Automatische Fehlerbaumerstellung und -analyse zeitinvarianter Netzwerke

Von der Fakultät Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart zur Erlangung der Würde eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von

Dipl.-Ing. Marius Riestenpatt genannt Richter

aus Stuttgart

Hauptberichter:	Priv.-Doz. Dr.-Ing. Stephan Rudolph
Mitberichter:	Prof. Dr. Stefan Leue
Mitberichter:	Prof. Dr.-Ing. Andreas Strohmayer

Tag der mündlichen Prüfung: 22.03.2021

Institut für Flugzeugbau
der Universität Stuttgart

2021

Vorwort

Diese Dissertation durfte ich als Mitglied der Arbeitsgruppe Entwurfstheorie und Ähnlichkeitsmechanik (π-Gruppe) erstellen. Bei deren zahlreichen, unterschiedlichen und auch in widrigen Zeiten motivierten Mitgliedern bedanke ich mich vielmals für die Unterstützung über die letzten Jahre.
Leiter dieser Arbeitsgruppe ist Priv.-Doz. Dr.-Ing. Stephan Rudolph, der dankenswerter Weise den Hauptbericht meiner Arbeit übernommen hat. Für seine fortwährende Unterstützung, sowie seine kontinuierlichen, vielen, neuen Ideen, seinen ansteckenden Enthusiasmus, das Bereitstellen eines tollen Arbeitsumfeldes und vieles mehr bedanke ich mich in höchstem Maße.
Prof. Dr. Stefan Leue danke ich vielmals für die Übernahme des Mitberichtes und die hilfreichen Anmerkungen, die mir neue, weitere, wissenschaftlich interessante Aspekte aufzeigten, die ich lernen und mit einbeziehen konnte. Für die Übernahme des Mitberichts und den damit einhergehenden Aufwand bedanke ich mich ebenfalls bei Prof. Dr.-Ing. Andreas Strohmayer.
Einen großen Dank spreche ich meinem Arbeitgeber IILS aus, der mir nach meiner anfänglichen Promotionszeit an der Universität Stuttgart im Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen (ISD) für die ich ebenfalls dankbar bin, die weitere Verfolgung meiner Dissertation ermöglichte.
Besonderer Dank gebührt Johannes Groß, der mir die Idee der Umsetzung der Domäne Zuverlässigkeit mit Entwurfssprachen und den ersten Kontakt mit Fehlerbäumen von einer Konferenz mitbrachte.
Zuletzt danke ich Familie und Freunden, die durch kontinuierliches Nachhaken konstant Druck ausübten und somit zur Vollendung der Arbeit beitrugen.

Inhaltsverzeichnis

Abkürzungsverzeichnis iv
Kurzfassung vii
Abstract ix

1 Einleitung **1**
1.1 Graphenbasierte Entwurfssprachen 2
1.2 Stellung der Zuverlässigkeitsanalyse im Produktentwurf 4
1.3 Problemstellung und Gliederung 6
1.4 Beiträge dieser Arbeit 7

2 Stand des Wissens **8**
2.1 Aktuelle Methoden der Zuverlässigkeitsanalyse 8
2.1.1 Fehlerbaumanalyse (FTA) 9
2.1.2 Fehlermöglichkeits- und -einflussanalyse (FMEA) 17
2.1.3 Weitere Methoden (HAZOP, RAMS, etc.) 19
2.2 Aspekte der Graphentheorie 21
2.2.1 Graphgrammatik 21
2.2.2 Graphtransformation 22
2.3 Offene Fragestellungen 23
2.3.1 Fehlerbäume und Kausalität 24
2.3.2 Funktionale Äquivalenz 26
2.3.3 Fehlerbaumanalyse in frühen Entwurfsphasen 27

3 Automatische Fehlerbaumerstellung und -analyse **29**
3.1 Graphenbasierte Netzwerkdarstellung 30
3.1.1 Netzwerksynthese 31
3.1.2 Zuverlässigkeitssystemmodellierung 31
3.2 Implementierung der automatischen Fehlerbaumanalyse 36
3.2.1 Fehlerbaumerstellung 44
3.2.2 Fehlerbaumanalyse 49
3.2.3 Brückensysteme 52
3.3 Einbindung in die Fehlermöglichkeits- und -einflussanalyse 55

3.3.1 Automatische Fehlermöglichkeits- und -einflussanalyse 56
3.3.2 Integrierte Fehlerbaumanalyse 58
3.3.3 Unterschiede zur Fehlermöglichkeits- und -einflussanalyse 58
3.4 Zusammenfassung und Zielsetzung 60
3.5 Weiterführende Fragen 61

4 Anwendungsbeispiele 62
4.1 Schiffsmotor 63
4.1.1 Modellierung 63
4.1.2 Fehlerbaumerstellung 71
4.1.3 Fehlermöglichkeits- und -einflussanalyse 73
4.1.4 Integration: Schiffsantriebssystem 76
4.2 Satellitenantriebssysteme 83
4.2.1 Modellierung 84
4.2.2 Fehlerbaumerstellung 85
4.2.3 Automatische Topologievariation 88
4.2.4 Fehlerbäume 93
4.3 Diskussion 100
4.3.1 Anwendungsbeispielvergleich 100
4.3.2 Vollständigkeit 101
4.3.3 Konsistenz 102

5 Zusammenfassung 103
5.1 Ergebnisse 103
5.2 Ausblick 105

Anhang 106

A Beispiel: Beleuchtungssystem 106
A.1 Modellierung 107
A.2 Fehlerbaumanalyse 110
A.3 Fehlermöglichkeits- und -einflussanalyse 111
A.4 Topologievariation 114

B Fehlerbäume 121
B.1 Schiffsmotor 122

B.2 Schiffsantriebssystem 125
B.3 Satellitenantriebssysteme 129

C Mathematische Ausführungen: Brückensystem 138

Literaturverzeichnis 142

Abkürzungsverzeichnis

43v2	Version 2 des Entwurfscompilers 43
CAD	rechnerunterstütztes Konstruieren (computer aided design)
CAS	Computeralgebrasystem
DfR	Design for Reliability
DOE	Design of experiments
DSL	domänenspezifische Sprache (domain-specific language)
FDIR	Fault-Detection, Fault-Isolation, and Recovery Techniques
FMEA	Fehlermöglichkeits- und -einflussanalyse (failure mode and effects analysis)
FMECA	Fehlermöglichkeits-, -einfluss- und -kritikalitätsanalyse (failure mode and effects and criticality analysis)
FTA	Fehlerbaumanalyse (fault tree analysis)
HAZOP	Hazard and Operability Study (englische Entsprechung von PAAG)
LPG	Lösungspfadgenerator (solution path generator)
M2M	Modell zu Modell Transformation (model-to-model transformation)
M2T	Modell zu Text Transformation (model-to-text transformation)
MDA	modellgetriebene Architektur (model driven architecture)
MDO	multidisziplinäre Entwurfsoptimierung (multidisciplinary design optimization)
MTBF	durchschnittliche Zeit zwischen Reparaturen bis zum erneuten Fehlerauftritt (mean time between failure)
MTTF	durchschnittliche Zeit bis zum Fehlerauftritt (mean time to failure)
PAAG	Prognose von Abweichungen, Auffinden der Ursachen, Abschätzen der Auswirkungen, Gegenmaßnahmen
RAMS	Zuverlässigkeit, Verfügbarkeit, Wartbarkeit und Sicherheit (reliability, availability, maintainability, and safety)
RBD	Zuverlässigkeitsblockdiagramm (reliability block diagram)
RPZ	Risikoprioritätszahl
SPG	Lösungspfadgenerator (solution path generator)
UML	Unified Modeling Language

Kurzfassung

Bereits in frühen Phasen eines Produktentwurfs werden wichtige Entscheidungen mit gravierendem Einfluss für den kompletten Produktlebenszyklus getroffen. Deshalb ist es in dieser zunehmend kürzer werdenden Phase von großer Wichtigkeit möglichst alle Aspekte des multidisziplinären Produktentwurfs in möglichst umfassender Detaillierung zu betrachten. Da frühe Entwurfsphasen durch viele Änderungen geprägt sind, ist die gewünschte Detaillierung manuell allerdings nicht in allen Domänen umsetzbar. Eine dieser Domänen ist bisher die Zuverlässigkeitsanalyse.

In der vorliegenden Arbeit wird eine Methode zur Zuverlässigkeitsanalyse über die vollautomatische Fehlerbaumanalyse (fault tree analysis (FTA)) von nahezu beliebigen Produktarchitekturen entwickelt. Im Gegensatz zur bisherigen sehr aufwendigen, in großen Teilen manuellen Fehlerbaumanalyse, die dadurch bedingt meist erst in späteren Entwurfsphasen in ausreichender Detaillierung angewendet wird, ermöglicht diese innovative Methode durch die Automatisierung - sowohl der Erstellung als auch der Auswertung - die Berücksichtigung der Fehlerbaumanalyse bereits in den frühen Entwurfsphasen.

Zur Umsetzung dieser vollautomatischen Fehlerbaumanalyse wird über graphenbasierte Entwurfssprachen ein vollständiges digitales Produktmodell erzeugt. Dieses Produktmodell beinhaltet u.a. die Definition der Topologie des Produktes, auf deren Grundlage dann die automatische Generierung des Fehlerbaumes sowie die vollständige Durchführung der Fehlerbaumanalysemethode erfolgt. Dafür wird das Modell semantisch hinsichtlich der gewünschten Fehlerbaumanalyse erweitert.

Im Rahmen dieser Arbeit zeigt sich, dass - unter Voraussetzung der Vollständigkeit des Produktmodells - die Fehlerbaumanalyse Teile der Fehlermöglichkeits- und -einflußanalyse (failure mode and effects analysis (FMEA)) beinhaltet. Wissenschaftlich bedeutet dies, dass die induktive Fehlermöglichkeits- und -einflußanalyse und die deduktive Fehlerbaumanalyse als zwei Sichten auf ein und das selbe Problem betrachtet werden können. Insbesondere aus einem zentralen Modell heraus kann somit die Bottom-up-Fehlermöglichkeits- und -einflußanalyse zusammen mit der Top-down-Fehlerbaumanalyse automatisch erzeugt und verknüpft werden.

Im Ablauf der automatischen Fehlerbaumanalyse wird zunächst ein Fehlerbaum erstellt. Diese automatische Fehlerbaumerstellung wird durch eine Zerlegung der semantisch annotierten Produkttopologie in Serien- und Parallelabschnitte erreicht. Der Fehlerbaum ermöglicht dann sowohl eine qualitative als auch eine quantitative Zuverlässigkeitsuntersuchung. Zur Abschätzung der Zuverlässigkeiten komplexer Topologien werden von diesen über eine Zerlegung serielle und parallele Ersatzsysteme gebildet, anhand derer eine Fehlerbaumerstellung und -analyse automatisch durchgeführt wird.

Durch die Automatisierung ist mit einer entsprechenden Entwurfssprache eine Optimierung des Systems mit Bezug auf Zuverlässigkeitsaspekte in frühen Entwurfsphasen automatisch möglich. Eine Optimierung wird in Form einer Heuristik in dieser Arbeit anhand eines Anwendungsbeispieles für den Entwurf von Satellitenantriebssystemen vorgestellt, wobei sowohl parametrische als auch topologische Änderungen des Systems automatisch durchgeführt werden, um eine Zielzuverlässigkeit zu erreichen.
Neben dieser gezielten Systemanpassung wird anhand des Anwendungsbeispieles für den Entwurf von Satellitenantriebssystemen eine Entwurfsraumuntersuchung vorgestellt. Die Entwurfsraumuntersuchung ermöglicht eine Darstellung des Zusammenhangs zwischen verschiedenen Entwurfsparametern für die verschiedenen parametrischen und topologischen Systemvarianten. Eine Analyse der Ergebnisse erlaubt Aussagen über die Auswirkung von Topologiewechseln auf die einzelnen Entwurfsparameter.

In einem weiteren Beispiel wird, neben der Illustration der Anwendbarkeit der automatischen Fehlerbaumerstellung und -analyse, anhand eines Schiffsmotors die Modularität der automatischen Fehlerbaumerstellung und -analyse für unterschiedlich komplexe Produkte und Prozesse demonstriert. Darüber hinaus erfolgt an diesem Beispiel eine Demonstration der Integration der automatischen Fehlerbaumanalyse in eine Fehlermöglichkeits- und -einflußanalyse.

Abstract

Important decisions with serious impact for the complete product life cycle are taken in early design phases of a product design. Considering that this design phases become increasingly shorter in current design, importance to consider preferably all key aspects of multi-disciplinary product design in preferably complete detail is very high. As early design phases include many design changes, in this design phases manual implementation in required detail is currently not viable in every domain. Nowadays, one of these domains is the reliability analysis.

In this thesis, a method is evolved to implement reliability analysis in early design phases. Therefor, a method is developed to fully automate the method of fault tree analysis (FTA) - a reliability analysis method - of any product architectures. Contrary to current high-effort in big extent manual fault tree analysis, due to that performed in late design phases in required detail, this innovative method enables a consideration of the fault tree analysis - creation as well as analysis - in early design phases using automation.

For implementation of the fully-automated fault tree analysis is generated a complete digital model of the product using graph-based design languages. This model of the product includes among others a definition of the topology of the product. Based on this topology a fault tree is automatically generated, and the fault tree analysis is automatically executed. Therefor, the product model is semantically extended towards the wanted fault tree analysis.

Besides automation of the method of fault tree analysis is investigated the link between the method of fault tree analysis and the method of failure mode and effects analysis (FMEA) - a risk analysis method. In this process is detected that a fault tree analysis - on condition that the product model is complete - includes parts of a failure mode and effects analysis. In scientific scope, the inductive failure mode and effects analysis and the deductive fault tree analysis are two views on a single problem. In particular, using a central digital model enables a simulatenously automatic creation of the bottom-up failure mode and effects analysis method and the top-down fault tree analysis method.

In the workflow of the automated method of the fault tree analysis is generated a fault tree first. This automated generation of a fault tree is accomplished by decomposition of

the semantically annotated product topology in serial and parallel sections. Afterwards, the automatically generated fault tree can be analyzed on reliability data qualitative and quantitative. Gaining reliability data of more complex topologies is handled by using serial and parallel spare systems for fault tree creation and analysis.

Automation of the reliability analysis facilitates by using design languages the optimization towards reliability in early design phases. An optimization in form of heuristics is illustrated on the basis of the application example of satellite propulsion systems design. Here, parametric as well as topologic design changes of the product model are executed automatically to reach a defined system reliability.
Beside targeted system change, the application example of satellite propulsion systems design is used to implement a design of experiments study. This design of experiments study enables an illustration of relation between different design parameters of different parametric and topologic system variants. Analyzing results, predications are possible for consequences of topology changes towards design parameters.

Another application example illustrates - in addition to the illustration of the automated fault tree creation and analysis - on the basis of a marine engine the modularity of the automated fault tree creation and analysis of differently complex products and processes. Moreover, this example demonstrates the integration of the automated fault tree analysis in the failure mode and effects analysis.

Kapitel 1

Einleitung

Produktlebenszyklen werden auf Grund sich schnell verändernden und verbessernden Technologien in einem kompetitiven Markt immer kürzer [118]. Diese Verkürzung hat zunehmend merklichen Einfluss auf den Entwurfsprozess eines neuen Produktes. Wichtige Entwurfsentscheidungen mit signifikanten Auswirkungen müssen im Entwurfsprozess in frühen Entwurfsphasen in immer kürzeren Zeitabschnitten getroffen werden.

Um diesem Problem bei gleichbleibender Qualität begegnen zu können, wird der Entwurfsprozess zunehmend mit Methoden der Digitalisierung unterstützt. Diese erlauben virtuelle Analysen des Produktentwurfs, aufbauend auf digitalen Modellen des Produktes. Dabei erfordern verschiedene Teilbereiche eines Produktes domänenspezifische, digitale Modelle, die eine entsprechende Untersuchung der Domäne ermöglichen. In Ab-

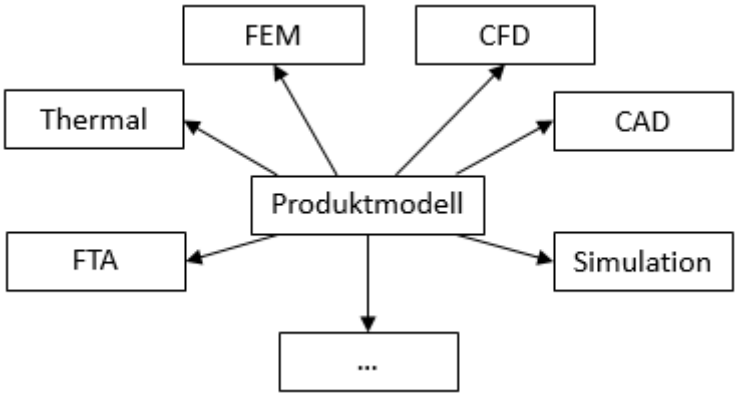

Abbildung 1.1: Ingenieurmodelle im Produktentwurf

bildung 1.1 sind für den Ingenieurentwurf eines Produktmodells verschiedene Domänen, die im Rahmen eines Produktentwurfs betrachtet werden, dargestellt. Die Erstellung der domänenspezifischen Modelle sowie deren Analyse bzw. Auswertung und Verwertung der Analyseergebnisse sind heutzutage weitestgehend manuelle Prozesse, die zeitintensiv und fehleranfällig hinsichtlich Konsistenz zu Modellen anderer Domänen sind.

Eine dieser Domänen ist die Zuverlässigkeitsanalyse, die die Berechnung von Zuverlässigkeitswerten und deren Einhaltung im Rahmen der Anforderungen zur Aufgabe hat. Die Automatisierung der konsistenten, zugehörigen Zuverlässigkeitsmodellerstellung und der Verwertung der Analyseergebnisse bietet Ansatzpunkte zur Verbesserung der Unterstützung des Entwurfsprozesses, die in dieser Arbeit aufgegriffen werden.

In diesem Kapitel werden zunächst graphenbasierte Entwurfssprachen vorgestellt, die zur Digitalisierung des Produktentwurfes beitragen und eine Lösung für die Konsistenzproblematik bieten. Danach wird die momentane Stellung der Zuverlässigkeitsanalyse im Produktentwurf dargestellt und eingeordnet. Abschließend wird die sich ergebende Problemstellung, der sich diese Arbeit widmet, erläutert sowie die Gliederung aufgezeigt.

