soll. Soll die Ausgabemodifikation ein Fehlverhalten der Implementierung widerspiegeln, so wird wie bei den Systemmodifikationen in den Modellen der Implementierungsebene nach potentiellen Fehlern mit dazugehörenden Modifikationen gesucht (siehe Abschnitt 4.5.2). Sollen die Ausgabemodifikationen die kritischen Fehlverhalten an der Nutzungsschnittstelle widerspiegeln, so werden die Modifikationen anhand der Nutzungsfälle abgeleitet. In der Fallstudie ergaben sich z.B. aus Tests mit dem Fahrwerkregelsystem die Fehlerbilder der Abweichung vom Sollwert mit Wertsprung und die Abweichung ohne Wertsprung. Übrig bleibt die Ermittlung der potentiellen Fehler auf den Zwischenebenen eines Systems, also unterhalb der Nutzungsschnittstelle und oberhalb der Systeme mit zugewiesenen Implementierungselementen. Hier müssen die Modifikationen möglichst so strukturiert sein, dass sie den Übergang von der Implementierung hin zur Schnittstelle ermöglichen. Da hier nicht generell vorgegeben werden kann, wie die Fehler an den Ausgaben aussehen, wird hier eine Methode vorgeschlagen, in der die Fehler wie in einem Baukasten ermittelt werden. Dazu wird im Folgenden die Kombinierbarkeit von Ausgabemodifikationen untersucht und ein Baukasten zur Ermittlung der passenden Fehlerbeschreibungen vorgestellt.

Minimalität

Eine für die Kombination notwendige Eigenschaft der Ausgabemodifikationen ist die Minimalität. Ist $[\![\overline{E}_{o_j}]\!]^{op} = (true)$, so ist die Minimalität trivialerweise gegeben, da das Verhalten nicht eingeschränkt wird. Gilt die Formel $[\![\overline{E}_{o_j}]\!]^{op} = (\Phi_{condition\_o_j} \Rightarrow \neg\Phi_{o_j})$, so ist unter der Bedingung $\Phi_{condition\_o_j}$ das Sollverhalten ausgeschlossen. Dieses wird in $[\![F_{o_j}]\!]^{op}$ allerdings auch nicht hinzugefügt, da $\Phi_{fail\_o_j} \wedge \Phi_{o_j} = false$ gilt. Somit sind die Modifikationen minimal. Diese Eigenschaft ist eine Grundlage für die im folgenden Absatz beschriebene Kombination von Modifikationen.

Die Kombination der Ausgabemodifikationen betrachtet zum Einen den Fall, dass mehrere Fehler auftreten können, und zum Anderen den Fall, dass ein komplexerer Fehler aus einfacheren Fehlern zusammengesetzt wird. Um mit der Kombination effizient umgehen zu können, ist hier wichtig, dass sie einfach überprüft und angewendet werden kann. Die Kombination der Ausgabemodifikationen wird in zwei Fälle unterteilt: *Kombination*

- Haben zwei Modifikationen $\mathcal{M}_{o_j}^{(1)}$ und $\mathcal{M}_{o_j}^{(2)}$ die gleichen Bedingungen unter denen sie auftreten, so sind sie voneinander unabhängig, da im Fall einer Formel $[\![\overline{E}_{o_j}^{(1)}]\!]^{op} = \mathit{true}$ die Unabhängigkeit trivialerweise gilt und im Fall $[\![\overline{\overline{E}}_{o_j}^{(1)}]\!]^{op} = (\Phi_{condition\_o_j}^{(1)} \Rightarrow \neg\Phi_{o_j})$ aufgrund dessen, dass Ausgabemodifikationen minimal sind, folgendes gilt:

$$
\begin{aligned}
[\![F_{o_j}^{(2)}]\!]^{op} &= (\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{fail\_o_j}^{(2)}) \wedge (\neg\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{o_j}) \\
&\Rightarrow \mathit{true} \wedge (\neg\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{o_j}) \\
&\Rightarrow (\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{fail\_o_j}^{(2)}) \\
&\Rightarrow (\Phi_{condition\_o_j}^{(2)} \Rightarrow \neg\Phi_{o_j}) \\
&\Rightarrow (\Phi_{condition\_o_j}^{(1)} \Rightarrow \neg\Phi_{o_j}) \\
&\Rightarrow [\![\overline{E}_{o_j}^{(1)}]\!]^{op}
\end{aligned}
\quad \Big| \quad
\begin{aligned}
&\Phi \Rightarrow \mathit{true} \\
& \\
&\Phi_{fail\_o_j}^{(2)} \Rightarrow \neg\Phi_{o_j} \\
&\Phi_{cond\_o_j}^{(1)} = \Phi_{cond\_o_j}^{(2)}
\end{aligned}
$$

  Analog gilt: $[\![F_{o_j}^{(1)}]\!]^{op} \Rightarrow [\![\overline{E}_{o_j}^{(2)}]\!]^{op}$.

  Durch die Unabhängigkeit der Fehler kann eine einfache Modifikationskombination angewendet werden. Bei den Prädikaten $[\![\overline{E}_{o_j}^{(1)}]\!]^{op}$ und $[\![\overline{E}_{o_j}^{(2)}]\!]^{op}$ überwiegt jeweils das stärkere Prädikat. Die Kombination der Modifikationen ergibt wieder eine Ausgabemodifikation. Es gilt:

$$
\begin{aligned}
[\![\mathcal{M}_{o_j}^{(1)}]\!]^{op} + [\![\mathcal{M}_{o_j}^{(2)}]\!]^{op} &= ([\![\overline{E}_{o_j}^{(1)}]\!]^{op} \wedge [\![\overline{E}_{o_j}^{(2)}]\!]^{op}, [\![F_{o_j}^{(1)}]\!]^{op} \vee [\![F_{o_j}^{(2)}]\!]^{op}) \\
&= ([\![\overline{E}_{o_j}^{(1,2)}]\!]^{op}, [\![F_{o_j}^{(1,2)}]\!]^{op})
\end{aligned}
$$

$$
\begin{aligned}
[\![F_{o_j}^{(1,2)}]\!]^{op} &= (\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{fail\_o_j}^{(1)}) \wedge (\neg\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{o_j}) \\
&\quad \vee (\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{fail\_o_j}^{(2)}) \wedge (\neg\Phi_{condition\_o_j}^{(2)} \Rightarrow \Phi_{o_j}) \\
&= ((\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{fail\_o_j}^{(1)}) \vee (\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{fail\_o_j}^{(2)})) \\
&\quad \wedge (\neg\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{o_j}) \\
&= (\Phi_{condition\_o_j}^{(1)} \Rightarrow (\Phi_{fail\_o_j}^{(1)} \vee \Phi_{fail\_o_j}^{(2)})) \\
&\quad \wedge (\neg\Phi_{condition\_o_j}^{(1)} \Rightarrow \Phi_{o_j})
\end{aligned}
\quad \Big| \quad \Phi_{cond\_o_j}^{(1)} = \Phi_{cond\_o_j}^{(2)}
$$

  Um zwei Modifikationen mit gleicher Bedingung zu kombinieren, genügt es also, die beiden $\Phi_{fail\_o_j}^{(x)}$ zu einem neuen $\Phi_{fail\_o_j}^{(xy)}$ zu verbinden.

- Haben zwei Modifikationen $\mathcal{M}_{o_j}^{(1)}$ und $\mathcal{M}_{o_j}^{(2)}$ verschiedene Bedingungen, so kann eine Abhängigkeit zwischen ihnen bestehen. Eine Formel $[\![F_{o_j}^{(1)}]\!]^{op}$ kann ein korrektes Verhalten des Systems unter der Vorbedingung $\Phi_{condition\_o_j}^{(2)}$ beinhalten, dieses aber in $[\![\overline{E}_{o_j}^{(2)}]\!]^{op}$ ausgeschlossen sein. Entsprechend kann hier entschieden werden, wie die Modifikationen kombiniert werden. Im Folgenden wird die schwierigste Variante der koordinierten Kombination für den Fall gezeigt, bei dem die $[\![E]\!]$ nicht *true* sind und überwiegen:

$$\begin{aligned}
&[\![\mathcal{M}_{o_j}^{(1)}]\!]^{op} +_E [\![\mathcal{M}_{o_j}^{(2)}]\!]^{op} \\
&\quad = ([\![\overline{E}_{o_j}^{(1)}]\!]^{op} \wedge [\![\overline{E}_{o_j}^{(2)}]\!]^{op}, ([\![F_{o_j}^{(1)}]\!]^{op} \wedge [\![\overline{E}_{o_j}^{(2)}]\!]^{op}) \vee ([\![F_{o_j}^{(2)}]\!]^{op} \wedge [\![\overline{E}_{o_j}^{(1)}]\!]^{op})) \\
&\quad = ([\![\overline{E}_{o_j}^{(1,2)}]\!]^{op}, [\![F_{o_j}^{(1,2)}]\!]^{op})
\end{aligned}$$

Gemäß Anhang D, Gleichung D.3 und Gleichung D.4 gilt:

$$\begin{aligned}
[\![E_{o_j}^{(1,2)}]\!]^{op} &= (\Phi_{condition_o_j}^{(1)} \Rightarrow \neg\Phi_{o_j}) \wedge (\Phi_{condition_o_j}^{(2)} \Rightarrow \neg\Phi_{o_j}) \\
&= (\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow \neg\Phi_{o_j}
\end{aligned}$$

$$\begin{aligned}
&[\![F_{o_j}^{(1,2)}]\!]^{op} \\
&= (\Phi_{condition_o_j}^{(1)} \Rightarrow \Phi_{fail_o_j}^{(1)}) \wedge (\neg\Phi_{condition_o_j}^{(1)} \Rightarrow \Phi_{o_j}) \wedge (\Phi_{condition_o_j}^{(2)} \Rightarrow \neg\Phi_{o_j}) \\
&\quad \vee (\Phi_{condition_o_j}^{(2)} \Rightarrow \Phi_{fail_o_j}^{(2)}) \wedge (\neg\Phi_{condition_o_j}^{(2)} \Rightarrow \Phi_{o_j}) \wedge (\Phi_{condition_o_j}^{(1)} \Rightarrow \neg\Phi_{o_j}) \\
&= (\neg(\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge ((\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow (\Phi_{fail_o_j}^{(1)} \wedge \Phi_{condition_o_j}^{(1)} \vee \Phi_{fail_o_j}^{(2)} \wedge \Phi_{condition_o_j}^{(2)}))
\end{aligned}$$

Sind zusätzlich die Bedingungen $\Phi_{condition\_o_j}^{(1)}$ und $\Phi_{condition\_o_j}^{(2)}$ konjunkt, so gilt gemäß Anhang D, Gleichung D.5:

$$\begin{aligned}
[\![F_{o_j}^{(1,2)}]\!]^{op} &= (\neg(\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge ((\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow \\
&\qquad (\Phi_{fail_o_j}^{(1)} \wedge \Phi_{condition_o_j}^{(1)} \vee \Phi_{fail_o_j}^{(2)} \wedge \Phi_{condition_o_j}^{(2)})) \\
&= (\neg(\Phi_{condition_o_j}^{(1)} \vee \Phi_{condition_o_j}^{(2)}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi_{condition_o_j}^{(1)} \Rightarrow \Phi_{fail_o_j}^{(1)}) \wedge (\Phi_{condition_o_j}^{(2)} \Rightarrow \Phi_{fail_o_j}^{(2)})
\end{aligned}$$

Basisklassen

Mit diesen beiden Regeln lassen sich Ausgabemodifikationen kombinieren. Die Ergebnisse sind wiederum Ausgabemodifikationen. Die Kombination der Modifikationen findet, insbesondere bei disjunkten Bedingungen, über einfache boolesche Operationen zwischen den Bedingungen und Fehlerformeln statt. Damit sind Ausgabemodifikationen geeignet, miteinander kombiniert zu werden. Die Kombination von Modifikationen ermöglicht die Kombination komplexer Fehler aus einfachen Fehlerklassen. Diese einfachen Fehlerklassen werden *Basisklassen* genannt. Fehlerbeschreibungen sollten für die Analysten und Entwickler möglichst eingängig sein. Eine Möglichkeit, eine Modifikation schnell zu verstehen, ist zu sehen, aus welchen Teilmodifikationen sie sich zusammensetzt. Die Beschreibungen der Fehler basiert so auf einer endlichen Menge einfacher Fehler, die wie in einem Baukasten zusammensetzbar sind. Im weiteren Abschnittsverlauf werden repräsentative Fehlerklassen und deren Kombinationen vorgestellt, die auch in der Fallstudie vorgefunden wurden.

zeitunabhängige Wertabweichung

Bei der Betrachtung der mechanischen Schnittstelle eines eingebetteten Systems ist die zeitunabhängige Abweichung des Ausgabewertes vom Sollwert häufig eine grobe, einfach ermittelbare obere Schranke für eine Fehlerbeschreibung.[4] So kann zum Beispiel der Ausgabewert der Überlagerungslenkung größer als der für die gegebene Situation spezifizierte Sollwinkel sein. Entsprechend würde das Fahrzeug zu stark lenken. Eine Möglichkeit, Wertabweichungen zu klassifizieren, ist in Tabelle 4.1 und Abbildung 4.5 vorgestellt. Diese Gruppierung der Fehler hat so auch in der Fallstudie stattgefunden. Allen Wertabweichungen gemeinsam ist der zeitunabhängige Bezug der Bedingungen auf Wertebereiche, in denen die Fehler gültig sind. Auch die Formeln, welche die Fehler beschreiben, haben bei dieser Art der Abweichung nur einen zeitunabhängigen Bezug zum Wertebereich.

Kürzel	Fehlerklasse für Formel $[\![F_{o_j}^{(base)}]\!]$	$\Phi^{base}_{condition\_o_j}$	$\Phi^{base}_{fail\_o_j}$
nZsN	negative Zahl statt Null	$(\tau_{o_j} = 0)$	$(o_j < \tau_{o_j})$
pZsN	positive Zahl statt Null	$(\tau_{o_j} = 0)$	$(o_j > \tau_{o_j})$
NsnZ	Null statt negativer Zahl	$(\tau_{o_j} < 0)$	$(o_j = 0)$
NspZ	Null statt positiver Zahl	$(\tau_{o_j} > 0)$	$(o_j = 0)$
BzgpZ	Betrag zu groß bei pos. Zahl	$(\tau_{o_j} > 0)$	$(o_j > \tau_{o_j})$
BzgnZ	Betrag zu groß bei neg. Zahl	$(\tau_{o_j} < 0)$	$(o_j < \tau_{o_j})$
BzkpZ	Betrag zu klein bei pos. Zahl	$(\tau_{o_j} > 0)$	$(0 < o_j < \tau_{o_j})$
BzknZ	Betrag zu klein bei neg. Zahl	$(\tau_{o_j} < 0)$	$(0 > o_j > \tau_{o_j})$
VzpZ	Vorzeichenfehler (Vzf.) bei pos. Z.	$(\tau_{o_j} > 0)$	$(o_j = -\tau_{o_j})$
VznZ	Vorzeichenfehler bei neg. Zahl	$(\tau_{o_j} < 0)$	$(o_j = -\tau_{o_j})$
BzgVzpZ	Betrag zu groß und Vzf. bei pos. Z.	$(\tau_{o_j} > 0)$	$(o_j < -\tau_{o_j})$
BzgVznZ	Betrag zu groß und Vzf. bei neg. Z.	$(\tau_{o_j} < 0)$	$(o_j > -\tau_{o_j})$
BzkVzpZ	Betrag zu klein und Vzf. bei pos. Z.	$(\tau_{o_j} > 0)$	$(0 > o_j > -\tau_{o_j})$
BzkVznZ	Betrag zu klein und Vzf. bei neg. Z.	$(\tau_{o_j} < 0)$	$(0 < o_j < -\tau_{o_j})$
TsF	True statt False	$(\tau_{o_j} = false)$	$(o_j = true)$
FsT	False statt True	$(\tau_{o_j} = true)$	$(o_j = false)$

Tabelle 4.1: Beispiel für eine Klassifizierung zeitunabhängiger Wertabweichungen

Baukasten

Der Baum in Abbildung 4.6 zeigt eine im Folgenden erklärte Möglichkeit, die in Tabelle 4.1 gegebenen Fehlerklassen in einem Baukastenprinzip zusammenzufassen. Zur Zusammenfassung kann entweder die Kombination mit gleicher Bedingung verwendet werden, oder die Kombination mit disjunkten Bedingungen $\Phi^{base}_{condition\_o_j}$. In Tabelle 4.1 werden die Bedingungen in positive Werte, negative Werte und die Null unterschieden. In manchen Fällen (z.B. Betrag zu groß) spielt es keine Rolle, ob der Sollwert positiv oder negativ ist. Es genügt eine gröbere Klasse, in der positive und negative Zahlen zusammengefasst sind. Diese Zusammenfassung der Bedingungen ist zum Beispiel in Abbildung 4.6, von der untersten Ebene zur darüber liegenden

[4] Die Betrachtung eines Fehlers als zeitunabhängige Wertabweichung wurde in den FMEA-Dokumenten der beiden Fallstudien vorgefunden.

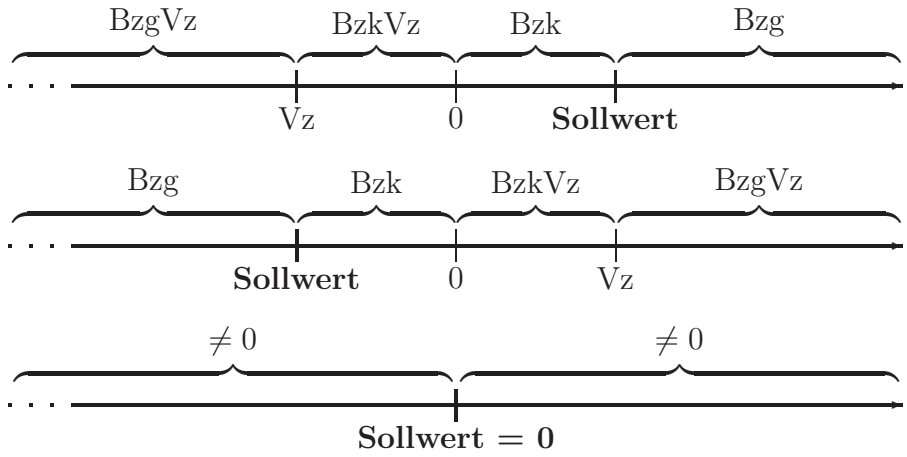


Abb. 4.5: Beispiel für eine Klassifizierung zeitunabhängiger Wertabweichungen

Ebene, zu finden. Umgekehrt werden z.B. in einer FMEA die Bedingungen hin zu den Wertebereichen der Eingabe, welche in den jeweiligen Szenarien vorkommen, angegeben (siehe [Str06]). Die Zusammenfassung bei gleicher Vorbedingung ist in der Tabelle 4.1 in den mittleren Ebenen zu finden. Es werden die Fehlerbeschreibungen in zu große, zu kleine und in Beträge gleich Null (Die Null steht hier als Vertreter für ein neutrales Element) unterschieden. Auch hier kann die Beschreibung des Fehlverhaltens gröber ausfallen, in dem z.B. mit einem Betragsfehler sowohl zu große, wie auch zu kleine Beträge zusammengefasst werden. Umgekehrt können z.B. die Abweichungen der Beträge in ihrer Stärke unterschieden werden.

relaxierte Formeln

Die Fehlerklassen in Tabelle 4.1 beziehen sich auf Spezifikationen, in denen Funktionen mathematisch eindeutige Abbildungen sind. In der Fallstudie können die Systeme bedingt durch die Implementierung und Anwendung Toleranzen zulassen. In der Sicherheitsspezifikation der Fallstudie waren zum Beispiel Fehler erst relevant, wenn sie die Toleranz von 15% bzw. 100Nm überschritten haben. Wird wie in Abschnitt 3.3.2 eine Funktion relaxiert gemäß der Form $\Phi_{o_j}^{\pm a}$ oder $\Phi_{o_j}^{\div a}$ beschrieben, hat dies auf die Ausgabefehler die Auswirkung, dass statt Φ_{o_j} die relaxierte Formel verwendet wird. Entsprechend gilt:

Definition 4.2.8 (relaxierte Ausgabemodifikation)
Eine relaxierte Ausgabemodifikation ist eine Modifikation

$$[\![\mathcal{M}_{o_j\pm}]\!]^{op} \stackrel{def}{=} ([\![\overline{E}_{o_j\pm}]\!]^{op}, [\![F_{o_j\pm}]\!]^{op}) \text{ mit } o_j \in O_S \text{ für ein } \Phi_{o_j}^{\pm a} = (o_j \stackrel{<\pm a>}{=} \tau_{o_j}).$$

Es gilt:

$$[\![\overline{E}_{o_j\pm}]\!]^{op} \in \{(true), (\Phi_{condition\_o_j} \Rightarrow \neg\Phi_{o_j}^{\pm a})\} \text{ und}$$

$$[\![F_{o_j\pm}]\!]^{op} \stackrel{def}{=} (\Phi_{condition\_o_j} \Rightarrow \Phi_{fail\_o_j}) \wedge (\neg\Phi_{condition\_o_j} \Rightarrow \Phi_{o_j}^{\pm a})$$

mit $free(\Phi_{condition\_o_j}), free(\Phi_{fail\_o_j}) \subseteq \{\tau_{o_j}, o_j, a\}$ und $\Phi_{fail\_o_j} \wedge \Phi_{o_j}^{\pm a} = false$.
Gleiches gilt für Formeln der Art $\Phi_{o_j}^{\div a}$ und $\Phi_{o_j}^{\pm a \div b}$. ⌟

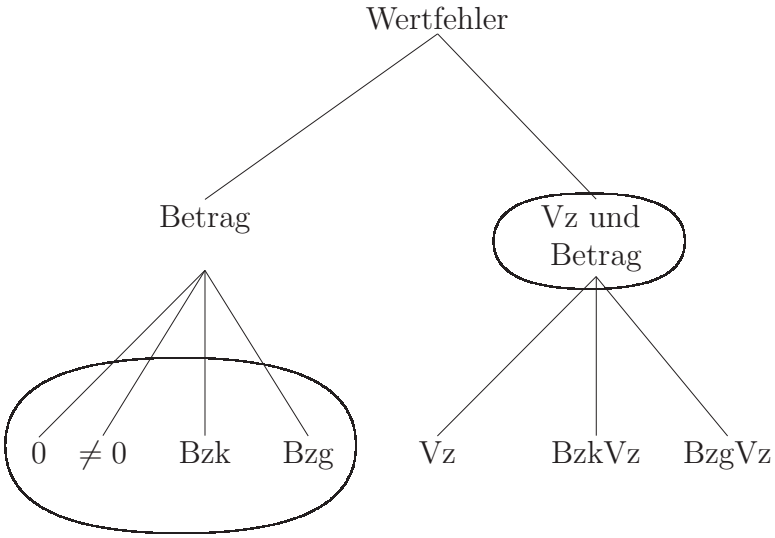


Abb. 4.6: Beispiel einer Klassenhierarchie (häufig verwendete Klassen eingekreist)

Anpassung der Bedingung und des Fehlers

Eine systematische Überführung einer Modifikation für eindeutige Formeln zu einer Modifikation für relaxierte Formeln ist nicht allgemein formulierbar. Sowohl die Bedingung $\Phi_{condition\_o_j}$, wie auch die Fehlerbeschreibung $\Phi_{fail\_o_j}$, sind nicht zwingend aufgelöste Gleichungen, sondern können beliebige Relationen sein. Entsprechend stehen hier beliebige Möglichkeiten zur Anpassung dieser Formeln zur Verfügung. Sie können sich auf die Terme $\tau_{o_j}^{upper}$ und $\tau_{o_j}^{lower}$ beziehen, oder auf $\Phi_{o_j}^{\pm a}$, $\Phi_{o_j}^{\div a}$ und $\Phi_{o_j}^{\div a \pm b}$. Wichtig ist, dass sich weiterhin die Fehlverhalten und Sollverhalten ausschließen, um die einfache Kombinierbarkeit der Basisklassen zu gewährleisten. Eine Möglichkeit, die Fehlerformeln $\Phi_{fail\_o_j}$ der Basisklassen in Tabelle 4.1 an relaxierte Formeln $\Phi_{o_j}^{\pm a}$ anzupassen, wird im Folgenden vorgestellt (siehe auch Abbildung 4.7):

- Für die Bedingungen gilt:

  $\Phi_{condition\_o_j} = (\tau_{o_j} = 0)$ wird zu $\Phi_{condition\_o_j} = (|\tau_{o_j}| \leq a)$.
  $\Phi_{condition\_o_j} = (\tau_{o_j} > 0)$ wird zu $\Phi_{condition\_o_j} = (\tau_{o_j} > a)$.
  $\Phi_{condition\_o_j} = (\tau_{o_j} < 0)$ wird zu $\Phi_{condition\_o_j} = (\tau_{o_j} < a)$.

- Für die Fehlerformeln gilt:

  $(< \tau_{o_j})$ wird zu $(< (\tau_{o_j} - a))$
  $(> \tau_{o_j})$ wird zu $(> (\tau_{o_j} + a))$
  $(o_j = -\tau_{o_j})$ wird zu $(o_j \overset{<\pm a>}{=} -\tau_{o_j})$
  $(= 0)$, (> 0) und (< 0) bleibt.

Bei dieser Interpretation der Fehlerbasisklassen für eine relaxierte Definition ist allerdings weiterhin eine scharfe Grenze gegeben, an der zwischen den Fehlern bezüglich 0 und $\neq 0$ an der Grenze a unterschieden wird. Um hier angemessene Fehler- und Fehlerauswirkungsbilder zu bekommen, ist die Grenze a so zu wählen, dass kein unerwünschtes „Flattern“ zwischen Sollverhalten und Fehlverhalten auftritt.

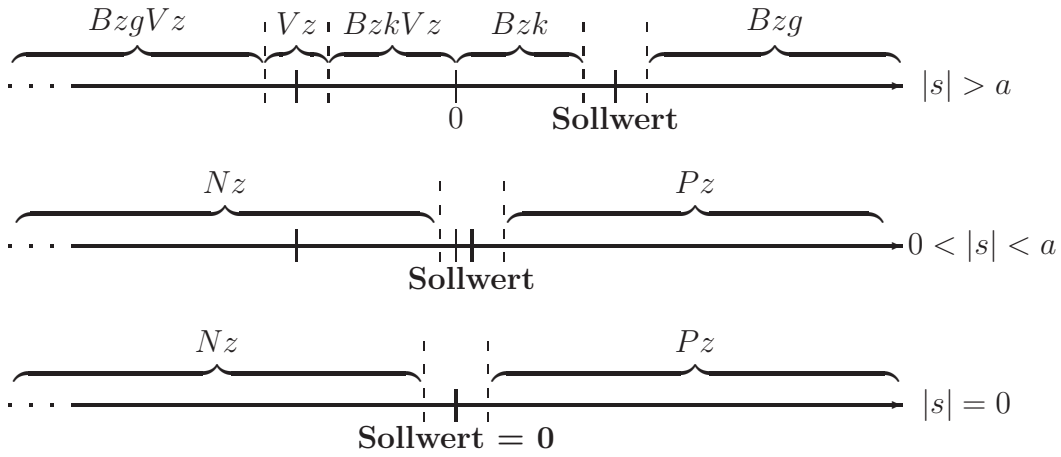


Abb. 4.7: Beispiel für eine Modifikation relaxierter Gleichungen

zeitabhängige Fehler

Neben den zeitunabhängigen Wertabweichungen gibt es weitere Fehler, die für die einzelnen Ausgaben definiert werden können und einen Bezug zum Ablauf über die Zeit haben. Im Folgenden wird hier der Sensordrift als ein Vertreter für einen komplexeren zeitabhängigen Fehler dargestellt. Weitere Fehlerbilder, die in den Fallstudien aufgefunden wurden, waren zum Beispiel Wertabweichungen mit Gradientenbegrenzung.

Beispiel 4.2.8 (Drift als Vertreter zeitabhängiger Abweichungen)
Dieses Beispiel zeigt die Beschreibung des Fehlers Sensordrift. Es wird schrittweise ein etwas komplexeres Fehlverhalten beschrieben, um einen Eindruck der Modellierung eines zeitabhängigen Fehlerbildes zu bekommen.

Bei diesem Fehler liefert der Sensor Daten mit steigender Abweichung. Das Sollverhalten kann nicht mehr auftreten, wenn der Fehler einmal aufgetreten ist. Ob der Fehler aufgetreten ist, wird in der Hilfsvariable $f^{drift} : bool$ festgehalten. Es gilt entsprechend:

$$[\![E^{drift}_{o_j}]\!] = (\Phi^{drift}_{condition\_o_j} \Rightarrow \neg\Phi_{o_j}) \text{ und } \Phi^{drift}_{condition\_o_j} = (\mathsf{delay}(f^{drift}))$$

Das Fehlerbild wird im einfachsten Fall mit 4 Formeln beschrieben. Es gilt:

$$\Phi^{drift}_{fail\_o_j} = (\Phi_a \wedge \Phi_b \wedge \Phi_c \wedge \Phi_d) \text{ mit}$$

$\Phi_a = ((o_j = \tau_{o_j} + a) \wedge f^{drift})$	Addition Abweichung
$\Phi_b = (\mathsf{delay}_{\langle 0\rangle}(o_j) = \mathsf{delay}_{\langle 0\rangle}(\tau_{o_j}) + \mathsf{delay}_{\langle 0\rangle}(a))$	Persistenz
$\Phi_c = (\lvert a\rvert \geq \lvert\mathsf{delay}_{\langle 0\rangle}(a)\rvert)$	monotone Steigung
$\Phi_d = (((\mathsf{sign}(a) = \mathsf{sign}(\mathsf{delay}_{\langle 0\rangle}(a))) \vee (\mathsf{delay}_{\langle 0\rangle}(a) = 0)) \wedge (a \neq 0))$	kein Vorzeichenwechsel

Charakteristisch für diese Art der Fehler ist, dass die Fehler persistent sind und die Abweichung nicht kleiner wird. Φ_c beschreibt den monoton wachsenden Betrag der Abweichung. Die Abweichung soll ebenso das Vorzeichen (abgesehen von

einem vorhergehenden Betrag Null) nicht wechseln. Diese wird in Φ_a auf den Sollwert aufaddiert. Um sicherzustellen, dass der Fehler persistent ist, wird das Aufaddieren zu aufeinander folgenden Taktzyklen gefordert. In diesem Falle zum vorhergehenden Takt in Φ_b.

Ein Fehlerhafter Sensor driftet üblicherweise nicht bis ins Unendliche, sondern nur bis zu einem bestimmten Grenzwert. Diese Begrenzung der Amplitude a kann durch einfaches Hinzufügen der Formel Φ_e erfolgen.

$$\Phi_{fail\_o_j}^{drift(aLimit)} = (\Phi_a \wedge \Phi_b \wedge \Phi_c \wedge \Phi_d \wedge \Phi_e)$$

$$\Phi_e = (aLimit \geq |a|) \qquad \text{| Begrenzung der Amplitude auf } aLimit$$

Eine weitere übliche Eigenschaft eines Drifts ist, das die Geschwindigkeit, mit der die Abweichung wächst, begrenzt ist. Dies lässt sich durch Hinzufügen einer Formel Φ_f ausdrücken:

$$\Phi_{fail\_o_j}^{drift(aLimit,\ dLimit)} = (\Phi_a \wedge \Phi_b \wedge \Phi_c \wedge \Phi_d \wedge \Phi_e \wedge \Phi_f)$$

$$\Phi_f = (|a| - |\mathsf{delay}_{\langle 0 \rangle}(a)| \leq dLimit) \qquad \text{| Begrenzung der Steigung auf } dLimit$$

Schließlich besteht noch die Möglichkeit eine Mindestdriftgeschwindigkeit festzulegen, und eine Minimalabweichung, die erreicht werden muss. Es ergibt sich so mit einer Formel Φ_g ein Fehlerbild, welches auch unter dem Begriff „Rampe“ bekannt ist.

$$\Phi_{fail\_o_j}^{drift(aLimit,\ dLimit,\ aMin\ ,\ dMin)} = (\Phi_a \wedge \Phi_b \wedge \Phi_c \wedge \Phi_d \wedge \Phi_e \wedge \Phi_f \wedge \Phi_g)$$

$$\Phi_g = ((0 < |a| < aMin) \Rightarrow (|a| - |\mathsf{delay}_{\langle 0 \rangle}(a)| \geq dMin)) \quad \left| \begin{array}{l} \text{Mindeststeigung bis} \\ \text{Mindestamplitude} \end{array} \right. \quad \lrcorner$$

Besondere Verwendbarkeit

Die Definition der Ausgabemodifikation in diesem Abschnitt hat Eigenschaften, die deren Verwendung im Vergleich zur allgemeinen Modifikation vereinfachen. Wesentlich ist, dass die Formeln als freie Variablen nur τ_{o_j} und o_j enthalten. Die tatsächliche Modifikation hängt also von dem Term τ_{o_j} ab. Die Beschreibung der Modifikation kann entsprechend für beliebige τ_{o_j} erfolgen, wenn sie mit dem Datentyp der Ausgabe vereinbar sind. Die Vorteile, die sich daraus ergeben sind:

(1) Die Modifikationen lassen sich unabhängig von Spezifikationen definieren. Sie können mit der syntaktischen Schnittstellenanpassung (siehe Definition 4.2.5) einfach den Systemen zugeordnet werden. Damit ist die Grundlage für allgemeine Fehlerbilder (Fehlerklassen) gelegt.

(2) Die exakte Gestaltung einer Steuerung physikalischer Größen wird häufig erst sehr spät anhand geeigneter Parameter festgelegt. Die Fehler und deren potentiellen Folgen können unter Berücksichtigung der Freiheitsgrade durch Parameter in relaxierten Formeln bereits abgeschätzt und modelliert werden, ohne die endgültige detaillierte Spezifikation zu kennen.

Fehler-ermittlung

Methodisch gibt es zwei Fälle für die Ermittlung der passenden Ausgabemodifikationen. Zum einen kann wie für die Systemmodifikationen vorgegangen werden, um Implementierungsfehler abzubilden. Zum Anderen leiten sich die passenden Ausgabemodifikationen aus der Nutzung bzw. als Folge von Fehlern in Teilkomponenten ab. Ein hierzu geeignetes Vorgehen ist:

`FMO1`: Definition eines geeigneten Baukastens zur effizienten Modellierung zusammengesetzter Fehler

`FMO2`: Ableitung der zu modellierenden Modifikation und Einordnung dieser im Baukasten (Kombinierbarkeit ist im Baukasten berücksichtigt)

`FMO3`: Anwendung der Modifikation auf die Ausgaben des Systems (Automatisiert mit einer nachgeschalteten Modifikationskomponente machbar)

Fallstudie

Mit den hier vorgestellten Modellierungstechniken haben sich die Abweichungen aus den Fallstudien modellieren lassen (siehe auch Abschnitt 4.5). Fast die Hälfte der Ausgabemodifikationen waren zeitunabhängige Wertabweichungen. Da die zeitunabhängigen Wertabweichungen Wertsprünge zulassen, was in der Fallstudie eine besondere Bedeutung an der Nutzungsschnittstelle hat, sind diese um ein Fehlerbild ergänzt worden, bei denen, so wie in Beispiel 4.2.8, der Gradient der Abweichung begrenzt ist.

Zeit-abweichung

Die bisherigen Beschreibungen haben sich nur auf die Dimension Werte bezogen. Analog kann auch eine Abweichung der Zeit modelliert werden. Die Modellierung von Abweichungen entlang der Zeit wird meistens dann verwendet, wenn die Signale Nachrichtencharakter haben, also das Anliegen eines Wertes mit einer bestimmten Nachricht verbunden ist. Um auszudrücken, dass gerade keine Nachricht anliegt, wird hier das neutrale Element $\perp$ eingeführt, welches in manchen Modellierungssprachen auch mit dem Wert 0 gleichgesetzt wird.

Definition 4.2.9 (wertneutrale Zeitabweichung)
Gegeben sei ein System S mit den Ausgabekanälen $o \subseteq O_S$.
Die Wertemenge des Kanals o sei: $W_{\mathcal{F}_o} = M \cup \perp$.

Eine wertneutrale Zeitabweichung ist definiert als:

$$[\![\mathcal{M}_o^{\text{time}}]\!] \stackrel{def}{=} ([\![\overline{E}_{o_j}^{\text{time}}]\!], [\![F_{o_j}^{\text{time}}]\!])$$

mit $[\![F_{o_j}^{\text{time}}]\!] = (M\circledS o = M\circledS \tau_o)$

Eine wertneutrale Verzögerung ist definiert als:

$$[\![F_{o_j}^{\text{delay}}]\!] = (\forall x \sqsubseteq o, y \sqsubseteq \tau_o : \#x = \#y \Rightarrow M\circledS x \sqsubseteq M\circledS y)$$ ⌟

Die oben gezeigte Definition verwendet strombasierte Formeln. Eine Zeitabweichung ist mit operationellen Modifikationen relativ schwer zu modellieren. Hier kann eine Transitionsmodifikation im nächsten Abschnitt weiterhelfen.

4.2.3 Modifikation von Zustandsautomaten

Ablaufänderung

Bei Zustandsautomaten steht nicht, wie bei den aufgelösten Gleichungen die allgemeine Berechnung eines Wertes im Vordergrund, sondern die Beschreibung von Abläufen. Wie in Definition 4.2.9 zu sehen, sind die den Ablauf betreffenden Fehler als Abweichungen vorzugsweise über stromgebundene Prädikate zu modellieren. Ziel ist es deshalb, die Wirkung von Fehlern auf einzelne Schritte des Verhaltens eines Systems auszudrücken, was dann ganze Ablaufänderungen zur Folge haben kann. Diese Beschreibung von Fehlern steht so als operative Alternative zu Blackbox-Spezifikationen mit Strömen als Variablen, in denen auch die Wirkung über die Zeit mit modelliert werden kann.

Allgemeine Modifikation

Das Verhalten von Zustandsautomaten kann durch das Hinzufügen von Transitionen und das Entfernen von Transitionen modifiziert werden. Neue Transitionen können auch auf neue Zustände verweisen. Insbesondere können sie auch auf Zustände neuer Variablen verweisen, die im Rahmen der Modifikation angelegt werden können. Eine Modifikation wird daher durch eine Änderung der Menge der Transitionen und Hinzufügen lokaler Variablen wie folgt definiert (siehe [Bre01]):

Definition 4.2.10 (Automatenmodifikation)
Sei A ein Transitionssystem, dann ist eine *Automatenmodifikation*

$$\mathcal{M}_{A(\delta)} = (\delta_{E_A}, \delta_{F_A}, L_A)$$

mit $\delta_{E_A}, \delta_{F_A} \subseteq VAL \times VAL$ und $L_A \in VAR$ für einen Automaten A wie folgend definiert:

$$A\Delta\mathcal{M}_{A(\delta)} = (I, O, L \cup L_A, S_0, (\delta \setminus \delta_{E_A}) \cup \delta_{F_A}, \Phi^{trigger}) \quad \lrcorner$$

Transitions-Modifikation

Das Hinzufügen von weiteren Variablen bringt eine Reihe von Aspekten bei der Modifikation mit sich, die unter Umständen aufwändig zu behandeln sind. So sind z.B. bei den Formeln für die bisherigen Transitionen eventuell zusätzlich die neuen Variablenwerte zu berücksichtigen. Eine ausführliche Beschreibung der zu berücksichtigenden Aspekte findet sich in [Bre01]. In einigen Fällen ist das Hinzufügen weiterer Variablen nicht notwendig. Es findet zwar oft eine Erweiterung des Zustandsraumes statt, diese kann jedoch eventuell mit neuen, noch nicht genutzten Werten der einzelnen Variablen ausgedrückt werden. Eine Modifikation lässt sich dann auf die Transitionen beschränken. In geeigneten Fällen kann eine Modifikation in der folgenden Kurzform gegeben werden (siehe [Bre01]):

Definition 4.2.11 (Transitionsmodifikation)
Sei A ein Transitionssystem, dann ist eine *Transitionsmodifikation*

$$\mathcal{M}_{A(\delta)} = (\delta_{E_A}, \delta_{F_A})$$

mit $\delta_{E_A}, \delta_{F_A} \subseteq VAL \times VAL$ für einen Automaten A wie folgend definiert:

$$A\Delta\mathcal{M}_{A(\delta)} = (I, O, L, S_0, (\delta \setminus \delta_{E_A}) \cup \delta_{F_A}, \Phi^{trigger}) \quad \lrcorner$$

Beispiel 4.2.9 (Transitionsmodifikation)
Eine in der Literatur häufig vorgefundene Modifikation eines Transitionssystems ist die Verklemmung. Hier besteht z. B. bei einem Elektromotor die Möglichkeit, dem Transitionssystem einen Übergang hinzuzufügen, der auf einen Fehlerzustand *klemmt* verweist, in dem er den alten Wert weiterhin sendet (vgl. [OR04]). Dieser Zustand wird hier durch die Modifikation $(\emptyset, (\mathit{bewegung}, \mathit{klemmt}))$ ausgedrückt. ⌟

Unabhängigkeit

Der Bezug der Modifikationen auf Transitionen, also einzelne Schritte in den Ausführungspfaden des Automaten, erlaubt eine Definition der Modifikationsunabhängigkeit, welche aus der allgemeinen Definition der Modifikationsunabhängigkeit abgeleitet werden kann. Zwei Modifikationen sind dann voneinander unabhängig, wenn nicht von einer Modifikation Transitionen entfernt werden, die von der anderen Modifikation hinzugefügt werden.

Definition 4.2.12 (Transitionsmodifikationsunabhängigkeit)
Die *Transitionsmodifikationsunabhängigkeit* $\overset{dep}{\#}$ zweier Modifikationen $\mathcal{M}^{(1)}_{A(\delta)}$ und $\mathcal{M}^{(2)}_{A(\delta)}$ ist gegeben, wenn gilt:

$$\overset{dep}{\#}: MOD_A(\delta) \times MOD_A(\delta) \rightarrow BOOL$$

$$\begin{aligned} \mathcal{M}^{(1)}_{A(\delta)} \overset{dep}{\#} \mathcal{M}^{(2)}_{A(\delta)} \quad \overset{def}{=} \quad & (\mathcal{M}^{(1)}_{A(\delta)} = (\delta^{(1)}_{E_A}, \delta^{(1)}_{F_A}) \\ & \wedge \mathcal{M}^{(2)}_{A(\delta)} = (\delta^{(2)}_{E_A}, \delta^{(2)}_{F_A}) \\ & \wedge \delta^{(1)}_{E_A} \cap \delta^{(2)}_{F_A} = \emptyset \\ & \wedge \delta^{(2)}_{E_A} \cap \delta^{(1)}_{F_A} = \emptyset) \end{aligned}$$

⌟

Formeln

Die Mengen δ, δ_{E_A} und δ_{F_A} sind mit den Formeln $[\![\delta]\!]$, $[\![\overline{\delta}_{E_A}]\!]$ und $[\![\delta_{F_A}]\!]$ beschreibbar (siehe Abschnitt 3.3.3). In diesem Fall kann die Definition des Automaten wie dort beschrieben in eine Blackbox-Spezifikation umgewandelt werden. Es gelten damit die Modifikationsoperatoren der Blackbox-Spezifikationen (siehe Abschnitt 4.2.1).

Beispiel 4.2.10 (Transitionsmodifikation eines Automaten)
Gegeben sei der Automat A_{UL} aus Beispiel 3.3.5.

Ein Implementierungsfehler dieses Automaten sei, dass das System vom Kontrollzustand

$$state_{UL} = \mathit{off}$$

direkt in den Kontrollzustand

$$state_{UL} = \mathit{on}$$

springt, und nicht, wie im Automaten spezifiziert in den Kontrollzustand

$$state_{UL} = \mathit{wait}.$$

Eine Modifikation $\mathcal{M}_A$, die diesen Fehler beschreibt, entfernt die korrekte Transition zu dem Kontrollzustand $state_{UL} = wait$. Die Menge δ_{E_A} ist also:

$$[\![\delta_{E_A}]\!] = (state_{UL} = off \wedge state'_{UL} = wait)$$
$$\text{bzw. } [\![\overline{\delta}_{E_A}]\!] = \neg(state_{UL} = off \wedge state'_{UL} = wait)$$

Hinzu kommt in dieser Modifikation die Transition vom Kontrollzustand $state_{UL} = off$ in den Kontrollzustand $state_{UL} = on$. Die Menge δ_{F_A} ist also:

$$[\![\delta_{F_A}]\!] = (state_{UL} = off \wedge sw = true \wedge state'_{UL} = on)$$

Die neue Transitionsmenge $\delta_{A_{UL}\Delta\mathcal{M}_A}$ ist definiert durch:

$$[\![\delta_{A_{UL}\Delta\mathcal{M}_A}]\!] = ([\![\delta_{A_{UL}}]\!] \wedge [\![\overline{\delta}_{E_A}]\!]) \vee [\![\delta_{F_A}]\!]$$ ⌟

permanent vs. temporär

Aufgrund des Bezugs der Fehler auf längere Zeiträume ist es wichtig, zu beschreiben, wie sich ein Fehler über die Zeit verhält. Hierbei gibt es im Wesentlichen zwei Punkte, die bei der Fehlermodellierung zu beachten sind. Ein Punkt ist die Angabe, wann ein Fehler auftritt. Das Auftreten eines Fehlers kann zu beliebigen Zeitpunkten zulässig sein, oder aber auf einen Zeitraum eingeschränkt werden. Ein Extremfall ist z.B. ein Implementierungsfehler, der bereits zum Zeitpunkt $t = 0$ erscheint. Umgekehrt ist zu beachten, wann das Auftreten eines Fehlers wieder aufgehoben wird. Auch hier kann das Zurückkehren in den Soll-Zustand auf einen Zeitraum eingeschränkt sein. Ein Extremfall ist hier der permanente Fehler, bei dem die Komponente nie mehr das Soll-Verhalten erreicht.

Fehlerklassen

Im Vergleich zu den Abweichungen zu aufgelösten Gleichungen aus Abschnitt 4.2.2, welche in generischen Klassen zusammengefasst wurden, sind die Transitionsmodifikationen meist spezifisch für den jeweiligen Automaten. Sie entsprechen von ihrem Charakter in etwa der Allgemeinheit der in Abschnitt 4.2.1 vorgestellten Modifikationen zu einzelnen Teilprädikaten. Es gibt im Wesentlichen zwei allgemeine Fehlerklassen. Eine Klasse ist die Verzögerung, also die Tatsache, dass Transitionen erst später schalten, als sie sollten. Eine zweite Klasse sind zustandsbezogene Wertabweichungen die eine Brücke zwischen den Abweichungen zu aufgelösten Gleichungen bilden. Im Folgenden werden die Verzögerungen vorgestellt. Die zustandsbezogenen Wertabweichungen werden im nächsten Abschnitt der Betriebsmodi vorgestellt.

Zeitverzug

Ein einfacher Fehler, der die Zustandsfolge eines Systems beeinflusst, ist der Zeitverzug, in dem ein System seinen Zustand nicht ändert. Grund für ein solches Verhalten kann zum Beispiel bei einem Kommunikationssystem eine gestörte Übertragung sein, welche dann erst zu einem späteren Senden des übertragenen Signals führt. Dieser Fehler kann als Modifikation so ausgedrückt werden, dass zu jedem Datenkontrollzustand eine Schleife existiert, welche diesen beibehält.

Definition 4.2.13 (zufällige Zeitverzögerung)
Gegeben sei ein Zustandsautomat $A_S = (I, O, L, S_0, \delta, \Phi^{trigger})$.

Eine *zufällige Zeitverzögerung* ist eine Modifikation $\mathcal{M}_{A(\delta)}^{(\mathtt{D},time)}$, so dass gilt:

$\mathcal{M}_{A(\delta)}^{(\mathtt{D},time)} = (\emptyset, \{(d,d)|d \subseteq \mathtt{D}\}$, wobei d einem Datenzustand entspricht. ⌟

Die oben definierte zufällige Zeitverzögerung ist eine sehr einfache Modifikation. Etwas komplizierter wird diese Fehlerbeschreibung bereits, wenn man die Verzögerung entweder nicht sporadisch, oder aber nur für eine bestimmte Zeitdauer zulassen möchte. In diesen Fällen müssen Zählzustände eingerichtet werden, welche die Dauer der Verzögerung festhalten.

Definition 4.2.14 (Zeitverzögerung)
Gegeben sei ein Zustandsautomat $A_S = (I, O, L, S_0, \delta, \Phi^{trigger})$.

Des Weiteren existieren Bijektionen p_i zwischen den Zustandsräumen k_i und k_{i+1}. Dabei seien in der Transitionsmenge δ nur die Zustände aus k_1 enthalten.

Eine *Zeitverzögerung* ist eine Modifikation $\mathcal{M}_{A(\delta)}^{(\mathtt{D},\mathsf{time}(\mathsf{n},\mathsf{m}))}$, so dass gilt:

$$\mathcal{M}_{A(\delta)}^{(\mathtt{D},\mathsf{time}(\mathsf{n},\mathsf{m}))} = (E_{\delta_A}, F_{\delta_A})$$

$$E_{\delta_A} = \begin{cases} \emptyset & \text{falls} \quad n = 0 \\ \{(\beta, \beta') | \beta \in \mathtt{D}\} & \text{falls} \quad n > 0 \end{cases}$$

$$F_{\delta_A} = \{(\beta, \beta') | \bigvee_{i \in [1..m]} (\beta' = p_i(\beta) \wedge ((i = 1) \Rightarrow \beta \in \mathtt{D})) \vee \bigvee_{i \in [n..m]} (\exists (\gamma, \beta') \in \delta : \beta = p_{1..i}(\gamma))\}$$ ⌟

Fehlereigenschaften

Die vorhergehenden Definitionen haben einen ersten Eindruck der Modifikationsmöglichkeiten von Automaten gegeben. Generelle Fehlerklassen sind eher schwer leserlich und kompliziert zu definieren, wie Definition 4.2.14 zeigt. Einfacher zu definieren sind spezifische Transitionen wie in Beispiel 4.2.10 zu sehen ist. Es gibt eine Reihe von Aspekten für Transitionsmodifikationen die bei der Definition berücksichtigt werden können. Hierzu gehören die Verzögerung der Zeit oder das Überspringen von Zuständen, die Eigenschaft, ob ein Fehler permanent oder temporär ist, ob er sporadisch auftreten kann oder sicher auftritt. In Abschnitt 4.5 finden sich einige in den Fallstudien vorgefundene Fehler zu Transitionssystemen.

Fehlerermittlung

Für die Fehlerermittlung von Transitionsmodifikationen gibt es zum Einen allgemeine Fehler, die sich aus den Implementierungselementen motivieren und wie Systemmodifikationen ermittelt werden. Die spezifischen Fehler können zum Anderen nicht allgemein abgeleitet werden. Hier muss spezifisch ein Weg gefunden werden, Transitionen individuell zu entfernen oder hinzuzufügen. Diese können entweder aus Fehlerdatenbanken abgeleitet werden (siehe [ES07a]) oder aus Szenarien, die Anforderungen widerspiegeln (siehe [PP08]).

4.2.4 Modifikation von Betriebsmodi-Automaten

Modifikationsarten

Modifikationen von Betriebsmodi-Automaten sind entweder Transitionsmodifikationen oder Modifikationen der einzelnen Subsysteme für die Betriebsmodi. In beiden Fällen kann durch das Wissen der Konstruktion einer Betriebsmodikomponente die Information zu Fehlern, mit dem Ziel diese besser interpretieren zu können, gezielter beschrieben werden. Im Falle einer Transitionsmodifikation zeigt sich die Auswahl eines falschen Modus nur indirekt, in dem die Ausgaben des falschen Subsystems weitergegeben werden. Im Falle eines Fehlers in einem Subsystem kann, wenn die Modi trotzdem weiterhin korrekt sind, die Wirkung auf bestimmte Situationen eingegrenzt werden, in denen das Subsystem aktiv ist.

Sind die Subsysteme datenflussorientiert, so sind die Fehler, die in den jeweiligen Modi verwendet werden als Abweichungen wie in Abschnitt 4.2.2 modelliert. Hier muss, wie im letzten Abschnitt angedeutet wurde, eine Brücke zwischen den Automaten und den aufgelösten Gleichungen geschlagen werden. Eine zustandsbezogene Abweichung ist das Senden falscher Werte in einer Datenkontrollzustandsmenge $\mathtt{D} = \{d_1, d_2, ...\}$ des Systems. Die Datenkontrollzustände entsprechen hierbei dem Betriebsmodi-Automaten. Modelliert werden diese Fehler, in dem alle Transitionen, die in die Datenkontrollzustandsmenge mit einer korrekten Ausgabe führen, nun in die Datenkontrollzustandsmenge mit einer inkorrekten Ausgabe führen. Im nächsten Rechenschritt wird von diesem Datenkontrollzustand in einen Datenzustand gesprungen der einem Folgezustand des eigentlichen Sollzustandes entspricht, oder in einen fehlerhaften Folgezustand.

zustandsbezogene Abweichung

Definition 4.2.15 (Zustandsabhängige Abweichung)
Gegeben sei ein Zustandsautomat $A_S = (I, O, L, S_0, \delta, \Phi^{trigger})$.

Ein *unabhängiger Nachrichtenfehler* ist eine Modifikation

$$\mathcal{M}_{A(\delta)}^{(\mathtt{D},[\![F_o]\!])} = (\delta_{E_A}^{(\mathtt{D},[\![F_o]\!])}, \delta_{F_A}^{(\mathtt{D},[\![F_o]\!])})$$

mit $[\![F_o]\!]$ einer zeitunabhängigen Wertabweichung (Abschnitt 4.2.2),
o dem Namen des Ausgabekanals, an dem der Fehler auftritt und
$\mathtt{D}$ der Datenkontrollzustandsmenge, in der der Fehler auftritt.

Des Weiteren gelte, dass die modellierten Transitionen unabhängig von o sind:

$$\forall k : (k, l') \in \delta \Rightarrow o \notin V_k$$

Dann gilt:

$$\Phi_{fail\_o} = [\![F_o]\!][\gamma.o/\tau_o][\beta'.o/o] \text{ und } V = (I \cup O \cup L) \setminus \{o\}$$

$$\delta_{E_A}^{(\mathtt{D},[\![F_o]\!])} = \{(\beta, \beta') | \beta' \in \mathtt{D}\}$$

$$\delta_{F_A}^{(\mathtt{D},[\![F_o]\!])} = \{(\beta, \beta') | \exists (\beta, \gamma) \in \delta : (\gamma \in \mathtt{D} \wedge \beta' \stackrel{V}{=} \gamma \wedge \Phi_{fail\_o}\}$$ ⌟

Unabhängigkeit

Bei den zustandsabhängigen Abweichungen soll der eigentliche Ablauf des Betriebsmodi-Automaten nicht gestört werden. In der oben gegebenen Definition ist dies sichergestellt, in dem die geänderte Ausgabe o keine Wirkung auf die Auslösung von Transitionen hat. Möchte man diese Wirkung nicht ausschließen, so wird die Definition der zustandsabhängigen Abweichung deutlich komplizierter.

Beispiel 4.2.11 (zustandsbezogene Abweichung)
Gegeben sei der Automat A_{RDS} aus Beispiel 3.3.5.

Ein Fehler sei, dass das System im Kontrollzustand $state_{RDS} = active$ unerwünscht die Bestätigung $conf = true$ sendet.

Die zeitunabhängige Abweichung, die hier zugrunde liegt, ist die Ausgabemodifikation (siehe Definition 4.2.7), bei der der Wert *true* statt dem Wert *false* (siehe Tabelle 4.1: TsF) am Kanal *conf* anliegt, also : $\mathcal{M}_{conf}^{TsF}$

Die zustandsabhängige Abweichung für den Automaten A_{UL} ist entsprechend:

$$\mathcal{M}_{A_{UL}(\delta)}^{D,[\![F_{conf}^{TsF}]\!]} \text{ mit } [\![D]\!] = (state = active)$$ ⌟

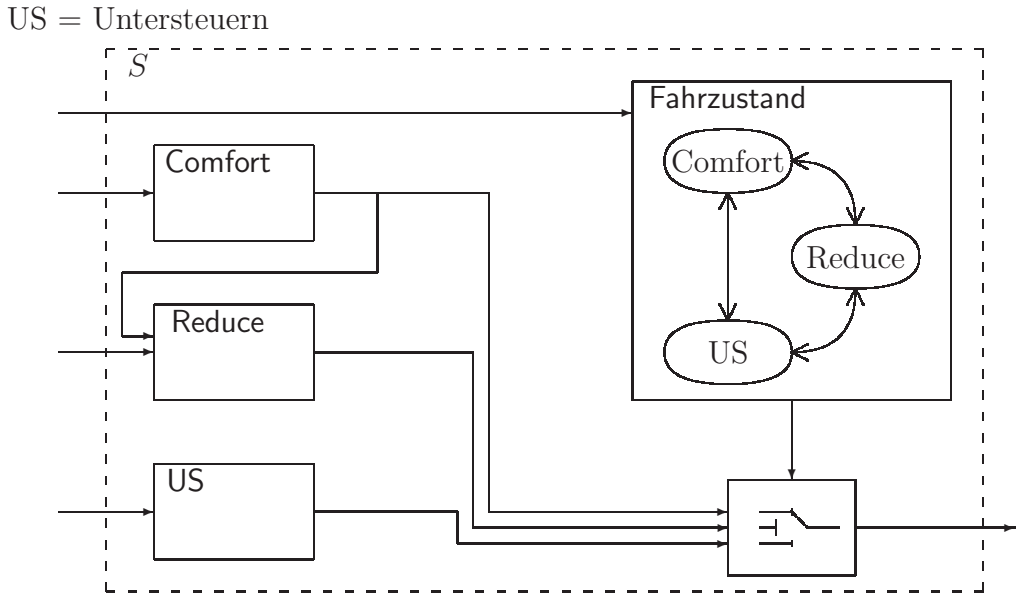


Abb. 4.8: Beispiel für Betriebsmodi-Fehler (siehe Bsp. 4.2.12)

Widerspiegeln der Umgebungssituation

Die einfachste Variante, das Wissen über Betriebsmodi zu nutzen, ist, diese als Vermerk an die Beschreibung eines Fehlers anzuhängen. Häufig werden Fehler als Ausgabemodifikationen beschrieben. Diese Ausgaben kommen aber, je nach aktuell gültigem Betriebsmodus, aus verschiedenen Subsystemen, welche an entsprechende Umstände der Umgebung angepasst sind (siehe auch Abschnitt 2.3). Unter der Annahme, dass die Modi die Umstände der Umgebung korrekt widerspiegeln, kann so die Folge einer Abweichung inklusive derer Bewertung spezifisch verschieden sein. Verdeutlicht wird dies an folgendem Beispiel:

Beispiel 4.2.12 (Widerspiegelung der Situation in der Umgebung)
Gegeben sei das System aus Abbildung 4.8. Dieses berechnet den Überlagerungswinkel *lw_add*, welcher situationsbedingt zum Lenkwinkel addiert werden soll. Das System kennt hierbei drei Betriebsmodi:

(1) Normalbetrieb (*Comfort*): Das System berechnet den zu addierenden Überlagerungswinkel, so dass das Fahrzeug eine geschwindigkeitsabhängige Lenkübersetzung hat.

(2) Untersteuern (*US*): Das System berechnet den zu addierenden Überlagerungswinkel entsprechend der Stabilisierungsfunktion bei Untersteuern.

(3) Reduzierter Eingriff (*Reduce*): Aufgrund hoher physikalischer Belastungen des Systems wird ein reduzierter Überlagerungswinkel für die geschwindigkeitsabhängige Lenkübersetzung berechnet.

Aufgrund von Programmierfehlern sind zu kleine berechnete Winkel möglich. Unter der Annahme, dass die Betriebsmodi die Situation des Fahrzeugs korrekt widerspiegeln, hat der gleiche Fehler verschieden kritische Wirkungen:

$\mathcal{M}^{BzK}_{lw\_add.Comfort}$ ⇢ 'Komfort reduziert'.

$\mathcal{M}^{BzK}_{lw\_add.US}$ ⇢ 'Kontrollierbarkeit reduziert.'

Mit Hilfe der Angabe der Modi kann so gezielt Hilfestellung zum Auffinden von Ursachen für Abweichungen der Ausgaben gegeben werden. ⌟

Bezug zu Ausgaben

Die Angabe der Betriebsmodi zu Modifikationen ist nur bedingt sinnvoll. Prinzipiell erschließt sich der Betriebsmodus aus den bisher erfolgten Eingaben. Wurde ein System mit allgemeinen Blackbox-Spezifikationen definiert, welche noch weitgehend abstrakt sind, so ist in dem einzelnen Teilprädikaten der Betriebsmodus einfach erschließbar. Handelt es sich aber bereits um detailliertere Modelle, welche im Sinne der Ausführbarkeit bereits zu den Ausgabekanälen hin aufgelöst sind, so sind die Fehlerbeschreibungen häufig Ausgabemodifikationen. In diesen Fällen ist der Bezug zu den Eingaben nur schwer zu beschreiben. Hier dienen die Informationen über die verschiedenen Modi als Stütze, um diese Zusammenhänge abstrakt festzuhalten.

Ausgaben der Subsysteme

Bezieht man sich bei der Beschreibung der Modifikationen auf Abweichungen der Ausgaben, so ist der Bezugspunkt bisher die Menge der Ausgaben des Systems gewesen. Diese Beschreibung der Modifikationen kann präzisiert werden, in dem als Bezugspunkte zusätzlich die Sollwerte der anderen Modi für eine Ausgabe mit zur Hilfe genommen werden können. Gerade bei Fällen, in denen die Modi eine Wirkung mit einer oberen und unteren Schranke eingrenzen können, ist dieser Mechanismus geeignet.

Beispiel 4.2.13 (Nutzung des Bezugs zwischen Modi)
Gegeben sei das System aus Beispiel 4.2.12 und das folgende Szenario: Im reduzierten Modus wird der Wert der Ausgabe des Teilsystems 'Comfort' mit einem Faktor zwischen 0 und 1 multipliziert, also passend reduziert. Ein Fehler bei der Ermittlung der Höhe der Belastung, welcher zu einer zu kleinen Reduktion führt, bedeutet, bezogen auf die Ausgabe $lw\_add.Reduce$, dass diese relativ zum Sollwert zu groß ist:

$$\mathcal{M}_{lw\_add.Reduce} = (|lw\_add.Reduce| \geq |\tau_{lw\_add.Reduce}|)$$

Durch das Wissen über die Modi kann dieses Verhältnis nun genauer beschrieben werden, da durch den Maximalfaktor 1 der Wert nicht größer als der im Modus $Comfort$ werden kann. Der Fehler ist dann eingegrenzt auf:

$$\mathcal{M}_{lw\_add.Reduce} = (|\tau_{lw\_add.Comfort}| \geq |lw\_add.Reduce| \geq |\tau_{lw\_add.Reduce}|)$$

Ebenso kann ein Fehler im Automaten der Betriebsmodikomponente, bei dem ein Modus nicht aktiviert wurde, einfach an den Ausgaben notiert werden. So wird eine nicht eingeschaltete Sicherheitsfunktion US ausgedrückt als:

$$\mathcal{M}_{lw\_add.US} = (lw\_add.US = \tau_{lw\_add.Comfort})$$ ⌟

Vereinfachung der Beschreibung

Mit der Möglichkeit, als Bezugspunkte für die Fehlerbeschreibungen Modi zu verwenden, kann so mit einfachen Mitteln eine Abweichung in einem System beschrieben werden, welche in anderen Fällen erst sehr aufwändig modelliert werden müsste.

Einen Einblick über die Größe des Aufwands bekommt man in [SSL+96]. Dort werden die Abweichungen der Ausgaben aufgrund einer Transitionsmodifikation erst spät und nur mit Hilfe eines Diagnose-Automaten ermittelt.

4.3 Fehler in der Implementierungsstruktur

Fehler als Tupel

Sinn der Modelle der Implementierungsstruktur ist die Angabe, in welchen Artefakten das spezifizierte Verhalten implementiert ist. Anhand des Wissens über die Eigenschaften dieser Artefakte können potentielle Fehler identifiziert werden, die eine Modifikation des Verhaltens zur Folge haben. Ein Fehler in der Implementierungsstruktur wird im Weiteren als *Implementierungsartefaktfehler* bezeichnet. Formal ist dieser folgendermaßen definiert:

Definition 4.3.1 (Implementierungsartefaktfehler)
Die Menge $\mathbb{E}$ der Implementierungsartefaktfehler ist:

$\mathbb{E} \stackrel{def}{=} \mathbb{I} \times \mathbb{L}$ mit
der Menge der Implementierungsartefakte $\mathbb{I}$ (siehe Abschnitt 3.2.2) und
der Menge der Fehlerlabel $\mathbb{L}$.

Ein Implementierungsartefaktfehler $\mathcal{E}$ ist ein 2-Tupel:
$\mathcal{E} \stackrel{def}{=} (\mathcal{I}, \mathcal{L})$ mit $\mathcal{E} \in \mathbb{E}$ ⌟

Gibt es für die Modelle der Implementierungsstruktur ein Metamodell in Form eines Klassenmodells (vgl. UML-Klassenmodell [Fow03]), so können auch den Klassen Fehler zugeordnet werden, welche dann an die einzelnen Artefakte weiter vererbt werden. Beispiele für diese Klassen sind in Abschnitt 4.5.2 zu finden. Die Fehlerbeschreibungen sind wie folgt definiert:

Definition 4.3.2 (Implementierungsartefaktklassenfehler)
Die Menge $\mathbb{E}$ der Implementierungsartefaktklassenfehler ist:

$\mathbb{E}^{class} \stackrel{def}{=} \mathbb{C} \times \mathbb{L}$ mit
der Menge der Implementierungsartefaktklassen $\mathbb{C}$ (siehe Abschnitt 3.2.2) und
der Menge der Fehlerlabel $\mathbb{L}$.

