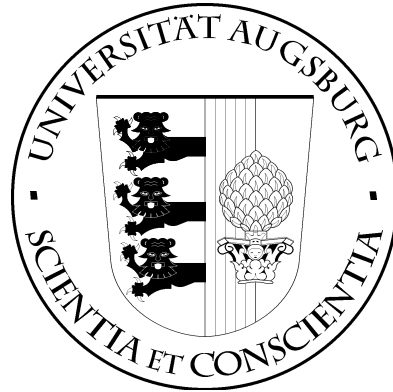


UNIVERSITÄT AUGSBURG

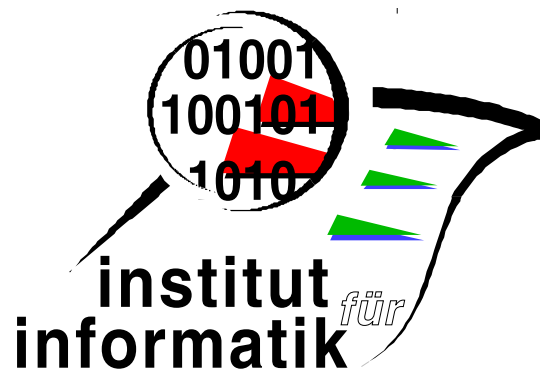


**Integration formaler Spezifikation und
Sicherheitsanalyse**

W. Reif, G. Schellhorn, A. Thums

Report 2001-6

Juni 2001



INSTITUT FÜR INFORMATIK

D-86135 AUGSBURG

Copyright © W. Reif, G. Schellhorn, A. Thums
Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
<http://www.Informatik.Uni-Augsburg.DE>
— all rights reserved —

Integration formaler Spezifikation und Sicherheitsanalyse

W. Reif, G. Schellhorn, A. Thums*
Abt. Programmiermethodik, Universität Ulm, 89069 Ulm
email: {reif, schellhorn, thums}@informatik.uni-ulm.de

5. Juni 2001

Zusammenfassung

Es wird eine Methode zur systematischen Erstellung sicherer Spezifikationen für software-basierte Ingenieur Anwendungen entwickelt. Dabei steht 'sicher' sowohl für korrektes Funktionieren als auch für eine ausfalltolerante Gestaltung. Formale Spezifikationen werden dabei zur Beschreibung des Systemmodells sowie zur Formulierung und Verifikation von Sicherheitseigenschaften verwendet. Mit den Techniken der Sicherheitsanalyse werden Schwachstellen und Designfehler ermittelt. Die Ergebnisse werden anhand der Referenzfallstudie 'Verkehrsleittechnik' des Schwerpunkts evaluiert.

1 Überblick

Das Gesamtziel ist eine Methode zur Entwicklung sicherer Spezifikationen für software-basierte Ingenieur Anwendungen. Der Begriff 'sicher' bezieht sich hierbei sowohl auf das korrekte Funktionieren eines Systems als auch auf das Vermeiden von Gefährdungen durch Ausfälle. Dieses Ziel soll durch eine Integration formaler Spezifikation und Verifikation mit Techniken der Sicherheitsanalyse erreicht werden. Die formalen Methoden werden zur präzisen Beschreibung des Systemmodells sowie zur Formulierung und zum Nachweis von Sicherheitseigenschaften verwendet. Mit den Techniken der Sicherheitsanalyse werden Schwachstellen, Designfehler sowie Sicherheitsanforderungen ermittelt. Dadurch ergibt sich der folgende, wechselseitige Zugewinn:

1. Formale Methoden profitieren von der Sicherheitsanalyse durch die Systematisierung der Suche nach Sicherheitsanforderungen, durch das Herunterbrechen von Anforderungen auf (Software-) Komponenten und – in geeigneten Fällen – durch die Möglichkeit einer quantitativen Beurteilung der Sicherheit.
2. Die Sicherheitsanalyse profitiert von den formalen Methoden durch die Möglichkeit der exakten Beurteilung des Systemverhaltens, durch die Chance auf Automatisierbarkeit der Analyse sowie durch die Möglichkeit, Korrektheit und Vollständigkeit der Analyseergebnisse zu garantieren.
3. Durch die Integration entstehen Systemmodelle, die den beiden komplementären Sicherheitsbegriffen standhalten: Korrektes Funktionieren und möglichst ausfalltolerante Gestaltung.

*Diese Arbeit wurde teilweise im Rahmen des DFG Schwerpunktes „Integration von Techniken der Software-spezifikation für ingenieurwissenschaftliche Anwendungen“ erstellt

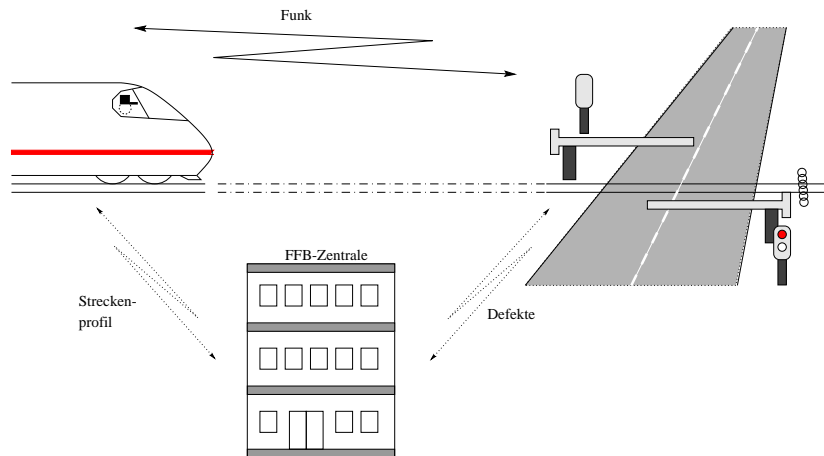


Abbildung 1: Systemstruktur der RefVL

Im Abschnitt 2 wird die Fallstudie Verkehrsleittechnik (RefVL), die im Bericht als Beispiel dient, beschrieben. Auf der formalen Statechart-Modellierung wird eine sicherheitstechnische Analyse mit Hilfe der *fault tree analysis* (FTA) und der *failure modes and effects analysis* (FMEA) durchgeführt, die in im Abschnitt 3 beschrieben wird. Anschließend wird im Abschnitt 4 das Vorgehensmodell für eine formale Sicherheitsanalyse und im Abschnitt 5 die formale Semantik der FTA erläutert.