1.1 Graphenbasierte Entwurfssprachen

Im multidisziplinären Produktentwurf sind unterschiedliche domänenspezifische Ansichten auf ein Produkt in Form spezifischer Modelle allgegenwärtig. Für die Erstellung dieser Modelle werden sowohl spezifische Eigenschaften der Domäne als auch sich mit anderen Domänen überschneidende Eigenschaften verwendet und auf Grund der sich ergebenden Analyseergebnisse verändert. Um der Konsistenzproblematik sich überschneidender und veränderlicher Eigenschaften zu begegnen, ist die Generierung der domänenspezifischen Modelle aus einer gemeinsamen Datenbasis, einem zentralen Produktmodell, heraus ein Ansatz zur Lösung.

Die Erstellung und Bearbeitung eines solchen zentralen Produktmodells statt einzelner Domänenmodelle ist für die Erhaltung eines konsistent korrekten Produktmodells für alle unterschiedlichen Domänen nur im Einklang mit allen Domänen möglich. Folglich benötigt dieser Prozess eine Gruppe aus Experten der einzelnen Domänen, um das zentrale Produktmodell mit entsprechendem Wissen konsistent korrekt hinsichtlich den einzelnen Domänen zu halten. Dies ist sowohl aus Kosten- als auch aus Zeitgründen im stetig komplexer werdenden Entwurfsprozess schwierig umsetzbar.

Eine mögliche Lösung besteht darin, die Erstellung eines ganzheitlich digitalen Produktmodells durch eine möglichst vollständige Automatisierung des digitalen Entwurfsprozesses zu unterstützen. Einen Ansatz für diese Entwurfsautomation stellt die Verwendung graphenbasierter Entwurfssprachen in Kombination mit einem Entwurfscompiler dar, wie dies in in Abbildung 1.2 dargestellt ist. Dabei werden insbesondere automatisierbare

Arbeitsschritte in der digitalen Modellerstellung durch modelliertes Entwurfswissen in einer Entwurfssprache abgebildet. Diese Abbildung zielt darauf ab, das Produktentwurfswissen ganzheitlich digital in einem zentralen Modell zu repräsentieren, aus dem heraus automatisch ausführbar digitale Produktmodelle entsprechend vorgegebener Randbedingungen generiert werden können.

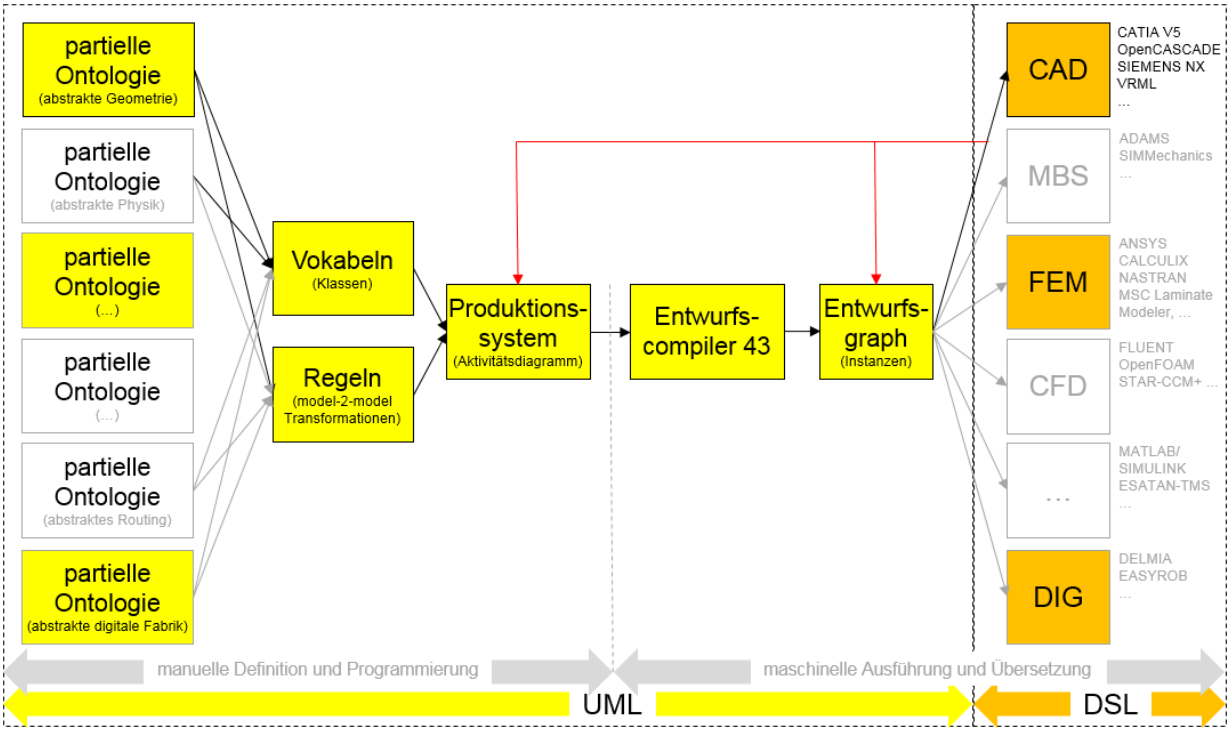

Abbildung 1.2: Informationsarchitektur graphenbasierter Entwurfssprachen mit zugehörigem Entwurfscompiler (hier: Design Compiler 43) nach [92]

Die Modellierung des Entwurfswissens in einer graphenbasierten Entwurfssprache orientiert sich an der Bildung einer natürlichen Sprache. Entsprechend besteht eine Entwurfssprache in den Grundbausteinen aus einem Wortschatz, den Vokabeln, und einer Grammatik, den Regeln, die die Vokabeln verwenden, in Beziehung zueinander setzen und Modelltransformationen (model-to-model (M2M)) vorgeben. Aufbauend auf diesen Grundbausteinen können Sätze erzeugt werden, die in einer Entwurfssprache einer Abfolge von Regeln entsprechen, einem sogenannten Produktionssystem. Aus einem solchen Produktionssystem wird durch die Ausführung mit einem Entwurfscompiler ein Entwurfsgraph erzeugt, der das Produkt, also den Inhalt, über Knoten und Kanten abstrakt abbildet. Das im Entwurfsgraphen repräsentierte Produkt ist in unterschiedlichen Domänen einzeln auspräg- und analysierbar, wie der Inhalt einer natürlichen Sprache unterschiedliche Interpretationsmöglichkeiten bietet.

Für die Erzeugung dieser spezifischen Modelle in den einzelnen Domänen werden jeweils einzelne partielle Ontologien erstellt, die in Form von Vokabeln und Regeln in eine spezifische Entwurfssprache eines Produktes wiederverwendbar integriert werden. Diese Integration partieller Ontologien erlaubt es, das Produktmodell um zugehörige domänenspezifische Eigenschaften zu erweitern. Aus einem entsprechend semantisch angereichertem zentralen Produktmodell, das sich im Entwurfsgraphen widerspiegelt, können zugehörige Modelle der einzelnen Domänen über Schnittstellen in textuell standardisierter Form der Zieldomäne bzw. der domänenspezifischen Sprache (domain specific Language (DSL)) des Zielprogramms generiert werden (model-to-text (M2T)). An einem ausgeprägtem Modell können in den Expertenprogrammen spezifische Untersuchungen für das Produkt durchgeführt werden. Neben der somit gegebenen Konsistenz der Domänenmodelle untereinander ermöglicht diese Herangehensweise eine frühe Einbindung unterschiedlicher Domänen im Produktentwurf, da eine analog konsistente Ausprägung in die einzelnen domänenspezifischen Modelle durch die automatische Ausführbarkeit schnell aus dem sich veränderndem zentralen Modell heraus ermöglicht wird.

Um eine möglichst unabhängige Wissensdarstellung der Entwurfssprachen zu gewährleisten ist eine standardisierte Basis für die Modellierung erforderlich. In der Umgebung des Entwurfscompiler 43 [55] wird deshalb auf die Unified Modelling Language (UML) und einzelne bereits vorhandene Diagrammtypen dieser zurückgegriffen, so dass beispielsweise Vokabeln in Klassendiagrammen und Produktionssysteme in Aktivitätsdiagrammen abgebildet werden. Diese allgemeine Informationsarchitektur graphenbasierter Entwurfssprachen mit zugehörigem Entwurfscompiler [92] zeigt Abbildung 1.2.

1.2 Stellung der Zuverlässigkeitsanalyse im Produktentwurf

Ein wichtiger Bestandteil für die domänenspezifischen Problemstellungen im multidisziplinären Produktentwurf ist die digitale Modellerstellung. Da die einzelnen, zu erstellenden Modelle meist Expertenmodelle sind, ist deren Erstellung und Pflege mit hohem hoch qualifiziertem, sprich teurem, wissensbasiertem Arbeitsaufwand verbunden. Dieser Arbeitsaufwand tritt insbesondere in frühen Entwurfsphasen durch die unvollständig gekoppelte Behandlung der spezifischen Problemstellungen, die zu vielen Anpassungen und Änderungen führen, in Form von hohem akkumuliertem Änderungsaufwand (Change Management) auf. Um diesen Änderungsaufwand gering zu halten, werden manche Problemstellungen mit geeigneten, ausreichend detaillierten Modellen erst spä-

ter im Produktentwurf berücksichtigt. Eine dieser Problemstellungen stellt häufig die Zuverlässigkeitsanalyse dar.

Die Stellung der Zuverlässigkeitsanalyse in späten Entwurfsphasen findet sich beispielsweise in [58] oder [96] wieder, die Möglichkeiten und Auswirkungen einer früheren Einbindung der Zuverlässigkeitsanalyse in den Entwurfsprozess aufzeigen. Das Ziel, die Zuverlässigkeitsanalyse als essentiellen Bestandteil in frühen Entwurfsphasen zu integrieren (Frontloading), hat sich allgemein in den letzten Jahren im aufstrebenden Ansatz ein System hinsichtlich der Zuverlässigkeit zu entwerfen (engl. Design for Reliability (DfR)), wozu in [76] ein aktueller Stand diskutiert wird, weiter verfestigt. In Abbildung 1.3 ist der Entwurfsprozess von komplexen Systemen nach [99] abgebildet, der als Zielvorstel-

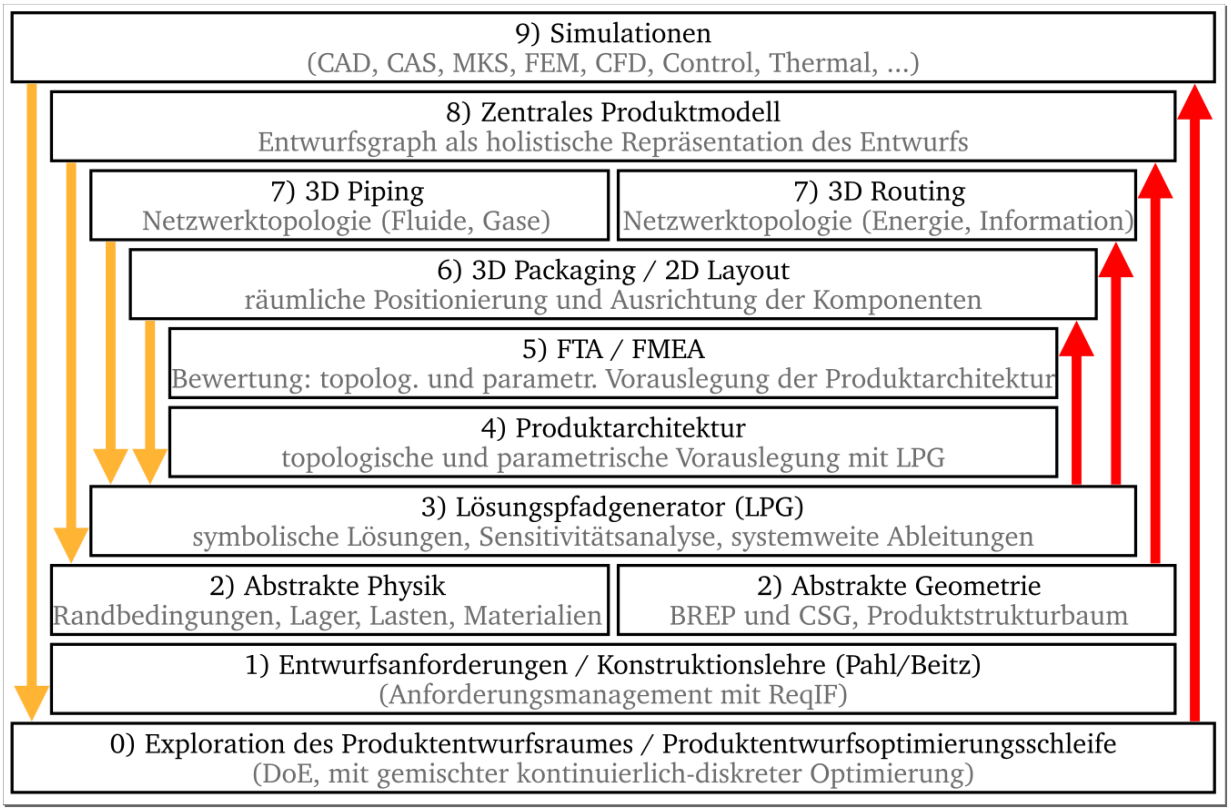

Abbildung 1.3: Entwurfsprozess von komplexen Systemen [99]

lung die Zuverlässigkeitsanalyse - in Form der Methoden Fehlerbaumanalyse (engl. fault tree analysis (FTA)) und Fehlermöglichkeits- und -einflussanalyse (engl. failure mode and effects analysis (FMEA)) - in einer zentralen Rolle zur Bewertung der topologischen und parametrischen Vorauslegung der Produktarchitektur beinhaltet.

1.3 Problemstellung und Gliederung

Die Zuverlässigkeitsanalyse als eine wichtige Domäne im Ingenieurentwurf wird, durch ihre starke Kopplung mit anderen Domänen sowie dem damit einhergehendem hohem Arbeitsaufwand zur Modellerstellung und -pflege in Kombination mit dem Verlust der Gültigkeit erstellter Modelle nach Änderungen am Produktentwurf, meist nicht von Beginn des Produktentwurfes an mit der wünschenswerten Tiefe betrachtet. Dies verhindert einen hinsichtlich der Zuverlässigkeitsanalyse gleichwertig entwickelten Produktentwurf, sowohl in der aktuellen Entwurfsphase als auch in Bezug auf die Gewichtung der Bedeutung der Zuverlässigkeitsanalyse des Entwurfs an sich, was sich in späteren Phasen des Produktlebenszyklus als bedeutend nachteilig herausstellen kann. Dieser Problemstellung wird im Rahmen dieser Arbeit mit einer Automatisierung über den Einsatz graphenbasierter Entwurfssprachen begegnet, die den Arbeitsaufwand der Zuverlässigkeitsanalyse auf ein Maß simultan zum Produktentwurf reduziert und somit eine entsprechend der Auswirkungen und Bedeutung sinnvolle frühzeitige Integration in den Entwurfsprozess ermöglicht.

Sowohl für die Automatisierung als auch die Zuverlässigkeitsanalyse an sich werden verschiedene Voraussetzungen angenommen. Eine dieser Voraussetzungen ist die Gültigkeit der in der Zuverlässigkeitsanalyse eingesetzten funktionalen Dekomposition. Diese funktionale Dekomposition setzt äquivalentes Verhalten analog zu anderen Domänen voraus. Ein anschauliches Beispiel aus der Domäne der Elektrik sind die Folgerungen aus den Kirchhoffschen Regeln zu Reihen- und Parallelschaltungen, welche eine Ersetzung von Widerständen mit funktional äquivalenten (Gesamt-)Widerständen erlauben, wie in Abbildung 1.4 dargestellt. Diese funktionale Äquivalenz wird auch in der Zuverlässigkeits-

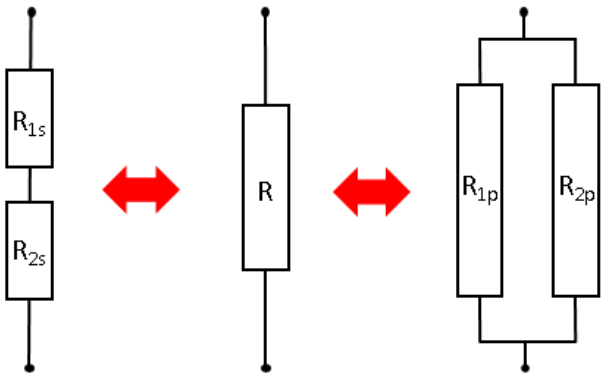

Abbildung 1.4: Funktionale Äquivalenz in der Elektrik

analyse verwendet, in der Systeme durch Kombinationen aus (Teil-)Serien- und Parallelsystemen beschrieben werden. Beschreibende Serien- und Parallelsysteme erhalten die funktionale Äquivalenz des Systems, sind topologisch aber verschieden. Dies bedeutet die Systeme sind in der Funktion ununterscheidbar, aber hinsichtlich der Betrachtung der Funktion unsichtbarer Kriterien bezüglich Zuverlässigkeit verschieden. Diese funktionale Äquivalenz wird bei einer automatischen topologischen Änderung des Systems, wie in dieser Arbeit, vorausgesetzt.

Für die Darstellung der Lösung der vorgestellten Problemstellung wird im folgenden Kapitel 2 der Stand des Wissens in der Zuverlässigkeitsanalyse aufgezeigt. In Kapitel 3 wird die Lösung über die automatische Fehlerbaumerstellung und -analyse dargestellt, die in Kapitel 4 anhand von Anwendungsbeispielen veranschaulicht wird. Im abschließenden Kapitel 5 folgt die Zusammenfassung der Arbeit.