Ein Implementierungsartefaktfehler $\mathcal{E}^{class}$ ist ein 2-Tupel

$\mathcal{E}^{class} \stackrel{def}{=} (\mathcal{C}, \mathcal{L})$ mit $\mathcal{E}^{class} \in \mathbb{E}^{class}$,

welches an die Instanzen weiter vererbt wird:

$\forall \mathcal{I} \in \mathbb{I}, \mathcal{C} \in \mathbb{C}, \mathcal{L} \in \mathbb{L} : (((\mathcal{C}, \mathcal{I}) \in CI) \wedge ((\mathcal{C}, \mathcal{L}) \in \mathbb{E}^{class})) \Rightarrow ((\mathcal{I}, \mathcal{L}) \in \mathbb{E}))$ ⌟

Kategoriensuche

Eine weitere Systematik, potentielle Fehler zu identifizieren, ist die Orientierung an Fehlerkategorien. Für jede Kategorie kann überprüft werden, ob diese Fehler auftreten können. Eine Einteilung nach [Ech90] ist:

- Entwurfsfehler (Das System ist von vornherein fehlerhaft.)
  - Spezifikationsfehler
    Diese Art der Fehler wird in dieser Arbeit nur hinsichtlich der Nicht-Erfüllung der Anforderungen (insbesondere der Sicherheitsanforderungen) aufgrund von Fehlerursachen auf tieferen Ebenen betrachtet.
  - Implementierungsfehler
    Die Implementierung kann von der Spezifikation abweichen. Gründe können Flüchtigkeitsfehler oder Fehlinterpretationen sein. Dies umfasst auch die Programmier- und Generierungsfehler nach [Mus99], S. 82 mit den Kategorien 'defekte', 'fehlende' oder 'zusätzliche' Programmzeilen[5] oder Kombinationen davon.
- Herstellungsfehler
  Unter diesen Fehlern versteht man die Abweichung eines Produktes von seinem Entwurf. Hierzu gehören fertigungstechnische Mängel, wie Maskenfehler bei der Chip-Herstellung, allgemeine Exemplarstreuung bei der Hardwareherstellung oder Kopierfehler bei dem Software-Deployment.
- Betriebsfehler
  - störungsbedingte Fehler
    Fehler, die aufgrund äußerer Einflüsse auftreten. Diese Fehler können mit gleicher Wahrscheinlichkeit über die gesamte Betriebsdauer eintreten. Folgende Störungsarten werden unterschieden: mechanisch, elektromagnetisch, elektrisch, magnetisch und thermisch.
  - Verschleißfehler
    Mit zunehmender Betriebsdauer treten bei der Hardware Verschleißerscheinungen auf, die bei entsprechender Intensität zu Fehlern führen können.
  - zufällige physikalische Fehler
    Die Funktion von Hardwarekomponenten kann ohne Verschleiß physikalisch bedingt ausfallen. Zu diesen Fehlern gehören z.B. Übertragungsfehler.

Eine mehr auf die Verhaltensabweichung konzentrierte Kategorisierung ist nach [MP01]:
- Dienstleistungsfehler (Signalausfall, unerwünschte Signalbelegung),
- Timing-Fehler (zu frühes oder zu spätes Senden),
- Wertfehler (Bereichsüberschreitung, Verklemmung, Verzerrung, lineares oder nichtlineares Abdriften, unberechenbare Werte) und
- Betriebsmodi-Fehler

[5] Beim Generieren der Software eines Steuergerätes verschwanden ganze Programmzeilen.

Design-Fehler

Die in der Implementierungsstruktur festgehaltenen Fehler werden als Designfehler verstanden (siehe Abschnitt 2.1.4). Auch wenn die Label der Fehler tendenziell eher auf Fehlerereignisse (Definition 2.1.2) und Fehlerzustände hinweisen, ist die Semantik ein Design-Fehler. Dies ist insbesondere wichtig, da in der Implementierungsstruktur die Abhängigkeiten zwischen den Fehlerbeschreibungen nicht zeitlich kausal geordnet sind, sondern nur angegeben wird, ob ein anderer Fehler auftreten kann, wenn ein verursachender Fehler auftritt. Diese sehr einfache Semantik ermöglicht eine schnelle Modellierung von Fehlerzusammenhängen, ist aber sehr abstrakt und kann kausale Wirkungen der Fehlerereignisse und Zustände nicht differenzieren. Welche Fehlerzusammenhänge in der Implementierungsstruktur sinnvoll modelliert werden, und welche Zusammenhänge durch die abstrakte Semantik nicht modelliert werden können, wird in Abschnitt 5.3 erläutert.

Ausblick auf Wirkung

Die Fehler, die in der Implementierungsstruktur als potentielle Fehler identifiziert werden, haben eine Modifikation des Verhaltens zur Folge. Wie diese Folge modelliert wird, ist in Abschnitt 4.5.2 zu finden. In Abschnitt 4.5.2 sind zum besseren Verständnis der Fehler Modifikationsklassen angegeben, wie ein mögliches Fehlverhalten als Folge aussehen kann.

4.4 Fehlerinduktion in Verhaltensmodelle

Um das Verhalten eines Systems $S\Delta\mathcal{M}$ zu erhalten, kann verschieden vorgegangen werden. Zum einen kann das modifizierte Verhalten komplett modelliert werden und als Ersatzkomponente in das System eingefügt werden. Da Modifikationen aber auf verschiedene Systeme angewendet werden können, bietet es sich an, bei der Modellierung das modifizierte Verhalten durch die Komposition des korrekten Systems S mit einer Modifikationskomponente $M_{\mathcal{M}}$ darzustellen. Die Modifikationskomponente kann dabei auf verschiedene Weisen mit dem System komponiert werden. Sie kann entweder (I) vor das korrekte System gestellt werden, (II) hinter das korrekte System gestellt werden, oder (III) das korrekte System umgeben (siehe Abbildung 4.9).

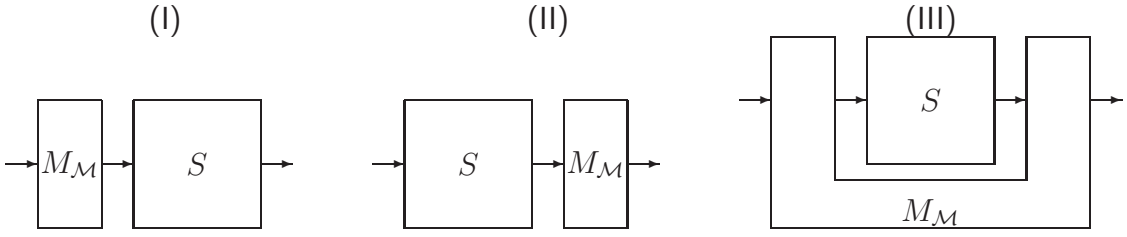


Abb. 4.9: Arten der Induktion von Fehlern

Ist die Modifikationskomponente vor das korrekte System gestellt, so können Fehler bei Eingaben von der Umwelt modelliert werden. Diese Art der Modellierung kann zum Beispiel genutzt werden, um Wirkungen falscher Eingaben auf die Ausgaben zu betrachten und so ein Fehlerauswirkungmodell (siehe Abschnitt 5.1.2) zu verifizieren.

Das Anhängen der Modifikationskomponente hinter dem korrekten System (II) ist eine der meist genutzten Varianten bei der Fehlermodellierung. Hier wird die Soll-Ausgabe des Systems abgeändert. Die in Abschnitt 4.2.2 beschriebenen Basis-Fehlerklassen können alle mit diesen Komponenten beschrieben werden. Ebenso gibt es eine Reihe weiterer Fehler, wie Knickstellen oder Unstetigkeiten, die mit diesen nachgeschalteten Komponenten dargestellt werden können.

Einige Modifikationen fordern jedoch sowohl Eingriffe bei den Eingaben, wie auch bei den Ausgaben des korrekten Systems (III). Mit dieser Variante können beliebige Modifikationen ausgedrückt werden. Bei Modifikationen dieses Umfangs ist meist abzuwägen, ob die Realisierung mit einer Modifikationskomponente oder die Ersetzung des Systems durch das fehlerhafte System mit weniger Aufwand verbunden ist.

4.5 Spezifische Fehlerbilder aus Fallstudien und Literatur

Hier werden spezifische Fehlerbilder, welche in der Fallstudie der Software eines Fahrwerkregelsystems vorgefunden wurden, aufgelistet und klassifiziert. Die Fehlerbilder wurden zum Einen an der Nutzungsschnittstelle der Regelungssoftware der beiden Fallstudien gesammelt, und zum Anderen sind es Sammlungen allgemeiner Fehler auf Ebene der Implementierung.

4.5.1 Fehlverhalten an der Nutzungsschnittstelle der Software

In diesem Abschnitt werden spezifische Fehlerbilder an den Kanälen der Software gelistet und klassifiziert. Zu jeder Klasse wird berichtet, wie sie in der Fallstudie vorgefunden wurde und wie sie als Modifikation mit den im vorherigem Abschnitt genannten Modellierungstechniken spezifiziert werden können. Tabelle 4.2 gibt einen Auszug aus konkreten Fehlerbeschreibungen, und den Klassen, denen Sie zugeordnet wurden. Im Folgenden werden die Fehlerbilder beschrieben[6].

[6]Zur Anonymisierung steht die Zahl 10 als Repräsentant für die Werte in der Fallstudie

Fehler an Eingaben/Ausgaben:	*Fehlerklasse*:
Der Betrag des Moments wird größer als $10Nm$	Schwellwertüberschreitung
Der Betrag des Momentengradienten ist größer $10Nm/s$	Gradientenüberschreitung
Es wird ein zu kleines Moment berechnet	absolute Abweichung
Das Moment wird zu stark reduziert (Schutzfunktion)	absolute Abweichung
Es wird ein zu großes Moment berechnet	absolute Abweichung
Das Moment ist geringfügig falsch ($< 10\%$)	relative Abweichung
Das Flag meldet true für mindestens $10ms$	Wertabweichung mit Latenzzeit
Kein Nulldurchlauf innerhalb der ersten Minute	Wertabweichung mit Zeitbezug
Es wird kein Moment berechnet	konstanter Wert
Kein Moment und gradientenbegrenzter Rückgang auf Soll	(Inverse-)Drift
Es wird fälschlicherweise ein Moment berechnet	Modusfehler
Das Moment wird nicht reduziert (Schutzfunktion)	Modusfehler
Das Moment entspricht einem falschem Fahrprogramm	Modusfehler
Eingeschränkter Fahrprogrammwechsel	Modusfehler
Das Moment wird zu schwach reduziert (Schutzfunktion)	Modus-Wertabweichung

Tabelle 4.2: Fehler an der Nutzungsschnittstelle

4.5.1.1 Schwellwertüberschreitung

Beschreibung: Der Wert eines Signals überschreitet eine bestimmte Obergrenze, bzw. eine bestimmte Untergrenze.

Vorkommen in der Fallstudie: Dieses Fehlerbild wird in Monitoring-Komponenten eingesetzt, um ein System in den Fail-Safe-Zustand zu versetzten, oder bei Redundanz als fehlerhaft identifizierte Signale auszublenden.

Modellierung als Modifikation: Bezieht sich der Fehler auf eine Verhaltensbedingung Φ_a welche die Einhaltung der Schwellwerte fordert, so kann diese mit einer konjunktiv eingeschränkten Modifikation $\mathcal{M}_{\Phi_a}^{Bayes}$ (Abschnitt 4.2.1) aufgehoben werden. Bezieht sich der Fehler allerdings auf ein ausführbares Modell im Sinne einer aufgelöster Gleichung Φ_k, so muss eine geeignete Wertabweichung (Abschnitt 4.2.2) gefunden werden, um die Wirkung möglichst präzise zu erfassen. So kann z.B. eine zeitlose Wertabweichung eines zu großen Betrages $\mathcal{M}_k^{Betrag}$ zusätzlich eine Gradientenverletzung mit sich bringen, was bei einem Drift $\mathcal{M}_k^{\text{drift}}$ nicht der Fall wäre.

4.5.1.2 Gradientenüberschreitung

Beschreibung: Der Gradient eines Signals überschreitet eine bestimmte Obergrenze, bzw. eine bestimmte Untergrenze.

Vorkommen in der Fallstudie: Dieses Fehlerbild wird in Monitoring-Komponenten eingesetzt, um ein System in den Fail-Safe-Zustand zu versetzten, oder bei Redundanz fehlerhaft identifizierte Signale auszublenden.

Modellierung als Modifikation: Bezieht sich der Fehler auf eine Verhaltensbedingung Φ_a welche die Einhaltung der Schwellwerte fordert, so kann wie bei der Schwellwertüberschreitung verfahren werden. Auch kann für eine aufgelöste Glei-

chung Φ_k entweder eine allgemeine Wertabweichung verwendet werden oder, um die Verletzung sicherzustellen, eine Modifikation mit einem minimalen Gradienten für einen Drift herangezogen werden.

4.5.1.3 absolute Wertabweichung

Beschreibung: Der Wert eines Signals weicht von dem Sollwert ab. Die Abweichung kann detaillierter in positive und negative Richtung, so wie in ihrer Stärke unterschieden werden.

Vorkommen in der Fallstudie: Dieses Fehlerbild wird für Regelungen eingesetzt, um eine maximale Abweichung einzugrenzen. In Testfällen wird die absolute Abweichung zum Vergleich der Ausgaben mit den Referenzausgaben von Steuergrößen verwendet. In der Fehleranalyse wird das Fehlerbild noch unterschieden, um anzugeben in welche Richtung die Abweichung stattfindet.

Modellierung als Modifikation: Modelliert wird diese Abweichung meist mit den im Abschnitt 4.2.2 beschriebenen zeitunabhängigen Wertabweichungen, welche sich aus Basisklassen zusammensetzen können. Je nach Verwendung (Test, exakte Wirkung, Verletzung der Sicherheitsanforderungen) kann hier die Modellierung mit Toleranzbereichen versehen sein.

4.5.1.4 relative Wertabweichung

Beschreibung: Der Wert eines Signals weicht von dem Sollwert ab. Die Abweichung wird dabei im Vorzeichen und relativ zum Betrag des Sollwerts unterschieden. Da in Nähe des Sollwertes 0 auch die relative Abweichung gegen 0 geht, wird üblicherweise eine untere Grenze bestimmt, ab der dann der Fehler absolut modelliert wird.

Vorkommen in der Fallstudie: Dieses Fehlerbild wird für Regelungen eingesetzt, um eine maximale Abweichung einzugrenzen. Es kommt häufig in Zusammenhang mit physischen Größen vor, in denen mit steigendem Betrag auch die Unschärfe relativ, ohne als Fehler betrachtet zu werden, wachsen kann. Auch im Kontext der Numerik können Abweichungen bei der Abbildung von kontinuierlichen Formeln auf die 'diskreten' Fließkommawerte entstehen, welche mit diesem Fehlerbild überprüft werden.

Modellierung als Modifikation: Modelliert wird diese Abweichung meist mit den im Abschnitt 4.2.2 beschriebenen zeitunabhängigen Wertabweichungen für relaxierte Gleichungen.

4.5.1.5 Wertabweichung mit Latenzzeit

Beschreibung: Der Wert eines Signals weicht von dem Sollwert am Stück für die Latenzzeit ab. Die Latenzzeit kann dabei auf eine Mindest- und eine Höchstzeit eingegrenzt werden.

Vorkommen in der Fallstudie: Die Regelsysteme sind häufig mit Mechanismen ausgestattet, welche die Wirkung kurzzeitiger Fehler dämpfen. Durch Angabe der Latenzzeit kann so die Wirkung unterschieden werden. Die in der Fallstudie vorkommenden Latenzzeiten waren $100ms$(5 Zeitschritte), $500ms$ und $3s$.

Modellierung als Modifikation: Modelliert wird diese Abweichung mit einer Modifikation, welche sich, ähnlich wie der Drift, in einer lokalen Variable den Startzeitpunkt der Abweichung merkt und dann einen Rücksprung erst innerhalb der Grenzen zulässt. Werden als Fehlerbild die zeitunabhängigen Wertabweichungen verwendet, so kann ein Fehlerbild nur dann auftreten, wenn die Bedingung dafür erfüllt ist. Ob diese Bedingung immer in der Latenzzeit erfüllt ist, muss über Laufzeitbedingungen bestimmt werden.

4.5.1.6 Wertabweichung mit absolutem Zeitbezug

Beschreibung: Der Wert eines Signals weicht von dem Sollwert am Stück für die Latenzzeit ab. Diese Latenzzeit beginnt zu einem bestimmten Punkt der absoluten Systembetriebszeit.

Vorkommen in der Fallstudie: Gerade verteilte Systemen müssen ihre Teilsysteme bei der Initialisierung aufeinander abstimmen. Ebenso ist bei eingebetteten Systemen in der Initialisierungsphase häufig eine Justierung notwendig, welche nur unter Referenzbedingungen möglich ist. Hier kommen Fehler zum Tragen, die die Bedingungen zum Abstimmen der Teilsysteme in der Initialisierungszeit verhindern.

Modellierung als Modifikation: Modelliert wird diese Abweichung meist mit den im Abschnitt 4.2.2 beschriebenen zeitunabhängigen Wertabweichungen, wobei die Bedingung für die Abweichung zusätzlich prüft, ob das zu der Modifikation gehörende Zeitfenster aktiv ist. Die Systemzeit wird analog der Operationalisierung (siehe Definition 3.3.3) modelliert.

4.5.1.7 konstanter Wert

Beschreibung: Statt dem Sollwert liegt ein konstanter Wert an. Dieser Fehler kann in dem Wert der Konstante unterschieden werden, bzw. auch allgemeiner als ein Wertebereich, in dem sich das Ausgabesignal befindet, definiert werden.

Vorkommen in der Fallstudie: Dieses Fehlerbild wurde in der Regelung sowohl mit dem Wert 0 vorgefunden, um Signalausfälle oder Abschaltverhalten auszudrücken, wie auch mit anderen konstanten Werten, um gezielt falsche Nachrichten auszudrücken.

Modellierung als Modifikation: Auch dieser Fehler kann wie die zuvor genannten Wertabweichungen modelliert werden. Er kann auch in Kombination mit Latenzzeiten und relativ zur absoluten Betriebszeit definiert werden.

4.5.1.8 Drift und Annäherung

Beschreibung: Statt dem Sollwert nähert sich der Signalwert mit einem begrenzten Gradienten entweder einem bestimmten konstanten Wert oder einer Abweichung vom Sollwert. Die Grenzen für den Gradienten können sich dabei entweder auf Werte oder Beträge beziehen, wie auch wahlweise auf die Abweichungen oder absolut auf die Signalwerte.

Vorkommen in der Fallstudie: Gerade an der Nutzungsschnittstelle von Fahrwerkregelsystemen spielt die Beherrschbarkeit des Fahrzeugs eine wesentliche Rolle. Gradientenbegrenzte Fehler sind meist wesentlich besser zu beherrschen, als sprunghafte Abweichungen. Diese Fehler werden so oft als gewünschte (Fail-Safe-) Fehlerfolge in Analysen aufgenommen.

Modellierung als Modifikation: Fehler mit einem komplexeren Verhalten über die Zeit müssen meist zusätzlich mit lokalen Variablen ausgestattet werden. Eine allgemeine Beschreibung ist hier schwierig. Eine mögliche Definition ist in Beispiel 4.2.8 zu finden.

4.5.1.9 Modusfehler

Beschreibung: Statt dem Sollwert, den ein Signal entsprechend einer aktiven Teilkomponente haben sollte, hat es die Werte einer anderen Teilkomponente. Unterschieden werden können diese Fehler in den jeweiligen Modi, auf welche sie sich beziehen.

Vorkommen in der Fallstudie: Dieses Fehlerbild wurde in der Regelung in Fällen vorgefunden, in denen bestimmte Situationen nicht erkannt werden oder Bedingungen nicht erfüllt werden, welche dann nicht zu beliebigem Fehlverhalten führen, sondern zu einem für die Situation nicht passendem Verhalten.

Modellierung als Modifikation: Je nach Verwendungszweck kann ein Modusfehler auf verschiedene Arten modelliert werden. Zum einen kann in einer White-Box-Sicht der Automat, welcher die Modi steuert mit einer Transitionsmodifikation (Abschnitt 4.2.3) abgeändert werden. Steht der Algorithmus des Automaten nicht im Vordergrund, sondern lediglich die Tatsache, dass ein falscher Modus aktiv ist, so können die Betriebsmodimodifikationen aus Abschnitt 4.2.4 verwendet werden, welche einfacher als Blackbox-Modifikationen zu gestalten sind.

4.5.1.10 Modus-Wertabweichung

Beschreibung: Statt dem Sollwert, den ein Signal entsprechend einer aktiven Teilkomponente haben sollte, hat es eine absolute oder relative Abweichung zu anderen inaktiven Teilkomponenten. Unterschieden werden können diese Fehler in den jeweiligen Modi, auf welche sie sich beziehen und in der Art der Abweichung.

Vorkommen in der Fallstudie: Dieses Fehlerbild wurde in der Regelung bei der Ausgabe der Regelungssoftware vorgefunden, um die Art der Abweichung besonders hinsichtlich ihrer Wirkung einschränken zu können. Häufig kann die Abweichung zwischen zwei Modi eingegrenzt werden.

Modellierung als Modifikation: Diese Modifikation wird als eine Betriebsmodimodifikation aus Abschnitt 4.2.4 definiert. Meist sind es Wertabweichungen, die auf die Ausgaben mehrerer Teilkomponenten Bezug nehmen.

4.5.2 Fehler in der Implementierungsstruktur

Im Folgenden wird eine Sammlung von Fehlern auf Implementierungsebene angegeben, welche in der Literatur ([BSN07], [MP01], [Mus99], [MK05]) und in den Fallstudien vorgefunden wurden. Zu jedem Fehler wird zusätzlich noch eine Modifikation aus den in dieser Arbeit definierten Modifikationstechniken zugeordnet. Der letzte Punkt 'Applikationslogik' ist aus der Informatik-Sicht interessant, da dessen Ursache in einer oder mehreren Phasen der Software-Entwicklung liegt.

Artefakt-Klasse $\mathcal{C} \in \mathbb{C}$	Fehlerlabel $\mathcal{L} \in \mathbb{L}$	Referenzmodifikation $\mathcal{M} \in MOD$	
Empfänger	Daten werden nicht empfangen	0	(Ausgabe)
	Daten werden verzögert emfangen	delay	(Transition)
	unwerwünschtes Empfangen von Daten	$\neq 0$	(Ausgabe)
Sender	Daten werden nicht gesendet	0	(Ausgabe)
	Daten werden verzögert gesendet	delay	(Transition)
	unwerwünschtes Senden von Daten	$\neq 0$	(Ausgabe)
Sensor	Justierfehler	drift	(Ausgabe)
		offset	(Ausgabe)
	Signalausfall	0	(Ausgabe)
	Wertfehler	*	(Ausgabe)
	Rauschen	*	(Ausgabe)
(exemplarisch)	Reduktion des Signalwerts auf 90%	$*0,9$	(Ausgabe)
	Erhöhung des Signalwerts auf 110%	$*1,1$	(Ausgabe)
Hardware	Elektromagnetische Störung	Bayes	(System)
	Elektromagnetisches Stören	–	(Impl.)
	Thermische Störung	Bayes	(System)
	Thermisches Stören	–	(Impl)
	Spannungsversorgung	*	(System)
Bus	Keine Übertragung	0	(Ausgabe)
	Wertfehler	Bayes	(Ausgabe)
	... Bitfehler Vorzeichen	Vz	(Ausgabe)
	... Fehler im Betrag	Betrag	(Ausgabe)
Gateway	Keine Übertragung	0	(Ausgabe)
	Wertfehler	Bayes	(Ausgabe)
Betriebssystem	Bereitstellen der Anwendungen zu spät	delay	(Transition)
	Kein Bereitstellen der Anwendung	0	(System)

Artefakt-Klasse $\mathcal{C} \in \mathbb{C}$	Fehlerlabel $\mathcal{L} \in \mathbb{L}$	Referenzmodifikation $\mathcal{M} \in MOD$	
Applikation	Implementierungsfehler (statisch)	Bayes	(System)
[SW/HW]	... algorithmische Fehler		(Anforderung)
	... Umschaltung zwischen Modi	Modi	(Modi)
	... unterschiedliche Datensemantik	Bayes	(System)
	... Überlauf von Datenstrukturen	Bayes	(Ausgabe)
	... Tasksynchronisation	delay	(Transition)
	... Berechnungs- und Genauigkeitsfehler	$*y \ll *x$	(Ausgabe)
	... uninitialisierter Speicher	Bayes	(System)
	... Hardwaretoleranzen	$-y \ll +x$	(Ausgabe)
	... Störungsszenarien	Bayes	(System)
	... Verletzung von Echtzeitanforderungen	0	(System)
	... unterschiedliche Quantisierung	Betrag	(Ausgabe)
(exemplarisch)	... versetzter Wertebereich	$+x$	(Ausgabe)

4.6 Zusammenfassung

In diesem Kapitel wurde ein Fehlermodell aus [Bre01] vorgestellt, welches Fehler als Modifikationen von Blackboxverhalten definiert, in dem Tupel aus der Verhaltensrelation R_S entfernt werden und andere hinzugefügt werden. Für dieses Fehlermodell wurde allgemein die Kombination von Fehlern im Fall der Unabhängigkeit und Abhängigkeit zwischen Modifikationen betrachtet.

Für die Modifikationen wurden Modellierungstechniken angegeben, mit denen die in Fallstudien aus dem Fahrwerkregelbereich vorgefundenen Fehler spezifiziert werden können. Die in [Bre01] vorgestellte Modellierungstechnik mit stromgebundenen Blackbox-Spezifikationen wurde hier um zustandsgebundene Formeln erweitert. Die Beschreibung der Modifikationen wurde in Anlehnung an ([PCB+55], [Str06], [Str04]) auch um die Möglichkeit erweitert, diese als Abweichungen relativ zum Sollverhalten darzustellen. Die Beschreibung als Abweichungen ist besonders für zu Ausgaben hin aufgelöste Gleichungen geeignet, welche den Verhaltensmodellen in Werkzeugen wie Simulink oder Scade entsprechen. Insbesondere hier wurde die Kombination von Fehlerklassen genauer diskutiert. Die Modifikation endlicher Automaten durch Hinzufügen von Transitionen und Zuständen (siehe [Thu04], [OR04]) zeigt sich in einer beobachtbaren Verhaltensänderung ([SSL+95], [SSL+96]). Diese wurde bei Betriebsmodi-Automaten ohne Rücksicht auf die Zeit in Bezug zu den Abweichungen gebracht, um so die Beschreibungen der Abweichungen präziser fassen zu können.

Für die Fehlverhalten wurden in Summe Modellierungstechniken vorgestellt, welche (1) getrennt von den Sollverhalten definiert werden können, (2) sowohl das Hinzufügen wie auch das Entfernen von Relationstupeln kennen und (3) die in den Fallstudien vorgefundenen informellen Fehlerbeschreibungen (siehe Anhang 4.5) formal darstellen können.

Letztlich wurde für die Implementierungsebene eine einfache Datenstuktur zusammen mit einer Menge von primären Fehlern aufgelistet, welche in den Fallstudien und in [BSN07], [MP01], [Mus99], [MK05], [PPG04] vorgefunden wurden.

Kapitel 5

Zusammenhangsmodelle, -modellierung und -ermittlung

Sichten der Fehlerzusammenhänge

Kapitel 4 befasste sich mit der Modellierung der in einer Funktionssicherheitsanalyse (FMEA bzw. FTA) auftretenden Fehler. Diese sind für Verhaltensmodelle wie auch für Implementierungsmodelle definiert worden und ihre Kombinierbarkeit zu zusammengesetzten Fehlern untersucht worden. Wie in Kapitel 2 beschrieben, werden bei der Durchführung einer Fehleranalyse nicht nur die in einem System auftretenden Fehler festgehalten, sondern auch die Zusammenhänge hinsichtlich Ursache und Wirkung zwischen diesen dokumentiert. In Kapitel 2 wurden diese Zusammenhänge eher informell in einem Fehlernetz festgehalten. Dieses Kapitel beschreibt verschiedene Arten von Fehlerzusammenhängen und definiert diese formal. Ziel dieses Kapitels ist es, mit formalen Modellen eine Basis für die systematische, softwaregestützte Ermittlung von Fehlerabhängigkeiten zu schaffen (siehe Kapitel 6).

Beitrag

Dieses Kapitel erarbeitet allgemeine formale Zusammenhangsmodelle, wie auch spezifische formale Zusammenhangsmodelle, die an die Modellierungstechniken aus Kapitel 4 angepasst sind. Die spezifischen formalen Zusammenhangsmodelle sind auf die Methoden FMEA und FTA zugeschnitten. Zu den Zusammenhangsmodellen werden anhand von Eigenschaften Methoden aus der Literatur zu deren teilautomatisierter Ermittlung mit angegeben. Basis für die allgemeinen Zusammenhangsmodelle sind die Definitionen von [Bre01] und [Str04], in denen ein Folgefehler durch Komposition fehlerhafter Systeme entsteht. Beide Definitionen dienen der Modellierung der exakten Folgefehler. Der erste Beitrag dieses Kapitels besteht deshalb in der Definition eines Zusammenhangmodells für Fehler, in dem auch obere und untere Schranken für die Folgefehler modelliert werden können. Ein weiterer Beitrag ist die Erweiterung dieser Zusammenhangsmodelle für zustandsgebundene Prädikate, inklusive derer Sonderformen 'aufgelöste Gleichungen' und Automaten. Der dritte Beitrag dieses Kapitels ist die Formalisierung von Abstraktionen der Zusammenhangsmodelle. Diese Abstraktionen werden in den informellen Sicherheitsanalysen in der Industrie vorgefunden. Mit den definierten bzw. formalisierten Zusammen-

hangsmodellen ist so in Verbindung mit in der Literatur vorgefundenen Ermittlungsmethoden eine Grundlage zur teilautomatisierten Sicherheitsanalyse gelegt.

Kapitelübersicht

Abschnitt 5.1 befasst sich mit der Modellierung von Fehlerzusammenhängen. Als Grundlage wird die Definition eines Fehlerzusammenhangs von [Bre01] verwendet, welche für Verhaltensrelationen in einem zusammengesetzten System gilt. Des Weiteren wird zur Ermittlung dieser Fehlerzusammenhänge die Fehlerauswirkung analysiert, welche Abweichungen von Kanalwerten aufgrund von Fehlverhalten entspricht. In beiden Fällen kann häufig nicht die exakte Folge angegeben werden. Dazu wird in diesem Abschnitt eine Abbildung definiert, anhand derer die Folgefehler eingegrenzt werden können. Ergänzend wird in Abschnitt 5.3 der Zusammenhang zu Implementierungsmodellen beschrieben, welche zur Induktion von Fehlverhalten genutzt werden.

Abschnitt 5.2 stellt eine Menge von Modellen für Zusammenhänge vor, die spezifisch auf die jeweiligen Modellierungstechniken zugeschnitten sind. Diese Modelle sind weiter spezifisch für die jeweiligen Methoden der Sicherheitsanalyse und die Möglichkeiten der Werkzeugunterstützung angepasst. Abschnitt 5.2.1 definiert Zusammenhänge, welche die Kombinatorik von Verhaltensbedingungen im Sinne von Blackbox-Spezifikationen betreffen. Abschnitt 5.2.2 stellt Zusammenhangsmodelle vor, die sich auf die Ausgabe der Systeme beziehen. Hier werden zusätzlich Abschätzungs- bzw. Abstraktionsmechanismen vorgestellt, welche die Kombinatorik der Zeit vereinfachen. Abschnitt 5.2.3 stellt die Zusammenhänge für die Transitionsmodifikationen vor, die auf endliche Automaten angewendet werden können.

5.1 Zusammenhänge auf Verhaltensebene

Folgefehler

Die Zusammenhänge zwischen Fehlern wurden informell in Definition 2.1.10 als Fehler bezeichnet, welche aufgrund von anderen Fehlern entstehen. Auf der Verhaltensebene werden in dieser Arbeit zwei verschiedene Arten unterschieden. Zum Einen gibt es das *Folgefehlverhalten*. Dieses ist das Fehlverhalten eines zusammengesetzten Systems, bei dem mindestens eine Teilkomponente ein Fehlverhalten aufweist. Diese Art des Zusammenhangs wird in Abschnitt 5.1.1 definiert.

Fehlerauswirkung

Zum Anderen gibt es die *Fehlerauswirkung*. Der Schwerpunkt liegt hier auf der Untersuchung der Abweichungen der Ausgaben von den Soll-Ausgaben in Abhängigkeit der Eingaben. Diese Abweichungen der Ausgaben können entweder durch Fehlverhalten des betrachteten Systems entstehen, oder durch fehlerhafte Eingaben. Im Gegensatz zum Folgefehlverhalten wird hier der Signalfluss verfolgt. In einer Produkt-Funktionssicherheitsanalyse werden die Folgefehlverhalten modelliert. Um diese aus mehreren Teilen zusammenzusetzen (siehe Abschnitt 2.5), wird die Fehlerauswirkung als Hilfsinformation genutzt. Diese wird in Abschnitt 5.1.2 definiert.

5.1.1 Folgefehler auf Verhaltensebene

Komposition

Die Modifikation einer Teilkomponente hat Folgen auf das zusammengesetzte System. Hat eine Teilkomponente ein geändertes Verhalten, so hat auch das Gesamtsystem ein geändertes Verhalten. Ein Fehlverhalten, welches eine Modifikation des Soll-Verhaltens ist, kann sowohl für die Gesamtkomponente S wie auch für die Teilkomponente S_1 definiert werden, wenn diese wie in Abbildung 5.1 dargestellt zusammenhängen, also gilt:

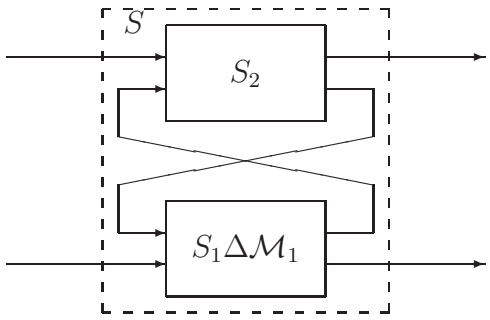


Abb. 5.1: Fehlerfortpflanzung in einem zusammengesetzten System [Bre01]

$$S = S_1 \otimes S_2$$

Allgemeingültigkeit

Aufgrund der Kompositionseigenschaften und der Zerlegungseigenschaften des Systemmodells ist die Darstellung der Definition mit nur zwei Komponenten keine Einschränkung der Anwendbarkeit der Regel, da beliebige Komponenten zusammengefasst werden können. So kann S_1 eine spezielle zu untersuchende Komponente sein, wohingegen S_2 den Rest des Systems darstellt. In Anlehnung an [Bre01] ist eine Fehlverhaltensabhängigkeit in einem System wie folgend definiert:

Definition 5.1.1 (Fehlverhaltensabhängigkeit)
Eine *Fehlverhaltensabhängigkeit* in einem System S mit $S = S_1 \otimes S_2$ mit dem Fehlverhalten R_{S^F} und dem Fehlverhalten $R_{S_1^F}$ bzw. den Fehlverhalten $R_{S_1^F}$ und $R_{S_2^F}$, ist gegeben, wenn gilt:

$$R_{S^F} = R_{S_1^F} \otimes R_{S_2} \text{ bzw. } R_{S^F} = R_{S_1^F} \otimes R_{S_2^F}$$

Werden die Fehlverhalten mit Modifikationen beschrieben (siehe Definition 4.1.2), so ist eine Modifikation $\mathcal{M}_S$ ein Folgefehlverhalten, wenn gilt:

$$R_S \Delta \mathcal{M}_S = R_{S_1} \Delta \mathcal{M}_{S_1} \otimes R_{S_2} \text{ bzw. } R_S \Delta \mathcal{M}_S = R_{S_1} \Delta \mathcal{M}_{S_1} \otimes R_{S_2} \Delta \mathcal{M}_{S_2}$$

Analog gilt für Blackbox-Spezifikationen:

$$[\![S]\!] \Delta [\![\mathcal{M}_S]\!] = [\![S_1]\!] \Delta [\![\mathcal{M}_{S_1}]\!] \wedge [\![S_2]\!]$$ ⌟

Damit ist definiert, wie sich die Wirkung einer Fehlerkombination in einem zusammengesetzten System zeigt. Es müssen lediglich die entsprechenden Fehlverhalten in die Spezifikation eingesetzt werden. Dieses Verfahren über Komposition der Komponenten mit Fehlverhalten ist in weiteren Ansätzen, wie [SSL+96] zu finden.

Angabe der Wirkung

Die obige Definition bestimmt genau den Folgefehler, der sich aus den gegebenen Teilfehlern ergibt. Möchte man die Wirkungen aller Fehlerkombinationen festhalten, so kann es sein, dass sich auf der Ebene des zusammengesetzten Systems eine große Menge an sehr spezifischen Folgefehlern ergibt (schlimmstenfalls $2^{\#\,Teilfehler}$). Diese spezifischen Folgefehler können des Weiteren sehr kompliziert zu beschreiben sein.

Diese Menge an Fehlern ist zwar mit Werkzeugen gut, manuell allerdings nicht geeignet bearbeitbar. Um die zu bearbeitenden Folgefehler auf eine handhabbare Menge zu reduzieren, werden in dieser Arbeit zwei Möglichkeiten in Betracht gezogen:

(1) Zusammenfassen der konkreten Fehlverhalten zu abstrakten Fehlverhalten

(2) Zuweisen der konkreten Fehlverhalten zu bereits definierten Fehlern

Zusammenfassen

Im Fall (1) des Zusammenfassens der Fehler ist die Aufgabe, möglichst ähnliche Fehlverhalten zu finden, und diese einem übergreifenden Fehler zuzuordnen. Für das übergreifende Fehlverhalten muss gelten, dass es unabhängig von allen Teilfehlern ist und alle $E_{Teilfehler}$ bzw. $F_{Teilfehler}$ der Teilfehler eine Verfeinerung von $E_{uebergreifend}$ bzw. $F_{uebergreifend}$ sind, also für jeden Teilfehler gilt:

Definition 5.1.2 (Teilmodifikation)
Eine Modifikation $\mathcal{M}_{sub}$ ist eine echte *Teilmodifikation* einer anderen Modifikation $\mathcal{M}_{super}$, geschrieben als

$\rightsquigarrow: MOD_S \times MOD_S \rightarrow BOOL$

$$\mathcal{M}_S^{super} \rightsquigarrow \mathcal{M}_S^{sub}$$

wenn gilt:

$$(\mathcal{M}_S^{sub} \overset{dep}{\#} \mathcal{M}_S^{super}) \wedge (E_S^{super} \rightsquigarrow E_S^{sub}) \wedge (F_S^{super} \rightsquigarrow F_S^{sub}) \qquad \lrcorner$$

Auf diese Weise lassen sich vom Menschen interpretierbare Fehler definieren, die verständlich und bewertbar sind. Möchte man den übergreifenden Fehler so klein wie möglich gestalten, so ergibt sich das Fehlverhalten wie folgt, in dem die unabhängigen beteiligten Teilfehler zusammengefasst werden:

$$\mathcal{M}_S^{super(max)} = \mathcal{M}_S^{sub\_1} + \mathcal{M}_S^{sub\_2} + ...$$

Abstraktion (obere Schranke)

Mit dieser Abstraktion werden nun in der Ursache-Wirkungs-Relation entsprechend nicht die exakten Fehler, sondern die übergreifenden Fehler als Folge genannt. Es wird statt einer präzisen Auswirkung einer Fehlerkombination ein Folgefehler angegeben, der eventuell mehr falsche Ausführungspfade zulässt und eventuell mehr korrekte Pfade entfernt hat. Es wird also eine obere Schranke für das Folgefehlverhalten angegeben, bei der auch Folgen enthalten sind, die in diesem konkreten Fehler nicht möglich sind. Handelt es sich hierbei um kritische Wirkungen, so ist es unter Umständen sinnvoll, die umfassenden Fehler zu verfeinern, bzw. zu teilen bevor die Soll-Spezifikation angepasst wird.

untere Schranke

Die Definition der Teilmodifikation kann auch umgekehrt genutzt werden, um eine minimale Auswirkung anzugeben. Die Schnittmenge[1] aller beteiligten Modifikationen liefert eine Modifikation, in der sicher ist, dass korrekte Pfade entfernt wurden und sicher fehlerhafte Pfade hinzugekommen sind.

$$\mathcal{M}_S^{super(min)} = \mathcal{M}_S^{sub\_1} \cap \mathcal{M}_S^{sub\_2} \cap ...$$

[1] Die Schnittmenge zweier Modifikationen $\mathcal{M}_1$ und $\mathcal{M}_2$ ist: $(E_1 \cap E_2, F_1 \cap F_2)$

Abschätzung

Mit dieser oberen und unteren Schranke ist so ein Instrumentarium zur Abschätzung von Folgefehlern gegeben. Diese können zum Beispiel genutzt werden, um mit der unteren Grenze sicherzustellen, dass eine Wirkung beobachtbar ist, um eventuell Maßnahmen treffen zu können (untere Schranke) oder dass eine Wirkung ausgeschlossen werden kann (obere Schranke).

Zuordnen

Das Zusammenfassen von Fehlern in übergreifenden Fehlern ermöglicht das Erreichen einer kleineren Menge an zu bearbeitenden Folgefehlern. Es ermöglicht aber nicht die Abbildung auf ein anderes Fehlerbild, welches aus dem Betrachtungswinkel auf der Ebene des zusammengesetzten Systems gewünscht wird. Ziel ist also auch Fall (2), konkrete Fehlverhalten definierten Fehlern zuzuweisen. Betrachtet man eine Modifikation $\mathcal{M}_S^{super}$, so kann diese als eine kombinierte Modifikation betrachtet, und in vollständige Teilmodifikationen $\mathcal{M}_S^{sub\_a}$, $\mathcal{M}_S^{sub\_b}$,... aufgespalten werden, die voneinander unabhängig sind. Jedes dieser Teilfehlverhalten kann nun wiederum einem bereits definierten Fehler zugewiesen werden, von dem es ein Teilfehler ist. Sind alle Teilfehler des ursprünglichen Fehlers den neu definierten Fehlern zugewiesen worden, so ist die komplette Wirkung des Fehlers berücksichtigt.

Abstraktion

Auch bei dieser abstrakten Zuordnung werden bei einer Ursache-Wirkungs-Angabe nicht die exakten Folgefehler genannt. Es werden mehr korrekte Pfade entfernt und mehr inkorrekte Pfade hinzugefügt. Die Zuordnung dient also zur Ermittlung einer oberen Schranke. Während das Zusammenfassen in übergreifenden Fehlern maximal zu Fehlern führen kann, in denen $E \cap F = \emptyset$, kann bei der Zuweisung eine Kombination erreicht werden, die eine nicht-leere Schnittmenge enthält. Bei der Nutzung dieses Abstraktionsmechanismus ist darauf zu achten, dass eine entsprechende Senkung der Abstraktion große Wirkung haben kann.

Modellierung des Zusammenhangs

Ein wesentliches Ergebnis der Fehleranalysen ist die explizite Modellierung der Fehlerzusammenhänge (siehe Abschnitte 2.3 und 2.4). Ein Modell der Fehlerzusammenhänge hat unter anderem 3 Funktionen:

(1) Die Zusammenhänge zwischen Fehlern sind sofort ablesbar.

(2) Bei fehlender automatischer Ermittlung der Abhängigkeiten können diese manuell nachmodelliert werden.

(3) Angaben über die Abhängigkeiten der Fehlverhalten können (im Sinne einer Spezifikation) gemacht werden, bevor die Komponente vollständig bzw. endgültig implementiert oder spezifiziert ist.

Modifikationsabhängigkeitsmodell

Zur Modellierung der Modifikationsabhängigkeiten werden Tupel verwendet, welche die Beziehungen zweier Modifikationen beschreiben. Die Tupel enthalten die Modifikationen der Teilsysteme, die Modifikation des Gesamtsystems und eine Angabe, wie diese zusammenhängen. Da die Abhängigkeiten der Modifikationen von dem Sollverhalten des Systems mitbestimmt werden, findet die Modellierung der Modifikationsabhängigkeiten immer im Kontext der Spezifikation eines Systems S statt. Die Angabe des Zusammenhangs ist wie folgt definiert:

Definition 5.1.3 (Modifikationsabhängigkeitsmodell)
Gegeben sei ein System $S = S_1 \otimes S_2$ mit $S, S_1, S_2 \in \mathbb{S}$ und die minimalen Modifikationen $\mathcal{M}_S \in MOD_S$, $\mathcal{M}_{S_1} \in MOD_{S_1}$ und $\mathcal{M}_{S_2} \in MOD_{S_2}$.

Eine Fehlverhaltensabhängigkeit wird modelliert mit einem Tupel γ_S, welches wie folgt definiert ist:

$$\gamma_S \in MOD_{S_1} \times MOD_{S_2} \times \{\dashrightarrow, \overset{\leq}{\dashrightarrow}, \overset{\geq}{\dashrightarrow}, \overset{\otimes<}{\dashrightarrow}, \overset{\emptyset}{\dashrightarrow}, \overset{?}{\dashrightarrow}\} \times MOD_S$$

Die Abhängigkeiten werden hierbei mit dem Abhängigkeitsoperator folgendermaßen definiert:

$$\{\dashrightarrow, \overset{\leq}{\dashrightarrow}, \overset{\geq}{\dashrightarrow}, \overset{\otimes<}{\dashrightarrow}, \overset{\emptyset}{\dashrightarrow}, \overset{?}{\dashrightarrow}\} : MOD_{S_1} \times MOD_{S_2} \times MOD_S \rightarrow BOOL$$

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \dashrightarrow \mathcal{M}_S \overset{def}{\Leftrightarrow} R_S\Delta\mathcal{M}_S = R_{S_1}\Delta\mathcal{M}_{S_1} \otimes R_{S_2}\Delta\mathcal{M}_{S_2}$$

Die Modifikationen $\mathcal{M}_{S_1}$ und $\mathcal{M}_{S_2}$ haben verhaltenstreu die Modifikation $\mathcal{M}_S$ zur Folge.

$$\begin{aligned}&(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\leq}{\dashrightarrow} \mathcal{M}_S\\ &\overset{def}{\Leftrightarrow} \exists \mathcal{M}_a \in MOD_S : ((R_{S_1}\Delta\mathcal{M}_{S_1} \otimes R_{S_2}\Delta\mathcal{M}_{S_2} = R_S\Delta\mathcal{M}_a) \wedge (\mathcal{M}_S \rightsquigarrow \mathcal{M}_a))\end{aligned}$$

Die Modifikationen $\mathcal{M}_{S_1}$ und $\mathcal{M}_{S_2}$ haben eine Teilmodifikation der Modifikation $\mathcal{M}_S$ zur Folge.

$$\begin{aligned}&(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\geq}{\dashrightarrow} \mathcal{M}_S\\ &\overset{def}{\Leftrightarrow} \exists \mathcal{M}_a \in MOD_S : ((R_{S_1}\Delta\mathcal{M}_{S_1} \otimes R_{S_2}\Delta\mathcal{M}_{S_2} = R_S\Delta\mathcal{M}_a) \wedge (\mathcal{M}_a \rightsquigarrow \mathcal{M}_S))\end{aligned}$$

Die Modifikationen $\mathcal{M}_{S_1}$ und $\mathcal{M}_{S_2}$ haben eine Modifikation zur Folge, für die die Modifikation $\mathcal{M}_S$ eine Teilmodifikation ist.

$$\begin{aligned}&(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\otimes<}{\dashrightarrow} \mathcal{M}_S\\ &\overset{def}{\Leftrightarrow} \exists \mathcal{M}_a, \mathcal{M}_b \in MOD_S^{minimal} :\\ &((R_{S_1}\Delta\mathcal{M}_{S_1} \otimes R_{S_2}\Delta\mathcal{M}_{S_2} = R_S\Delta\mathcal{M}_a)\\ &\quad \wedge(\mathcal{M}_a \rightsquigarrow \mathcal{M}_b \wedge \mathcal{M}_S \rightsquigarrow \mathcal{M}_b \wedge (E_b \neq \emptyset \vee F_b \neq \emptyset)))\end{aligned}$$

Die Modifikationen $\mathcal{M}_{S_1}$ und $\mathcal{M}_{S_2}$ haben eine Modifikation zur Folge, die sich mit der Modifikation $\mathcal{M}_S$ überschneidet.

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\emptyset}{\dashrightarrow} \mathcal{M}_S \overset{def}{\Leftrightarrow} \neg((\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\otimes<}{\dashrightarrow} \mathcal{M}_S)$$

Die Folge der Modifikationen $\mathcal{M}_{S_1}$ und $\mathcal{M}_{S_2}$ hat mit der Modifikation $\mathcal{M}_S$ keine gemeinsamen Pfade.

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{?}{\dashrightarrow} \mathcal{M}_S \overset{def}{\Leftrightarrow} true$$

Keine Aussage. *Hilfsmarkierung für Bearbeitung.*

Ein Fehlverhaltenabhängigkeitsmodell ist eine Menge von Anhängigkeiten:

$$\Gamma_S \subseteq MOD_{S_1} \times MOD_{S_2} \times \{\dashrightarrow, \overset{\leq}{\dashrightarrow}, \overset{\geq}{\dashrightarrow}, \overset{\otimes<}{\dashrightarrow}, \overset{\emptyset}{\dashrightarrow}, \overset{?}{\dashrightarrow}\} \times MOD_S$$ ⌟

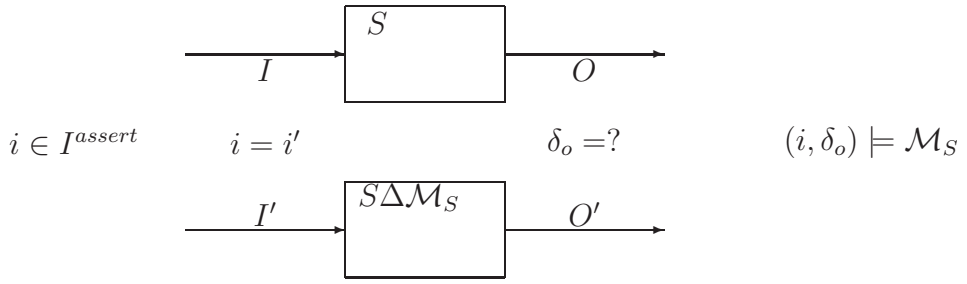


Abb. 5.2: Beobachtbares Fehlverhalten

Aus der Information eines Modifikationsabhängigkeitsmodells kann ein Fehlernetz für Fehleranalysen wie FMEA und FTA aufgebaut werden, da für die Fehler der Folgefehler der umgebenden Komponente gegeben ist. Allerdings ist zur Ermittlung einer Fehlverhaltensabhängigkeit auf oberster Ebene gerade bei Systemen mit Rückkopplung das gesamte Systemverhalten zu betrachten, welches häufig zu groß für automatische Analysen ist, und zu komplex, um vom Entwickler sicher korrekt beurteilt werden zu können. Ein Ansatz, der diesem Problem entgegen kommt, ist die Verknüpfung der Auswirkungen, welche im folgenden Absatz beschrieben werden. *Fehlernetz*

5.1.2 Fehlerauswirkung auf Verhaltensebene

In der Praxis sind Beschreibungen von Fehlern häufig beobachtbare Abweichungen der Ausgabewerte von den Sollwerten (z.B.: Wert zu groß, Nachricht kommt zu spät, ...). Diese Betrachtung stammt historisch aus der Betrachtung von einfachen Systemen, welche nicht redundant waren, und in denen auch die Rückkopplung keine wesentliche Rolle spielte. Für Systeme, in denen die Kombinatorik der Signale und die Rückkopplung wesentliche Bestandteile sind, ist diese Betrachtungsweise nicht mehr ausreichend. Dieser Abschnitt setzt die Betrachtung der Abweichungen in Bezug zu der Modellierung der Fehler und Folgefehler des vorhergehenden Abschnitts. Dazu wird der Begriff der Abweichung und der Fehlerauswirkung von Eingaben zu Ausgaben von Komponenten definiert. Auf Basis dessen wird die Auswirkung von Abweichungen modelliert. Schließlich wird die Kombinatorik innerhalb dieser Auswirkungsmodelle betrachtet, also die korrekte Abbildung der Auswirkungsmodelle auf die Modelle zum Folgefehlverhalten, die Probleme damit, und letztlich die daraus resultierenden Abstraktionen, die in der Praxis vorzufinden sind. *Abweichung statt Fehlverhalten*

Mit der in Abbildung 5.1 dargestellten allgemeinen Komposition von Komponenten lässt sich die Wirkung des Fehlverhaltens einer Teilkomponente S_1 nach außen hin genau bestimmen. Das Fehlverhalten ist dadurch bestimmt, dass sich die Menge der Tupel aus R_S des zusammengesetzten Systems geändert hat. Diese Änderung lässt sich beobachten, in dem für eine exemplarische Eingabe im fehlerhaften System andere Ausgaben möglich sind, als im Soll-System (siehe Abbildung 5.2). Die Abweichung der Ausgaben δ_o entspricht hierbei der Verhaltensänderung der Modifikation. *beobachtbares Fehlverhalten*

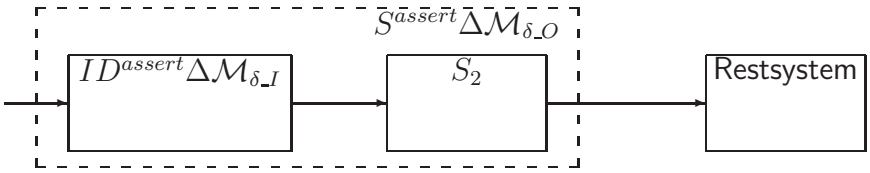


Abb. 5.3: Fehlerfortpflanzung durch ein System

Betrachtet man die Abweichungen der Ausgaben individuell für jede Eingabe i, so kann damit eine Modifikation $\mathcal{M}_S$ vollständig beschrieben werden. Eingaben, welche zu gleichen Abweichungen der Ausgaben führen, können mittels Laufzeitbedingungen zu Mengen I^{assert} zusammengefasst werden. Ähnlich, wie bei den Modifikationen, können auch die Abweichungen gruppiert werden, um die Menge der Beschreibungen zu reduzieren.

Definition Abweichung

Eine Abweichung ist mathematisch als eine Relation $\mathcal{P}(M^*) \rightarrow \mathcal{P}(M^*)$ definiert, welche Mengen von Strömen durch Mengen von Strömen ersetzt. Die Mengen können auch einelementig sein. Die Beschreibungen der Ersetzungen können wie die Beschreibungen der Ausgabemodifikationen gestaltet sein (siehe Abschnitt 4.2.2), wobei die Sollwerte den Werten τ_o bzw. $\tau_o^{\{\}}$ entsprechen. Wie bei der Ausgabemodifikation können auch bei der Beschreibung der Abweichungen mehrere Kanäle zusammengefasst werden, um eine Ausgabe zu beschreiben. Da Abweichungsbeschreibungen unabhängig von dem Verhalten der Komponenten sind, können sie auch zur Beschreibung von Abweichungen der Eingaben in ein System genutzt werden.

Wirkung in korrekter Teilkomponente

Mit der in Abbildung 5.1 dargestellten allgemeinen Komposition von Komponenten lässt sich die Wirkung des Fehlverhaltens einer Teilkomponente S_1 nach außen hin genau bestimmen. Eine intuitive Möglichkeit, auf diese Wirkung zu schließen, ist zu verfolgen, wie sich die Abweichungen der Ausgaben des Systems S_1 auf die Ausgaben des Systems S_2 auswirken. Die Fragestellung ist also nicht mehr, wie sich das Fehlverhalten eines Produktes auswirkt, sondern wie fehlerhafte Eingaben auf die Ausgaben des Systems wirken. Abbildung 5.3 zeigt diesen Zusammenhang. Analog zur Darstellung 5.2 kann eine Wirkung für bestimmte Eingaben definiert werden. Dies wird über die Laufzeitbedingung ^{assert} bestimmt. Die Angabe der Abweichung an den Eingabekanälen der Komponente entspricht einer vorgeschalteten Komponente, welche die Identität als Verhalten hat, und mit einer entsprechenden Ausgabemodifikation abgeändert wird. Die resultierende Abweichung der Ausgaben des Systems S_2 ist dann die Ausgabemodifikation des Gesamtsystems. Zur Ermittlung der Zusammenhänge können bei diesem Aufbau in Abbildung 5.3 die gleichen Mechanismen wie für die Beschreibung der Modifikationsabhängigkeiten verwendet werden. Entsprechend ergibt sich folgende Definition für ein Abweichungsabhängigkeitsmodell:

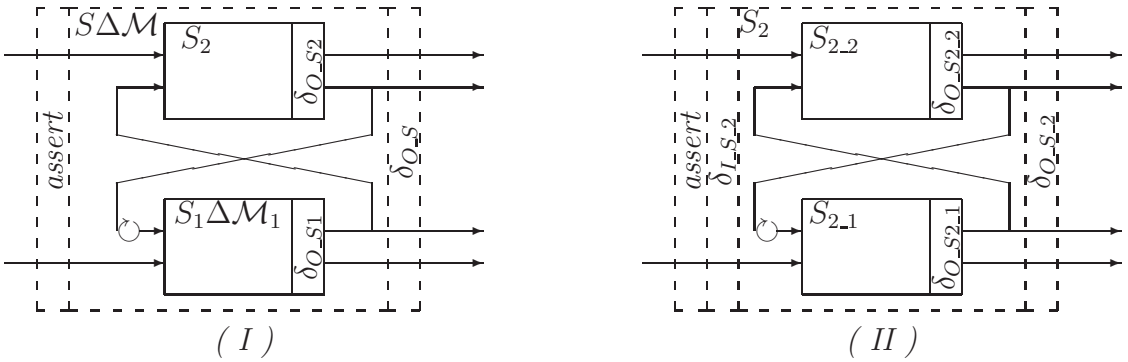


Abb. 5.4: Fehlerausbreitung anhand Abweichung der Ausgaben

Definition 5.1.4 (Abweichungsabhängigkeitsmodell)
Gegeben sei ein System S mit der Schnittstelle (I_S, O_S) und die Abweichungen δ_{O_S} und δ_{I_S}. Eine Abweichungsabhängigkeit wird modelliert mit einem Tupel ψ_S, welches wie folgt definiert ist:

$$\psi \in \mathbb{S}^{assert} \times DELTA_{I_S} \times \{\dashrightarrow, \overset{\leq}{\dashrightarrow}, \overset{\geq}{\dashrightarrow}, \overset{\otimes<}{\dashrightarrow}, \overset{\emptyset}{\dashrightarrow}, \overset{?}{\dashrightarrow}\} \rightarrow DELTA_{O_S}$$

Die Beziehungen zwischen den Modifikationen seien definiert, wie in Definition 5.1.3.

Ein *Abweichungsabhängigkeitsmodell* ist eine Menge von Abhängigkeiten:

$$\Psi \subseteq S^{assert} \times DELTA_{I_S} \times \{\dashrightarrow, \overset{\leq}{\dashrightarrow}, \overset{\geq}{\dashrightarrow}, \overset{\otimes<}{\dashrightarrow}, \overset{\emptyset}{\dashrightarrow}, \overset{?}{\dashrightarrow}\} \times DELTA_{O_S}$$ ⌟

Formeln

Beschreibt man ein Fehlerauswirkungsmodell genau, so kann die Menge der in dem Modell enthaltenen Tupel sehr groß sein. Analog zur Beschreibung des Verhaltens eines Systems, in dem die Beziehungen zwischen den Belegungen der Eingabe- und Ausgabekanäle mit Formeln beschrieben werden, kann dies genauso für die Beziehungen zwischen den Abweichungen geschehen. Für die Belegung der Abweichungen können dann entweder Ströme oder Werte verwendet werden (siehe Abschnitt 3.3.1).

Abhängigkeit

Mit Hilfe dieser Abweichungsabhängigkeitsmodelle wird eine Fehlverhaltensabhängigkeit abgeschätzt bzw. im Idealfall abgeleitet. Abbildung 5.4(I) stellt dieses Szenario zur Ermittlung einer Fehlverhaltensabhängigkeit dar. Die Modifikation $\mathcal{M}$ wird bestimmt, in dem jeweils für mit R_S^{assert} zusammengefasste Eingaben die entsprechenden Abweichungen der Ausgaben betrachtet werden (siehe Abbildung 5.2). Durch das Fehlverhalten des Systems S_1 entsteht ein δ an dessen Ausgabe. Dieses führt unmittelbar zu einem δ an der Ausgabe des Gesamtsystems S aber auch zu einem δ an der Eingabe des Systems S_2 und damit an dessen Ausgabe. Ist das System rekursiv, so kann die veränderte Ausgabe des Systems S_2 wiederum zu einer Änderung der Ausgabe des Systems S_1 führen. Der bei dem Rückkopplungskanal angegebene Kreis deutet an, dass pro Rückkopplungsschleife mindestens ein Zeitschritt vergehen muss. Dies ermöglicht eine schrittweise Erschließung des Folgefehlers in dem pro Rekursionsschritt in der Zeit vorangeschritten wird:

Definition 5.1.5 (inkrementelle Fehlererschließung)
Ein Folgefehler für endliche Zeitrahmen $[0, .., n]$ lässt sich über folgende Induktion ermitteln:

Im Initialzustand sind keine Abweichungen der Ausgaben vorhanden und die Ausgabeströme sind leer.

$$\begin{aligned} \text{Induktionsanker } t < 0: \qquad & \delta^{(t)}_{O_{S\_2}} = \delta^{\emptyset} \\ & \mathsf{reverse}^{(t)}_{O_{S\_1}}(x) = \langle\rangle \\ & \mathsf{reverse}^{(t)}_{O_{S\_2}}(x) = \langle\rangle \end{aligned}$$

In jedem Induktionsschritt enthält die Folgeabweichung einer Ausgabe die Ströme, die sich aus dem Fehlerauswirkungsmodell ergeben und nicht die kausale Monotonie (siehe Definition 3.2.6) verletzen.

$$\begin{aligned} &\text{Induktionsschritt } 0 \leq t \leq n: \\ &\quad \delta^{(t)}_{O_{S\_1}} = \{x | x \in \psi^{\dashrightarrow}_{S_1}(S^{assert}, \delta^{(t)}_{I_S} || \delta^{(t-1)}_{O_{S\_2}}) \wedge \mathsf{reverse}^{(t-1)}_{O_{S\_1}}(x) \sqsubset x\} \\ &\quad \delta^{(t)}_{O_{S\_2}} = \{x | x \in \psi^{\dashrightarrow}_{S_2}(S^{assert}, \delta^{(t)}_{I_S} || \delta^{(t)}_{O_{S\_1}}) \wedge \mathsf{reverse}^{(t-1)}_{O_{S\_2}}(x) \sqsubset x\} \\ &\quad \delta^{(t)}_{O_S} = \delta^{(t)}_{O_{S\_1}} || \delta^{(t)}_{O_{S\_2}} \end{aligned}$$ ⌟

Erläuterung zu Def. 5.1.5

An dieser Stelle wird auf drei Besonderheiten der obigen Definition eingegangen: (1) die Aufteilung der Systeme, (2) die fortschreitende Zeit in Schleifen und (3) die kausale Monotonie. Die Definition bezieht sich in Punkt (1) auf eine Aufteilung der Systeme in zwei Teilsysteme. Aufgrund der assoziativen Eigenschaft der Komposition können auch Systeme, welche aus mehreren Teilsystemen bestehen, auf diese Zwei-System-Betrachtung reduziert werden. Abbildung 5.4 (II) zeigt, dass das Fehlerauswirkungsmodell des Systems S_2 wieder aus entsprechenden Teilmodellen aufgebaut sein kann. Das Induktionsverfahren ist aber auch hier dasselbe wie für Abbildung 5.4 (I).

Rückkopplung

Die Induktion in Definition 5.1.5 läuft über die Zeit. Entsprechend muss nach Punkt (2) pro Induktionsschritt im Falle einer Rekursion die Zeit voranschreiten. Mit anderen Worten muss sich mindestens eine Zuweisung der Abweichungsabhängigkeitsmodelle auf einen vergangenen Wert beziehen. Natürlich muss der Verweis auf vergangene Zeitpunkte nicht, wie in Abbildung 5.4 um genau einen Zeitschritt genau vor einer Komponente geschehen, sondern kann auch weiter zurück und an mehreren Stellen geschehen. Beobachtungen der Praxis haben gezeigt, dass die Rückkopplung jedoch meist explizit als Schleife mit einem Verzögerungs-Modul gestaltet ist, in der das Modul die Nachrichten um einen Zeitschritt verzögert[2]. Modelliert wird der Verbrauch der Zeit wie bei dem delay-Operator (siehe Definition 3.3.4), bei dem dem verzögerten Strom Initialwerte vorangestellt werden.