2 Modellierung der Referenzfallstudie Verkehrsleittechnik

Die Referenzfallstudie Verkehrsleittechnik (RefVL) ist an eine neuartige Entwicklung der Deutschen Bahn, dem funkbasierten Fahrbetrieb (FFB), angelehnt. Der FFB kann für Bahnstrecken mit einer Höchstgeschwindigkeit von 160 km/Std eingesetzt werden. Der Unterschied dieser neuen Technologie gegenüber der herkömmlichen Sicherung besteht darin, dass Sensoren und Signale durch Berechnungen in der Zug- und Bahnübergangsteuerung und durch Kommunikation über Funk ersetzt werden. Dies bietet einen flexibleren und billigeren Einsatz. So kann der Schließzeitpunkt des Bahnübergangs vom Zug in Abhängigkeit von seiner momentanen Geschwindigkeit berechnet und die Wartezeiten am Bahnübergang minimiert werden.

Der Schließvorgang eines Bahnübergangs wird von dem sich nähernden Zug angestoßen (siehe Abb. 1). Erhält der Bahnübergang ein Schließsignal, schaltet er die Lichtzeichenanlage an, d. h. zuerst das Gelblicht und etwas später das rote Licht. Dann werden die Schranken geschlossen. Wenn diese vollständig geschlossen sind, ist der Bahnübergang nur für eine bestimmte Zeitspanne – damit die Wartezeiten für andere Verkehrsteilnehmer nicht zu lang werden – gesichert. Kann der Bahnübergang aufgrund von Defekten nicht gesichert werden, ist er ungesichert. Durch eine Statusabfrage erhält die Zugsteuerung Auskunft über den Zustand des Bahnübergangs. Ist er gesichert, kann der Zug den Bahnübergang passieren, ansonsten muss die Zugsteuerung dafür sorgen, dass der Zug rechtzeitig vor dem Bahnübergang zum Stehen kommt. Der Bahnübergang kann dann nur noch durch ein manuelles Sicherungsverfahren passiert werden.

3 Sicherheitstechnische Analyse der Referenzfallstudie Verkehrsleittechnik

Auf der formalen Modellierung der Referenzfallstudie Verkehrsleittechnik (Klose & Thums, 2000), die das in Abschnitt 2 beschriebene Szenario spezifiziert, ist eine sicherheitstechnische Analyse durchgeführt worden. Es wurden die Techniken der *failure modes and effects analysis* (FMEA) und der *fault tree analysis* (FTA) angewandt. Die Analyse diente zum einen dazu, eine sichere Modellierung der Fallstudie zu erhalten und mögliche sicherheitstechnische Risiken der neuen Technologie des funkbasierten Fahrbetriebs zu evaluieren, zum anderen aber auch dazu, eine Vorstellung dafür zu bekommen, welche Eigenschaften eine formale Semantik der Analysemethoden (insbesondere Fehlerbäume, siehe Abschnitt 5) besitzen muss.

FMEA. Bei der FMEA wird der Ausfall einer Systemkomponente betrachtet und die Auswirkung dieses Ausfalls auf das Gesamtsystem untersucht. Der Ausfall einer Komponente wird in verschiedene Ausfallmodi zerlegt, die dann einzeln untersucht werden. Ein Ausfallmodus beschreibt die Art des Ausfalls (bei einer Bremse wurden z. B. die Modi *Totalausfall*, *geringere Bremskraft* und *nur Vollbremsung möglich* untersucht). Für jeden Ausfallmodus wird der Grund des Ausfalls untersucht, welche Auswirkungen dieser Ausfall auf die lokalen Komponenten besitzt und welche auf die übergeordneten Komponenten bzw. auf das Gesamtsystem. Des Weiteren wird die Kritikalität des Ausfalls (bzw. der Wirkung des Ausfalls) und mögliche Verbesserungsvorschläge notiert. Um die verschiedenen Punkte systematisch zu bearbeiten, wird für die FMEA eine Tabelle wie in Abbildung 2 oder (Bahr, 1997; Storey, 1996; Leveson, 1995) benutzt.

In dieser Studie wurden sowohl Ausfälle der Hardware als auch Ausfälle bei der Software untersucht. Da die Untersuchung von Hardwareausfällen in den Ingenieurwissenschaften schon hinreichend bekannt ist, wird im Folgenden eine FMEA der Softwarekomponente *velocity_curve* beschrieben.

Diese Softwarekomponente berechnet die Geschwindigkeitskurve für den Zug, in der zu jedem Streckenpunkt eine Maximalgeschwindigkeit zugeordnet ist. Berechnet sie zu hohe Maximalgeschwindigkeiten, da falsche Verzögerungswerte benutzt wurden oder die Berechnungsvorschrift fehlerhaft ist, kann der Zug vor einem ungesicherten Bahnübergang eventuell nicht mehr rechtzeitig anhalten. Daraus kann eine Kollision resultieren und da kein Ausfall einer weiteren Komponente für diese Auswirkung notwendig ist, handelt es sich um einen *single-point of failure*, der als sehr kritisch eingestuft wird. Die Betrachtung weiterer Fehlermodi ist aus der Abbildung 2 zu entnehmen.

Der Nachteil einer FMEA ist, dass systematisch nur Auswirkungen einzelner Komponentenausfälle berücksichtigt werden können. Um untersuchen zu können, ob ein Systemfehler durch Zusammenwirken verschiedener Ausfälle entstehen kann, wird die FTA benutzt. Sie ist im Gegensatz zur FMEA, die eine *bottom up* Analyse ist, eine *top down* Analyse.