1.4 Beiträge dieser Arbeit

In dieser Arbeit wird eine Methode vorgestellt, die es ermöglicht automatisch Fehlerbäume für sich parametrisch und topologisch verändernde Modelle zu erstellen. Darauf aufbauend wird gezeigt, wie eine automatische Fehlerbaumanalyse die Variation der Modelle unter Einbeziehung von Analyseergebnissen vornimmt und somit eine gezielte Entwurfsraumuntersuchung erlaubt. Die automatische Durchführung der iterativen Schritte der Methode der Fehlerbaumanalyse stellt ein Novum dar.
In der gedanklichen Vorstellung einer Darstellung als Schaltkreis als Beispiel eines Systems aus der Elektrik bedeutet dies, dass ein Schaltkreis bzw. der Teil eines Schaltkreises maschinell modifiziert wird. Im Zuge dieser maschinellen Modifikation bleibt die Funktion äquivalent erhalten, während die Parametrik und/oder Topologie des Systems automatisch verändert und analysiert wird. Dabei werden prinzipiell zwei unterschiedliche Strategien verfolgt. In der ersten parametrischen Strategie werden funktional identische Systemelemente hinsichtlich ihres Zuverlässigkeitswerts variiert, in der zweiten topologischen Strategie Systemelemente funktional äquivalent, redundant dupliziert.

Kapitel 2

Stand des Wissens

In diesem Kapitel wird der aktuelle Stand des Wissens für die Lösung der Problemstellung einer automatisierten Zuverlässigkeitsanalyse aufbereitet und skizziert. Zunächst wird auf verschiedene Methoden der Zuverlässigkeitsanalyse eingegangen, dann werden die für die Arbeit verwendeten Aspekte der Graphentheorie besprochen und zum Abschluss auf weitere offene Fragestellungen hinsichtlich Kausalität, funktionaler Äquivalenz und der Änderung der Stellung der Zuverlässigkeitsanalyse im Produktentwurf eingegangen. Dabei wird deutlich, dass die Bedeutung der Zuverlässigkeit für die Bewertung von alternativen Produktentwürfen und -architekturen immer weiter zunimmt.

2.1 Aktuelle Methoden der Zuverlässigkeitsanalyse

Im Produktentwurf werden verschiedene Methoden zur Zuverlässigkeitsanalyse angewendet, die unterschiedliche Vor- und Nachteile aufweisen. Die meist verbreiteten sind die *Fehlerbaumanalyse (fault tree analysis (FTA))* und die *Fehlermöglichkeits- und -einflussanalyse (failure mode and effects analysis (FMEA))*, die auch im Rahmen dieser Arbeit zum Einsatz kommen und deshalb näher betrachtet werden. Zusätzlich zur Beschreibung dieser Methoden werden weitere Methoden zur Zuverlässigkeitsanalyse kurz beschrieben. Einen guten ausführlicheren Überblick liefert in dieser Hinsicht beispielsweise das Buch Safeware von Leveson [67].
Leveson [67] geht dabei zusätzlich auf Probleme und Begrenzungen von Zuverlässigkeits- und Risikoanalysemethoden ein. Hierbei wird aufgeführt, dass Begrenzungen durch die ansteigende Interaktion verschiedener Domänen, speziell abseits rein technischer, zu neuen Anforderungen an die Analysemethoden führen. Diese voranschreitende Entwicklung wird beispielsweise durch die vermehrte Einbindung von Software oder in Bezug auf Management aufgezeigt.

2.1.1 Fehlerbaumanalyse (FTA)

Die Methode der *Fehlerbaumanalyse*[1] (engl. *fault tree analysis*) hat ihre Wurzeln im Bereich der Luft- und Raumfahrt bzw. der Reaktortechnik und wird vor allem bei komplexen, sicherheitskritischen Prozessen und Systemen eingesetzt. Erstmals kam die von Watson von den Bell Telephone Laboratories entwickelte Methode 1961 zur Untersuchung des Raketenabschusssystems für das Minuteman Missile Programm zum Einsatz [115]. Danach fand sie Einzug in verschiedene Industriebereiche und ist dort bis heute präsent. Einen detaillierteren historischen Abriss zur Methode der Fehlerbaumanalyse findet man beispielsweise bei Ericson [34].

Ziele der Methode der Fehlerbaumanalyse sind die Identifikation potenzieller Fehlerquellen sowie die Beurteilung und Optimierung der Sicherheit einer Funktion. Die Identifikation potentieller Fehlerquellen ist ein Teil der qualitativen Fehlerbaumanalyse[2], die systematisch alle möglichen Ausfallkombinationen, verstanden als diejenigen Ursachen, die zu einem vorgegebenen Ereignis, der Wirkung, führen, findet. Folglich handelt es sich bei der Fehlerbaumanalyse um eine deduktive Top-Down-Methode, die kausale Zusammenhänge beschreibt. Die kausalen Zusammenhänge der Ereignisse werden dabei in einem grafischen Systemmodell dargestellt, bei dem das unerwünschte Ereignis an einem Ende steht und die in Frage kommenden Fehlerquellen am gegenüberliegenden Ende zu finden sind. Dazwischen sind die Ereignisse über Boolesche Operatoren logisch und kausal verknüpft. Die Verknüpfungen der Ereignisse ermöglichen über die Ausfallwahrscheinlichkeiten der Basisereignisse die Bestimmung der Auftretenswahrscheinlichkeit des Wurzelereignisses. Diese quantitative Beurteilung bildet einen weiteren Aspekt der Methode der Fehlerbaumanalyse, auf der eine nachfolgende Optimierung des Systems erfolgen kann.

Die Methode der Fehlerbaumanalyse ist in verschiedenen Normen beschrieben, wie beispielsweise in DIN 25424-1/-2 [24] und DIN EN 61025 [27]. In diesen werden sowohl die Symbole des grafischen Modells erläutert, als auch Anwendungsschritte zur Durchführung der Methode beschrieben. Nach DIN 25424-1 sind im Rahmen einer Fehlerbaumanalyse beispielsweise folgende Schritte durchzuführen:

1 Es sind auch leichte Abweichungen des Namens der Methode zu finden, so wird in der Elektrotechnik von einer *Fehlzustandsbaumanalyse* gesprochen. [27]

2 Hier ist nicht die Methode gemeint, sondern die Analyse bzw. Auswertung als Schritt der Methode. Auf Grund dieser Namensproblematik mit dem Begriff der *Fehlerbaumanalyse* wird in der Arbeit wie folgt differenziert: Steht der Begriff *Fehlerbaumanalyse* für sich, ist die Methode gemeint. Mit dem Zusatz eines Adjektivs, wie qualitativ oder quantitativ, ist die Analyse des Fehlerbaumes gemeint, die Teil der genannten Methode ist.

1. Detaillierte Untersuchung des Systems mit Hilfe einer Systemanalyse.
2. Festlegung des „unerwünschten Ereignisses“ (Top-Ereignis, auch Wurzelereignis) und der Ausfallkriterien.
3. Festlegung der relevanten (zu untersuchenden) System-Zuverlässigkeitsgröße und der zu betrachtenden Zeitintervalle (Missionsdauer).
4. Erstellung des Fehlerbaums vom Top-Ereignis bis zu den Ausfallzuständen der Komponenten des Systems, die mit probabilistischen Größen beschrieben werden können. Sie bilden damit die Endzustände des Fehlerbaums (Top-down Analyse).
5. Bestimmung und Zusammenstellung der probabilistischen Größen der Komponenten des Systems (Ausfallraten, Nichtverfügbarkeiten) und deren Prüf- und Instandsetzungszeiten.
6. Auswertung bzw. Berechnung des Fehlerbaums.
7. Bewertung der Ergebnisse.

Diese Schritte gehen - ebenso wie die nach DIN EN 61025 - auf das Fault Tree Handbook von Vesely et al. [121] zurück. Insbesondere die nötige fundierte Systemkenntnis für die Durchführung einer effektiven Fehlerbaumanalyse führt dazu, dass es sich um eine Expertenmethode handelt.
Im Folgenden wird auf die für die Fehlerbaumerstellung notwendigen Fehlerbaumelemente sowie auf die Berechnungsgrundlagen von Fehlerbäumen im Detail eingegangen.

Fehlerbaum Die grafische Repräsentation der kausalen und logischen Zusammenhänge des Eintretens eines unerwünschten Ereignisses wird in einem Fehlerbaum dargestellt. Innerhalb dieser Baumstruktur sind verschiedene Elemente zur kausalen und logischen Verknüpfung und Darstellung gebräuchlich. Die für diese Arbeit wichtigsten Fehlerbaumelemente sind in Tabelle 2.1 dargestellt, weitere Elemente sind beispielsweise im Standardwerk Fault Tree Handbook von Vesely et al. [121] oder in den Normen beschrieben.

Prinzipiell wird zwischen zwei Gruppen von Fehlerbaumelementen unterschieden: Gatter und Ereignisse. Gatter dienen der kausalen und logischen Kopplung von Ereignissen, Ereignisse werden im Rahmen der Systemuntersuchung der Fehlerbaumanalyse definiert.
Auf Grundlage der Systemuntersuchung wird ein Fehlerbaum aufgebaut. Dessen Visualisierung erfolgt häufig in einer Baumdarstellung, in der das Wurzelereignis an der Spitze

Symbol	Name	Beschreibung
	Basisereignis	Eingangsereignis entsprechend der Systemgrenzenbeschreibung.
	Zwischenereignis	Ereignis, das aus einem Gatter hervorgeht.
	UND-Gatter	Ausgangsereignis tritt auf, wenn alle Eingangsereignisse auftreten.
	ODER-Gatter	Ausgangsereignis tritt auf, wenn mindestens ein Eingangsereignis auftritt.

Tabelle 2.1: Fehlerbaumelemente

abgebildet ist und die meist mehreren Basisereignissen am Boden zu finden sind.[3] In Abbildung 2.1 ist zur Illustration ein einfacher Fehlerbaum mit den grundlegenden Fehlerbaumelementen in einer solchen Baumdarstellung mit Wurzelereignis an der Spitze dargestellt. Anhand dieses Fehlerbaumes werden die einzelnen für diese Arbeit wichtigen Elemente im Folgenden erläutert.

Neben den hier vorgestellten statischen, zeitinvarianten Fehlerbäumen sind in den letzten Jahren viele Erweiterungen untersucht worden, die Zeitverhalten in einen Fehlerbaum integrieren. Solche Fehlerbäume werden als dynamische Fehlerbäume bezeichnet, die entsprechende zusätzliche Gatter und Ereignisse verwenden. In der Dissertation von Schilling [102] beschreibt dieser Unterschiede zwischen statischen und dynamischen Fehlerbäumen, die unter seinem temporalen Ansatz erweitert werden. Thums [109] Erläuterungen zur formalen Fehlerbaumanalyse beinhalten auch eine semantische Auseinandersetzung zu Fehlerbaumelementen und zur Erweiterung dieser für eine dynamische Fehlerbaumanalyse mit zeitvarianten Komponenten.

Gatter Bei den Gattern, auch Verknüpfungen oder Tore genannt, sind das Und- und das Oder-Gatter (AND und OR gates) die für diese Arbeit wichtigsten Gatter. Das Und-Gatter, das mit einem Schildelement mit gerader unterer Seite dargestellt wird, ermöglicht die Darstellung einer abhängigen Verknüpfung mehrerer Ereignisse innerhalb der Fehlerbaumdarstellung, die im gemeinsamen Zusammenspiel zu hierarchisch höheren Er-

3 Weitere Darstellungsformen, wie eine horizontale, sind in den Normen DIN 25424-1/-2 [24] und DIN EN 61025 [27] beschrieben.

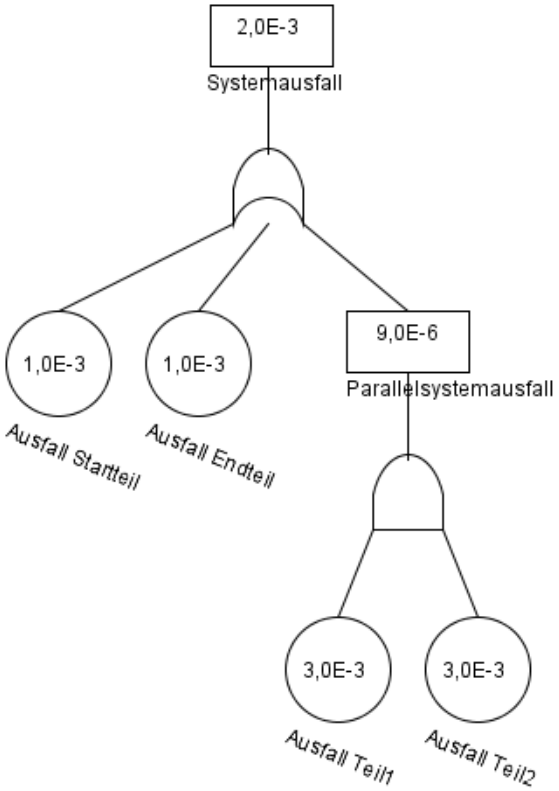

Abbildung 2.1: Einfacher Fehlerbaum

eignissen führen, also multikausal für dieses Ereignis sind. In Abbildung 2.1 ist unter dem Ereignis *Parallelsystemausfall* ein Und-Gatter dargestellt. Ein Oder-Gatter, das mit einem Schildelement mit geschwungener unterer Seite dargestellt wird, verknüpft Ereignisse, aus deren jeweiligen Einzelereignis ein gemeinsames, weiterführendes Ereignis kausal folgt. In Abbildung 2.1 ist unter dem Wurzelereignis *Systemausfall* ein Oder-Gatter zu sehen. Mit Hilfe dieser Gatter lässt sich ein Fehlerbaum mit boolescher Logik aufbauen. Weitere Gatter (k-aus-n-Gatter, Nicht-Gatter, Weder-Noch-Gatter, usw.) dienen größtenteils der übersichtlicheren Darstellung bei komplexeren Fehlerbäumen.

Ereignis Die Ereignisse, auch Events genannt, unterscheiden sich vornehmlich in ihrer Hierarchiestufe. Auf der obersten Hierarchiestufe, dem Beginn eines Fehlerbaumes, steht ein Wurzelereignis, auch Topereignis (top event) oder unerwünschtes Ereignis genannt, das die Grundlage des Fehlerbaumes bildet und welches je nach Darstellungsart ganz oben oder unten bzw. links oder rechts zu finden ist. Dieses Wurzelereignis ist ein Zwischenereignis (intermediate event), auch Kommentar oder Sekundärereignis genannt, wie die auf den Zwischenebenen folgenden Ereignisse, welche dort zur besseren Übersichtlichkeit und Darstellung von modularen Ereignissen dienen. Zwischenereignis-

se werden durch ein Rechteck dargestellt, wie in Abbildung 2.1 an dem Wurzelereignis *Systemausfall* und dem Zwischenereignis *Parallelsystemausfall* zu sehen ist. Auf der untersten Hierarchieebene eines Fehlerbaumes sind die Basisereignisse (basic events), auch Komponenten, Fehlermodi oder Primärereignisse genannt, zu finden, die dem Detailgrad des zu untersuchenden Fehlerbaumes entsprechend die kleinste zu betrachtende Einheit innerhalb des Analyserahmens repräsentieren. Diese werden durch Kreise dargestellt, wie in Abbildung 2.1 an den Ereignissen *Ausfall Startteil*, *Ausfall Endteil*, *Ausfall Teil1* und *Ausfall Teil2* zu sehen ist. Der Zahlenwert im grafischen Element eines Ereignisses gibt die Auftretenswahrscheinlichkeit dieses Ereignisses an. Abseits dieser Ereignisse sind weitere Ereignisse einheitlich definiert. Mit einem Transferereignis wird die Übersicht bei großen Modellen gewahrt bzw. dient dieses dazu um Schnittstellen zu anderen Bereichen und deren Fehlerbäumen aufzuzeigen. Darüber hinaus ist die Verwendung von nicht weiter zu entwickelnden Ereignissen, die die weitere Untersuchung von irrelevanten bzw. unwichtigen Zwischenereignissen angibt, üblich. Gerade diese Ereignisse können durch den Einsatz der automatischen Generierung der Fehlerbäume ohne großen Mehraufwand, in der maschinellen Verarbeitung, bis zu den Basisereignissen aufgelöst werden. Hierbei ist der Nutzen schwierig abzuschätzen, da diese Ereignisse meist nicht grundlos[4] nicht weiter untersucht werden und deren Einfluss als sehr gering einzuschätzen ist, der Fehlerbaum aber durch die weitere Detaillierung größer und somit unübersichtlicher wird. Für eine Berechnung bedeutet dies hingegen eine höhere Genauigkeit.

Auswertung Anhand eines Fehlerbaumes können sowohl quantitative als auch qualitative Analysen durchgeführt werden. Im Folgenden werden beide Analyseformen, insbesondere in Bezug auf die in dieser Arbeit eingesetzten Formeln und Analysewerkzeuge, erläutert. Dabei wird auf Kenngrößen, Schnitte und Pfade eingegangen und deren Einsatzbereich beschrieben. Weiterführende Analysewerkzeuge und Formeln, sowie detaillierte, mathematische Grundlagen sind beispielsweise im Fault Tree Handbook von Vesely et al. [121] zu finden.
Anmerkung: Die Analysen werden im Folgenden als Fehlerbaumanalysen bezeichnet. Dieser hier verwendete Begriff für die quantitative und qualitative *Fehlerbaumanalyse* ist allerdings vom Begriff der Methode der *Fehlerbaumanalyse*[5] zu unterscheiden.

4 Der Ingenieur verfügt häufig über Zusatzwissen, beispielsweise aus Erfahrung, das vorteilhaft zur „Abkürzung“ von algorithmischen Suchverfahren zum Einsatz kommen kann.

5 Die Verwendung des Begriffs der Fehlerbaumanalyse für die Methode ist gängig, aber es gibt auch Ausnahmen, wie beispielsweise bei Schneeweiss [101], der den Ausdruck der Fehlerbaum-Methode nutzt. Siehe dazu auch Fußnote 2 auf Seite 9.

Qualitative Fehlerbaumanalyse Die qualitative Fehlerbaumanalyse betrachtet die Zusammenhänge der einzelnen Ereignisse eines Fehlerbaumes, um kritische Systemstellen sowie in Bezug auf die Zuverlässigkeit irrelevante (Fehlerbaum-)Komponenten zu identifizieren. Im Rahmen dieser Identifikation werden entsprechende Ereignismengen mit Hilfe von Schnitten und Pfaden dargestellt.

Schnitte Ein Schnitt im Verständnis von Fehlerbäumen ist eine Menge an Ereignissen, dessen gleichzeitiges Auftreten der beinhalteten Ereignisse zum Auftreten des Wurzelereignisses führt.