Fehlerereignis

Die mit der Rückkopplung voranschreitende Zeit ist ebenso eine Schnittstelle zu einem punktuellem Fehlerbild, in dem Fehler nicht im Sinne einer Modifikation in-

[2]Es gibt auch eine Menge von Rückkopplungen, welche komplex gestaltet sind. Diese sind aber meist innerhalb von Modulen, welche zu klein sind, um aus mehreren Abweichungsabhängigkeitsmodellen zusammengesetzt zu sein.

terpretiert werden, sondern als Fehlerereignisse oder Fehlerzustände zu einem bestimmten Zeitpunkt (siehe Def. 2.1.2 und 2.1.4). Ein Fehlerereignis zu einem Zeitpunkt t kann an der gleichen Schnittstelle ein Fehlerereignis zum Zeitpunkt $t+1$ verursachen. Bei der Modellierung als Modifikation ist dieser Zusammenhang über die Zeit implizit mit modelliert, da diese zeitlos einem Systemverhalten zugeordnet werden.

kausale Monotonie

Die Rückkopplung führt dazu, dass fehlerhafte Ausgaben wieder als Eingaben in ein System kommen. In einer Rückkopplungsschleife muss dabei mindestens ein Zeittakt vergehen. Mit fortschreitender Zeit darf sich nach Punkt (3) an den bereits vergangenen Abweichungen nichts mehr ändern (siehe Definition 3.2.6). Die vorangegangenen Ausgaben welche durch die Abweichungen entstanden sind, müssen also Präfixe der in der aktuellen Iteration entstehenden Ausgaben sein. Dieser Umstand wird mit einer Funktion reverse modelliert, welche zu einer Abweichung angibt, welches Präfix in der Iteration zuvor gegolten hat. Mit dieser Funktion kann die kausale Monotonie sichergestellt werden, welche später als Grundlage für den Umgang mit unendlichen Strömen genutzt wird. Die Anwendung der Iteration und insbesondere die Notwendigkeit der Einhaltung der kausalen Monotonie wird in folgendem Beispiel nochmals verdeutlicht:

Beispiel 5.1.1 (inkrementelle Erschließung des Folgefehlers)
Gegeben seien zwei Systeme S_1 und S_2 mit

$$I_{S_1} = \{a, d\} \qquad I_{S_2} = \{c\} \qquad O_{S_1} = \{c\} \qquad O_{S_2} = \{d\}$$

Das Verhalten der Systeme sei ein stark vereinfachtes Integral, welches aus einer Summe und einer Rückkopplung besteht:

$$[\![S_1]\!]^{op} = (c = d + a) \qquad \text{und} \qquad [\![S_2]\!]^{op} = (d = \mathsf{delay}_{\langle 0 \rangle}(c))$$

Gegeben sei des Weiteren eine Modifikation, bei der sporadisch die Summe um den Wert 1 zu hoch sein kann, also zu viel aufsummiert wird:

$$[\![\mathcal{M}^1_{S_1}]\!]^{op} = (true, (c = d + a + 1))$$

Anmerkung: Im Folgenden wird zur Vereinfachung der Kanal a nicht weiter beachtet. Da von diesem kein Fehler ausgeht, hat dies keine Wirkung auf die Aussage des Beispiels.

Gegeben seien folgende Abweichungen, welche sich aus dem Fehlverhalten des Systems S_1 ergeben:

$[\![\delta^{\emptyset}_{I_{S\_1}}]\!] = (true, false)$ – *kein Fehler*

$[\![\delta^{+1}_{O_{S\_1}}]\!]^{op} = (true, c = \tau_c + 1)$ – *Wert sporadisch 1 zu hoch*

$[\![\delta^{0 \mapsto +1}_{I_{S\_1}}]\!] = (true, (d.0 = \tau_d.0) \wedge \forall t \geq 1 : (0 \leq d.t - \tau_d.t \leq 1))$
– *Wert sporadisch 1 zu hoch ab 2. Zeittakt*

$[\![\delta^{1\mapsto+2}_{O_{S\_1}}]\!] = (true, (0 \leq d.0 - \tau_d.0 \leq 1) \wedge \forall t \geq 1 : (0 \leq d.t - \tau_d.t \leq 2))$
– *Wert im 1. Zeittakt evtl. 1 zu hoch und sporadisch 1 oder 2 zu hoch ab 2. Zeittakt*

Für die Systeme gelten folgende Auswirkungsmodelle welche mit den eben genannten Fehlern gebildet werden können. Die Gruppierung der Eingaben mit Laufzeitbedingungen wird hierbei nicht berücksichtigt und entsprechend immer mit *true* angegeben.

$$\Psi_{S_1 \Delta \mathcal{M}^1_{S_1}} = \{(\delta^{\emptyset}_{I_{S\_1}}, \dashrightarrow, \delta^{+1}_{O_{S\_1}}), (\delta^{0\mapsto+1}_{I_{S\_1}}, \dashrightarrow, \delta^{1\mapsto+2}_{O_{S\_1}})\} \quad \Big| \quad \Psi_{S_2} = \{(\delta^{+1}_{O_{S\_1}}, \dashrightarrow, \delta^{0\mapsto+1}_{I_{S\_1}})\}$$

Auf Basis dieser Abweichungsmodelle wird nun das Gesamtfehlverhalten für Ströme bis zur Länge 2 aufgebaut. Initial ist keine Abweichung (dargestellt als ein Strom, der die absolute Abweichung angibt) an der Eingabe zu S_1 vorhanden, da dieses Signal aus der Rückkopplung stammt (Induktionsverankerung):

$$\delta^{\emptyset}_{I_{S\_1}} \Rightarrow (\delta^{(0)}_{I_{S\_1}} = \{\langle 0 \rangle\})$$

Verfolgt man diese Fehler über die Auswirkungsmodelle $\psi^{\dashrightarrow}$, so ergeben sich folgende Fehler:

$$\delta^{+1}_{O_{S\_1}} \Rightarrow (\delta^{(0)}_{O_{S\_1}} = \{\langle 0 \rangle, \langle 1 \rangle\}) \qquad \text{und} \qquad \delta^{0\mapsto+1}_{I_{S\_1}} \Rightarrow (\delta^{(1)}_{I_{S\_1}} = \{\langle 0,0 \rangle, \langle 0,1 \rangle\})$$

Der Fehler des rückgekoppelten Kanals ergibt nun laut Fehlerauswirkungsmodell folgenden Fehler, welcher hier aus den zwei möglichen Abweichungsströmen und $\mathcal{M}_{S_1}$ hergeleitet wird.

$$\delta^{1\mapsto+2}_{O_{S\_1}} \Rightarrow \left(\delta^{(1)}_{I_{O\_1}} = \begin{cases} \{\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,0 \rangle, \langle 1,1 \rangle\} & \text{falls } d = \langle 0,0 \rangle \\ \{\langle 0,1 \rangle, \langle 0,2 \rangle, \langle 1,1 \rangle, \langle 1,2 \rangle\} & \text{falls } d = \langle 0,1 \rangle \end{cases} \right)$$

An dieser Stelle wird deutlich, dass die Kombination der Abweichungsabhängigkeitsmodelle die Abhängigkeit durch die Rückkopplung nicht berücksichtigt. Ein rückgekoppeltes δ kann nicht bereits ausgegebene Werte verändern. Der Strom $\langle 0,2 \rangle$ kann nicht korrekt sein, da er nur durch eine vorangegangene Ausgabe $\langle 1 \rangle$ entstehen kann. Der verursachende Strom (ausgedrückt mit der Funktion reverse) muss also ein Präfix sein:

$$\mathsf{reverse}^0_{O_{S\_1}}(\langle 0,0 \rangle) = \langle 0 \rangle \qquad \text{und} \qquad \mathsf{reverse}^0_{O_{S\_1}}(\langle 0,1 \rangle) = \langle 1 \rangle$$

Damit reduziert sich die Folgeabweichung auf:

$$\delta^{(1)}_{O_{S\_1}} = \begin{cases} \{\langle 0,0 \rangle, \langle 0,1 \rangle\} & \text{falls } d = \langle 0,0 \rangle \\ \{\langle 1,1 \rangle, \langle 1,2 \rangle\} & \text{falls } d = \langle 0,1 \rangle \end{cases}$$

⌟

Kombinatorik

Das eben gezeigte Beispiel verdeutlicht, dass die kausale Monotonie sichergestellt sein muss, um nicht eine zu große Menge an Abweichungen zu bekommen. Die Sicherstellung funktioniert über eine Historie, welche zu jedem abweichenden Strom die Ursachen speichert. Der gleiche Mechanismus gilt für die Kombinatorik der Ausgabekanäle. Auch hier kann ohne Berücksichtigung der Historie meist nur eine Übermenge bestimmt werden. Eine Bestimmung der Fehlerwirkung ist also entsprechend aufwändig und steigt mit der Länge des zu betrachtenden Zeitraums.

unendliche Ströme

Die iterative Bestimmung der Auswirkung einer Modifikation ist nur für endliche Ströme geeignet. Ziel der meisten Analysen ist die Erhebung von Aussagen über unendliche, oder zumindest lange Ströme. Eine Möglichkeit zum Finden dieser Fehler ist der Einsatz von Fehlerbeschreibungen, welche als Fixpunkte in die Gleichungen eingesetzt werden können. Diese Fixpunkte sind, je nach Güte der Abschätzung, eine obere Grenze für die Auswirkung. Da die Iteration aus Definition 5.1.5 kausal monoton ist, gibt es nach dem Satz von Knaster-Tarski (siehe [Bro98], S. 119) immer einen Fixpunkt. Folgendes Beispiel zeigt den Einsatz eines Fixpunktes:

Beispiel 5.1.2 (Fixpunkt als Folgefehler)
Der Folgefehler des System S_1 in Beispiel 5.1.1 ist ein Drift, welcher sporadisch in jedem Rechenschritt die Abweichung um 1 erhöhen kann.

$$[\![\delta^{drift}_{O_{S\_1}}]\!]^{op} = (0 \leq (c - \tau_c - \mathsf{delay}_{\langle 0 \rangle}(c) + \mathsf{delay}_{\langle 0 \rangle}(\tau_c)) \leq 1)$$

Angewendet auf die Eingabe des Systems S_2 ergibt sich so der gleiche Fehler an dessen Ausgabe, jedoch mit einem vorangestellten $\langle 0 \rangle$, also $\delta^{0 \mapsto drift}_{I_{S\_1}}$. Diese Abweichung an der Eingabe ergibt wiederum an der Ausgabe von S_1 die Abweichung $\delta^{drift}_{O_{S\_1}}$. Auch der Induktionsanker wird mit $\delta^{0 \mapsto drift}_{I_{S\_1}}$ erfüllt, welches zu Zeitpunkten $t < 0$ keine Abweichung annimmt. ⌟

hoher Aufwand

Die Ermittlung eines Folgefehlers über die Betrachtung der Abweichungen an den Ausgaben fordert, wie in Beispiel 5.1.1 und 5.1.2 zu sehen, mit der Berücksichtigung der kausalen Monotonie und des Auffindens der Fixpunkte zwei nicht triviale Aktivitäten. Das führt dazu, dass der Aufwand, allgemeine Abweichungsabhängigkeitsmodelle komplexerer Systeme zu verbinden, sehr hoch ist. Die während dieser Arbeit vorgefundenen Ansätze zur Fehlerauswirkungsanalyse bilden deshalb Kompromisse, mit denen dieser Aufwand reduziert werden kann.

Reduktion der Komplexität

Abhängig von den zu betrachtenden Systemen, der zur Verfügung stehenden Mittel und der notwendigen Güte der Ergebnisse einer Fehlerauswirkungsanalyse können Methoden angewandt werden, um den Aufwand einer Auswirkungsanalyse zu reduzieren. Im Folgenden werden einige Möglichkeiten aufgelistet, welche auch in den während der Erstellung dieser Arbeit vorgefundenen Ansätzen vorhanden sind:

(I) Begrenzung des Betrachtungszeitraums
Prinzipiell lassen sich mit Hilfe von Abweichungsabhängigkeitsmodellen die Folgefehler exakt bestimmen. Der Aufwand ist allerdings sehr hoch, da zum einen die Abweichungsabhängigkeitsmodelle exakt für die möglichen eingehenden Abweichungen beschrieben sein müssen und zum anderen die Berechnung immer die Historie der einzelnen Abweichungsströme festhalten muss. Um die Wirkung für kurze Zeiträume zu betrachten, lassen sich, ähnlich wie beim Bounded-Modell-Checking, zeitnahe Folgen abschätzen. Hierbei ist zu berücksichtigen, dass mit jeder nichtdeterministischen Alternative der Abweichungsströme und jedem Rückkopplungsschritt der zu untersuchende Folgefehlerraum exponentiell wächst. Diese Abstraktion findet bei der Ermittlung von Abhängigkeiten der Ausgabemodifikationen in Abschnitt 5.2.2 Anwendung.

(II) Verzicht auf Verknüpfung von Zusammenhangsmodellen
Da die Problematik der Verknüpfung von Abweichungsabhängigkeitsmodellen wegen Restriktionen der Zeit nicht leistbar ist, wenn man die Kombinatorik voll berücksichtigen will, ist ein Kompromiss, auf die Verknüpfung dieser Modelle zu verzichten. Damit müssen die Zusammenhänge aus den Verhaltensmodellen ermittelt werden, in denen die Fehlverhalten entsprechend induziert werden. Die manuelle Ermittlung dieser Zusammenhänge ist dann meist nicht mehr möglich, da diese Modelle zu komplex sind. Auch für eine werkzeuggestützte Ermittlung sind die Verhaltensmodelle meist geeignet zu abstrahieren, wie in Kapitel 6 vorgestellt wird. Der Vorteil ist in diesem Falle, dass die Kombinatorik voll berücksichtigt werden kann. Der Nachteil ist, dass komplexere Modelle auf Verhaltensebene zusammengesetzt und am Stück analysiert werden müssen, was bei großen Modellen die Verifikationswerkzeuge überfordert. Diese Art der Analyse ist für kleinere Systeme geeignet, bei denen die Kombinatorik überwiegt. Dieser Ansatz, alles in Verhaltensmodellen zu rechnen, wird bei der Ermittlung von Abhängigkeiten zwischen Anforderungsmodifikationen (siehe Abschnitt 5.2.1) und Transitionsmodifikationen (siehe Abschnitt 5.2.3) erläutert.

(III) Einschränkung der Betrachtung der Kombinatorik
Häufig besteht nicht die Möglichkeit, die Verhalten der Systeme einheitlich zu modellieren und automatisch zu verifizieren. In diesen Fällen müssen die Abweichungsabhängigkeitsmodelle verknüpft werden. Diesem Anspruch wird begegnet, indem die Betrachtung der Kombinatorik zwischen den Fehlern so stark eingeschränkt wird, dass der Aufwand zur Bearbeitung handhabbar ist. Die Kombinatorik kann hierbei in zwei Dimensionen eingeschränkt werden. Erstens kann die Kombinatorik zwischen verschiedenen Komponenten außer Acht gelassen werden. Bei dieser Art der Abstraktion werden Redundanzen und Fehleraufhebungen nicht mehr berücksichtigt, aber ebenso auch Effekte, bei denen sich Fehler multiplizieren, übersehen. In Verfahren wie der FMEA, wird unter Umständen sogar die Kombinatorik zwischen den einzelnen Ausgaben nicht weiter berücksichtigt. Die zweite Dimension ist die Zeit. Auch hier kann die Kombinatorik bewusst vernachlässigt werden. In diesem Falle wird die Aufhebung und die Potenzierung der Fehler, die aufgrund der Zeit stattfindet, z.B. durch Rückkopplung außer Acht gelassen. Der Vorteil dieser Maßnahme ist, dass die Fehlermodelle manuell bearbeitbar und getrennt analysierbar werden. Der Nachteil liegt in der fehlenden Betrachtung der Kombinatorik bei Fehlerbeschreibungen hinsichtlich der Zeit und zwischen den Ausgabekanälen. Dieser Mechanismus ist entsprechend für mechanische Systeme, Regelungsbetrachtungen ohne Sicherheitsfunktionen und Funktionen ohne Rückkopplung geeignet. Ansätze hierzu sind bei den Zusammenhängen zu Anforderungsmodifikationen (Abschnitt 5.2.1) und Ausgabemodifikationen (Abschnitt 5.2.2) zu finden.

(IV) Abstraktion der Fehlerauswirkung
Letztlich wird häufig einhergehend mit den beiden zuvor genannten Mechanismen die Wirkung von Fehlern durch eine obere und untere Schranke eingegrenzt. Hierzu wird in den Zusammenhangsmodellen die Fehlerauswirkung nicht mehr nur durch die exakte Relation $\dashrightarrow$ angegeben, sondern auch durch die Relationen $\overset{\leq}{\dashrightarrow}$, $\overset{\geq}{\dashrightarrow}$ und $\overset{\otimes<}{\dashrightarrow}$. Diese Vereinfachungen helfen dabei, bestimmte Wirkungen von Fehlern auszuschließen, aber auch bestimmte Wirkungen sicherzustellen, um den Einsatz von Sicherheitsfunktionen zu prüfen.

Zusammenfassung

In diesem Abschnitt 5.1 wurden gegebene Definitionen für Folgefehler und Fehlerauswirkung gezeigt. In beiden Fällen wurde festgestellt, dass Möglichkeiten zur Abstraktion der Zusammenhänge notwendig sind. Deshalb wurden die Teilmodifikation und Modelle definiert, mit denen Zusammenhänge abstrakt modelliert werden können. Weiter wurden Ansätze zur Reduktion des Aufwandes der Zusammenhangsermittlung vorgestellt, die mit den abstrakten Zusammenhangsmodellen verwendet werden können. Ein Einsatz der verschiedenen Ansätze ist in Kapitel 6 zu lesen. Dort werden anhand der Fallstudie die verschiedenen anwendbaren Techniken, aber auch die generellen Schwächen angegeben. In Abschnitt 5.2 werden spezifische Arten der Fehlerzusammenhänge und Techniken zur Ermittlung dieser Zusammenhänge beschrieben, welche sich aus den hier vorgestellten Ansätzen und Modellen ergeben.

5.2 Spezifische Modelle und Ermittlungstechniken

Abschätzung

Fast allen Analyseverfahren gemeinsam ist, dass die exakte Modellierung von Zusammenhängen zwischen Fehlern zu aufwändig und komplex ist. Deshalb werden in der Praxis häufig die Zusammenhänge abstrahiert (siehe Punkt (IV) im vorhergehenden Abschnitt) und sind nur in Idealfällen exakt. Folgendes Beispiel zeigt die Abschätzung einer Wirkung anhand einer oberen und unteren Schranke.

Beispiel 5.2.1 (Schranken bei der Abschätzung)
Gegeben sei das System aus Beispiel 5.1.1. Bei exakter Betrachtung der Fehlerauswirkung hat sich der Fehler $\delta_{O_{S\_1}}^{drift}$ (siehe Beispiel 5.1.2) ergeben.

Wird in der Iteration nun $\dashrightarrow$ durch $\overset{\leq}{\dashrightarrow}$ ersetzt, so erfüllen auch Drifts mit größeren Maximalsteigungen die Gleichung. Es gilt also auch die obere Schranke:

$$[\![\delta_{O_{S\_1}}^{x\_drift}]\!]^{op} = (0 \leq (c - \tau_c - \mathsf{delay}_{\langle 0 \rangle}(c) + \mathsf{delay}_{\langle 0 \rangle}(\tau_c)) \leq x) \text{ mit } x \geq 1$$

Durch Ersetzung mit $\overset{\geq}{\dashrightarrow}$ erfüllen auch Drifts mit kleinerer Maximalsteigung die Iteration. Es gilt die untere Schranke:

$$[\![\delta_{O_{S\_1}}^{y\_drift}]\!]^{op} = (0 \leq (c - \tau_c - \mathsf{delay}_{\langle 0 \rangle}(c) + \mathsf{delay}_{\langle 0 \rangle}(\tau_c)) \leq y) \text{ mit } 0 \leq y \leq 1$$

Die Wirkung kann so auf mindestens eine Abweichung aus $\delta_{O_{S\_1}}^{y\_drift}$ und höchstens eine Abweichung aus $\delta_{O_{S\_1}}^{x\_drift}$ eingegrenzt werden. ⌟

Ursache \ Wirkung	*Systeme*	*Verhaltensbed.*	*Ausgaben*	*Transitionen*
Systeme (Blackbox)	✔	✔	✘	✘
Verhaltensbedingungen	✔	✔	✘	✘
Ausgaben	✔	✔	✔	✘
Transitionen	✔	✔	✘	✔

Tabelle 5.1: Betrachtete Zusammenhänge in dieser Arbeit

Schranken Durch die Abschätzung mit der oberen Schranke $\overset{\leq}{\dashrightarrow}$ kann das Auffinden eines Fixpunktes deutlich einfacher gestaltet sein, denn unter anderem kann so auf die Übereinstimmung des Induktionsankers mit dem Start der beschriebenen Abweichung verzichtet werden. Damit wären beliebige Modifikationen gültig, welche größer als der echte Fixpunkt sind. Mit Hilfe dieser Abschätzung kann die Überprüfung eines Zusammenhangs hin auf wenige potentielle Folgefehlverhalten untersucht werden und so die Eingrenzung der Wirkung von Fehlern bzw. die Folgefehler einfacher geschätzt werden.

Techniken Entsprechend der Modellierungstechniken aus Kapitel 3 sind spezifische Ansätze zur Zusammenhangsermittlung anwendbar. Innerhalb der jeweiligen Modellierungstechniken und zwischen diesen gibt es Ansätze in der Literatur. Diese werden in diesem Abschnitt vorgestellt und an das Fehlermodell aus Kapitel 4 angepasst. Des Weiteren werden in der Fallstudie vorgefundene informelle Zusammenhänge, die in der Literatur nicht formalisiert gefunden wurden, formal beschrieben und Ansätze zu deren Ermittlung vorgestellt. Tabelle 5.1 gibt eine Übersicht über die Inhalte dieses Abschnitts. Hier wird angegeben, von welcher zu welcher Modellierungstechnik Zusammenhangsmodelle und Techniken zu deren Ermittlung angegeben sind. Die Unterabschnitte dieses Abschnitts sind entsprechend der Zeilen sortiert. Für jeden Ansatz wird beschrieben, was abstrahiert wird und welche Folgen für die Anwendung daraus entstehen. Zu den Folgen gehören Verbesserungen der Handhabbarkeit, Ausblendung vorhandener Wirkungen, ungenaue Wirkungsangaben, usw.

5.2.1 Zusammenhänge bei Black-Box-Spezifikationen

Beitrag Dieser Abschnitt behandelt die Verknüpfungen von Fehlern zwischen Prädikaten, die Systeme beschreiben und einzelnen Verhaltensbedingungen. Das Modell aus Abschnitt 5.1.1 wird auf Formeln übertragen und allgemein die Ermittlung der Folgefehler über Formeln sowohl für stromgebundene wir auch zustandsgebundene Formeln analysiert. Zur Ermittlung der Fehlerfolgen in hierarchischen Systemen wird der Parallelvergleich vorgeschlagen. In den Fallstudien wurden die Fehler häufig nicht einem ganzen System zugeordnet, sondern Verhaltensbedingungen. Deshalb werden in diesem Abschnitt weiter spezifische Zusammenhänge formalisiert, die sich auf diese Verknüpfung der Bedingungen konzentrieren. Zu diesen gehört die Funktionsvernetzung und die Einfachfehlervernetzung.

Bindung an Ströme

Basis für diesen Abschnitt ist das Zusammenhangsmodell aus Abschnitt 5.1.1. Prinzipiell gilt für die Modellierung der Zusammenhänge von Blackbox-Spezifikationen das Gleiche, wie für die Verhaltensmodelle. Sind diese an Ströme gebunden, so kann die Teilmengenrelation durch die logische Implikation und die Komposition durch ein $\wedge$ ersetzt werden. Die Bedingung für eine Teilmodifikation in Definition 5.1.2 ist z.B. für Blackbox-Spezifikationen erfüllt, wenn gilt:

Theorem 5.2.1 (Teilmodifikation)
Eine Modifikation $\mathcal{M}_S^{sub}$ ist eine echte *Teilmodifikation* einer anderen Modifikation $\mathcal{M}_S^{super}$, geschrieben als

$$\rightsquigarrow: PRED_{MOD_S}^{Stream} \times PRED_{MOD_S}^{Stream} \rightarrow BOOL$$
$$[\![\mathcal{M}_S^{super}]\!] \rightsquigarrow [\![\mathcal{M}_S^{sub}]\!]$$

wenn gilt:

$$([\![\mathcal{M}_S^{sub}]\!] \overset{dep}{\not\Leftrightarrow} [\![\mathcal{M}_S^{super}]\!]) \wedge ([\![\overline{E}_S^{super}]\!] \Rightarrow [\![\overline{E}_S^{sub}]\!]) \wedge ([\![F_S^{sub}]\!] \Rightarrow [\![F_S^{super}]\!])$$

(Siehe Def. 5.1.2 Teilmod., 4.2.3 Unabhängigkeit und 3.4.3 Verfeinerung) ⌟

Ermittlung über Fehlverhalten

Eine Modifikation des Systems S zeigt sich in einer beobachtbaren Änderung des Sollverhaltens R_S. Es kommen also entweder beliebige Tupel hinzu, oder es werden Tupel entfernt. Eine Modifikation $\mathcal{M}_S$ ist also genau dann eine Folge der Modifikationen $(\mathcal{M}_{S_1}, \mathcal{M}_{S_2})$, wenn weder zu viele Tupel entfernt werden, noch zu wenige hinzugefügt werden. Die korrekte Modellierung eines Folgefehlers kann also überprüft werden, in dem die Fehlverhalten, welche sich aus den Modifikationen ergeben, verglichen werden. Dieser Vergleich kann in seiner Aussagekraft gesteigert werden, wenn nicht auf Gleichheit geprüft wird, sondern die logische Implikation verwendet wird. So lässt sich ermitteln, ob für eine korrekte Modellierung weitere Tupel entfernt und bzw. oder hinzugefügt werden müssen. Die folgende Formel stellt die zu prüfende Bedingung für einen Folgefehler dar. Um die Lesbarkeit zu erhöhen, wird das komponierte Verhalten der modifizierten Teilsysteme durch $[\![S]\!]\Delta[\![\mathcal{M}_S^a]\!]$ ersetzt[3]. Die zu prüfende Eigenschaft für ein System $S = S_1 \otimes S_2$ ist:

$$\begin{aligned}
&([\![\mathcal{M}_{S_1}]\!],[\![\mathcal{M}_{S_2}]\!]) \dashrightarrow [\![\mathcal{M}_S]\!] \\
&\quad \Leftrightarrow [\![S_1]\!]\Delta[\![\mathcal{M}_{S_1}]\!] \wedge [\![S_2]\!]\Delta[\![\mathcal{M}_{S_2}]\!] = [\![S]\!]\Delta[\![\mathcal{M}_S]\!] \\
&\quad \Leftrightarrow [\![S]\!]\Delta[\![\mathcal{M}_S^a]\!] = [\![S]\!]\Delta[\![\mathcal{M}_S]\!] \qquad \text{mit} \quad \mathcal{M}_S^a \simeq (\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \\
&\quad \Leftrightarrow ([\![S]\!]\Delta[\![\mathcal{M}_S]\!] \Rightarrow [\![S]\!]\Delta[\![\mathcal{M}_S^a]\!]) \wedge ([\![S]\!]\Delta[\![[\![\mathcal{M}_S^a]\!]]\!] \Rightarrow [\![S]\!]\Delta[\![\mathcal{M}_S]\!])
\end{aligned} \tag{5.1}$$

Teilmodifikation

Die Überprüfung eines exakten Folgefehlers findet meist nicht statt, da die Beschreibungen der Fehler häufig kompliziert sind. Alternativ wird auf einen übergreifenden Fehler hin geprüft, welcher leicht zu interpretieren ist. Die Überprüfung kann ähnlich der Überprüfung auf einen Folgefehler anhand des Fehlverhaltens stattfinden. Allerdings erfüllt hier eine zu große Menge F_S bzw. eine zu große Menge E_S auch

[3] Die Modifikation $\mathcal{M}_S^a$ muss beim Fehlverhaltensvergleich nicht explizit berechnet werden. Sie entspricht
$[\![\overline{E}_S^a]\!] = [\![\overline{E}_{S_1}]\!] \wedge [\![\overline{E}_{S_2}]\!]$ und $[\![F_S^a]\!] = ([\![F_{S_1}]\!] \wedge [\![F_{S_2}]\!] \vee [\![F_{S_1}]\!] \wedge [\![\overline{E}_{S_2}]\!] \wedge [\![S_2]\!] \vee [\![\overline{E}_{S_1}]\!] \wedge [\![F_{S_2}]\!] \wedge [\![S_1]\!]) \wedge [\![\overline{E}_S^a]\!]$.

die Bedingung der korrekten Modellierung eines Folgefehlers. Entsprechend wird für die Folge $\mathcal{M}_S$ die Korrektheit in zwei Teilen geprüft: Es dürfen von $\mathcal{M}_S^a$ keine Pfade hinzugefügt werden, die in F_S nicht enthalten sind, und umgekehrt keine Pfade entfernt werden, welche in $\mathcal{M}_S$ nicht entfernt werden. Daraus ergibt sich folgende Anfrage:

$$\begin{aligned}&([\![\mathcal{M}_{S_1}]\!], [\![\mathcal{M}_{S_2}]\!]) \overset{\leq}{\dashrightarrow} [\![\mathcal{M}_S]\!] \\ &\Leftrightarrow (([\![S]\!]\Delta([\![\overline{E}_S]\!], \mathit{false}) \Rightarrow [\![S]\!]\Delta[\![\mathcal{M}_S^a]\!]) \wedge ([\![S]\!]\Delta[\![\mathcal{M}_S^a]\!] \Rightarrow [\![S]\!]\Delta(\mathit{true}, [\![F_S]\!])))\end{aligned} \tag{5.2}$$

Verfeinerung Die Überprüfung auf übergreifende Fehler findet in dem Kontext statt, dass auf höheren Ebenen der Systemhierarchie die Spezifikationen bzw. die Fehlerbeschreibungen abstrakter werden. Die bisherigen Beschreibungen zu Fehlerzusammenhängen berücksichtigen zwar eine Abstraktion der Fehler selbst, nicht aber eine Abstraktion bzw. Verfeinerung der Systembeschreibungen (siehe Abschnitt 3.4.2). Besteht eine Verfeinerungsbeziehung, so muss entsprechend berücksichtigt werden, dass beim verfeinerten System bereits einige Verhaltenstupel zum allgemeineren System fehlen, welche nicht auf die Modifikationen zurückzuführen sind. Die bestehenden Ansätze zu Sicherheitsanalysen betrachten die Verfeinerung nur in der Hinsicht, dass die F-Mengen der folgenden Modifikationen ausreichend groß sind. So werden z.B. in [Bre01] Modifikationen mit E- und F-Mengen betrachtet, im Falle der Verfeinerung jedoch die E-Mengen nicht mehr berücksichtigt. Zur Prüfung einer Teilmodifikation leiten sich im Falle eines verfeinerten Systems die Bedingung folgendermaßen ab[4]:

$$\begin{aligned}&([\![\mathcal{M}_{S_1}]\!], [\![\mathcal{M}_{S_2}]\!]) \overset{\leq}{\dashrightarrow} [\![\mathcal{M}_S]\!] \quad \text{mit} \quad S \rightsquigarrow S^a, \quad S^a = S_1 \otimes S_2, \quad S^{\hat{a}} = S \setminus S^a \\ &\Leftrightarrow (([\![S]\!]\Delta[\![\overline{E}_S]\!] \Rightarrow ([\![S^a]\!] \vee [\![S^{\hat{a}}]\!])\Delta[\![\mathcal{M}_S^a]\!]) \wedge ((([\![S^a]\!] \vee [\![S^{\hat{a}}]\!])\Delta[\![\mathcal{M}_S^a]\!] \Rightarrow [\![S]\!]\Delta[\![F_S]\!])) \\ &\Leftrightarrow \quad \text{vgl. Gleichung D.1 und Gleichung D.2} \\ &\Leftrightarrow (([\![S^a]\!]\Delta[\![\overline{E}_S]\!] \Rightarrow [\![S^a]\!]\Delta[\![\mathcal{M}^a]\!]) \wedge ([\![S^a]\!]\Delta[\![\mathcal{M}_S^a]\!] \Rightarrow [\![S]\!]\Delta[\![F_S]\!]))\end{aligned} \tag{5.3}$$

operative Bindung Sind die Formeln für die Systeme nicht an Ströme gebunden, sondern an Zustände, so ist die oben gezeigte Formel zur Überprüfung einer korrekten Ermittlung eines übergreifenden Fehlers ungeeignet. Die Ermittlung der Menge $[\![\overline{E}_S]\!]^{op}$ kann wie in Gleichung 5.2 ermittelt werden, da eine Reduktion der möglichen operativen Verhalten auch eine Reduktion der möglichen Ausführungspfade mit sich bringt. Die Menge $[\![F_S]\!]^{op}$ ergibt jedoch zusammen mit den verbleibenden Zuständen aus $[\![S]\!]^{op}$ die Menge der möglichen Pfade für $[\![F_S]\!]$. Die Menge $[\![F_S]\!]$ muss also, wenn sie operativ beschrieben werden soll, als ein Verhalten $[\![S]\!]^{op}\Delta[\![\mathcal{M}_S^F]\!]^{op}$ ausgedrückt werden, welches auf die möglichen falschen Ausführungspfade schließen lässt. Damit gilt für operative Beschreibungen[5]:

[4] Kurzschreibweise: $[\![S]\!]\Delta[\![\overline{E}_S]\!]$ für $[\![S]\!]\Delta([\![\overline{E}_S]\!], \mathit{false})$ und $[\![S]\!]\Delta[\![F_S]\!]$ für $[\![S]\!]\Delta(\mathit{true}, [\![F_S]\!])$

[5] Zur Vereinfachung wird implizit angenommen, dass die stromgebundenen und wertgebundenen Formeln das Gleiche beschreiben, also: $[\![S]\!] = [\![S]\!]^{op}[\uparrow T \uparrow]$, usw.

$$
\begin{aligned}
&(\llbracket\mathcal{M}_{S_1}\rrbracket^{op}, \llbracket\mathcal{M}_{S_2}\rrbracket^{op}) \overset{\leq}{\dashrightarrow} (\llbracket\overline{E}_S\rrbracket^{op}, \llbracket S\rrbracket^{op}\Delta\llbracket\mathcal{M}_S^F\rrbracket^{op})\\
&\Leftrightarrow ((\llbracket S\rrbracket^{op}\Delta\llbracket\overline{E}_S\rrbracket^{op} \Rightarrow \llbracket S\rrbracket^{op}\Delta\llbracket\mathcal{M}_S^a\rrbracket^{op}) \wedge (\llbracket S\rrbracket^{op}\Delta\llbracket\mathcal{M}_S^a\rrbracket^{op} \Rightarrow \llbracket S\rrbracket^{op}\Delta\llbracket\mathcal{M}_S^F\rrbracket^{op}))[\uparrow T\uparrow]\\
&\Rightarrow (\llbracket S\rrbracket\Delta\llbracket\overline{E}_S\rrbracket \Rightarrow \llbracket S\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket) \wedge (\llbracket S\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket \Rightarrow \llbracket S\rrbracket\Delta\llbracket S\rrbracket^{op}\Delta\llbracket\mathcal{M}_S^F\rrbracket^{op}[\uparrow T\uparrow])\\
&\Rightarrow (\llbracket\mathcal{M}_{S_1}\rrbracket^{op}[\uparrow T\uparrow], \llbracket\mathcal{M}_{S_2}\rrbracket^{op}[\uparrow T\uparrow]) \overset{\leq}{\dashrightarrow} \llbracket\mathcal{M}_S\rrbracket^{op}[\uparrow T\uparrow]\\
&\overset{\backsim}{\Rightarrow} (\llbracket\mathcal{M}_{S_1}\rrbracket, \llbracket\mathcal{M}_{S_2}\rrbracket) \overset{\leq}{\dashrightarrow} \llbracket\mathcal{M}_S\rrbracket
\end{aligned}
\tag{5.4}
$$

Mit den obigen Gleichungen ist gezeigt, dass die Zusammenhänge zwischen Modifikationen von den allgemeinen Modifikationen aus Abschnitt 5.1 auf stromgebundene und wertgebundene Formeln übertragen werden können. Dies ist die Grundlage dafür, die Zusammenhänge mit logischen Werkzeugen zu verifizieren. Im Folgenden werden Ansätze zur Umsetzung der Verifikation der Zusammenhänge vorgestellt.

5.2.1.1 Parallelvergleich

Systeme vergleichen

Eine intuitive Möglichkeit, einen Fehlerzusammenhang zu überprüfen, ist, die jeweils modifizierten Systeme nebeneinander zu halten und zu vergleichen. Die beiden Systeme $\llbracket S\rrbracket\Delta\llbracket\mathcal{M}_S\rrbracket$ und $\llbracket S^a\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket$ werden als zwei Systeme modelliert, welche verschiedene disjunkte Variablen haben. Besonders das Prädikat $\llbracket S^a\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket$ kann so Schrittweise entlang der Systemhierarchie parallel zu $\llbracket S\rrbracket\Delta\llbracket\mathcal{M}_S\rrbracket$ erstellt werden und die einzelnen Teilfehler können induziert werden.

Ersetzung

Um die Systeme zu vergleichen werden für eines der parallelen Systeme die Variablen von S durch die von S' ersetzt. Für das System gelte:

$$S = R \succ S' \succ A \quad \text{oder} \quad \llbracket S\rrbracket = \llbracket R\rrbracket \wedge \llbracket S'\rrbracket \wedge \llbracket A\rrbracket \quad \text{mit} \quad R_R = R_A = R_{ID}$$

Da die Abstraktion und die Realisierung die Identität sind, lässt sich die Modifikation $\llbracket\mathcal{M}_S\rrbracket$ durch eine syntaktische Schnittstellenanpassung (siehe Definition 4.2.5) auf $\llbracket S'\rrbracket$ anwenden. Es gilt:

$$\llbracket\mathcal{M}_{S'}\rrbracket = \llbracket\mathcal{M}_S\rrbracket[\updownarrow \pi_{(S,S')}]$$

Ein modellierter Zusammenhang kann überprüft werden, in dem in den Gleichungen entsprechend das parallele System eingesetzt wird. Für die Überprüfung auf einen übergreifenden Folgefehler gilt zum Beispiel:

$$
\begin{aligned}
(\llbracket\mathcal{M}_{S_1}\rrbracket, \llbracket\mathcal{M}_{S_2}\rrbracket) \overset{\leq}{\dashrightarrow} \llbracket\mathcal{M}_S\rrbracket \quad \Leftarrow \quad & (\llbracket R\rrbracket \wedge \llbracket S'\rrbracket\Delta\llbracket\overline{E}_{S'}\rrbracket \wedge \llbracket A\rrbracket \Rightarrow \llbracket S^a\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket)\\
& \wedge(\llbracket S^a\rrbracket\Delta\llbracket\mathcal{M}_S^a\rrbracket \Rightarrow \llbracket R\rrbracket \wedge \llbracket S'\rrbracket\Delta\llbracket F_{S'}\rrbracket \wedge \llbracket A\rrbracket)
\end{aligned}
$$

Komplexität

Der Vorteil dieser Methode des nebeneinander Haltens, also dem einfachen Aufbau einer Anfrage zur Überprüfung einer Abhängigkeit fordert ein hohes Maß der Leistung der Verifikationswerkzeuge. Zum Einen verdoppelt sich durch das nebeneinander Halten die Menge der zu beobachtenden Variablen. Sind die Algorithmen zur Analyse vom Rechenaufwand her polynomial oder sogar exponentiell, so steigt der Rechenaufwand dadurch um das Vielfache. Zum Anderen berücksichtigt diese Art der Anfragen nicht die Mächtigkeit der Verifikationswerkzeuge. Die Anfragen gehen schnell auf die Ebene der First-Order-Logic oder höher, was eine automatisierte Analyse tendenziell ausschließt. Kapitel 6 beschäftigt sich mit Möglichkeiten, diese Anfragen zu verifizieren.

5.2.1.2 Verhaltensanforderungsbezug

Bezugspunkt Folgefehler

Die im vorhergehenden Abschnitt vorgestellten Anfragen bezogen sich auf die gesamte Beschreibung eines Systems oder Teilsystems. Besonders in den oberen Ebenen der Systemhierarchie müssten bei detaillierten Systembeschreibungen für eine Anfrage alle Variablen des Systems berücksichtigt werden, wodurch der Rechenaufwand sehr groß wird. Geht man davon aus, dass für die Systeme eine Spezifikation vorhanden ist, in welcher die Verhaltensanforderungen in einer konjunktiven Form aufgelistet sind (siehe Abschnitt 3.3.1), so bietet es sich an, die Modellierung der Zusammenhänge an diese anzupassen. Ein Folgefehler bezieht sich dann nicht mehr auf das gesamte System, sondern nur noch auf eine Menge von Verhaltensanforderungen (siehe Abschnitt 4.2.1). Eine getrennte Betrachtung von Fehlern zu einzelnen Verhaltensanforderungen ist oftmals einfacher zu ermitteln und auch in der informellen Dokumentation einer Sicherheitsanalyse eher üblich (siehe Abschnitt 2.3 und 2.4). Zur Überprüfung eines Fehlerzusammenhangs für einen Folgefehler reicht es, nicht mehr das ganze System, sondern nur die jeweils betroffenen Verhaltensanforderungen zu modellieren. Der Vorteil dieser Sicht ist, dass die Fehler besser Verhaltensanforderungen zugeordnet und damit strukturierter notiert, sowie einfacher ermittelt werden können. Auch die Integrierbarkeit in eine Anforderungsspezifikation, welche zusätzlich informelle Anforderungen enthält, ist so einfach zu realisieren und auch die Zusammenhänge sind einfacher nachzuvollziehen, da nur Teile der gesamten Spezifikation betrachtet werden.

Verlust der Präzision

Die Betrachtung von Teilen der Spezifikation für die Folge eines Fehlers führt allerdings hinsichtlich der Kombinatorik zwischen den Verhaltensanforderungen zu einer Abstraktion der Zusammenhänge. Umfasst eine Sicherheitsanalyse mehr als zwei Ebenen in der Systemhierarchie, wie zum Beispiel in Abschnitt 2, so wird die ermittelte Folge unterer Ebenen wieder als Grundlage für die Fehlerermittlung zur nächsten Ebene der Systemhierarchie genutzt. Es werden über die Relation Γ (siehe Definition 5.1.3) iterativ die Folgen eines Fehlers bis hin zur Gesamtsystemschnittstelle modelliert. Mit dieser Abstraktion verliert man eine Menge von Informationen, die zur Bestimmung von Fehleraufhebungen und Fehlerpotentierungen notwendig sind. Folgendes Beispiel veranschaulicht diesen Sachverhalt:

Beispiel 5.2.2 (Blackbox-Spezifikationsabhängigkeiten)
Gegeben sei das System aus Beispiel 5.1.1. Weiter gegeben seien für das System zwei Verhaltensanforderungen Φ_S^1 und Φ_S^2. Φ_S^1 beschreibt die Berechnung der Ausgabe c und Φ_S^2 die Berechnung der Ausgabe d.

Die aus $\mathcal{M}_{S_1}^1$ folgende Modifikation ist für die Anforderung Φ_S^1 ein Drift, wie auch für die Anforderung Φ_S^2. Allerdings driften die beiden in den Anforderungen beschriebenen Kanäle immer exakt gleich, lediglich um einen Takt verschoben. Für den Folgefehler bezogen auf die gesamte Spezifikation gilt:
$[\![\mathcal{M}_{O_S}^{drift_{cd}}]\!]^{op} =$
$(true, 0 \leq (c - \tau_c - \mathsf{delay}_{\langle 0 \rangle}(c) + \mathsf{delay}_{\langle 0 \rangle}(\tau_c)) \leq 1 \wedge (d - \tau_d = \mathsf{delay}_{\langle 0 \rangle}(c) - \mathsf{delay}_{\langle 0 \rangle}(\tau_c)))$

Beziehen sich die Fehler jeweils auf die Anforderungen Φ_S^1 und Φ_S^2, kann dieser Zusammenhang nicht mehr berücksichtigt modelliert werden. Die Kombination der Fehler wäre dann in diesem Falle größer als die Fehlerbeschreibung mit Bezug auf das Gesamtsystem:

$$(\mathcal{M}_c^{drift} +^{\Phi_c, \Phi_d} \mathcal{M}_d^{drift}) \rightsquigarrow \mathcal{M}_{O_S}^{drift_{cd}}$$ ⌟

Einfachfehler

Die separate Betrachtung der Fehler zu den jeweiligen Blackbox-Spezifikationen(BS) kann im Sinne eines Einfachfehlers so betrachtet werden, dass man davon ausgeht, dass ein Einfachfehler mit einem Fehler an einer BS gleichzusetzen ist. Entsprechend ist ein Mehrfachfehler eine Menge von Fehlern, die sich jeweils auf verschiedene Verhaltensanforderungen beziehen. Dies entspricht der in der Fallstudie vorgefundenen Verknüpfung der Fehler in der FMEA und FTA. Diese Beziehungen zwischen einzelnen BS werden abhängig von der Analyseart auf zwei Arten betrachtet. Zum Einen überprüft man, ob ein Einfachfehler mit alleinigem Auftreten einen Folgefehler verursachen kann (FMEA). Zum Anderen überprüft man, wie ein Einfachfehler im Zusammenspiel mit anderen Fehlern zu einem Folgefehler beitragen kann.

5.2.1.3 anforderungsbezogene Einfachfehlerabhängigkeit

Einfachfehler

Der in der Standard-FMEA übliche Fall ist die Beschreibung der Zusammenhänge, bei denen einzelne Fehler eine Wirkung haben. Diese Fehler beziehen sich aus Verhaltenssicht auf Verhaltensanforderungen. Da die Kombinationen nicht betrachtet werden, sind weder Fehlerpotenzierungen, noch Fehleraufhebungen in die Analyse mit einbezogen. Als Ursache wird nur die Manipulation einer einzelnen Anforderung verwendet. Die Folgemodifikation wird dann nur aus dem Blickwinkel der zu untersuchenden Verhaltensanforderung, welche sich in der nächst höheren Ebene der Systemhierarchie befindet, betrachtet. Der tatsächliche Folgefehler ist dann, wie im letzten Abschnitt vorgestellt, eine Kombination aus konjunktiv eingeschränkt zusammengesetzten Folgefehlern zu Verhaltensanforderungen. Eine anforderungbezogene Einfachfehlerabhängigkeit ist definiert als:

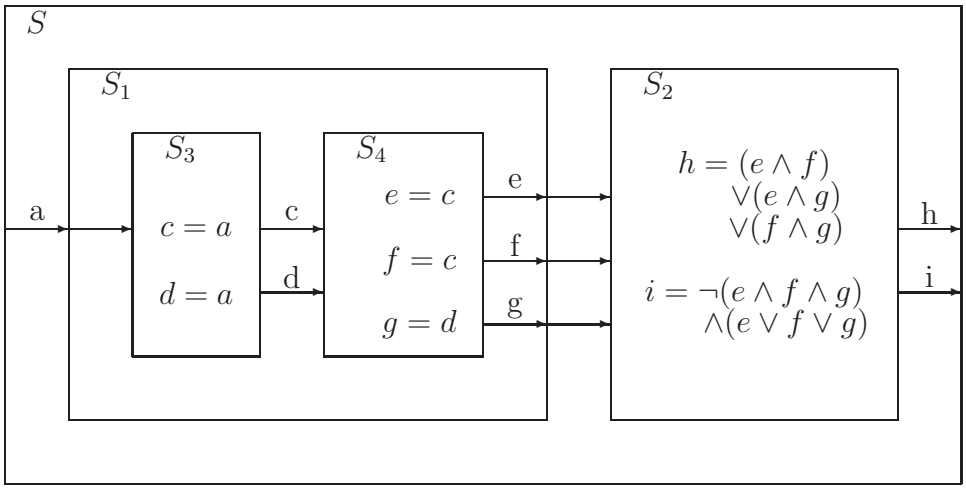


Abb. 5.5: Beispiel eines redundanten Systems

Definition 5.2.1 (anforderungsbezogene Einfachfehlerabhängigkeit)
Gegeben sei ein System $S \rightsquigarrow S_1 \otimes S_2$ mit $S, S_1, S_2 \in \mathbb{S}$. Die Systeme seien beschrieben mit den konjunkten Prädikaten $[\![S]\!] = \Phi_x \wedge ... \wedge \Phi_y$ und $[\![S_1]\!] = \Phi_a \wedge ... \wedge \Phi_b$.

Eine Modifikation einer Anforderung Φ_a hat dann eine Modifikation einer Anforderung Φ_x im Sinne einer *anforderungsbezogenen Einfachfehlerabhängigkeit* zur Folge, geschrieben als $\overset{||^1\leq}{\dashrightarrow}$, wenn gilt:

$$[\![\mathcal{M}_a]\!] \overset{||^1\leq}{\dashrightarrow} [\![\mathcal{M}_x]\!] \quad \overset{def}{=}$$
$$\exists [\![\mathcal{M}_y]\!], \ldots, [\![\mathcal{M}_z]\!] : ([\![\mathcal{M}_a]\!], [\![\mathcal{M}^{\emptyset}_{S_2}]\!]) \overset{\leq}{\dashrightarrow} \sum^{\langle \Phi_x, \Phi_y, \ldots \rangle} \langle [\![\mathcal{M}_x]\!], \ldots, [\![\mathcal{M}_y]\!] \rangle \quad \lrcorner$$

Abstraktion

Betrachtet man diese Definition, so ist der Folgefehler, der sich aus den Teilfehlern zusammensetzt eine obere Schranke für den tatsächlichen Folgefehler. Bei einer Einfachkanalbetrachtung werden auf der nächsthöheren Ebene nur die einzelnen Teilfehler betrachtet. Hierdurch wird explizit die Potenzierung und die Aufhebung von Effekten einzelner Kanäle nicht weiter betrachtet. Folgendes Beispiel verdeutlicht nochmals diesen Effekt:

Beispiel 5.2.3 (anforderungsbezogene Einfachfehlerabhängigkeit)
Gegeben sei das System S in Abbildung 5.5. Weiter seien eine Menge von Anforderungen gegeben, welche die Berechnung der jeweiligen Kanalwerte beschreiben. Das System berechnet aus in Kanal a eingehenden Werten die Ausgabe h. Zur Vereinfachung wird hier stellvertretend für die Berechnungen die Identität verwendet. Die Teilsysteme S_3 und S_4 sind Vertreter für redundante Berechnungen. Das System S_2 nimmt die Ergebnisse auf und fügt diese im Sinne einer 3n-Redundanz zusammen. Die Ausgabe i indiziert, ob ein Berechnungsfehler gefunden wurde.

Für das System gelten folgende Beziehungen:

Ein Einfachfehler an der Ausgabe e, f bzw. g hat keine Wirkung auf die Ausgabe h, da die 3n-Redundanz dessen Wirkung beseitigt.

$$\mathcal{M}_e^{TsF} \overset{||^1\leq}{\dashrightarrow} \mathcal{M}_h^{\emptyset}$$

Ein Fehler der Ausgabe c hat sowohl auf die Ausgaben e und f eine Wirkung.

$$\mathcal{M}_c^{TsF} \overset{||^1\leq}{\dashrightarrow} \mathcal{M}_e^{TsF} \quad \text{und} \quad \mathcal{M}_c^{TsF} \overset{||^1\leq}{\dashrightarrow} \mathcal{M}_f^{TsF}$$

Wäre die Systemgrenze S_1 nicht vorhanden, so würde sich der Fehler der Ausgabe c über den Mehrfachfehler $\mathcal{M}_{e,f}$ auf die Ausgabe h auswirken. So werden die Fehler an der Systemgrenze S_1 getrennt betrachtet und die Wirkung dieses *Common-Cause*-Fehlers abstrahiert. ⌟

Eignung

Die Tatsache, dass Fehleraufhebungen und Potenzierung nicht beachtet werden, schränken die Einsatzmöglichkeiten eines solchen Modells deutlich ein. Im Wesentlichen sind diese Modelle für Systeme gedacht, in denen noch keine Sicherheitsmaßnahmen ergriffen worden sind. Um die Wirkung von Common-Cause-Fehlern nicht zu übersehen, dürfen die Systeme keine Redundanzen und keine Sicherheitsfunktionen enthalten, welche die Fehlertoleranz erhöhen (siehe [DIN04]). Ein derartiges System ist die Startsituation, in der eine FMEA idealerweise gestartet wird und bei der das Hinzufügen der Sicherheitsmechanismen dann im Maßnahmenteil steht und nicht im Funktionsnetz. Enthalten die Systeme bereits Sicherheitsmaßnahmen, so muss entweder das Fehlerzusammenhangsmodell geeignet um Mehrfachfehler erweitert werden oder das Analyseergebnis mit dem Bewusstsein verwendet werden, dass Common-Cause-Fehler unentdeckt sind.

Abweichung

Bei sequentiell komponierten Systemen, deren Anforderungen an Kanäle gebunden sind (vgl. Abbildung 5.5), ist die Ermittlung der Zusammenhänge der Modifikationen über die Abweichungen (siehe Abschnitt 5.1.2) relativ einfach durchzuführen. Zu den Systemen müssen jeweils die Laufzeitbedingungen der Redundanz angegeben werden und die Zusammenhangsmodelle der Abweichungen wären identisch mit den in Beispiel 5.2.3 dargestellten. Der Vorteil ist hier, dass die Entwickler im Falle einer manuellen Analyse durch Verfolgen des Signalflusses die Wirkung bestimmen können, was im Falle einer Rückkopplung meist zu komplex wird.

5.2.1.4 Funktionsabhängigkeit

Funktionsnetz

Bei der anforderungsbezogenen Einfachfehlerabhängigkeit werden die Kombinationsmöglichkeiten zwischen Verhaltensanforderungen komplett außer Acht gelassen und so mögliche Fehlerpotenzierungen nicht entdeckt. Die gezeigten Beziehungen zwischen den Fehlern sind, wie in Abschnitt 2.3 zu sehen, immer innerhalb des Funktionsnetzes. Dieses stellt eine maximale Begrenzung dar, innerhalb derer sich die Fehlerauswirkungen in Kombination mit anderen Fehlern bewegen können. Informell gesagt bedeutet eine Beziehung von einer Anforderung Φ_x zu einer Anforderung Φ_y im Funktionsnetz, dass es mindestens eine, wenn auch durch Fehler hervorgerufene Situation gibt, bei dem eine Modifikation der Anforderung Φ_x existiert, die zu einer Modifikation der Anforderung Φ_y führt.

Definition 5.2.2 (Funktionsabhängigkeit)
Gegeben sei ein System $S \rightsquigarrow S_1 \otimes S_2$ mit $S, S_1, S_2 \in \mathbb{S}$.

Die Systeme seien beschrieben mit den Formeln:

$$[\![S_1]\!] = \bigwedge_{i\in[1..n]} \Phi_{a_i} \quad , \quad [\![S_2]\!] = \bigwedge_{j\in[1..m]} \Phi_{b_j} \quad , \quad [\![S]\!] = \bigwedge_{k\in[1..l]} \Phi_{c_k}$$

Eine Funktionsabhängigkeit zwischen zwei Blackbox-Spezifikationen ist gegeben, wenn gilt:

$$\Phi_{a_x} \overset{FKT}{\dashrightarrow} \Phi_{c_y} \overset{def}{=} \exists I \subseteq [1..n], J \subseteq [1..m] : \\ (\bigwedge_{i\in I} \Phi_{a_i} \wedge \bigwedge_{j\in J} \Phi_{b_j} \Rightarrow \Phi_{c_y}) \wedge \neg(\bigwedge_{i\in I\setminus\{x\}} \Phi_{a_i} \wedge \bigwedge_{j\in J} \Phi_{b_j} \Rightarrow \Phi_{c_y})$$

mit $x \in [1..n]$ und $y \in [1..l]$ ⌟

Modifikationsbezug

Die Besonderheit der obigen Definition ist, dass sich die modellierte Abhängigkeit nicht auf Modifikationen, sondern auf die Prädikate zu den Systemen bezieht. Die Definition wurde hier so gestaltet, um ein Funktionsnetz bereits dann aufbauen zu können, wenn noch keine Fehler definiert worden sind. Die definierte Beziehung kann alternativ mit Modifikationen definiert werden, in dem die ausgelassenen Teilprädikate mit der konjunktiv eingeschränkten bayesschen Modifikation (false,true) modifiziert werden. Die Definition der Funktionsabhängigkeit ergibt sich so zu:

Theorem 5.2.2 (Funktionsabhängigkeit (modifikationsbasiert))
Gegeben sei das System aus Definition 5.2.2. Zusätzlich sei zu jedem Teilprädikat eine bayesche Modifikation $\mathcal{M}^B$ gegeben. Eine Funktionsabhängigkeit zwischen zwei Teilprädikaten ist gegeben, wenn für deren bayesschen Modifikationen gilt:

✍ *Anmerkung*: Zur einfacheren Lesbarkeit wurden die Klammern $[\![]\!]$ weggelassen und bei der Modifikationskombination + nur die Indizes x statt Φ_x angegeben.

Sei $x \in [a_1..a_n]$, $y \in [k_1..k_l]$ und $L = [k_1..k_l]$
dann gilt:

$$\begin{aligned}
&\Phi_x \overset{FKT}{\dashrightarrow} \Phi_y \\
&\quad = \exists I \subseteq [a_1..a_n], J \subseteq [b_1..b_m] : \\
&\qquad (\bigwedge_{i\in I} \Phi_i \wedge \bigwedge_{j\in J} \Phi_j \Rightarrow \Phi_y) \wedge \neg(\bigwedge_{i\in I\setminus\{x\}} \Phi_i \wedge \bigwedge_{j\in J} \Phi_j \Rightarrow \Phi_y) \\
&\quad = \exists \overline{I} \subseteq [a_1..a_n], \overline{J} \subseteq [j_1..j_m] : \qquad |\text{bayessche Mod. zum Präd. streichen} \\
&\qquad (S_1 \underset{i\in\overline{I}}{\Delta^{\Phi_i}} \mathcal{M}_i^B \wedge S_2 \underset{j\in\overline{J}}{\Delta^{\Phi_j}} \mathcal{M}_j^B \Rightarrow S \underset{k\in L\setminus\{y\}}{\Delta^{\Phi_k}} \mathcal{M}_k^B) \\
&\qquad \wedge \neg(S_1 \underset{i\in\overline{I}\cup\{x\}}{\Delta^{\Phi_i}} \mathcal{M}_i^B \wedge S_2 \underset{j\in\overline{J}}{\Delta^{\Phi_j}} \mathcal{M}_j^B \Rightarrow [\![S]\!] \underset{k\in L\setminus\{y\}}{\Delta^{\Phi_k}} \mathcal{M}_k^B) \\
&\quad = \exists \overline{I} \subseteq [a_1..a_n], \overline{J} \subseteq [j_1..j_m] : \qquad |\text{bayessche Mod. erweitert } R_S \\
&\qquad (\mathcal{M}^\emptyset +^{I,\overline{I}} \sum_{i\in\overline{I}}^{\Phi_i} \mathcal{M}_i^B, \mathcal{M}^\emptyset +^{J,\overline{J}} \sum_{j\in\overline{J}}^{\Phi_j} \mathcal{M}_j^{Bayes}) \overset{\leq}{\dashrightarrow} (\mathcal{M}^\emptyset +^{y,L\setminus y} \sum_{k\in L\setminus\{y\}}^{\Phi_k} \mathcal{M}_k^B) \\
&\qquad \wedge \neg(\mathcal{M}^\emptyset +^{I\setminus x,\overline{I}\cup x} \sum_{i\in\overline{I}\cup\{x\}}^{\Phi_i} \mathcal{M}_i^B, \mathcal{M}^\emptyset +^{\overline{J},J} \sum_{j\in J}^{\Phi_j} \mathcal{M}_j^B) \overset{\leq}{\dashrightarrow} (\mathcal{M}^\emptyset +^{y,L\setminus y} \sum_{k\in L\setminus\{y\}}^{\Phi_k} \mathcal{M}_k^B) \\
&\quad \overset{def}{=} [\![\mathcal{M}_x^B]\!] \overset{FKT}{\dashrightarrow} [\![\mathcal{M}_y^B]\!]
\end{aligned}$$ ⌟

Mit der anforderungsbezogenen Funktionsabhängigkeit und der Einfachfehler-Abhängigkeit ist formal definiert, welche Bedeutung hinter einer Verbindung zwischen zwei Fehlern bzw. Anforderungen in einem FMEA-Dokument (Abschnitt 2.3) steht. Liegen die Anforderungen und Modifikationen formal vor, so können die Zusammenhänge auf Basis der gegebenen Definitionen formal verifiziert werden. Techniken hinsichtlich der Verifikation mit Werkzeugen sind in Kapitel 6 zu finden. *Anwendung*

5.2.1.5 Obere Schranke für Mehrfachfehlerabhängigkeit

Um die Hülle für Folgefehler weiter einzugrenzen, gibt es neben dem Funktionsnetz, bei dem eine beliebige Abweichung an einem Kanal angenommen wird, die Möglichkeit, nur bestimmte Abweichungen anzunehmen. Damit kann bestimmt werden, ob ein Fehler an einem Kanal auch in Verbindung mit Fehlern an anderen Kanälen zu einem bestimmten Folgefehler beitragen kann. Die Ermittlung, ob eine Funktionsabhängigkeit bzw. eine obere Schranke existiert, ist von der Definition her komplexer als die Definition der Einfachfehlerabhängigkeit. Durch die Zielsetzung, die Wirkung zusammen mit anderen Fehlern zu bestimmen, ist zusätzlich zu prüfen, ob ein Fehler überhaupt durch den hier als Ursache angegebenen Fehler verursacht wird. *obere Schranke*

Die Definition dieser Art des Fehlerzusammenhangs findet in der Praxis keine Anwendung, da sie kompliziert zu ermitteln ist und dafür einen geringen Nutzen bringt. Trotzdem wird diese Art des Zusammenhangs vorgestellt, da hier deutlich wird, dass informell einfach zu beschreibende Zusammenhangsbilder zwischen Fehlern formal schwierig zu interpretieren sind. Leser, die ihren Fokus auf den anwendbaren Modellen haben, wird vorgeschlagen, diesen Abschnitt zu überspringen. *Anwendung*

Die Überprüfung entspricht grundlegend der Funktionsabhängigkeitsüberprüfung. Es wird verglichen, ob es eine ergänzende Modifikationskombination gibt, so dass zum Einen die zugehörige Anforderung der Folgemodifikation ohne die Ursache eingehalten werden kann und zum Anderen mit der Ursache verletzt wird. Zusätzlich kann die zu untersuchende Folgemodifikation kleiner der bayesschen Modifikation, also der maximalen Modifikation, sein. Aus diesem Grund wird die Definition um zwei Punkte erweitert: *formale Definition*
(a) Die kleinere obere Schranke muss immer noch größer als die Folgemodifikation sein. Dies wird in Teil (4) der unten stehenden Definition berücksichtigt.
(b) Die obere Schranke muss für jede mögliche Ergänzung gelten. Dass heißt, eine obere Schranke wird nur akzeptiert, wenn es keine größere Folge einer anderen Ergänzung gibt. Dies wird in der unten stehenden Definition in Teil (1) bestimmt.

Definition 5.2.3 (obere Schranke für Mehrfachfehlerabhängigkeit)
Gegeben sei ein System $S = S1 \otimes S2$ mit $S, S1, S2 \in \mathbb{S}$.

Die Systeme seien beschrieben mit den disjunkten Prädikaten $[\![S]\!] = \Phi_{S\_a} \wedge \Phi_{S\_b} \wedge ...$ und $[\![S1]\!] = \Phi_{S1\_a} \wedge \Phi_{S1\_b} \wedge ...$

Die disjunkte umfassende Abhängigkeit zwischen zwei Fehlern $[\![\mathcal{M}_{S1}^{(1)}]\!]$ und $[\![\mathcal{M}_{S}^{(1)}]\!]$ ist gegeben, wenn gilt:

1.) Maximalität

Für jede mögliche Ergänzung mit Fehlern muss der Folgefehler eine obere Schranke sein.

$\overset{||^n\leq}{\dashrightarrow}: PRED_{MOD_{S1}} \times PRED_{MOD_S} \to BOOL$

$[\![\mathcal{M}_{S1}^{(1)}]\!] \overset{||^n\leq}{\dashrightarrow} [\![\mathcal{M}_S^{(1)}]\!] \overset{def}{\Leftrightarrow}$

$(\mathsf{causeSet}([\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_S^{(1)}]\!]) \neq \emptyset \wedge \forall [\![\mathcal{M}_{S1}^{(x)}]\!] :$

$([\![\mathcal{M}_{S1}^{(x)}]\!] \rightsquigarrow [\![\mathcal{M}_{S1}^{(1)}]\!]) \Rightarrow (\mathsf{causeSet}([\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_S^{(x)}]\!]) \subseteq \mathsf{causeSet}([\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_S^{(1)}]\!]))$

$\vee (\mathsf{causeSet}([\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_S^{Bayes}]\!]) = \emptyset)$

2.) Ergänzungsmenge

Die Menge der möglichen Ergänzungen mit anderen Modifikationen, so dass die Abhängigkeitsbeziehungen erfüllt sind.

$causeSet : PRED_{MOD_{S1}} \times PRED_{MOD_S} \to \mathcal{P}((PRED_{MOD_{S1}})^2)$

$\mathsf{causeSet}([\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_S^{(1)}]\!]) \overset{def}{=}$

$\{([\![\mathcal{M}_{S1}^{(a)}]\!], [\![\mathcal{M}_{S1}^{(a')}]\!]) |$

$[\![\mathcal{M}_{S1}^{(a)}]\!] = \sum^{\langle \Phi_{S1\_a}, \Phi_{S1\_b}, \ldots \rangle} \langle [\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_{S1}^{(2)}]\!], \ldots \rangle \wedge [\![\mathcal{M}_{S1}^{(1)}]\!], [\![\mathcal{M}_{S1}^{(2)}]\!], \ldots \in MOD_{S1}$

$\wedge [\![\mathcal{M}_{S1}^{(a')}]\!] = \sum^{\langle \Phi_{S1\_a}, \Phi_{S1\_b}, \ldots \rangle} \langle [\![\mathcal{M}_{S1}^{\emptyset}]\!], [\![\mathcal{M}_{S1}^{(2)}]\!], \ldots \rangle \wedge [\![\mathcal{M}_{S1}^{(2)}]\!], \ldots \in MOD_{S1}$

$\wedge \mathsf{existsConsequence}([\![\mathcal{M}_{S1}^{(a)}]\!], [\![\mathcal{M}_{S1}^{(a')}]\!], [\![\mathcal{M}_S^{(1)}]\!])\}$

3.) Ergänzung der Folge

Überprüfe, ob es eine geeignete Ergänzung für die Folge gibt, so dass die Bedingungen wahr sind.