FTA. Eine FTA betrachtet eine auszuschließende Gefahr (Top Event) und zerlegt sie schrittweise in einzelne Ursachen, die zu dieser Gefahr führen. Die Ursachen werden über Und- bzw. Oder-Gatter¹ miteinander verbunden, d. h. dass ein Ereignis eintritt, wenn alle Unter-Ereignisse (Und-Gatter) oder wenn mindestens ein (Oder-Gatter) Unter-Ereignis eintritt. Dieser Vorgang wird für die entstehenden Unter-Ereignisse wiederholt, bis ein gewünschter

¹es gibt für die FTA noch weitere Gatter (Vesely *et al.*, 1981), die hier jedoch nicht verwendet wurden

(Sub-)System: _____		Failure Mode and Effects Analysis		Prepared by: _____		Date: _____		
name (identification)	function	failure mode	failure cause	local effect	system effect	single-point of failure	failure class	control, recommendation
velocity_curve	Diese Funktion berechnet die Geschwindigkeitskurve. Diese dient dazu, jedem Streckenpunkt eine Maximalgeschwindigkeit zuzuordnen. Dazu betrachtet die Funktion das Streckenprofil, die gesetzten Gefahrepunkte sowie die Verzögerungswerte des Zuges.	<p>1. zu hoch</p> <p>2. zu niedrig</p> <p>3. willkürlich</p>	<ul style="list-style-type: none"> fehlerhafte Berechnungsvorschrift fehlerhafte Parameter – Verzögerung – Streckenatlas – Gefahrepunkte <p>siehe oben</p>	<p>es wird (zu spät) gebremst</p> <p>es wird zu früh gebremst</p> <p>unnötiges Bremsen/Beschleunigen</p>	<p>zu hohe Geschwindigkeit:</p> <ul style="list-style-type: none"> • pot. Entgleisen • pot. Einfahrt in ungesicherten Bahnhübelgang <p>zu niedrige Geschwindigkeit:</p> <ul style="list-style-type: none"> • Blockierung Autos • Blockierung Zug • Einfahrt in ungesicherten Bahnhübelgang, wenn sich Schranken nach 240 Sek. wieder öffnen <p>andauerndes Beschleunigen/Bremsen:</p> <ul style="list-style-type: none"> • Verschleiß • Defekte 	<p>y</p> <p>y</p> <p>y</p> <p>y</p> <p>y</p> <p>y</p>	<p>A</p> <p>A</p> <p>C</p> <p>C</p> <p>A</p> <p>B</p> <p>B</p>	<p>Plausibilitätschecks</p> <p>Verifikation der kritischen Berechnungsfunktion</p>

failure class: A = catastrophic, B = critical, C = major, D = minor

Abbildung 2: FMEA: velocity_curve

Detaillierungsgrad erreicht ist. Durch die Verknüpfung der Blätter (sog. Primary Events) entsprechend der Gatter im Fehlerbaum kann man berechnen, welche Primary Events eintreten müssen, damit das Top Event eintritt. Eine Menge solcher Primary Events nennt man *Cut Set*. Ein Cut Set ist ein *minimaler Cut Set*, wenn das Weglassen eines beliebigen Primary Events nicht mehr das Top Event verursacht.

In Abbildung 3 sehen wir einen Fehlerbaum für die Gefahr einer *Kollision*. Wir definieren, dass eine (potentielle) Kollision vorliegt, wenn der Zug sich auf dem Bahnübergang befindet und die Schranken nicht geschlossen sind. Eine Kollision kommt dann zustande, wenn der Zug nicht bremst, obwohl der Bahnübergang nicht signalisiert hat, dass er 'sicher' ist (keine Freigabe), *oder* wenn der Bahnübergang sich als gesichert betrachtet (sendet Freigabe), obwohl die Schranken noch offen sind (oberste Zerlegung im Fehlerbaum). Auf diese Art und Weise werden die erzeugten (Unter-)Knoten weiter zerlegt. Für die genaue Bedeutung der Gatter wird auf Abschnitt 5 verwiesen.

Bei der Auswertung des Fehlerbaums sieht man, dass die Modellierung keine Redundanzen hat, da keine Und-Gatter vorkommen. Deshalb sind die minimal Cut Sets einelementig, d. h. alle Primary Events sind, wie schon bei der FMEA, single-points of failure. Des Weiteren erkennt man, dass die manuelle Freigabe, bei der das Zugpersonal den Bahnübergang sichert, ein Risiko darstellt, das nicht durch technische Maßnahmen verringert werden kann. Deshalb müssen strenge Vorschriften für diesen Vorgang definiert werden.

Durch die FTA wurde auch ein Fehler im Modell entdeckt, der zu einer Kollision führen kann. Der Bahnübergang ist nur für eine maximale Schließzeit im gesicherten Zustand und öffnet sich danach wieder. Hat der Zug jedoch einmal eine Freigabe erhalten, so nimmt er ab diesem Zeitpunkt den Bahnübergang als gesichert an und löscht den Gefahrenpunkt. Benötigt der Zug nun länger als die maximale Schließzeit zum Passieren des Bahnübergangs, so öffnet dieser wieder. Der Zug gelangt so auf einen geöffneten Bahnübergang und es besteht die Gefahr einer Kollision. Die Lösung des Problem liegt darin, dass die Bahnübergang-Steuerung nach dem Timeout nicht die Schranken öffnet, sondern den Vorgang der Zentrale meldet und auf weitere Anweisungen wartet.

Die Sicherheitsanalyse brachte diesen kritischen Modellierungsfehler in einer frühen Phase der Softwareentwicklung zu Tage und trug dadurch zur Qualitätssicherung auch des informellen Modells bei.

Ergebnisse. Im vorigen Abschnitt wurde lediglich die Kollision als Gefährdung betrachtet. Während der FMEA haben sich aber noch weitere Gefährdungen bzw. unerwünschte Zustände herausgestellt. So kann es zur vermeidbaren Blockierung von Zug und Verkehrsteilnehmern kommen, und es besteht die Gefahr, dass durch Ausfall der Lichtzeichenanlage Verkehrsteilnehmer unter der Schranke eingeklemmt werden. Abbildung 4 stellt die wichtigsten Ursachen für die Situationen *Kollision*, *Blockierung Zug* und *Blockierung Verkehr* dar und gibt die Analysemethode an, mit der die Ursache-Wirkung Beziehung entdeckt wurde.