Minimale Schnitte Minimale Schnitte beinhalten die Menge aller Ereignisse, die bei gleichzeitigem Auftreten zum Auftreten des Wurzelereignisses führt, aber keine Untermenge an Ereignissen beinhaltet, die zum Auftreten des Wurzelereignisses führt.

Pfade Eine Menge an Ereignissen, die bei gleichzeitigem Nichtauftreten das Auftreten des Wurzelereignisses ausschließt.

Minimale Pfade Minimale Pfade sind die Pfade, die keine anderen Pfade beinhalten, sondern die minimale Menge an Ereignissen beinhalten, um der Definition eines Pfades zu entsprechen.

Betrachtet man diese Definitionen, dann ermöglichen Schnitte das Auffinden kritischer Systemstellen, wohingegen mit Pfaden für das Modell irrelevante - im Bezug auf die Zuverlässigkeit - (Fehlerbaum-)Komponenten identifiziert werden können. Schnitte sind im Allgemeinen die für eine Fehlerbaumanalyse relevanten Mengen, insbesondere minimale Schnitte. Beinhaltet ein minimaler Schnitt exakt ein Ereignis, so handelt es sich um einen Einzelfehler (single point failure)[6], der nicht redundant ausgelegt ist. Ereignisse oder Ereignismengen, die in mehreren minimalen Schnitten auftreten sind sogenannte gleiche Ursachen (common cause failure).

Quantitative Fehlerbaumanalyse Bei der quantitativen Fehlerbaumanalyse werden den Basisereignissen eines Fehlerbaumes Wahrscheinlichkeiten über deren Auftreten, basierend auf Experimenten und statistischen Datenanalysen, zugewiesen, die über entsprechende Formeln nach der logischen Struktur des Fehlerbaumes verrechnet werden und somit die Ausfallwahrscheinlichkeiten für die Zwischenereignisse in den Fehlerbäumen liefern. Im Folgenden werden für die Arbeit wichtige Kenngrößen und Formeln für die quantitative Auswertung voneinander unabhängiger Ereignisse[7] beschrieben.

6 auch singuläres Punktereignis

7 Die Unabhängigkeit der Ereignisse in einer hierarchischen Ebene des Fehlerbaums ist die Grundlage der gängigen Fehlerbaumanalyseberechnungen. Darüber hinausführende Wahrscheinlichkeitsberechnungen mit unterschiedlichen Abhängigkeiten finden sich beispielsweise in [121].

Ausfallwahrscheinlichkeit und Zuverlässigkeit Basisereignissen wird für die qualitative Fehlerbaumanalyse eine Ausfallwahrscheinlichkeit F zugeordnet, die auf Grund von Testdaten und Erfahrungswerten ermittelt wurde. Die Zuverlässigkeit R wird entsprechend des folgenden Zusammenhangs zwischen Ausfallwahrscheinlichkeit und Zuverlässigkeit berechnet:

$$R(t) + F(t) = 1 \tag{2.1}$$

Eine Berechnung der Zuverlässigkeit bzw. Ausfallwahrscheinlichkeit eines Fehlerbaumes respektive seines Wurzelereignisse beginnt am gegenüberliegenden Ende des Baumes mit den Daten der Basisereignisse. Ausgehend von den Daten der Basisereignisse werden für die darüber stehenden Zwischenereignisse die Ausfallwahrscheinlichkeiten unter Verwendung ihrer logischen und kausalen Verknüpfung untereinander berechnet. Dieser Vorgang wiederholt sich bis die Ergebnisse von unten nach oben propagiert wurden und somit die Ausfallwahrscheinlichkeit des Wurzelereignisses bestimmt wurde.
Und-Gatter stellen eine logische und kausale Verknüpfung von voneinander unabhängigen Ereignissen dar, die multikausal für ein gemeinsames Folgeereignis sind, welches die Ausfallwahrscheinlichkeit als Produkt der Einzelausfallwahrscheinlichkeiten der zu Grunde liegenden Ereignisse berechnet:

$$F(t) = \prod^{i} F_i(t) \tag{2.2}$$

Oder-Gatter verknüpfen voneinander unabhängige Ereignisse, die monokausal für das selbe Folgeereignis sind, logisch und kausal. Die Ausfallwahrscheinlichkeit berechnet sich aus den Einzelausfallwahrscheinlichkeiten der zu Grunde liegenden Ereignisse nach folgender Gleichung:

$$F(t) = 1 - \prod^{i} (1 - F_i(t))^{8} \tag{2.3}$$

Importanz (Wichtigkeit) Die Wichtigkeit eines Ereignisses in Bezug auf das Auftreten des Wurzelereignisses eines Fehlerbaumes wird als Importanz bezeichnet. In der Literatur zur Zuverlässigkeit[9] sind verschiedene Definitionen von Importanzen zu finden. Im Rah-

8 Logisch bedeutet ein *Oder* eine Aufsummierung der Summanden. In Bezug auf die Wahrscheinlichkeitsrechnung wird der sich überschneidende Teil bei Unabhängigkeit der „Summanden“, der somit eine Dopplung darstellt, entfernt. Dies wird in der Fehlerbaumanalyse umgerechnet verwendet. Exkurs Wahrscheinlichkeitsrechnung: $F(t) = F_A(t) + F_B(t) - F_A(t) * F_{B|A}(t)$, da A unabhängig von B $F_{B|A}(t) = F_B(t)$ –> $F(t) = F_A(t) + F_B(t) - F_A(t) * F_B(t) = 1 - (1 - F_A(t)) * (1 - F_B(t))$

9 siehe beispielsweise [27]

men dieser Arbeit wird die Birnbaumsche Importanz [12] verwendet. Die Birnbaumsche Importanz wird über die Ableitung der Gleichung zur Berechnung der Ausfallwahrscheinlichkeit des Wurzelereignisses nach der Ausfallwahrscheinlichkeit des zu untersuchenden Ereignisses bestimmt.

$$I_i = dF/dF_i \tag{2.4}$$

I_i: Importanz des Ereignisses i; F: Gesamtauftretenswahrscheinlichkeit des Wurzelereignisses; F_i: Auftretenswahrscheinlichkeit des Ereignisses i.

Weitere Kenngrößen Neben den beschriebenen Ausfallwahrscheinlichkeiten und Importanzen werden für eine spezifischere quantitative Fehlerbaumanalyse weitere Kenngrößen basierend auf der Ausfallwahrscheinlichkeit berechnet. Beispielhaft seien hier die durchschnittliche Zeit zwischen Reparaturen bis zum erneuten Fehlerauftritt (mean time between failure (MTBF)) und die durchschnittliche Zeit bis zum Fehlerauftritt (mean time to failure (MTTF)) erwähnt, die für reparierbare Systeme bestimmt werden können. Im Rahmen dieser Arbeit wird auf diese Kenngrößen nicht weiter eingegangen, da es sich hierbei um reine Fehlerbaumauswertung aufbauend auf statistischen Daten handelt, was mit entsprechender Fehlerbaumsoftware bei einer hinreichenden Datenlage einfach zu bestimmen ist.[10]

Automatisierung Die Fehlerbaumanalyse wird im Bezug auf Automatisierung bisher eher stiefmütterlich behandelt. Zwar sind im Rahmen der Digitalisierung verschiedene Produkte auf dem Markt, die die Verwaltung von Fehlerbäumen ermöglichen und deren Erstellung grafisch unterstützen, sowie die beschriebenen Analysen ausführen können, doch ist die Erstellung dieser Fehlerbäume manueller Natur und bietet keine Konsistenz zu anderen Modellen des zu untersuchenden Systems. Freese [39] beschreibt eine Methode zur Erstellung von Fehlerbäumen und verweist darauf, dass zur Erstellung von Fehlerbäumen wenig veröffentlicht wurde und sich die meisten Publikationen auf die Analyse stützen. Die Analyse des Fehlerbaumes deckt aber nach dem eingangs beschriebenen Fehlerbaumanalyseablauf lediglich einen Teil der letzten Schritte einer Fehlerbaumanalyse ab. Vorhergehende Schritte zur automatischen Synthese von Fehlerbäumen aus einem Modell heraus sind im Bereich der Fehlerbaumanalyse ein weiterer kleinerer Teilbereich der in verschiedenen Arbeiten beschrieben wurde, vgl. hierzu Vemuri [113], Dugan [77] und Kocza [60]. Mit QuantUM [65] bzw. SpinCause [66] und xSAP [14] stehen auch Werkzeuge zur Modellprüfung zur Verfügung, die eine automatische Synthese und Fehlerbaumerstellung beinhalten. Mitunter werden auch dynamische Fehlerbäume erstellt

10 In der Literatur sind entsprechende Definitionen und Berechnungen beispielsweise in [29] zu finden.

und Zeitverhalten berücksichtigt sowie Zustandsautomaten verwendet. Allerdings dient der generierte Fehlerbaum häufig nur zur Ausgabe und wird teilweise direkt aus Analyseergebnissen, wie minimalen Schnitten, erstellt. Eine vollständige Automatisierung der Methode der Fehlerbaumanalyse ist jedoch in keiner dieser Arbeiten beschrieben.

2.1.2 Fehlermöglichkeits- und -einflussanalyse (FMEA)

Neben der Fehlerbaumanalyse ist die *Fehlermöglichkeits- und -einflussanalyse* (engl. *failure mode and effects analysis*)[11] die am weitesten verbreitete und eingesetzte Methode zur Risikoanalyse im Produktentwurf. Hervorgehend aus einer ersten Beschreibung der Fehlermöglichkeits-, -einfluss- und -kritikalitätsanalyse (engl. failure mode and effects and criticality analysis, Abkürzung: FMECA) von 1949 [111] wurde die FMEA parallel zur Fehlerbaumanalyse in den 1960er Jahren von der NASA entwickelt [73]. Eingesetzt wurde sie erstmalig beim Apolloprogramm. Über den Kraftwerksbau hat die Methode später breiten Eingang in die Automobilindustrie gefunden. Mittlerweile findet die FMEA Anwendung bei der Neuentwicklung von Erzeugnissen, dem Einsatz neuer Fertigungsverfahren, Produkten mit sicherheitstechnischen Anforderungen, Änderungen am Erzeugnis, Material oder Verfahren, Änderungen der Einsatzbedingungen bekannter Erzeugnisse, Beanstandungen und Forderungen durch den Kunden.
Im Gegensatz zur Fehlerbaumanalyse - die ein Vertreter der Top-Down-Methode ist - gehört die Fehlermöglichkeits- und -einflussanalyse zu den Bottom-Up-Methoden. FMEA und Fehlerbaumanalyse sind verbundene Instrumente, die sich gegenseitig ergänzen und in Kombination ihre größte Wirkung im Hinblick auf die Risikoidentifikation entfalten. Anstatt wie bei der Fehlerbaumanalyse zu untersuchen, welche Produktkomponenten eine gegebene Fehler- oder Risikosituation (Top-Event) hervorrufen könnten, versucht die FMEA herauszufinden, welche Art von Fehler oder Risiko durch die gegebenen Produktkomponenten(fehler) ausgelöst werden. Im Rahmen des Qualitätsmanagements wird die FMEA somit verwendet, um das entstehende Risiko durch das Auftreten von Fehlern zu minimieren. Dabei werden potentielle Fehler in Systemen, Konstruktionen und Prozessen analysiert und Maßnahmen definiert, um diese so früh wie möglich zu entdecken.

11 Daneben gibt es weitere gebräuchliche Bezeichnungen, wie Ausfalleffektanalyse [25] oder Fehlzustandsart- und -auswirkungsanalyse [26].

Die Fehlermöglichkeits- und -einflussanalyse ist in verschiedenen Normen, wie beispielsweise VDA Band 4, AIAG, TS 16949, DIN EN 60812, DIN EN 14971, DIN 25488 und DGQ Band 13-11, beschrieben und beinhaltet folgende Schritte:

1. Strukturanalyse zur Erfassung aller Systemelemente und der Systemstruktur
2. Funktionsanalyse zur Funktionszuordnung zu den Systemelementen und Verknüpfung dieser
3. Fehleranalyse zur Festlegung und Zuordnung von Fehlfunktionen zu den Funktionen und zur Verknüpfung dieser Fehlfunktionen
4. Maßnahmenanalyse / Bewertung
5. Optimierung

Die Struktur- und Funktionsanalyse stellen eine Systemanalyse dar, bevor mit der Fehleranalyse der eigentliche Ansatzpunkt einer FMEA beginnt. Die Ergebnisse aus diesen Analysen finden sich in Formblättern, wie beispielsweise in Abbildung 2.2, wieder. Im Rahmen der Bewertung werden verschiedene Kenngrößen verwendet, um das Risiko abzuschätzen, damit für die Optimierung relevante Maßnahmen identifiziert werden können. Diese Kenngrößen werden von Experten nach Zuordnung zu einer Kategorie gesetzt. Im Folgenden werden die Kenngrößen beschrieben.[12]

System: Performed by: Page: of
Ref. drawing no.: Date:

Description of unit			Description of Failure			Effect of Failure		Failure rate	Severity ranking	Risk reducing measures	Comments
Ref. No.	Function	Operational Mode	Failure mode	Failure Mechanisms	Detection of Failure	On Components in the subsystem	On the system function				

Abbildung 2.2: Formblatt einer FMEA

Bedeutung Die Bedeutung B der Fehlfunktion für die definierte Funktion. Dieser Wert ist entsprechend der FMEA Kategorisierung ganzzahlig aus dem Wertebereich 1 bis 10, in der zur groben Beschreibung 1 als keine Auswirkung und 10 als katastrophale Auswirkung angesehen werden kann.

12 In bestimmten Bereichen sind spezielle detailliertere Formeln für die Ermittlung der Kenngrößen üblich. Hier wird der Fokus auf die allgemeinen Kenngrößen der FMEA gelegt.

Entdeckungswahrscheinlichkeit Die Entdeckungswahrscheinlichkeit E der Fehlfunktion. Diese ist entsprechend der FMEA Kategorisierung ganzzahlig von 1 bis 10, wobei 1 als wird instantan entdeckt und 10 als keine Entdeckung als grobe Beschreibung dient.

Auftretenswahrscheinlichkeit Die Auftretenswahrscheinlichkeit bzw. die Ausfallwahrscheinlichkeitszahl (A) der Fehlfunktion, welche ganzzahlig von 1 bis 10 gesetzt wird. 1 steht hierbei für kein Ausfall und 10 demgegenüber für eine hohe Ausfallwahrscheinlichkeit in Bezug auf die geforderte Zuverlässigkeit des zu untersuchenden Systems.

Risikoprioritätszahl Zur Abschätzung des Risikos wird über die Bedeutung, die Entdeckungswahrscheinlichkeit und die Auftretenswahrscheinlichkeit die Risikoprioritätszahl (RPZ) als Produkt der Kennzahlen berechnet:

$$RPZ = A * B * E \tag{2.5}$$

RPZ: Risikoprioritätszahl, $\mathbb{Z} \in [1;1000]$; A: Auftretenswahrscheinlichkeit bzw. Ausfallwahrscheinlichkeitszahl, $\mathbb{Z} \in [1;10]$; B: Bedeutung, $\mathbb{Z} \in [1;10]$; E: Entdeckungswahrscheinlichkeit, $\mathbb{Z} \in [1;10]$

2.1.3 Weitere Methoden (HAZOP, RAMS, etc.)

Abseits der beschriebenen Fehlerbaumanalyse und Fehlermöglichkeits- und -einflussanalyse werden weitere Methoden im Produktentwurf im Bereich Zuverlässigkeit verwendet. Diese Methoden sind für speziellere Untersuchungen der Zuverlässigkeit mit dem Übertritt zu verwandten Gebieten der Sicherheit, Wartbarkeit und Verfügbarkeit von größerer Bedeutung und werden hier ohne Anspruch auf Vollständigkeit kurz beschrieben.

Hazard and Operability Study (HAZOP)[13]
Bei dieser aus der Chemieindustrie hervorgehenden Methode, die im Deutschsprachigen auch unter der Bezeichnung Prognose von Abweichungen, Auffinden der Ursachen, Abschätzen der Auswirkungen, Gegenmaßnahmen (PAAG) geläufig ist, werden von einer Expertengruppe Abweichungen vom Soll-Verhalten eines gegebenen Prozessablaufes betrachtet. Hierbei wird sowohl auf Gefährdungen und Gefahren als auch auf funktionelle Auswirkungen eingegangen. In der Expertengruppe werden mit sogenannten Leitworten in der Diskussion die entsprechend möglichen Abweichungen thematisiert und analysiert.

13 In manchen Beschreibungen der Methode sind leicht unterschiedliche Namensgebungen zu finden, wie beispielsweise in Levesons Safeware [67] mit Hazards and Operability Analysis.

Diese Analyse erfolgt rein qualitativ. Einen ausführlicheren Einblick in diese Risikoanalysemethode liefert Redmill et al.[85].

Zuverlässigkeit, Verfügbarkeit, Wartbarkeit und Sicherheit (RAMS)
Die RAMS-Methode (englisch: reliability, availability, maintainability, and safety) hat zum Ziel methodisch Systemabhängigkeiten zu untersuchen. Hierfür wird ein System über den kompletten Lebenszyklus zur Vorhersage des (zeitlichen) Ausfallverhaltens begleitet. Währenddessen wird auf Zuverlässigkeits- und Risikoanalysemethoden zurückgegriffen und deren Einsatz koordiniert. Häufig werden Zuverlässigkeit, Verfügbarkeit und Wartbarkeit (RAM) hierzu zusammen betrachtet. Die Methode hat unterschiedliche Erweiterungen für spezielle Systembetrachtungen. RAMSS fügt beispielsweise eine zusätzliche Sicherheitsbetrachtung (englisch: security) für äußere Fehler hinzu, RAMSAS [43] stellt einen modellbasierten, iterativen Ansatz mit Simulation dar.