$existsConsequence : (PRED_{MOD_{S1}})^2 \times PRED_{MOD_S} \to BOOL$

$\mathsf{existsConsequence}([\![\mathcal{M}_{S1}^{(a)}]\!], [\![\mathcal{M}_{S1}^{(a')}]\!], [\![\mathcal{M}_S^{(1)}]\!]) \overset{def}{\Leftrightarrow}$

$\exists [\![\mathcal{M}_S^{(a)}]\!], [\![\mathcal{M}_S^{(a')}]\!], [\![\mathcal{M}_S^{(1')}]\!] : [\![\mathcal{M}_S^{(1)}]\!] \rightsquigarrow [\![\mathcal{M}_S^{(1')}]\!]$

$\wedge [\![\mathcal{M}_S^{(a)}]\!] = \sum^{\langle \Phi_{S\_a}, \Phi_{S\_b}, \ldots \rangle} \langle [\![\mathcal{M}_S^{(1')}]\!], [\![\mathcal{M}_S^{(2)}]\!], \ldots \rangle \wedge [\![\mathcal{M}_S^{(1)}]\!], [\![\mathcal{M}_S^{(2)}]\!], \ldots \in MOD_S$

$\wedge [\![\mathcal{M}_S^{(a')}]\!] = \sum^{\langle \Phi_{S\_a}, \Phi_{S\_b}, \ldots \rangle} \langle [\![\mathcal{M}_S^{\emptyset}]\!], [\![\mathcal{M}_S^{(2)}]\!], \ldots \rangle \wedge [\![\mathcal{M}_S^{(2)}]\!], \ldots \in MOD_S$

$\wedge \mathsf{isEssential}([\![\mathcal{M}_{S1}^{(a)}]\!], [\![\mathcal{M}_{S1}^{(a')}]\!], [\![\mathcal{M}_S^{(a)}]\!], [\![\mathcal{M}_S^{(a')}]\!])$

4.) Bedingungen

$\mathsf{isEssential} : (PRED_{MOD_{S1}})^2 \times (PRED_{MOD_S})^2 \to BOOL$

$\mathsf{isEssential}([\![\mathcal{M}_{S1}^{(a)}]\!], [\![\mathcal{M}_{S1}^{(a')}]\!], [\![\mathcal{M}_S^{(a)}]\!], [\![\mathcal{M}_S^{(a')}]\!]) \overset{def}{\Leftrightarrow}$

$[\![\mathcal{M}_{S1}^{(a)}]\!] \overset{\leq}{\dashrightarrow} [\![\mathcal{M}_S^{(a)}]\!]$ |(I.) Es gibt übergreifenden Folgefehler mit$[\![\mathcal{M}_S^{(1)}]\!]$

$\wedge [\![\mathcal{M}_{S1}^{(a)}]\!] \overset{\leq}{\not\dashrightarrow} [\![\mathcal{M}_S^{(a')}]\!]$ |(II.) an dem$[\![\mathcal{M}_S^{(1)}]\!]$wesentlicher Bestandteil ist

$\wedge [\![\mathcal{M}_{S1}^{(a')}]\!] \overset{\leq}{\dashrightarrow} [\![\mathcal{M}_S^{(a')}]\!]$ |(III.) und auch$[\![\mathcal{M}_{S1}^{(1)}]\!]$wesentlicher Bestandteil ist.

⌟

Die in obiger Definition enthaltenen Teile sind nicht zwingend trivial zu verstehen. Im Folgenden werden anhand eines Beispiels 5.2.4 diese Bedingungen verdeutlicht.

Beispiel 5.2.4 (obere Schranke für Mehrfachfehlerabhängigkeit)
Gegeben sei das boolesche System aus Abbildung 5.5, in dem ein Signal redundant berechnet wird (Die Berechnung wird hier mit der Identität repräsentiert). Für die jeweiligen Kanäle x gebe es die Modifikationen
$[\![\mathcal{M}_x^{TsF}]\!] = (true, [\![F_x^{TsF}]\!])$ und $[\![\mathcal{M}_x^{FsT}]\!] = (true, [\![F_x^{FsT}]\!])$ (- siehe Abschnitt 4.2.7).

Im Folgenden werden einige Abhängigkeiten aus dem System erklärt, um die drei Bedingungen der Definition der oberen Schranke für Mehrfachfehlerabhängigkeit zu erklären.

(1.) $[\![\mathcal{M}_d^{TsF}]\!] \overset{||^n \leq}{\not\dashrightarrow} [\![\mathcal{M}_g^{FsT}]\!]$

Kein Folgefehler der $[\![\mathcal{M}_g^{FsT}]\!]$ oder Teilmodifikationen davon enthält lässt sich aus einem Fehler der exakt $[\![\mathcal{M}_d^{TsF}]\!]$ enthält ableiten, da ein TsF bei der Identität in dem Folgefehler vorkommen muss.

(2.) $[\![\mathcal{M}_c^{TsF}]\!] \overset{||^n \leq}{\dashrightarrow} [\![\mathcal{M}_g^{\emptyset}]\!]$

Kein Fehler, der $[\![\mathcal{M}_c^{TsF}]\!]$ enthält, hat zwingend einen Fehler zur Folge, der $[\![\mathcal{M}_g^{TsF}]\!]$ enthält. Diese Aussage resultiert aus zwei zu prüfenden Fällen:

(2.a.) Der ergänzende Fehler am Kanal d ist $[\![\mathcal{M}_d^{\emptyset}]\!]$

In diesem Fall gilt:
$[\![\mathcal{M}_c^{TsF}]\!] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{\emptyset}]\!] \dashrightarrow (true, [\![F_e^{TsF}]\!] \wedge [\![F_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{\emptyset}]\!]$

und damit die kleinste Folge für die Bedingung (I):
$[\![\mathcal{M}_c^{TsF}]\!] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{\emptyset}]\!] \overset{\leq}{\dashrightarrow} ([\![\mathcal{M}_e^{TsF}]\!] +^{\Phi_e,\Phi_f} [\![\mathcal{M}_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{\emptyset}]\!]$

Allerdings kann hier für jede $[\![\mathcal{M}_g^{xxx}]\!]$ eine Änderung zu $[\![\mathcal{M}_g^{\emptyset}]\!]$ nicht zu einem Fehler führen, der kleiner als der Soll-Folgefehler ist. Die Bedingung (II) ist so nicht zu erfüllen.

(2.b.) Der ergänzende Fehler am Kanal d ist ungleich $[\![\mathcal{M}_d^{\emptyset}]\!]$.

In diesem Fall gilt:
$[\![\mathcal{M}_c^{TsF}]\!] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{xxx}]\!] \dashrightarrow (true, [\![F_e^{TsF}]\!] \wedge [\![F_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{xxx}]\!]$

und damit für alle $[\![\mathcal{M}_g^{yyy}]\!] \geq [\![\mathcal{M}_g^{xxx}]\!]$ die Bedingung (I):
$[\![\mathcal{M}_c^{TsF}]\!] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{xxx}]\!] \overset{\leq}{\dashrightarrow} ([\![\mathcal{M}_e^{TsF}]\!] +^{\Phi_e,\Phi_f} [\![\mathcal{M}_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{yyy}]\!]$

Ersetzt man $[\![\mathcal{M}_g^{yyy}]\!]$ durch die leere Modifikation, so gilt für jedes $[\![\mathcal{M}_d^{xxx}]\!]$:
$[\![\mathcal{M}_c^{TsF}]\!]] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{xxx}]\!] \overset{<}{\not\dashrightarrow} ([\![\mathcal{M}_e^{TsF}]\!] +^{\Phi_e,\Phi_f} [\![\mathcal{M}_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{\emptyset}]\!]$

Allerdings ist die Modifikation $\mathcal{M}_c^{TsF}$ für jede Modifikation $[\![\mathcal{M}_g^{yyy}]\!]$ nicht wesentlich und somit die Bedingung (III) nie erfüllt:
$[\![\mathcal{M}_c^{\emptyset}]\!] +^{\Phi_c,\Phi_d} [\![\mathcal{M}_d^{xxx}]\!] \overset{\leq}{\not\dashrightarrow} ([\![\mathcal{M}_e^{TsF}]\!] +^{\Phi_e,\Phi_f} [\![\mathcal{M}_f^{TsF}]\!]) +^{\Phi_e \wedge \Phi_f, \Phi_g} [\![\mathcal{M}_g^{\emptyset}]\!]$

(3.) $[\![\mathcal{M}_e^{TsF}]\!] \overset{||^n \leq}{\dashrightarrow} [\![\mathcal{M}_h^{TsF}]\!]$

Die Modifikation $[\![\mathcal{M}_e^{TsF}]\!]$ hat eine Modifikation $[\![\mathcal{M}_h^{TsF}]\!]$ zur Folge, da hier die Bedingungen (I), (II) und (III) erfüllt sind:

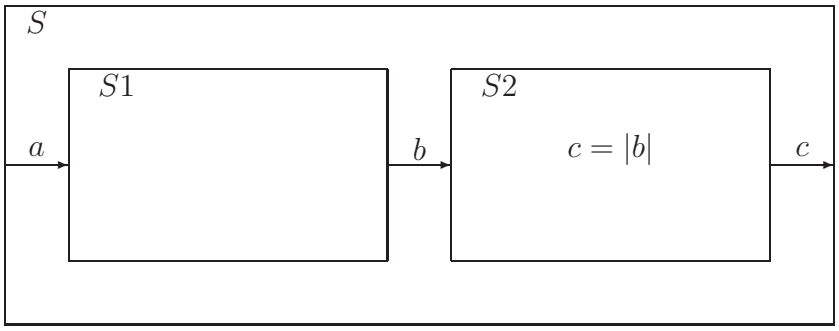


Abb. 5.6: Beispiel eines Systems mit reellen Werten

$$(\llbracket\mathcal{M}_e^{TsF}\rrbracket +^{\Phi_e,\Phi_f} \llbracket\mathcal{M}_f^{TsF}\rrbracket) +^{\Phi_e\wedge\Phi_f,\Phi_g} \llbracket\mathcal{M}_g^{\emptyset}\rrbracket \overset{\leq}{\dashrightarrow} (\llbracket\mathcal{M}_h^{TsF}\rrbracket +^{\Phi_h,\Phi_i} \llbracket\mathcal{M}_i^{TsF}\rrbracket)$$
$$(\llbracket\mathcal{M}_e^{TsF}\rrbracket +^{\Phi_e,\Phi_f} \llbracket\mathcal{M}_f^{TsF}\rrbracket) +^{\Phi_e\wedge\Phi_f,\Phi_g} \llbracket\mathcal{M}_g^{\emptyset}\rrbracket \overset{<}{\not\dashrightarrow} (\llbracket\mathcal{M}_h^{\emptyset}\rrbracket +^{\Phi_h,\Phi_i} \llbracket\mathcal{M}_i^{TsF}\rrbracket)$$
$$(\llbracket\mathcal{M}_e^{\emptyset}\rrbracket +^{\Phi_e,\Phi_f} \llbracket\mathcal{M}_f^{TsF}\rrbracket) +^{\Phi_e\wedge\Phi_f,\Phi_g} \llbracket\mathcal{M}_g^{\emptyset}\rrbracket \overset{\leq}{\dashrightarrow} (\llbracket\mathcal{M}_h^{\emptyset}\rrbracket +^{\Phi_h,\Phi_i} \llbracket\mathcal{M}_i^{TsF}\rrbracket)$$ ⌟

Interpretation

Die Schwierigkeit dieser Definition eines Zusammenhangs ist, dass wenn keine Funktionsabhängigkeit vorhanden ist, auch keine Fehlerabhängigkeit angegeben werden kann. Umgekehrt kann, wenn eine funktionale Abhängigkeit existiert, immer eine Fehlerabhängigkeit angegeben werden. Zumindest die leere Modifikation kann dann als Folge angegeben werden. Dieser Zusammenhang stellt also nicht nur eine obere Schranke dar, sondern prüft gleichzeitig, ob überhaupt Effekte auf eine Anforderung bestehen können. Folgendes Beispiel 5.2.5 zeigt die Darstellung eines Fehlers ohne Folgefehler:

Beispiel 5.2.5 (Zusammenhang im Funktionsnetz)
Gegeben sei das System $S \rightsquigarrow S1 \otimes S2$ aus Abbildung 5.6, bei dem die Kanäle kontinuierliche Werte haben. Weiter sei eine Modifikation $\llbracket\mathcal{M}_b^{Sign}\rrbracket = (true, \llbracket F_b^{Sign}\rrbracket)$ gegeben, bei der das Vorzeichen geändert wird.

Diese Modifikation $\llbracket\mathcal{M}_b^{Sign}\rrbracket$ hat auf den Kanal c keine Wirkung, da die Betragsfunktion einen Folgefehler verhindert. Entsprechend gilt bei allen nichtleeren Modifikationen $\llbracket\mathcal{M}_c^{xxx}\rrbracket$ die Bedingung (II) nicht und damit:

$$\llbracket\mathcal{M}_b^{Sign}\rrbracket \overset{||^n\leq}{\dashrightarrow} \llbracket\mathcal{M}_c^{\emptyset}\rrbracket$$

Alle nichtleeren Modifikationen $\llbracket\mathcal{M}_c^{xxx}\rrbracket$ können ebenfalls als obere Schranke angegeben werden, also es gilt für alle $\llbracket\mathcal{M}_c^{xxx}\rrbracket$:

$$\llbracket\mathcal{M}_b^{Sign}\rrbracket \overset{||^n\leq}{\dashrightarrow} \llbracket\mathcal{M}_c^{xxx}\rrbracket$$ ⌟

Schranke

An diesem Beispiel und den Definitionen dieses Abschnitt ist zu erkennen, dass sich die Interpretation eines Fehlerzusammenhangs auf die klaren Beziehungen der Teilmodifikationen beschränken sollten, da andere Definitionen mit dem gegebenen

Zusammenhangsmodell schwer vereinbar sind. Die folgenden Abschnitte befassen sich deshalb mit Mechanismen, eine obere oder untere Schranke für Folgefehler abzuschätzen.

5.2.2 Zusammenhänge bei aufgelösten totalen Gleichungen

Abweichung

Die aufgelösten totalen Gleichungen sind eine Sonderform der Verhaltensanforderungen und als solche können auf diese alle im vorhergehenden Abschnitt vorgestellten Zusammenhangsmodelle und Methoden angewendet werden. Durch die besondere Form der Gleichungen kann allerdings auch der Signalfluss und damit die Abweichungsmodelle angewendet werden. Die in Abschnitt 5.1.2 genannte Herausforderung liegt darin, über die gesamte (theoretisch unendliche) Laufzeit des Systems Aussagen über die Folgefehlverhalten zu erstellen. Der Umgang kann prinzipiell durch verschiedene Mechanismen erreicht werden, wobei es immer das Ziel ist, auf einen in endlicher (akzeptabler) Zeit berechenbaren Variablen-/Zustandsraum zu kommen. Für einfache diskrete Systeme mit kleinem Variablenraum funktioniert häufig Symbolic Modell Checking Modulo Theories (siehe Kapitel 6). Ein möglicher Ansatz, die Zeit in einem endlichen Zustandsraum zu fassen, ist die Zeitpunkte zu Intervallen zu abstrahieren. Ein so angepasstes Verhaltensmodell z.B. kann mit ITL beschrieben werden und dann mit einem symbolischen Modell Checker überprüft werden (siehe [Thu04]). Dieser Abschnitt zeigt drei Mechanismen, die komplexen zeitlichen Zusammenhänge zu abstrahieren. Mit diesen Mechanismen lassen sich die Folgefehler insbesondere auch vom Menschen einfacher abschätzen. Abschnitt 5.2.2.1 beschreibt die komplette Abstraktion der zeitlichen Zusammenhänge, Abschnitt 5.2.2.2 die Abstraktion der zeitlichen Vorgeschichte zu einem Beobachtungsintervall und schließlich Abschnitt 5.2.2.3 die Abstraktion der zeitlichen Kombination verschiedener Teilfehler als obere Schranke für eine Wirkung.

5.2.2.1 Abstraktion der Zeit

Zeit

Dieser Abschnitt beschreibt die Einschränkung der Kombinatorik in der Dimension Zeit nach Punkt *(III)* aus Abschnitt 5.1.2. Diese Einschränkung bedeutet nicht, dass die Fehlverhalten zeitlos sein müssen, wie z.B. bei den zeitlosen Wertabweichungen in Abschnitt 4.2.2. Wesentlich ist, dass zeitliche Zusammenhänge nicht betrachtet werden. Als Folge dieser Abstraktion kann eine Wirkung unter Umständen nicht mehr exakt angegeben werden, da sie sich durch Rückkopplung zu bestimmten Zeitpunkten unbeobachtet potenzieren oder einschränken kann. Diese Abstraktion wird fast immer mit dem Mechanismus *(IV)* aus Abschnitt 5.1.2 kombiniert, in dem für die Folgen jeweils geprüft wird, ob die Beziehung $\overset{\leq}{\dashrightarrow}$ erfüllt ist.

Stufen

In dieser Arbeit werden zwei Stufen der Abstraktion näher betrachtet. In der kleinen Stufe wird die kausale Monotonie abstrahiert und in der stärkeren Stufe werden die kompletten zeitlichen Zusammenhänge zwischen Fehlern abstrahiert. Diese beiden Stufen werden im Folgenden beschrieben.

kausale Monotonie

Vernachlässigt man die kausale Monotonie, so erfolgt die Ermittlung einer Folge weiterhin genauso iterativ, wie in Definition 5.1.5. Allerdings muss die Funktion reverse nicht mehr beachtet werden, was zu einem übergreifenden Folgefehlverhalten führt. Dieses zeichnet sich dadurch aus, dass in jedem Rechenschritt bei nichtdeterministischem Verhalten zwischen jeder Alternative gesprungen werden kann. Es wird so also eine Hülle der erreichbaren Werte zu den jeweiligen Zeitpunkten ermittelt. Theoretisch müssen für einen unendlichen Strom immer noch unendlich viele Iterationen durchgegangen werden. Es ist jedoch zu jedem Zeitpunkt t deutlich einfacher, die maximale Menge möglicher Zustände zu ermitteln. Folgendes Beispiel verdeutlicht diese Abstraktion:

Beispiel 5.2.6 (Abstraktion der kausalen Monotonie)
In Beispiel 5.1.1 wurde für einen einfachen Fehler die Fehlerfolge ermittelt. Dazu wurde einmal die Folge ohne und mit Berücksichtigung der kausalen Monotonie betrachtet. Mit Berücksichtigung der kausalen Monotonie ergaben sich monoton steigende Abweichungen vom Sollwert:

$$\delta^{(1)}_{O_{S\_1}} = \begin{cases} \{\langle 0,0\rangle, \langle 0,1\rangle\} & \text{falls } d = \langle 0,0\rangle \\ \{\langle 1,1\rangle, \langle 1,2\rangle\} & \text{falls } d = \langle 0,1\rangle \end{cases}$$

Ohne Monotonie ergaben sich nach der zweiten Iteration die Folgen:

$$\delta^{1\mapsto+2}_{O_{S\_1}} \Rightarrow (\delta^{(1)}_{I_{O\_1}} = \begin{cases} \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\} & \text{falls } d = \langle 0,0\rangle \\ \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,1\rangle, \langle 1,2\rangle\} & \text{falls } d = \langle 0,1\rangle \end{cases})$$

Beim Vergleich beider Ergebnisse erkennt man, dass in Summe jeder zu jedem Zeitpunkt mögliche Wert mit jedem anderen kombiniert wird. Es hat zwar die obere und untere Schranke für die erreichbaren Werte die Charakteristik eines Drifts, nicht aber alle im Fehler enthaltenen Ströme. Schließt man so auf einen Fixpunkt, so ergibt sich:

$$[\![\delta^{driftBound}_{O_{S\_1}}]\!]^{op} = ((0 \leq c - \tau_c \leq t) \wedge (t = 1 + \mathsf{delay}_{\langle 0\rangle}(t)))$$ ⌟

obere Schranke

Diese Art der Abstraktion eignet sich besonders für Worst-Case-Abschätzungen, bei denen überprüft wird, ob bestimmte Zustände bzw. Ausgabewerte erreicht werden können. Die Menge der erreichbaren Ausgabewerte ist zu jedem Zeitpunkt t angegeben und so ist auch hinsichtlich der F-Mengen der Modifikationen eine Fehlerpotenzierung mit berücksichtigt. Lediglich die Aufhebung von Fehlern kann durch Aussetzen der Kombinatorik verloren gehen. Hinsichtlich der E-Mengen werden umgekehrt alle Fehleraufhebungen mit berücksichtigt und es kann so eine Mindestwirkung ermittelt werden.

Zeitabstraktion

Die zweite Stufe der Abstraktion ist die komplette Vernachlässigung zeitlicher Zusammenhänge. Bei jedem Fehlerbild wird davon ausgegangen, dass es irgendwann auftreten kann. Die Fehlerbilder selbst können immer noch zeitbezogene Verhalten haben, aber die Ableitung anderer zeitbezogener Fehlerbilder ist meist, wie in folgendem Beispiel zu sehen, nicht möglich:

Beispiel 5.2.7 (Abstraktion der Zeit)
In Beispiel 5.2.6 wurde gezeigt, wie die maximal zu erreichenden Werte abgeschätzt werden können. Um diese zu erreichenden Werte zu ermitteln, wurde die Rückkopplung und damit die Entwicklung über die Zeit betrachtet. In dem Auswirkungsmodell des Systems S_2 wird durch die Zeitverzögerung das Fehlerbild so abgebildet, dass es erst zum x-ten Zeitschritt eintreten kann. Abstrahiert man diese zeitliche Einschränkung durch den Verlauf der Zeit, so bleibt der Fehler in diesem Fall in der Komponente gleich, also:

$$(\delta^{+1}_{O_{S\_1}}, \overset{\geq}{\dashrightarrow}, \delta^{+1}_{I_{S\_1}})$$

Damit werden ohne Zeitbezug die maximal zu erreichenden Wertabweichungen ermittelt. Der sich daraus ergebende Fixpunkt ist:

$$[\![\delta^{+\infty}_{O_{S\_1}}]\!]^{op} = (0 \leq c - \tau_c \leq \infty)$$ ⌟

Anwendung

Durch Vernachlässigen der Zeit wird die Abschätzung sehr grob. Allerdings kann damit trotzdem noch eine Menge spezifischer Fehlerbilder erfasst werden. Gerade bei einfachen Anforderungen an Regelfunktionen ist diese Betrachtung ausreichend. So ist zum Beispiel in der Fallstudie zu überprüfen, ob eine Ungenauigkeit des Sensors für den Lenkradwinkel zu einer Ungenauigkeit des zu berechnenden Vorderradwinkels führen kann.

5.2.2.2 Modelle für begrenzte Zeitspannen

Zeitfenster

Möchte man die zeitlichen Zusammenhänge zwischen Fehlern nicht abstrahieren sondern exakt erfassen, so ist eine Alternative, die Zusammenhangsmodellierung aus der Betrachtung begrenzter Zeitspannen heraus abzuschätzen. Die Wirkung eines Fehlers wird nur über einen begrenzten Zeitraum hin angegeben und entsprechend kann als Fehlerfolge nur eine Übermenge der Folgefehler angegeben werden. Die Folgefehler werden innerhalb des Betrachtungszeitraumes mit der genauen Folge beschrieben. Nach und vor dem Betrachtungszeitraum ist jedoch ein beliebiges Verhalten denkbar, welches nicht überprüft wurde. Die Abschätzung findet dadurch statt, dass das Zeitfenster einmal über die komplette Betriebszeit geschoben wird. Die in einer derart beschränkten Überwachung zu verbindenden Fehler haben meist kein über längere Zeiträume zu beobachtendes Verhalten, da nach der Beobachtungszeit bereits klar sein muss, ob ein Folgefehler eintritt.

Abschätzung

Die Abschätzung von Fehlerzusammenhängen über Zeitfenster wird im Folgenden beschrieben, wobei Abbildung 5.7 als Beispiel die Erklärungen verdeutlicht. Prinzipiell bedeutet die Abschätzung über eine Zeitspanne, dass nicht mehr die unendlichen Ströme betrachtet werden, sondern nur begrenzte Zeitspannen. Eine exakte Aussage über den Folgefehler kann getroffen werden, wenn für jeden Zeitpunkt t betrachtet wird, in welchem Zustand sich das System bereits befinden kann, und welcher Folgefehler im nächsten Rechenschritt möglich ist. In Abbildung 5.7 ist die fehlerhafte Abweichung als graue gestrichelte Linie modelliert, welche einen exemplarischen Ablauf eines Drifts widerspiegelt. Die daraus resultierende Wirkung wird durch das

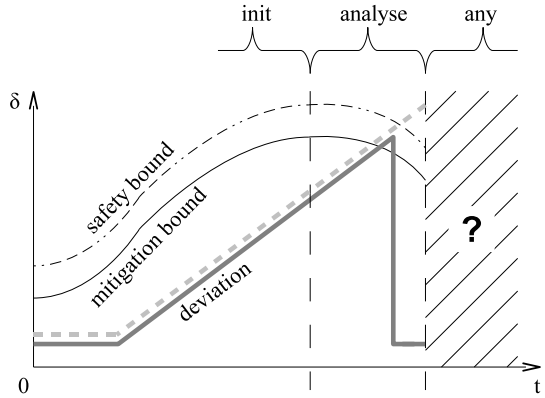


Abb. 5.7: Betrachtung eines Zeitfensters

System S begrenzt, wodurch die Ausgabe (graue Linie) nach Überschreitung einer Eingriffsgrenze (schwarze dünne Linie) zurückgesetzt wird. Damit kann die Abweichung der Ausgabe nie die Sicherheitsgrenze (dünne gestrichelte Linie) überschreiten. Graphisch kann man sich nun vorstellen, mit dem Analyse-Zeitfenster immer entlang der t-Achse zu fahren und für jeden Zeitpunkt zu bestimmen, in welchem Zustand sich das System und die Modifikation gerade befinden kann und welche Folge dann maximal möglich ist.

$$\forall t : [\![\mathcal{M}_{S_1}]\!]^{op} \overset{\leq}{\dashrightarrow} [\![\mathcal{M}_S]\!]^{op}$$

Startzustände Da die Anzahl der Zeitpunkte t bei reaktiven Systemen meist unendlich ist, kann statt dessen die Menge der erreichbaren Zustände eines Systems zu Beginn des Analysefensters verwendet werden. Diese kann zwar auch unendlich sein, lässt aber zumindest schon einmal die Verwendung zeitlich endlicher Analysen zu. Allerdings gibt es hier ein Paradoxon: Um die Menge der in unendlicher Zeit erreichbaren Zustände zu ermitteln, muss die Ermittlung unendlich oft stattfinden. Hier kommt dann wieder ein Abschätzungsmechanismus ins Spiel. Die größte Worst-Case-Abschätzung ist gegeben durch die Annahme, dass zu Beginn der Analysespanne jeder Zustand möglich ist. Für einfache Fehler kann so bereits relativ exakt eine Fehlerfolge ermittelt werden, wie in folgendem Beispiel dargestellt wird.

Beispiel 5.2.8 (Worst-Case-Zeitfenster)
In Beispiel 5.1.1 wird der Zustand des Systems mit $\mathsf{delay}_{\langle 0 \rangle}(c)$ festgehalten. Setzt man nun für den Zustand einen beliebigen Wert $x.t$ ein und setzt man dann für die aktuelle Abweichung eine beliebige Differenz $c.t - \tau_c.t$ ein, so ergibt sich für $c.(t+1) - \tau_c.(t+1)$, dass das Maximum um eines größer als zuvor sein kann. Daraus ergibt sich:

$$[\![\delta^{drift}_{O_{S\_1}}]\!]^{op} = (0 \leq (c - \tau_c - \mathsf{delay}_{\langle x \rangle}(c) + \mathsf{delay}_{\langle y \rangle}(\tau_c)) \leq 1) \qquad \lrcorner$$

Die Annahme, dass ein System zu Beginn einer Periode einen beliebigen Zustand haben kann, führt meist zu einer Übermenge der möglichen Folgen. Eingeschränkt kann dies werden, indem vor der Analysespanne noch eine Initialisierungsphase stattfindet, in der die Wirkung der Fehler noch nicht beurteilt wird. In dieser Initialisierungsphase kann das System die erreichbaren Zustände eingrenzen. Schneidet die Initialisierungphase den Zeitpunkt $t = 0$, so kann das System mit den Startzuständen initialisiert werden. Verdeutlicht wird dies bei Betrachtung der Abbildung 5.7. Nimmt man für den Ausgabekanal an, dass dieser erst korrigiert wird, wenn die Eingriffsgrenze für 3 Zeitschritte überschritten wurde, so wäre bei einer beliebigen Abweichung der Eingabe für die 2 Zeitschritte der Ausgabe auch ein Wert oberhalb der Sicherheitsgrenze möglich. Da ein Drift aber immer von der Abweichung 0 startet, kann er mit seiner Maximalsteigung in diesem Beispiel die Sicherheitsgrenze nicht verletzen. Um zu diesem Ergebnis zu kommen, ist eine Initialisierungsphase von 2 Zeitschritten notwendig, da nach diesen der Ausgabewert bei nicht erreichbaren Abweichungen bereits korrigiert wurde. Die Worst-Case-Abschätzung wird so schärfer gefasst.

Initialisierung

5.2.2.3 Abstraktion der Stromtreue

Mit den bisher genannten Abstraktionsmechanismen kann die Zeit immer stärker vernachlässigt werden, mit dem Effekt die Folgefehler immer gröber abzuschätzen. Wie in Abschnitt 5.1.1 beschrieben, wird zum Beispiel modelliert, ob ein Folgefehlverhalten vollständig die Konsequenz eines Teilfehlverhaltens ist, also gilt:

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\leq}{\dashrightarrow} \mathcal{M}_S$$

Statt nun das Folgefehlverhalten mit einer einzigen Modifikation zu beschreiben, also als Einfachfehler zu betrachten, wird in der Praxis dieses Fehlverhalten häufig einer Menge von Modifikationen zugeordnet, so dass der zusammengesetzte Fehler größer als das tatsächliche Folgefehlverhalten ist. Bei dieser Zusammensetzung werden lediglich die F-Mengen zusammengesetzt. Die E-Mengen bleiben für alle Teilfehlverhalten gleich:

Menge als Folge

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\leq}{\dashrightarrow} (\mathcal{M}_S^{(1)} + \mathcal{M}_S^{(2)} + \ldots) \text{ mit } \mathcal{M}_S^{(1)} = (E_S, F_S^{(1)}), \mathcal{M}_S^{(2)} = (E_S, F_S^{(2)}), \ldots$$

Erfolgt die Abschätzung durch Vernachlässigen der Zeit, so ist es naheliegend, auch bei der Betrachtung der Zusammensetzung der Teilfehler die Zeit zu abstrahieren. Gerade bei Modifikationen, welche operativ beschrieben sind, ist es meist verführerisch einfach, entsprechend die Teilmodifikationen nicht mit der stromtreuen Modifikationskombination zusammenzusetzen, sondern mit der operativen Modifikationskombination (siehe Abschnitt 4.2.1). Damit ergibt sich eine Modellierung der Form:

Zerlegung

$$([\![\mathcal{M}_{S_1}]\!], [\![\mathcal{M}_{S_2}]\!]) \overset{\leq}{\dashrightarrow} ([\![\mathcal{M}_S^{1'}]\!]^{op} \oplus [\![\mathcal{M}_S^{2'}]\!]^{op} \oplus \ldots)$$
$$\text{mit } [\![\mathcal{M}_S^{1'}]\!]^{op} = ([\![\overline{E}_S]\!]^{op}, [\![F_S^{1'}]\!]^{op}), [\![\mathcal{M}_S^{2'}]\!]^{op} = ([\![\overline{E}_S]\!]^{op}, [\![F_S^{2'}]\!]^{op}), \ldots$$

Abstraktion Betrachtet man nun diese Teilmodifikationen zur weiteren Verfolgung als Fehlerursachen, so müssen zur vollständigen Erfassung der Folgefehlverhalten alle Modifikationen kombiniert werden. Werden Sie jedoch nur einzeln als Ursachen betrachtet, so wird damit implizit angenommen, dass nicht von einem Fehlerbild in ein anderes gewechselt werden kann (siehe Beispiel 4.2.2). Damit wird die tatsächliche Menge an Folgefehlern bei einer Weiterverfolgung kleiner! Folgendes Beispiel verdeutlicht diesen Zusammenhang:

Beispiel 5.2.9 (Abstraktion der Stromtreue)
Das Fehlverhalten eines Systems S sei aus den permanenten Teilfehlern $\mathcal{M}_{O_S}^{TsF} \oplus \mathcal{M}_{O_S}^{FsT}$ zusammengesetzt. Betrachtet man die Folgen der Teilfehler nun einzeln, so wird der Wert entweder auf *true* oder auf $false$ eingefroren. Es wird allerdings nicht betrachtet, welche Wirkung ein Strom hat, der den Ausgabestrom invertiert, also TsF und FsT enthält. Hierzu wäre eine stromtreue Aufteilung notwendig. ⌟

Anwendung Diese Zuordnung der Folgefehlverhalten ist in den Fallstudien öfter vorgefunden worden. Gerade bei Systemen, bei denen keine langen Algorithmen durchlaufen werden, kann diese Abstraktion durchgeführt werden, ohne wesentliche Fehlerfolgen zu übersehen. Eine obere Abschätzung der Fehler ist bei diesem Vorgehen nicht gegeben. Eigentlich gilt also nur noch $\overset{\otimes\leq}{\dashrightarrow}$ was im Wesentlichen keine Aussage über die Stärke der Folge zulässt. Um die Vollständigkeit im Zustandsraum zu modellieren, wird statt dessen der Operator $\overset{\not\leq}{\dashrightarrow}$ eingeführt:

Definition 5.2.4 (Stromuntreue Folge)
$$\begin{aligned}&([\![\mathcal{M}_{S_1}]\!], [\![\mathcal{M}_{S_2}]\!]) \overset{\not\leq}{\dashrightarrow} ([\![\mathcal{M}_S^{(1')}]\!] + [\![\mathcal{M}_S^{(2')}]\!] + \ldots)\\ &\quad \overset{def}{=} ([\![\mathcal{M}_{S_1}]\!], [\![\mathcal{M}_{S_2}]\!]) \overset{\leq}{\dashrightarrow} ([\![\mathcal{M}_S^{(1')}]\!] \oplus [\![\mathcal{M}_S^{(2')}]\!] \oplus \ldots)\\ &\quad \text{mit } [\![\mathcal{M}_S^{(1')}]\!] = ([\![E_S]\!], [\![F_S^{(1')}]\!]), [\![\mathcal{M}_S^{(2')}]\!] = ([\![E_S]\!], [\![F_S^{(2')}]\!]), \ldots\end{aligned}$$ ⌟

5.2.3 Zusammenhänge bei Zustandsautomaten

Automaten Algorithmische Verhalten werden häufig als Automaten modelliert, welche eine besondere Black-Box-Spezifikation für ganze Systeme sind (siehe Abschnitt 3.3.3). Ihr Vorteil ist, dass zwei Systeme relativ einfach komponiert werden können, in dem der Produktautomat gebildet wird. Dies führt insbesondere dazu, dass Systeme, die als Automaten modelliert sind, effizient analysiert werden können. In diesem Abschnitt werden zwei Methoden vorgestellt, die in der Literatur vorgeschlagen werden. Eine Methode dient der Überprüfung der Einhaltung von Sicherheitsanforderungen, dient also der Modellierung von Zusammenhängen von Transitionsmodifikationen zu Anforderungs- oder Systemmodifikationen. Die zweite vorgestellte Methode dient

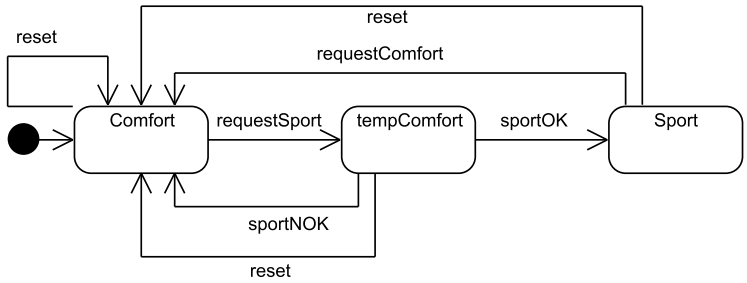


Abb. 5.8: Beispiel für eine Programmlogik

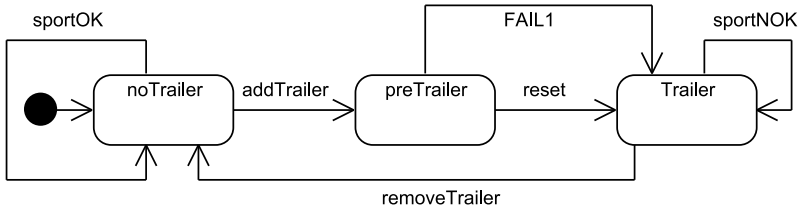


Abb. 5.9: Beispiel für eine Überwachung

der Überprüfung von Zusammenhängen zwischen Transitionsmodifikationen. Beide Ansätze wurden in der Literatur nur für Modifikationen definiert, die keine E-Mengen enthalten. In diesem Abschnitt wird zusätzlich die Erweiterung der Ansätze um den vollen Modifikationsumfang dieser Arbeit diskutiert. Zur Veranschaulichung dieser zwei Ansätze dient folgendes Beispiel, welches aus der Fallstudie inspiriert ist:

Beispiel 5.2.10 (Automaten)
Gegeben sei die Möglichkeit, zwischen einem sportlichen und einem komfortablen Fahrprogramm zu wechseln, wie auch eine Überwachung, welche festhält, ob das Fahrzeug im Anhängerbetrieb ist, oder nicht. Die zugehörigen Automaten sind in den Abbildungen 5.8 und 5.9. Die Automaten sind endliche Automaten, welche sich über Ereignisse koordinieren[6]. Gemeinsame Ereignisse sind hier die Ereignisse *reset*, *sportOK* und *sportNOK*. Die Automaten koordinieren sich so, dass die Anforderung "Der Anhängerbetrieb und das sportliche Fahrprogramm können nicht gleichzeitig aktiviert werden." erfüllt ist. Der Automat in Abbildung 5.9 enthält mit der Transition *FAIL1* einen modellierten Fehler, dessen Wirkung zu bestimmen ist. ⌟

[6]Dies ist eine Vereinfachung der Automaten aus Abschnitt 3.3.3. Die Beispiele werden hierdurch jedoch einfacher nachvollziehbar.

5.2.3.1 Einhaltung von Safety/Bedingungen

Erreichbarkeit

Verfahren zur Überprüfung der Einhaltung von Safety-Bedingungen sind in [OR04, ORS05] zu finden. Die Systeme werden hierbei komplett als Automaten wie in Beispiel 5.2.10 modelliert. Die Fehlerursachen werden modelliert, indem Transitionssysteme verändert werden, also Transitionsmodifikationen aus Abschnitt 4.2.3 darauf angewendet werden. Die Folgefehler werden hier nicht als Transitionsmodifikationen beschrieben, sondern sind Zustände oder Folgen von Zuständen des Systems, welche nicht erreicht werden dürfen. Diese Zustände werden meist mit Prädikaten aus 3.3.1 beschrieben.

Effizienz

Der große Vorteil dieser Verfahren ist die hohe Effizienz, mit der die Systeme verifiziert werden können. Da das System nicht mit einem Sollsystem verglichen werden muss, reicht es, für dieses einmal den Produktautomaten aus den Teilsystemen aufzubauen, und dann die Einhaltung der Bedingung zu überprüfen. Dies ist so eines der wenigen Verfahren, welches in polynomieller Zeit die Berechnungen durchführen kann. Folgendes Beispiel verdeutlicht dieses Verfahren:

Beispiel 5.2.11 (Produktautomat)
Die Automaten aus Beispiel 5.2.10 werden in Abbildung 5.10 zu einem Produktautomat zusammengefasst. Die gemeinsamen Ereignisse der Automaten sind in eckige Klammern gestellt, da sie nach außen hin nicht sichtbar sind, sondern nur der Synchronisation der Automaten dienen. In dem Produktautomaten ist der Zustand unten rechts derjenige, der bei korrektem Betrieb nicht erreicht werden soll, da er die Anforderung aus Beispiel 5.2.10 verletzt. Hier ist zu erkennen, dass dieser nur über die Transition *FAIL1* erreicht werden kann. Ein Folgefehler der Transition *FAIL1* ist also die Verletzung der Anforderung. ⌟

Erweiterung

Dieser Ansatz kann verwendet werden, um die F-Mengen einer Modifikation zu überprüfen, da die Basis der Produktautomat mit den möglichen erreichbaren Zuständen ist. Es gilt hier die Gleichung 5.3, wobei statt dem System $[\![S^a]\!]\Delta[\![\mathcal{M}_S^a]\!]$ der modifizierte Produktautomat eingesetzt wird. Zur Überprüfung der E-Mengen müsste allerdings nicht geprüft werden, ob der Automat eine Bedingung einhält, sondern ob die Bedingung immer den Automaten impliziert. Diese Anfrage wird von den meisten Analysewerkzeugen nicht zur Verfügung gestellt bzw. erfordert einen Rechenaufwand zur Verifikation, der denen bei allgemeinen Anforderungen entspricht. Sinnvoll ist es hier, die Analyse auf die F-Mengen zu beschränken.

Anwendbarkeit

Ein Nachteil dieser Ansätze ist, dass bereits eine formale Spezifikation existieren muss, welche die nicht akzeptablen Zustände eindeutig identifizieren kann. Sicherheitsanforderungen an Abweichungen (z.B. Überschreitungen von Toleranzen) können hier nicht gestellt werden. Ein weiterer Nachteil dieses Verfahrens ist die Einschränkung auf endliche Automaten. Diese können zwar, wie in [Hen96] beschrieben, zu hybriden endlichen Automaten erweitert werden, aber die Ausrichtung auf Beschreibungen von Abläufen und nicht auf komplexe Datenflussoperationen bleibt erhalten. Die Beschreibung größerer Datenflusssysteme nur mit Automaten ist meist

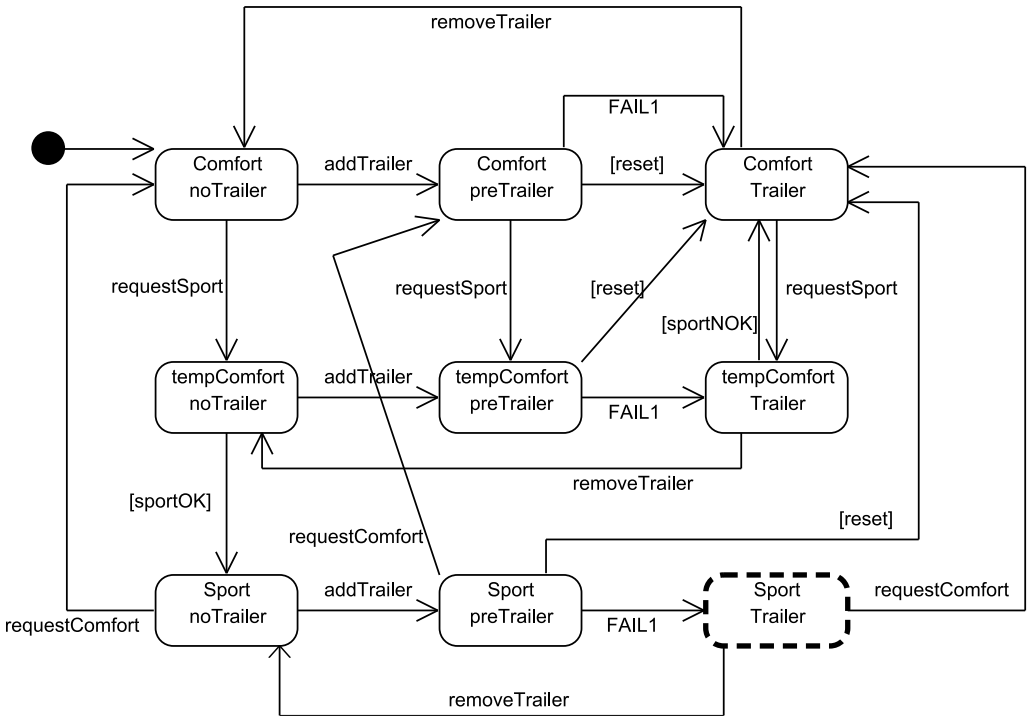


Abb. 5.10: Produktautomat der obigen Programmlogik

nicht möglich. Diese Ansätze sind deshalb für Fahrwerkregelsysteme, in denen der Anteil an Datenfluss-Operationen weit über 90% ausmacht, nicht geeignet. Anwendung finden diese Analysen eher in diskreten Fahrzeugsystemen, wie z.B. einem Fahrtrichtungsanzeiger.

5.2.3.2 Zusammenhänge zwischen Transitionsmodifikationen

Diagnose-automat

Ein Verfahren zur Überprüfung der Angabe der Transitionsmodifikation, welche als Folge von Transitionsmodifikationen in Teilsystemen eines Systems entstehen, ist in [SSL+95, SSL+96] zu finden. Der Vorteil dieses Verfahrens ist es, dass keine Anforderungen vorhanden sein müssen, sondern für einen gegebenen Automaten mit Transitionsmodifikationen die daraus folgenden Abweichungen ablesbar sind. Das Verfahren nimmt eine Menge von Automaten und setzt diese, wie in Beispiel 5.2.11 zusammen. Geht man davon aus, dass die gemeinsamen Ereignisse der Teilsysteme und die durch Fehler hinzugefügten Transitionen nicht nach außen sichtbar sind, so erhält man das Verhalten des zusammengesetzten Automaten. Zu diesem wird ein deterministischer Automat zur Diagnose gebildet, welcher in den Zuständen notiert, ob in dem bisherigen Pfad eine fehlerhafte Transition zum Erreichen des Zustands ausgeführt werden musste. Folgendes Beispiel demonstriert das Verfahren:

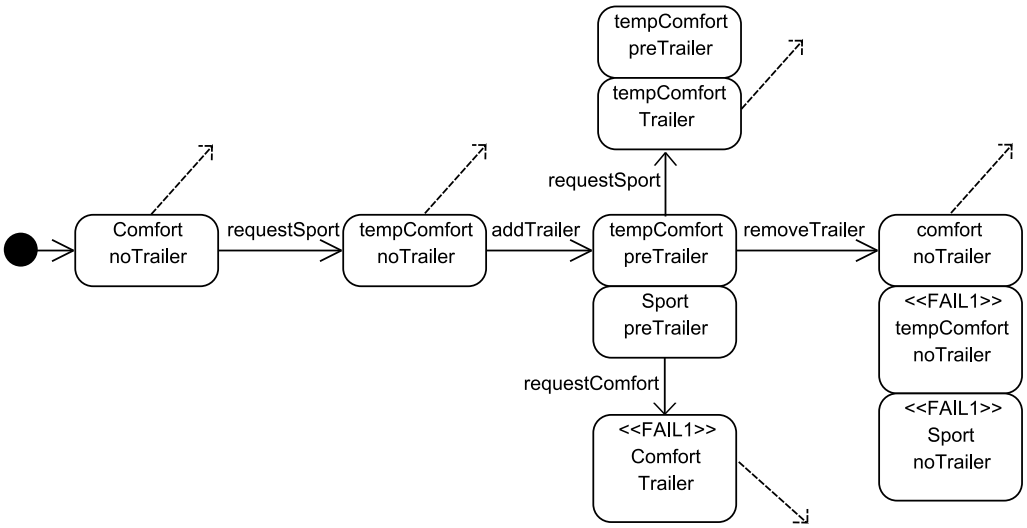


Abb. 5.11: Automat zur Ermittlung der Fehlerfolgen als Abweichungen

Beispiel 5.2.12 (Diagnoseautomat)
In diesem Beispiel wird die Erstellung eines Automaten zur Diagnose der Folgefehler aufbauend auf dem Beispiel 5.2.11 gezeigt. Der Diagnoseautomat in Abbildung 5.11 wird hier ausgehend von dem Startzustand *Comfort,noTrailer* gebildet. Durch das Ereignis *requestSport* kann der Automat nur in den Zustand *tempComfort,noTrailer* springen. Folgt nun das Ereignis *addTrailer*, so kann eventuell zuvor das gemeinsame Ereignis *[sportOK]* stattgefunden haben. Der Automat befindet sich dann entweder im Zustand *Sport,preTrailer* oder *tempComfort,preTrailer*. Im Folgenden werden drei Fälle demonstriert:

(1.) Findet nun das Ereignis *requestComfort* statt, so kann dies nur stattfinden, wenn die Transition *FAIL1* zuvor geschalten hat. Diese Transition ist also eine Folge der Transition *FAIL1*. Dies wird in dem folgenden Zustand *Comfort,Trailer* mit dem Stereotypen $\ll FAIL1 \gg$ notiert, und ist so explizit ablesbar.

(2.) Findet stattdessen das Ereignis *requestSport* statt, so kann zwar nicht ausgeschlossen werden, das die Transition *FAIL1* stattgefunden hat, aber jeder der möglichen Folgezustände ist auch ohne die Transition zu erreichen, und deshalb nicht beobachtbar.

(3.) Findet das Ereignis *removeTrailer* statt, so kann das System entweder in einem durch die Transition *FAIL1* abweichenden Zustand sein, oder nicht. Hier zeigt sich erst im weiteren Verlauf, ob die Folge nach außen hin beobachtbar ist. ⌟

Erweiterung Die Methode wird in der Literatur lediglich für Modifikationen vorgestellt, welche nur Transitionen hinzufügen. Diese Methode lässt sich auf Modifikationen mit E- und F-Mengen erweitern, in dem im Produkt- und im Diagnoseautomaten nicht nur die hinzugefügten, sondern auch die gelöschten Transitionen mitgeführt werden. Durch einen Abgleich mit dem Automaten auf einer höheren Ebene kann dann die Folgetransition bestimmt werden.

Diese Methode hat zwei Vorteile. Zum Einen kann explizit festgestellt werden, welche Modifikationen auf dem Verhalten des komponierten Automaten angewendet werden müssen, um ein äquivalentes Verhalten zu bekommen. Zum Anderen kann mit dieser Methode in einer Berechnung eine Menge von Modifikationen analysiert werden. Damit ist das Verfahren sehr effizient hinsichtlich der Analyse von vielen Einfachfehlern und Mehrfachfehlern. *Vorteile*

Diese Methode setzt voraus, dass alle Ebenen der Systemhierarchie als Automaten modelliert sind. In der Praxis ist diese Voraussetzung allerdings selten gegeben. So konnte in den Fallstudien diese Methode nicht angewendet werden, das keine abstrakten Automaten vorhanden waren. *Anwendbarkeit*

5.3 Integration der Implementierungsstruktur

Fehler in der Implementierungsebene werden nicht in den Verhaltensmodellen modelliert, sondern können als Label lediglich Fehlverhalten zugeordnet werden. Kausale Zusammenhänge zwischen ihnen sind über die Struktur des Implementierungsmodells gegeben, nicht aber über die Verhaltensebene. Existiert in der Sollspezifikation eine Verhaltensabhängigkeit, so werden diese Fehlerabhängigkeiten dort modelliert. Da in den Verhaltensmodellen nur Auswirkungen zwischen Systemen modelliert werden können, zwischen denen eine Funktionsabhängigkeit besteht, werden Auswirkungen, die über die Soll-Verhaltensstruktur hinaus gehen, innerhalb der Implementierungsstruktur modelliert. Ein Beispiel für eine solche Fehlerfolge ist eine elektromagentische Abstrahlung eines Bauteiles, welches ein anderes in Folge stört. In der Struktur des Sollverhaltenmodells ist in diesen Fällen meist keine Signalabhängigkeit gegeben. Entsprechend wird diese im Implementierungsmodell gegeben. *Struktur*

Definition 5.3.1 (Implementierungsartefaktfehlerabhängigkeit)
Die Menge *EE* aller Implementierungsartefaktfehlerabhängigkeiten ist:

$EE \stackrel{def}{=} \mathcal{P}(\mathbb{E}) \times \mathbb{E}$
mit der Menge der Implementierungsartefaktfehler $\mathbb{E}$ (siehe Definition 4.3.1).

Eine Implementierungsartefaktfehlerabhängigkeit $\mathfrak{e}$ ist ein 2-Tupel:
$\mathfrak{e} \stackrel{def}{=} (\{\mathcal{E}_{1a}, \mathcal{E}_{1b}, \ldots\}, \mathcal{E}_2)$ mit $\mathfrak{e} \in EE$ ⌟

Analog der Modellierung der Implementierungsstruktur mit Klassen kann auch die Fehlerabhängigkeit auf Ebene von Klassen angegeben werden. Eine Implementierungsartefaktfehlerabhängigkeit besteht dann, wenn zwischen den Klassen eine Abhängigkeit besteht und die Instanzen der Klassen miteinander verbunden sind. Da Klassenstukturen relativ viele Freiheitsgrade bei der Instanziierung haben, können theoretisch bei den Klassen verschiedene Angaben gemacht werden, wie die Fehlerabhängigkeiten zwischen den Implementierungen gestaltet sein sollen (z.B. $\wedge$- oder $\vee$-Verknüpfung, ...). Im Folgenden wird repräsentativ die Abhängigkeit von Einfachfehlern zwischen Klassen definiert: *Klassen*

Definition 5.3.2 (Implementierungsklassenfehlerabhängigkeit)
Die Menge EE^{class} aller Implementierungsklassenfehlerabhängigkeiten ist:

$EE^{class} \stackrel{def}{=} \mathbb{E}^{class} \times \mathbb{E}^{class}$

mit der Menge der Implementierungsklassenfehler $\mathbb{E}^{class}$ (siehe Definition 4.3.2).

Eine Implementierungsklassenfehlerabhängigkeit $\mathfrak{c}$ ist ein 2-Tupel
$\mathfrak{c} \stackrel{def}{=} (\mathcal{E}_1^{class}, \mathcal{E}_2^{class})$ mit $\mathfrak{c} \in EE^{class}$

so dass gilt:
$\forall((\mathcal{C}_1, \mathcal{L}_1), (\mathcal{C}_2, \mathcal{L}_2)) \in EE^{class}, (\mathcal{I}_1, \mathcal{I}_2) \in II :$
$((\mathcal{C}_1, \mathcal{I}_1) \in CI \wedge (\mathcal{C}_2, \mathcal{I}_2) \in CI) \Rightarrow \exists(\{(\mathcal{I}_1, \mathcal{L}_1)\}, (\mathcal{I}_2, \mathcal{L}_2)) \in EE$ ⌟

Rückkopplung

Die Implementierungsmodelle müssen nicht, wie die Verhaltensmodelle hierarchisch als Komponenten aufgebaut sein, sondern können auch, wie in UML-Objektdiagrammen, flach in einer Ebene modelliert sein. Die Zusammenhänge zwischen Fehlern werden in diesen Modellen also nicht wie in einem Fehlernetz der Produkt-FMEA nur entlang einer hierarchischen Struktur modelliert, sondern können beliebig sein. Möchte man einen Fehlerbaum aufbauen, so ist in der Implementierungsebene klarzustellen, wie mit Kreisen in dem gerichteten Graphen umgegangen wird.

Ermittlung

Die Ermittlung der Fehlerabhängigkeiten in der Implementierungsstruktur erfolgt über Erfahrungen, Statistiken und systematischem Erschließen. Ein automatisches Verfahren hierzu gibt es nur in so fern, dass die angegebene Fehlerabhängigkeitsfunktion transitiv ist, also automatisch aus den Relationstupeln die transitive Hülle gebildet werden kann. Ebenso gilt für die Fehlerzusammenhangsmodelle, dass auf dieser Ebene zur einfacheren Handhabung die Zeit nicht betrachtet wird.

Beispiel 5.3.1 (Fehlerabhängigkeit in der Implementierungsstruktur)
Gegeben seien Steuergeräte $\mathcal{I}_{ECU\_x}$, welche über ein gemeinsames Bussystem $\mathcal{I}_{CAN}$ kommunizieren (siehe Klassenmodell Abbildung 3.5).

Eine 'elektomagnetische Störung' (EM-Stoerung) des Busses kann zu einer 'Empfangsverzögerung' (Verzug-I) der Kommunikationseinheit der Steuergeräte führen. Modelliert wird dies als ein Tupel: $(\{(CAN, EM-Stoerung)\}, (ECU, Verzug-I))$ ⌟

Fehlerauswirkung

An die Implementierungsmodelle sind in dieser Arbeit nur Verhaltensmodelle im Sinne des logischen Sollverhaltens geknüpft. Zur Ermittlung anderer Wirkungen, wie zum Beispiel thermischen Effekten oder mechanischen Effekten können andere Modelle zur Hilfe gezogen werden. So ist z.B. denkbar, mit einem CAD-Programm zu prüfen, ob eine lose Schaube weitere mechanische Schäden verursachen kann. Diese Thematik wird in dieser Arbeit nicht behandelt.

Zusammenhänge

Bei der Betrachtung der Fehlerauswirkung in einem Verhaltensmodell, welches mit einem Implementierungmodell verbunden ist, gibt es vier Möglichkeiten der Folgefehler: (1) die beiden Fehler sind in der Verhaltensebene, (2) beide sind in der

Implementierungsebene, (3) die Ursache ist in der Impementierungsebene und die Wirkung in der Verhaltensebene und schließlich (4) umgekehrt die Folge in Richtung der Implementierungsebene.

zum Verhalten

Die Zusammenhänge der Fehler rein in der Verhaltensebene (1) und rein in der Implementierungsebene (2) wurden in den vorhergehenden Absätzen behandelt. In dieser Arbeit noch nicht modelliert sind die Beziehungen zwischen diesen Ebenen. Bei der Systemmodellierung werden die Implementierungselemente und die zugehörigen Verhalten über die Relation *SI* verbunden. Ein zu einem Implementierungselement gehörender Fehler müsste entsprechend mit einem Fehlverhalten verbunden werden können. Die Fehlverhalten können mittels Modifikationen ausgedrückt werden. Der Zusammenhang zweier Fehler im Sinne des Falls (3) wird in einer Relation $\mathcal{EM} \subseteq \mathbb{E} \times MOD$ festgehalten. Je nach dem, ob Einfachfehler oder Mehrfachfehler betrachtet werden, können die zugehörigen Modifikationen gleichzeitig oder einzeln in das Verhaltensmodell induziert werden.

zur Implementierung

In Richtung der Implementierungsebene (4) ist die Beschreibung der Zusammenhänge schwieriger. Trotz korrektem Verhalten kann eine unsachgemäße Nutzung im Sinne einer Fehlerauswirkung δ zu einem Implementierungsfehler als Folge führen. Der Zusammenhang in Richtung der Implementierungsebene wird also in einer Relation $\mathcal{DE} \subseteq MOD \times \mathbb{E}$ festgehalten. Eine Abweichung des Verhaltens kann von verschiedenen Ursachen abhängen und abhängig davon unterschiedliche weitere Folgefehler auf der Implementierungsebene verursachen. Dieses Problem ist ähnlich dem der Rückkopplung, in der die vorhergehenden Ausgaben relevant sind. Hier ist auf eine geeignete Modellierung der Systeme zu achten. Folgendes Beispiel verdeutlicht diese Problematik:

Beispiel 5.3.2 (Modellierungsanpassung für Fehlerzusammenhänge)
Gegeben sei ein System, welches mit einem Temperatursensor $\mathcal{I}_{Sensor}$ die Temperatur einer Kupplung abgreift und entsprechend bei erhöhter Temperatur die auf der Kupplung lastende Leistung reduziert.

Aufgrund eines Offsetfehlers misst der Sensor zu geringe Temperaturen. Im Verhaltensmodell ergibt sich so am Ausgabekanal o_{temp} des Sensorverhaltens S_{Sensor} der Fehler $\mathcal{M}^{BzK}_{o_{temp}}$. Es gilt:
$((\mathcal{I}_{Sensor}, \textit{'Offset zu niedrig'}), \mathcal{M}^{BzK}_{o_{temp}}) \in \mathcal{EM}$

Betrachtet man das Folgefehlverhalten, so verhält sich das Gesamtsystem wie in dem Falle, in dem die Temperatur der Kupplung normal ist. Verfolgt man die Fehlerauswirkung bis zu dem Verhalten, welches auf dem Element $\mathcal{I}_{Kupplung}$ abgelegt ist, so kann eine Fehlerauswirkung $\delta^{BzG}_{o_{leistung}}$ entstehen. Dies kann zu dem Implementierungsfehler 'Überhitzung der Kupplung' führen, welcher im Extremfall zu einem Übertragungsausfall der Leistung führen kann.
$((\mathcal{I}_{Kupplung}, \textit{'Überhitzung der Kupplung'}), \mathcal{M}^{0}_{o_{leistung}}) \in \mathcal{EM}$

Die generelle Abweichung $\delta^{BzG}_{o_{leistung}}$ führt aber nur dann zu einem Fehler 'Überhitzung der Kupplung', wenn die Ursache eine fehlende Leistungsreduktion wegen

Hitze ist. Um die Folge des Fehlverhaltens möglichst korrekt zu fassen, sollte auf die Fehlerauswirkung $\delta^{nichtreduziert}_{temperatur}$ hin untersucht werden. Diese kann dann zu dem Implementierungsfehler führen.
$(\delta^{nichtreduziert}_{temperatur}, (\mathcal{I}_{Kupplung}, \text{'}\ddot{U}berhitzung\ der\ Kupplung\text{'})) \in \mathcal{DE}$

Eine derartige Fehlermodellierung fordert allerdings auch von dem Verhaltensmodell, dass es geeignet mit Betriebsmodi modelliert wurde. ⌟

Anwendung In der Praxis spielen die Implementierungsmodelle hinsichtlich der Elektronik kaum eine Rolle. Wesentlich ist hier die Sammlung potentieller Fehler und deren Abbildung auf Modifikationen des Verhaltens. In seltenen Fällen werden auch die Zusammenhänge zwischen Implementierungsfehlern betrachtet. Meist sind jedoch alle denkbaren Zusammenhänge in den Verhaltensmodellen enthalten und ein Ausweichen auf die Implementierungsmodelle ist nicht notwendig. Hinzugezogen werden diese Modelle bei Bedarf an der Schnittstelle zur Mechanik, an der durch die Konstruktion eines Systems Seiteneffekte entstehen können. Diese Seiteneffekte werden meist beim Entwurf sicherer eingebetteter Systeme bereits vorgesehen und sind dann als Sicherheitsanforderungen für die Systeme geben. So ist zumindest teilweise der Zusammenhang von Verhaltensfehlern zu Implementierungsfehlern intuitiv gegeben.

5.4 Fallstudie

Aufgaben Einige der in diesem Kapitel vorgestellten Zusammenhänge und Abstraktionen wurden in der Analyse der Applikationssoftware eines Fahrwerkregelsystems (siehe Abschnitt 1.3) angewendet. Eine vereinfachte Skizze zur Einordnung der Analysen des Systems in der Umgebung ist Abbildung 5.12. Es wurden zwei Arten der Zusammenhänge betrachtet. Zum Einen wurden anforderungsbasierte Zusammenhänge innerhalb der Software entlang der Systemhierarchie ermittelt und dokumentiert. Zum Anderen wurden speziell für aufgelöste Gleichungen die Zusammenhänge der Abweichung von Eingaben zu den Ausgaben der Software modelliert, damit das System mit anderen Zusammenhangsmodellen integriert werden kann (siehe Abschnitt 2.5). Die Fehlerbeschreibungen umfassten hier auch den Zusammenhang zu Betriebsmodi. Reine Transitionssysteme waren in der Fallstudie nicht vorhanden. In den betrachteten Modellen konnten Automaten, aber nicht Automateninteraktionen im Sinne einer Anwendung der Zusammenhangsmodelle aus Abschnitt 5.2.3 vorgefunden werden. Für die Zusammenhangsmodellierung wurden keine Mehrfachfehler betrachtet, wie zum Beispiel für eine Fehlerbaumanalyse. Die modellierten Zusammenhänge entsprechen denen für anforderungsbezogene Einfachfehler im Sinne von Abschnitt 5.2.1.3.

Anwendbarkeit Die Modelle der Fallstudie waren nicht mit stromgebundenen Formeln festgehalten, sondern mit wertgebundenen Formeln. Entsprechend musste zur Modellierung der Zusammenhänge die Bedeutung für Semantiken aus Abschnitt 5.2.1 verwendet werden. Zur Modellierung der Zusammenhänge wurden als Bezug anforderungsba-

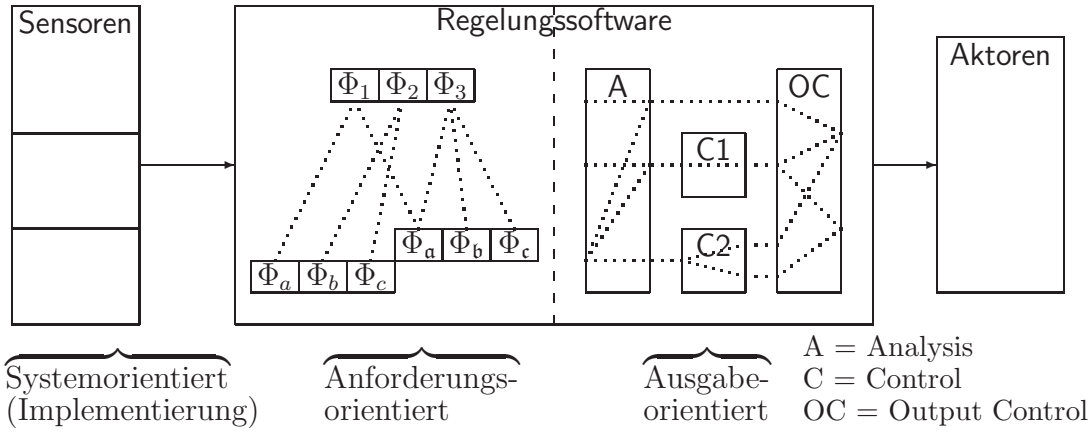


Abb. 5.12: Einordnung der Analysen der Fallstudie

sierte Einfachfehler verwendet. Die Bedeutung der in Abschnitt 5.2.1.2 vorgestellten Abstraktion der Aufhebung aber auch der Potenzierung von Fehlern ist für die Fallstudie schwer zu bewerten. Die Modelle der Fallstudie haben bereits Sicherheitsfunktionen enthalten. Ist die Sicherheitsfunktion (zum Beispiel Abschaltung) in einer anderen Anforderung als die überwachte Funktion, so kann die Aufhebung der Fehlerwirkungen häufig nicht erkannt werden. Gerade beim Bezug auf aufgelöste Gleichungen war es notwendig, Alternativen zu suchen. Diese bestanden entweder in der Verwendung von Mehrfachfehlermodellen oder im Zusammenfassen von Anforderungen.