Bei der Betrachtung der Tabelle scheint es, dass der funkbasierte Fahrbetrieb eine sehr unsichere Sache ist, doch muss man bedenken, dass es sich um die Modellierung der Minimalversion handelt, in der noch keine redundanten Sicherheitsmechanismen bei Bremsen, Odometer, etc. modelliert wurden. Insgesamt eignen sich sowohl FMEA als auch FTA hervorragend zur Analyse softwarebasierter Systeme.

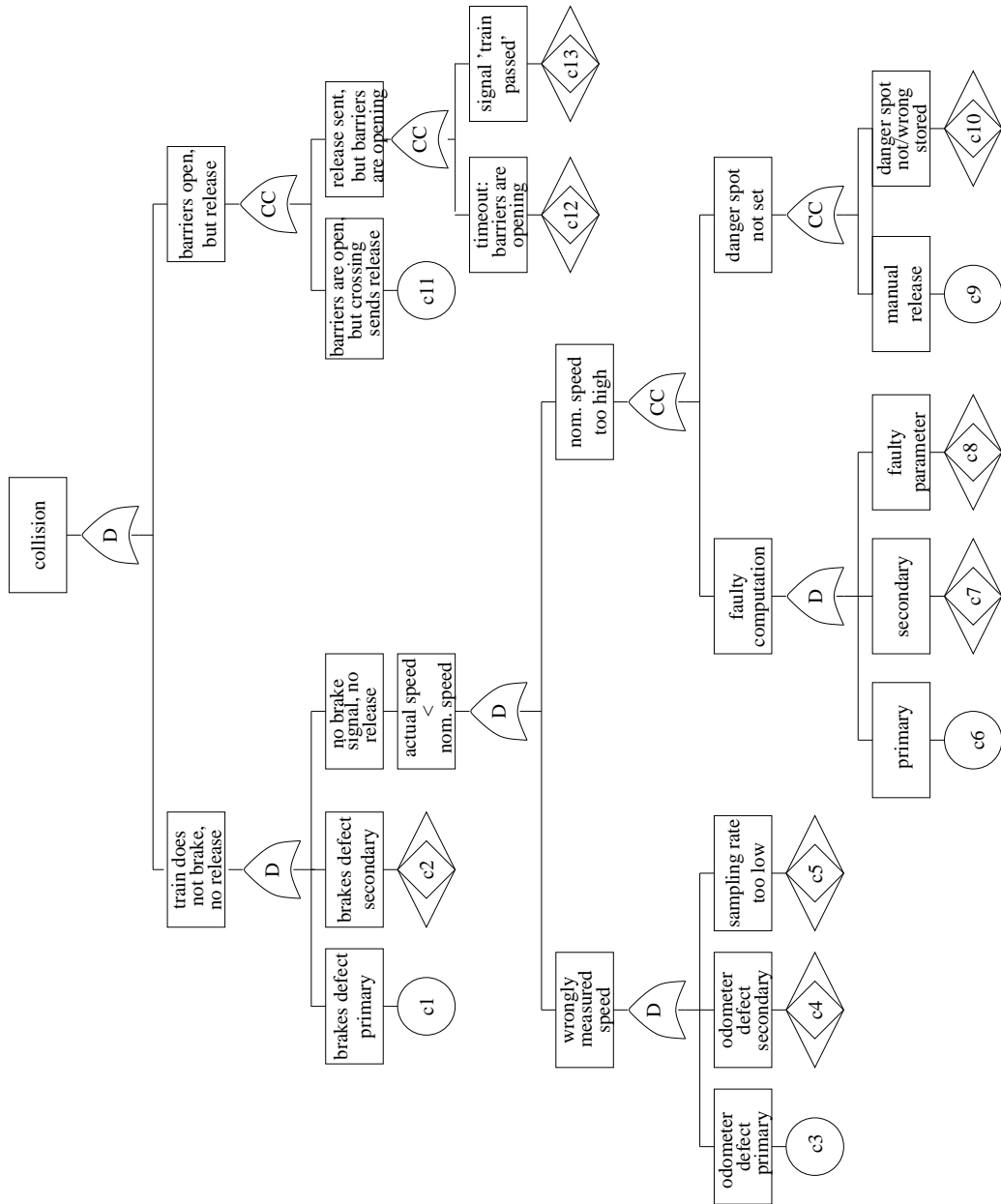


Abbildung 3: FTA: Kollision

Ursache	Kollision	Blockierung	
		Zug	Verkehr
Odometer	FMEA/FTA	FMEA	FMEA
Bremsen	FMEA/FTA	FMEA	FMEA
Berechnung Geschwindigkeitskurve	FMEA/FTA	FMEA	FMEA
Gefahrenpunkt falsch gesetzt	FTA	FMEA	FMEA
Timeoutfehler im Zug	FTA		
Schrankendefekt	– ^a	FMEA	FMEA
Schrankensensor unten	–	FMEA	FMEA
Ausschaltkontakt	–	FMEA	FMEA
Funkkanal	–	FMEA	FMEA

^akeine Ursache für Kollision

Abbildung 4: Sicherheitsanalyse: Ergebnisse

4 Vorgehensmodell für FTA und formale Spezifikation

Im Folgenden wird ein geeignetes Vorgehensmodell für die Kombination von formaler Spezifikation und FTA entwickelt. Übergeordnetes Ziel beider Maßnahmen ist es, die Sicherheit des modellierten bzw. analysierten Systems zu verbessern oder im Idealfall sogar zu garantieren. Formale Spezifikation wird eingesetzt, um die Funktionsweise, also die möglichen Zustände und die möglichen Abläufe des Systems zu präzisieren. FTA wird eingesetzt, um zu analysieren, welche Ursachen dazu beitragen, in einen bestimmten unerwünschten Fehlerzustand, den sog. „Top Event“ zu gelangen.

Ein erster Ansatz ist es, zuerst eine formale Spezifikation zu erstellen, und dann die FTA in den Begriffen der formalen Spezifikation durchzuführen. Dazu werden die informellen Inhalte der Knoten von Fehlerbäumen durch Formeln über der formalen Spezifikation ersetzt. Ist eine formale Semantik für die Verknüpfungen des Fehlerbaums definiert, so wird dadurch eine formale Untersuchung (durch Verifikation) möglich, inwieweit der Fehlerbaum die Ursachen des obersten Fehlers adäquat und/oder vollständig wiedergibt. Bestenfalls kann der Fehlerbaum dazu eingesetzt werden, den Nachweis, dass der oberste Fehler nicht (oder nur unter klar definierten Bedingungen mit einer vernachlässigbar kleinen Wahrscheinlichkeit) auftreten kann, zu vereinfachen oder zu modularisieren. Die FTA auf der formalen Spezifikation aufzusetzen hat den Vorteil, dass die formale Spezifikation evtl. zur automatischen Generierung von Fehlerbäumen eingesetzt werden kann.