Zuverlässigkeitsblockdiagramm (ZBD)
Zuverlässigkeitsblockdiagramme (englisch: reliability block diagram (RBD)) werden zur Zuverlässigkeits- und Verfügbarkeitsanalyse von Systemen verwendet. Beginnend mit einem Startknoten auf der linken Seite und endend auf der rechten Seite an einem Endknoten wird dazwischen der funktionale Zusammenhang des Systems mit Zuverlässigkeitsblöcken in seriellen und parallelen Abschnitten sowie über Kombinationen beider Formen abgebildet. Anhand eines solchen Zuverlässigkeitsblockdiagrammes und daraus ableitbarer Strukturfunktion kann sowohl eine qualitative als auch eine quantitative Zuverlässigkeitsanalyse des Systems durchgeführt werden. [28]
Die Methode ähnelt vom Aufbau und der zu Grunde liegenden Mathematik her Teilen der Fehlerbaumanalyse, die Unterschiede liegen im Ansatz und in den Zielen der Methoden. Eine Fehlerbaumanalyse untersucht ein fehlerhaftes Verhalten des Systems, ein Zuverlässigkeitsblockdiagramm untersucht ein erfolgreiches Verhalten des Systems, das durch eine einfache Überführung in einem Erfolgsbaum (success tree) abgebildet werden kann. Ziel der Fehlerbaumanalyse ist das Erreichen von Fehlerausfallzielen, Zuverlässigkeitsblockdiagramme ermöglichen eine Zuverlässigkeitsberechnung.

Neben den hier beschriebenen Methoden sind noch viele weitere Methoden im Bereich der Zuverlässigkeit- und Risikoanalyse[14] präsent. Beispielhaft seien hier die Markovanalyse, Zuverlässigkeitsketten, Petrinetze und Hazard Analysis and Risk Assessment erwähnt.

14 siehe auch [67]

2.2 Aspekte der Graphentheorie

Für die Modellierung der automatischen Zuverlässigkeitsanalyse kommen in dieser Arbeit graphenbasierte Entwurfssprachen zum Einsatz. In diesem Abschnitt werden die für die Entwurfssprachen wichtigen theoretischen Grundlagen der Graphentheorie beschrieben. Es wird zunächst auf Graphgrammatiken und danach auf Graphtransformationen eingegangen. Weiterführende Erläuterungen sind beispielsweise in Rozenbergs „Handbook of Graph Grammars and Computing by Graph Transformation“ [90] zu finden.

2.2.1 Graphgrammatik

Eine Graphgrammatik ist eine Vorschrift, die einen Graphen in einen anderen Graphen transformiert. Diese Vorschrift kann als eine Regel bzw. eine Menge an Regeln angesehen werden. Eine solche Regel besteht aus der sogenannten Linke-Hand-Seite (Left-Hand-Side, LHS), die einen Ausgangsgraph, hier auch als Suchgraph bezeichnet, beschreibt, und aus der sogenannten Rechte-Hand-Seite (Right-Hand-Side, RHS), die den zugehörigen Zielgraph beschreibt.
Zur Visualisierung von Graphgrammatiken sind verschiedene Lösungen verbreitet, wie beispielsweise das Ein-Quadranten-Schema, das Zwei-Quadranten-Schema und das Vier-Quadrantenschema, wie unter anderem von Göttler [45] eingehender beschrieben. In dem in dieser Arbeit verwendetem Entwurfscompiler 43 [55] werden Regeln im Zwei-Quadranten-Schema dargestellt, wie in Abbildung 2.3 zu sehen. Hierbei liegt eine grafische Trennung der Graphen der LHS und der RHS vor, aber ohne weitere grafische Trennung innerhalb der jeweiligen Seite wie beim Vier-Quadrantenschema [1].

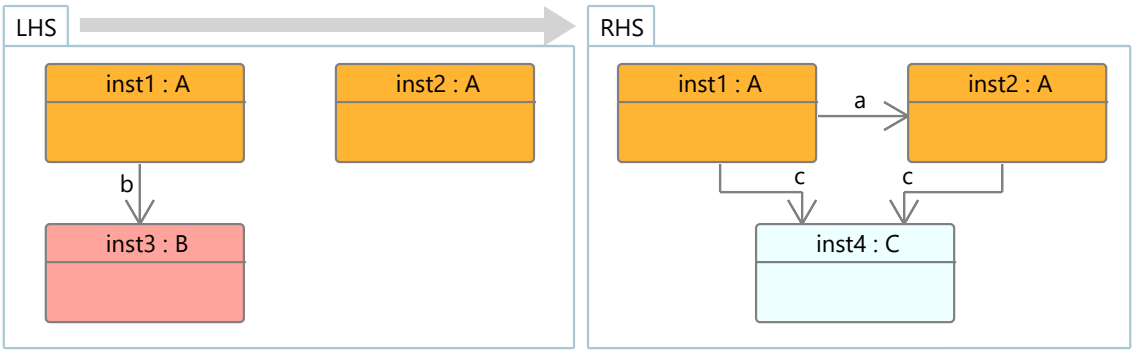

Abbildung 2.3: Grafische Regel im Zwei-Quadranten-Schema

2.2.2 Graphtransformation

Im Sinne der Philosophie der Model-Driven Architecture können Modelltransformationen auch über Graphtransformationen durchgeführt werden. In dieser Sichtweise ist jede Graphtransformation eine Modell-zu-Modell (model-to-model (M2M)) Transformation. Eine Graphtransformation verwendet dabei die Graphgrammatiken, um die Umformung eines Graphen in einen neuen validen Graphen zu beschreiben. Die Umformung erfolgt dann durch die Erkennung des zu Grunde liegenden Graphen der LHS im aktuellen Modellgraphen[15]. Dies wird als musterbasierte Suche (englisch: pattern matching)[16] bezeichnet. Dieser Graph, der LHS im Modellgraphen, wird durch die Transformation in den Graphen der RHS überführt, wodurch ein neuer Modellgraph erstellt wird, der ein neues Modell repräsentiert.
Aufbauend auf diesem theoretischen Grundgerüst über Graphtransformationen wird die im Entwurfscompiler 43 verwendete Anwendung der Graphtransformationen eingehender beschrieben. Dies geschieht in Bezug auf Abfolge und Determinismus, sowie auf die Durchführung der Transformation.
Das Ausführen von Graphtransformationen an sich hat keine festgelegte (zeitliche) Abfolge, wodurch eine entsprechende Anwendung von Graphtransformationen auf einen Modellgraphen zu nichtdeterministischen Ergebnisgraphen führen kann - je nach Reihenfolge und Abhängigkeit. In den in dieser Arbeit verwendeten Entwurfssprachen werden die Graphtransformationen über ein Produktionssystem geordnet und nacheinander ausgeführt, so dass eine nachfolgende Graphtransformation deterministisch auf den Ausgangsgraph der vorhergehenden Graphtransformation angewandt wird.
Eine Graphtransformation wird durchgeführt, sofern über die Prüfung der Isomorphie im Modellgraphen eine Anwendungsstelle ermittelt wurde. Dabei wird diese an einer theoretisch zufälligen Stelle[17] angewandt, für den Fall, dass sich im Modellgraphen mehrere mögliche Anwendungsstellen für die Graphtransformation befinden. Diese Vorgehensweise kann ebenfalls zu nichtdeterministischen Ergebnisgraphen führen, weswegen in dieser Arbeit eine Graphtransformation immer auf alle detektierten (Teil-)Graphen angewandt wird. Somit erfolgt gegebenenfalls eine mehrfache Ausführung einer Graphtransformati-

15 Der Modellgraph ist der Graph, auf den die Transformation angewandt wird.
16 Das Auffinden von Teilgraphen wird in der Graphentheorie als Isomorphieproblem bezeichnet. Dies spielt jedoch im Falle der hier verwendeten Implementierung im Entwurfscompiler 43 [55] keine Rolle, da dort die Knoten in der Regel mit eindeutigen Namen (Identifikation; englisch: Identifier) versehen sind, die das Auffinden eines speziellen Knoten im Graphen deutlich vereinfachen.
17 Diese hier als „zufällig“ bezeichnete Stelle ergibt sich aus der relativen Position des Graphenknotens innerhalb der zur Datenhaltung des Graphen verwendeten Datenstruktur.

on auf einen festgehaltenen[18] Graphen. Das ebenfalls existierende theoretische Problem einer Axiomgrapherstellung[19] ist dabei damit gelöst, dass angenommen wird, dass exakt ein „leerer“ Graph existiert.
Die Durchführung einer Graphtransformation wird durch das Neuschreiben des Graphen (englisch: graph rewriting) [74] vollzogen, weswegen Graphtransformationen in der Informatik auch Graphersetzungssysteme genannt werden. Hierbei wird der mit der Mustersuche gefundene Graph der LHS im Modellgraphen gelöscht und der Graph der RHS anschließend erstellt. Die Neuerstellung der Objekte kann zu Problemen führen, z.B. im Bezug auf die Verfolgbarkeit von Änderungen (englisch: traceability)[20], weshalb Graphtransformationen in dieser Arbeit, aber auch in anderen Graphtransformationsmethoden [36], analog zu den drei Typen von Modelltransformationen *erstellen*, *löschen* und *modifizieren* auf den vorhandenen Objekten bzw. Graphelementen durchgeführt werden. Ein dadurch entstehendes Problem stellen sogenannte baumelnde Kanten (englisch: tangling edges), Kanten die nach einer Graphtransformation nur noch einen Knoten haben und somit ungültig sind, wodurch der Graph invalide ist, dar. Diese werden deshalb mit einer impliziten (invaliden) Graphtransformation durch die Transformationsmaschine (englisch: engine) entfernt, um einen validen Graphen zu erhalten. Diese implizite Validierung des Graphen ist bei korrekter Modellierung nicht nötig, ist aber in der Anwendung nützlich.

2.3 Offene Fragestellungen

Bevor die Umsetzung der Lösung der Aufgabenstellung im nächsten Kapitel beschrieben wird, wird kurz auf verschiedene Fragestellungen zum besseren Verständnis eingegangen. Zunächst wird auf den Begriff der Kausalität in Zusammenhang mit Zuverlässigkeitsanalysemethoden, insbesondere der Fehlerbaumanalyse, zur Eingrenzung der Methode eingegangen. Des Weiteren wird die Fragestellung der funktionalen Äquivalenz in der Zuverlässigkeitsanalyse aufgegriffen. Zum Abschluss werden die Vor- und Nachtei-

18 Der Graph auf den die Graphtransformation angewendet wird, ist jeweils der selbe Graph, in den die anderen Ausführungen noch nicht berücksichtigt sind, sozusagen eine parallele Ausführung auf einen Graphen.

19 Der Suchgraph zur Erstellung eines ersten Graphen enthält keine Suchelemente. Da der Graph festgehalten wird, ist diese immer (unendlich häufig) erfüllt.

20 Die Verfolgbarkeit von Änderungen an einem Modell gewinnt durch die Verwendung eines zentralen Modells und die Automatisierung von Modellveränderungen zunehmend an Bedeutung. Die Rückverfolgbarkeit ermöglicht es, nachzuvollziehen was wann wo wie verändert wurde, was insbesondere bei mehreren domänenspezifischen Graphtransformationen wichtige Informationen darstellen kann.

le einer Verschiebung der Fehlerbaumanalyse in frühere Entwurfsphasen (front-loading) unter Einbezug der aktuellen Stellung im Produktentwurf aufgezeigt.

2.3.1 Fehlerbäume und Kausalität

Historisch ist der Begriff der Kausalität und die Existenz einer solchen ein umstrittenes Thema sowohl in der Philosophie, als auch in der Wissenschaft. Im Buch „Kausalität - Geschichte und Probleme“ von Bunge [18] findet sich hierfür eine umfassende Beschreibung. Eine Interpretation des Begriffs der Kausalität geht auf Hume zurück, der diese wie folgt beschreibt:

> „Dem entsprechend kann man die Ursache definieren, als einen Gegenstand, dem ein anderer folgt, und wo alle dem ersten ähnlichen Gegenstände, solche, die dem zweiten ähnlich sind, zur Folge haben. Oder mit anderen Worten: wo, wenn das erste Ding nicht gewesen wäre, das zweite niemals hätte entstehen können.“ Hume [54]

Aus Humes vermeintlich äquivalenter (zweiten) Aussage über das Nichtauftreten eines Ereignisses heraus begründete Lewis [68] die *kontrafaktische Kausalität*. Diese kontrafaktische Kausalität bezieht ihre Existenz durch eine Aussage über das Nichtauftreten eines Ereignisses in Folge eines vorangegangen Nichtauftreten eines Ereignisses. Durch diese Aussage ist es folglich nicht hinreichend hinsichtlich einer Kausalität, dass ein Wirkungsereignis einem Ursachenereignis nachfolgt, sondern das Konträre ebenfalls gelten muss, nämlich dass beim Ausbleiben des Ursachenereignisses kein Wirkungsereignis auftreten wird.
Halpern [49] beschreibt die kontrafaktische Kausalität als tatsächliche Kausalität (actual causality) gegenüber der aus Humes (ersten) Aussage hervorgegangenen Regularitätstheorie, die eine generelle Kausalität (type causality) beschreibt. Zusammen mit Pearl entwickelte Halpern für die kontrafaktische Kausalität verschiedene mathematischen Beschreibungen [51].[21] Hierbei verwenden sie auch Ansätze für multikausale Zusammenhänge, wie Mackies [71] INUS[22]-Bedingung und stellen das verwendete Kausalmodell[23] als wichtigen Faktor für die Ergebnisse einer kausalen Betrachtung heraus.

21 siehe auch [48] und [50]
22 INUS: an Insufficient but Non-redundant part of an Unnecessary but Sufficient condition (deutsch: ein nicht hinreichender, aber nicht vernachlässigbarer Teil einer nicht notwendigen, aber hinreichenden Bedingung)
23 Die Modellbildung spielt sowohl für die Kausalität als auch das für die Fehlerbaumerstellung zu Grunde liegende Systemmodell eine entscheidende Rolle. Für eine umfassende Modelltheorie mit sich ergebenden Einschränkungen sei dazu auch auf Stachowiaks *Allgemeine Modelltheorie* [105] verwiesen.

Betrachtet man die Fehlerbaumanalyse unter dem Aspekt der Kausalität, so ist das Ziel der Methode deduktiv Ursachen zu ermitteln, die zu einer - in diesem Fall - unerwünschten Wirkung in Form eines Ereignisses führen. Speziell unter dem naheliegenden Gesichtspunkt der kontrafaktischen Kausalität betrachtet, erlaubt dies eine Aussage darüber zu treffen, was passiert, wenn die Ursache ausbleibt, was mitunter eine implizite Zielstellung der Fehlerbaumanalyse in Form der Funktionalität ist. Das Ausbleiben der Ursache hat dementsprechend das Ausbleiben der Wirkung zur Folge, was wiederum die Aussage im Rahmen des Modells erlaubt, dass das unerwünschte Ereignis nicht eintritt. Daraus folgend muss zur Sicherstellung der Funktionalität eine Welt existieren, in der nicht alle Ursachen eintreten und das unerwünschte Ereignis nicht eintritt. Im Detail bedeutet dies, dass der Fehlerbaum kausale Zusammenhänge zwischen Ereignissen aufzeigt und in dem Ereignisse aus Ereignissen hervorgehen bzw. kausal auf diese folgen. Somit stellt dieser quasistatisch eine implizite zeitliche Abfolge der Ereignisse dar, da die Komponente der Zeit bereits in dem Fehlerbaummodell bzw. dessen Ereignissen beinhaltet ist.
Im Bezug auf kritische Stellen in der Theorie der kontrafaktischen Kausalität, liefert der Fehlerbaum ebenfalls interessante Erkenntnisse. Die Präemption[24] eines Wirkungsereignisses lässt im Sinne eines Fehlerbaumes auf eine Unvollständigkeit dieses schließen, da ein solches Ereignis in den Fehlerbaum integriert werden muss, um den Fehlerbaum zu vervollständigen. Dies ist insbesondere in der Identifikation der Wurzelereignisse ein wichtiges Hilfsmittel.
Ein weiterer beachtenswerter Punkt der kontrafaktischen Kausalität sind Epiphänomene. Epiphänomene sind Begleiterscheinungen bzw. -ereignisse, die mit einem Ursachenereignis auftreten, aber kein Wirkungsereignis darstellen.[25] Im Fehlerbaum sind solche Ereignisse auf Grund der Baumstruktur bei ausreichender Detaillierung ersichtlich, was im Gegenzug bedeutet, dass ein Fehlerbaum durch die Beachtung von Epiphänomenen eine entsprechende Modularisierung erhält, welche zur strukturierten Analyse beiträgt.
Zusammengefasst liefern Fehlerbäume durch Vollständigkeit und Detaillierung des zugehörigen Modells im Sinne der kontrafaktischen Kausalität eine Struktur, die neben Aussagen über das Auftreten eines unerwünschten Ereignisses implizit Aussagen über die Bedeutung eines Ausbleibens dieses Ereignisses erlaubt. Dabei ist die Qualität des Modells, das sich den Einschränkungen, wie von Stachowiak [105] in der Modelltheorie beschrieben, unterwirft, von essentieller Bedeutung.