Anforderungen

Die anforderungsbasierte Modellierung der Zusammenhänge setzte sich aus der Modellierung eines Funktionsnetzes und eines Fehlernetzes zusammen. Bei den formal modellierten Anforderungen wurden Zusammenhänge zwischen Anforderungen dann modelliert, wenn die Bedingungen aus Abschnitt 5.2.1.4 erfüllt waren. Die informellen Anforderungen wurden mit dem gleichen Grundgedanken vernetzt, ohne jedoch die korrekte Modellierung nachweisen zu können. Die Fehlermodellierung zu den Anforderungen bestand jeweils in der Verletzung der Anforderungen, die zu den Systemen gegeben waren. Zusammenhänge zwischen den Fehlern wurden entsprechend der Bedingung aus Abschnitt 5.2.1.3 modelliert. Auch hier galt, dass sich die Vernetzung von Fehlern zu informellen Anforderungen an den formalen Grundgedanken anlehnten. Ermittelt wurden die Zusammenhänge mittels des in Abschnitt 5.2.1.1 vorgestellten Vergleichs der modifizierten Systeme. Auf spezifische technische Herausforderungen hierzu wird in Abschnitt 6.2 eingegangen. Der Vergleich des in dieser Arbeit definierten Zusammenhangs mit dem intuitiv modellierten Zusammenhang der Ingenieure hatte eine hohe Übereinstimmung. Damit ist gezeigt, dass die in diesem Kapitel vorgestellt Formalisierung der Anforderungen gut in einen Prozess integriert werden kann, bei dem auch manuell und teilweise auf informeller Ebene Zusammenhänge modelliert werden. Gleichzeitig besteht grundsätzlich bei

passenden Analysemethoden durch die Formalisierung des Zusammenhangsbegriffs die Möglichkeit der automatisierten Analyse.

Gleichungen

Um die Erstellung einer Sicherheitsanalyse zu einer oberhalb der Software liegenden Ebene zu erleichtern, wurde zusätzlich zu der anforderungsbasierten Analyse ein Zusammenhangsmodell für Abweichungen erstellt. Die Software ist, wie in Abbildung 5.12 zu sehen in einer Umgebung eingebettet, in der sie von Sensoren Signale bekommt und entsprechende Steuerungen vornimmt. Die Eingaben entsprechen größtenteils den Werten von Sensoren bzw. Schaltern. Bei den Sensoren können spezifische Fehler auftreten, die aus der Implementierung (siehe Kapitel 4 und Abschnitt 5.3) abgeleitet sind. Die Fehler dieser Sensoren zeigen sich meist als Abweichungen der Ausgaben. Entsprechend ist eine Ermittlung der Zusammenhänge von Abweichungen der Eingaben der Software hin zu Ausgaben der Software eine Unterstützung der Ermittlung der Folge auf einer höheren Ebene (siehe Abschnitt 5.1.2). Der Nutzen dieser Unterstützung ist nur indirekt zu beziffern. Beim Gleichsetzen eines Einfachfehlers mit der Abweichung an einem Eingabekanal konnten über 80% der Wirkungen an den Ausgabekanälen als unkritisch eingestuft werden. Für über 30% der Eingabekanäle sind Common-Cause-Abweichungen auszuschließen, da sie von einzelnen Sensoren, usw. kommen.

Als Basis für diese Analyse wurden nicht die Anforderungen verwendet, sondern ein Simulink-Modell, welches das Verhalten der Software auf allen Ebenen in gleicher Detaillierung beschreibt. Da aufgelöste Gleichungen auch Anforderungen sind, wurden zur Modellierung der Zusammenhänge die gleichen Definitionen wie bei Anforderungen (Abschnitt 5.2.1.3 und 5.2.1.4) verwendet. Allerdings wurden die Fehler feiner als die Abweichungen vom Sollverhalten unterschieden. Die Ermittlung der Zusammenhänge fand im Wesentlichen über einen Parallelvergleich statt. Um die komplexe Software analysieren zu können, wurde diese wiederum zerlegt und das Ergebnis aus Teilanalysen zusammengesetzt. Da hier einige Zusammenhänge manuell modelliert werden mussten, wurden diese weiter abstrahiert, um sie für den Anwender fassbar zu machen. Zum Beispiel wurde bei der Rückkopplung die Zeit abstrahiert (siehe Abschnitt 5.2.2.1). Da innerhalb der Software die Kanäle an einigen Stellen bei getrennter Beobachtung eine zu große Abstraktion der Wirkung mit sich brachten, wurde teilweise die Analyse der Wirkung auf einen Zeitraum begrenzt. Genaueres hierzu wird in Kapitel 6 vorgestellt. Wichtig war auch die Berücksichtigung der Stromtreue, falls mehrere Folgefehler zu einem Kanal angegeben wurden. Fanden bei der Fallstudie der Überlagerungslenkung im Vorfeld der Analyse noch Aufteilungen der Fehler statt, deren Vollständigkeit nicht intuitiv sicher war, wurde bei Software zur Berechnung des Differenzenmoments von vorne herein darauf geachtet, diese korrekt einzuhalten.

5.5 Zusammenfassung

In diesem Kapitel werden formale Zusammenhangsmodelle zwischen Fehlern, Modellierungstechniken und Methoden zur Ermittlung der Zusammenhänge vorgestellt. Als Grundlage wird die Definition für einen Fehlerzusammenhang aus [Bre01] verwendet, in dem sich ein Folgefehlverhalten aus der Komposition der fehlerhaften Systeme ergibt. In der Datenfluss-Sicht bietet es sich aufgrund der Gestaltung der Anforderungen als Gleichungen an, den Signalfluss zu verfolgen, um auf mögliche Folgefehler zu schließen. Zusammenhänge werden hier in Abweichungsmodellen, wie in [Str04] festgehalten. Der Vorteil ist, dass große Modelle durch Dekomposition in Teilmodelle zerlegt werden können, wodurch eine Integration mit manuellen Analysen ermöglicht wird. In beiden Fällen liefern die Definitionen zu Fehler- und Abweichungszusammenhängen immer die exakte Folge. Eine Definition abstrakter Zusammenhänge zwischen Fehlern, um effizientere Analysen zu ermöglichen, bleibt offen. Auch sind ohne abstrakte Zusammenhänge die Abweichungszusammenhänge für komplexe Systeme mit Rückkopplung schwer ermittelbar. In diesem Kapitel wird deshalb ein allgemeines Fehlerzusammenhangsmodell definiert, bei dem nicht immer der exakte Folgefehler angegeben werden muss, sondern auch eine obere und untere Schranke für die Folgefehler modelliert werden kann. Die Modifikationen können hierbei entlang der Systemhierarchie im Sinne einer Ursache-Wirkungs-Beziehung verbunden werden oder als Abweichungen zwischen Kanälen. Es werden weiter Mechanismen vorgeschlagen, um mittels der abstrakten Zusammenhangsmodelle die Analysen effizienter zu gestalten. Diese sind Abschätzungsmechanismen, also Abstraktionen der Fehlerwirkung, die durch Begrenzung der Betrachtungszeit oder durch Abstraktionen der Kombinatorik der Zeit und der Anforderungen entstehen.

Auf Basis der in diesem Kapitel definierten abstrakten Zusammenhangsmodelle werden für die Modellierungstechniken aus Kapitel 3 spezifische Zusammenhangsmodelle und Methoden zu deren Ermittlung vorgestellt. Für allgemeine Black-Box-Spezifikationen wird ein Vorgehen abgeleitet, um Fehlerzusammenhänge anhand der Fehlverhalten von Komponenten zu ermitteln. Dabei wird insbesondere berücksichtigt, dass Modifikationen Tupel der Verhaltensrelation sowohl hinzufügen, als auch entfernen können. Dieses Vorgehen wird sowohl für stromgebundene, wie auch für wertgebundene Prädikate gezeigt.

Da Black-Box-Spezifikationen nicht immer am Stück vorliegen, sondern häufig konjunktiv zusammengesetzt sind, wird in vielen Ansätzen der Sicherheitsanalyse, um Rechenleistung zu sparen, für die Folge nicht das System, sondern jeweils einzelne Anforderungen als Bezug genommen (siehe [OR04],[ORS05],[Str06],[PCB+55]). Alle diese Ansätze betrachten die Zusammenhänge nur über eine Ebene der Systemhierarchie. Diese Arbeiten stellen sich so nicht der Tatsache, dass in manuellen Analysen (siehe Kapitel 2) die Analyse über mehrere Ebenen geht. Deshalb werden in diesem Kapitel formal die Fehlerzusammenhänge im Sinne einer anforderungsbasierten Sicherheitsanalyse definiert und die daraus entstehenden Folgen für die Sicherheitsanalyse angegeben.

Ein Spezifikum der Datenfluss-Sicht ist die Modellierung der Fehler als Abweichungen. Ansätze zur Ermittlung der Folgen von Abweichungen sind [Str06], [PCB+55]. Diese Ansätze basieren auf einer automatisierten Analyse und bieten Möglichkeiten, die Zusammenhänge durch lineare Approximation und Diskretisierung abzuschätzen, nicht aber die Kombinatorik zwischen Kanälen und über die Zeit zu abstrahieren. Es werden deshalb Abschätzungsmechanismen vorgestellt, um die Wirkung von Abweichungen geeignet abzuschätzen. Diese Abstraktionen zielen insbesondere auf eine Integration mit manuellen oder einfachen automatischen Analysen, in denen die komplexen Zusammenhänge nicht bewältigt werden können. Wesentlich ist hier auch eine Kompatibilität der Definitionen mit den informell in der Fallstudie vorgefundenen Fehlerzusammenhängen.

In der Interaktionssicht wird das Verhalten meist mit Automaten ausgedrückt. Hier sind die Ansätze bereits am weitesten fortgeschritten. Zwei relevante Ansätze [SSL+95, SSL+96, OR04, ORS05] werden vorgestellt. Diese Ansätze erfassen nicht die in Kapitel 4 vorgestellten Modifikationen, die Transitionen sowohl hinzufügen, als auch entfernen. Diese Ansätze werden in diesem Kapitel um dieses Fehlerbild erweitert.

Letztlich wird die Integration der Implementierungsebene diskutiert und die Abstraktionen in der Fallstudie bewertet.

Kapitel 6

Werkzeuge und Analysetechniken

Ziel der Verifikationswerkzeuge ist die Überprüfung der in Kapitel 5 vorgestellten Abhängigkeiten. Dieses Kapitel diskutiert die Möglichkeiten, mit Softwarewerkzeugen diese Zusammenhänge automatisch zu überprüfen. Auf Basis der Überprüfungsmöglichkeiten kann eine teilautomatisierte Erstellung eines Analysedokuments für Funktionssicherheit (siehe Kapitel 2) durchgeführt werden.

Kapitelübersicht

Das Kapitel besteht im Wesentlichen aus zwei Teilen. Abschnitt 6.1 umreißt allgemein, welche Möglichkeiten zur Überprüfung bzw. Abschätzung der Gültigkeit der Zusammenhangsformeln bestehen. Abschnitt 6.2 stellt vor, wie die konkreten Fragestellungen der Zusammenhangsanalyse mit Werkzeugen beantwortet werden können.

6.1 Analysetechniken für Verhaltensmodelle

Ein Überblick über Analysetechniken und -werkzeuge kann [Lig02] und [RN03] entnommen werden. In diesem Abschnitt werden verschiedene Möglichkeiten vorgestellt, um Verhaltensmodelle zu verifizieren. Die vorgestellten Techniken dienen als Grundlage, um im folgenden Absatz die konkreten Verifikationsmethoden zu beschreiben. Zu allgemeinen Analysemöglichkeiten gehören die syntaktische Analyse für strukturelle Fragestellungen, Simulationen und Tests zur stichprobenartigen Überprüfung, die boolesche Verifikation in Verbindung mit Constraint-Solving und schließlich die automatische Abstraktion als Hilfsmittel für die boolesche Verifikation.

6.1.1 Syntaktische Analyse

Technik

Ein sehr einfaches wenn auch nicht besonders mächtiges Verfahren, Modelle zu verifizieren, ist die *syntaktische Analyse*. In dieser werden auf Modelle Algorithmen der Graphentheorie angewendet und so von den Eigenschaften der gerichteten Graphen

auf die Zusammenhänge geschlossen. Die einzige Voraussetzung für diese Analyse ist, dass aus dem Modell ein gerichteter Graph gebildet werden kann, der die zu untersuchenden Eigenschaften erfasst. Diese Eigenschaften sind meist die Existenz von Pfaden zwischen Knoten. Neben der Subsystem-Beziehung wird im Wesentlichen der Signalfluss zur Analyse hinzugezogen. Dieser kann sowohl aus den Simulink-Modellen, wie auch aus dem C-Code (Anweisungsfluss) extrahiert werden.

Abstraktion Zusammenhänge, die nicht auf Graphen abbildbar sind, werden nicht erfasst. Die Aussagekraft dieser Analysen hängt vom Grad der Abstraktion relevanter Aspekte bei der Bildung des Graphen ab: Der Systemstrukturbaum und der Signalfluss kann zum Beispiel voll in einem Graphen wiedergegeben werden. Die Ausgaben- und Fehlerabhängigkeiten, welche das Verhalten der Knoten berücksichtigen müssten, können nicht wiedergegeben werden. Tautologien oder die Multiplikation mit 0, deren Verhalten Signalabhängigkeiten eliminiert, werden bei der Abschätzung der Ausgabeabhängigkeiten über den Signalfluss nicht berücksichtigt. So kann nur eine obere Schranke angegeben werden.

Architektur-Hilfsmittel Die syntaktische Analyse muss verhaltensneutrale architekturelle Hilfsmittel in den Modellen explizit filtern. So können zum Beispiel im Modell verhaltensneutrale Elemente zur Codegenerierung enthalten sein, zur besseren Wartbarkeit Signale gebündelt werden, wobei einzelne Subkomponenten nur Teile des Bündels verwenden, und in der Komponentenhierarchie Hilfskomponenten als Zwischenschicht eingezogen sein. Anhang A zeigt ein Beispiel einer Graphengenerierung, welches explizit in der Fallstudie vorgefundene architekturelle Hilfsmittel filtert.

6.1.2 Simulation und Test

Technik Ein Pendant zur syntaktischen Analyse eines Systems ist die Analyse von Verhaltenszusammenhängen mittels *simulationsbasierter Tests*. Ein System wird manipuliert, mit bestimmten Eingaben simuliert und die Ausgaben auf bestimmte Eigenschaften geprüft oder mit Referenzausgaben verglichen. Die Voraussetzung ist die Simulierbarkeit der zu überprüfenden Systeme und die Beobachtbarkeit der Zusammenhänge an den Ausgaben der Systeme. Abgesehen von der Manipulation des Systems ist diese Analysemethode unabhängig von der White-Box-Sicht und somit von verhaltensneutralen architekturellen Hilfsmitteln. Ebenso werden auch Zusammenhänge erfasst, welche in den Modellen nicht explizit modelliert sind.

Fehlerbild Testfälle können zur Ermittlung beliebiger Zusammenhänge verwendet werden. Sie eignen sich sowohl zur Überprüfung von Zusammenhängen zwischen Anforderungen wie auch von kanalorientierten Zusammenhängen. Die Modellierung der Folgefehler ist für Versagen und Abweichungen verschieden. Zur Bewertung von Versagen werden Teile des Systems modifiziert und es wird überprüft, wie sich das Verhalten des Gesamtsystems ändert. Zur Ermittlung der Abweichungen werden Eingabesequenzen modifiziert und die Abweichungen der Ausgaben überprüft. Ein Beispiel zur Ermittlung der Fehlerauswirkung durch eine Komponente hindurch ist in Anhang C zu finden.

Auch wenn mit Testfällen beliebige Systeme überprüft werden können, so ist, um einen Zusammenhang ausschließen zu können, eine Überprüfung mit allen möglichen Eingaben notwendig (Pfadüberdeckung). Selbst bei booleschen endlichen Eingabesequenzen der Länge t mit n Variablen ist die Anzahl der Simulationen, die für eine sichere Aussage notwendig sind 2^{n*t}. Bei unendlichen Eingabesequenzen ($t = \infty$) wird auch die Anzahl der notwendigen Testeingaben unendlich. Ebenso wird bei Eingabesignalen deren Wertemenge ganzzahlig ($\mathbb{Z}$) oder kontinuierlich ($\mathbb{R}$) ist, die Menge der Tests unendlich. Für die meisten Systeme kann so mit Tests nur eine stichprobenartige Abschätzung der Zusammenhänge gemacht werden, welche eine untere Schranke für die tatsächlich vorhanden Zusammenhänge ist. Es gibt eine ganze Menge von Testansätzen, um möglichst geschickt aussagekräftige Eingabesequenzen zu finden [Lig02, Loe03, Pre03]. Die Wertebereiche mehrwertiger oder kontinuierlicher Kanäle können z.B. in wenige Äquivalenzklassen eingeteilt werden, welche sich aus dem Verhalten des Systems ergeben (siehe Abschnitt 6.1.4.1). Weiter kann sich die Auswahl der Eingabesequenzen z.B. an den Anforderungen an ein System orientieren [ES07a, ES07b]. *Abschätzung*

Der Vorteil der Verwendung von Testfällen ist, dass diese als Anforderungstests in der Entwicklung des Sollsystems bereits vorliegen und so die Spezifikation in Form von exemplarischen Anwendungen widerspiegeln [Bec03]. Der zusätzliche Aufwand für eine formale Beschreibung der Anforderungen zu Fehlerauswirkungsanalysezwecken ist so nicht mehr notwendig. Der zusätzliche Aufwand besteht dann nur noch in der Manipulation der Systeme und Eingaben und in der Analyse der Folgen. Aus diesem Grund sind Testfälle ein in der Industrie häufig verwendetes Mittel, welches auch zur Unterstützung von Analyseaufgaben hinzugezogen wird. *Anforderungstests*

6.1.3 Boolesche Verifikation modulo Theorien

Tests lassen durch ihren exemplarischen Charakter meist keine Zusicherung zu, um eine Wirkung auszuschließen. Um diese Aussagen zu treffen, müssten die Systeme vollständig getestet werden. In diesen Fällen ist es eine Alternative, Verifikationswerkzeuge zu nutzen, die effizienter arbeiten, in dem sie die formale Beschreibung der Systeme nutzen. Diese Effizienz erkaufen sich die Werzeuge unter anderem damit, dass nur noch bestimmte Systeme analysiert werden können. Allgemein bekannt ist die Analyse boolescher Systeme. Diese kann, wie in Abschnitt 6.1.3.2 vorgestellt, um darüber hinausgehende Systeme erweitert werden. *Effizienz*

6.1.3.1 Boolesche Verifikation

Dieser Abschnitt skizziert Werkzeuge zur Analyse boolescher Systeme. Hier bieten sich Techniken wie SAT-Solving, Model Checking [CG01] und Theorembeweisen [Duf91] an. Üblicherweise werden Eigenschaften in einem Modell verifiziert, oder zwei Modelle miteinander verglichen. Ersteres findet bei der klassischen Anforderungsverifikation statt. Beim Vergleich von Modellen können verschiedene Differenzen *Technik*

aufgedeckt werden. Entweder es wird auf Gleichheit geprüft, oder es wird geprüft, ob eine Verfeinerungsbeziehung im Sinne von Definition 3.4.3 vorliegt. Die Anfragen sind jedoch nicht eingeschränkt und können den Vergleich auch auf Modifikationen beziehen. Für diese Werkzeuge ist prinzipiell die Art der Anfrage und die zugrunde liegenden Modelle (bei Eigenschaftenverifikation) relevant.

Komplexität

Gemeinsam ist allen die Prüfung, ob eine Formel für alle möglichen Belegungen wahr ist, also die Formel allgemein gültig ist. Von der Komplexität der booleschen Formeln sind hier drei Stufen zu unterscheiden: Die einfachste Stufe sind boolesche *Aussagen*. Diese setzen sich aus den booleschen Operatoren $\wedge, \vee, \neg$ und den daraus konstruierbaren Operatoren zusammen. Vorteil dieser Aussagen ist, dass sie mit einfachen Werkzeugen wie SAT-Solvern in die boolesche Normalform gebracht werden können und dann vollautomatisch überprüft werden können. Die Tatsache, dass die Formel in boolescher Normalform vorliegt, ermöglicht häufig ein effizientes Optimieren der Anfragen. Die Effizienz der Systeme liegt in der Größenordnung von NP zur Anzahl der booleschen Variablen. Eine Komplexitätsstufe höher sind die *First Order Logik Prädikate (FOL)*. Diese Formeln können bereits Existenzquantoren enthalten, die aber eine Schachtelungstiefe von 1 nicht überschreiten. Auch für diese Formeln gibt es vollautomatische (semi-entscheidbare) Lösungsverfahren. Die Implementierungen sind allerdings noch Prototypen aus der Forschung, die entweder unvollständig sind, also nicht für alle Formeln funktionieren, oder von der Effizienz her nicht in akzeptablen Bereichen sind. Die dritte Stufe sind die *High Order Logic Prädikate (HOL)*. Diese Formeln lassen zu, dass die Quantoren beliebig tief geschachtelt werden können. Zur Überprüfung der Allgemeingültigkeit dieser Formeln gibt es keine vollautomatische Lösungsmöglichkeit. Hier werden die Formeln interaktiv mit Theorembeweisern, wie Isabelle [NPW02] überprüft. Im allgemeinen wird jedoch im Laufe der Arbeit festgestellt, dass es sowohl für FOL und HOL Prädikate keine für die Industrie reifen Produkte gibt, welche eine einfache Bedienung und Produktbetreuung bieten. Im Abschnitt 6.2 wird deshalb als Alternative die Vereinfachung mancher Formeln auf Aussagenlogik vorgestellt.

6.1.3.2 Integration mit Theorien

Aufspaltung

Der Umgang mit Systemen, welche nicht nur boolesche Variablen verwenden, erfordert eine Erweiterung der Verifikationswerkzeuge. Eine mögliche Erweiterung ist die boolesche Verifikation modulo Theorien [BPT07]. Die Fragestellung, ob eine Formel für alle Belegungen wahr ist, kann wie in Abbildung 6.1 dargestellt, beantwortet werden. Die zu überprüfende Formel kann (1) boolesche Variablen enthalten, (2) Variablen mit (wenigen) diskreten Werten enthalten und zusätzlich Variablen, die (3) eine sehr große oder (4) unendliche Anzahl an Werten annehmen können. Da eine Formel per Definition 3.2.3 immer zu einem booleschen Wert ausgewertet werden kann, sind die Variablen der Kategorie (3) und (4) immer in Teilformeln enthalten, die mit relationalen Operatoren die Schnittstelle zur booleschen Interpretation bieten. Daraus ergibt sich der in Abbildung 6.1 folgende erste Schritt: Um die Gültigkeit der Formeln zu ermitteln und dem hybriden Charakter der Formeln zu entgegnen, wer-

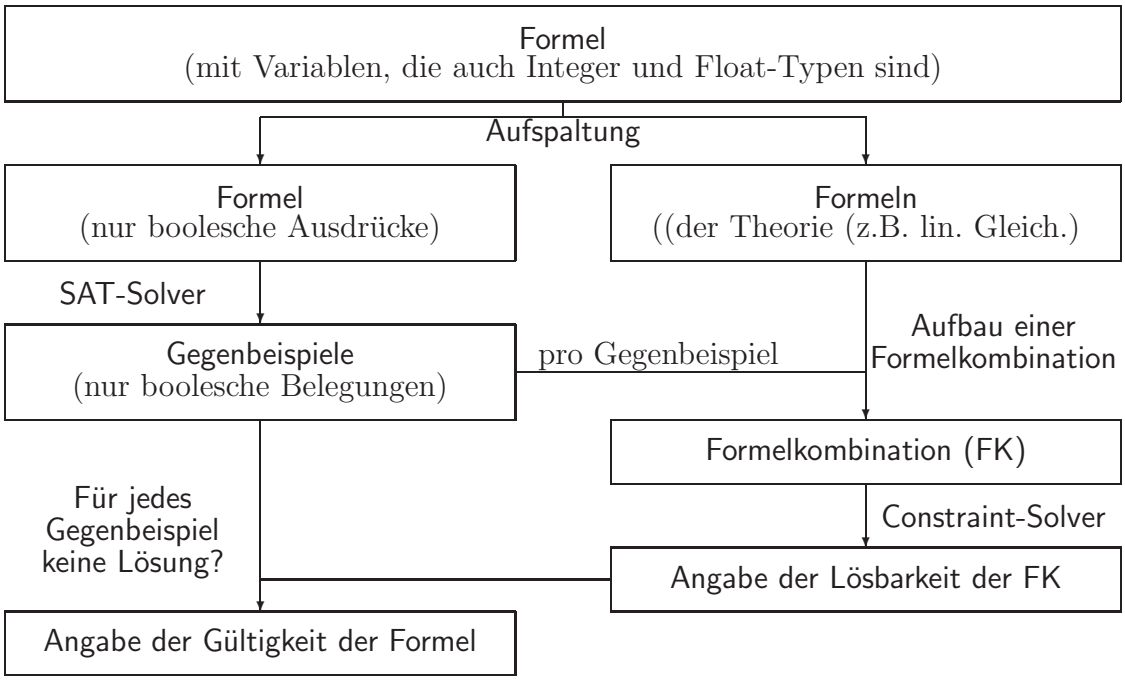


Abb. 6.1: Gültigkeitsprüfung bei hybriden Formeln

den die Formeln in Anteile aufgespalten, die effizient mit boolescher Analyse gelöst werden können und in Anteile, die mit einer Berechnung anhand einer mathematischen Theorie gelöst werden. Diese Aufspaltung ist notwendig, da zum Einen die verschiedenen Berechnungsverfahren bei verschiedenen Problemstellungen zu sehr unterschiedlich langen Berechnungszeiten führen können und sich zum Anderen die verschiedenen Berechnungsmethoden in ihrer Berechnungsmächtigkeit ergänzen. So können zum Beispiel mit SAT-Solving keine Aussagen überprüft werden, die Variablen der Kategorie (4) enthalten und umgekehrt mit linearen Lösungsverfahren nur Konstrukte überprüft werden, die einer linearen Abbildung entsprechen.

Die Entscheidung der Aufteilung der Formel ist meist so gewählt, dass möglichst alle Variablen der Kategorie (3) und (4) nicht in der booleschen Berechnung des Analysewerkzeugs enthalten sind und den Berechnungsmethoden einer Theorie zusortiert werden. Die Formeln werden aufgespalten, indem die zugehörigen relationalen Teilformeln $\Phi_i^{\mathcal{T}}$ als boolesche Variable $L_i^{\mathcal{T}}$ dargestellt werden, die genau dann wahr ist, wenn die Teilformel gilt und die Teilformel $\Phi_i^{\mathcal{T}}$ für die andere Theorie extrahiert wird. Diese booleschen Variablen heißen im Weiteren 'formelrepräsentierende' Variablen.

boolesche Analyse

Der allgemeine Teil der Formel, der auch den Rahmen der Gesamtformel bildet, enthält rein boolesche Variablen und Variablen mit wenigen Werten. Die Variablen mit wenigen Werten können zu einer Repräsentation aus mehreren booleschen Variablen transformiert werden. Hier können dann die Algorithmen für boolesche Formeln angewendet werden, um geeignete Gegenbeispiele zu finden (siehe Abbildung 6.1, 2. Schritt, links).

Constraint Solving

In jedem der gefundenen booleschen Gegenbeispiele können 'formelrepräsentierende' Variablen enthalten sein. Ein boolesches Gegenbeispiel kann nur dann gültig sein, wenn es eine Variablenbelegung gibt, für die die Formeln einer Theorie, die durch die Variablen repräsentiert werden, gültig bzw. nicht gültig sind. Dementsprechend wird für jedes Gegenbeispiel ein Formelsystem einer Theorie aufgebaut, welches dahin zu überprüfen ist, ob es eine Variablenbelegung gibt, die die Formelkombination zu true auswerten lässt (Constraint Solving). Nach [Hen96] kann eine Formelkombination einer Theorie $\mathcal{T}$ dann überprüft werden, wenn sie entscheidbar und unter den booleschen Operatoren geschlossen ist.

Definition 6.1.1 (Entscheidbarkeit)
Eine Theorie $\mathcal{T}$ ist *entscheidbar*, wenn für jede Formel $\Phi^{\mathcal{T}}$ angeben werden kann, ob es eine Belegung der Variablen gibt, die eine Auswertung zu *true* zulässt. ⌟

Definition 6.1.2 (geschlossene Theorie)
Eine Theorie $\mathcal{T}$ ist *unter den booleschen Operatoren geschlossen*, wenn Sie zu jeder Formel $\Phi^{\mathcal{T}}$ auch eine Formel $\neg\Phi^{\mathcal{T}}$ enthält und zu jedem Formel-Paar $\Phi_i^{\mathcal{T}}$, $\Phi_j^{\mathcal{T}}$ auch eine Formel $\Phi_i^{\mathcal{T}} \vee \Phi_j^{\mathcal{T}}$ bzw. $\Phi_i^{\mathcal{T}} \wedge \Phi_j^{\mathcal{T}}$ enthält. ⌟

Die Überprüfung der Formeln einer Theorie findet mit Satisfiability-Constraint-Solvern statt. Diese sind in den meisten Fällen sehr spezifisch auf die jeweiligen Theorien zugeschnitten. Die am weitesten verbreitete Theorie zur Angabe der Lösbarkeit einer Formelkombination ist die Theorie der linearen Gleichungssysteme, in der die Werte der Variablen einer totalen kreisfreien Ordnung unterliegen und jede Formel folgender Form entspricht:

$$\alpha_1 x_1 + \alpha_2 x_2 + ...[< \mid = \mid > \mid <= \mid >=]\beta_1 y_1 + \beta_2 y_2 ...$$

Zusammenführung

Ist für alle booleschen Gegenbeispiele gezeigt, dass diese zu Formelkombinationen führen, die keine zu true auswertbare Variablenbelegung zulassen, so gibt es kein valides Gegenbeispiel und die Formel ist gültig. Das Zusammenspiel zwischen dem Satisfyability-Solver und dem booleschen Analysewerkzeugen kann weiter optimiert werden, was in Abbildung 6.1 nicht angegeben wurde, da es nicht den Kern der Idee betrifft. Folgendes Beispiel verdeutlicht nochmals den Ablauf:

Beispiel 6.1.1 (Boolesche Verifikation modulo Theorien)
Gegeben sei die Aussage:
$((y = b) \wedge (z > c)) \Rightarrow \neg(3b - c < 3y - z)$ mit b,c,y,z vom Typ 'float'.

Der boolesche Teil der Formel ist: $(l_1 \wedge l_2) \Rightarrow \neg l_3$
mit den relationalen Anteilen:
$l_1 = (y = b)$
$l_2 = (z > c)$
$l_3 = (3b - c < 3y - z)$

Es gibt nur folgendes boolesches Gegenbeispiel:
(l_1, l_2, l_3)

Das (Un-)Gleichungssystem für das Gegenbeispiel ist:
$(y = b; z > c; 3b - c < 3y - z)$
$\Rightarrow (z > c \; ; \; 3b - c < 3b - z)$
$\Rightarrow (z > c \; ; \; -c < -z)$
$\Rightarrow (z > c \; ; \; c > z)$
$\Rightarrow$ Gleichungssysten nicht lösbar

Das einzige boolesche Gegenbeispiel ist nicht gültig und somit ist die Formel für alle Belegungen korrekt. ⌟

6.1.4 Abstraktionsmechanismen

Eine Menge realer Systeme erfüllt nicht die Einschränkungen, die zu einer booleschen Analyse modulo Theorien gegeben sein müssen. Ebenso gilt für Testfälle, dass nur ein Teil der möglichen Pfade aus Aufwandsgründen geprüft werden kann. In vielen Ansätzen [Thu04, Str06] ist es deshalb notwendig, das Verhalten des Modells so zu abstrahieren, dass dieses mit den Analyseverfahren mit akzeptablem Aufwand bearbeitet werden kann. Mit geeigneten Abstraktionsmechnismen lässt sich so die Menge der analysierbaren Systeme erweitern. Für alle Abstraktionen muss gelten, das der Erhalt von Eigenschaften, insbesondere hinsichtlich des Ausschlusses eines Folgefehlers gegeben ist.

Mechanismen

In dieser Arbeit wurden zwei Abstraktionsmechanismen mit verschiedenen Mächtigkeiten hinsichtlich der bearbeitbaren Modelle und der Aussagekraft der darauf stattfindenden Analysen betrachtet. Der erste Mechanismus ist die Abstraktion der numerischen Effekte bei Fließkommazahlen und Integer-Zahlen. Der zweite ist die Abstraktion der Fließkomma- oder Integer-Variablen zu wenigen Äquivalenzklassen (siehe [Sac01]), der auch Grundlage für einige Testverfahren ist. Diese Abstraktionsmechnanismen können mit den Abstraktionsmechanismen aus Abschnitt 5.2 kombiniert werden.

Numerik

Die Abstraktion der Numerik bei Integer und Fließkommazahlen kann in manchen Fällen zu Systemen führen, welche mit den Analysewerkzeugen aus Abschnitt 6.1.3.2 überprüft werden können. Hierbei liegt allerdings keine Verfeinerungsbeziehung im Sinne von Abschnitt 3.4.2 vor, sondern es werden Rundungsfehler abstrahiert. Abbildung 6.2.a hat als Originalsystem eine Typisierung eines Kanals auf Integer, was bei nicht ganzzahligen Summen zu einer Ausgabe false führt. Werden nun alle Integer zu kontinuierlichen Typen (Abbildung 6.2.c), so kann das System nie false ausgeben. Eine Analyse muss also die tatsächlichen numerischen Abweichungen berücksichtigen. Für die kontinuierlichen Werte wird hierzu eine obere und eine untere Schranke des tatsächlichen Werts angegeben (siehe [Hen96]). Genügen die abstrahierten Modelle der booleschen Analyse modulo linarer Gleichungssysteme, so können auf diese Weise sehr effiziente Analysen eines Systems stattfinden.

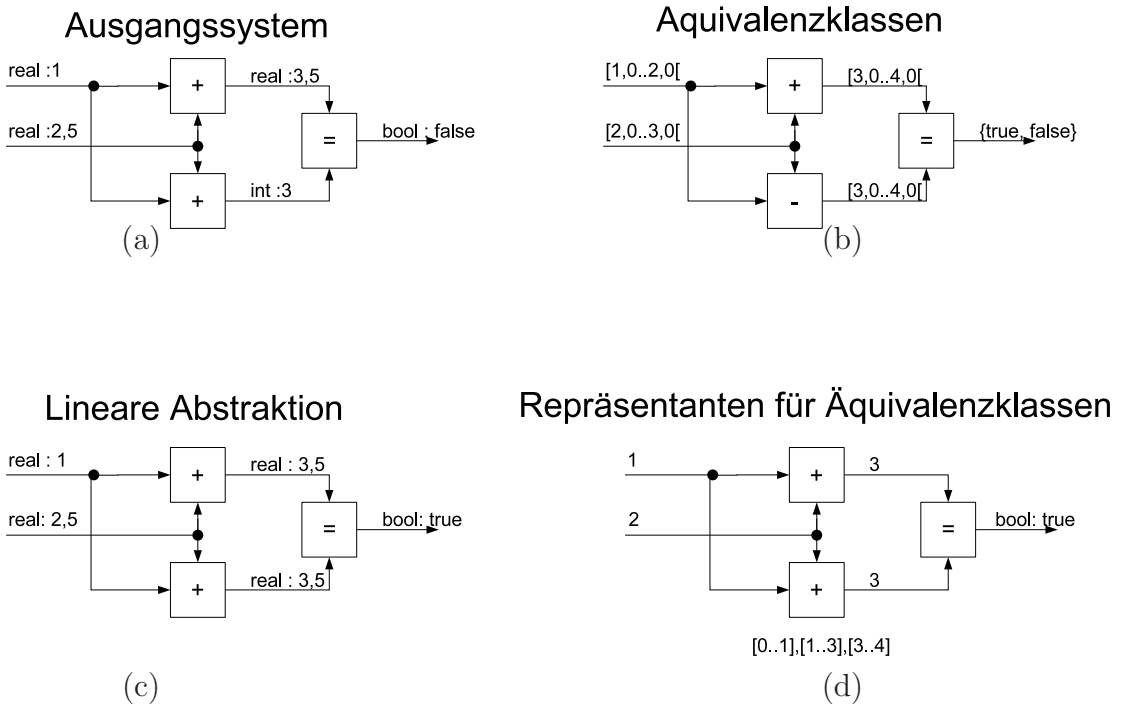


Abb. 6.2: Fehlerpotential bei der Verhaltensabstraktion eines Systems

Diskretisierung

In den Modellen der Fallstudien war es allerdings häufig nicht gegeben, dass die Systeme zu linearen Systemen abstrahiert werden konnten. In diesen Fällen bleibt nur der Weg, die Integer und Fließkomma-Variablen zu wenigen Äquivalenzklassen zusammenzufassen. Dieses Vorgehen hat zwei Vorteile: Es können Systeme mit beliebigen Operationen abstrahiert werden und die Abstraktionen erfüllen die Verfeinerungsbeziehungen aus Abschnitt 3.4.2. Damit ist der Erhalt von Eigenschaften hinsichtlich des Ausschlusses von Fehlerfolgen gegeben. Ein Nachteil der Abstraktion ist, dass die Systeme unter Umständen nichtdeterministisch werden und so Fehleraufhebungen eventuell nicht mehr erkannt werden. Werden die Werte, wie in Abbildung 6.2.b, hinter der Kommastelle zusammengefasst, so kann keine Aussage mehr über das Ergebnis getroffen werden. Ein weiterer Nachteil ist, dass die Effizienz der Analyse deutlich geringer ist, als bei der Abstraktion der Numerik. Die Abstraktion der Fließkomma- und Integer-Variablen muss sich dabei nicht unbedingt auf die Werte beziehen, sondern kann auch Eigenschaften der Wertverläufe wie Stetigkeit, Knickstellen, Frequenzen usw. enthalten. Auch die Auswahl von Testeingaben, welche Repräsentanten der Äquivalenzklassen sind, kann, wie in Abbildung 6.2.d gezeigt, die Ergebnismenge nicht voll erfassen.

Im Folgenden wird die Abstraktion zu wenigen Äquivalenzklassen genauer vorgestellt und der daraus resultierende Umgang mit dem Nichtdeterminismus weiter beleuchtet.

6.1.4.1 Abstraktion von Fließkomma- und Integer-Variablen

Die Abstraktion von Fließkomma- und Integer-Variablen verschieden gestaltet sein. Bei allen Abstraktionen gilt eine minimale Einhaltung der Verhaltensverfeinerung. Hier wird eine Möglichkeit vorgestellt, die Variablen zu Intervallen zusammenzufassen. Dies umfasst eine Definition der Intervallabstraktion und ein Verfahren zur geschickten Wahl der Intervalle. Die Idee für dieses Verfahren stammt aus [Sac01].

Verfeinerung

Ein Kriterium zur Einhaltung der Verfeinerung ist gegeben, wenn das abstrakte System $[\![S_a]\!]$ zur Schnittstellenverfeinerung $[\![S_r]\!]$ führt (siehe Abschnitt 3.4.2). Die abstrakten Modelle bilden dabei das Verhalten der Realisierungen vollständig ab:

$$S_a \overset{(R_{i\_1},R_{i\_2},\ldots,A_{o\_1},A_{o\_2},\ldots)}{\rightsquigarrow_{(true,true)}} S_r$$

Minimalität

Das wesentliche Ziel dieser Abstraktion ist die Reduktion der Zustandsmenge auf endlich viele diskrete Zustände. Das Verhalten soll dabei so wenig wie möglich abstrahiert werden. In der abstrakten Verhaltensbeschreibung sind deshalb nur Tupel zugelassen, welche auch in der Realisierung vorhanden sind. Es gilt:

$$S_a = (R_{i\_1} \otimes R_{i\_2} \otimes \ldots) \succ S_r \succ (A_{o\_1} \otimes A_{o\_2} \otimes \ldots)$$

bzw.

$$[\![S_a]\!] = ([\![S_r]\!] \wedge [\![R_{i\_1}]\!] \wedge [\![R_{i\_2}]\!] \wedge \ldots \wedge [\![A_{o\_1}]\!] \wedge [\![A_{o\_2}]\!] \wedge \ldots\})$$

Intervall-abstraktion

Ein geeigneter Abstraktionsmechanismus ermöglicht die systematische Ableitung endlicher Verhaltensbeschreibungen (vgl. Anhang E). Hier wird auf die Abstraktion mittels der Aufteilung des Wertebereichs in Intervalle genauer eingegangen:

Definition 6.1.3 (Intervalabstraktion)
Das Abstraktionsverhältnis zwischen einem abstrakten Kanal k_a und einem verfeinerten Kanal k_r ist eine Intervalabstraktion, wenn gilt:

Der statische Datentyp des abstrakten Kanals ist ein endliches Interval:

$$W_{k_a} = [0..n] \subset \mathbb{N}_0$$

Der statische Datentyp des verfeinerten Kanals ist Teilemenge einer total geordneten Menge:

$$W_{k_r} \subseteq \mathbb{R}$$

Die Menge der Elemente im Datentyp von k_a ist kleiner als die Menge der Werte im Datentyp von k_r:

$$\#W_{k_a} < \#W_{k_r}$$

Gegeben sei eine Intervalaufteilung $\mathsf{ABS}_k(i)$, welche die in W_{k_r} enthaltenen Werte vollständig partitioniert:
$ABS_k : W_{k_a} \rightarrow \mathcal{P}(W_{k_r})$
$\mathsf{ABS}_k(i) = \{(0, \S_1 x_1..x_2 \S_2), (1, \S_3 x_3..x_4 \S_4), \ldots, (n, \S_{2n-1} x_{2n-1}..x_{2n} \S_{2n})\}$
mit $x_{i-1} \leq x_i$ und $\S_i \in \{'[', ']'\}$ und $W_{k_r} = \bigcup_{i \in [0..n]} \mathsf{ABS}_k(i)$

Die Abstraktions- bzw. Repräsentationsrelation des abstrakten Kanals k_a zu einem verfeinerten Kanal k_r ist definiert durch:

$$[\![R_k]\!]^{op} = [\![A_k]\!]^{op} = k_r \in ABS_k(k_a)$$ ⌟

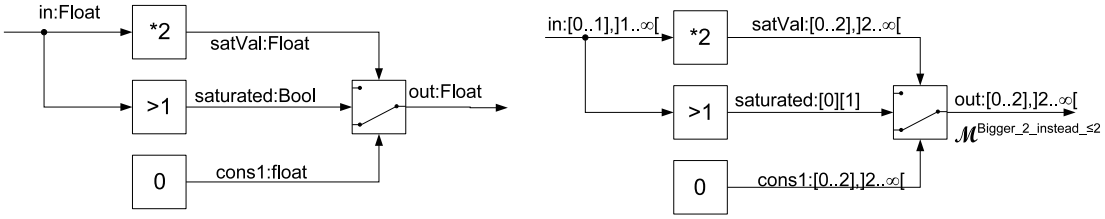


Abb. 6.3: Intervalabstraktion eines Systems

Verfahren

Theoretisch können für einen Kanal beliebige Intervallabstraktionen ausgewählt werden. Sinnvoll ist hier jedoch eine maximale Reduktion der Anzahl der Intervalle bei einer gleichzeitig maximal zu erhaltenden Aussagemöglichkeit über Fehlerwirkungen. Eine Möglichkeit ist eine Modellbildung, welche abhängig von den zu beobachtenden Folgefehlern und den in den Systemen vorhandenen Operatoren die geeigneten Intervalle durch eine Rückwärtsverfolgung des Signalflusses ermittelt. Abbildung 6.3 zeigt zu einer Realisierung (links) eine Abstraktion (rechts) zur Ermittlung einer Fehlerfolge, von der dann rückwärts die Intervalle zu den einzelnen Kanälen bestimmt werden. Eine weitere Auflistung zu den Abstraktionen ist in Anhang E zu finden.

6.1.4.2 Umgang mit nichtdeterministischen Systemen

Mächtigkeit der Tools

Bei der Abstraktion können nichtdeterministische Systeme entstehen. Bei der Addition kann z.B. wie in Abbildung 6.4.a nicht gesagt werden, ob das Ergebnis im Interval [0..1] oder]1..2] liegt. Eine große Menge an Verifikationsprogrammen können damit umgehen. Es gibt allerdings auch eine Reihe von Verifikationswerkzeugen, welche auch gleichzeitig Codegeneratoren sind (z.B. Scade^{TM}), und mit nichtdeterministischen Systemen nicht umgehen können. In diesem Fall müssen die Systeme zu deterministischen Systemen umgewandelt werden.

Teilmengen

Die nichtdeterministischen Systeme können mit einer Teilmengenkonstruktion (siehe [Bro98], Band 2, S. 232) zu deterministischen Systemen umgewandelt werden, in denen die maximale Erreichbarkeit angegeben wird. Es gilt:

$$[\![S_d]\!] = (o_d = \{(o_{a\_1}, o_{a\_2,\ldots}) | [\![S_a]\!] \wedge (i_{a\_1}, i_{a\_2}, \ldots) \in i_d\})$$

Kombinatorik

Die Teilmengenkonstruktion wird hinsichtlich der Komposition von Systemen und der großen Tupelmenge in der Verhaltensrelation schnell unhandlich. Bei manchen Anfragen spielt die Kombinatorik zwischen den Eingangssignalen und den Ausgangssignalen hinsichtlich des Nichtdeterminismus keine Rolle (siehe Abschnitt 5.2.1). In diesen Fällen kann die Beschreibung der Abhängigkeiten deutlich vereinfacht werden, in dem nicht mehr die Potenzmenge der Eingangskombinationen auf die Potenzmenge der Ausgangskombinationen abgebildet wird, sondern jeweils von einer

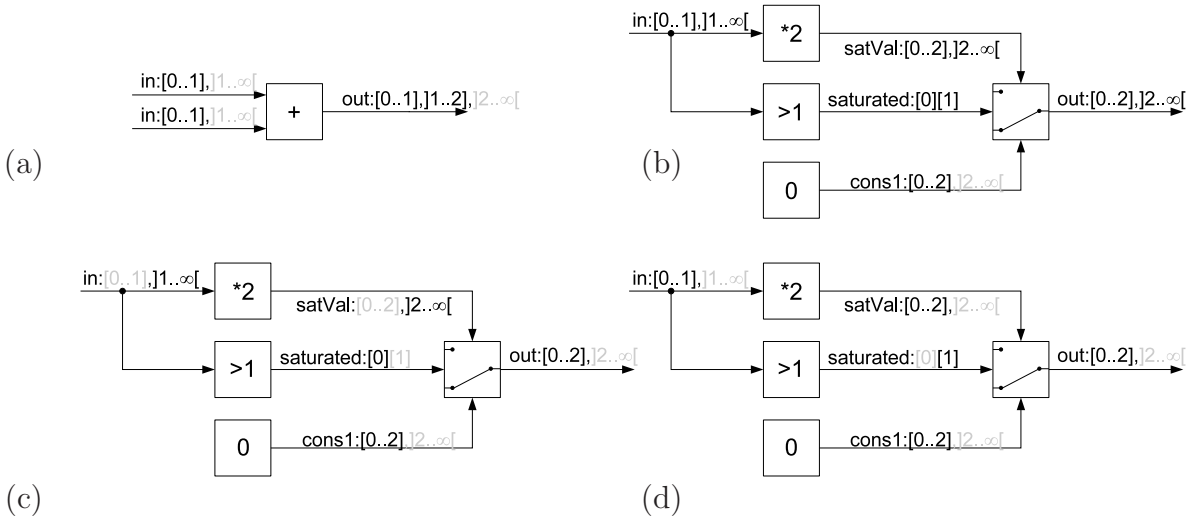


Abb. 6.4: Kombinatorik bei der Abstraktion

Kombination der Potenzmengen der einzelnen Eingangskanäle auf die Potenzmengen der einzelnen Ausgangskanäle. In diesem Falle gilt:

$$[\![S_{d\_ind}]\!] = \\ ([\![S_d]\!] \wedge i_{d\_ind\_1} = \bigcup_{x \in i_d} \pi_1(x) \\ \wedge i_{d\_ind\_2} = \bigcup_{x \in i_d} \pi_2(x) \wedge \ldots \wedge \bigcup_{x \in o_d} \pi_1(x) \wedge \bigcup_{x \in o_d} \pi_2(x) \wedge \ldots)$$

Beispiel

Diese starke Vereinfachung der Systeme, in der von der Kombinatorik abstrahiert wird, ignoriert die Fehleraufhebung, bzw. den kombinatorischen Ausschluss von erreichbaren Werten. Abbildung 6.4 veranschaulicht dieses Beispiel: Für ein System (b) sind beide Werte als Eingangswerte möglich. Wird nur die Potenzmenge einzelner Signale betrachtet, so sind in dem System beide Ergebnisse möglich. Es wird also eine schlechte Worst-Case-Abschätzung geliefert. Bei Betrachtung der Kombinatorik, bei der die Werte einzeln betrachtet werden, ist das gesamte System pro nichtdeterministischer Alternative zu betrachten (Abb. 6.4.c und 6.4.d). So kann die Wirkung feiner eingegrenzt werden.

Hilfskanäle

Es bietet sich für diese Systeme als zweite Möglichkeit die Methode an, zu den Komponenten einen weiteren Eingangskanal hinzuzufügen, dessen Wert beliebig sein kann und anhand dessen entschieden wird, welches Tupel aus der Relation im Falle einer nichtdeterministischen Entscheidung entnommen wird.

Beispiel 6.1.2 (nichtdeterministische Hilfsvariable)
Gegeben sei eine Summenfunktion mit der Intervalabstraktion aus Abbildung 6.4.a.

Das Verhalten des Systems ist:
$Sum_a : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1, 2\}$
$R_{Sum_a} = \{(0, 0, 0), (0, 0, 1), (0, 1, 1), (0, 1, 2), \ldots\}$

Für die Eingabe $(0,0)$ kommt sowohl die Ausgabe 0 wie auch die Ausgabe 1 in Frage, bzw. für die Eingabe $(0,1)$ die Ausgaben 1 und 2, usw. Die maximale Anzahl an nichtdeterministischen Ausgaben pro Eingabe ist hier 2. Durch Einführen einer zweiwertigen Zufallsvariable ergibt sich folgende deterministische Komponente:

$$Sum_d : \{0,1\} \times \{0,1\} \times \{a,b\} \to \{0,1,2\}$$
$$R_{Sum_d} = \{(0,0,a,0),(0,0,b,1),(0,1,a,1),(0,1,b,2),\ldots\}$$ ⌟

Analysemodelle

Dieser Abschnitt hat mit der Abstraktion bereits einen Weg skizziert, die Modelle zu vereinfachen und abzuändern, um die Analyseverfahren anzuwenden. Ein weiterer Schritt ist die Bildung von Modellen, aus denen die Zusammenhänge bei Simulationen direkt ablesbar sind. Zu diesen Verfahren gehören die Abweichungsmodellbildung aus [Str04] und die Diagnosemodelle aus [SSL+95, SSL+96]. Die Abweichungsmodellbildung ist ein Verfahren, bei der für ein bestimmtes Szenario ein Modell gebildet wird, welches als Signalwerte Abweichungen vom Sollwert enthält. Simuliert man dieses Modell mit Abweichungen als Eingaben, so lassen sich an den Ausgaben die Folgeabweichungen ablesen. Diagnosemodelle für Automaten werden aus Automaten gebildet, welche fehlerhafte Transitionen enthalten. Sie sind akzeptierende Automaten, in denen die Ausgaben des fehlerhaften Automaten als Eingabe dienen und bei fehlerhaften Ausgaben die verursachende Transition angeben.

6.2 Spezifische Umsetzungen mit Analysewerkzeugen

Dieser Abschnitt skizziert, wie mit den Analysetechniken aus Abschnitt 6.1 die Informationen für Sicherheitsanalysen (Kapitel 2) teilweise automatisch generiert werden können. Überprüft werden hierbei die in Kapitel 5 definierten Zusammenhänge. Die Gliederung dieses Abschnitts orientiert sich an den Phasen einer Sicherheitsanalyse gemäß Abschnitt 2.3. Es wird zuerst die Generierung der Information zu Systemstruktur, Anforderungen und Fehlern betrachtet, dann die Zusammenhänge zwischen Anforderungen und schließlich die Zusammenhänge zwischen Fehlern beleuchtet.

6.2.1 Systemstrukturbaum, Anforderungs- und Fehlerzuordnung

Strukturbaum

Die Erstellung des Systemstrukturbaums und die Anforderungs- und Fehlerzuordnung fällt dort, wo sie automatisiert werden kann, in den Bereich der syntaktischen Analyse. Grundlage für die Ermittlung des Systemstrukturbaums ist ein Systemmodell, in dem die Komponentenhierarchie enthalten ist. Der Systemstrukturbaum kann daraus direkt entnommen werden und in die entsprechenden Formate der Zielplattformen übernommen werden. Sind in dem Systemmodell Hilfskomponenten

enthalten, welche z.B. zur Wartung oder Codegenerierung hinzugefügt wurden, so können diese zur besseren Lesbarkeit aus dem Systemstrukturbaum herausgefiltert werden. Die in der Fallstudie angewendete Filterung wird in Anhang A vorgestellt. In einigen Fällen stimmen die Bezeichnungen der Subsysteme nicht mit den für die Fehlerauswirkungsanalyse gewünschten Bezeichnern überein. In diesen Fällen können noch Ersetzungstabellen hinzugezogen werden.

Anforderungszuordnung

Auch die Anforderungszuordnung erfolgt aus der syntaktischen Analyse. Hier liegen für die Verhaltensbedingungen und Qualitätsanforderungen meist in Tabellen Informationen über die Zugehörigkeit zu den Modulen vor. Ebenso können aus den Modellen die zugehörigen Ausgabekanäle ermittelt werden und die Beschreibungen der Signale aus Tabellen entnommen werden. Insgesamt ist diese Aufgabe bei Vorhandensein der Informationen weitgehend automatisch zu bewältigen. Manueller Bedarf besteht nur dann, wenn zusätzliche Qualitätsanforderungen aus den Implementierungsmodellen abgeleitet werden. Hier müssen dann meist zur Testbarkeit der Anforderungen noch Metriken und Grenzen in Vorlagen eingesetzt werden.

Fehlerzuweisung

Die Fehlerzuweisung kann über eine syntaktische Analyse in Zusammenhang mit einem Zuweisungsregelwerk vorgenommen werden. Hierzu können unter Anderem die Implementierungsmodelle, Informationen über Wertebereiche der Signale und deren Datentypen zu Hilfe genommen werden. Zusätzlich können Abstraktionsmechanismen mit in die Zuweisung einbezogen werden, um aussagekräftige Schnitte zwischen den Fehlerbildern zu ziehen. Während die Zuordnung der Anforderungen einheitlich für alle Sicherheitsanalysen ist, hängt die Zuordnung der Fehler stark von den Parametern im Zuweisungsregelwerk ab. Die Einstellungen berücksichtigen die in Kapitel 4 und 5 gewünschten Eigenschaften der Fehlerauswirkungsanalyse. Besonders mit dem Hintergrund der formalen Bedeutung der Fehler können hier unterschiedliche Fehlermodelle zugeordnet werden. Da die Fehlerzuweisung generisch für ein Modell erfolgt, ist es meistens notwendig, während der Analyse die Zuordnung der Fehler manuell zu überarbeiten. Eine genaue Beschreibung des Zuweisungsregelwerks der Fallstudie ist in Anhang A zu finden.

6.2.2 Anforderungsabhängigkeit

Die Ermittlung der Anforderungsabhängigkeit ist spezifisch für die jeweilige Art der Anforderung. Im Folgenden werden Möglichkeiten der Zusammenhangsermittlung für Verhaltensbedingungen, Qualitätsanforderungen und Ausgabekanalbeschreibungen vorgestellt.

Verhaltensbedingungen

Die formale Bedeutung einer Abhängigkeit zwischen Anforderungen wurde in Definition 5.2.2 vorgestellt. Es ist zu prüfen, ob diese Bedingung eingehalten wird. Diese Überprüfung findet mit booleschen Analysen modulo Theorien statt. Die Formel kann so mehr oder weniger direkt in die Analysewerkzeuge eingegeben werden. Wichtig ist hier, dass das Analysewerkzeug mächtig genug ist, um die Anfrage verarbeiten zu können. Die Formel der Funktionsabhängigkeit ist von sich aus bereits mindestens eine First-Order-Formel. Bei einfachen Spezifikationen kann diese Formel

durch Negation zu einer Formel auf Ebene der Aussagenlogik reduziert werden. Als Hilfsmittel kann über Tests bzw. Prüfung auf Einfachfehler-Abhängigkeiten zumindest eine Vorauswahl der gefundenen Anforderungsabhängigkeiten gegeben werden. Häufig müssen hier jedoch die Abhängigkeiten von Hand modelliert werden.

Ausgabekanäle

Die Beschreibungen des Verhaltens anhand von zu Ausgaben aufgelösten Gleichungen, wie sie in Definition 3.3.5 vorgestellt wurden, sind eine Sonderform der Verhaltensbedingungen. Zur exakten Ermittlung der Abhängigkeiten gelten die gleichen Bedingungen wie bei den Verhaltensbedingungen. Allerdings besteht hier im Vergleich zu den Verhaltensbedingungen die Möglichkeit, die Ausgabeabhängigkeiten mit einer syntaktischen Analyse abzuschätzen. Da die Gleichungen immer zu den Ausgaben hin aufgelöst sind, kann der Signalfluss als ein Indiz zur Abschätzung der Zusammenhänge verwendet werden. Der zugrundeliegende gerichtete Graph ist hierbei identisch mit dem Signalfluss. Die Schwäche dieser Analyse ist, dass Tautologien und sinnlose Operationen wie Multiplikation mit 0 nicht erkannt werden und so nur eine obere Schranke für die Abhängigkeiten angegeben werden kann. Wird statt einem Komponentenmodell wie Simulink das Verhalten mit einer Programmiersprache wie C beschrieben, so kann durch den Einsatz von Variablen, welche in Prozeduren mehrmals verwendet werden, die Abschätzung nochmals schlechter werden. Eine Beschreibung der Ermittlung der Zusammenhänge in der Fallstudie erfolgt in Anhang B.

Qualitätsanforderungen

Die Ermittlung von Zusammenhängen nichtfunktionaler Anforderungen ist mit Verhaltensanalysen prinzipiell nicht möglich. Hier kann die Unterstützung mit Werkzeugen nur bedingt erfolgen. Der einzig verwertbare Teil des Verhaltensmodells ist der Strukturbaum, entlang dessen die Anforderungsabhängigkeiten laufen müssen. Letzten Endes werden im einfachen Fall die Anforderungen immer eine Ebene höher weitergereicht. Komplexere Zusammenhänge müssen fast immer manuell modelliert werden.

6.2.3 Fehlerabhängigkeit

Qualitätsanforderungen

Die Fehlerabhängigkeit zu Qualitätsanforderungen ist bei einem gegebenen Anforderungsnetz meist trivial. Fast immer werden die Fehler zu der Anforderung auf nächst höherer Ebene weitergereicht. Besonderheiten müssen dann manuell nachgezogen werden.

Verhalten

Schwieriger gestaltet sich die Ermittlung der Abhängigkeiten bei Verhaltensbedingungen und aufgelösten Gleichungen. Bei beiden muss das Verhalten analysiert werden. Ähnlich wie bei der Vernetzung der Verhaltensbedingungen stoßen die Verifikationswerkzeuge schnell an ihre Grenzen. Deshalb werden in diesem Abschnitt zwei Ansätze vorgestellt. Der einfache Ansatz ist das Ausweichen auf eine manuelle Modellierung der Zusammenhänge. Diese Modellierung kann durch Testfälle und Simulationen unterstützt werden. Der alternative Ansatz ist, wenn möglich die Anfragen an die Werkzeuge so weit zu vereinfachen, dass die Zusammenhänge automatisch ermittelt werden können. Im Folgenden werden zwei Wege vorgestellt, die Anfragen

für die Werkzeuge passend zu gestalten. Der eine Weg besteht in der passenden Formulierung der Anfragen und einer Beschränkung der Modifikationen, welche als Folge modelliert werden können. Der andere Ansatz besteht in der Dekomposition des Systems und der Komposition der Auswirkungsmodelle, mit der Einschränkung auf bestimmte Architekturmuster.