Andererseits ist eine wesentliche Idee der FTA die Aufdeckung von Fehlerursachen die beim Systementwurf, der vor allem die Funktionalität und weniger die Sicherheit im Auge hat, nicht oder nicht ausreichend bedacht wurden. Dementsprechend kann FTA nicht nur Fehler innerhalb des Modells, sondern auch Modellierungsfehler aufdecken, und so zur Validierung der Spezifikation beitragen.

Ein typischer Fall ist die häufig anzutreffende Modellierungsannahme, dass intern gespeicherte Zustände mit extern vorhandenen übereinstimmen. Teilt beispielsweise ein Steuerprogramm einem Motor mit, dass er gestartet werden soll, und vermerkt das Starten in einer Zustandsvariable z , so wird häufig der reale Zustand nicht mehr explizit mitmodelliert, sondern es wird angenommen, dass z auch den realen Zustand des Motors wiedergibt. Diese einfache Modellierung ist aber unzulässig, wenn durch FTA Abweichungen der Realität vom

im Rechner angenommenen Zustand als relevante Gefahr erkannt werden.

Deshalb wählen wir zur Zeit eine verzahnte Entwicklung der FTA und der formalen Spezifikation, um die genannten Synergieeffekte auszunutzen. Wir entwickeln zunächst ein informelles Systemmodell, und führen eine informelle FTA durch. Aus dieser ergeben sich die sicherheitsrelevanten Begriffe für das anschließend zu erstellende formale Systemmodell. Schließlich wird die FTA formalisiert, und es wird verifiziert, dass die untersuchten Hazards nicht auftreten können.

5 Definition einer Semantik für Fehlerbäume

Grundlage der Definition einer geeigneten Semantik für Fehlerbäume (engl. fault trees) ist eine vorliegende formale Spezifikation eines Zustandsübergangssystems und ein Fehlerbaum, dessen Knoten (sowohl Und- als auch Oder-Knoten) mit logischen Formeln beschriftet sind. Die Formeln beschreiben bestimmte Fehlerereignisse (engl. fault events). Zur Formalisierung von Fehlerereignissen verwenden wir Formeln des Duration Calculus (Hansen & Zhou Chaochen, 1992; Chaochen *et al.*, 1991), wie dies auch von (Hansen, 1996) vorgeschlagen wird. Damit ist im Gegensatz zu einfacheren Temporallogiken wie LTL oder CTL* auch die Beschreibung von zeitlich ausgedehnten Ereignissen wie z. B. „länger als 10 sec Stromausfall“ möglich. Ergebnis unserer Semantik ist eine Menge von *Bedingungen* (ebenfalls Formeln des Duration Calculus) für die einzelnen Knoten des Fehlerbaums. Lassen sich alle Bedingungen in einem System verifizieren, so modelliert der Fehlerbaum in diesem System tatsächlich die Zerlegung des Top Events an der Wurzel in die angegebenen Primary Events. Entsprechende Theoreme werden am Ende dieses Abschnitts bewiesen.

Ein erster naiver Ansatz zur Definition der Bedingungen, der durch praktisch alle informellen Darstellungen von FTA wie z. B. (Vesely *et al.*, 1981) oder (Leitsch, 1995) suggeriert wird, ist es, die Semantik von Und- und Oder-Knoten einfach durch die Konjunktion und Disjunktion zu definieren. Es wird dann gefordert, dass die Formel eines Konjunktions- bzw. Disjunktionknotens äquivalent zur Konjunktion bzw. Disjunktion der Unterknoten ist. Ein derartiger Ansatz scheint auch deshalb attraktiv, weil er nahelegt, den Aufbau eines Fehlerbaums entlang der Formelstruktur durchzuführen: Wird der Top Event durch eine Formel der Form $\varphi_1 \wedge \varphi_2$ beschreiben, so bilde man eine Konjunktionsverzweigung mit Unterknoten φ_1 und φ_2 .

Leider ist dieser einfache Ansatz aus mehreren Gründen fehlerhaft. Die Probleme lassen sich zum Beispiel an einem Knoten unserer Analyse der Referenzfallstudie demonstrieren. Dieser wird durch die Formel

Schranken offen \wedge Freigabe

beschrieben. Zerlegt man die Formel in die zwei Konjunktionsglieder, so sieht man, dass die entstehenden Teilformeln keine Fehlerereignisse mehr beschreiben. Weder sind Zustände, in denen die Schranke offen ist, *für sich allein* schon fehlerhaft, noch sind es Zustände in denen der Zug ein Freigabesignal erhalten hat. Die Zerlegung einer Konjunktion durch einen Konjunktionsknoten ist also *nur* in den Fällen sinnvoll, wenn das System durch den Ausschluss eines der beiden Konjunktionsglieder sicher gemacht werden soll. Im Beispiel hieße das, dass die Zerlegung nur sinnvoll wäre, wenn die Schranken ständig geschlossen bleiben sollen oder nie eine Freigabe erfolgen soll.

Wir halten daher fest, dass beim Aufbau des Fehlerbaums, der nach Ursachen für die unerwünschte Situation des Wurzelknotens sucht, die Formelstruktur nur dann ausgenutzt werden kann, wenn diese tatsächlich *Fehlersituationen* konjunktiv oder disjunktiv verknüpft.

Im Beispiel folgt eine sinnvolle Zerlegung des Ausgangsfehlers nicht der Formelstruktur, sondern zeigt zwei (disjunktiv verknüpfte) mögliche Ursachen auf:

$$\begin{aligned} &(\text{Übergang sendet Freigabe} \wedge \text{Schranke offen}) \vee \\ &(\text{Freigabe} \wedge \text{Schranke öffnet}) \end{aligned}$$

Die Zerlegung zeigt ein weiteres Problem, wenn die Semantik eines Knotens einfach als eine Disjunktion definiert wird: Sie lässt ausser acht, dass die Ursachen für die Kollision *zeitlich vor* der Wirkung liegen müssen.