24 Präemption bedeutet, dass ein Wirkungsereignis, das kausal mit einem Ursachenereignis verknüpft ist, mit einem weiteren Ursachenereignis kausal verknüpft ist.
25 siehe auch INUS-Bedingung [71]

2.3.2 Funktionale Äquivalenz

Die Zusammenhänge der Zuverlässigkeitsanalyse für ein Netzwerk können über eine Strukturfunktion beschrieben werden. In dieser Strukturfunktion wird das System funktional äquivalent in Serien- und Parallel(teil)systemen dargestellt. Eine solche funktionale Äquivalenz findet sich in verschiedenen Domänen wieder und wird hier analog für die Zuverlässigkeitsanalyse angewendet und als wichtige Arbeitshypothese angenommen.
Unter dem Aspekt der Strukturfunktion in der das Netzwerk dargestellt werden kann, ist eine Analogie zur mathematischen Äquivalenz von Funktionsausdrücken, wie in Abbildung 2.4 zu sehen, naheliegend. Einen entsprechenden speziell die Äquivalenz - bei nicht identischer Topologie - von Operator und Struktur betreffenden Zusammenhang bilden die in Teilbereichen zusammenfassbaren Serien- und Parallelsysteme, die in den in Kapitel 2.1.1 beschriebenen Formeln dargestellt werden können.[26]

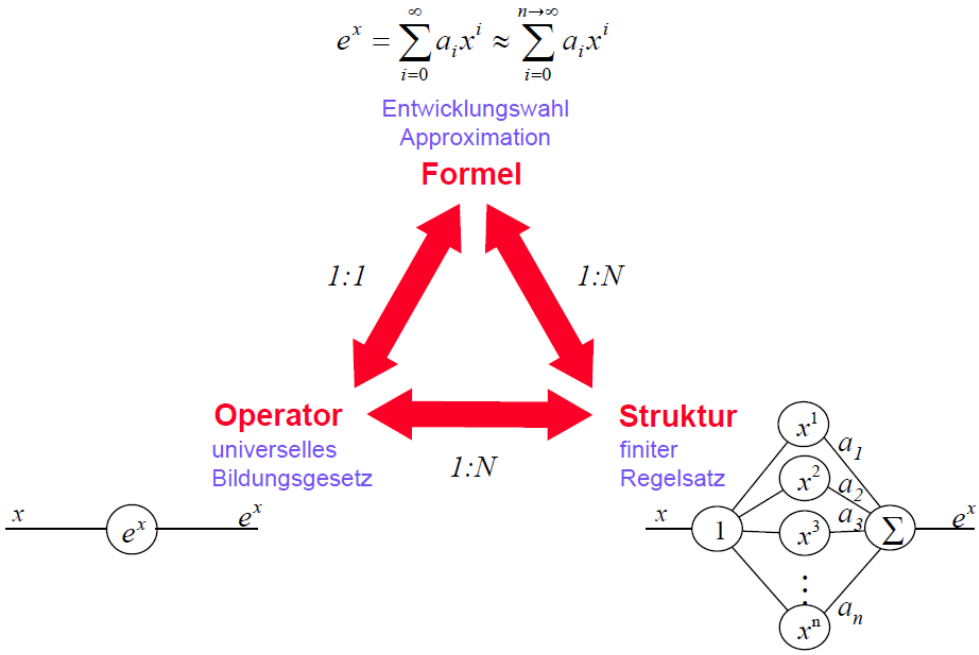

Abbildung 2.4: Mathematische Äquivalenz [93]

Ein bildlich ähnliches Beispiel sind die Folgerungen aus den Kirchhoffschen Regeln zu Reihen- und Parallelschaltungen in der Elektrik, welche eine Ersetzung von Widerständen mit funktional äquivalenten (Gesamt-)Widerständen erlauben. Diese funktionale Äquivalenz ist analog zur funktionalen Dekomposition, die in der Zuverlässigkeitsanalyse für Serien- und Parallelsysteme verwendet wird.

26 Für ein spezielles im nachfolgenden Kapitel 3.2.3 Brückensysteme vorgestelltes System wird im Anhang C beispielhaft die mathematische Berechnung der Äquivalenz gezeigt.

2.3.3 Fehlerbaumanalyse in frühen Entwurfsphasen

Billinton beschreibt die effektive Einbindung der Zuverlässigkeitsanalyse in den Entwurfsprozess, als zeitlich genau abgestimmten Arbeitsaufwand, der den Entwurfsprozess begleitet. So führt er aus:

> „The single most important factor that differentiates between effective and ineffective implementation of a reliability program is timing of the reliability effort. The reliability activity must proceed as an integral part of the development project. If not, it becomes either a purely academic function or historical documentation process. Reliability engineering often becomes a numbers game after the real game is over. Reliability cannot be economically added after the system or product has been conceived, designed, manufactured, and placed in operation.“ Billinton [10]

Insbesondere erwähnt er auch die Nachteile bzw. Ineffektivität einer zu späten Zuverlässigkeitsanalyse. Was bedeutet dies nun konkret für den Produktentwurf?
Im Produktentwurf werden zunächst konzeptionelle Entwürfe erstellt, die eine Lösung für das entworfene Problem als Antwort auf die vorgegebenen Anforderungen (requirements) und Randbedingungen darstellen. Diese Entwürfe sind anfangs nicht mit konkreten (Simulations-)Modellen hinterlegt, die für die virtuelle Absicherung der Funktionalität des tatsächlichen Produkts im Entwurfsprozess eine immer größer werdende Rolle spielen. Es stellt sich daher die Frage, ob zum Zeitpunkt der Modellerstellung in diesen Phasen es nicht bereits wünschenswert wäre eine Fehlerbaumanalyse durchzuführen.
Betrachtet man sich hierbei die Gründe, warum solche (Fehlerbaum-)Modelle nicht von Beginn an mit erstellt werden, dann kristallisiert sich als Antwort heraus, dass der Aufwand zur manuellen Erstellung und insbesondere das Pflegen eines (Fehlerbaum-)Modells eines sich ständig verändernden Systems als die ausschlaggebenden Gründe anzusehen sind. Wenn dieser Aufwand sich allerdings in Grenzen hält, bzw. automatisiert erfolgen kann, dann ist das Vorhandensein eines zum Zeitpunkt des Produktentwurfs vollständigen, konsistenten (Fehlerbaum-)Modells so früh als möglich wünschenswert.
Nachdem allgemein die Abbildung des Systems in Modellen zu möglichst frühen Entwurfsphasen als wünschenswert erachtet wird, stellt sich die Frage, wie sich dies im Spezialfall der Fehlerbaumanalyse als Repräsentant der Zuverlässigkeitsanalyse verhält. Hierbei verhält es sich nicht anders, als mit anderen Modellen. Der manuelle Aufwand zur Erstellung und die Pflege des Fehlerbaumes sind Hauptpunkte, warum eine detaillier-

te, manuelle Fehlerbaumanalyse meist erst in späteren Entwurfsphasen eingesetzt wird. Ergebnisse der Fehlerbaumanalyse können aber zu Topologieänderungen führen, die an sich möglichst früh im Entwurfsprozess erkannt werden sollten, um den Aufwand der Änderung im Entwurf möglichst gering zu halten[27]. Als Beispiel ist in Abbildung 2.5 der Änderungsaufwand in Form von Kosten über der Entwurfsphase, in der die Änderungsanforderung offenbar wurde, für den Softwareentwurf dargestellt.

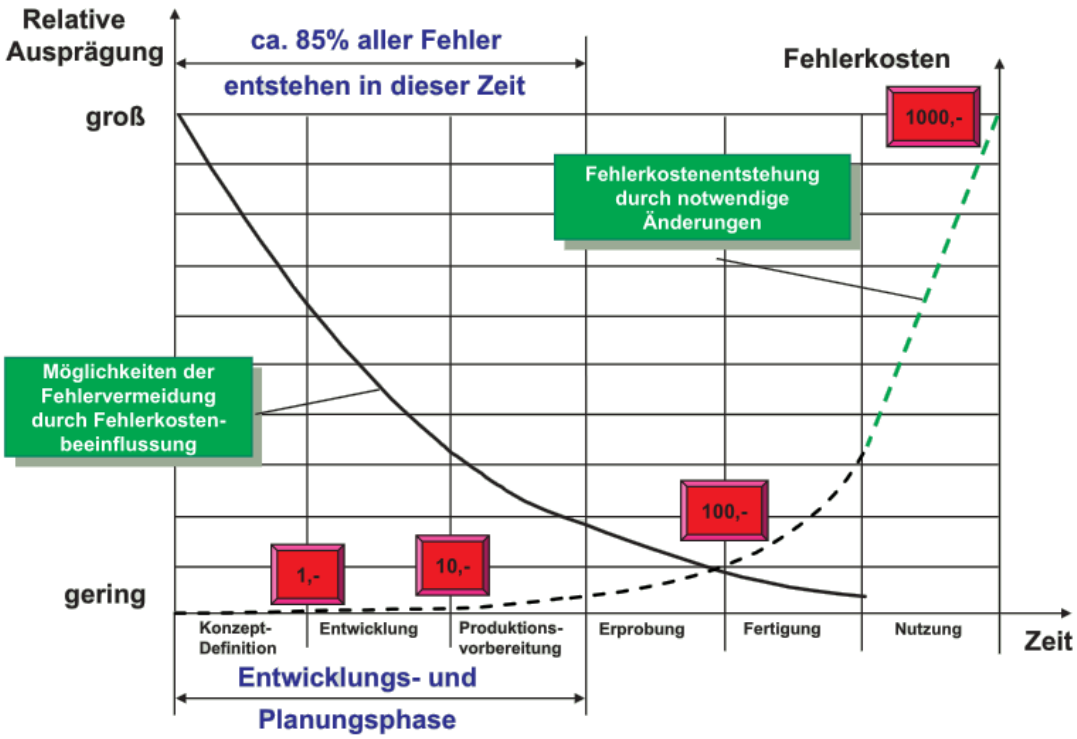

Abbildung 2.5: Änderungsaufwand über den Entwurfsphasen [22]

Neben den genannten Vorteilen die es bietet, wenn die Fehlerbaumanalyse in frühen Entwurfsphasen mit akzeptablem Aufwand durchgeführt werden kann, ist gerade diese Bedingung des akzeptablem Aufwandes ein Diskussionsgegenstand. Um einen klareren Eindruck von diesem zu bekommen, ist hier präzisiert der im Vergleich zum bisherigen Vorgehen einhergehende Mehraufwand auf den kompletten Entwurfsprozess gesehen gemeint, sofern einer besteht. Bei diesem Mehraufwand handelt es sich vornehmlich um die Abbildung des Expertenwissens im Modell, damit dieses korrekt ist und bleibt. In Kapitel 5 *Zusammenfassung* wird auf die beschriebenen Vor- und Nachteile nach der Automatisierung nochmals eingegangen.

27 Im Ingenieurentwurf geht man davon aus, dass die finanziellen Aufwendungen für Änderungen (design changes) umso höher ausfallen, je später sie vorgenommen werden.

Kapitel 3

Automatische Fehlerbaumerstellung und -analyse

Als Lösungsansatz für die in dieser Arbeit in Kapitel 1.3 aufgestellte Problemstellung, der Verbesserung der Anwendbarkeit der Zuverlässigkeitsanalyse im Produktentwurf, wird die Automatisierung einer Methodik der Zuverlässigkeitsanalyse untersucht. Diese Methode ist die im Kapitel 2.1.1 vorgestellte Fehlerbaumanalyse. Voraussetzung der Automatisierung der Fehlerbaumanalyse ist hierbei eine formalisierte Modellierung des zu untersuchenden Systems. Unter der Annahme eines solchen Systemmodells ist der prinzipielle Ablauf der Problemlösung im Folgenden beschrieben.
Zunächst wird das Systemmodell für die automatische Fehlerbaumanalyse entsprechend semantisch angereichert. Auf Grundlage dieses erweiterten Modells, erfolgt eine Auswertung hinsichtlich der Zuverlässigkeit, die das Modell mit direktem Bezug zur Methode der Fehlerbaumanalyse automatisch erweitert. Nach dieser automatischen Modellerweiterung ist eine automatische Erstellung eines zugehörigen Fehlerbaumes aus dem Modell möglich. Auf dem zu Grunde liegenden Fehlerbaum werden dann entsprechend der Fehlerbaumanalyse Daten zur Zuverlässigkeit des zu Grunde liegenden Systems ermittelt, die direkt in das Modell zurückgeschrieben werden.
In den folgenden Abschnitten wird auf die einzelnen Aspekte des beschriebenen Ablaufs der automatischen Fehlerbaumerstellung und -analyse eingegangen. Dafür werden zunächst die Voraussetzungen abgesteckt, die erforderlich sind, um die beschriebene automatische Fehlerbaumerstellung durchzuführen. Danach wird die Erweiterung des Modells für die automatische Fehlerbaumerstellung beschrieben, gefolgt von der darauf basierenden automatischen Fehlerbaumerstellung und der automatischen Fehlerbaumanalyse. Daneben wird die Integration in eine Risikoanalysemethode, die automatische Fehlermöglichkeits- und -einflussanalyse, aufgezeigt. Abschließend wird auf die Zielsetzung sowie weiterführende Fragen in der Diskussion des Lösungsansatzes eingegangen.

3.1 Graphenbasierte Netzwerkdarstellung

Die grundlegende Voraussetzung zur Automatisierung der Fehlerbaumanalyse ist das Vorhandensein einer Modellierung[1] des Systems. Diese Modellierung muss semantisch hinsichtlich der Fehlerbaumanalyse annotiert[2] sein und die funktionalen, kausalen Abhängigkeiten, die für eine Fehlerbaumanalyse relevant sind, der zu betrachtenden Modellelemente beinhalten. Die Abhängigkeiten müssen dabei einen zusammenhängenden unidirektional gerichteten Graphen ohne Kreisschlüsse aufspannen. Dieser aufgespannte Graph wird in dieser Arbeit als Netzwerk bzw. funktionsgekoppeltes System bezeichnet.
Um die semantische Erweiterung der Modellierung des Systems fundiert wissensbasiert durchzuführen, ist ein Domänenexperte im Bereich der Zuverlässigkeit erforderlich. Dies bedingt auch der Umstand, dass die semantische Annotation der Modellelemente mit für die Fehlerbaumerstellung relevanten Eigenschaften die ersten Schritte der Methode der Fehlerbaumanalyse[3] beinhaltet. Die entsprechenden Werkzeuge zur semantischen Annotation der Modellelemente werden in den nachfolgenden Abschnitten zur *Netzwerksynthese* und *Zuverlässigkeitssystemmodellierung* genauer erläutert.
Ein einfaches Beispiel, an dem direkt in der Modellierung ein funktionsgekoppeltes System erkannt werden kann, ist ein elektrischer Schaltkreis, dargestellt in Abbildung 3.1. Dieser elektrische Schaltkreis, der die Funktion hat eine Beleuchtung zu ermöglichen, wird in diesem Kapitel als begleitendes Beispiel verwendet, um die theoretischen Zusammenhänge praktisch und visuell zu verdeutlichen.

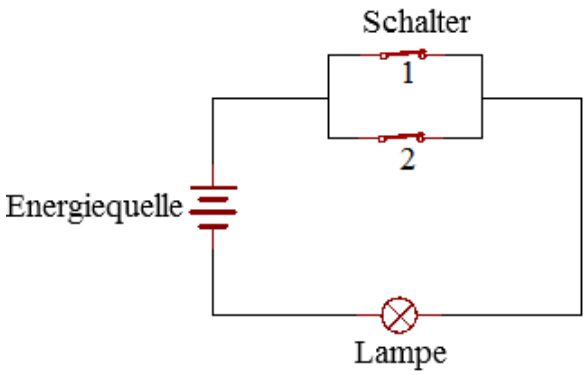

Abbildung 3.1: Netzwerk: Schaltkreis

1 Zur Modellbildung sowie den Einschränkungen eines Modells sei auf Stachowiak [105] verwiesen.
2 Semantische Annotation wird in grafischen Entwurfssprachen auf natürliche Art und Weise unterstützt.
3 siehe Kapitel 2.1.1

3.1.1 Netzwerksynthese

Aufbauend auf einem modellierten System, unter den zuvor beschriebenen Voraussetzungen, wird das für die Zuverlässigkeitsberechnung und nachfolgende Fehlerbaumerstellung sowie -analyse nötige Netzwerk in Form eines (partiellen) Graphen synthetisiert. Hierbei werden die hinsichtlich der Zuverlässigkeitsanalyse semantisch zu annotierenden Modellelemente[4] ermittelt. Prinzipiell handelt es sich um die Ermittlung des für die Zuverlässigkeitsanalyse relevanten (Teil-)Graphen aus dem (Gesamt-)Modell heraus, das neben für die Zuverlässigkeitsanalyse relevante Modellelemente weitere, beispielsweise für andere Analysen, relevante Modellelemente beinhalten kann. Dieser Expertenschritt erfolgt manuell und entspricht einer Identifikation der für die Zuverlässigkeitsanalyse relevanten Modellelemente und deren funktionale Zusammenhänge. Bezogen auf das in Abbildung 3.1 dargestellte Netzwerk ist in Abbildung 3.3a das zugehörige synthetisierte Netzwerk schematisch veranschaulicht.

3.1.2 Zuverlässigkeitssystemmodellierung

Das synthetisierte Netzwerk wird untersucht und regelbasiert semantisch erweitert, wie im nachfolgenden Abschnitt *Netzwerkuntersuchung* beschrieben. Auf Grundlage des semantisch annotierten Netzwerkes wird für eine automatische Zuverlässigkeitsberechnung das Modell automatisch hinsichtlich der Zuverlässigkeitsanalyse erweitert. Im Rahmen dieser Erweiterung wird zusätzlich zum existierenden Modell, ein Modell für die Zuverlässigkeitsberechnung erstellt. Dieses Modell enthält die Verknüpfungen zum ursprünglichen Modell und ist insofern eine Modellerweiterung.
Die regelbasierte Modellerweiterung wird im darauffolgenden Abschnitt *Netzwerkzerlegung* beschrieben. Die Grundlage für diese Modellerweiterung - im Sprachgebrauch der Entwurfssprachen das *Vokabular* - ist durch das erstellte Klassendiagramm für die Zuverlässigkeitsberechnung, zu sehen in Abbildung 3.2, gegeben. Die Berechnung von Zuverlässigkeitsdaten mit in diesem Klassendiagramm hinterlegten Gleichungen ist in Abschnitt *Netzwerkberechnung* beschrieben. Eine detaillierte Beschreibung des Klassendiagramms findet sich in Abschnitt 3.2.