6.2.3.1 Aussagenlogische Verifikation

lokale Variablen

Die bisher in der Arbeit gegebenen Definitionen gelten für Blackbox-Spezifikationen, in denen keine lokalen Variablen vorkommen. Die Verfeinerungsbeziehung soll anhand der externen Variablen bestimmt werden, wobei von internen Variablen abstrahiert werden soll. Sowohl im Abstrakten, wie auch im konkreten System müssen lokale Variablen behandelt werden können. Beispiel 6.2.1 verdeutlicht diesen Unterschied. Hierzu wird die Menge O_S eines Systems S aufgeteilt in eine Menge lokaler Variablen O_S^l und externer Variablen O_S^e. Die Implikationsbeziehung zwischen den Systemen entspricht dann keiner Verfeinerung mehr, sondern einer Simulations- bzw. Bisimulationsbeziehung (siehe [Mil89]) hinsichtlich der extern sichtbaren Variablen. Möchte man den Parallelvergleich aus Abschnitt 5.2.1.1 verwenden, so gilt:

$$[\![S]\!] \Rightarrow \exists O_T^l : [\![T]\!] \qquad \text{mit} \quad O_S^l \cap O_T^l = \emptyset \quad \text{und} \quad I_S = I_T \quad \text{und} \quad O_S^e = O_T^e$$

Beispiel 6.2.1 (lokale Variablen - Simulation)
Gegeben sei ein System S und ein System T. Zur Vereinfachung seien die Systeme rückkopplungsfrei mit Funktionen definiert als:

$$[\![S]\!] = (O_S = F_1(F_2(I_S))) \text{ und } [\![T]\!] = (O_T = F_3(F_4(I_T)))$$

Die Funktionen $F_1, \ldots, F_4$ seien verschieden:

$$\forall i, j \in \{1, \ldots, k\} : \exists x : F_i(x) \neq F_j(x)$$

Es gelte aber:

$$[\![S]\!] \Leftrightarrow [\![T]\!]$$

Das gleiche Verhalten der Systeme kann auch mittels zusammengesetzter Komponenten beschrieben werden, die Verfeinerungen der Systeme $[\![S]\!]$ und $[\![T]\!]$ sind:

$$[\![S^{\otimes}]\!] = (O_{S^{\otimes}}^e = F_1(O_{S^{\otimes}}^l) \wedge O_{S^{\otimes}}^l = F_2(I_{S^{\otimes}})) \text{ und}$$
$$[\![T^{\otimes}]\!] = (O_{T^{\otimes}}^e = F_3(O_{T^{\otimes}}^l) \wedge O_{T^{\otimes}}^l = F_4(I_{T^{\otimes}}))$$

Da die Funktionen verschieden sind, können die Werte der lokalen Variablen $O_{S^{\otimes}}^l$ und $O_{T^{\otimes}}^l$ nicht übereinstimmen. Damit gilt:

$$[\![S^{\otimes}]\!] \not\Leftrightarrow [\![T^{\otimes}]\!]$$

Nach aussen hin zeigen die Systeme jedoch weiterhin das gleiche Verhalten. Damit gilt:

$$[\![S^{\otimes}]\!] \cong_{I_S \cap O_S} [\![T^{\otimes}]\!]$$ ⌟

Effizienz

Mit den lokalen Kanälen sind in den zu prüfenden Formeln gebundene Variablen. Die Anfragen gehen somit über die Aussagenlogik hinaus zur Prädikatenlogik (First Order Logic). In einer verkürzten Schreibweise entspicht der Existenzquantor einer Oder-Verknüfung mit allen möglichen Belegungen der lokalen Variablen:

$$[\![S]\!] \Rightarrow \bigvee_{\beta \in O_T^l} [\![T]\!]$$

Die so gestellte zu prüfende Aussage ist deutlich aufwändiger zu verifizieren. SAT-Probleme liegen im Bereich von $O(NP(n))$. Das so erstellte Problem liegt allerdings aufgrund der Oder-Verknüpfung der lokalen Variablenbelegungen im Bereich von $O(NP(n) \cdot 2^l)$. Bei großen Systemen wächst die Rechenzeit hierbei schnell an. Universelle Möglichkeiten zur effizienten Lösung dieser Probleme sind noch nicht bekannt. Ein automatisierter Ansatz zur Lösung dieser Probleme ist die Resolution [CL71]. Dieser Ansatz widerlegt die Allgemeingültigkeit einer Aussage, in dem die Unerfüllbarkeit deren Negation gezeigt wird.[1] Das Verfahren ist von der Effizienz her bei $O(NP(n))$, terminiert aber nicht zwingend. Weitere Verfahren zum Prüfen der Bisimulation bzw. Simulation von Büchi-Automaten sind in [EWS05] zu finden. Diese verifizieren die Eigenschaften sogar in $O(n^{10})$.

externe Modifikation

Die Berechnungszeit zur Verifikation der Prädikate ist, wie eben dargestellt relativ lang. Für einige Modifikationen können die zu verifizierenden Aussagen deutlich verkürzt und vereinfacht werden. Die zu überprüfende Wirkung der Modifikation $\mathcal{M}_{sub}$ wird mit einer Modifikation $\mathcal{M}_{super}$ verglichen, welche auf das Gesamtsystem und nicht auf ein Teilsystem angewendet wird. Einige dieser Modifikationen sind so gestaltet, dass sie sich, wie in Abschnitt 4.4, Variante II gezeigt, durch Anhängen einer Modifikationskomponente an die Funktionskomponente gestalten lassen, wobei die Modifikationskomponente keine lokalen Variablen hat. Zu diesen Modifikationen gehören unter anderem die in Abschnitt 4.2.2 beschriebenen Modifikationen zu aufgelösten Gleichungen. Diese Modifikationen können als Kontrollkriterium für eine obere Schranke der Auswirkung genutzt werden. Es gilt:

$$[\![S]\!]\Delta[\![\mathcal{M}_{super}]\!] = [\![S']\!] \wedge [\![M_{\mathcal{M}_{super}}]\!] \quad \text{mit} \quad [\![S']\!] = [\![S']\!][O_S^{e'}/O_S^e] \quad \text{und} \quad O_S^e = O_{M_{\mathcal{M}}}^e$$

Die Variablen des Systems $[\![S]\!]$ werden zu den Eingangsvariablen der Komponente $[\![M_{\mathcal{M}_{super}}]\!]$ umbenannt. Die Ausgabevariablen der Modifikationskomponente sind die Ausgabevariablen der Komponente $[\![S]\!]$. Mithilfe dieses Konstrukts lässt sich eine Formel herleiten, welche allgemeingültig sein muss, um eine Fehlerwirkung hinsichtlich hinzugefügter Pfade zu akzeptieren:

[1] Die negierte Formel mit den Quantoren wird in die bereinigte Pränexform gebracht, anschließend in die Skolemform transformiert, welche erfüllbarkeitsäquivalent ist und schießlich die leere Klausel mittels Resolution abgeleitet.

$$
\begin{aligned}
&[\![T]\!] \wedge [\![S']\!] \Rightarrow [\![M_{\mathcal{M}^F_{super}}]\!] \\
&\quad \Leftrightarrow (\neg([\![T]\!] \wedge [\![S']\!]) \vee [\![M_{\mathcal{M}^F_{super}}]\!]) \\
&\quad \Leftrightarrow (\neg[\![T]\!] \vee \neg[\![S']\!] \vee [\![M_{\mathcal{M}^F_{super}}]\!]) \\
&\quad \Leftrightarrow (\neg[\![T]\!] \vee (\neg[\![S']\!] \vee [\![M_{\mathcal{M}^F_{super}}]\!])) \\
&\quad \Leftrightarrow ([\![T]\!] \Rightarrow ([\![S']\!] \Rightarrow [\![M_{\mathcal{M}^F_{super}}]\!])) \\
&\quad \Rightarrow ([\![T]\!] \Rightarrow ((\exists O_S^{e'} : [\![S']\!]) \wedge ([\![S']\!] \Rightarrow [\![M_{\mathcal{M}^F_{super}}]\!]))) \qquad //S \textit{ ist total} \\
&\quad \Rightarrow ([\![T]\!] \Rightarrow (\exists O_S^{e'} : ([\![S']\!] \wedge [\![M_{\mathcal{M}^F_{super}}]\!]))) \\
&\quad \Rightarrow ([\![T]\!] \Rightarrow (\exists O_S^{e'} : [\![S]\!]\Delta[\![\mathcal{M}^F_{super}]\!]))
\end{aligned}
$$

obere Schranke

Mit dieser Definition der Auswirkungsbestimmung ist implizit gegeben, dass auch Abweichungen der Ausgaben, welche aus dem Nichtdeterminismus der Systeme folgen, mit enthalten sind. Eine Modifikation $[\![\mathcal{M}_{super}]\!]$ kann also im Falle nichtdeterministischer Komponenten bei dieser Formulierung der Auswirkung nicht leer sein. Verdeutlicht wird dies in Beispiel 6.2.2. Dieser Effekt ist hinsichtlich einer Fehlerauswirkungsanalyse sogar wünschenswert. Er zwingt dazu, explizit zu berücksichtigen, dass weitere Verfeinerungen der Systeme innerhalb des gegebenen Nichtdeterminismus voneinander abweichen können und dies zu unerwünschten Fehlern führen kann. Umgekehrt zwingt es die Entwickler zum Finden geeigneter Abstraktionen um bei Auswirkungsanalysen in abstrakten Modellen eine begrenzte Wirkung nachzuweisen.

Beispiel 6.2.2 (Ausgabenvergleich als obere Schranke der Wirkung)
Gegeben seien zwei Systeme $[\![S]\!]$ und $[\![T]\!]$, welche beide für eine Eingabebelegung $\beta.I_S = \beta.I_T = 1$ nichtdeterministisch verschiedene Ausgaben $\alpha.O_S \in \{1,2\}$ und $\alpha.O_T \in \{1,2\}$ liefern können. Das System $[\![S]\!]$ enthalte die Modifikation $\mathcal{M}_{sub}$, welche in diesem Beispiel die leere Modifikation ist. Obwohl die leere Modifikation tatsächlich nur die leere Modifikation $\mathcal{M}_{super}$ zur Folge haben kann, muss die Komponente auch akzeptieren, wenn eine 1 statt einer 2 kommt und umgekehrt, da nur dann gilt:

$$[\![S]\!] \wedge [\![T]\!] \wedge I_S = I_T = 1 \Rightarrow [\![M_{\mathcal{M}_{super}}]\!]$$

mit $[\![M_{\mathcal{M}_{super}}]\!] = (O_S = O_T) \vee (O_S + O_T = 3)$ ⌟

Teilfolge

Das Gegenstück zu den Beziehungen, in denen sichergestellt wird, dass eine Folge maximal abgefangen ist, ist die Betrachtung der Angabe einer minimalen Folge. Dies kann z.B. bei der in Abschnitt 5.1.2 gezeigten Angabe notwendig sein, um sicherzustellen, dass eine Folge vorhanden ist, z.B. einem Fail Safe Verhalten entspricht. Es soll gelten:

$$S_1\Delta\mathcal{M}_{sub} \overset{\geq}{\dashrightarrow} S\Delta\mathcal{M}_{super}$$

Feststellung

Die Überprüfung dieser Teilfolgenbeziehung ist das Spiegelbild zu der Teilmengenbeziehung, in der eine obere Schranke für die Folge geprüft wird. Hier gilt, dass jedes durch $\mathcal{M}_{super}$ hinzugekommenes Tupel auch tatsächlich in $\mathcal{M}_r$ ist und jedes durch $\mathcal{M}_{super}$ entferntes Tupel auch von $\mathcal{M}_r$ entfernt wird:

$$
\begin{aligned}
&([\![S_1]\!]\Delta[\![\mathcal{M}_{sub}]\!] \overset{\geq}{\dashrightarrow} [\![S]\!]\Delta[\![\mathcal{M}_{super}]\!]) \\
&\quad \Leftrightarrow (([\![S]\!]\Delta(true, [\![F_{super}]\!]) \Rightarrow [\![S]\!]\Delta(true, [\![F_r]\!])) \\
&\qquad \wedge([\![S]\!]\Delta([\![E_r]\!], false) \Rightarrow [\![S]\!]\Delta([\![E_{super}]\!], false)))
\end{aligned}
$$

externe Modifikation

Auch hier können durch die Verwendung externer Modifikationskomponenten hinsichtlich der Wirkungsanalyse die Anfragen zur Aussagenlogik reduziert werden. Im Gegensatz zu der Überprüfung der hinzugefügten Tupel, wird hier das Entfernen von Tupeln untersucht. Die Modifikationskomponente, welche die Folge beschreibt, kann allerdings nicht einfach als eine Modifikation $(E_{super}, false)$ beschrieben werden, da die mit F_{sub} hinzugefügten Modifikationen ja auch abgefangen werden müssen. Entsprechend wird hier auf eine Folgemodifikation hin untersucht, in welcher nur die durch E_{super} gestrichenen Tupel fehlen:
$[\![\mathcal{M}^{E'_{super}}]\!] = ([\![\overline{E}_{super}]\!], [\![\overline{E}_{super}]\!])$

Es gilt:

$$
\begin{aligned}
&[\![T]\!] \wedge [\![S']\!] \Rightarrow [\![M_{\mathcal{M}^{E'}_{super}}]\!] \\
&\quad \Leftrightarrow ([\![T]\!] \Rightarrow ([\![S']\!] \Rightarrow [\![M_{\mathcal{M}^{E'}_{super}}]\!])) \\
&\quad \Rightarrow ([\![T]\!] \Rightarrow ((\exists O^{e'}_S : [\![S']\!]) \wedge ([\![S']\!] \Rightarrow [\![M_{\mathcal{M}^{E'}_{super}}]\!]))) \qquad //S \textit{ ist total} \\
&\quad \Rightarrow ([\![T]\!] \Rightarrow (\exists O^{e'}_S : [\![S]\!] \Delta [\![\mathcal{M}^{E'}_{super}]\!]))
\end{aligned}
$$

Jede minimale Modifikation $([\![\overline{E}_r]\!], [\![\overline{F}_r]\!])$, welche als tatsächliche Folge in Frage kommt, kann weniger gültige Tupel akzeptieren, d.h. die so untersuchte Menge E_{super} ist minimal.

Teilüberschneidung der Wirkung

Wie in der Fallstudie in Kapitel 2 zu sehen, kann in einer Fehlerauswirkungsanalyse wie einer FMEA, statt einem übergreifenden Folgefehler eine Menge an Folgefehlern angeben werden, welche in Summe die komplette mögliche Wirkung umspannt (siehe Abschnitt 2.3). In diesem Rahmen wird nicht überprüft, ob zu einem Folgefehler die Teilmengenbedingung erfüllt ist, sondern ob sich die Modifikationen überschneiden, also gilt:

$$S_1 \Delta \mathcal{M}_{sub} \overset{\otimes\leq}{\dashrightarrow} S \Delta \mathcal{M}_{super}$$

Die Ermittlung, ob sich zwei Modifikationen überschneiden, ist zum Einen gegeben, wenn von einer Modifikation ein Tupel hinzugefügt wird, dass nicht im Sollverhalten ist und aber trotzdem in der zweiten Modifikation enthalten ist. Zum anderen überschneiden sich zwei Modifikationen, wenn Pfade sowohl in der einen Modifikation, wie auch in der anderen Modifikation entfernt werden. Die Bestimmung dieser Abhängigkeit lässt sich einfacher anhand der Negation überprüfen. Es gilt:

$$
\begin{aligned}
&S_1 \Delta \mathcal{M}_{sub} \overset{\otimes\leq}{\not\dashrightarrow} S \Delta \mathcal{M}_{super} \\
&\Leftrightarrow (([\![S]\!] \Delta (true, [\![F_r]\!]) \Rightarrow (\neg [\![S]\!] \Delta (true, [\![F_{super}]\!]) \vee [\![S]\!])) \\
&\qquad \wedge ([\![S]\!] \Rightarrow ([\![S]\!] \Delta (true, [\![E_r]\!]) \vee [\![S]\!] \Delta (true, [\![E_{super}]\!]))))
\end{aligned}
$$

deterministische Komponenten

Auch bei dieser Formel ist das Problem der Bisimulation gegeben. In der Praxis zeigen sich gerade in der Automobilindustrie meist Modelle, welche in Simulink definiert sind. Diese Modelle sind nicht nur total, sondern auch deterministisch. Verifikationswerkzeuge wie Scade™ können allerdings nur deterministische Modelle verarbeiten. Geht man von diesem Umstand aus, so lässt sich die Formel zur Fehlerüberprüfung folgendermaßen auf die Aussagenlogik reduzieren:

$$
\begin{aligned}
&([\![T]\!] \wedge [\![S']\!]) \Rightarrow (\neg M_{\mathcal{M}^F} \vee M_{\mathcal{M}^\emptyset}) \\
&\quad \Leftrightarrow ([\![T]\!] \wedge [\![S']\!]) \Rightarrow (\neg M_{\mathcal{M}^F} \vee (M_{\mathcal{M}^\emptyset} \wedge [\![S']\!])) \\
&\quad \Leftrightarrow ([\![T]\!] \wedge [\![S']\!]) \Rightarrow (\neg M_{\mathcal{M}^F} \vee [\![S]\!]) \\
&\quad \Leftrightarrow \neg([\![T]\!] \wedge [\![S']\!]) \vee (\neg M_{\mathcal{M}^F} \vee [\![S]\!]) \\
&\quad \Leftrightarrow \neg[\![T]\!] \vee \neg[\![S']\!]) \vee \neg M_{\mathcal{M}^F} \vee [\![S]\!]) \\
&\quad \Rightarrow [\![T]\!] \Rightarrow \exists O_S^l : (\neg([\![S']\!] \wedge M_{\mathcal{M}^F}) \vee [\![S]\!])) \\
&\quad \Rightarrow [\![T]\!] \Rightarrow \exists O_S^l : (\neg([\![S']\!] \Delta \mathcal{M}^F) \vee [\![S]\!]))
\end{aligned}
$$

obere Schranke

Auch bei dieser mit Aussagenlogik prüfbaren Aussage ist für die neutrale Modifikation $\mathcal{M}^\emptyset$ eine geeignete maximale neutrale Modifikation zu wählen. Wird bei nichtdeterministischen Komponenten die kleinste neutrale Modifikationskomponente $M_{\mathcal{M}^\emptyset} = ID$ verwendet, so ist eine Fehlerbeziehung auch dann wahr, wenn diese durch den vorhandenen Nichtdeterminismus in verfeinerten Systemen entstehen könnte. Bei deterministischen Systemen reicht hier jedoch die Identität als neutrales Element aus.

6.2.3.2 Abstraktion von Modellteilen

Aufwandsreduktion

Werden die Zusammenhänge zwischen Fehlern isoliert für Anforderungen betrachtet, so kann durch die in den letzten Abschnitten entstehenden Abstraktionen der Kombinatorik häufig nur eine grobe Abschätzung gegeben werden. Insbesondere bei der manuellen Modellierung der Zusammenhänge ist eine Strategie hilfreich, um den Aufwand zur Ermittlung der Zusammenhangsmodelle gering zu halten. Ziel ist es, für einzelne Komponenten nicht gleich die gesamten Abweichungszusammenhangstabellen zu erstellen, sondern anhand geschickter Auswahl der Wirkungs-Pfade nur die relevanten Zusammenhänge zu modellieren. Um dies zu unterstützen, müssen die Zusammenhänge attributiert werden können. Im Folgenden wird eine Attributierung vorgestellt, anhand derer eine aufwandsreduzierte Methode entwickelt werden kann.

Attribute

Die in der Fallstudie vorkommenden Teilsysteme können als Teil sicherheitsrelevanter Systeme in drei Klassen eingeteilt werden: die *Funktionskomponenten*, welche die eigentliche Funktionalität berechnen, die *Überwachungskomponenten*, welche die diagnostizierbaren Fehler erkennen und an die *Steuerungskomponenten* weitergeben, welche die Berechnung der Funktion im Sinne der Sicherheit (Fail-Safe-Verhalten) steuern. Im allgemeinen ist für eine Analyse der Systeme die Abschätzung interessant, ob eine modellierte Folgemodifikation mindestens die echte Folge der Ursachenmodifikation umfasst, also $\overset{\leq}{\dashrightarrow}$. Führen die Überwachungs- zusammen mit den Steuerungskomponenten gezielt zu einem Fail-Safe-Verhalten, so muss die weitere Folge durch eine Funktion nicht verfolgt werden. Um diese zu modellieren, werden zwei weitere Zusammenhangsattribute eingeführt. Beide stellen für eine Folgemodifikation sicher, dass das abweichende Verhalten der F-Mengen gewählt wird.

Definition 6.2.1 (sicher beobachtbare Folge)
Gegeben sei ein System $S = S_1 \otimes S_2$ mit $S, S_1, S_2 \in \mathbb{S}$ und die minimalen Modifikationen $\mathcal{M}_S \in MOD_S$, $\mathcal{M}_{S_1} \in MOD_{S_1}$ und $\mathcal{M}_{S_2} \in MOD_{S_2}$.

Eine *sicher beobachtbare Folge* $\overset{\blacktriangleright}{\dashrightarrow}$ setzt sich zusammen aus einer oberen Schranke für die hinzugefügten Ausführungspfade und einer unteren Schranke für die entfernten Pfade.

$$(\mathcal{M}_{S_1}, \mathcal{M}_{S_2}) \overset{\blacktriangleright}{\dashrightarrow} \mathcal{M}_S$$
$$\overset{def}{\Leftrightarrow}$$
$$\exists \mathcal{M}_a \in MOD_S : ((R_{S_1}\Delta\mathcal{M}_{S_1} \otimes R_{S_2}\Delta\mathcal{M}_{S_2} = R_S\Delta\mathcal{M}_a) \wedge (F_S \rightsquigarrow F_a) \wedge (E_a \rightsquigarrow E_S))$$

⌟

Die Arten der sicheren Abweichung aus Def. 6.2.1 können nun, genau wie bei der anforderungsbezogenen Fehlerabhängigkeit in die Einfachfehlerabhängigkeit und die obere Schranke für Mehrfachfehlerabhängigkeit unterschieden werden. Diese sind definiert als:

Definition 6.2.2 (anforderungsbezogene sichere Einfachfehlerfolge)
Gegeben sei ein System $S = S_1 \otimes S_2$ mit $S, S_1, S_2 \in \mathbb{S}$.
Die Systeme seien beschrieben mit den disjunkten Prädikaten

$[\![S]\!] = \Phi_x \wedge \Phi_y \wedge ...$ und
$[\![S_1]\!] = \Phi_a \wedge \Phi_b \wedge ...$

Eine Modifikation einer Anforderung Φ_a hat dann eine Modifikation einer Anforderung Φ_x zur Folge, geschrieben als $\overset{||^1\blacktriangleright}{\dashrightarrow}$ oder $\overset{||^n\blacktriangleright}{\dashrightarrow}$, wenn gilt:

$$\mathcal{M}_a \overset{||^1\blacktriangleright}{\dashrightarrow} \mathcal{M}_x$$
$$\overset{def}{=} \exists \mathcal{M}_y, \mathcal{M}_z, \ldots : (\mathcal{M}_a, \mathcal{M}^{\emptyset}_{S_2}) \overset{\blacktriangleright}{\dashrightarrow} \sum^{\langle \Phi_x, \Phi_y, \ldots \rangle} \langle \mathcal{M}_x, \mathcal{M}_y, \ldots \rangle$$

$$\mathcal{M}_a \overset{||^n\blacktriangleright}{\dashrightarrow} \mathcal{M}_x$$
$$\overset{def}{=} \forall \mathcal{M}_b, \mathcal{M}_c, \ldots : \exists \mathcal{M}_y, \mathcal{M}_z, \ldots :$$
$$(\sum^{\langle \Phi_x, \Phi_y, \ldots \rangle} \langle \mathcal{M}_a, \mathcal{M}_b, \ldots \rangle, \mathcal{M}^{\emptyset}_{S_2}) \overset{\blacktriangleright}{\dashrightarrow} \sum^{\langle \Phi_x, \Phi_y, \ldots \rangle} \langle \mathcal{M}_x, \mathcal{M}_y, \ldots \rangle$$

⌟

Abweichungsbezug

Mit Hilfe dieser Abhängigkeitsmodelle kann nun durch Finden der sicheren Pfade die Untersuchung der Abhängigkeiten reduziert werden. Im Wesentlichen konzentriert sich diese Unterscheidung auf Modelle, bei denen Abweichungen in Zusammenhang gebracht werden, um daraus Folgefehlverhalten abzuleiten. In der Anwendung interessieren hier also meist nicht, wie in obigen Definitionen gegeben, die Zusammenhänge zwischen Fehlverhalten, sondern zwischen Abweichungen. Auch hinsichtlich der Zeit ist die Betrachtung der Zusammenhänge hinsichtlich Zeitfenster meist die verwendete Anwendung. Folgendes Beispiel verdeutlicht die sichere Folge.

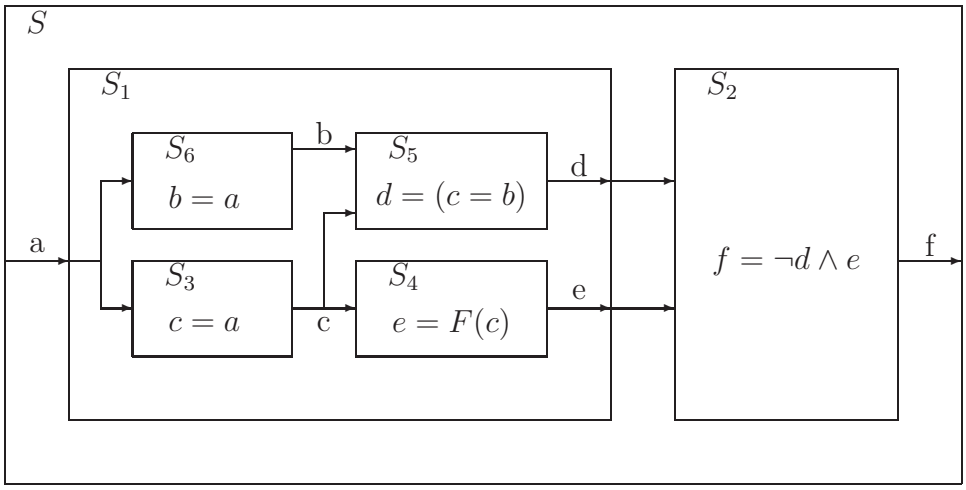


Abb. 6.5: Beispiel eine Systems mit Überwachung S_5 und Steuerung S_2 (Fail Safe: $d = false$)

Beispiel 6.2.3 (sichere Folge)
Gegeben sei das System in Abbildung 6.5. Die Teilsysteme S_3 und S_6 stehen für eine redundante Funktion (z.B. zwei Sensoren), deren Ergebnisse vom System S_5 überwacht werden. Erkennt dieses einen Fehler, so gibt die Steuerkomponente S_2 nur das Fail-Safe-Verhalten $false$ aus. Für dieses (in diesem Fall) zeitlose System gelten die Fehlermodelle:

$$\Psi_{S_2} = \{(\delta_d^{0\to1}, \overset{||^n\blacktriangleright}{\dashrightarrow}, \delta_f^{1\to0}\} \qquad \Psi_{S_5} = \{(\delta_c^{\neq\cdot}, \overset{||^1\blacktriangleright}{\dashrightarrow}, \delta_d^{0\to1}\} \qquad \Psi_{S_3} = \{(\delta_a^{\emptyset}, \overset{\leq}{\dashrightarrow}, \delta_c^{\neq\cdot}\}$$

Für das Steuerungsystem S_2, gelte, dass immer, wenn das Signal d auf $true$ geht, das System auf den Fail-Safe Wert $false$ geht. Für das Fehlermodell kann so modelliert werden, dass eine Abweichung zu $true$ an d unabhängig von allen anderen Eingaben immer zu einem $false$ an f führt. Für das Überwachungssystem S_5 führt eine kanalbezogene Einfachabweichung sicher zu einem gesetzten Kanal d. Um die Wirkung eines Fehlers in dem System S_3 zu erkennen, reicht es, diese Systeme zu untersuchen, ohne dabei das komplexe System S_4 untersuchen zu müssen. ⌟

6.3 Fallstudie

6.3.1 Struktur

Strukturbaum

Die automatisierte Erstellung des Systemstrukturbaums lief mit einer syntaktischen Analyse, in der eine Extraktion der Struktur aus einem Simulink-Modell stattfand. Zu berücksichtigen war hier die Einbindung der Modellierungsrichtlinien aus Anhang A. So fern die Modellierungsrichtlinien eingehalten wurden, war eine schnelle Generierung möglich. In der Praxis wurden gerade bei Zwischenzuständen in frühen

Phasen der Entwicklung unter anderem wegen Zeitdruck und starker Veränderungen des Systems die Modellierungsrichtlinien nicht immer exakt eingehalten. In diesen Fällen war häufig die extrahierte Struktur zu groß und entsprechende Korrekturen der extrahierten Struktur mussten vorgenommen werden. Das System umfasste in etwa 50 Module, wobei der Strukturbaum bis zu 4 Ebenen enthielt und durchschnittlich jedes zusammengesetzte System ca. 5 (Standardabweichung ca. 3) Subsysteme enthielt. In dem für die Fallstudie verwendeten Zwischenstand waren noch Nachbesserungen bei ca. 5 Systemen notwendig, die sich aus Abweichungen von den Modellierungsrichtlinien ergeben haben. Die Generierung der Struktur verlief in wenigen Sekunden. Für die Korrektur der Abweichungen war in etwa eine halbe Stunde notwendig. Eine manuelle Extraktion des Systemstrukturbaums hat bisher bei erstmaliger Erstellung nicht wesentlich mehr Zeit gekostet, allerdings funktioniert bei Änderungen des Modells der Abgleich im automatisierten Fall dann wesentlich schneller. Bei strikter Einhaltung der Modellierungsrichtlinien ist eine starke Steigerung der Effizienz zu erwarten.

Anforderungszuweisung

Die Zuweisung der Anforderungen zu den Elementen des Systemstrukturbaums konnte nur bedingt automatisiert über eine syntaktische Analyse realisiert werden. Es wurden Verhaltensbedingungen und Qualitätsanforderungen aus den Spezifikationen in Word zugewiesen und Ausgabebeschreibungen aus dem TargetLink™ Data-Dictionary extrahiert. Eine genaue Beschreibung des Vorgehens findet sich in Anhang A. Die in der Fallstudie realisierte Verknüpfung der Elemente des Systemstrukturbaums und der Anforderungen umfasste 520 Ausgabebeschreibungen, 859 Verhaltensbedingungen bzw. Qualitätsanforderungen und zu diesen 1648 Testfälle, die zugewiesen wurden. Theoretisch fordert dies Dank heutiger Technik einen Aufwand von wenigen Sekunden. Allerdings traten in der Fallstudie Inkonsistenzen auf, welche die Identifikatoren und die Systemstruktur betrafen. So unterschieden sich die Identifikatoren der Testfälle von denen der Verhaltensbedingungen und diese wiederum von denen im Simulink™-Modell. Zur Kompensation mussten manuell Abbildungstabellen angelegt werden, welche ca. 2 Std. Aufwand erforderten. Weiter schlichen sich bei ca. 5% der in Word festgehaltenen Anforderungen Tippfehler ein, die aufwändig nachgebessert werden mussten. Diese Probleme traten bei den Ausgabebeschreibungen nicht auf, da hier bereits durch TargetLink™ per Konstruktion konsistente Identifikatoren gesichert waren. Diese Erfahrungen zeigen, dass konsequente Ansätze einer durch Software gestützten Verfolgbarkeit und ein automatischer Abgleich der Konsistenz der Dokumente einen starken Einfluss auf die Effizienz der Automatisierung haben und unter diesen Umständen die Extraktion vollständig automatisierbar ist.

Fehlerzuweisung

In der Fallstudie wurde eine generische Zuweisung von Fehlern implementiert. Den Verhaltensbedingungen und den Qualitätsanforderungen wurden die Verletzungen der jeweiligen Anforderungen als Fehler zugeordnet. In seltenen Fällen musste bei Qualitätsanforderungen feiner unterschieden werden und deshalb die Fehlerbeschreibungen bearbeitet werden. So wurden z.B. die Fehler zur Usability-Anforderung „Komfort gewährleisten“ unterschieden in die Stufen „eingeschränkt“ und „nicht gewährleistet“. Der Aufwand zur Nachbearbeitung dieser Anforderungen war aller-

dings gering (1 Stunde). Die Zuordnung der Fehler zu den Signalen wurde auf zwei verschiedene Arten realisiert. Zum Einen wurden abhängig von Datentypen, Wertebereichen und Semantikbeschreibungen die Fehlerbeschreibungen generiert. Zum Anderen wurden die Module mittels Intervall-Abstraktion diskretisiert und dann diskrete Abweichungen angegeben. In beiden Fällen wurde bei der Fehlerbeschreibung die Zeit vernachlässigt. Hier zeigte sich, dass die Möglichkeiten, generische Fehler vorzugeben, gering sind. Die in [Sac01] festgestellte Abhängigkeit geeigneter Abstraktionen von zu verifizierenden Eigenschaften kann auf Fehlerbeschreibungen übertragen werden (siehe Abschnitt 6.1.4). Liegen die Fehlerbeschreibungen zu den Ausgaben an der Nutzungsschnittstelle noch nicht vor, so müssen diese generiert werden, ohne sie von anderen Fehlerbeschreibungen ableiten zu können. Die Generierung und manuelle Nachbearbeitung der Fehlerbeschreibungen zeigte, dass die vorgeschlagene Einteilung der Intervalle fast immer nachgebessert werden musste und auch das zeitliche Verhalten der Fehler nicht generisch zuweisbar war. Hilfreich war hier nur die Verwendung der existierenden Informationen zu den Signalen, wie Wertebereiche, Semantiken diskreter Werte oder Belegungen von Bits. Eine automatisierte Zuweisung von Fehlern zu den Ausgabebeschreibungen zeigte sich hier nicht als besonders effizient.

6.3.2 Anforderungsabhängigkeit

Die Ermittlung der Zusammenhänge zwischen Anforderungen konnte in der Fallstudie bedingt automatisiert werden. Im Wesentlichen wurden anhand der Fallstudie die Verknüpfung zwischen Ausgabebeschreibungen und Verhaltensbedingungen untersucht.

Die Zusammenhänge zwischen Ausgabebeschreibungen wurden über eine syntaktische Analyse abgeschätzt. Eine genaue Beschreibung des Vorgehens hierzu ist in Anhang B zu finden. Die Ermittlung dieser Zusammenhänge orientierte sich am Signalfluss des Simulink-Modells der Fallstudie. Die Semantik der Operatoren wurde hierbei vernachlässigt. Für die eingebetteten C-Code-Module wurden statt dem Signalfluss die Zuweisungen zu Variablen und die Variablen der Bedingungen von Verzweigungen verfolgt. Auch hier wurde die Semantik der Ausdrücke, Operatoren und Funktionen vernachlässigt. Dadurch konnte nur eine obere Schranke für die tatsächlichen Zusammenhänge zwischen Signalen ermittelt werden. In der Fallstudie mit über 10000 Operatoren wurden allerdings bei einem Review nur weniger als 10 Stellen erkannt, an denen die syntaktische Analyse irrtümlich einen Zusammenhang identifiziert hat. Trotz der Modellierungsrichtlinie, alle Abhängigkeiten explizit im Modell als Signalfluss darzustellen, konnte diese an Stellen, an denen ins EEPROM geschrieben wurde, nicht eingehalten werden. Obwohl vorhanden, konnte an dieser Stelle die syntaktische Analyse keine Abhängigkeiten identifizieren. Hier war bei einem Modul eine Nachbesserung notwendig. In Summe konnte diese Aufgabe jedoch in hohem Grade automatisiert werden.

Die Ermittlung der Zusammenhänge zwischen Anforderungen konnte nur exemplarisch an einfachen Teilsystemen der Fallstudie angewendet werden, da hier die Rechenleistung der Verifikationswerkzeuge sehr eingeschränkt war. Aufgrund der Komplexität der Anforderungen reichte häufig die Aussagenlogik nicht mehr aus. Deshalb wurde, ähnlich wie bei Testfällen, eine Überprüfung durchgeführt, die lediglich zur Unterstützung der Zusammenhangsmodellierung verwendet werden konnte. Es wurde zur Abschätzung angenommen, dass die Anforderungen nicht redundant sind und sich auch gegenseitig nicht überschneiden. Weiter wurden die Folgen nur für Anforderungen geprüft, die nur freie Variablen enthalten. Auf dieser kleinen Untermenge der Anforderungen konnte gezeigt werden, dass prinzipiell die Ermittlung der Zusammenhänge unterstützt werden kann. Für den Großteil der Anforderungen fand die Verknüpfung jedoch manuell in Traceability-Tabellen statt, deren Inhalte in die Dokumente der Sicherheitsanalyse integriert wurden.

6.3.3 Fehlerzusammenhänge

In der Fallstudie wurden sowohl die Ermittlung zwischen Fehlerzusammenhängen zu Verhaltensbedingungen, wie auch die Ermittlung von Fehlerzusammenhängen zu Ausgabebeschreibungen umgesetzt. Die Fallstudie konzentrierte sich auf Zusammenhänge zwischen Einfachfehlern.

Anforderungsmodifikationen

Den Verhaltensbedingungen war jeweils die Verletzung der jeweiligen Bedingung zugeordnet. Während für das Funktionsnetz noch redundante bzw. überschneidende Anforderungen dazu führten, dass mindestens Prädikatenlogik zur Überprüfung der Eigenschaften notwendig war, so kann bei Einfachfehlern die Anfrage in Aussagenlogik umgesetzt werden, wenn die Anforderung, zu der der Folgefehler gehört, nur freie Variablen enthält. Für kleine Teile der Fallstudie konnte so exemplarisch die Anwendbarkeit der Analyse demonstriert werden. Allerdings ist auch hier die Leistungsfähigkeit der Werkzeuge und die Verfügbarkeit von formalen geeigneten Anforderungen ein entscheidender Faktor, der eine praxisnahe Anwendung der Zusammenhangsermittlung ausschließt. Für alle Anforderungen, welche nicht auf diskrete Formeln ohne gebundene Variablen reduziert werden konnten, musste die Angabe manuell ergänzt werden.

Ausgabemodifikationen

Die Umsetzung der Zusammenhänge zwischen Ausgabemodifikationen wurde auf verschiedene Weisen angegangen. Die Fallstudie war im Wesentlichen auf den Datenfluss ausgerichtet und für einzelne Umsetzungen zu komplex. Deshalb wurden hier verschiedene Umsetzungen, die sich ergänzen, durchgeführt und bewertet. Die verwendeten Mechanismen waren Testen, SAT-Verifikation, Umformungen von Anfragen zu Aussagenlogik, automatisierte Abstraktion, Abstraktion von Modellteilen und letztlich die manuelle Modellierung.

Testen

Eine Umsetzung der Analyse der Zusammenhänge von Ausgabemodifikationen war die Verwendung von Tests. Um hier eine effiziente Integration in die Entwicklung zu ermöglichen, wurde die bestehende Testumgebung EXACT™ verwendet. Der Vorteil dieser Umgebung war, dass die Anforderungstests bereits erstellt waren und

so das Sollverhalten des Systems exemplarisch vorgaben. Mit diesen Tests konnten Zusammenhänge zwischen Eingabe- und Ausgabemodifikationen einer Komponente ermittelt werden. Mittels einer syntaktischen Analyse konnte dann, wie bei der Funktionsvernetzung, auf die Wirkung geschlossen werden. Um eine Anwendung mit den bestehenden Werkzeugen zu realisieren, mussten für das Fehlermodell zwei Abstraktionen angewendet werden. Es wurden erstens die zeitlichen Zusammenhänge zwischen den Modifikationen abstrahiert (siehe Abschnitt 5.2.2.1). So konnte die syntaktische Analyse zur Ermittlung der Folgen verwendet werden. Die zweite Abstraktion war die Abstraktion der Stromtreue (siehe 5.2.2.3). Die Testumgebung ließ nur Abweichungsanalysen für einzelne Kanäle zu und konnte deshalb wesentlich effizienter verwendet werden. Der Vorteil der Verwendung von Tests war, dass jedes Modul analysiert werden konnte. Der Nachteil der Tests war, dass diese immer nur exemplarische Durchläufe ermöglichten, und so anhand dieser nicht zwingend alle Zusammenhänge erkannt werden konnten. Wesentliche Kriterien zur Bewertung des Vorgehens mit Tests waren zum Einen die Ersparnis der Zeit, also die Effizienz und zum Anderen der Grad der erkannten Zusammenhänge, also die Effektivität. Hinsichtlich der Effizienz waren die Testläufe in der Testumgebung relativ langsam. So kostete das Testen eines Moduls bis zu 6 Stunden. Bei einer geeigneten Umgebung und ausreichender Rechenleistung besteht hier allerdings Potential zu einer akzeptablen Effizienz. Allerdings war die Effektivität der Testfälle nur bedingt hilfreich. Bei Modulen, bei denen die Eingänge und Ausgänge boolesch waren oder wenige Werte hatten, konnten bis zu 100% der Zusammenhänge erkannt werden. Auch bei Modulen, bei denen aussagekräftige Äquivalenzklassen gebildet werden konnten, war die Erkennungsquote relativ hoch. Bei Systemen, deren Eingaben und Ausgaben nicht diskret waren, fiel die Erkennungsquote teilweise auf nur 25%. Da in der Fallstudie der Großteil der Module nicht diskrete Werte enthielt, war die Verwendung von Testfällen nur eine geringe Unterstützung der Sicherheitsanalyse.

Beweiser

Eine weitere Umsetzung der Analyse der Zusammenhänge zwischen Ausgabemodifikationen war die Verwendung von SAT-Solver-basierten Verifikationswerkzeugen. Der unmittelbare Vorteil dieser Werkzeuge war, dass die Ergebnisse der Analysen als Nachweis verwendet werden konnten, ob ein Zusammenhang besteht oder nicht. Der Preis der Anwendungen war dafür, dass nur kleine Teile der Fallstudie verifiziert werden konnten. Auch hier wurden Abstraktionen verwendet, um die Anfragen an die Werkzeuge geeignet zu vereinfachen. Zum Einen wurde, wenn möglich, die in Abschnitt 6.1.4.1 vorgestellte diskrete Abstraktion der Modelle verwendet. Diese lies sich zumindest dann anwenden, wenn für die wertkontinuierlichen Signale (auch interne Signale) klare Intervalle ermittelt werden konnten. In der Praxis ließ sich diese Abstraktion ohne Probleme auf ca. 20% der Module anwenden. Weiter wurden für die Fallstudie nur Fehler betrachtet, die nur freie Variablen hatten und deshalb die Anfragen auf Aussagenlogik (siehe Abschnitt 6.2.3.1) vereinfacht werden konnten. Damit konnten zumindest 50% der Fehler betrachtet werden. Um die Zustandsexplosion der SAT-Solver in Grenzen zu halten, fanden nur Betrachtungen über Zeiträume von maximal 5 Zeitschritte statt (siehe Abschnitt 5.2.2.2). Damit konnten einige längerfristige Folgen nicht mehr ausgeschlossen werden. Hin-

sichtlich der Bewertung der Systemsicherheit reichte die Zeitspanne meistens. Da die Modelle in Summe immer noch zu komplex und groß waren, wurden weiter die Sicherheitsfunktionen explizit extrahiert und untersucht. Damit konnte zumindest das Fail-Safe-Verhalten halbautomatisch untersucht werden. Die einfachen Diagnosekomponenten konnten verifiziert werden, der komplexe Weg hin zu den Diagnosekomponenten musste manuell modelliert werden. Der mit SAT-Solvern verifizierbare Anteil der Fallstudie betrug so weniger als 10%. Die theoretische Umsetzbarkeit der automatischen Verifikation konnte somit gezeigt werden, aber es wurden deutliche Lücken der Implementierungen der Werkzeuge aufgedeckt. Hier besteht noch Verbesserungspotenzial, um die in dieser Arbeit vorgestellten Modelle praxistauglich anwendbar zu machen.

6.4 Zusammenfassung

Dieses Kapitel diskutiert die Möglichkeiten, die in Kapitel 5 beschriebenen Zusammenhänge automatisiert zu verifizieren. Hierzu werden allgemein die Möglichkeiten einer automatisierten Analyse aufgezeigt, wie auch die konkrete Umsetzung spezifischer Fragestellungen vorgestellt und bewertet.

Die in diesem Kapitel diskutierten allgemeinen Möglichkeiten einer automatisierten Analyse werden aufgezeigt und hinsichtlich ihrer Anwendbarkeit diskutiert. Faktoren sind hierbei die Modelle bzw. Modellelemente, die beschrieben werden können, die mögliche Größe der Modelle und die Aussagekraft der Resultate der Analyse. Die diskutierten Möglichkeiten waren (1) die syntaktische Analyse, (2) die Simulation und der Test, (3) die Verifikation auf rein boolescher Ebene und die Einbindung von Theorien wie linearer Gleichungssysteme und (4) das Hinzuziehen von Abstraktionsmechanismen.

Die Umsetzung spezifischer Fragestellungen richtet sich nach den Aufgaben in Kapitel 2. Die Extraktion des Systemstrukturbaums und die Zuweisung von Anforderungen wird anhand einer syntaktischen Analyse in Kombination mit manueller Ergänzung realisiert. Die Zuweisung der Fehler geschieht anhand eines Regelwerks, welches unter Berücksichtigung von Datentypen, Wertebereichen, Elementen der Implementierungsmodelle und der eingesetzten Abstraktionsmechanismen die Fehlerbilder aussucht. Die Umsetzung der Anforderungsvernetzung hängt vom Anforderungstyp ab. Zusammenhänge zwischen Verhaltensbedingungen können mit First Order Logic Beweisern ermittelt oder über Simulation und Zusammenhänge zwischen Einfachfehlern abgeschätzt werden. Meist ist eine manuelle Nachbearbeitung notwendig. Eine obere Schranke für die Zusammenhänge zwischen Ausgaben kann mit einer syntaktischen Analyse ermittelt werden. Qualitätsanforderungen werden manuell verknüpft. Die Vernetzung der Fehler kann mit Hilfe von Tests und Abstraktionsmechanismen abgeschätzt werden. Alternativ wird ein Weg vorgestellt, einige Anfragen auf Aussagenlogik zu reduzieren und so berechenbar zu machen oder Zusammenhänge zwischen Ausgabemodifikationen durch Auswahl bestimmter Pfade

im Signalfluss effizient zu ermitteln. Die Umsetzungen spezifischer Fragestellungen werden anhand der Fallstudie bewertet.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit stellt Modelle und Modellierungstechniken für eingebettete Systeme vor, mit denen Fehlverhalten und Zusammenhänge zwischen diesen modelliert werden können. Hierbei werden die in der Industrie vorgefundene Vorgehensweise bei Sicherheitsanalysen und die verwendete Systemmodellierung berücksichtigt. Weiter analysiert die Arbeit die Möglichkeiten, mit bestehenden Verifikationswerkzeugen die Sicherheitsanalysen zu automatisieren. Mit dieser Arbeit wird eine teilautomatisierte und in die Entwicklung integrierbare Sicherheitsanalyse ermöglicht.

Grundlagen

Einleitend erfasst die Arbeit die derzeitige Vorgehensweise der Sicherheitsanalysen FMEA und FTA. Die Schritte zur Erstellung der Dokumente dieser Analysen sind das Aufbauen eines Systemstrukturbaums mit Anforderungen, die Zuordnung von Fehlern zu den Anforderungen, die Fehlernetzerstellung und die Bewertung der Fehler. Zu den einzelnen Schritten werden Ansatzpunkte zur formalen Modellierung und Automatisierung identifiziert.

Weiter listet die Arbeit bereits bekannte Modelle und Modellierungstechniken zur Beschreibung von eingebetteten Systemen auf. Die Verhaltensmodelle der Systeme sind Relationen zwischen Funktionen, welche Eingaben und Ausgaben eines Systems beschreiben. Diese Modelle können mit Modellierungstechniken spezifiziert werden. Modellierungstechniken sind unter anderem Prädikate und Automaten. Sie sind meist auf die Sichten (Regelungssicht, Interaktionssicht, usw.) zugeschnitten. Diese Modellierungstechniken für Verhalten ermöglichen spezifische Fehlerbeschreibungen. Die Verhaltensmodelle werden mit Modellen ergänzt, welche die Struktur und qualitative Eigenschaften der Implementierung des Verhaltens enthalten. Diese qualitativen Eigenschaften sind ein Ansatzpunkt zur Identifikation potentieller Hardwareausfälle und systematischer Ausfälle. Deren Wirkung kann als Fehlverhalten im Verhaltensmodell dargestellt und in einer Funktionssicherheitsanalyse weiter analysiert werden.

Fehlermodell Der erste wesentliche Beitrag besteht in der Erarbeitung einer Menge von Modellierungstechniken für Fehler. Diese Modellierungstechniken basieren auf der von Breitling [Bre01] vorgestellten Fehlerdefinition als Modifikation des Verhaltens. Sie sind auf die Modellierungstechniken für das Sollverhalten eingebetteter Systeme zugeschnitten und können als System-, Ausgabe-, Anforderungs- und Transitionsmodifikationen modelliert werden. Es wird berücksichtigt, dass
(1) Fehler getrennt vom Sollverhalten als Modifikationen definierbar sind,
(2) Tupel der Verhaltensrelation sowohl hinzugefügt, wie auch entfernt werden können,
(3) die vorgefundenen Fehler der Fallstudien aus dem Fahrwerkregelbereich formal darstellbar sind,
(4) die Fehlerkombination insbesondere bei Unabhängigkeit der Modifikationen definiert ist und
(5) die Modifikationen auf Einträge der bestehenden Sicherheitsanalysedokumente abbildbar sind.
Die Anwendbarkeit der Modellierungstechniken wird anhand einer Sammlung von Fehlern aus den Fallstudien und der Literatur demonstriert.

Zusammenhangsmodell Der zweite Kernbeitrag ist die Erarbeitung spezifischer Zusammenhangsmodelle zwischen Fehlverhalten und Methoden zu deren Ermittlung. Grundlegend wird hierzu die Definition von Breitling [Bre01] eines Fehlerzusammenhangs als Komposition von Fehlverhalten aufgegriffen. Zu dieser Definition werden in dieser Arbeit allgemeine Abstraktionen der Zusammenhänge ermittelt. Aufbauend auf diesen werden spezifische Zusammenhangsmodelle erarbeitet, mit denen System-, Ausgabe-, Anforderungs- und Transitionsmodifikationen verknüpft werden können. Die Beschreibung der Zusammenhangsmodelle umfasst dabei
(1) Modellierungstechniken und Methoden zur Ermittlung der Zusammenhänge,
(2) die Integrierbarkeit mit der manuellen Analyse,
(3) die Modellierung über mehrere Ebenen in der Systemhierarchie
(4) die Anwendbarkeit in den Fallstudien und
(5) die Vereinbarkeit mit Modifikationen, die Relationentupel hinzufügen und entfernen.
Letztlich werden die Möglichkeiten der Integration der Implementierungsebene analysiert.

Automatisierung Zum Nachweis der Anwendbarkeit der Modelle und Modellierungstechniken werden Umsetzungen spezifischer Aufgaben der teilautomatisierten Sicherheitsanalyse demonstriert. Als Basis werden allgemein die Möglichkeiten der syntaktischen Analyse, des Test und der Simulation, der booleschen Verifikation und von Abstraktionsmechanismen vorgestellt. Die Extraktion der Systemstruktur und die Zuweisung von Anforderungen und Fehlern werden hauptsächlich mittels syntaktischer Analysen realisiert. Die Ermittlung der Zusammenhänge zwischen Fehlern und Anforderungen wird mittels Tests und boolescher Verifikation in Kombination mit Abstraktionsmechanismen realisiert. Die Effizienzsteigerung der Aufgaben der Sicherheitsanalyse durch die Automatisierung wird anhand der Fallstudie bewertet.

Die Arbeit leistet mit der Angabe von Modellen und Modellierungstechniken und der Betrachtung der Automatisierbarkeit einen Beitrag zur effizienten Sicherheitsanalyse eingebetteter Systeme.

7.2 Ausblick

Verifikationstechniken

In dieser Arbeit wird diskutiert und aufgezeigt, in wie fern die Sicherheitsanalyse eingebetteter Systeme automatisiert werden kann. Die Betrachtung von Verifikationswerkzeugen wie SAT-Solvern und Model-Checkern in Kombination mit Analysetechniken für Datenfluss-Anteile der Modelle zeigt Probleme bei der Effektivität und der Effizienz der Werkzeuge. Die Werkzeuge scheitern häufig bereits bei kleinen Systemen, so dass eine für die Industrie anwendbare automatisierte Analyse noch nicht realisiert werden kann. Erste Versuche in [BPT07] zeigen, dass hier Optimierungspotenzial und Forschungsbedarf besteht.

Abstrakte Modelle

Grundlage für diese Arbeit sind detaillierte Systemmodelle. Die Modelle sind zu komplex für eine automatisierte Sicherheitsanalyse (z.B. Größe, Datenfluss- gemischt mit Interaktionssicht). Auch automatische Abstraktionen dieser Modelle können die Komplexität nur bedingt reduzieren. Offen bleiben Konzepte zur Erstellung von abstrakten Modellen, deren Eigenschaften hinsichtlich der Sicherheitsanalyse automatisiert verifiziert werden können und deren Eigenschaften in den Implementierungen per Konstruktion oder Verifikation nachgewiesen werden können. Erste Ansätze hierzu sind im Bereich der Betriebsmodi-Automaten zu finden (siehe [KTB+07]). Interessante Forschungsfelder sind diesbezüglich die Verfeinerung, die Techniken der Erstellung und der Konsistenzabgleich der Modelle.

Nutzung

Der Fokus der Arbeit liegt auf der Analyse der eingebetteten Systeme hinsichtlich der Fehler, die ein System im Sinne von Fehlverhalten haben kann. Nicht betrachtet werden Fehler, welche die Nutzung der Systeme betreffen. Gerade bei Analysen von Mehrfachfehlern müsste in weiteren Schritten die Nutzung und Nutzungsfehler analysiert werden. So könnte einerseits die Wirkungen von Fehlern feiner analysiert werden und andererseits die Toleranz eines Systems gegen fehlerhafte Nutzungen bewertet werden. Forschung im Bereich der Analyse der Nutzung wird z.B. in [Win08] betrieben.

Entwicklungsprozess

Fehler in Produkten können Hardwareausfälle sein, oder es können systematische Ausfälle sein, deren Ursache im Entwurf oder der Fertigung von Produkten liegt. In dieser Arbeit beginnt die Betrachtung der Ursache-Wirkungs-Ketten mit dem Auftreten eines Fehlers im Produkt. Die Verfolgung dieser Fehler zurück in den Entwicklungsprozess zu Aktivitäten, welche diese verursacht haben, ist ein weiteres Feld, in dem Forschungsbedarf gerade für den Bereich Software-Engineering besteht, um die Prozesse anpassen zu können.

Integration weiterer Dimensionen

Schließlich konzentriert sich die Arbeit auf Modelle des Sollverhaltens, um Ursache-Wirkungs-Beziehungen zu analysieren. Die Entwicklung eingebetteter Systeme muss allerdings auch andere Dimensionen, wie z.B. den räumlichen Aufbau der Systeme

berücksichtigen. In diesen Dimensionen können z.B. thermische Zusammenhänge festgehalten werden, die in den Modellen des Sollverhaltens nicht berücksichtigt werden. In dieser Arbeit wird die Einbeziehung anderer Modelle, welche nicht auf der Beschreibung des Sollverhaltens bestehen, diskutiert. Eine eventuelle Folgearbeit wäre die Erarbeitung eines Konzepts, in dem Modelle anderer Dimensionen ganzheitlich mit betrachtet werden.

Anhang A

Generierung der FMEA-Struktur

A.1 Systemstrukturbaum

Subsysteme

Die modellierten Systeme dienen in der Systemhierarchie verschiedenen Zwecken. Die einfachste Abbildung ist, dass ein System einem Modul entspricht, also einer Funktionseinheit oder einer Komponente, die in sich eine Funktionalität ist und klare Schnittstellen nach außen hat. Diese kann wiederum aus anderen Systemen zusammengesetzt sein und so eine Teilmodul-Beziehung aufbauen. Diese Struktur entspricht der Architektur des Systems.

Module

Wird eine bestimmte Modellierungstechnik und Sprache verwendet, so ergeben sich spezifische Eigenheiten, die zu weiteren modellierten Systemen führen, die man aber nicht in der Architektur erfassen möchte. In Simulink ist es z. B. üblich, eine Division durch Null zu vermeiden, in dem man ein "Triggered Subsystem" verwendet, welches nur dann Berechnungen durchführt, wenn der Divisor ungleich Null ist (siehe Abbildung A.1). Des Weiteren gibt es auch eine Menge von Subsystemen, die aufgrund ihrer Bedeutung und voraussichtlichen Veränderung über die Zeit nicht aufgenommen werden. In diesem Rahmen werden als pragmatische Lösung Module mit der Endung "_module" gekennzeichnet. Hat ein Modul unter sich keine weiteren Module, so wird es automatisch und unabhängig von der darunter liegenden Struktur zu einem Blatt im Systemstrukturbaum. Ist ein Modul in der Hierarchie weiter oben, besteht also aus anderen Modulen, so kann eine vollständige Partitionierung des Verhaltens in Teilmodule erreicht werden, wenn außerhalb der Teilmodule kein Verhalten definiert ist.

Signalbündelung

Die Module haben, was bisherige Erfahrungen gezeigt haben, meist zwischen 5 und 30 Ausgabekanäle. Möchte man in einem hierarchischen Modul die enthaltene Architektur abbilden, ist dies besser zu visualisieren und insbesondere bei Änderungen der Architektur zu behandeln, wenn man die einzelnen Signale abstrahiert, und nur betrachtet, von welchem Modul ein Datenfluss zu welchem anderen Modul läuft. In der Praxis werden die Module entsprechend von Systemen umgeben, welche die Signale zusammenfassen (multiplexen). Diese umgebenden Systeme werden mit der Endung

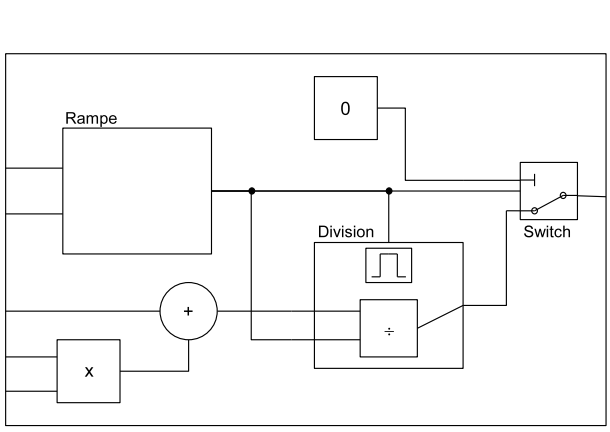


Abb. A.1: Subsysteme, die keine Module der FMEA sind

"_struct" gekennzeichnet und enthalten nur ein Untersystem und eine Menge von Multiplexern / Bus-Creators und Demultiplexern / Bus-Selectors. Diese Strukturkomponenten werden nicht in der FMEA-Struktur abgebildet, da sie in der Implementierung nicht mehr erscheinen. Abbildung A.2 verdeutlicht diese Strukturierung.

Code-Generator-Spezifika

In der Industrie werden die Simulink-Modelle meist weiterverarbeitet und zur Code-Generierung angepasst. In den konkreten Fallstudien sind die Modelle an die Software Target-Link angepasst. Dies geschieht unter anderem dadurch, dass die Eingabe- und Ausgabesignale der Module mit Zusatzinformationen zur Generierung versehen werden. Konkret wird direkt nach dem Eingabeport bzw. direkt vor dem Ausgabeport ein Target-Link-Block gesetzt, der in Simulink als Subsystem auftaucht. Auch diese Blöcke sollen in der FMEA-Struktur nicht erscheinen. Die pragmatische Lösung ist auch hier, diese Blöcke zu kennzeichnen. In der Fallstudie ist dies entweder mit einem beginnenden 'Unterstrich "_" oder mit der Endung "_TL" geschehen. Geht man davon aus, dass alle Verhaltenszusammenhänge explizit modelliert sind, so können Blöcke, die nicht mit den Ausgaben des umgebenden Moduls verbunden sind, herausgenommen werden, da sie keine Wirkung auf das System haben können. Abbildung A.3 zeigt diese zu entfernenden Elemente anhand eines Beispiels eines (abgeänderten) Moduls der Fallstudie.

Umbenennungen

Da die Spezifikation und die Modelle meist parallel entstehen und sich auch erst mit diesen Konventionen für die Namensgebung der Komponenten ergeben, können die Benennungen der Komponenten in den Simulink-Modellen unterschiedlich zu den Benennungen in den Spezifikationen sein. Um in den Sicherheitsanalysen konsistente Namen zu erhalten, können Ersetzungstabellen verwendet werden, welche die Namen zueinander abbilden. Typische Abbildungen, die in der Fallstudie vorgenommen wurden, waren:

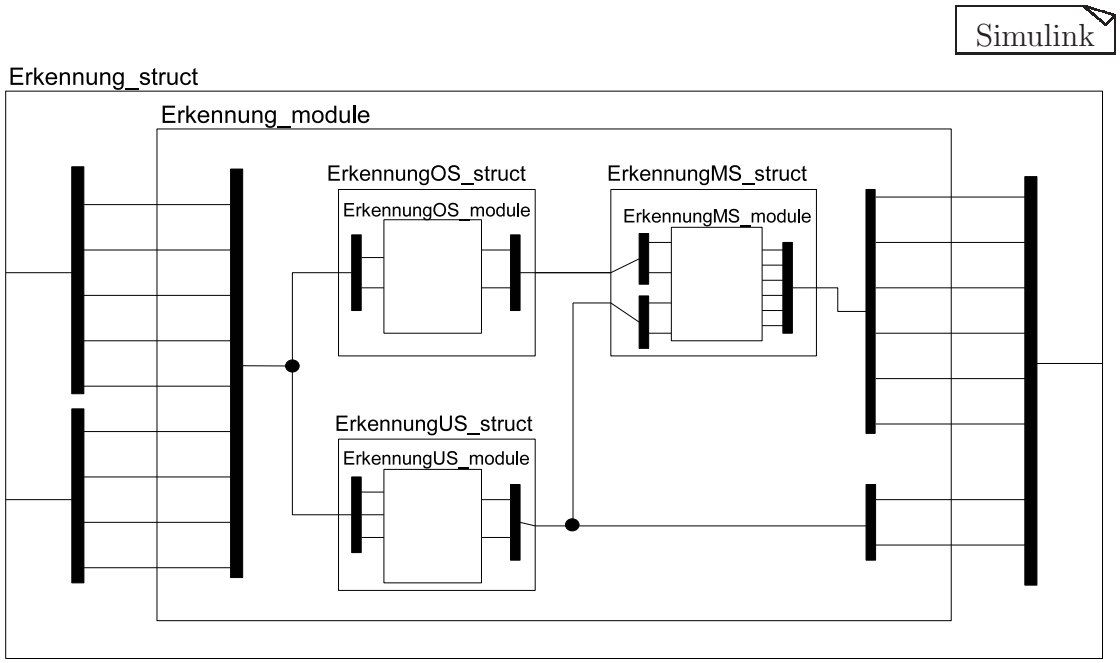


Abb. A.2: Einsatz von Strukturelementen

Namen

<Konvention>	↔	*<Kürzel>*	↔	*<provisorischer Name>*
SDA_U_URG	↔	URG	↔	U_URG_SDA
XXX_XXX_FCU	↔	FCU	↔	FlowControlUnit

Bewertung

Die extrahierte Struktur weist eine Menge von verbleibenden Abweichungen auf, die noch eine manuelle Nachbearbeitung erfordern. Von den in der Fallstudie automatisch erkannten 56 Modulen mussten bei 9 Modulen manuelle Nachbesserungen gemacht werden. 3 Module, die wegen Verletzung der Modellierungsrichtlinien als Module erkannt wurden, mussten entfernt werden. 5 Module mussten als Module gekennzeichnet werden, damit deren Teilsysteme nicht mehr in den Baum aufgenommen werden. Ein Modul musste umbenannt werden. Tabelle A.1 auf Seite 210 zeigt das Ergebnis im Detail.

A.2 Anforderungszuweisung

Anforderungen

Den einzelnen Modulen sind eine Menge von Anforderungen zugeordnet. Je nach Art der Anforderung und Datenquelle können diese übernommen werden. In den Fallstudien gab es zwei Datenquellen:

- *Verhaltensbedingungen und Qualitätsanforderungen in MS Word*
- *Signalbeschreibungen in Targetlink (Data-Dictionary)*

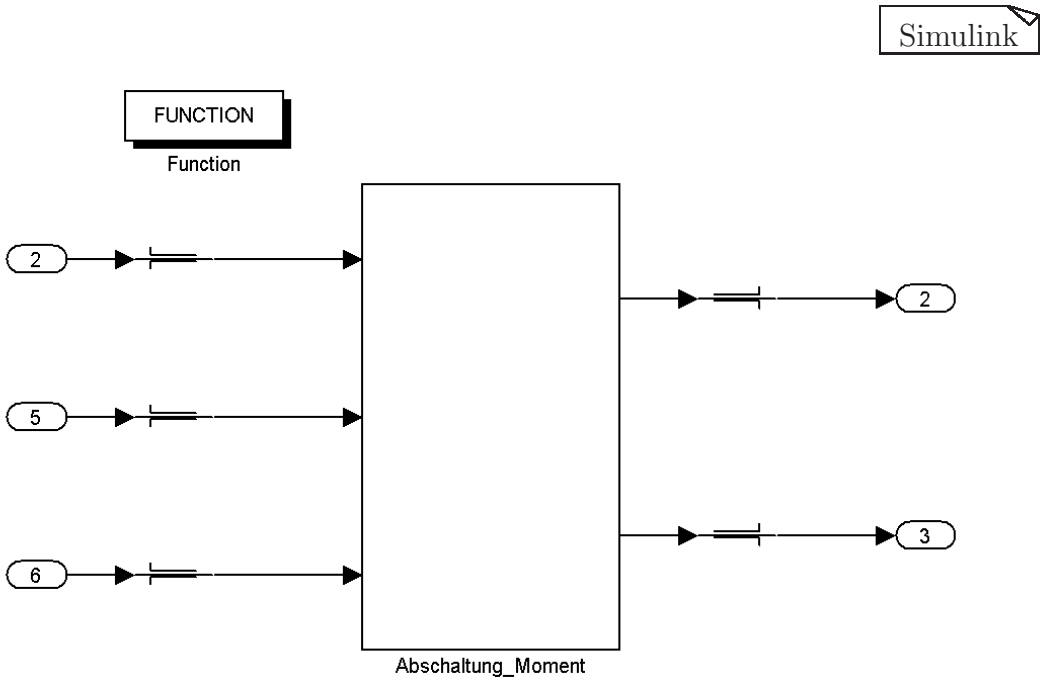


Abb. A.3: Codegenerator-Spezifika: TL-Ports und FUNCTION-Blöcke

MS Word

Die informell in Word festgehaltenen Anforderungen entsprechen Templates, in denen zumindest eine eindeutige ID und eine Überschrift notiert ist (siehe auch [RR99]). Ein automatische Übernahme der Anforderungen ist aus Word eher ungeeignet. Liegen die Daten in einem Tabellenkalkulationsformat (z.B. csv) vor, so können die Überschriften und IDs der Anforderungen den Strukturkomponenten zugeordnet werden. Tabelle A.2 auf Seite 211 zeigt die Überschriften einiger Anforderungen, welche in der Fallstudie extrahiert und zugewiesen wurden. In den Fallstudien waren die Anforderungen nur informell beschrieben. Bei der Angabe des Typs wurde hier nachträglich gekennzeichnet, wie die Anforderungen der Klassifizierung aus Kapitel 2 zugeordnet werden können und ob diese auch (teilweise) formal festgehalten werden können. Interessant ist hier, dass viele Verhaltensbedingungen formalisiert werden können. Allerdings ist hier keine Aussage getroffen, wie komplex die Formalisierung ist. In Anforderung 2 muss zumindest die Komplexität der Prädikatenlogik zur formalen Formulierung der Anforderung verwendet werden, da hier eine obere Grenze für die eingehenden Signalwerte gefordert wird, ab der abgeschaltet wird, welche aber noch nicht in Zahlen gegeben ist. Andere Anforderungen lassen sich deutlich einfacher formulieren, in dem auch Bezug auf interne Signale genommen wird. So kann z.B. in Anforderung 5 Bezug auf den Zustand des Reglers genommen werden oder in Anforderung 13 die Bedingung für eine Rechtskurve referenziert werden. Allerdings findet in diesen Fällen bereits eine Vorgabe für die Architektur statt bzw. um diese Signale an der Systemschnittstelle prüfen zu können, muss diese entsprechend erweitert werden (interne Signale sind kursiv):

Prädikate

REQ_XXX_XX.2: $\exists a, b, \ldots :$
$((\mathsf{XXXdw} > a) \vee (\mathsf{XXXel} > b) \vee \ldots) \Rightarrow (\mathsf{XXXom} = 0)$
REQ_XXX_XX.5: $(XXSP \neq 10) \Rightarrow (\mathsf{XXXom} = 0)$
REQ_XXX_XX.14: $(XXXrP > 0) \Rightarrow (\mathsf{XXXom} > 0)$

Die in diesem und in den weiteren Beispielen gegebenen Daten sind anonymisiert. Teile der Bezeichner werden durch XXX ersetzt. Weiter werden die Zahlen der Fallstudie mit der Zahl 10 ersetzt.