Dass dieses Problem in der Literatur nicht zum Vorschein kommt, und fast immer die einfache Semantik angegeben wird, liegt nach unserer Analyse daran, dass meist nur sehr einfache technische Beispielsysteme analysiert werden, bei denen Zeitbedingungen keine Rolle spielen. Alle Knoten sind dann Zerlegungsknoten (im folgenden kurz: Z-Knoten), die nur eine Zerlegung der Menge der unerwünschten Zustände in Teilmengen beschreiben. Da wir dynamische Systeme betrachten, unterscheiden wir davon Knoten wie den oben beschriebenen, deren Inhalt eine Ursache-Wirkungs-Beziehung ist (im folgenden kurz: UW-Knoten). Eine derartige Aufteilung in UW-Knoten (engl. causal nodes) und Z-Knoten (engl. decompositional nodes) wird informell auch in (Górski, 1994) vorgeschlagen.

Ein einfaches Beispiel eines konjunktiven Z-Knotens wäre etwa die Zerlegung von „kein Strom“ in „kein Strom von Generator“ und „kein Strom von Batterie“. Derartige Z-Knoten gibt es auch in unserer Referenzfallstudie, z. B. die disjunktive Zerlegung des Fehlers „Zug bremsst nicht, obwohl keine Freigabe“ in verschiedene Teilmengen von Zuständen (Bremsse defekt, fehlerhafte Berechnung des Bremszeitpunkts etc.). Es fällt auf, dass Fehlerzustände, die aus einem Z-Knoten resultieren (wie hier: „Bremsse defekt“) häufig weiter durch UW-Knoten verfeinert werden (ein primärer Defekt als Ursache wie „Bremserschlauch gerissen“ muss wieder zeitlich vorher auftreten).

Aus der Tatsache, dass bei UW-Knoten Ursache und Wirkung nicht gleichzeitig eintreten, ergibt sich, dass wir die Semantik nicht wie bei Z-Knoten als einfache Äquivalenz, sondern durch zwei Bedingungen wiedergeben müssen: die *Korrektheitsbedingung* garantiert, dass wenn die angegebene Ursache eintritt, die Wirkung nicht ausbleiben kann. Die *Vollständigkeitsbedingung* stellt sicher, dass die Ursachen vollständig sind, d. h. dass wenn keine der Ursachen vorliegt auch die Wirkung nicht eintritt. Wir nehmen eine analoge Aufteilung der Äquivalenz auch für Z-Knoten vor: Die Korrektheitsbedingung entspricht hier der Implikation vom Zerlegungsergebnis zum zerlegten Ereignis, die Vollständigkeitsbedingung ihrer Umkehrung. Schließlich ist für die Definition der Semantik noch zu beachten, dass bei konjunktiven UW-Knoten unterschieden werden muss, ob die Ursachen *gleichzeitig* auftreten müssen, um zur Wirkung zu führen (dies scheint die übliche Semantik in informellen Darstellungen zu sein). Wir nennen derartige konjunktive UW-Knoten „synchron“. Bei einer asynchronen UW-Konjunktion genügt das Auftreten der Ursachen in beliebiger Reihenfolge für die Wirkung (beachte: bei Z-Knoten gibt es nur den Fall der Gleichzeitigkeit).

Zusammenfassend erhalten wir also 5 Typen von Knoten: konjunktive und disjunktive Z-Knoten, sowie synchron konjunktive, asynchron konjunktive und disjunktive UW-Knoten. Die Korrektheits- und Vollständigkeitsbedingungen für jeden dieser Knotentypen (in der obigen Reihenfolge) gibt Abb. 5. Diese Klassifikation ist neu.

	Korrektheit	Vollständigkeit
	$\boxtimes (\varphi_1 \wedge \varphi_2 \rightarrow \psi)$	$\boxtimes (\psi \rightarrow \varphi_1 \wedge \varphi_2)$
	$\boxtimes (\varphi_1 \vee \varphi_2 \rightarrow \psi)$	$\boxtimes (\psi \rightarrow \varphi_1 \vee \varphi_2)$
	$\diamond (\varphi_1 \wedge \varphi_2) \rightarrow \diamond \psi$	$\neg (\neg \diamond (\varphi_1 \wedge \varphi_2) ; \diamond \psi)$
	$\diamond \varphi_1 \wedge \diamond \varphi_2 \rightarrow \diamond \psi$	$\neg (\neg \diamond \varphi_1 ; \diamond \psi) \wedge \neg (\neg \diamond \varphi_2 ; \diamond \psi)$
	$\diamond \varphi_1 \vee \diamond \varphi_2 \rightarrow \diamond \psi$	$\neg (\neg \diamond (\varphi_1 \vee \varphi_2) ; \diamond \psi)$

Abbildung 5: Semantik von Fehlerbäumen

Einige Bemerkungen zur Erklärung der Formeln: In den Formeln werden die Operatoren „ $\boxtimes \psi$ “, „ $\diamond \psi$ “ und „ $\diamond \psi$ “, verwendet. Diese bedeuten „in jedem Teilabschnitt des betrachteten Ablaufs gilt ψ “, „es gibt einen Teilabschnitt des Ablaufs, in dem ψ gilt“ und „es gibt ein Anfangsstück des Ablaufs, in dem ψ gilt“. Alle diese Operatoren lassen sich im Duration Calculus mit Hilfe des „;“-Operators (chop) als Abkürzungen definieren, z. B. $\diamond \psi \leftrightarrow \text{finite}; \psi$; true (wobei $\text{finite} \leftrightarrow \neg (\text{true}; \text{false})$). Die Formel „ $\varphi_1; \psi$ “ bedeutet: Es gibt eine Zerlegung des Ablaufs, so dass auf der ersten Hälfte φ_1 , auf der zweiten Hälfte ψ gilt.