4 Modellelemente sind Knoten und Kanten im Modellgraph.

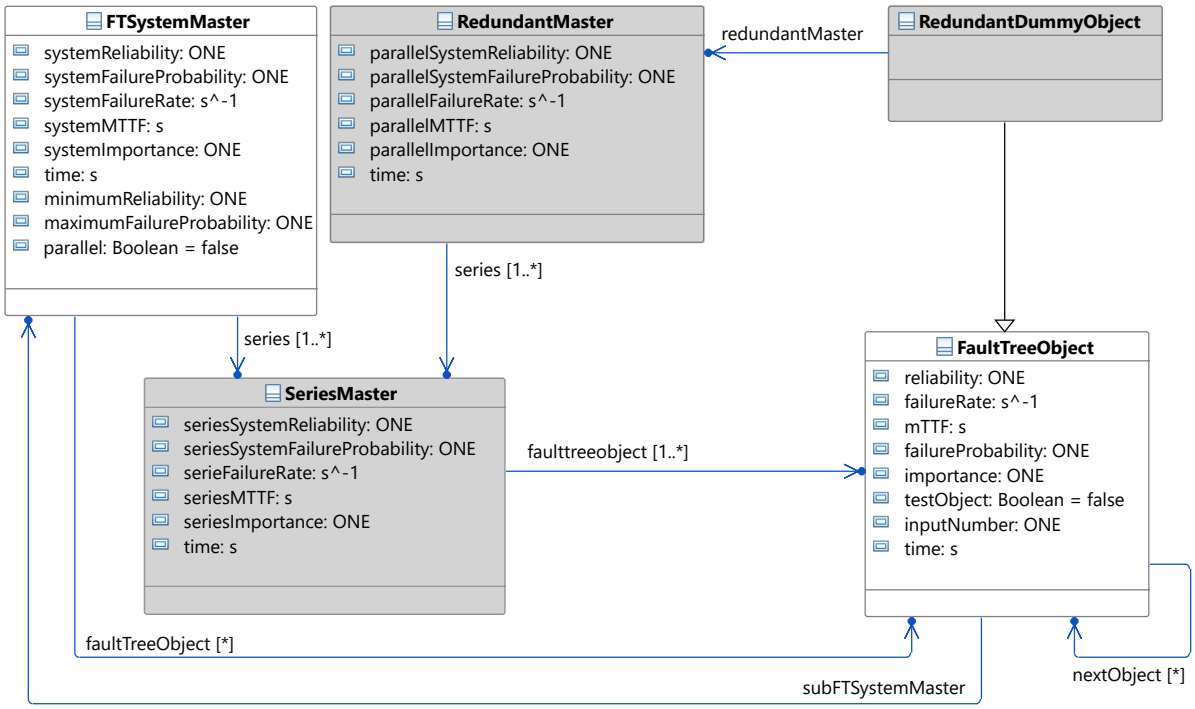

Abbildung 3.2: Klassendiagramm für die Zuverlässigkeitsberechnung in der UML[5]

3.1.2.1 Netzwerkuntersuchung

Das aus dem Modell heraus synthetisierte Netzwerk wird regelbasiert semantisch erweitert. Diese regelbasierte semantische Erweiterung besteht darin, dass zu existierenden Modellelementen Modellelemente aus dem Vokabular der Zuverlässigkeitsberechnung hinzugefügt und mit diesen verknüpft werden. Des Weiteren werden die funktionalen, kausalen Abhängigkeiten dieser Zuverlässigkeitsmodellelemente untereinander erstellt. Regelbasiert bedeutet in Bezug auf das Schaltkreisbeispiel, in dem jede Komponente des Schaltkreises auch ein Zuverlässigkeitsmodellelement ist, beispielsweise dass zwischen diesen Komponenten zusätzliche, in Regeln hinterlegte funktionale, kausale Verknüpfungen für die Zuverlässigkeitsanalyse erstellt werden. Diese Verknüpfungen entsprechen im Fall des Schaltkreisbeispiels dem extrahierten funktionalen Netzwerk.

5 Klassen mit weißem Hintergrund sind öffentliche Klassen (*public classes*), die z.B. von externen Entwurfssprachen verwendet werden können. Klassen mit hellgrauem Hintergrund sind private Klassen (*private classes*), auf die nur innerhalb des Projekts (hier: Plugin) zugegriffen werden kann.
Erläuterung zu Eigenschaften: <Bezeichnung>:<Datentyp/Einheit> [= <Standardwert>].
Erläuterung zu Pfeiltypen: Pfeil ohne Bezeichnung ist Vererbung, mit Bezeichnung ist Verknüpfung; <Bezeichnung>[<Multiplizität>]; Multiplizität: keine entspricht eins; * entspricht n; 1..* entspricht mindestens eins

3.1.2.2 Netzwerkzerlegung

Das semantisch annotierte Netzwerk wird über einen Algorithmus analysiert, welcher das Ziel hat den Modellgraphen für eine Zuverlässigkeitsanalyse zu erweitern, so dass das zu Grunde liegende Modell ein unveränderter Teil des Gesamtmodells mit entsprechender Funktionalität bleibt. Dieser Algorithmus 1 identifiziert serielle Minimalsysteme[6] und ersetzt mit Hilfe von Ersatzelementen parallele Systemteile, um diese zu seriellen Minimalsystemen zusammenzuführen. Dies erfolgt bis das komplette Netzwerk als serielles Minimalsystem mit Ersatzsystemen, die serielle Teilminimalsysteme darstellen, zerlegt wurde und dementsprechend als ein einziges serielles Minimalsystem dargestellt werden kann. Die Suche erfolgt hierbei generisch und kann bei verschiedenen Ausführungen des Algorithmus an verschiedenen Elementen beginnen. Erst im komplett im Algorithmus zerlegten und wieder zusammengeführten Netzwerk ist die Konsistenz bei gleichen Ausgangsmodellen bzw. Netzwerken wieder sichergestellt.
Die Visualisierung dieses theoretisch und mit dem Algorithmus 1 beschriebenen Ablaufes ist anhand des in Abbildung 3.3a schematisch extrahierten Netzwerkes des Schaltkreisbeispiels zu sehen. Zunächst erfolgt die Aufteilung des Netzwerkes aus Abbildung 3.3a, die in Abbildung 3.3b dargestellt ist. In derselben Abbildung 3.3b sind die inneren Seriensysteme des Netzwerkes zu sehen. Deren Zusammenführung zu einem Parallelsystem zeigt Abbildung 3.3c, in einer vereinfachten Repräsentation des aufgeteilten Gesamtnetzwerkes. Abschließend ist in Abbildung 3.3d die Zusammenführung aller Teilgraphen zu einem Gesamtnetzwerk illustriert.
Ein wichtiger zu erwähnender Punkt ist, dass der ursprüngliche Modellgraph in der im Algorithmus 1 vorgestellten Netzwerkzerlegung lediglich erweitert, aber das zu Grunde liegende Modell nicht verändert wird. Es wird sozusagen ein zusätzliches Modell erstellt, welches auf das ursprüngliche Modell verweist und aus dessen Struktur, wie im Algorithmus beschrieben, hervorgeht. Eine Modellveränderung findet lediglich innerhalb der Modellelemente der Zuverlässigkeitselemente im Rahmen der Modellgenerierung statt.[7]

6 Ein serieller Abschnitt im Modellgraphen, der keine Verzweigung bzw. keine nicht durch ein Ersatzobjekt ersetzte Verzweigung enthält.
7 Konkret: Verzweigung bekommt zusätzliche Verbindung zu Ersatzobjekt.

Algorithmus 1: Netzwerkzerlegung als Pseudocode

Eingabe: Graph $G(V,E)$

Ausgabe: transformierter Graph $G(V,E)$

relevanten (annotierten) Teilgraph ermitteln (Abbildung 3.3a)

Verzweigungen und Vereinigungen identifizieren (Mehrfachwegpunkte)

solange *nicht alle Verzweigungen auf Ersatzobjekte zeigen und diese von Seriensystemen assoziiert sind* **tue**

- Teilgraphen zwischen Verzweigungen auswerten
- **wenn** *keine Verzweigung in Teilgraph* **dann**
  - Seriensystem erstellen
  - Seriensystem mit Komponenten assoziieren (Abbildung 3.3b)
- **Ende**
- **wenn** *Verzweigung in Teilgraph und alle Elemente zwischen Verzweigungen von Seriensystemen assoziiert* **dann**
  - Parallelsystem erstellen
  - Parallelsystem mit Seriensystemen assoziieren (Abbildung 3.3b)
  - Ersatzobjekt für abgebildeten Teilgraphen der Zuverlässigkeitsanalyse erstellen
  - Ersatzobjekt mit Parallelsystem assoziieren
  - Assoziationen auf und von Ersatzobjekt zu Verzweigungen erstellen (Abbildung 3.3c)
- **Ende**
- **wenn** *keine Verzweigung in Teilgraph oder nur Verzweigungen, die auf Ersatzobjekte zeigen, und Start- und Endelement nicht in Teilgraph* **dann**
  - Seriensystem erstellen
  - Seriensystem mit Komponente(n) und Ersatzobjekt(en) assoziieren (Abbildung 3.3c)
- **Ende**
- **wenn** *keine Verzweigung in Teilgraph oder nur Verzweigungen, die auf Ersatzobjekte zeigen, und Start- und Endelement in Teilgraph* **dann**
  - Seriensystem erstellen
  - Seriensystem mit Komponente(n) und Ersatzobjekt(en) assoziieren
  - Fehlerbaumhauptinstanz erstellen
  - Fehlerbaumhauptinstanz mit Seriensystem assoziieren (Abbildung 3.3d)
- **Ende**

Ende

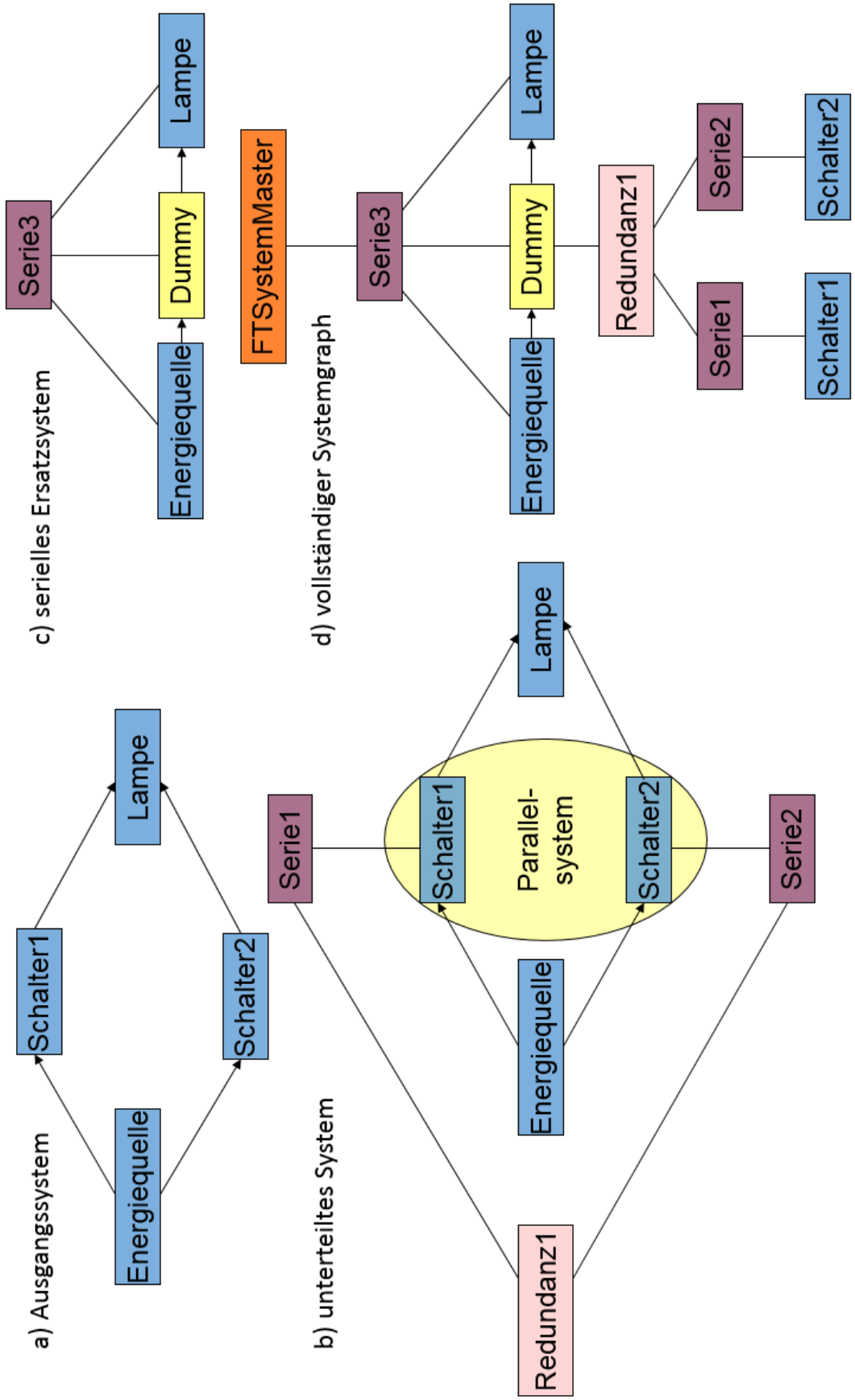

Abbildung 3.3: Netzwerkzerlegung

3.1.2.3 Netzwerkberechnung

Für die Vergleichbarkeit unterschiedlicher Parametriken oder Topologien wird in dieser Arbeit eine Netzwerberechnung mit der Zuverlässigkeit als Bewertungsgröße durchgeführt. Die Bewertungsgröße der Zuverlässigkeit eines Netzwerkes wird hierbei über die Strukturfunktion mit Hilfe einer Fehlerbaumanalyse berechnet, die im folgenden Abschnitt 3.2 *Implementierung der automatischen Fehlerbaumanalyse* beschrieben wird.
Auf Grundlage der errechenbaren Zuverlässigkeitsdaten ist unter Vorgabe von Randbedingungen eine automatische - gegebenenfalls mit iterativen Methoden unterstützte - Erstellung eines die Randbedingungen erfüllenden Systems möglich. Diese Fortführung der Berechnungsergebnisse ist, durch die Vorgabe von Regeln auf Grundlage von Berechnungsergebnissen in verschiedenen Teilaspekten oder auch im Gesamtaspekt von Netzwerken, anwendbar. So kann eine geforderte Erhöhung der Zuverlässigkeit durch den Einsatz - in Bezug auf die Zuverlässigkeit - besserer Komponenten genauso wie mit der Einführung zusätzlicher, funktional äquivalent redundanter Stränge erreicht werden. Setzt man beispielsweise hohe Zuverlässigkeit von Einzelkomponenten mit einem hohen Preis in einen korrelativen Zusammenhang, kann durch eine Optimierung auf ein Zuverlässigkeitsziel hin mit der Wahl unzuverlässigerer Komponenten ein niedrigerer Preis erreicht werden. Diese Netzwerkberechnung bzw. -anpassung an verschiedene Randbedingungen erfordert zum sinnvollen Einsatz einen hohen Grad an Wissen über das Produkt und die verschiedenen Ausführungskombinationen, die zwar mit Sicht auf Zuverlässigkeit und Kosten eventuell abgedeckt werden kann, aber bei denen auf weitere Aspekte im Rahmen des Produktentwurfes eingegangen werden muss. So sind beispielsweise Kopplungsprobleme, sowie konstruktive Probleme, neben dem Optimierungsproblem zu betrachten und in die regelbasierte Modellbildung mit einzubinden, um entsprechende, repräsentative Topologievariationen zu ermöglichen.

3.2 Implementierung der automatischen Fehlerbaumanalyse

Die ersten Schritte einer Fehlerbaumanalyse werden mit der Systemmodellierung eines graphenbasierten Netzwerkes manuell ausgeführt. Auf Grundlage dieser erweiterten Systemmodellierung wird das System automatisch im Sinne der Zuverlässigkeit analysiert und berechnet. Zusätzlich wird automatisch ein zugehöriger Fehlerbaum erstellt und analysiert. Die letzten iterativen Schritte der Fehlerbaumanalyse über zu treffende Maßnahmen, die zur Systemanpassung führen, können automatisiert ausgeführt werden.

Die ersten Schritte der Fehlerbaumanalyse stellen eine Systemanalyse dar, die ein Vokabular, für die semantische Erweiterung des Modells mit den für die Zerlegung im Sinne der Zuverlässigkeit nötigen Elementen benötigt. Dieses Vokabular ist in einem Klassendiagramm der Unified Modeling Language (UML) in Abbildung 3.2 dargestellt.[8] Im Folgenden wird das Klassendiagramm im Detail beschrieben.

Klassendiagramm zur Systemanalyse

In Abbildung 3.2 ist das Klassendiagramm als Repräsentation in der UML und damit das Vokabular für die Erweiterung eines Modells zur Zuverlässigkeitsberechnung mit Hilfe von Fehlerbäumen zu sehen. Das Klassendiagramm besteht aus fünf Klassen, die nachfolgend detaillierter beschrieben werden. Von diesen fünf Klassen ist für den Nutzer dieser Sprache zur Zuverlässigkeitsberechnung die Klasse *FaultTreeObject*, die das Fehlerbaumobjekt repräsentiert, von höchster Relevanz. Für spezielle Unterscheidungen in Fehlerbäumen ist zusätzlich die Klasse *FTSystemMaster* von Bedeutung und öffentlich[9]. Die Klassen *SeriesMaster*, *RedundantMaster* und *RedundantDummyObject* stellen nicht öffentliche Klassen[10] im Rahmen der automatischen Systemanalyse dar.

FaultTreeObject Das Fehlerbaumobjekt ermöglicht die Erweiterung eines Modells, um die Zuverlässigkeitsberechnung und die Fehlerbaumerstellung durchzuführen. Es ist dabei sowohl möglich ein bereits vorhandenes Modellelement durch eine Vererbung zu einem Fehlerbaumobjekt zu machen, als auch die Assoziation auf entsprechende Fehlerbaumobjekte, die zu einem Modellelement gehören. Hierunter sind beispielsweise verschiedene Fehlermodi eines Modellelementes zu verstehen.

Die Eigenschaften (UML properties) des Fehlerbaumobjektes umfassen als hier wichtigste Eigenschaften den Zuverlässigkeitswert (*reliability*), die Fehlerrate (*failureRate*), die Ausfallwahrscheinlichkeit (*failureProbability*) und die Importanz (*importance*). Des Weiteren sind in der Klasse entsprechende Gleichungen (UML constraints) hinterlegt, die die Umrechnungsvorschrift der Werte untereinander angeben. Exemplarisch sei hier der Zusammenhang zwischen der Zuverlässigkeit R und der Ausfallwahrscheinlichkeit F angegeben, wie er in den Grundlagen[11] vorgestellt wurde.

8 Anmerkung: Im Folgenden wird für eine bessere Lesbarkeit der Erläuterungen auf das korrekte Beschreiben in UML Begriffen im Einzelnen verzichtet und stattdessen vorab eine begriffliche Zuordnung getroffen: Eine *Klasse* ist eine *UML class*, eine *Instanz* ist eine *UML instance*, eine *Assoziation* ist eine *UML unidirectional association*, eine *Eigenschaft* ist eine *UML property* in einer *UML class*, eine *hinterlegte Gleichung* ist ein *UML constraint* in einer *UML class*, eine *instanziierte Eigenschaft* ist ein *UML slot* und eine *Verbindung* ist ein *UML link*.