TargetLink

Die einzelnen Signale zwischen den Modulen werden in der Fallstudie mit eindeutigen IDs und Beschreibungen der Semantik der Signale versehen. Dazu gehören neben der informellen Beschreibung auch Angaben zu Grenzwerten, Einheiten und Datentypen[1]. Diese Informationen lassen sich relativ einfach nach ACSII-Tabellenformaten wie CSV exportieren und dann automatisch zuordnen. Ein Auszug des Ergebnisses der Ermittlung der Signalbeschreibungen für die Ausgaben ist in Tabelle A.3 auf Seite 212 zu finden. Wesentlich für die Identifikation von potentiellen Fehlerbildern ist die Semantik der Daten. Hier gibt es

- Fließkomma-Zahlen mit einem Wertebereich und evtl. einer physikalischen Einheit
- boolesche Werte, welche die Erfüllung bestimmter Bedingungen repräsentieren
- Integerzahlen mit Wertebereichen, deren Werte bestimmte Bedingungen oder Nachrichten repräsentieren

Darstellung

Um beim Review einer Sicherheitsanalyse die Bedeutung der Signale besser zu verstehen, werden bei der Generierung eines Analyse-Dokuments die zusätzlichen Informationen mit angegeben. Ein Eintrag in einem FMEA-Dokument, wie in Kapitel 2 vorgestellt, sieht dann zu einer Ausgabe folgendermaßen aus:

Berechnung: Gesamt-Soll-Differenzmoment (XXXom [Float32, Nm, -10/10])

Konsistenz

Eines der größten Probleme bei der Anwendung der Anforderungszuweisung waren Inkonsistenzen zwischen den verschiedenen Modellen und Dokumenten der Entwicklung. Hier mussten manuell Abbildungstabellen erstellt werden, mit denen die jeweiligen Identifikatoren in den Dokumenten verbunden werden konnten. In umgekehrte Richtung eignete sich die automatisierte Zuweisung als nützliches Feedback-Werkzeug, um diese Inkonsistenzen aufzudecken.

[1]Die Angabe eines Datentyps ist in Modellierungswerkzeugen wie Simulink nicht immer notwendig, da dieses über einen (häufig genutzten) Datentyp *Auto* verfügt, der bei Simulationsstart dynamisch die Typen zuweist.

A.3 Fehlerzuweisung

Fehlerzuweisung

Den einzelnen Anforderungen werden vorgefertigt generische Fehler zugeordnet. Diese hängen von der Art der Anforderung, den zugehörigen Informationen und den Regeln zur Fehlererstellung und den Implementierungsinformationen zu den Modulen ab:

- *Verhaltensbedingungen und nichtfunktionale Anforderungen*
  Die generische Vorgabe für Fehler zu einzelnen Verhaltensbedingungen und nichtfunktionalen Anforderungen ist mit dem einfachen Fehler *Anforderung nicht erfüllt* gegeben. Möchte man diese feiner unterscheiden, so ist hier manuelle Nacharbeit zu leisten, bei der dann meist individuell die Modifikation der Anforderung auch formal spezifiziert werden muss, wenn man deren Wirkung automatisch analysieren möchte.

- *Ausgabe- / Signalbeschreibungen*
  Zu den Ausgaben eines Systems ist die Zuordnung von Fehlern stark von den vorhandenen Informationen zu den Ausgaben und dem Regelwerk abhängig. Liegen Daten zur Toleranz vor, so muss eine Abweichung mindestens diese überschreiten, um als Fehler gewertet zu werden. Liegen Daten zum Wertebereich vor, so kann die Überschreitung der Minima und Maxima der Wertemenge als Fehler hinzugefügt werden. Ist der Datentyp zusammen mit einer Datensemantik gegeben, so können generische Wertabweichungen vorgeschlagen werden. Bei Fließkomma-Variablen mit einer linearen Ordnung der Werte kann eine Untermenge der zeitneutralen Wertabweichungen aus Abschnitt 3.3.2 vorgeschlagen werden. Bei diskreten Bit- oder Wertbelegungen kann jeweils die Ausgabe eines falschen Wertes vorgeschlagen werden.

Informelle Anforderungsmodifikation

Die Zuweisung der Fehler zu Verhaltensbedingungen und Qualitätsanforderungen ist in informellen Dokumenten durch einfaches Zuweisen des Fehlers *Anforderung nicht erfüllt* einfach automatisiert. Bei der Betrachtung der Fallstudie der Berechnung des Differenzenmoments traf diese einfache Fehlerbeschreibung fast immer zu. Qualitätsanforderungen waren in dieser Fallstudie zu dem Zeitpunkt der Bewertung kaum vorhanden. Anders war es in der Fallstudie der Überlagerungslenkung. Hier gab es eine Menge von Qualitätsanforderungen. Hier mussten bei ca. 30% der Qualitätsanforderungen eine feinere Unterscheidung der Fehler getroffen werden. Im Wesentlichen waren hier die Anforderungen zur Benutzbarkeit (Usability) betroffen. Hier fand eine Einteilung der Fehler in *eingeschränkt* und *nicht gewährleistet* statt. Die Fehler zu Verhaltensbedingungen konnten auch hier generiert verwendet werden.

formale Anforderungsmodifikation

Die formalen Beschreibungen zu den Fehlern sind im Falle der Qualitätsanforderungen nicht vorhanden, da diese das Verhalten nicht unmittelbar betreffen. Da auch die Vernetzung dieser Anforderungen nicht von den formalen Beschreibungen abhängt, spielt dies auch keine weitere Rolle. Die formale Beschreibung von Fehlern zu Verhaltensbedingungen wurde generisch mit der bayesschen Modifikation $(false, true)$

gesetzt, die also beliebiges Verhalten zulässt. Da in der Fallstudie die Fehler zu Verhaltensbedingungen nicht feiner unterschieden wurden, traf hier die automatische Zuweisung immer zu.

Signale

Bei den Ausgaben bzw. Signalen ist die Zuordnung von Fehlern deutlich aufwändiger und weniger effektiv gestaltbar, als bei den Verhaltensbedingungen. Grund hierfür ist die Datenfluss-Sicht. Zu einer Ausgabe gibt es eine von Fehlern, die von verschiedenen Faktoren abhängen, die nicht immer automatisch ermittelbar sind. In der Fallstudie wurden Fehler, welche die Zeit betrafen, nicht automatisch generiert. Die Zuweisung der Fehler wurde abhängig von einer Reihe von Faktoren anhand des Vorgehens in Tabelle A.4 auf Seite 213 ermittelt.

Bewertung

Ein Vergleich der Fallstudien mit den über den eben genannten Algorithmus zur Fehlerzuweisung zeigte, dass zwar bei einfachen Modulen, welche Hardware mit einer einfachen Logik präsentieren, eine Zuweisung größtenteils funktioniert. Auch [PCB+55] bestätigt diese Feststellung. Allerdings zeigte die Fallstudie auch, dass für große Regler-Software eine generische Zuweisung kaum möglich ist. So mussten an der Schnittstelle der Software meistens an die 50% der Fehler angepasst bzw. ergänzt werden. Tabelle A.5 auf Seite 214 zeigt die Notwendigkeit der Anpassungen anhand eines Signals aus der Fallstudie.

Xxxort			Generated
Xxxern			
Xxxung	Xxxung		
Xxxung			
Xxxung			
SDA_struct			✘überflüssiges Element wg.
X_SDA			Namenskonvention-Verletzung
X_XX_SDA	X_XX_SDA		
X_XXX_SDA	X_XX_SDA		
XX_SDA			
XX_SDA			
XX_XX_SDA	XX_XXX_SDA		
XX_XXX_SDA	XX_XXX_SDA		
XX_SDA			
XXX_SDA			
XXX_XXX_SDA	XXX_XX_SDA		
XXX_XXX_SDA	XXX_XXX_SDA		
XXX_XXX_SDA	XXX_XX_SDA		
XX_SDA			
XX_XXX_SDA	XX_XX_SDA		
XX_XXX_SDA	XX_XX_SDA		
XX_XXX_SDA	XX_XXX_SDA		
XX_XXX_SDA	XX_XX_SDA		
XX_XX_SDA	XX_XXX_SDA		
XX_XX_SDA	XX_XXX_SDA		
Rueckkopplung			
Signalmanipulation_dummy			✘Element nur zur Entwicklung
Signalmanipulation			✘Element nur zur Entwicklung
XX_SDA			
XX_XX_SDA	XX_XX_SDA		
XX_XX_SDA	XX_XX_SDA		
EepromIn			✘Module, deren Verhalten
EepromRead			einem Speicher entspricht
define			✘Kein Modul
Xxxand			
Xxxung	Xxxsel		
Xxxung	Xxxell		
Xxxtor			
Xxxtor			
Ablaufsteuerung_struct			✘Namenskonvention-Verletzung
Ablaufsteuerung	**Unit Delay**		✘kein Modul
Xxxung			
Xxxung	Xxxung		
Xxxung			
Xxxung			
Xxxung	Xxxung		
Xxxaer			
Xxxtat	Xxxtat		
FlowControlUnit			
BewertungSDA		BewertungDefProg	✘Namenskonvention-Verletzung
OutputControl_struct			

Tabelle A.1: Generierte Systemstruktur der SW des Fahrwerkregelsystems

MODUL: XXX_XX		
ID	*Typ*	*Überschrift*
REQ_XXX_XX.1	Q/A (I)	Funktionsblöcke und Module
REQ_XXX_XX.2	Q/A/V(F/I)	Defensive Programmierung
REQ_XXX_XX.3	V (F)	Zykluszeit und Bearbeitungszeit
REQ_XXX_XX.4	V (F)	Zustände des Reglers
REQ_XXX_XX.5	V (F)	Nur Moment im Zustand -Run-
REQ_XXX_XX.13	V (F)	Untersteuern, Rechtskurve
REQ_XXX_XX.14	V (F)	Untersteuern, Linkskurve
REQ_XXX_XX.15	V (F)	Untersteuern, Modus -comfort-
REQ_XXX_XX.16	V (F)	Untersteuern, Modus -dynamic-
REQ_XXX_XX.17	V (F)	Thermische Belastung
REQ_XXX_XX.18	V (F)	Geradeausfahrt
REQ_XXX_XX.33	V (F)	Rückwärtsfahrt
REQ_XXX_XX.34	V (F)	Differenzdrehzahlgrenze
REQ_XXX_XX.19	V (F)	Reglerzustand - keine Umschaltaufforderung
REQ_XXX_XX.20	V (F)	Reglerzustand - gültige Umschaltaufforderung
REQ_XXX_XX.21	V (F)	Reglerzustand - ungültige Umschaltaufforderung
REQ_XXX_XX.25	V (F)	Reglerzustand - Anhänger
REQ_XXX_XX.26	V (F)	Fehlmomente
REQ_XXX_XX.27	V (F)	Momentenabweichung
REQ_XXX_XX.29	V (F/I)	interne Abschaltung (Eingangssignalfehler)
REQ_XXX_XX.30	V (F)	externe Abschaltung
V: Verhaltensbedingung Q: Qualitätsanforderung A: Architekturanf.	F: Formal darstellbar I: Informell	

Tabelle A.2: extrahierte Anforderungen aus Word (Typ manuell gekennzeichnet)

MODUL: XXX_XX			
ID	*Überschrift*	*Typ*	*Datensemantik [Min/Max]*
XXXom XXXrP XXXxP	Gesamt-Soll-Differenzmoment Unsicherheit Nutzsignal Fahrzeuglängsgeschwindigkeit	Float32 Float32 Float32	Nm [-10/10] rad/s [0/10] m/s [-10/10]
XXXxP	Status der Fahrzeuglängs-geschwindigkeit	UInt8	B0: kein Nutzsignal B1: Eingeschränkte Qualität B2: Richtung unsicher B3: Extrapolation aktiv B4: - ...
XXXer	Fehlerwort zur Zuordnung der erkannten Fehler im Modul XX (defensive Programmierung)	UInt16	Jedes Bit ist Stellvertreter (siehe Spez.)
XXXSP	Indiziert Eingriff des Systems	Bool	0: kein Eingriff 1: Eingriff
XXXst	Regler-Modus, der aktuell aktiv ist	UInt8	1: Comfort 2: Dynamic 3: Sport [1/3]

Tabelle A.3: Extrahierte Informationen zu den Signalen

1.) **Datentyp ermitteln** [aus TargetLink™-Data-Dictionary]

- BOOL: ⇒ falscher Wert
  ⇒ Wert unerwünscht *wahr*
  ⇒ Wert unerwünscht *nicht wahr*
  → (Implementierungsfehler ermitteln)
- UINT / INT: ⇒ falscher Wert
  → (Datensemantik ermitteln)
  → (Wertebereich ermitteln)
  → (Implementierungsfehler ermitteln)
- FLOAT: ⇒ falscher Wert
  ⇒ Betrag größer als Sollwert
  ⇒ Betrag kleiner als Sollwert
  ⇒ Wert ist unerwünscht 0
  ⇒ Wert ist unerwünscht $\neq 0$
  ⇒ Betrag und/oder Vorzeichenfehler
  → (Wertebereich ermitteln)
  → (Äquivalenzklassen ermitteln)
  → (Implementierungsfehler ermitteln)

2.) **Wertebereich ermitteln** [aus TargetLink™-Data-Dictionary]

- Wert max im Wertebereich des Datentyps: ⇒ Wert überschreitet max
- Wert min im Wertebereich des Datentyps: ⇒ Wert unterschreitet min

3.) **Datensemantik ermitteln** [aus TargetLink™-Data-Dictionary]

- Bit-Semantik semBX gegeben: ⇒ B0 (semB0) unerwünscht *wahr*
  ⇒ B0 (semB0) unerwünscht *nicht wahr*
  ⇒ B1 (semB1) unerwünscht *wahr*
  ...
- Wert-Semantik semX gegeben: ⇒ Wert ist unerwünscht sem0
  ⇒ Wert ist unerwünscht sem1
  ...

4.) **Äquivalenzklassen ermitteln** [aus Intervall-Abstraktion]

- Intervalle I_x ermittelt: ⇒ Wert unerwünscht im Intervall I_0
  ⇒ Wert unerwünscht im Intervall I_1
  ...

5.) **Implementierungsfehler ermitteln** [aus Implementierungsmodell]

- Fehler impl_X konnten ermittelt werden: ⇒ Fehler impl_1
  ⇒ Fehler impl_2
  ...

Tabelle A.4: Ausgabemodifikations-Zuweisung in der Fallstudie

Berechnung: Gesamt-Soll-Differenzmoment (XXXom [Float32, Nm, -10/10])	
generierte Fehler	*überarbeitete Fehler*
- falscher Wert	- falsches Moment (allgemein) - falsches Moment (Extrapolation bei Haftungsverlust) - falsches Moment (weniger 10% Abweichung)
- Wert ist unerwünscht 0	- Wert ist unerwünscht 0 Nm (Gradientenbegrenzung: 0 bis Max. in 10 Sekunden) - Nach Fahrmanöver bleibt der Wert 0 (keine Wirkung während Manöver)
- Wert ist unerwünscht $\neq 0$	- Werte werden weiterhin berechnet
- Betrag größer als Sollwert	- Betrag größer als Sollwert (allgemein) (Gradientenbegrenzung: 0 bis Max. in 10 Sekunden) - Betrag größer als Sollwert (mit Wertsprung) - keine Temperaturreduktion (Gradientenbegrenzung: 0 bis Max. in 10 Sekunden)
- Betrag kleiner als Sollwert	- Betrag kleiner als Sollwert (allgemein) (Gradientenbegrenzung: 0 bis Max. in 10 Sekunden) - Betrag kleiner als Sollwert (mit Wertsprung) - ungewollte Temperaturreduktion (Gradientenbegrenzung: 0 bis Max. in 10 Sekunden)
- Vorzeichenf. (+Betrag)	
	- falsches Fahrprogramm
- Wert überschreitet 10	- Moment überschreitet 10 Nm
- Wert unterschreitet -10	- Moment überschreitet -10 Nm

Tabelle A.5: Zugewiesene Fehler zum Differenzenmoment

Anhang B

Ermittlung der Funktionsabhängigkeiten in Komponenten

B.1 Ausgabeabhängigkeiten

Die Ermittlung der Abhängigkeiten zwischen Ausgaben erfolgt über eine syntaktische Analyse. Im Fall der Simulink-Komponenten wird der Datenfluss, bei den eingebetteten C-Code-Modulen wird der Anweisungsfluss verfolgt. Nicht analysiert werden hier Automaten. Für sie wird eine vollständige Vernetzung angenommen.

Schritte

Technisch findet die Ermittlung des Funktionsnetzes für Ausgaben in der Fallstudie in drei Schritten statt. Da die C-Code-Module immer in Simulink-Module eingebettet sind, und nie umgekehrt, werden, wie in Abbildung B.1 dargestellt, zuerst die C-Code-Abhängigkeiten ermittelt. Darauf folgend werden die Abhängigkeiten zwischen den Eingaben und Ausgaben der Systeme berechnet. In einem dritten Schritt werden dann die Ausgaben entlang der Systemhierarchie vernetzt.

C-Code

Abschätzung

Die syntaktische Analyse der Zusammenhänge zwischen Eingaben und Ausgaben liefert eine Abschätzung, die kein Verhalten berücksichtigen kann. Eine Abhängigkeit von einer Variable a zu einer Variable b wird dann angenommen, wenn:

- *Zuweisung* — Die Variable a in einem Ausdruck enthalten ist, dessen Ergebnis der Variable b zugewiesen wird.
- *Verzweigung* — Die Variable a in einem Ausdruck enthalten ist, der eine Programmverzweigung beeinflusst und innerhalb der bedingten Zweige eine Zuweisung zur Variablen b stattfindet.

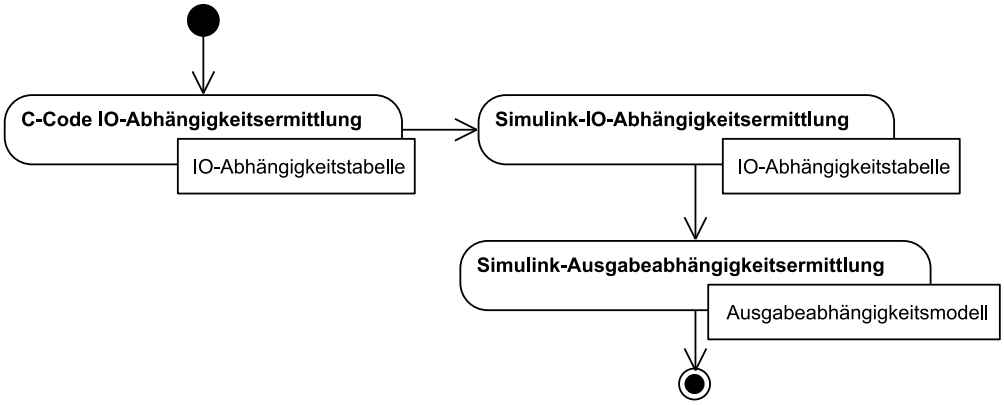


Abb. B.1: Vorgehen zur Ermittlung des Funktionsnetzes (Ausgabevernetzung)

Richtlinien Die Güte dieser Art der Abschätzung hängt davon ab, in wie fern innerhalb des C-Codes Strukturen vorhanden sind, bei denen das Verhalten interpretiert werden muss, um Zusammenhänge auszuschließen. Weiter können keine zeitbedingten Abhängigkeiten ausgeschlossen werden. Daraus ergeben sich eine Menge von Richtlinien, welche auch eine starke Überschneidung mit dem MISRA-Standard haben. Die Richtlinien werden im Folgenden angegeben:

- Innerhalb der Module dürfen nur eindeutige, globale Variablen-Namen vergeben werden. Es darf keine Verschattung von Variablen stattfinden.
- Innerhalb der Sub-Blöcke von Modulen dürfen keine lokalen Variablen deklariert werden.
- Temporäre Variablen müssen eindeutig sein und dürfen nicht mehrmals verwendet werden. Folgendes Konstrukt liefert zum Beispiel eine zu große Abhängigkeitsmenge:
$$(t := a; b := t; t := c; d := t) \overset{?}{\Rightarrow} (a \overset{FKT}{\dashrightarrow} d)$$
- Variablen dürfen nicht temporär mit Werten belegt werden, wenn diese keine Wirkung auf den endgültigen Wert der Variable haben. Folgendes Konstrukt liefert eine zu große Abhängigkeitsmenge:
$$(c := a; c := b) \overset{?}{\Rightarrow} (a \overset{FKT}{\dashrightarrow} c)$$

Darstellung Die einzelnen Abhängigkeiten können tabellarisch oder graphisch dargestellt werden. Die graphischen Darstellungen, wie in Abbildung B.2[1] werden schnell unübersichtlich. Allerdings geben sie einen guten Eindruck, wie stark die Vernetzung der Eingaben und Ausgaben ist. Tabellarisch lassen sich die einzelnen Abhängigkeiten gut auslesen und maschinell verarbeiten, sind aber vom Aussehen nicht intuitiv hinsichtlich des Grads der Vernetzung.

[1] Ist der Datentyp noch nicht ermittelt, so wird stellvertretend im Prototyp *UNKNOWN* ausgegeben.

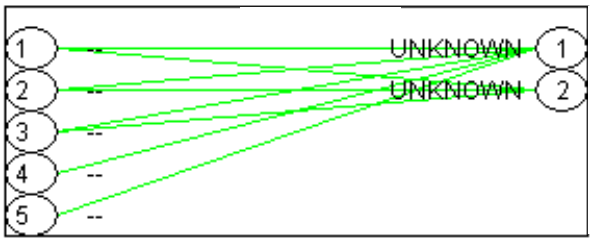


Abb. B.2: Erkannte Abhängigkeiten aus syntaktischer C-Code-Analyse.

Modulname:	XXX_A	XXX_B	XXX_C	XXX_D
Abhängigkeiten bei Vollvernetzung (a_a):	10	340	39	77
Anzahl entdeckter Abhängigkeiten (a_f):	8	220	25	75
Anzahl entfernter Abhängigkeiten (a_r):	2	120	14	2
Ausschlussquote (a_a/a_r):	0.2	0.3	0.3	0.0

Tabelle B.1: Funktionsabhängigkeiten aus syntaktischer C-Code-Analyse.

Bewertung

Die Ermittlung der Funktionszusammenhänge läuft weitgehend vollautomatisch. Es müssen lediglich die entsprechenden Dateien zusammengestellt werden, was bei der Fallstudie einen Aufwand von etwa einer Stunde fordert. Der Rechenaufwand beträgt mit der prototypischen Implementierung mehrere Stunden für 32 C-Code-Module. Mit der Erstellung des Funktionsnetzes können in der Fallstudie in etwa ein Viertel der Zusammenhänge einer Vollvernetzung ausgeschlossen werden (siehe Beispiel in Tabelle B.1). Um diesen Faktor kann die Fehlerauswirkungsanalyse effizienter gestaltet werden.

Simulink-Datenfluss

Abschätzung

Bei Komponenten, die in Simulink modelliert sind, wird der Signalfluss mittels einer Breitensuche verfolgt. Die Abschätzung entspricht dann der einfachen Fragestellung: Gibt es einen Pfad im Signalfuss-Graphen, der von der Eingabe bzw. Ausgabe a zu der Ausgabe b führt? In dem Prototypen werden hierzu die Abhängigkeitsgraphen von den Blättern hin zur Wurzel der Systemhierarchie aufgebaut. Es wurden sowohl die Eingaben und Ausgaben eines Systems verknüpft, wie auch die Ausgaben von Systemen mit den Ausgaben der jeweiligen Subsysteme.

Richtlinien

Da die Zusammenhänge zwischen Signalen über den Signalfluss abgeschätzt werden, steigt die Güte der Abschätzung, je weniger im Signalfluss nicht sichtbare Operationen vorhanden sind. Daraus ergeben sich für die Modellierung zwei Richtlinien:

Modulname:	XXX_A	XXX_B
Abhängigkeiten bei Vollvernetzung (a_a):	64	48
Anzahl entdeckter Abhängigkeiten (a_f):	29	38
Anzahl entfernter Abhängigkeiten (a_r):	35	10
Ausschlussquote (a_a/a_r):	0.5	0.2

Tabelle B.2: Funktionsabhängigkeiten aus syntaktischer Datenfluss-Analyse.

- In den Modellen sind keine Zusammenhänge zwischen Signalen, welche nicht explizit im Datenfluss sichtbar sind.
- Die Modelle enthalten keine Tautologien, Multiplikationen mit der Konstante 0 oder andere Operationen, die Ergebnisse permanent aufheben.

Verletzungen In der Fallstudie sind in einem Modell mit über 50 Modulen nur 3 Stellen, bei denen diese Richtlinien verletzt sind. Zu Zwecken der Übersichtlichkeit sind an einigen Stellen Tautologien enthalten, die aber keine nennenswerte Wirkung auf die Angabe der Funktionszusammenhänge hatten. Kritischer ist ein Modul, in dem Daten in einen EEPROM gespeichert werden und bei Neustart des Systems in einem anderen Modul wieder eingelesen werden. Hier müssen die Zusammenhänge explizit eingetragen werden.

Bewertung Die Ermittlung der Zusammenhänge zwischen den Signalen der Systeme läuft automatisch. Bei der prototypischen Anwendung dauerten die Berechnungen knapp zwei Tage, was aber unter anderem auf eine ineffiziente Implementierung zurückzuführen ist. Allgemein ist damit aber gezeigt, dass diese Generierung auch auf große Systeme anwendbar ist. Der Nutzen der Funktionsvernetzung, also der identifizierte Anteil der Zusammenhänge, bei denen keine Fehlerwirkung bestehen kann, liegt in der Fallstudie bei ca. 20% (siehe Tabelle B.2).

B.2 Verhaltensbedingungen

Werkzeuge Die teilautomatisierte Ermittlung des Funktionsnetzes für Verhaltensbedingungen wird in der Fallstudie mittels der Verifikationswerkzeuge SAL[2] und Scade™ umgesetzt. Im Weiteren wird hier SAL beschrieben, da dessen Sprache besser lesbar ist. Beide Werkzeuge basieren auf einem SAT-Solver, der Aussagenlogik verarbeiten kann. Weiter ist es in beiden Werkzeugen möglich, lineare Logik einzubinden. Die Mächtigkeit der Werkzeuge ist also Aussagenlogik modulo linearer Gleichungen. Beide Werkzeuge arbeiten relativ effizient in großen Variablenräumen. Allerdings hatten beide Werkzeuge Defizite bei der Verfügbarkeit. So lieferten die Werkzeuge zwar korrekte Ergebnisse, aber die Anfragen terminierten in den meisten Fällen leider nicht. Trotzen wird hier vorgestellt, wie die Analysen bei der Fallstudie umgesetzt werden, auch in der Hoffnung leistungsfähigere Werkzeuge in der Zukunft.

[2] http://sal.csl.sri.com

Anforderungen

Die Funktionsweise beider Werkzeuge fordert eine modellierte Komponente zu welcher dann Eigenschaften überprüft werden können. Die Anforderungen werden in einem Modul des Verifikationswerkzeugs umgesetzt. Alle Variablen, die in den Anforderungen vorkommen, sind Eingänge des Moduls. Jede Anforderung wird durch einen booleschen Ausgang repräsentiert, der angibt, ob eine Anforderung gültig ist. Die Anforderungen werden in dem Verhalten des Moduls definiert, in dem der Ausgabe mittels eines Ausdrucks die Semantik der Anforderung zugewiesen wird. Nachteil dieser Umsetzungsart ist der damit verbundene globale Namensraum. Bei hierarchischen Systemen mit geschachtelten Namensräumen werden diese erst flachgedrückt.

Beispiel

Tabelle B.3 zeigt eine stark vereinfachte formalisierte Anforderungsstruktur einer Ausgabesteuerung der Software eines Fahrwerkregelsystems. Die hier vereinfachte Teilaufgabe besteht darin, bei Missständen, also dem gesetzten Flag XXXOM den Eingriff des Differentials mittels der Ausgabe flagEingriffXXX zu unterbinden. Dies wird mit der Anforderung r1_01 (Modul Nr. 1, Anforderung Nr. 01) ausgedrückt. Um die Umsetzung nachvollziehbar zu machen, wurden nur einfache boolesche Variablen verwendet. So repräsentiert die Variable idxXXX_is_10 die Tatsache, dass die Variable den Wert 10 hat. In der Ebene darunter wird die Anforderung über zwei Pfade realisiert, die auf die Subsysteme 11, 12 und 13 verteilt sind.

Verfeinerung

Voraussetzung für die Ermittlung des Funktionsnetzes ist, dass die Anforderungen der verschiedenen Systemebenen hinsichtlich der Verfeinerungsbeziehung korrekt sind. Die Prüfung der Verfeinerung geschieht über Theoreme (siehe Tabelle B.4), welche gültig sein müssen. Für jede Anforderung auf der höheren Ebene wird ein Theorem aufgestellt, welches prüft, ob die Summe der darunter liegenden Anforderungen diese impliziert. Bei Verwendung eines Werkzeuges mit Aussagenlogik, wie SAL, darf die Anforderung auf höherer Ebene keine lokalen Variablen haben. Wäre dies der Fall, so müsste ein FOL-Werkzeug verwendet werden. Diese Überprüfung wird für jede Anforderung einmal durchgeführt, wenn nicht sicher ist, ob die Anforderungen korrekt sind.

Funktionsnetz

Die in Kapitel 5 vorgestellte formale Definition einer Anforderungsabhängigkeit ist in einem auf Aussagenlogik basierenden Werkzeug nicht möglich. Eine Umsetzung sähe bei ausreichend mächtigen Werkzeugen wie in Tabelle B.5 oben aus. Der Existenzquantor aus Definition 5.2.2 wird mittels boolescher Variablen angegeben, die dann mit einer Oder-Verknüpfung an die eigentlichen Variablen gebunden werden.

Unterstützung

Da die Funktionsvernetzung nicht garantiert vollautomatisch erstellt werden kann, ist es eine Alternative, ähnlich wie bei Tests, die Analysen als Unterstützung zur Funktionsvernetzung zu verwenden. Hierbei eignen sich zwei Arten der Unterstützung. Zum Einen besteht die Möglichkeit, dass Anforderungen eine Ebene tiefer nicht verteilt wurden, sondern nur konkreter formuliert wurden. In diesen Fällen kann man die Zusammenhänge erkennen, in dem man einfach die einzelnen Anforderungen auf Implikation prüft. Ist dies der Fall, so kann trivialerweise auf die Definition 5.2.2 geschlossen werden, in dem die leeren Existenzquantoren angenommen werden. Tabelle B.5 zeigt in der Mitte diese Prüfung exemplarisch. Die zweite Art der Unterstützung ist die Ermittlung der Zusammenhänge zwischen Einfachfeh-

SAL

```
requirements: MODULE =
 BEGIN
   INPUT XXXOM : BOOLEAN
   INPUT flgXXX : BOOLEAN
   INPUT idxXXX_is_10 : BOOLEAN
   INPUT flagEingriffXXX: BOOLEAN
   INPUT trqXXX_is_0 : BOOLEAN

   OUTPUT r1_01: BOOLEAN
   OUTPUT r1_02: BOOLEAN
   OUTPUT r11_01: BOOLEAN
   OUTPUT r12_01: BOOLEAN
   OUTPUT r13_01: BOOLEAN
   OUTPUT r13_02: BOOLEAN
   OUTPUT r13_03: BOOLEAN

   DEFINITION
     r1_01 = XXXOM => NOT(flgEingriffXXX);
     r1_02 = XXXOM => trqXXX_is_0;
     r11_01 = XXXOM => flgXXX;
     r12_01 = flgXXX => idxXXX_is_10;
     r13_01 = idxXXX_is_10 => NOT(flgEingriffXXX);
     r13_02 = flgXXX <=> NOT(flgEingriffXXX);
     r13_03 = idxXXX_is_10 => trqXXX_is_0;
 END;
```

Tabelle B.3: Definition eines Moduls zur Anforderungsanalyse

lern. Führt ein Einfachfehler einer Anforderung eines Subsystems zu eine Verletzung der Anforderung des umgebenden Systems, so kann auch hier auf eine Anforderungsabhängigkeit geschlossen werden, in dem die leeren Existenzquantoren in Theorem 5.2.2 eingesetzt werden. Die Erwägung, die Abschätzungen mit der Überprüfung von Zweifachfehlern oder der Verfeinerung auf zwei Sub-Anforderungen zu überprüfen, macht kombinatorisch keinen Sinn. Bei einem System mit 20 Anforderungen wären bereits statt 40 Anfragen 380 Anfragen notwendig.

Bewertung Die teilautomatisierte Ermittlung in der Fallstudie zeigte, dass die in dieser Arbeit gegebenen Definitionen für Funktionsnetze anwendbar sind. Allerdings zeigte sich in der Praxis, dass formale Anforderungen auf höheren Abstraktionsebenen meist nur spärlich vorhanden sind. Dies liegt unter anderem in der Charakteristik der Systeme der Fallstudie, da hier starkes Prototyping betrieben werden musste. Die größte Einschränkung der Anwendung in der Praxis liegt bei den Verifikationswerkzeugen selbst. Diese sind zwar sehr effizient, es wird aber ein deutlich höheres Maß der Verfügbarkeit benötigt, um eine realistische Anwendung in der Praxis zu finden.

SAL

```
isVerfeinerung_r1_01: THEOREM requirements |-
    G(r11_01 AND r12_01 AND r13_01 AND r13_02 AND r13_03 => r1_01);
```

Tabelle B.4: Anfrage zur Verfeinerungsanalyse

SAL

```
%% n:m Funktionsabhängigkeit nicht ermittelbar (First Order Logic!)
  fkt__r13_02__r1_01: THEOREM EXISTS (a1,a2,a3,a5:BOOLEAN):
      requirements |-
        G(((r11_01 OR a1) AND (r12_01 OR a2) AND
          (r13_01 OR a3) AND r13_02 AND (r13_03 OR a5)
          => r1_01))
        AND NOT
        G(((r11_01 OR a1) AND (r12_01 OR a2) AND
          (r13_01 OR a3) AND (r13_03 OR a5)
          => r1_01));

%% Deshalb Annäherung über direkte Anforderungsverfeinerung ...
  isDirectRefinement: THEOREM requirements |- G(r11_01 => r1_01);

%% ... und Annäherung über Einfachfehler
  isNoSPF__r11_01__r1_01: THEOREM requirements |-
      G(r12_01 AND r13_01 AND r13_02 AND r13_03 => r1_01);
```

Tabelle B.5: Anfragen zur Funktionsnetzanalyse

Anhang C

Ermittlung der Fehlerabhängigkeiten

C.1 Verhaltensbedingungen

bayessche Modifikation

Aufbauend auf der Funktionsvernetzung aus Anhang B.2 kann die Fehlervernetzung umgesetzt werden. Werden zu den Anforderungen nur bayessche Modifikationen $[\![\mathcal{M}_r]\!] = (true, true)$ zugewiesen, so können die Anfragen aus Tabelle B.5 verwendet werden, um eine Fehlervernetzung zwischen einzelnen Anforderungen zu erlangen.

F-Mengen

Sollen statt bayesscher Modifikationen komplexere Modifikationen verwendet werden, so müssen die Anfragen entsprechend der Beschreibung aus Abschnitt 5.2.1 zerlegt werden. Es wird jeweils eine Anfrage zur Prüfung der korrekten Angabe der F-Mengen und eine Anfrage zur Prüfung der E-Mengen erstellt. Tabelle C.1 oben zeigt ein Beispiel für die Modifikation zweier Anforderungen. Zur Bestimmung der Abhängigkeit der F-Mengen wird die Anforderungsmodifikation der unteren Ebene angewendet und auf eine Implikation der um F erweiterten Anforderung geprüft. Wie für die Anforderungen bei der Funktionsvernetzung gilt für die Beschreibung der F-Menge der Modifikation der höheren Ebene, dass diese keine lokalen Variablen enthalten darf, wenn die Analyse mit Aussagenlogik realisiert wird. Ein Beispiel für eine Anfrage hinsichtlich der F-Mengen ist die erste Anfrage in Tabelle C.1.

E-Mengen

Die Modifikation der E-Mengen ist mit Aussagenlogik auf die Fälle begrenzt, in denen keine Dekomposition stattfindet. Ursache ist, dass zu deren Überprüfung die Implikation der Formel in umgekehrte Richtung läuft. So stehen durch die Zerteilung der Architektur nach der Implikation lokale Variablen. Dies fordert einen Existenzquantor zur Ermittlung der lokalen Variablen. Die Ermittlung der Abhängigkeit der E-Mengen fordert deshalb fast immer ein Analysewerkzeug mit FOL-Mächtigkeit. Tabelle C.1 zeigt mittig eine Beispielanfrage. In dieser Anfrage ist zu erkennen, dass die Modifikation der oberen Ebene auf die Anforderungen der unteren Ebene angewendet wird. Der Grund ist die mögliche Abstraktion des Verhaltens zur nächst höheren Ebene, welche zur Verifikation der E-Mengen ausgeblendet werden muss.

SAL

```
%% minimale Modifikationen
%% e01__r1_01 = NOT XXXOM => flgEingriffXXX;
%% f01__r1_01 = NOT XXXOM => flgEingriffXXX;
%% e02__r11_01 = NOT flgXXX;
%% f02__r11_01 = XXXOM AND NOT flgXXX;

%% Einfachfehler F-Mengen
  isNoSPF__m02_m01: THEOREM requirements |-
      G(((r11_01 AND e02__r11_01 ) OR f02__r11_01) AND
        r12_01 AND r13_01 AND r13_02 AND r13_03
        =>
        (r1_01 OR f01__r1_01));

%% Einfachfehler E-Mengen (nicht ermittelbar wg. FOL)
  isNoSPE__m02_m01: THEOREM requirements |-
      G(e01__r1_01 AND r11_01 AND
        r12_01 AND r13_01 AND r13_02 AND r13_03
        =>
        EXISTS (l1, l2): RENAME flgXXX TO l1, idxXXX_is_10 TO l2 IN
          (((r11_01 AND e02__r11_01 ) OR f02__r11_01) AND
          r12_01 AND r13_01 AND r13_02 AND r13_03));

%% Einfachfehler F-Mengen -> Systemmodifikation
  isNoSPF__m02_m01: THEOREM requirements |-
      G(((r11_01 AND e02__r11_01 ) OR f02__r11_01) AND
        r12_01 AND r13_01 AND r13_02 AND r13_03
        =>
        (( r1_01 AND r1_02 ) OR f01__r1_01));
```

Tabelle C.1: Anfragen zur Fehlernetzanalyse

System-Modifikation

Geht die Vernetzung über mehrere Ebenen, so ist zu berücksichtigen, dass die Folge auf darüber liegenden Ebenen mehrere Anforderungen betreffen kann. Betrachtet man zur dann folgenden Ebene wiederum die Anforderungen einzeln, so kann es sein, dass eine Potenzierung der Fehler übersehen wird. Eine Maßnahme ist, statt einzelner Anforderungen auf nächst höherer Ebene zu prüfen, die Folge für das Gesamtsystem, also die Summe aller Anforderungen zu prüfen. Auf diese Weise erhält man für Zwischenebenen die Möglichkeit, Einfachfehler bis hin zur Nutzungsschnittstelle ohne übersehene Fehlerpotenzierungen zu verfolgen. An der Nutzungsschnittstelle können die Folgen dann, wenn gewünscht, wieder einzeln betrachtet werden. Tabelle C.1 zeigt unten eine Beispielanfrage für die F-Mengen. Bezüglich der E-Mengen besteht in den Anfragen kein Unterschied.

Bewertung Die Ermittlung der Zusammenhänge zwischen Fehlern mit SAT-Solvern kann im Falle von bayesschen Anforderungsmodifikationen automatisiert werden, wenn die Folgeanforderungen keine lokalen Variablen haben. Liegen jedoch komplexere Modifikationen vor, so sind die Zusammenhänge der F-Mengen mit SAT-Solvern bestimmbar, die E-Mengen jedoch nicht. In der Praxis wurden zu Anforderungen fast nur bayessche Modifikationen vorgefunden, welche auch automatisiert analysierbar wären. Es zeigte sich leider auch hier, das die Verifikationswerkzeuge sehr oft keine Ergebnisse lieferten, womit eine praxistaugliche Anwendung der Analyse noch von einer Leistungssteigerung der Werkzeuge abhängt.

C.2 Ausgaben

Die Ermittlung der Zusammenhänge zwischen Fehlern zu Ausgaben wird mittels eines Parallelvergleichs umgesetzt (siehe Abschnitt 5.2.1.1). Dieser ist zwar vom Rechenaufwand intensiver, kann aber sowohl mittels Tests angenähert werden, wie auch mit komplexen Anforderungen und Systembeschreibungen umgehen. Der Grund für die Wahl des Parallelvergleichs ist, dass sich in der Fallstudie die Berechnungen der Ausgaben meist aus komplexen Funktionen mit lokalen Variablen zusammensetzen. Es wurden zwei Umsetzungen zur Unterstützung der Analysen realisiert. Zum Einen wurden vorhandene Testfälle manipuliert und zum Anderen wurden die Abweichungsfolgen wenn möglich mittels eines SAT-Solvers ermittelt.

Analyse über Tests

Tests finden über exemplarische Ausführungen des Systems statt, bei dem die Soll-Ausgaben mit den Ausgaben verglichen werden. Die Ermittlung von Folgefehlern ist deshalb an die Zusammenhänge zwischen Eingaben und Ausgaben gebunden (siehe Abschnitt 5.1.2). Zur Ermittlung eines Folgefehlverhaltens können Tests direkt zur Ermittlung entlang der Systemhierarchie verwendet werden oder sie können verwendet werden, um indirekt über die Eingabe- und Ausgabe-Beziehungen auf Folgefehler zu schließen.

Fehlverhalten Die direkte Ermittlung der Zusammenhänge entlang der Systemhierarchie funktioniert ähnlich, wie der im Folgenden erklärte Testaufbau für die Ermittlung der Ein-Ausgabeabhängigkeiten. Das Verfahren besteht im Wesentlichen darin, mit dem korrekten System und den Eingabevektoren der Anforderungstests Referenzvektoren für die Ausgabe zu erzeugen. Mit den gleichen Eingabevektoren wird dann das fehlerhafte System ausgeführt. Die Ausgabevektoren werden mit den Referenzvektoren verglichen, und so auf Folgefehler im Sinne von Ausgabemodifikationen geschlossen. Findet die Ermittlung der Folgen über mehrere Ebenen einer Systemhierarchie statt, so ergeben sich die gleichen Aufgaben wie bei der indirekten Ermittlung über Ein- und Ausgaben. Deshalb wird im folgenden nur die indirekte Ermittlung weiter beschrieben.

Ein-Ausgabe-Ketten

Die Umsetzung der indirekten Ermittlung der Zusammenhänge findet mittels einer Kombination aus Tests und syntaktischer Analyse statt. Sind die Zusammenhänge zwischen Abweichungen bei der Eingabe und Abweichungen bei der Ausgabe über Tests erkannt, so kann daraus bei rückkopplungsfreien Systemen die Gesamtfolge abgeschätzt werden. Die meisten Systeme der Fallstudie sind oberhalb der Modul-Ebene rückkopplungsfrei. Lediglich in einem System wurde eine Rückkopplung implementiert und explizit in einem Modul festgehalten. An dieser Stelle ist die Abstraktion der Zeit (Abschnitt 5.2.2.1) zu berücksichtigen.

Testaufbau

Die verwendete Testumgebung EXACT™ wird in der Entwicklung verwendet, um Systeme zu testen. Es handelt sich hierbei um eine reine Blackbox-Test-Umgebung. Die Umsetzung der Tests in der EXACT™-Umgebung ist in Abbildung C.1 dargestellt. In einem ersten Schritt werden die vorhandenen Testeingaben der Entwicklung mit dem Modul verwendet, um die Soll-Ausgaben bei korrekten Eingaben als Referenz zu ermitteln. Zu jedem Eingabevektor erhält man so einen Referenzvektor für die Ausgabe. In einem zweiten Schritt findet dann die Manipulation der Eingaben statt. Hier werden die Eingaben mit möglichst aussagekräftigen Abweichungen manipuliert, um Fehlerzusammenhänge zu entdecken. Das zu testende Modul wird in einem dritten Schritt mit den manipulierten Eingabevektoren ausgeführt und die Ausgabevektoren gespeichert. In einem vierten Schritt werden Unterschiede zwischen Ausgabevektoren und den Referenzvektoren ermittelt und damit auf eine Folgeabweichung geschlossen. Möchte man bei diesem Aufbau Fehler entlang der Systemhierarchie ermitteln, so ist statt der Eingabevektoren das System zu manipulieren. Allerdings hat das System der Fallstudie diese Funktion nicht geboten. Deshalb wird im Weiteren auf die Ein-Ausgabe-Abhängigkeiten eingegangen.

Testmanipulation

Eine spezielle Herausforderung der automatisierten Zuweisung von aussagekräftigen Abweichungen der Eingaben ist, dass die Eingabevektoren bei der verwendeten Werkzeug-Konstellation nicht ausgelesen werden können. Während man bei einer manuellen Eingabevektor-Manipulation nur die aussagekräftigen Eingaben mit Abweichungen versieht und testet, ist es bei der prototypischen Implementierung hier notwendig, jeden Eingabevektor mit jeder Abweichung zu manipulieren. Dadurch steigt die Anzahl der notwendigen Testläufe um ein Vielfaches. Die Problematik sinnloser Manipulationen und Testläufe ist in Abbildung C.2 verdeutlicht. Für eine Abweichung, die statt einem Wert kleiner x einen größeren Wert liefern soll, muss die Eingabe auch Werte kleiner x enthalten. Auf dem linken Eingabevektor kann die Modifikation nicht angewendet werden, da kein passender Eingabevektor vorhanden ist. Auf der rechten Seite kann hingegen die Modifikation in einigen Bereichen (mit Balken gekennzeichnet) angewendet werden. Die gleiche Problematik trifft im Falle der direkten Ermittlung von Folgefehlverhalten zu, in denen die manipulierten Systeme entsprechend aufgerufen werden müssen.

Äquivalenzklassen

Eine weitere Herausforderung ist die Angabe einer passenden Modifikation zu den Eingabevektoren. Bei vorhandenem Wissen über Äquivalenzklassen von Werten und Gradienten könnten passende Modifikationen zu den Eingabevektoren ermittelt werden. In der Fallstudie sind diese Daten nur begrenzt vorhanden bzw. mit der vorhan-

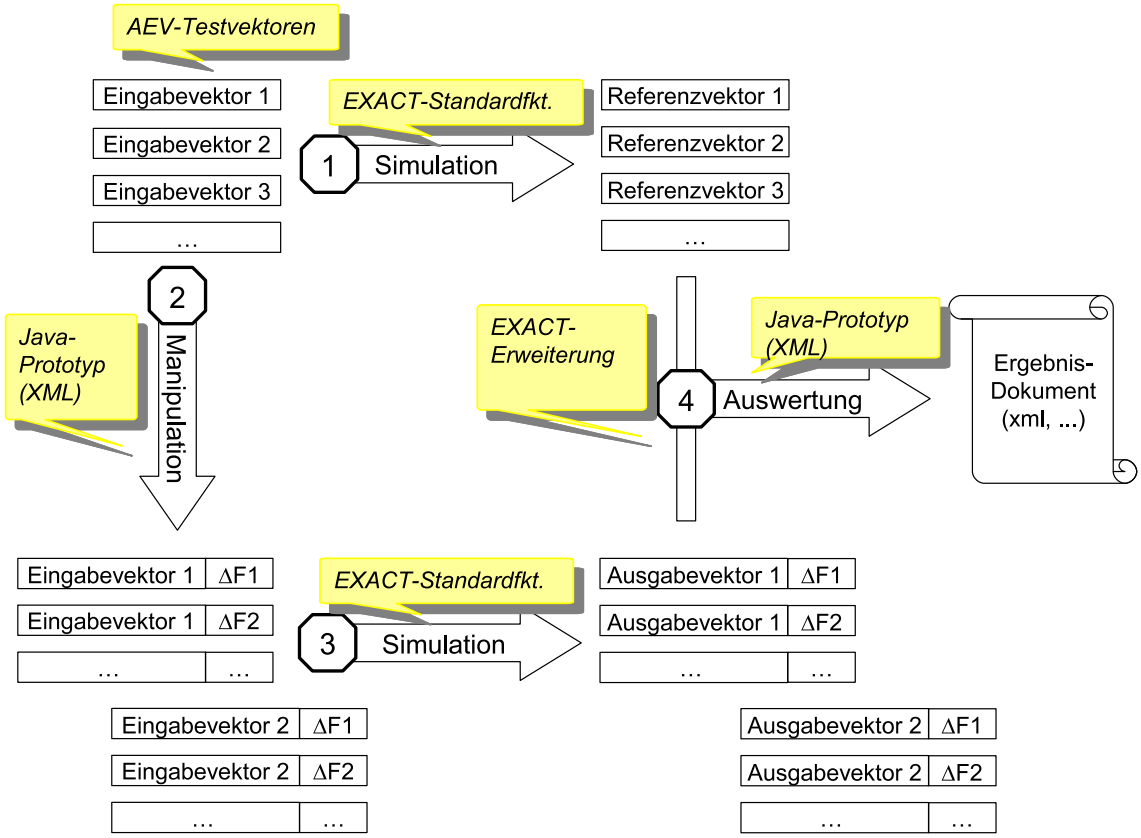


Abb. C.1: Ermitteln der Eingabe-Ausgabeabhängigkeiten mit Tests

denen Software nur manuell zu erstellen. Ohne Ermittlung von Äquivalenzklassen ist das Auffinden einer Eingabe-Modifikation mit aussagekräftiger Wirkung an der Ausgabe nicht gezielt, sondern nur über zufällige Manipulationen möglich. Nimmt man für eine Abweichungsklasse jeweils nur einen Vertreter, so sinkt die Erkennung der gefundenen Folgen teilweise auf unter 50%.

Das in der Fallstudie implementierte Erkennen von Fehlerzusammenhängen mittels Testfällen zeigt, dass Testfälle die Ermittlung von Folgefehlverhalten unterstützen, aber den Einsatz komplexer Testmethoden benötigen, um Ergebnisse mit passender Qualität zu liefern. Tabelle C.2 zeigt die Effizienz der implementierten Ermittlung der Zusammenhänge über Testfälle. Wesentlich an dieser Tabelle ist zu erkennen, dass der große manuelle Aufwand im Ausschluss der Fehlerzusammenhänge liegt. Diese Leistung können Tests jedoch nie liefern. Deshalb werden im nächsten Abschnitt Techniken verwendet, die den Ausschluss einer Folge ermöglichen.

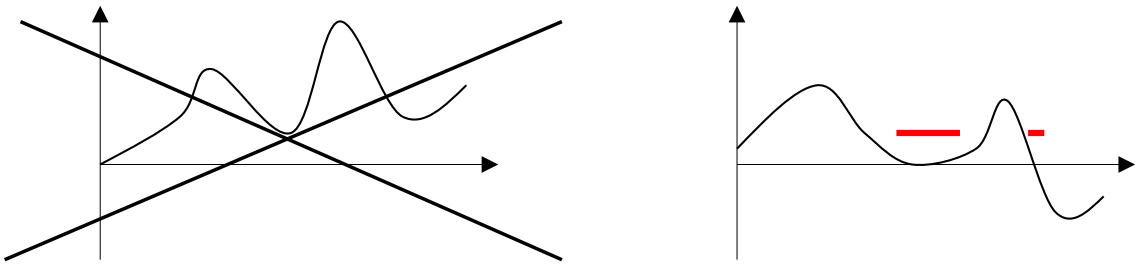


Abb. C.2: Manipulierbarkeit von Eingabevektoren

Modulname:	XXX_A
potenzielle Fehlerabhängigkeiten:	632
gefundene Abhängigkeiten über Tests:	32
systematisch abgeleitete Abhängigkeiten :	18
manuell angegebene Fehlerabhängigkeiten:	73
systematisch ausgeschlossene Abhängigkeiten:	172
manuell ausgeschlossene Fehlerabhängigkeiten:	329
Trefferquote der Tests:	$32/123 = 0,26$

Tabelle C.2: Fehlerzusammenhänge aus Testfällen

Analyse über SAT-Solver

Die Umsetzung der Ermittlung der Fehlerzusammenhänge mittels SAT-Solvern hat im Vergleich zu der Ermittlung mit Testfällen den Vorteil, dass hier auch Zusammenhänge zwischen Fehlern explizit ausgeschlossen werden können. Mit diesen Werkzeugen kann also komplementär zu der über Testfälle entdeckten Zusammenhangsmenge eine Menge von Zusammenhängen ausgeschlossen werden. Allerdings gilt auch hier, dass ein vollständiger Ausschluss aller Zusammenhänge oft wegen der Leistung der Verifikationswerkzeuge nicht möglich ist, und so die Modellierung der Zusammenhänge nur unterstützt werden kann.

Effizienz

Die Ermittlung der Zusammenhänge kann hier wie bei den Testfällen direkt geschehen, oder indirekt über Ein-Ausgabe-Ketten. Im Vergleich zu der Anwendung der Testumgebung findet mit SAT-Solvern im übertragenen Sinne ein vollständiger Test statt, der die Glass-Box-Sicht nutzt. Deshalb sind die analysierbaren Systeme tendenziell kleiner. Weiter ist ein Nachteil der Verifikation, dass hier pro Folgefehler eine eigene Verifikation angestoßen werden muss und nicht wie bei der Testumgebung auf mehrere alternative Folgefehler gleichzeitig geprüft werden kann.

Aufbau

In der Fallstudie wird die Verifikation mit dem Werkzeug SCADE™ durchgeführt. Dieses Werkzeug wurde gewählt, da es im Vergleich zu anderen Werkzeugen sehr schnell ist und dessen Einsatzmöglichkeiten in der Entwicklung auch an anderen Stellen untersucht wurden. Die Sprache des Werkzeugs ist Lustre. Die Schritte für eine Analyse in der prototypischen Umsetzung bestehen darin, das zu untersuchen-

de System in die Sprache Lustre zu übersetzen, ein weiteres modifiziertes System zu kreieren, die Anfrage für den Parallelvergleich zu erstellen und schließlich die Verifikation durchzuführen und auszuwerten. Der Ablauf ist in Abbildung C.3 grob skizziert.

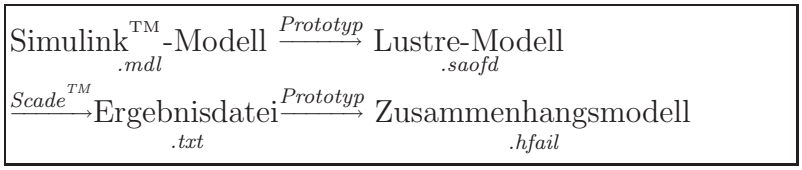


Abb. C.3: Ermitteln der Eingabe-Ausgabeabhängigkeiten über Scade™

Abstraktion Eine Herausforderung der Umsetzung ist die Abstraktion der Variablenwerte zu geeigneten Äquivalenzklassen. Hier lässt sich für einige Systeme der Fallstudie die Abstraktion automatisiert durchführen, bei ca. der Hälfte der Systeme scheitert die Abstraktion daran, dass die Intervalle so klein gewählt werden müssten, so dass der Zustandsraum immer noch zu groß ist. Durch die Abstraktion wurde allerdings erst die Umsetzung der Verifikation für die verbleibenden Module ermöglicht.

Zeit Ein wesentlicher Faktor der Betrachtung von Fehlerzusammenhängen ist die Zeit. Sollen Zusammenhänge über mehrere Zeitschritte erfolgen, muss für jeden Zeitschritt der Variablenraum nochmals hinzugefügt werden. Entsprechend sind Anfragen über mehrere Zeitschritte nicht möglich. Das Verifikationswerkzeug Scade™ bietet Operatoren für maximal 5 Zeitschritte an. In der Fallstudie sind die Systeme selbst bereits sehr groß, so dass eine Verdopplung des Variablenraums die Leistungsgrenze der Verifikation schnell überschreitet. Etwa ein Drittel der Zusammenhänge bei Ausgaben ist im gleichen Zeittakt zu ermitteln. Alternativ muss für die Zusammenhänge die Abstraktion der Zeit in Abschnitt 5.2.2 berücksichtigt werden.

sichere Pfade Die Einschränkung der Verifikation auf kleinere Systeme führt in der Fallstudie dazu, dass nur Teile formal verifiziert werden. Im Wesentlichen wird die Sicherheitsarchitektur verifiziert, da hier die Strukturen gut abstrahierbar sind und die Systeme nicht zu groß sind. Die in Abschnitt 6.2.3.2 beschriebene Abstraktion von Modellteilen erlaubt für die Erstellung einer Ein-Ausgabe-FMEA an der Software-Schnittstelle so eine Unterstützung, in dem wenigstens 30% der Zusammenhänge vorgefertigt werden können.

Bewertung Die Verifikation mittels SAT-Solver ist für einen praxistauglichen Einsatz noch nicht geeignet. Probleme liegen hier im Wesentlichen in der Effektivität und Effizienz der Werkzeuge. Diese liefern zu häufig keine Ergebnisse, bzw. benötigen zu viel Rechenaufwand. Mit steigender Leistungsfähigkeit dieser Werkzeuge ist jedoch eine praxistaugliche Anwendung wahrscheinlicher.