Somit bedeutet z. B. die Vollständigkeitsbedingung für einen disjunktiven UW-Knoten: Es darf keinen Ablauf des Systems geben, der sich so in zwei Hälften teilen läßt, so dass in der ersten Hälfte nie eine der Ursachen wahr wird, bei dem aber zu Anfang der zweiten Hälfte die Wirkung eintritt (d.h. die Wirkung tritt auf, ohne dass zuvor die Ursachen wahr geworden wären).

Für die Korrektheitsbedingungen von UW-Knoten ist zu beachten, dass nur gefordert wird, dass es keine Abläufe geben darf, auf denen zwar die Ursache eintritt aber nie die Wirkung. Es kann nicht gefordert werden, dass *nach* dem Auftreten der Ursache die Wirkung eintritt, da die Wirkung, durch eine andere Ursache ausgelöst, auch schon vorher stattgefunden haben kann. Sie muss dann auch nicht nochmals auftreten: wenn z. B. durch eine nicht geschlossene Bahnschranke (die andere Ursache) eine Kollision (Wirkung) ausgelöst wird, und dabei die Bremse des Zugs zerstört wird (und somit die Ursache auftritt), muss danach nicht noch eine Kollision folgen.

Ein gelegentlich auftretendes Phänomen, sowohl bei UW-Knoten als auch bei Z-Knoten ist, dass die Ursachen nicht notwendigerweise die Wirkung implizieren bzw. die Menge der Fehlerzustände vergrößert wird. Für derartige Knoten gilt die Korrektheitsbedingung nicht. Sie treten häufig im Zusammenhang mit Designentscheidungen auf. Ein Beispiel eines derartigen

Z-Knotens ist etwa der oberste Knoten unserer FTA der Referenzfallstudie. Die Formalisierung von „Kollision“ im Toplevel-Knoten lautet:

Zug auf Bahnübergang \wedge Schranke offen

Dieser unerwünschte Zustand wird disjunktiv in zwei Fälle zerlegt, je nachdem, ob Freigabe vorliegt oder nicht. Im ersten Fall wird die Formel „Zug auf Bahnübergang“ weggelassen (und somit die Zustandsmenge vergrößert). Dies reflektiert die Entwurfs-Entscheidung, dass bereits die Zustände, in denen die Schranke bei gleichzeitiger Freigabe geöffnet ist, als fehlerhaft angesehen werden, egal ob sich der Zug nun zu diesem Zeitpunkt auf dem Bahnübergang befindet oder nicht.

Analog kann es auch für UW-Knoten vorkommen, dass das Vorliegen einer Ursache nicht garantiert zur Wirkung führt (in der Referenzfallstudie muss ein einmaliger Messfehler bei der Ist-Geschwindigkeit nicht zwangsläufig dazu führen, dass (später) die Sollgeschwindigkeit überschritten wird).

Für die Korrektheit lässt sich folgendes Theorem zeigen:

Theorem 1 *Korrektheitstheorem*

Enthält ein Fehlerbaum keine synchronen, konjunktiven UW-Knoten und keine konjunktiven Z-Knoten so gilt: Treten während eines Ablaufs alle Primary Events eines minimalen Cut Sets auf, so tritt auch der Top-Level Event auf.

Ein minimaler Cut Set wird dabei wie üblich (siehe z. B. (Vesely *et al.*, 1981)) dadurch berechnet, dass zunächst die Primary Events in den Blättern entsprechend der booleschen Verknüpfungen des Fehlerbaums verknüpft werden. Ein minimaler Cut Set besteht dann aus den Formeln einer Konjunktion aus der disjunktiven Normalform der erhaltenen Formel.

Das Theorem gilt nicht für synchrone, konjunktive UW-Knoten und auch nicht für konjunktive Z-Knoten mit Ursachen φ_1 und φ_2 , da auch das Vorliegen aller Primary Events für φ_1 und φ_2 nicht die Gleichzeitigkeit von φ_1 und φ_2 erzwingen kann.

Für die Vollständigkeit gilt ein stärkeres Theorem:

Theorem 2 *Vollständigkeitstheorem (auch Minimal-Cut-Set-Theorem)*

Können die Vollständigkeitsbedingungen aller Knoten eines Fehlerbaums gezeigt werden, so gilt: Kann für jeden minimalen Cut Set ausgeschlossen werden, dass dessen Ereignisse alle auf einem Pfad auftreten, so kann auch der Top-Level Event nicht auftreten.

Für einen vollständigen Fehlerbaum genügt es also, je einen Primary Event aus jedem minimalen Cut Set zu verhindern, um den untersuchten Fehler auszuschliessen. Eine FTA, bei der alle Knoten als vollständig nachgewiesen wurden, liefert somit einen Teilbeweis für die Sicherheit des Systems.

Das bewiesene Vollständigkeitstheorem gibt eine formale Rechtfertigung für die bei den Ingenieurdisziplinen übliche Verwendung von minimalen Cut Sets, auch für den Fall, in dem Zeitbedingungen eine Rolle spielen.

Aus der Literatur bekannt ist uns nur ein analoges Theorem in (Hansen *et al.*, 1994), das Zeitbedingungen berücksichtigt. Dort wird aber für die Vollständigkeit eines disjunktiven UW-Knotens nur

$$\square (\diamond \psi \rightarrow \diamond (\varphi_1 \vee \varphi_2))$$

gefordert. Da dies nicht garantiert, dass bei Ursachen und Wirkungen, die eine Dauer haben, der Beginn der Wirkung *frühestens mit dem Ende* der Ursache einsetzt, halten wir diese Bedingung aber für eine ungeeignete Formalisierung der Vollständigkeit (nehmen wir z. B. an, dass ein defektes Ventil eine mögliche Ursache für Überdruck ist und betrachten wir einen Ablauf, in dem sofort „50 Sekunden Überdruck“ und nach 30 Sekunden der Defekt des Ventils eintritt. Dann ist in diesem Ablauf das Ventil sicher *nicht* die Ursache, die obige Formel läßt sich aber trotzdem nachweisen). In der späteren Arbeit (Hansen, 1996) sind dann auch nur noch Z-Knoten zugelassen, deren Korrektheit und Vollständigkeit sich beweisen läßt, was die Formeln in den Zwischenknoten überflüssig macht (sie werden als boolesche Verknüpfungen der primary events definiert). Eine Fehleranalyse unserer Referenzfallstudie würde bei diesem Ansatz schon nach der ersten Zerlegung des Top Event steckenbleiben. Die vorher betrachtete Abschwächung um „Zug auf Bahnübergang“ könnte zwar noch durch einen Verfeinerungsknoten (refinement) realisiert werden (auch wenn dies methodisch fragwürdig ist, da keine Verfeinerung des Systemmodells vorliegt), aber UW-Knoten wären nicht möglich. Das in (Hansen et al., 1998) definierte Beispiel zeigt ausserdem, dass in diesem Ansatz Knoten, die keine Fehler beschreiben („Observation interval less than 30 sec“) nicht verboten sind.

UW-Knoten sind im Ansatz von (Bruns & Anderson, 1993) möglich. Dieser Ansatz gibt drei Semantiken: die einfache boolesche Semantik (in unserem Ansatz die Semantik von Z-Knoten) und zwei weitere, die CTL verwenden. Deren Idee stimmt mit unserer Korrektheits- und Vollständigkeitsbedingung überein. Ein globales Korrektheits- und Vollständigkeitstheorem wird aber nur für die einfache Semantik angegeben. (Górski, 1994) definiert einen rein prädikatenlogischen Ansatz zur Modellierung von Fehlerbäumen (mit expliziter Zeit). Eine Logik zur Verifikation wird nicht gegeben. Auch wird die Existenz einer Relation gefordert, die explizit Kausalbeziehungen zwischen Ereignissen beschreibt. Eine solche Relation ist bei einer üblichen Beschreibung von Zustandsübergangssystemen (wie etwa der Spezifikation von RefVL in Statemate) nicht vorhanden.

Zum Schluss wollen wir anmerken, dass sowohl das Korrektheits- als auch das Vollständigkeitstheorem für eine Formalisierung der Semantik des Duration Calculus und der Fehlerbäume in KIV (Reif, 1999) formal bewiesen wurden. Somit sind die semantischen Grundlagen für eine Integration von formalen Methoden und FTA gelegt.

6 Ausblick

Es wurde am Beispiel der Referenzfallstudie „Verkehrsleittechnik“ eine Sicherheitsanalyse durchgeführt. Diese Analyse zeigte, dass die bis jetzt in der Literatur vorgeschlagenen Semantiken für Fehlerbäume unzureichend sind. Aus diesen Erkenntnissen wurde eine formale Semantik für Fehlerbäume und ein Vorgehensmodell für die formale Sicherheitsanalyse entwickelt.

Alle bisher betrachteten Beispiele kommen mit den fünf verschiedenen Gattertypen, die in Abschnitt 5 beschrieben sind, aus. Weitere Fallstudien sollen klären, ob in anderen Fällen weitere Gattertypen benötigt werden. Für diese müssten dann eine formale Semantik angegeben werden und sowohl das *Korrektheits*- wie auch das *minimal-cut-set* Theorem gezeigt werden.

Durch die formale Semantik der Fehlerbäume ist weiterhin möglich, die Fehlerbaumanalyse zu automatisieren. Die bisher durchgeführten Beispiele lassen hoffen, dass zumindest eine (semi-) automatische Erstellung von Fehlerbäumen möglich ist. Diese wollen wir untersuchen

und entwickeln.

Die beschriebene Formalisierung wird noch für die Sicherheitsanalysetechnik FMEA nachgezogen und die Möglichkeit der Automatisierung untersucht.

References

- BAHR, N. J. 1997. *System Safety Engineering and Risk Assessment: A Practical Approach*. Taylor & Francis. Chap. Fault Tree Analysis.
- BRUNS, G., & ANDERSON, S. 1993. Validating Safety Models with Fault Trees. *Pages 21 – 30 of: GÓRSKI, J. (ed), SafeComp'93: 12th International Conference on Computer Safety, Reliability, and Security*. Springer-Verlag.
- CHAOCHEN, ZHOU, HOARE, C. A. R., & RAVN, ANDERS P. 1991. A calculus of durations. *Information Processing Letters*, **40**(5), 269–276.
- GÓRSKI, J. 1994. Extending Safety Analysis Techniques with Formal Semantics. *Pages 147 – 163 of: REDMIL, F. J., & ANDERSON, T. (eds), Technology and Assessment of Safety Critical Systems*. London: Springer Verlag.
- HANSEN, K. 1996 (August). *Linking Safety Analysis to Safety Requirements*. Ph.D. thesis, Danmarks Tekniske Universitet, Lyngby.
- HANSEN, K., RAVN, A., & STAVRIDOU, V. 1998. From Safety Analysis to Software Requirements. *IEEE Transactions on Software Engineering*, **24**(7), 573 – 584.
- HANSEN, K. M., RAVN, A. P., & STAVRIDOU, V. 1994. *From Safety Analysis to Formal Specification*. ProCoS II document [ID/DTH KMH 1/1]. Technical University of Denmark.
- HANSEN, M. R., & ZHOU CHAOCHEN. 1992. Semantics and Completeness of the Duration Calculus. *Pages 209–225 of: DE BAKKER, J. W., HUIZING, K., DE ROEVER, W.-P., & ROZENBERG, G. (eds), Real-Time: Theory in Practice*. Lecture Notes in Computer Science, vol. 600. Springer-Verlag.
- KLOSE, J., & THUMS, A. 2000. *Das Statemate-Referenzmodell der Referenzfallstudie Verkehrsleittechnik*. Tech. rept. Universität Oldenburg, Universität Augsburg. to appear.
- LEITSCH, R. D. 1995. *Reliability Analysis for Engineers: An Introduction*. Oxford Science Publications.
- LEVESON, N. 1995. *Safeware: System Safety and Computers*. Addison Wesley.
- REIF, W. 1999. Formale Methoden für sicherheitskritische Software – Der KIV-Ansatz. *Informatik – Forschung und Entwicklung*, **14**(3).
- STOREY, N. 1996. *Safety-Critical Computer Systems*. Addison-Wesley.
- VESELY, W. E., GOLDBERG, F. F., ROBERTS, N. H., & HAASL, D. F. 1981. *Fault Tree Handbook*. Washington, D.C. NUREG-0492.