Anhang D

Aquivalenzumformungen und Beweise

$$
\begin{aligned}
&([\![S]\!]\Delta[\![\overline{E}_S]\!] \Rightarrow ([\![S^a]\!] \vee [S^{\hat{a}}])\Delta[\![\mathcal{M}^a_S]\!]) \\
&\Leftrightarrow ([\![S]\!] \wedge [\![\overline{E}_S]\!] \Rightarrow [\![S^a]\!] \wedge [\![\overline{E}_a]\!] \vee [\![S^{\hat{a}}]\!] \wedge [\![\overline{E}_a]\!] \vee [\![F_a]\!]) \\
&\Leftrightarrow ([\![S^a]\!] \wedge [\![\overline{E}_S]\!] \Rightarrow ([\![S^a]\!]\Delta[\![\mathcal{M}^a]\!]) \vee [\![S^{\hat{a}}]\!] \wedge [\![\overline{E}_a]\!]) \qquad \text{da} \quad [\![S^a]\!] \Rightarrow [\![S]\!]) \\
&\Leftrightarrow ([\![S^a]\!]\Delta[\![\overline{E}_S]\!] \Rightarrow [\![S^a]\!]\Delta[\![\mathcal{M}^a]\!]) \qquad \text{da} \quad \neg([\![S^a]\!] \wedge [\![S^{\hat{a}}]\!])
\end{aligned}
\tag{D.1}
$$

$$
\begin{aligned}
&([\![S^a]\!] \vee [S^{\hat{a}}])\Delta[\![\mathcal{M}^a_S]\!] \Rightarrow [\![S]\!]\Delta[\![F_S]\!]) \\
&\Leftrightarrow ([\![S^a]\!] \vee [\![S^{\hat{a}}]\!]) \wedge [\![\overline{E}_a]\!] \vee [\![F_a]\!] \Rightarrow [\![S]\!] \vee [\![F_S]\!]) \\
&\Leftrightarrow ([\![S^a]\!] \wedge [\![\overline{E}_a]\!] \vee [\![S^{\hat{a}}]\!] \wedge [\![\overline{E}_a]\!] \vee [\![F_a]\!] \Rightarrow [\![S]\!] \vee [\![F_S]\!]) \\
&\Leftrightarrow ([\![S^a]\!] \wedge [\![\overline{E}_a]\!] \vee [\![F_a]\!] \Rightarrow [\![S]\!] \vee [\![F_S]\!]) \qquad \text{da} \quad [\![S^{\hat{a}}]\!] \Rightarrow [\![S]\!]) \\
&\Leftrightarrow ([\![S^a]\!]\Delta[\![\mathcal{M}^a_S]\!] \Rightarrow [\![S]\!]\Delta[\![F_S]\!])
\end{aligned}
\tag{D.2}
$$

$$
\begin{aligned}
[\![E^{(1,2)}_{o_j}]\!] &= (\Phi^{(1)}_{cond} \Rightarrow \neg\Phi_{o_j}) \wedge (\Phi^{(2)}_{cond} \Rightarrow \neg\Phi_{o_j}) \\
&= (\neg\Phi^{(1)}_{condition} \vee \neg\Phi_{o_j}) \wedge (\neg\Phi^{(2)}_{condition} \vee \neg\Phi_{o_j}) \\
&= (\neg\Phi^{(1)}_{condition} \vee \neg\Phi_{o_j}) \wedge \neg\Phi^{(2)}_{condition} \\
&\quad \vee (\neg\Phi^{(1)}_{condition} \vee \neg\Phi_{o_j}) \wedge \neg\Phi_{o_j} \\
&= \neg\Phi^{(1)}_{condition} \wedge \neg\Phi^{(2)}_{condition} \vee \neg\Phi_{o_j} \wedge \neg\Phi^{(2)}_{condition} \\
&\quad \vee \neg\Phi^{(1)}_{condition} \wedge \neg\Phi_{o_j} \vee \neg\Phi_{o_j} \wedge \neg\Phi_{o_j} \\
&= \neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \vee \neg\Phi_{o_j} \wedge \neg\Phi^{(2)}_{condition} \\
&\quad \vee \neg\Phi^{(1)}_{condition} \wedge \neg\Phi_{o_j} \vee \neg\Phi_{o_j} \\
&= \neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \vee \neg\Phi_{o_j} \\
&= (\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \neg\Phi_{o_j}
\end{aligned}
\tag{D.3}
$$

$$
\begin{aligned}
[[F_{o_j}^{(1,2)}]] &= (\Phi_{condition}^{(1)} \Rightarrow \Phi_{fail\_o_j}^{(1)}) \wedge (\neg\Phi_{condition}^{(1)} \Rightarrow \Phi_{o_j}) \wedge (\Phi_{condition}^{(2)} \Rightarrow \neg\Phi_{o_j}) \\
&\quad \vee (\Phi_{condition}^{(2)} \Rightarrow \Phi_{fail\_o_j}^{(2)}) \wedge (\neg\Phi_{condition}^{(2)} \Rightarrow \Phi_{o_j}) \wedge (\Phi_{condition}^{(1)} \Rightarrow \neg\Phi_{o_j}) \\
&= (\neg\Phi_{condition}^{(1)} \vee \Phi_{fail\_o_j}^{(1)}) \wedge (\Phi_{condition}^{(1)} \vee \Phi_{o_j}) \wedge (\neg\Phi_{condition}^{(2)} \vee \neg\Phi_{o_j}) \\
&\quad \vee (\neg\Phi_{condition}^{(2)} \vee \Phi_{fail\_o_j}^{(2)}) \wedge (\Phi_{condition}^{(2)} \vee \Phi_{o_j}) \wedge (\neg\Phi_{condition}^{(1)} \vee \neg\Phi_{o_j}) \\
&= (\neg\Phi_{condition}^{(1)} \vee \Phi_{fail\_o_j}^{(1)}) \\
&\quad \wedge ((\Phi_{condition}^{(1)} \vee \Phi_{o_j}) \wedge \neg\Phi_{condition}^{(2)} \vee (\Phi_{condition}^{(1)} \vee \Phi_{o_j}) \wedge \neg\Phi_{o_j}) \\
&\quad \vee (\neg\Phi_{condition}^{(2)} \vee \Phi_{fail\_o_j}^{(2)}) \\
&\quad \wedge ((\Phi_{condition}^{(2)} \vee \Phi_{o_j}) \wedge \neg\Phi_{condition}^{(1)} \vee (\Phi_{condition}^{(2)} \vee \Phi_{o_j}) \wedge \neg\Phi_{o_j}) \\
&= (\neg\Phi_{condition}^{(1)} \vee \Phi_{fail\_o_j}^{(1)}) \\
&\quad \wedge (\Phi_{condition}^{(1)} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{condition}^{(1)} \wedge \neg\Phi_{o_j}) \\
&\quad \vee (\neg\Phi_{condition}^{(2)} \vee \Phi_{fail\_o_j}^{(2)}) \\
&\quad \wedge (\Phi_{condition}^{(2)} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)} \wedge \neg\Phi_{o_j}) \\
&= \neg\Phi_{condition}^{(1)} \wedge (\Phi_{condition}^{(1)} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{condition}^{(1)} \wedge \neg\Phi_{o_j}) \\
&\quad \vee \Phi_{fail\_o_j}^{(1)} \wedge (\Phi_{condition}^{(1)} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{condition}^{(1)} \wedge \neg\Phi_{o_j}) \\
&\quad \vee \neg\Phi_{condition}^{(2)} \wedge (\Phi_{condition}^{(2)} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)} \wedge \neg\Phi_{o_j}) \\
&\quad \vee \Phi_{fail\_o_j}^{(2)} \wedge (\Phi_{condition}^{(2)} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{o_j} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)} \wedge \neg\Phi_{o_j}) \\
&= \neg\Phi_{condition}^{(1)} \wedge \Phi_{o_j} \wedge \neg\Phi_{condition}^{(2)} \\
&\quad \vee \Phi_{fail\_o_j}^{(1)} \wedge \Phi_{condition}^{(1)} \wedge \neg\Phi_{condition}^{(2)} \vee \Phi_{fail\_o_j}^{(1)} \wedge \Phi_{condition}^{(1)} \wedge \neg\Phi_{o_j} \\
&\quad \vee \neg\Phi_{condition}^{(2)} \wedge \Phi_{o_j} \wedge \neg\Phi_{condition}^{(1)} \\
&\quad \vee \Phi_{fail\_o_j}^{(2)} \wedge \Phi_{condition}^{(2)} \wedge \neg\Phi_{condition}^{(1)} \vee \Phi_{fail\_o_j}^{(2)} \wedge \Phi_{condition}^{(2)} \wedge \neg\Phi_{o_j} \\
&= \neg(\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)}) \wedge \Phi_{o_j} \\
&\quad \vee \Phi_{fail\_o_j}^{(1)} \wedge \Phi_{condition}^{(1)} \\
&\quad \vee \Phi_{fail\_o_j}^{(2)} \wedge \Phi_{condition}^{(2)} \\
&= \neg(\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)}) \wedge \Phi_{o_j} \\
&\quad \vee \Phi_{fail\_o_j}^{(1)} \wedge \Phi_{condition}^{(1)} \wedge (\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)}) \\
&\quad \vee \Phi_{fail\_o_j}^{(2)} \wedge \Phi_{condition}^{(2)} \wedge (\Phi_{condition}^{(1)} \vee \Phi_{condition}^{(2)})
\end{aligned}
\tag{D.4}
$$

$$
\begin{aligned}
&= \neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \wedge \Phi_{o_j} \\
&\quad \vee (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition}) \wedge (\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \\
&= ((\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \vee \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \vee \neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition})) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge ((\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition}))
\end{aligned}
$$

$$
\begin{aligned}
[[F^{(1,2)}_{o_j}]] &= (\Phi^{(1)}_{condition} \Rightarrow \Phi^{(1)}_{fail\_o_j}) \wedge (\neg\Phi^{(1)}_{condition} \Rightarrow \Phi_{o_j}) \wedge (\Phi^{(2)}_{condition} \Rightarrow \neg\Phi_{o_j}) \\
&\quad \vee (\Phi^{(2)}_{condition} \Rightarrow \Phi^{(2)}_{fail\_o_j}) \wedge (\neg\Phi^{(2)}_{condition} \Rightarrow \Phi_{o_j}) \wedge (\Phi^{(1)}_{condition} \Rightarrow \neg\Phi_{o_j}) \\
&= ((\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \vee \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \vee \neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition})) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \wedge \neg\Phi^{(1)}_{condition} \vee \neg\Phi^{(1)}_{condition} \\
&\qquad \wedge \neg\Phi^{(2)}_{condition}) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{fail\_o_j} \wedge \Phi^{(1)}_{condition} \vee \neg\Phi^{(1)}_{condition} \wedge (\Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \vee \neg\Phi^{(2)}_{condition})) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{condition} \Rightarrow \Phi^{(1)}_{fail\_o_j}) \wedge (\Phi^{(1)}_{condition} \vee (\Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \vee \neg\Phi^{(2)}_{condition})) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{condition} \Rightarrow \Phi^{(1)}_{fail\_o_j}) \wedge (\Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition} \vee \neg\Phi^{(2)}_{condition}) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{condition} \Rightarrow \Phi^{(1)}_{fail\_o_j}) \wedge (\Phi^{(2)}_{condition} \Rightarrow (\Phi^{(2)}_{fail\_o_j} \wedge \Phi^{(2)}_{condition})) \\
&= (\neg(\Phi^{(1)}_{condition} \vee \Phi^{(2)}_{condition}) \Rightarrow \Phi_{o_j}) \\
&\quad \wedge (\Phi^{(1)}_{condition} \Rightarrow \Phi^{(1)}_{fail\_o_j}) \wedge (\Phi^{(2)}_{condition} \Rightarrow (\Phi^{(2)}_{fail\_o_j})
\end{aligned}
\tag{D.5}
$$

$$
\begin{aligned}
[\![S]\!]\Delta^{(\Phi_a,\Phi_b)}(\mathcal{M}_S^{(1)},\mathcal{M}_S^{(2)}) &= ([\![S]\!]\Delta^{\Phi_a}\mathcal{M}_S^{(1)})\Delta^{\Phi_b}\mathcal{M}_S^{(2)} \\
&= ((\Phi_a \wedge \Phi_b \wedge \Phi_c)\Delta^{\Phi_a}\mathcal{M}_S^{(1)})\Delta^{\Phi_b}\mathcal{M}_S^{(2)} \\
&= ((\Phi_a \wedge [\![E_S^1]\!] \vee [\![F_S^1]\!]) \wedge \Phi_b \wedge \Phi_c)\Delta^{\Phi_b}\mathcal{M}_S^{(2)} \\
&= (\Phi_a \wedge [\![E_S^1]\!] \vee [\![F_S^1]\!]) \wedge (\Phi_b \wedge [\![E_S^2]\!] \vee [\![F_S^2]\!]) \wedge \Phi_c \\
&= (\Phi_a \wedge [\![E_S^1]\!] \wedge \Phi_b \wedge [\![E_S^2]\!] \\
&\quad \vee \Phi_b \wedge [\![E_S^2]\!] \wedge [\![F_S^1]\!] \\
&\quad \vee \Phi_a \wedge [\![E_S^1]\!] \wedge [\![F_S^2]\!] \vee [\![F_S^1]\!] \wedge [\![F_S^2]\!]) \wedge \Phi_c \\
&= [\![S]\!]\Delta^{\Phi_a\wedge\Phi_b}([\![E_S^1]\!] \wedge [\![E_S^2]\!], \\
&\qquad \Phi_b \wedge [\![E_S^2]\!] \wedge [\![F_S^1]\!] \vee \Phi_a \wedge [\![E_S^1]\!] \wedge [\![F_S^2]\!] \vee [\![F_S^1]\!] \wedge [\![F_S^2]\!]) \\
&= [\![S]\!]\Delta^{\Phi_a\wedge\Phi_b}(\mathcal{M}_S^{(1)} +^{\Phi_1,\Phi_2} \mathcal{M}_S^{(2)})
\end{aligned}
\tag{D.6}
$$

$$
\begin{aligned}
[\![\mathcal{M}_S^{(1)}]\!]^{op} + [\![\mathcal{M}_S^{(2)}]\!]^{op} &\backsimeq [\![\mathcal{M}_S^{(1)}]\!] + [\![\mathcal{M}_S^{(2)}]\!] \\
&\backsimeq ([\![\overline{E}_S^{(1)}]\!]^{op}[\uparrow \mathbb{Z}_0^+ \uparrow], ([\![S]\!]^{op}\Delta[\![\mathcal{M}_S^{(1)}]\!]^{op})[\uparrow \mathbb{Z}_0^+ \uparrow]) \\
&\quad + ([\![\overline{E}_S^{(2)}]\!]^{op}[\uparrow \mathbb{Z}_0^+ \uparrow], ([\![S]\!]^{op}\Delta[\![\mathcal{M}_S^{(2)}]\!]^{op})[\uparrow \mathbb{Z}_0^+ \uparrow]) \\
&\backsimeq ((\overline{[\![E_S^{(1)}]\!]}^{op} \wedge \overline{[\![E_S^{(2)}]\!]}^{op})[\uparrow \mathbb{Z}_0^+ \uparrow], [\![F_S^{(1,2)}]\!])
\end{aligned}
\tag{D.7}
$$

$$
\begin{aligned}
[\![F_S^{(1,2)}]\!] &= ([\![S]\!]^{op}\Delta[\![\mathcal{M}_S^{(1)}]\!]^{op} \wedge \overline{[\![E_S^{(2)}]\!]}^{op} \wedge (l=1))[\uparrow \mathbb{Z}_0^+ \uparrow] \\
&\quad \vee ([\![S]\!]^{op}\Delta[\![\mathcal{M}_S^{(2)}]\!]^{op} \wedge (l=2))[\uparrow \mathbb{Z}_0^+ \uparrow] \\
&\stackrel{?}{=} (\neg(\mathsf{delay}_{\langle 0\rangle}(l)=2) \wedge (l=1) \wedge [[S]]^{op}\Delta[\![\mathcal{M}_S^{(1)}]\!]^{op} \wedge \overline{[\![E_S^{(2)}]\!]}^{op} \\
&\qquad \vee \neg(\mathsf{delay}_{\langle 0\rangle}(l)=1) \wedge (l=2) \wedge [[S]]^{op}\Delta[\![\mathcal{M}_S^{(2)}]\!]^{op})[\uparrow \mathbb{Z}_0^+ \uparrow]
\end{aligned}
$$

Induktionsbeweis Erschließen von $\mathbb{Z}_0^+$mit Teilmengen$\{0,..n\}$

Hilfsformeln: $\Phi_{l1} = ((l=1) \wedge [[S]]^{op}\Delta[\![\mathcal{M}_S^{(1)}]\!]^{op} \wedge \overline{[\![E_S^{(2)}]\!]}^{op})$
$\Phi_{l2} = ((l=2) \wedge [[S]]^{op}\Delta[\![\mathcal{M}_S^{(2)}]\!]^{op})$

Induktionsanker: $n=0$

$$
\begin{aligned}
[\![F_S^{(1,2)}]\!] &= \Phi_{l1}[\uparrow \{0\} \uparrow] \vee \Phi_{l2}[\uparrow \{0\} \uparrow] \\
&= (\Phi_{l1} \vee \Phi_{l2})[\uparrow \{0\} \uparrow] \\
&= (\neg(\mathsf{delay}_{\langle 0\rangle}(l) \neq 2) \wedge \Phi_{l1} \vee \neg(\mathsf{delay}_{\langle 0\rangle}(l) \neq 1) \wedge \Phi_{l2})[\uparrow \{0\} \uparrow]
\end{aligned}
$$

Induktionsschluss: $n > 0$

$$
\begin{aligned}
[\![F_S^{(1,2)}]\!] &= \Phi_{l1}[\uparrow \{0..(n+1)\} \uparrow] \vee \Phi_{l2}[\uparrow \{0..(n+1)\} \uparrow] \\
&= \Phi_{l1}[\uparrow \{0..n\} \uparrow] \wedge \Phi_{l1}[\uparrow \{n+1\} \uparrow] \vee \Phi_{l2}[\uparrow \{0..n\} \uparrow] \wedge \Phi_{l2}[\uparrow \{n+1\} \uparrow] \\
&= \Phi_{l1}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1})[\uparrow \{n+1\} \uparrow] \\
&\quad \vee \Phi_{l2}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{n+1\} \uparrow] \\
&= \Phi_{l1}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1})[\uparrow \{n+1\} \uparrow] \\
&\quad \vee \Phi_{l1}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{n+1\} \uparrow] \\
&\quad \vee \Phi_{l2}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1})[\uparrow \{n+1\} \uparrow] \\
&\quad \vee \Phi_{l2}[\uparrow \{0..n\} \uparrow] \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{n+1\} \uparrow] \\
&= (\Phi_{l1}[\uparrow \{0..n\} \uparrow] \vee \Phi_{l2}[\uparrow \{0..n\} \uparrow]) \\
&\quad \wedge ((\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1})[\uparrow \{n+1\} \uparrow] \\
&\qquad \vee (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{n+1\} \uparrow]) \\
&= (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1} \vee \neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{0..n\} \uparrow] \\
&\quad \wedge (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1} \vee \neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{n+1\} \uparrow] \\
&= (\neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 2) \wedge \Phi_{l1} \vee \neg(\mathsf{delay}_{\langle 0 \rangle}(l) = 1) \wedge \Phi_{l2})[\uparrow \{0..(n+1)\} \uparrow]
\end{aligned}
$$

Anhang E

Umsetzung der Abstraktionsmechanismen

E.1 Abstraktion der Komponenten

Summe

Realisierung der Summe Sum:

$Sum : W_{a_r} \times W_{b_r} \rightarrow W_{c_r}$
$I_{Sum} = \{a_r, b_r\}$
$O_{Sum} = \{c_r\}$
$[\![Sum]\!]^{op} = (c_r = a_r + b_r)$

intervalabstrakte Summe (Wert-Äquivalenzklassen) Sum_a mit Abstraktionsrelationen R_a, R_b, R_c:

$Sum_a : W_{a_a} \times W_{b_a} \rightarrow W_{c_a}$
$I_{Sum_a} = \{a_a, b_a\}$
$O_{Sum_a} = \{c_a\}$

$$\begin{aligned}[\![Sum_a]\!]^{op} &= ([\![Sum]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op} \wedge [\![R_c]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3}\\ &= (\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) + \mathsf{min}(\mathsf{ABS}_b(b_a)) > \mathsf{max}(\mathsf{ABS}_c(c_a)))) && \text{| Linearität}\\ &\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) + \mathsf{max}(\mathsf{ABS}_b(b_a)) < \mathsf{min}(\mathsf{ABS}_c(c_a))))\end{aligned}$$

intervalabstrakte Summe (Wert-Äquivalenzklassen) deterministisch Sum^{eq_det}:

$Sum_d : \mathcal{P}(W_{a_d}) \times \mathcal{P}(W_{b_d}) \rightarrow \mathcal{P}(W_{c_d})$
$I_{Sum_d} = \{a_d, b_d, \}$
$O_{Sum_d} = \{c_d\}$

$$[\![Sum_d]\!]^{op} = \boldsymbol{c_d = \{x \mid \exists a_a, b_a, c_a \in I_{Sum^{equiv}} \cup O_{Sum^{equiv}} : [\![Sum^{equiv}]\!]^{op} \wedge a_a \in a_d \wedge b_a \in b_d \wedge c_a \in c_d\}}$$

Maximum

Realisierung des Maximum $Maximum$:

$Maximum : W_{a_r} \times W_{b_r} \rightarrow W_{c_r}$
$I_{Maximum} = \{a_r, b_r\}$
$O_{Maximum} = \{c_r\}$
$[\![Maximum]\!]^{op} = (c_r = \mathsf{max}(a_r, b_r))$

intervalabstrakte Summe (Wert-Äquivalenzklassen) $Maximum_a$ mit Abstraktionsrelationen R_a, R_b, R_c:

$Maximum_a : W_{a_a} \times W_{b_a} \rightarrow W_{c_a}$
$I_{Maximum_a} = \{a_a, b_a\}$
$O_{Maximum_a} = \{c_a\}$

$$\begin{aligned}[Maximum_a]^{op} &= ([\![Maximum]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op} \wedge [\![R_c]\!]^{op}) \\ &= (\neg(\mathsf{max}(\mathsf{min}(\mathsf{ABS}_a(a_a)), \mathsf{min}(\mathsf{ABS}_b(b_a))) > \mathsf{max}(\mathsf{ABS}_c(c_a)))) \\ &\quad \wedge(\neg(\mathsf{max}(\mathsf{max}(\mathsf{ABS}_a(a_a)) + \mathsf{max}(\mathsf{ABS}_b(b_a))) < \mathsf{min}(\mathsf{ABS}_c(c_a))))\end{aligned}$$

relationaler Operator ==

Realisierung des relationalen Operators $RelOp^{==}$:

$RelOp^{==} : W_{a_r} \times W_{b_r} \rightarrow BOOL$
$I_{RelOp^{==}} = \{a_r, b_r\}$
$O_{RelOp^{==}} = \{c_r\}$
$[\![RelOp^{==}]\!]^{op} = (c_r = (a_r == b_r))$

intervalabstrakter relationaler Operator $RelOp_a^{==}$ mit Abstraktionsrelationen R_a, R_b:

$RelOp_a^{==} : W_{a_a} \times W_{b_a} \rightarrow BOOL$
$I_{RelOp_a^{==}} = \{a_a, b_a\}$
$O_{RelOp_a^{==}} = \{c_a\}$

$$\begin{aligned}[\![RelOp_a^{==}]\!]^{op} &= ([\![RelOp^{==}]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\ &= (\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > \mathsf{max}(\mathsf{ABS}_b(b_a)) \wedge c_a)) && \text{| Linearität} \\ &\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) < \mathsf{min}(\mathsf{ABS}_b(b_a)) \wedge c_a)) \\ &\quad \wedge(\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) \\ &\qquad == \mathsf{min}(\mathsf{ABS}_b(b_a)) \\ &\qquad == \mathsf{max}(\mathsf{ABS}_a(a_a)) \\ &\qquad == \mathsf{max}(\mathsf{ABS}_b(b_a)) \\ &\qquad \wedge\neg c_a))\end{aligned}$$

relationaler Operator $>$

Realisierung des relationalen Operators $RelOp^{>}$:

$RelOp^{>} : W_{a_r} \times W_{b_r} \to BOOL$
$I_{RelOp^{>}} = \{a_r, b_r\}$
$O_{RelOp^{>}} = \{c_r\}$
$[\![RelOp^{>}]\!]^{op} = (c_r = (a_r > b_r))$

intervalabstrakter relationaler Operator $RelOp_a^{>}$ mit Abstraktionsrelationen R_a, R_b:

$RelOp_a^{>} : W_{a_a} \times W_{b_a} \to BOOL$
$I_{RelOp_a^{>}} = \{a_a, b_a\}$
$O_{RelOp_a^{>}} = \{c_a\}$

$$\begin{aligned}
[\![RelOp_a^{>}]\!]^{op} &= ([\![RelOp^{>}]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > \mathsf{max}(\mathsf{ABS}_b(b_a)) \wedge \neg c_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) \leq \mathsf{min}(\mathsf{ABS}_b(b_a)) \wedge c_a))
\end{aligned}$$

relationaler Operator $\neq$

Realisierung des relationalen Operators $RelOp^{\neq}$:

$RelOp^{\neq} : W_{a_r} \times W_{b_r} \to BOOL$
$I_{RelOp^{\neq}} = \{a_r, b_r\}$
$O_{RelOp^{\neq}} = \{c_r\}$
$[\![RelOp^{\neq}]\!]^{op} = (c_r = (a_r \neq b_r))$

intervalabstrakter relationaler Operator $RelOp_a^{\neq}$ mit Abstraktionsrelationen R_a, R_b:

$RelOp_a^{\neq} : W_{a_a} \times W_{b_a} \to BOOL$
$I_{RelOp_a^{\neq}} = \{a_a, b_a\}$
$O_{RelOp_a^{\neq}} = \{c_a\}$

$$\begin{aligned}
[\![RelOp_a^{\neq}]\!]^{op} &= ([\![RelOp^{\neq}]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > \mathsf{max}(\mathsf{ABS}_b(b_a)) \wedge \neg c_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) < \mathsf{min}(\mathsf{ABS}_b(b_a)) \wedge \neg c_a)) \\
&\quad \wedge\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) \\
&\quad\ == \mathsf{min}(\mathsf{ABS}_b(b_a)) \\
&\quad\ == \mathsf{max}(\mathsf{ABS}_a(a_a)) \\
&\quad\ == \mathsf{max}(\mathsf{ABS}_b(b_a)) \\
&\quad\ \wedge c_a)
\end{aligned}$$

Set-Reset-Flip-Flop

Realisierung des Set-Reset-Flip-Flops $SRFF$:

$SRFF : W_{a_r} \times W_{b_r} \to BOOL \times BOOL$
$I_{SRFF} = \{a_r, b_r\}$
$O_{SRFF} = \{c_r, d_r\}$
$[\![SRFF]\!]^{op} = (c_r = ((b_r \leq 0) \wedge (\mathsf{delay}_{\langle false \rangle}(c_r) \vee (a_r > 0))) \wedge (d_r = \neg c_r))$

intervalabstrakter Set-Reset-Flip-Flops $SRFF_a$ mit Abstraktionsrelationen R_a, R_b:

$SRFF_a : W_{a_a} \times W_{b_a} \to BOOL \times BOOL$
$I_{SRFF_a} = \{a_a, b_a\}$
$O_{SRFF_a} = \{c_a, d_r\}$

$$\begin{aligned}
[\![SRFF_a]\!]^{op} &= ([\![SRFF]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{min}(\mathsf{ABS}_a(b_a)) > 0 \wedge c_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(b_a)) \leq 0 \wedge \mathsf{delay}_{\langle false \rangle}(c_a) \wedge \neg c_a)) \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(b_a)) \leq 0 \wedge (\mathsf{min}(\mathsf{ABS}_a(a_a)) > 0 \wedge \neg c_a)) \\
&\quad \wedge(d_a = \neg c_a)
\end{aligned}$$

logischer Operator lOp^{OR}

Realisierung des logischen Operators lOp^{OR}:

$lOp^{OR} : W_{a_r} \times W_{b_r} \to BOOL$
$I_{lOp^{OR}} = \{a_r, b_r\}$
$O_{lOp^{OR}} = \{c_r\}$
$[\![lOp^{OR}]\!]^{op} = (c_r = (a_r > 0) \vee (b_r > 0))$

intervalabstrakter logischer Operator lOp_a^{OR} mit Abstraktionsrelationen R_a, R_b:

$lOp_a^{OR} : W_{a_a} \times W_{b_a} \to BOOL$
$I_{lOp_a^{OR}} = \{a_a, b_a\}$
$O_{lOp_a^{OR}} = \{c_a\}$

$$\begin{aligned}
[\![lOp_a^{OR}]\!]^{op} &= ([\![lOp^{OR}]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > 0 \wedge \neg c_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{min}(\mathsf{ABS}_a(b_a)) > 0 \wedge \neg c_a)) \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) \leq 0 \wedge \mathsf{max}(\mathsf{ABS}_a(b_a)) \leq 0 \wedge c_a))
\end{aligned}$$

logischer Operator lOp^{AND}

Realisierung des logischen Operators lOp^{AND}:

$lOp^{AND} : W_{a_r} \times W_{b_r} \rightarrow BOOL$
$I_{lOp^{AND}} = \{a_r, b_r\}$
$O_{lOp^{AND}} = \{c_r\}$
$[\![lOp^{AND}]\!]^{op} = (c_r = (a_r > 0) \vee (b_r > 0))$

intervalabstrakter logischer Operator lOp_a^{AND} mit Abstraktionsrelationen R_a, R_b:

$lOp_a^{AND} : W_{a_a} \times W_{b_a} \rightarrow BOOL$
$I_{lOp_a^{AND}} = \{a_a, b_a\}$
$O_{lOp_a^{AND}} = \{c_a\}$

$$\begin{aligned}
[\![lOp_a^{AND}]\!]^{op} &= ([\![lOp^{AND}]\!]^{op} \wedge [\![R_a]\!]^{op} \wedge [\![R_b]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) \leq 0 \wedge c_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{max}(\mathsf{ABS}_a(b_a)) \leq 0 \wedge c_a)) \\
&\quad \wedge(\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > 0 \wedge \mathsf{min}(\mathsf{ABS}_a(b_a)) > 0 \wedge c_a))
\end{aligned}$$

logischer Operator lOp^{NOT}

Realisierung des logischen Operators lOp^{NOT}:

$lOp^{NOT} : W_{a_r} \rightarrow BOOL$
$I_{lOp^{NOT}} = \{a_r\}$
$O_{lOp^{NOT}} = \{b_r\}$
$[\![lOp^{NOT}]\!]^{op} = (b_r = \neg(a_r > 0))$

intervalabstrakter logischer Operator lOp_a^{NOT} mit Abstraktionsrelationen R_a:

$lOp_a^{NOT} : W_{a_a} \rightarrow BOOL$
$I_{lOp_a^{NOT}} = \{a_a\}$
$O_{lOp_a^{NOT}} = \{b_a\}$

$$\begin{aligned}
[\![lOp_a^{NOT}]\!]^{op} &= ([\![lOp^{NOT}]\!]^{op} \wedge [\![R_a]\!]^{op}) && \text{| Abstraktion aus Def. 6.1.3} \\
&= (\neg(\mathsf{max}(\mathsf{ABS}_a(a_a)) \leq 0 \wedge \neg b_a)) && \text{| Linearität} \\
&\quad \wedge(\neg(\mathsf{min}(\mathsf{ABS}_a(a_a)) > 0 \wedge b_a))
\end{aligned}$$

E.2 Modifikationsabhängige Abstraktionen

Im Folgenden werden Konsequenzen bestimmter Modifikationsarten auf die Abstraktion gelistet:

Schwellwertüberschreitung Für die Intervallabstraktion müssen hier mindestens die Ober- und die Untergrenze die Intervalle teilen.

Gradientenüberschreitung Hinsichtlich der Intervallabstraktion muss hier die Unterteilung so feingranular sein, dass innerhalb der zu bewertenden Zeit diese Deltas erkennbar sind. Ist die zu bewertende Zeit ein Rechenschritt (z.B $10ms$), so sind die Intervalle meist sehr klein und es findet eine entsprechende Zustandsexplosion statt. In der Fallstudie hat sich gezeigt, dass häufig Aufgrund der Physik die Zeitspanne erhöht werden kann oder die Intervalle größer gewählt werden können, da nur generell die Wirkung einer Überschreitung betrachtet wird. In anderen Fällen kann eine lineare Abschätzung herangezogen werden.

absolute Wertabweichung Für die Intervallabstraktion ist entscheidend, welche Stärke der Abweichung für den Fehler relevant ist. Entsprechend dieser Stärke wird das Signal in ϵ-Intervalle unterteilt. Auch hier kann eventuell die Zustandsexplosion eingeschränkt werden, in dem die Intervalle spezifisch größer gewählt werden können. (z.B. Verwendung der Safety-Spezifikation: 0.25rad statt Komfort-Spezifikation: 0.03rad)

relative Wertabweichung Hinsichtlich der Intervallabstraktion ist entscheidend, welche Stärke der Abweichung für den Fehler relevant ist. Das Signal wird hier nicht in ϵ-Intervalle unterteilt, sondern mir steigendem Betrag des Sollwertes werden auch die Intervalle entsprechend der relativen Abweichung größer. An der unteren Grenze der relativen Angabe wird dann wie bei der absoluten Wertabweichung vorgegangen.

konstanter Wert Hinsichtlich der Intervallabstraktion ist hier lediglich der Wert der Konstante bzw. der konstante Wertebereich relevant. Es reichen hier zwei Werte aus.

Modusfehler Die Intervallabstraktion entspricht im Wesentlichen dem Abstand der Signalwerte der Modi und den beschriebenen Abweichungen.

Anhang F

Methodischer Blickwinkel

Dieser Abschnitt gibt einen groben Überblick über die Struktur der Arbeit aus Sicht des methodischen Vorgehens. Es wird ein Vorgehen vorgeschlagen, um die automatisierte Analyse eines Systems durchzuführen. Zu den Aktivitäten des Vorgehens wird auf die relevanten Abschnitte der Arbeit verwiesen.

Aktivitätsfolge

Prozessmodelle formulieren häufig eine strikte Reihenfolge von Aktivitäten oder zumindest eine kausale Abhängigkeit zwischen Aktivitäten. Auf hohen Beschreibungsebenen werden die Prozesse häufig mit Techniken modelliert, die auf Petri-Netzen ([Pet62, Rei85]) beruhen. In dieser Arbeit wird auf oberen Ebenen das Petri-Netz-Derivat der Unified Modelling Language ([Fow03]), nämlich Aktivitätsdiagramme, verwendet. Allgemein sieht der Prozess der prototypischen Anwendung wie in Abbildung F.1 aus. Das Vorgehen entspricht dem Kreislauf, der in den Kapiteln 1 und 2 vorgestellt wurde. Genauere Erläuterungen finden im weiteren Verlauf des Abschnitts statt.

kausale Ordnung

Die direkte Modellierung der kausalen Abhängigkeiten macht allerdings nur begrenzt Sinn. Obwohl die Aktivitäten in der Realität kausal voneinander abhängig sind, kann eine explizite Modellierung unnötig komplex sein. Man kann davon ausgehen, dass die Rollen, welche die Aktivitäten ausführen, wissen, wann eine Aktivität Sinn macht. Besonders in der Funktionssicherheitsanalyse ist die Anwendbarkeit einer Aktivität von vielen Faktoren abhängig und eine eindeutige Ausführungsreihenfolge nur schwer festzulegen. Die Aktivitätsdiagramme, wie sie in Abbildung F.1 zu sehen sind, machen entsprechend nur Vorschläge zu einer Ausführungsreihenfolge.

Übersicht

Im Folgenden wird der Ablauf des Vorgehens dargestellt. Dabei wird bei den Schritten zurück in die Kapitel der Arbeit verwiesen. Dem Leser steht so eine vorgehensbasierte Referenz dieser Arbeit zur Verfügung.

Methode

Das Vorgehen auf oberster Ebene ist in Abbildung F.1 dargestellt. Der Ablauf des Vorgehens ist für informelle Artefakte in Kapitel 2 beschrieben. Die Schritte sind folgende: Die Entwicklung erstellt modellbasiert in der Aktivität *Systemdefinition und -entwicklung* ein System. Dabei werden auch Spezifikationen im Sinne von Anforderungen und Modellen erstellt. Diese Artefakte der Entwicklung werden bei

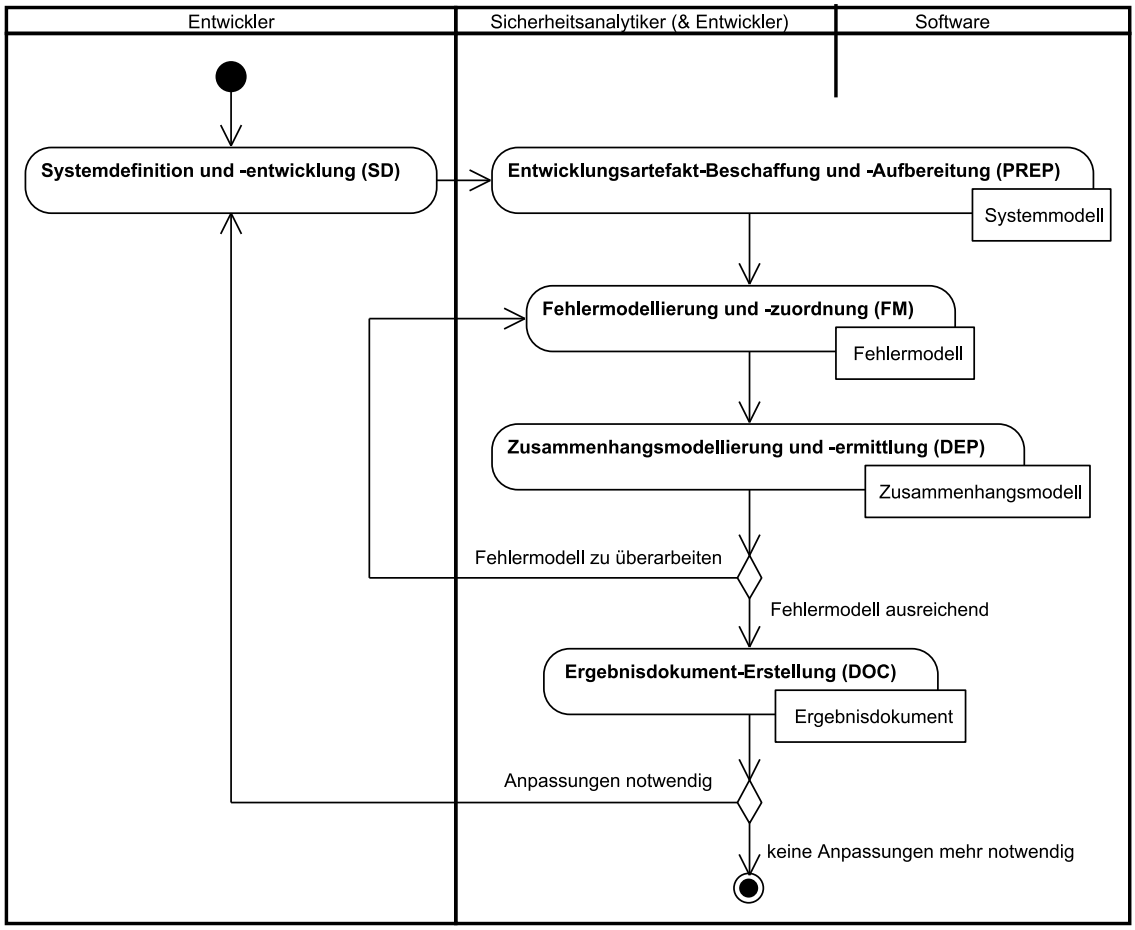


Abb. F.1: Der Prozess auf oberer Ebene

der Aktivität *Beschaffung und Aufbereitung* zusammengestellt und so aufbereitet, dass sie für die teilautomatisierte Funktionssicherheitsanalyse verwertbar sind. Eine genauere Beschreibung der Aktivitäten ist in Abbildung F.2 zu finden. Kapitel 3 widmet sich den formalen Artefakten, welche Ergebnis dieser Aktivitäten sind. Im Anschluss an die Aufbereitung der Entwicklungsartefakte werden mit der Aktivität *Fehlermodellierung und -zuordnung* für die Artefakte spezifische Fehler modelliert und ihnen zugewiesen. Somit erhält man ein Systemmodell, welches um Fehler angereichert ist. Die Modelle und Modellierungstechniken für Fehler sind in Kapitel 4 erklärt. Die Zusammenhänge zwischen den vorhandenen Artefakten der Entwicklung und zwischen den Fehlern werden mit der Aktivität *Zusammenhangsmodellierung und -ermittlung* festgehalten. Die hier verwendeten Modelle und Techniken sind in Kapitel 5 zu finden. Ergibt sich aus der Zusammenhangsanalyse, dass die Fehlermodelle angepasst werden müssen, so können die Modelle iterativ angepasst und vervollständigt werden. Das Ergebnis ist ein Modell, welches die Informationen enthält, um ein Fehlernetz zu erstellen. Diese Informationen werden mit der Aktivität *Ergebnisdokument-Erstellung* in eine für die Entwickler und Analysten

geeignete Darstellung gebracht. Diese bereits verwendete Darstellung ist in Kapitel 2 bei der Vorstellung des in der Industrie vorgefundenen informellen Vorgehens zu finden. Alle Aktivitäten der Abbildung F.1 werden teilweise automatisiert. Die Automatisierung wird in Kapitel 6 behandelt.

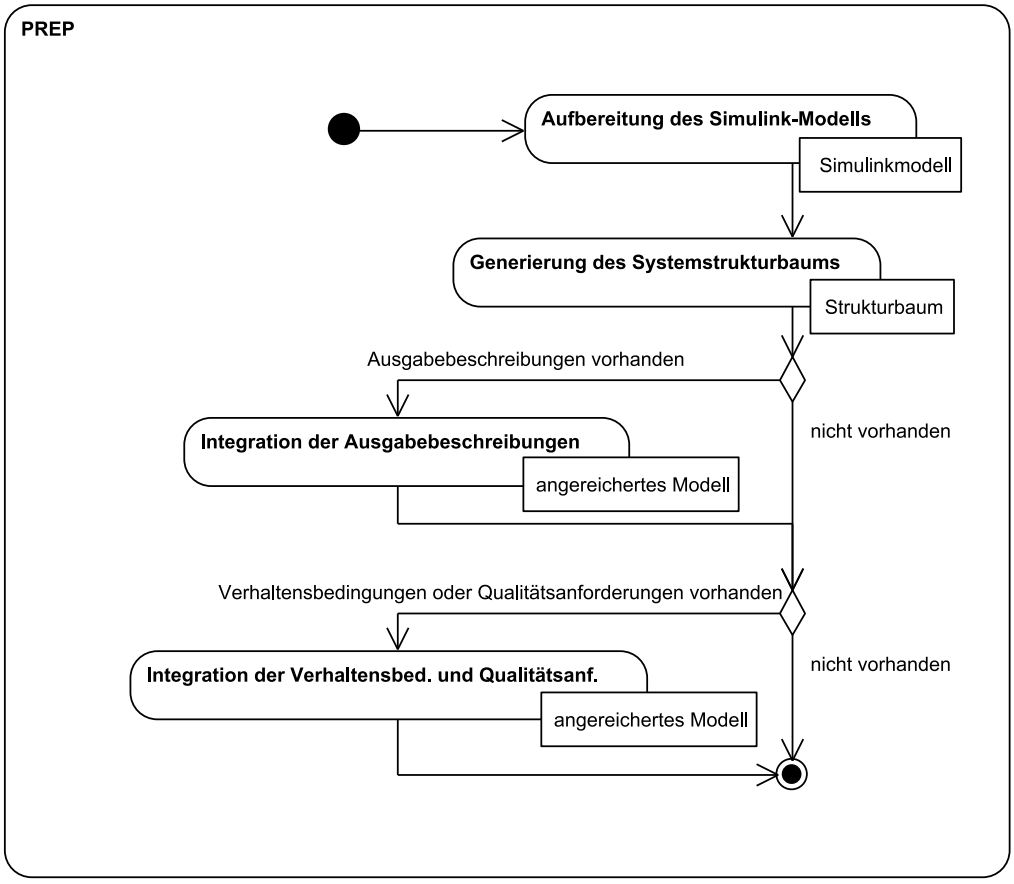


Abb. F.2: Beschaffung und Aufbereitung der Entwicklungsartefakte

Aufbereitung Die Beschaffung und Aufbereitung der Entwicklungsartefakte ist in Abbildung F.2 dargestellt. Diese Artefakte werden in den Kapiteln 2 und 3 vorgestellt. Die Beschaffung und Aufbereitung besteht im Wesentlichen aus vier Teilen. Grundlage für die Sicherheitsanalyse ist ein Simulink-Modell des Systems. Dieses kann, abhängig von den Entwicklungsständen, noch inkonsistent und unvollständig sein. Für die Analysen sind die Modelle entsprechend aufzubereiten. Das grundlegende Artefakt für die Sicherheitsanalyse ist der Systemstrukturbaum. Dieser wird mit der Aktivität *Generierung des Systemstrukturbaums* erstellt. Weiter werden dann Informationen zu den Ausgaben der Elemente des Baumes gesammelt und vorhandene Verhaltensbedingungen und Qualitätsanforderungen zugeordnet. Der Aufbau eines Systemstrukturbaums und die Zuordnung von Anforderungen ist in Abschnitt 2.3 gezeigt. Eine Umsetzung der Generierung ist in Anhang A zu finden.

Fehlermodellierung Die Fehlermodellierung und -zuordnung ist in Abbildung F.3 dargestellt. Sie wird in der Arbeit in Kapitel 4 beschrieben. Sie besteht aus einer Menge von Aktivitäten, die

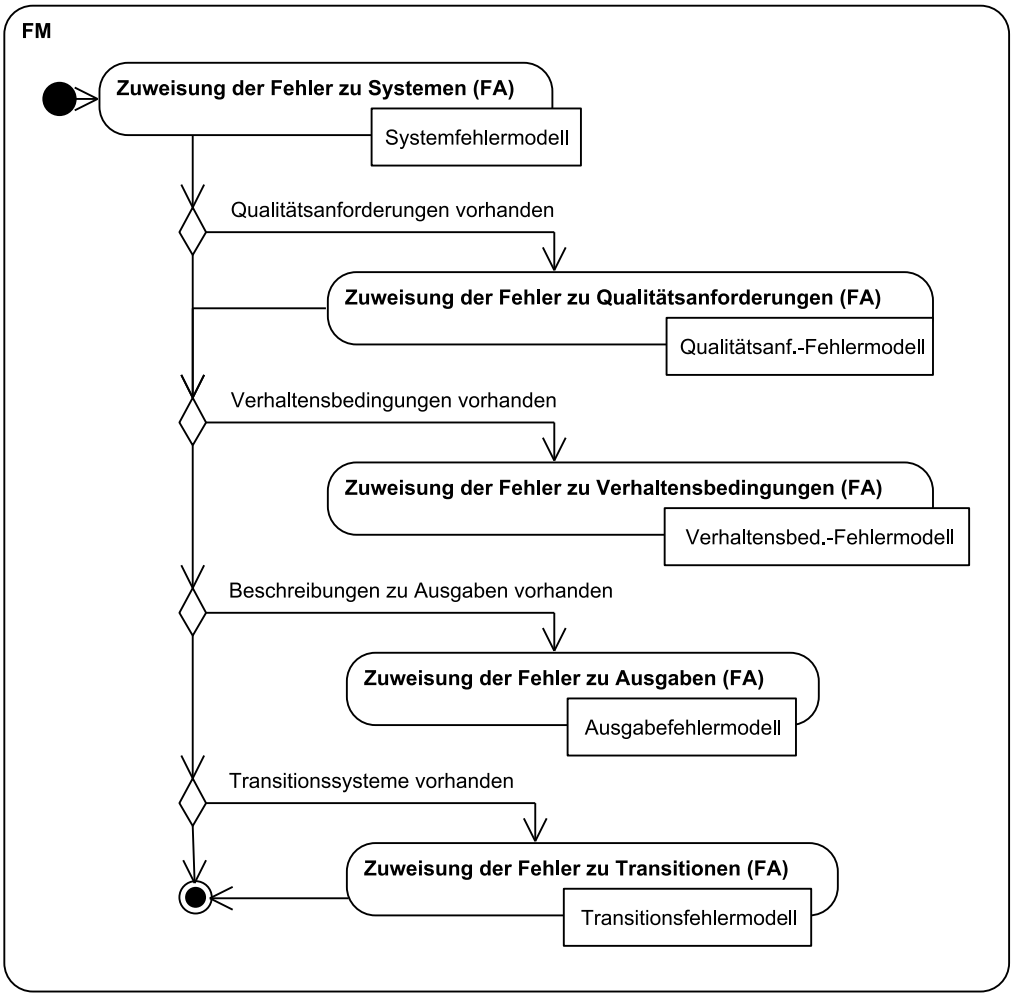


Abb. F.3: Fehlermodellierung und -zuweisung

auf die spezifischen Modellierungstechniken des Systemmodells zugeschnitten sind. Diese Techniken sind Systemmodelle als Ganzes, Qualitätsanforderungen, Verhaltensbedingungen, Beschreibungen zu Ausgaben und Transitionssysteme. Entsprechend dieser Techniken sind auch jeweils die Kapitel 3, 4 und 5 eingeteilt. Jede der Aktivitäten aus Abbildung F.3 besteht aus drei Teilaktivitäten, die in Abbildung F.4 gezeigt sind. Da eine Zuweisung von Fehlern meist nicht für alle Systeme einheitlich zu realisieren ist, müssen zuerst Vorschriften zur Generierung erstellt werden. Anhang A enthält eine solche Vorschrift für Ausgaben. Diese Vorschriften erfassen auch Informationen aus den in Abschnitt 4.3 vorgestellten Implementierungsfehlern. Anhand der Vorschriften werden Fehlermodelle generiert und zugewiesen. Diese werden dann in einem dritten Schritt überarbeitet. Die genaue Ausprägung der Schritte ist in Kapitel 4 bei den jeweiligen Modellierungstechniken beschrieben. Auf Basis des aufbereiteten Systemmodells und des Fehlermodells kann die Zusammenhangsanalyse stattfinden.

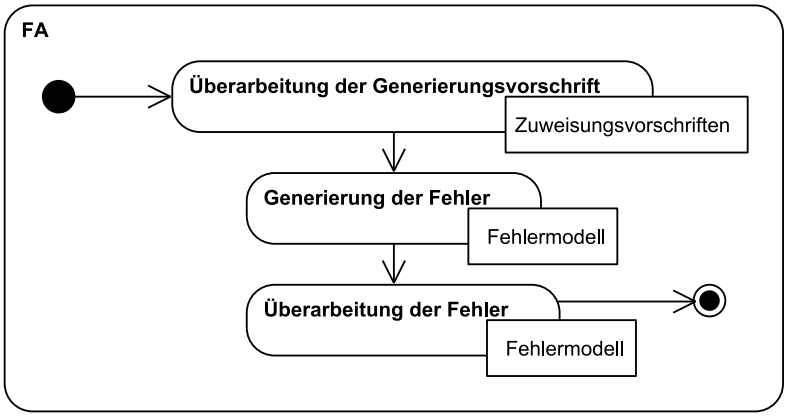


Abb. F.4: Fehlermodellierung und -zuweisung (Teilschritte)

Zusammenhang

Die Zusammenhangsmodellierung und -ermittlung ist in Abbildung F.5 skizziert. Diese Modellierung und die Ermittlungstechniken sind in Kapitel 5 beschrieben. Wesentlich sind auch hier die Modellierungstechniken, die für Ursache und Wirkung verwendet werden. Abhängig davon steht eine Reihe möglicher Techniken zur Ermittlung zur Verfügung, die in Kapitel 5 erfasst sind. Ein wesentlicher weiterer Einflussfaktor sind die Software-Werkzeuge und die Komplexität der Modelle. Kapitel 3 behandelt diese Werkzeug-Faktoren. Die Entscheidung, welche konkreten Zusammenhangsmodelle und Methoden verwendet werden, ist immer eine Abwägung zwischen dem Aufwand, den technischen Möglichkeiten und der Qualität des Ergebnisses. Ein Aktivitätsdiagramm mit klaren Entscheidungen ist hier nicht zu erstellen. Abbildung F.6 zeigt für die Vernetzung von Ausgaben einen oberflächlichen Vorschlag, welche Aktivitäten zur Ermittlung der Zusammenhänge verwendet werden könnten. Allerdings ist hier die Abstraktion außer Acht gelassen. Eine Entscheidung hinsichtlich der zu verwendenden Abstraktionen, Methoden und Werkzeuge muss im konkreten Falle immer ein Experte treffen, der die Eigenschaften der Modelle, die zur Verfügung stehenden Werkzeuge, die Kapazitäten der Entwickler und die Anforderungen an die Sicherheit kennt.

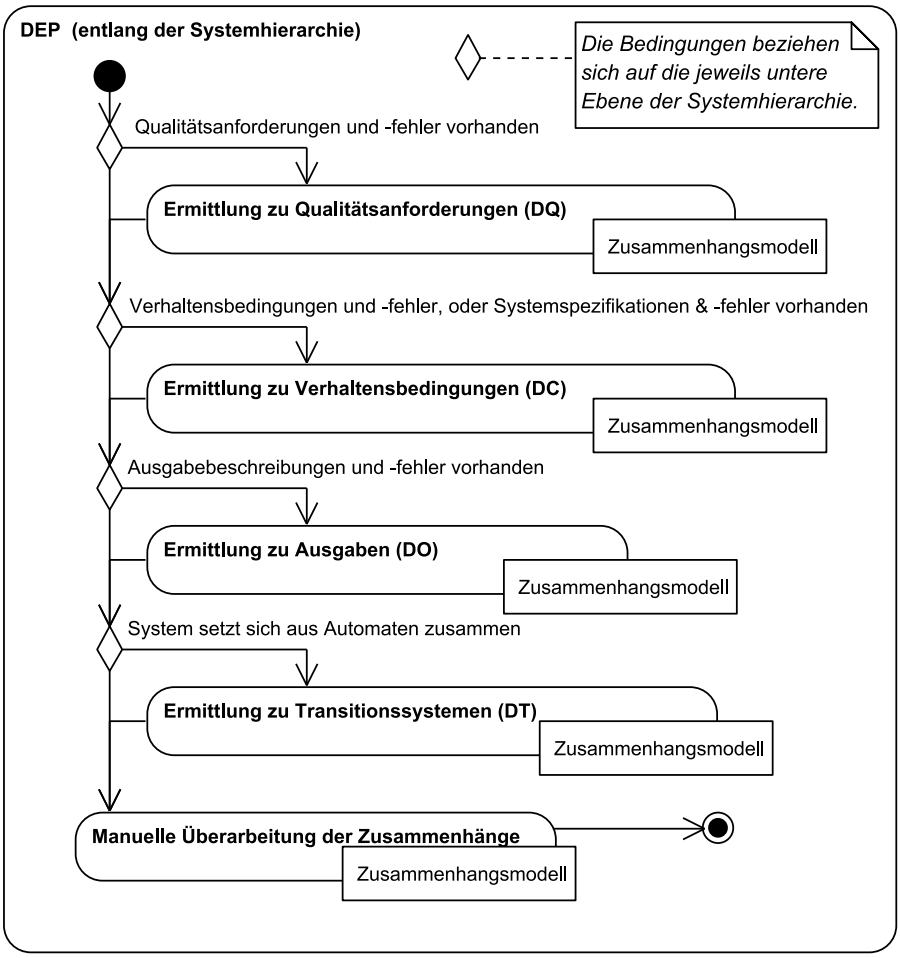


Abb. F.5: Zusammenhangsmodellierung und -ermittlung

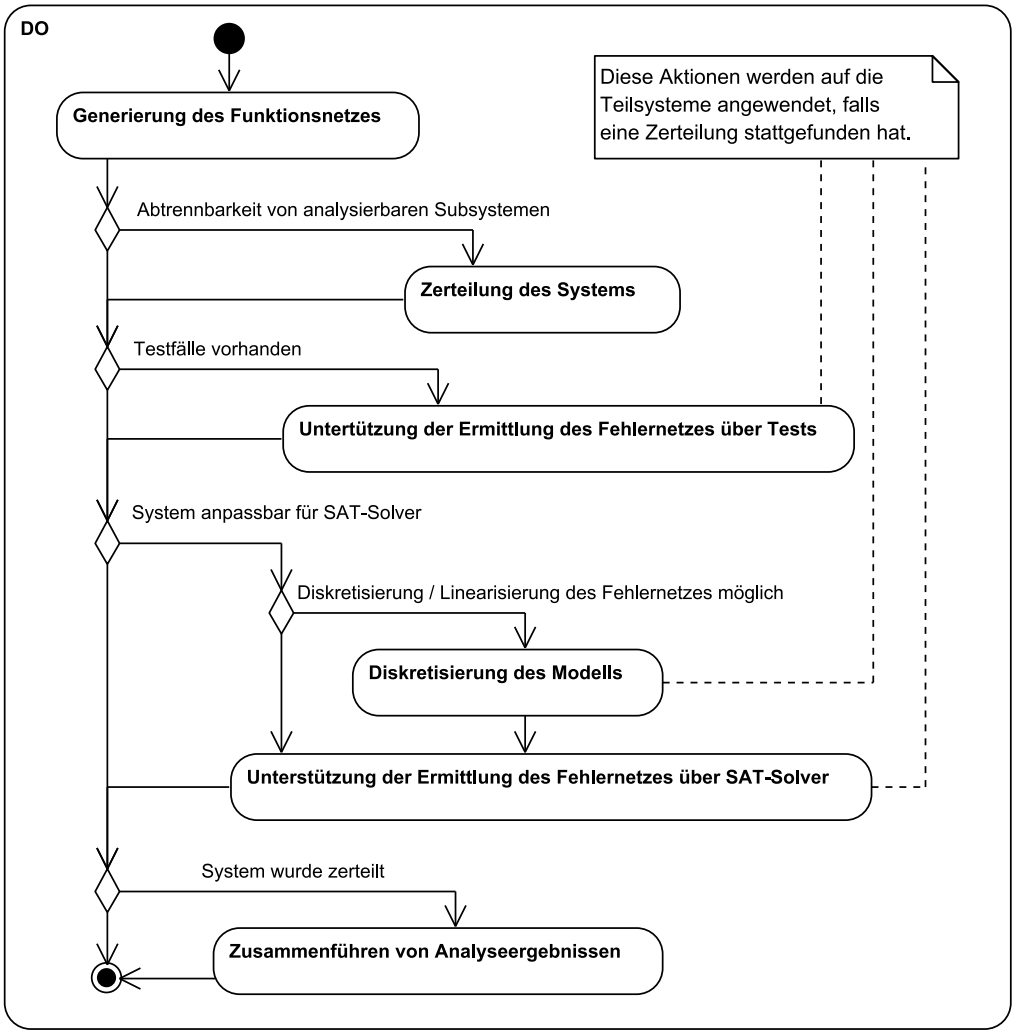


Abb. F.6: Ermittlung der Zusammenhänge zwischen Ausgaben

Abbildungsverzeichnis

Literaturverzeichnis

[AD94] Alur, Rajeev ; Dill, David L.: *A theory of timed automata*. In: *Theoretical Computer Science* 126 (1994), Nr. 2, S. 183–235

[Ang07] Angermann, Anne: *Matlab, Simulink, Stateflow - Grundlagen, Toolboxen, Beispiele*. Bd. 5., aktualisierte Aufl. München : Oldenbourg, 2007

[AUT06] AUTOSAR GbR. *Technical Overview V2.0.1*. 2006

[Bal01] Balzert, Helmut: *Lehrbuch der Software-Technik: Software-Entwicklung*. 1. Spektrum Akademischer Verlag GmbH, 2001

[BBR+05] Bauer, Andreas ; Broy, Manfred ; Romberg, Jan ; Schätz, Bernhard ; Braun, Peter ; Freund, Ulrich ; Mata, Núria ; Sandner, Robert ; Ziegenbein, Dirk: *AutoMoDe — Notations, Methods and Tools for Model-Based Development of Automotive Software*. In: *Proceedings of the SAE 2005 World Congress*. Detroit, MI : Society of Automotive Engineers, April 2005

[BDD+92] Broy, Manfred ; Dederich, Frank ; Dendorfer, Claus ; Fuchs, Max ; Gritzner, Thomas ; Weber, Rainer: *The Design of Distributed Systems - An Introduction to FOCUS* / Institut für Informatik, Technische Universität München. 1992 (I-9202). – Forschungsbericht. www.tum.de

[Bec03] Beck, Kent: *Test Driven Development: By Example*. Addison-Wesley, 2003 (The Addison-Wesley Signature Series)

[BPT07] Bauer, Andreas ; Pister, Markus ; Tautschnig, Michael: *Tool-support for the analysis of hybrid systems and models*. In: *Proceedings of the 2007 Conference on Design, Automation and Test in Europe (DATE)*. Los Alamitos, CA : IEEE Computer Society, April 2007

[Bre01] Breitling, Max D.: *Formale Fehlermodellierung für verteilte reaktive Systeme*, Lehrstuhl für Software und Systems Engineering, Institut für Informatik, Technische Universität München, Diss., 2001

[Bro98] Broy, Manfred: *Compositional Refinement of Interactive Systems Modelled by Relations*. In: *International Symposium Compositionality*, Springer, 1998, S. 130 – 149

[Bro02] Broy, M.: *Unifying Models and Engineering Theories of Composed Software Systems.* In: Broy, M. (Hrsg.) ; Pizka, M. (Hrsg.): *Models, Algebra and Logic of Engineering Software. Marktoberdorf Summer School 2002* Bd. 191, Springer, 2002

[BRS00] Braun, Peter ; Rappl, Martin ; Schaeuffele, Joerg: *Softwareentwicklungen fuer Steuergeraetenetzwerke Eine Methodik fuer die fruehe Phase.* In: *VDI Kongress: Elektrik im KFZ, BadenBaden, VDI-Berichte Nr. 1547*, 2000, S. 265 ff.

[BRS05] Bauer, Andreas ; Romberg, Jan ; Schätz, Bernhard. *The AUTOMODE Design Notation: Concepts, Consistency and Timing.* 2005

[BS01] Broy, M. ; Stoelen, K.: *Specification and Development of Interactive Systems: Focus on Streams, Interfaces and Refinement.* 1. Springer, 2001

[BSN07] Balzer, Heinrich ; Stein, Benno ; Niggemann, Oliver: *Diagnose in verteilten automotiven Systemen.* In: Gausemeier, Jürgen (Hrsg.): *5. Paderborner Workshop Entwurf mechatronischer Systeme* Bd. 210, Heinz Nixdorf Institut, March 2007. – ISBN 978–3–939350–29–3, S. 243–254

[BSOB04] Bünte, T. ; Schweiger, C. ; Odenthal, D. ; Baumgarten, G. *Modellierung, Regelung, Simulation und Bewertung der Fahrdynamik.* Erster verkehrstechnischer Tag, DLR Braunschweig. November 2004

[CG01] Clarke, Edmund ; Grumberg, Oma: *Model checking.* Bd. 3. MIT Press, 2001

[CL71] Chang, C.-L. ; Lee, R. C.-T.: *The Resolution Principle.* In: Chang, C.-L. (Hrsg.) ; Lee, R. C.-T. (Hrsg.): *Symbolic Logic and Mechanical Theorem Proving.* New York : Academic Press, 1971, S. 70–99

[CMZ02] Cau, A. ; Moszkowski, B. ; Zedan, H. *ITL - Interval Temporal Logic.* Software Technology Research Laboratory, De Monfort University. 2002

[DIN04] Normenreihe 2004. *DIN EN 61508: Funktionale Sicherheit - Sicherheitssystem (E / E / PES), VDE-Verlag (Berlin)*

[Duf91] Duffy, David A.: *Principles of automated theorem proving.* Wiley, 1991 (Wiley professional computing). – XVIII, 243 S. S

[Ech90] Echtle, Klaus: *Fehlertoleranzverfahren.* 1. Springer, 1990

[EM07] Essigkrug, Andreas ; Mey, Thomas: *Rational Unified Process kompakt.* 2. Elsevier, Spektrum Akademischer Verlag, 2007

[Eri05] Ericson, Clifton A.: *Hazard Analysis Techniques for System Safety.* 1. John Wiley and Sons, Inc., Hoboken,New Jersey, 2005

[ES07a] Esser, Michael ; Struss, Peter: *Fault-model-based Test Generation for Embedded Software.* In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007

[ES07b] Esser, Michael ; Struss, Peter: *Obtaining Models for Test Generation from Natural-language-like Functional Specifications.* In: *G. Biswas et. al. (eds.), DX'07, 18th International Workshop on Principles of Diagnosis*, 2007

[EWS05] Etessami, Kousha ; Wilke, Thomas ; Schuller, Rebecca A.: *Fair Simulation Relations, Parity Games, and State Space Reduction for Buchi Automata.* In: *SIAM J. Comput.* 34 (2005), Nr. 5, S. 1159–1175

[FGP04] Fleischmann, Andreas ; Geisberger, Eva ; Pister, Markus: *Herausforderungen für das Requirements Engineering eingebetteter Systeme* / Institut für Informatik, Technische Universität München. 2004 (I-0414). – Forschungsbericht. www.tum.de

[Fow03] Fowler, Martin: *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* 1. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2003. – ISBN 0321193687

[GSB97] Grosu, R. ; Stoelen, K. ; Broy, M.: *A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing* / Institut für Informatik, Technische Universität München. 1997 (I-9724). – Forschungsbericht. www.tum.de

[HCRP91] Halbwachs, N. ; Caspi, P. ; Raymond, P. ; Pilaud, D.: *The synchronous data-flow programming language LUSTRE.* In: *Proceedings of the IEEE* 79 (1991), September, Nr. 9, S. 1305–1320

[Hen96] Henzinger, Thomas: *The Theory of Hybrid Automata.* In: *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96).* New Brunswick, New Jersey : IEEE, 1996, S. 278–292

[HHK03] Henzinger, T.A. ; Horowitz, B. ; Kirsch, C.M.: *Giotto: a time-triggered language for embedded programming.* In: *Proceedings of the IEEE* 91 (2003), Nr. 1, S. 84–99

[IK00] Ishii, Kosuke ; Kmenta, Steven: *Scenario-based FMEA: A Life Cycle Cost Perspective.* In: *DETC 2000.* Baltimore, Maryland : 2000 ASME Design Engineering Technical Conferences, 2000

[Int90] International Electronical Commission. *IEC 1025 - International Standard - Fault Tree Analysis (FTA).* 1990

[ISO94] ISO. *ISO 7498 - Information processing systems - open systems interconnection - basic reference model.* 1994

[ISO01] ISO. *ISO 9126 - Software engineering – Product quality – Part 1: Quality model*. 2001

[ISO05] ISO. *ISO/IEC 19501 - Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2*. 2005

[Jon00] Jones, Capers: *Software assessments, benchmarks, and best practices*. 1. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2000. – ISBN 0–201–48542–7

[Kor05] Koreimann, Dieter S.: *Projekt-Controlling*. 1. WILEY-VCH GmbH und Co. KGaA, Weinheim, 2005

[Kre06] Kreutz, Martin: *Sicherheitstesten kontextbezogener Dienste*, Technische Universität München, Diplomarbeit, Mai 2006

[KTB+07] Kugele, Stefan ; Tautschnig, Michael ; Bauer, Andreas ; Schallhart, Christian ; Merenda, Stefano ; Haberl, Wolfgang ; Kühnel, Christian ; Müller, Florian ; Wang, Zhonglei ; Wild, Doris ; Rittmann, Sabine ; Wechs, Martin: *COLA - The Component Language* / TUM, Institut für Informatik. 2007 (TUM-I0714). – Forschungsbericht

[Lam02] Lamport, Leslie: *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Pearson Education, Inc, 2002

[Lev95] Leveson, N.: *Safeware - System, Safety and Computers*. 1. Addison-Wesley Publishing Company, 1995

[Lig02] Liggesmeyer, Peter: *Software-Qualität*. Spektrum Akademischer Verlag, 2002

[Loe03] Loetzbeyer, Heiko: *Modellbasierte Testfallermittlung für eingebettete Systeme in sicherheitskritischen Anwendungen*, Technische Universitaet Muenchen, Diss., 2003

[LSV95] Lynch, Nancy A. ; Segala, Roberto ; Vaandrager, Frits W.: *Hybrid I/O automata*. In: *82*. ISSN 0169-118X : Centrum voor Wiskunde en Informatica (CWI), 31 1995, S. 16

[LSV01] Lynch, Nancy ; Segala, Roberto ; Vaandrager, Frits: *Hybrid I/O Automata Revisited*. In: *Lecture Notes in Computer Science* 2034 (2001), S. 403+

[Mil89] Milner, Robin: *Communication and Concurrency*. 1. Prentice Hall, 1989

[Min65] Minsky, M.: *Matter, mind and models*. In: Kalenichl, A. (Hrsg.): *Proceedings International Federation of Information Processing Congress*, Spartan Books, 1965

[MK05] Maeckel, Oliver ; Kaiser, Bernhard: *Sicherheits- und Zuverlaessigkeitsanalysetechniken.* In: Liggesmeyer, Peter (Hrsg.) ; Rombach, Peter (Hrsg.): *Software Engineering eingebetteter Systeme.* Elsevier Verlag, 2005

[MP01] Maruhn, M. ; Papadopoulos, Y.: *Model-based automated synthesis of fault trees from Mathlab-Simulink models.* In: *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2001)*, 2001, S. 77–82

[MR98a] Maraninchi, F. ; Remond, Y. *Running modes in a data-flow language: Modeautomata and their implementation.* 1998

[MR98b] Maraninchi, Florence ; Rémond, Yann: *Mode-Automata: About Modes and States for Reactive Systems.* In: *Lecture Notes in Computer Science* 1381 (1998), S. 185 ff.

[Mus99] Musa, J.: *Software Reliability Engineering.* 1. McGraw-Hill, 1999

[NPW02] Nipkow, Tobias ; Paulson, Lawrence C. ; Wenzel, Markus: *LNCS.* Bd. 2283: *Isabelle/HOL — A Proof Assistant for Higher-Order Logic.* Springer, 2002

[OR04] Ortmeier, F. ; Reif, W.: *Failure-Sensitive Specification: A formal method for finding failure modes* / Universität Augsburg, Institut für Informatik. 2004 (2004-3). – Forschungsbericht. www.uni-augsburg.de

[ORS05] Ortmeier, Frank ; Reif, Wolfgang ; Schellhorn, G. *Formal Safety Analysis Of A Radio-Based Railroad Crossing Using Deductive Cause Consequence Analysis.* 2005

[PCB+55] Picardi, Claudia ; Console, Luca ; Berger, Frederic ; Breeman, Jan ; Kanakis, Tony ; Moelands, Jeroen ; Collas, Stephan ; Arbaretier, Emanuel ; Domenico, Nino D. ; Girardelli, Ermanno ; Dressler, Oskar ; Struss, Peter ; Zilbermann, Benjamin. *AUTAS: a tool for supporting FMECA generation in aeronautic systems.* 5555

[Pet62] Petri, Carl A.: *Kommunikation mit Automaten.*, Diss., 1962

[PP08] Pfaller, Christian ; Pister, Markus. *Combining Structural and Functional Test Case Generation.* 2008

[PPG04] Papadopoulos, Y. ; Parker, D. ; Grante, C.: *A method and Tool Support for Model-based Semi-automated Failure Modes and Effects Analysis of Engineering Designs.* In: *Conferences in Research and Practice in Information Technology* Bd. 38. Brisbane, Australia : Australian Computer Society, 2004

[Pre03] Pretschner, Alexander W.: *Zum modellbasierten funktionalen Test reaktiver Systeme*, Technische Universitaet Muenchen, Diss., 2003

[Rei85] Reisig, Wolfgang: *Petri nets: an introduction*. New York, NY, USA : Springer-Verlag New York, Inc., 1985. – ISBN 0–387–13723–8

[RN03] Russell, Stuart (Hrsg.) ; Norvig, Peter (Hrsg.): *Artificial Intelligence - A Modern Approach*. 2. Alan R. Apt, 2003

[RR99] Robertson, Suzanne ; Robertson, James: *Mastering the requirements process*. 1. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1999. – ISBN 0–201–36046–2

[Sac01] Sachenbacher, Martin: *Automated Qualitative Abstraction and its Application to Automotive Systems*, Technische Universitaet Muenchen, Diss., 2001

[Sch03] Schneider, Dr.-Ing. R. *Grundlagen der Simulationstechnik*. 2003

[SSL+95] Sampath, Meera ; Sengupta, Raja ; Lafortune, Stephane ; Sinnamohideen, Kasim ; Teneketzis, Demosthensis: *Diagnosability of discrete-event systems*. In: *IEEE Transactions on Automatic Control* Bd. 40. IEEE, September 1995. – ISSN: 0018-9286, DOI: 10.1109/9.412626, S. 1555–1575

[SSL+96] Sampath, Meera ; Sengupta, Raja ; Lafortune, Stephane ; Sinnamohideen, Kasim ; Teneketzis, Demosthensis: *Failure diagnosis using discrete-event models*. In: *IEEE Transactions on Control Systems Technology* Bd. 4. IEEE, March 1996. – ISSN: 1063-6536, DOI: 10.1109/87.486338, S. 105–124

[Str04] Struss, Peter. *Deviation Models Revisited*. August 2004

[Str06] Struss, Peter: *A Model-Based Methodology For The Integration Of Diagnosois And Fault Analysis During The Entire Life Cycle*. In: *Preprints of the 6th IFAC Symposium on Fault Detection*. Beijing, China : Supervision and Safety of Technical Processes (Safeprocess), August 2006

[Thu04] Thums, Andreas: *Formale Fehlerbaumanalyse*, Fakultät für angewandte Informatik, Universität Augsburg, Diss., 2004

[VDI04] VDI. *Entwicklungsmethodik für mechatronische Systeme*. Beuth Verlag. 2004

[VGRH81] Vesely, W.E. ; Goldberg, F.F. ; Roberts, N.H. ; Haasl, D.F.: *Fault Tree Handbook - (NUREG-0492)*. 1. Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1981

[Voe02] Voelker, Lars. *Die Bedeutung von Public Relations und Corporate Identity fuer den Erfolg von Unternehmen*. Ausarbeitung fuer Entrepreneurship Seminar, Universitaet Karlsruhe. Juli 2002

[Wag07] Wagner, Stefan: *Cost-Optimisation of Analytical Software Quality Assurance*, Technische Universität München, Diss., 2007

[WFH+06] Wild, Doris ; Fleischmann, Andreas ; Hartmann, Judith ; Pfaller, Christian ; Rappl, Martin ; Rittmann, Sabine: *An Architecture-Centric Approach towards the Construction of Dependable Automotive Software.* In: *Proceedings of the SAE 2006 World Congress*, 2006

[Win08] Winter, Sebastian: *Modellbasierte Analyse von Nutzerschnittstellen*, Diss., 2008. – to be published