9 vergleichbar mit öffentlichen Klassen in Programmiersprachen

10 vergleichbar mit privaten Klassen in Programmiersprachen

11 Im Grundlagenkapitel sind die Zuverlässigkeit und Ausfallwahrscheinlichkeit jeweils als Funktionen abhän-

$$R + F = 1 \tag{3.1}$$

R: Zuverlässigkeit des Fehlerbaumobjekts (*reliability*), [-]; F: Ausfallwahrscheinlichkeit des Fehlerbaumobjekts (*failureProbability*), [-]

Darüber hinaus hat die Klasse eine Assoziation auf sich selbst mit der Bezeichnung *nextObject*, die für die Erstellung eines kausal gerichtet verknüpften Netzwerkes verwendet wird. Hierarchisch untergeordnete (Teil-)Fehlerbäume werden über die Assoziation *subFTSystemMaster* mit dem Fehlerbaumobjekt verknüpft.
Im Kontext eines Fehlerbaumes entspricht eine Instanz des Fehlerbaumobjekts einem Basisereignis. Darauf wird bei der späteren Beschreibung des Fehlerbaumklassendiagramms genauer eingegangen.

FTSystemMaster Der Fehlerbaumsystemmaster stellt das Wurzelereignis eines Fehlerbaumes dar. Als Eigenschaften des Wurzelereignisses werden die Zuverlässigkeit bzw. Ausfallwahrscheinlichkeit über die Berechnung der einzelnen Teilabschnitte des mit Hilfe der Fehlerbaumobjekte erweiterten Modells berechnet und weiterverarbeitet.
Die Erstellung eines Fehlerbaumsystemmaster erfolgt entweder manuell durch eine Integration in die Modellierung des Systems oder - ohne manuelle Integration - automatisch beim Ausführen der Entwurfssprache der Zuverlässigkeitsberechnung. Für die manuelle Integration wird über die Assoziation *faultTreeObject* zur Klasse *FaultTreeObject* das zu untersuchende System bzw. werden die zu berücksichtigenden Modellelemente festgelegt, was im Gegensatz zur automatischen Erstellung, bei der alle im Modell hinterlegten Instanzen der Klasse *FaultTreeObject* über die Assoziation *faultTreeObject* verbunden werden, eine flexible, spezifische Fehlerbaumanalyse ermöglicht.
Die im Rahmen dieser Arbeit wichtigsten Eigenschaften des Fehlerbaumsystemmasters bestehen aus dem Systemzuverlässigkeitswert (*systemReliability*), der Systemfehlerrate (*systemFailureRate*), der Systemausfallwahrscheinlichkeit (*systemFailureProbability*) und der Systemimportanz (*systemImportance*). Die Werte der Eigenschaften werden über Gleichungen zugewiesen, die über die Assoziation *series* die zur Lösung der Gleichungen erforderlichen Werte aus der assoziierten Klasse *SeriesMaster* beziehen.
Die Eigenschaften *minimumReliability* und *maximumFailureProbability* können z.B. im Rahmen einer Optimierung zur Zielsetzung verwendet werden. Über die boolesche Eigenschaft *parallel*[12] kann festgelegt werden, ob intern nicht verknüpfte Fehlerbaumob-

gig von der Zeit angegeben. In dieser Arbeit wird Zeitinvarianz angenommen, so dass die entsprechenden Bezeichnungen im Folgenden ohne diese Parameter notiert werden.

12 Die Zuweisung eines Wertes im Klassendiagramm ist eine Möglichkeit zur Standardwerteinstellung bzw.

jekte[13] parallel oder seriell behandelt werden sollen, also kausal abhängig oder unabhängig sind. Am Beispiel des Beleuchtungssystems kann das funktional automatisch als paralleles System erkannte Parallelsystem, manuell mit einer Instanz der Klasse *FTSystemMaster* erzeugt werden, die die beiden Komponenten *Schalter1* und *Schalter2* als Instanz der Klasse *FaultTreeObject* verknüpft. Durch Setzen der Eigenschaft *parallel* auf wahr, wird dies bei entsprechender Modifikation des Zuverlässigkeitsmodells analog der automatischen Erzeugung funktional parallel verarbeitet. Eine serielle Anwendung ist beispielsweise die Modellierung verschiedener Fehlermodi mit den kausal selben Auswirkungen, z.B. *Lampe* erfüllt Funktion nicht, da Schirm defekt oder (Glüh-)Draht defekt.

SeriesMaster Bei der Klasse *SeriesMaster* handelt es sich um eine interne Klasse, die während des Ausführens der Entwurfssprache zur Zuverlässigkeitsberechnung verwendet wird, um entsprechende Instanzen für die erforderlichen Teilabschnitte zu erstellen. In Bezug auf die Fehlerbaumbegrifflichkeiten handelt es sich hierbei um ein Gatter, das als Oder-Gatter[14] interpretiert werden kann.
Die Eigenschaften der Klasse *SeriesMaster* sind der Seriensystemzuverlässigkeitswert (*seriesSystemReliability*), die Seriensystemfehlerrate (*seriesFailureRate*), die Seriensystemausfallwahrscheinlichkeit (*seriesSystemFailureProbability*) und die Seriensystemimportanz (*seriesImportance*). Neben Umrechnungsgleichungen zwischen diesen Eigenschaften ist eine Gleichung zur Berechnung des Seriensystemzuverlässigkeitswertes hinterlegt, die auf die Assoziation *faulttreeobject* zur Klasse *FaultTreeObject* zurückgreift. Diese Gleichung leitet sich aus der Gleichung für Oder-Gatter, die in den Grundlagen beschrieben wurde und logische und kausale Unabhängigkeit bedingt, ab[15] und berechnet folglich die Zuverlässigkeit einer Instanz der Klasse *SeriesMaster* aus dem Produkt der Zuverlässigkeiten der über die Assoziation *faulttreeobject* verbundenen Instanzen der Klasse *FaultTreeObject*:

$$R_{serie} = \prod^{i} R_i \tag{3.2}$$

R_{serie}: Zuverlässigkeit des Seriensystems (*seriesSystemReliability*), [-]; R_i: Zuverlässigkeit eines Fehlerbaumobjekts (*reliability*), [-]

zum Setzen eines Wertes. Für die Eigenschaft *parallel* ist dieser auf *false* gesetzt.

13 Keine Verknüpfungen über die Verbindung *nextObject* der Instanzen der Klasse *FaultTreeObject*, die über die Verbindung *faultTreeObject* mit dem *FTSystemMaster* verknüpft sind, untereinander.

14 Dies wirkt intuitiv falsch, da eine Serie mit dem Und-Gatter assoziiert wird. Hier handelt es sich allerdings um Ausfälle, was kausal die Bedeutung hat, dass ein Ausfall in einem Seriensystem den Ausfall des kompletten Seriensystems bedingt. Sprich Systemausfall, wenn Komponente1 oder Komponente2 oder ... oder KomponenteN ausfällt.

15 Umrechung: $F_{serie} = 1 - \prod(1 - F_i)$ mit $R + F = 1$ -> $F = 1 - R$ -> $1 - R_{serie} = 1 - \prod(1 - (1 - R_i))$

RedundantMaster Neben der Klasse *SeriesMaster*, die für die Modellierung der Seriensystemabschnitte verwendet wird, wird die Klasse *RedundantMaster* - innerhalb der Entwurfssprache für die Zuverlässigkeitsberechnung - für die Modellierung der Parallelsystemabschnitte eingesetzt. Im Kontext eines Fehlerbaumes repräsentiert diese Klasse die Und-Gatter[16].

Die Eigenschaften der Klasse *RedundantMaster* sind der Parallelsystemzuverlässigkeitswert (*parallelSystemReliability*), die Parallelsystemfehlerrate (*parallelFailureRate*), die Parallelsystemausfallwahrscheinlichkeit (*parallelSystemFailureProbability*) und die Parallelsystemimportanz (*parallelImportance*). Abseits der Umrechnungsgleichungen zwischen diesen Eigenschaften ist eine Gleichung zur Berechnung der Parallelsystemausfallwahrscheinlichkeit aus den Ausfallwahrscheinlichkeiten der über eine Assoziation *series* verknüpften Klasse *SeriesMaster* hinterlegt. Die Parallelsystemausfallwahrscheinlichkeit ergibt sich entsprechend der in den Grundlagen vorgestellten Gleichung für Und-Gatter, die logische und kausale Unabhängigkeit bedingt, aus dem Produkt der Seriensystemausfallwahrscheinlichkeit der über die Assoziation *series* verknüpften Instanzen der Klasse *SeriesMaster*:

$$F_{parallel} = \prod^{i} F_i \tag{3.3}$$

$F_{parallel}$: Ausfallwahrscheinlichkeit des Parallelsystems (*parallelSystemFailureProbability*), [-]; F_i: Ausfallwahrscheinlichkeit eines Seriensystems (*seriesSystemFailureProbability*), [-]

RedundantDummyObject Die Klasse *RedundantDummyObject* stellt eine Hilfsklasse im Vokabular der Entwurfssprache für die Zuverlässigkeitsberechnung dar. Diese Klasse wird zur Erstellung von Ersatzobjekten für Parallelsysteme verwendet, wodurch ein Bezugsobjekt zur Fehlerbaumerstellung entsteht, welches das zu einem Parallelsystem gehörige Zwischenereignis darstellt.

Die Klasse *RedundantDummyObject* erbt von der Klasse *FaultTreeObject* und ist dementsprechend ein Fehlerbaumobjekt mit denselben Eigenschaften. Über die geerbten Eigenschaften hinaus wird die Klasse um eine Assoziation *redundantMaster* zur Klasse *RedundantMaster* erweitert, die zusammen mit einer zusätzlich hinterlegten Gleichung die nötige Übernahme des Zuverlässigkeitswertes aus der Klasse *RedundantMaster*, da für ein Zwischenereignis keine Werte gesetzt, sondern berechnet werden, erlaubt.

16 Dies wirkt intuitiv falsch, da Redundanz/Parallelität mit dem Oder-Gatter assoziiert wird. Hier handelt es sich allerdings um Ausfälle, was kausal die Bedeutung hat, dass ein Ausfall eines redundanten Systems den Ausfall jeder einzelnen redundanten Komponenten bedingt. Sprich Systemausfall, wenn Komponente1 und Komponente2 und ... und KomponenteN ausfällt.

$$R_{dummy} = R_{parallel} \tag{3.4}$$

R_{dummy}: Zuverlässigkeit des Ersatzobjektes für Parallelsysteme (*reliability*), [-]; $R_{parallel}$: Zuverlässigkeit des Parallelsystems (*parallelSystemReliability*), [-]

Aufbauend auf der Systemanalyse folgt die automatische Fehlerbaumerstellung, die entsprechend ein eigenes Vokabular mit Verbindungen zum Vokabular der Systemanalyse benötigt. Das Klassendiagramm wird im Nachfolgenden beschrieben.

Klassendiagramm zur Fehlerbaumerstellung

Das Vokabular zur Erstellung von Fehlerbäumen ist im Klassendiagramm in Abbildung 3.4 dargestellt. Hier sind die für eine Fehlerbaumerstellung typischen Elemente, bestehend aus Gattern und Ereignissen, repräsentiert, wobei besonderes Augenmerk auf die Einpflegung der Kernelemente gelegt wurde. Mit diesem Vokabular kann im Modell ein entsprechender Fehlerbaum erstellt werden, der dann in externen Fehlerbaumvisualisierungsprogrammen grafisch aufbereitet dargestellt werden kann. Im Folgenden werden die einzelnen Klassen des Klassendiagramms im Detail beschrieben.

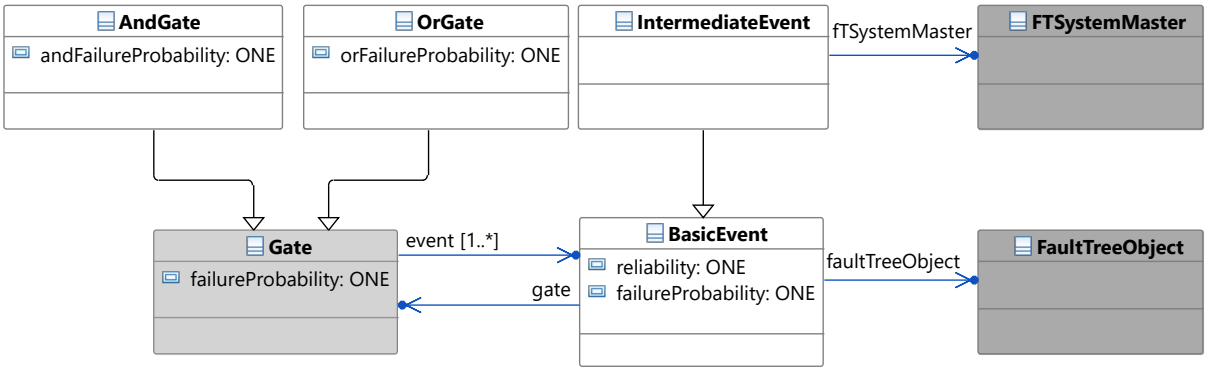

Abbildung 3.4: Klassendiagramm der Fehlerbaumelemente in der UML[17]

BasicEvent Ein Fehlerbaum, der typischer Weise ausgehend von einem Wurzelereignis aufgebaut wird, endet auf der untersten Ebene mit den Basisereignissen, die durch den Grad der angestrebten Detaillierung bestimmt werden. Diese Basisereignisse können beispielsweise Komponentenausfälle in einer Baugruppe oder einzelne Fehlermodi sein, die kausal für einen (Teil-)Systemausfall sind.

Die Klasse *BasicEvent*, die die Basisereignisse repräsentiert, besitzt, um neben einer qualitativen auch eine quantitative Fehlerbaumanalyse zu ermöglichen, die quantitativen

17 (Dunkel-)Grau hinterlegte Klassen sind aus anderen Klassendiagrammen importierte Klassen.

Eigenschaften Zuverlässigkeit (*reliability*) und Ausfallwahrscheinlichkeit (*failureProbability*). Die Abhängigkeit beider Eigenschaften voneinander ist über die in Kapitel 2.1.1 beschriebene Gleichung in der Klasse hinterlegt:

$$R + F = 1 \tag{3.5}$$

R: Zuverlässigkeit des Basisereignisses (*reliability*), [-]; F: Ausfallwahrscheinlichkeit des Basisereignisses (*failureProbability*), [-]

Des Weiteren sind Assoziationen zur Klasse *FaultTreeObject*, bezeichnet mit *faultTreeObject* und zur Klasse *Gate*, bezeichnet mit *gate*, implementiert. Diese dienen der Übergabe bzw. Übernahme von Eigenschaftswerten (*reliability* und *failureProbability*) bezüglich der Klassen *FaultTreeObject* und *Gate*. Die Assoziation *faultTreeObject* ist die Verknüpfung zwischen den Klassendiagrammen für die Systemanalyse und dem zur Fehlerbaumerstellung. Dargestellt als Fehlerbaumsymbol wird diese Klasse durch einen Kreis, wie er im Kapitel 2 *Stand des Wissens* in Tabelle 2.1 zu sehen ist.

IntermediateEvent Neben den Basisereignissen sind in Fehlerbäumen Zwischenereignisse grundlegende Elemente. Diese werden in der Fehlerbaumsymbolik durch Rechtecke dargestellt. Die Zwischenereignisse bilden im Gegensatz zu den Basisereignissen keinen Fehlerzustand auf unterster Detaillierungsebene eines Fehlerbaumes ab, sondern stellen die kausale Wirkung untergeordneter Ereignisse dar.
Die Klasse *IntermediateEvent* erbt von der Klasse *BasicEvent*[18] und hat somit die entsprechenden Eigenschaften. Zusätzlich zu den Eigenschaften der Klasse *BasicEvent* ist eine Gleichung hinterlegt, die die Übernahme der Eigenschaftswerte eines über die Assoziation *gate* verknüpften Gatters ermöglicht. Für die Ausfallwahrscheinlichkeit eines Zwischenereignisses bedeutet dies, dass sie mit der Ausfallwahrscheinlichkeit des verknüpften Gatters gleichgesetzt wird:

$$F_{event} = F_{gatter} \tag{3.6}$$

F_{event}: Ausfallwahrscheinlichkeit des Zwischenereignisses (*failureProbability*), [-]; F_{gatter}: Ausfallwahrscheinlichkeit des Gatters (*failureProbability*), [-]

Darüber hinaus ist eine Assoziation zur Klasse *FTSystemMaster* mit der Bezeichnung *fTSystemMaster* integriert. Diese direkte Verknüpfung kann beispielsweise zur modularen und spezifischen (Teil-)Fehlerbaumerstellung genutzt werden.

18 Diesen Zusammenhang kann man gut aus dem Zitat im englischen Sprachgebrauch ableiten: *All failures are faults, but not all faults are failures!* [121]

Gate Ereignisse in einem Fehlerbaum sind über Gatter verbunden, die den logischen Aufbau des Fehlerbaumes festlegen. Hierzu wurde eine Oberklasse *Gate* erstellt, die als abstrakt anzusehen ist und die grundlegenden Eigenschaften von Gattern zur Verfügung stellt. Als Eigenschaft ist die Ausfallwahrscheinlichkeit (*failureProbability*) hinterlegt, die zu Übergabe- und Übernahmezwecken für die Berechnung verwendet wird. Zu diesem Zweck ist auch die Assoziation *event* zur Klasse *BasicEvent* erstellt worden.

AndGate Die Klasse *AndGate* ist eine Gatterklasse, die von der Oberklasse *Gate* erbt. Sie dient zur logischen und kausalen Verknüpfung von voneinander unabhängigen Ereignissen, die multikausal für ein gemeinsames abhängiges Folgeereignis sind. Die Darstellung in der Fehlerbaumsymbolik erfolgt durch ein Schild mit gerader unterer Kante.
Über die Eigenschaften der Oberklasse hinausgehend, besitzt die Gatterklasse *AndGate* eine weitere Eigenschaft für die spezielle Ausfallwahrscheinlichkeit des Gatters (*andFailureProbability*). Des Weiteren sind zwei Gleichungen hinterlegt. Die eine ermöglicht den Austausch von Werten mit der Oberklasseneigenschaft *failureProbability*, die andere repräsentiert die Berechnung der speziellen Ausfallwahrscheinlichkeit des Gatters über die Assoziation *event*:

$$F_{und} = \prod^{i} F_{event,i} \tag{3.7}$$

F_{und}: Ausfallwahrscheinlichkeit des Und-Gatters (*andFailureProbability*), [-]; $F_{event,i}$: Ausfallwahrscheinlichkeit eines Ereignisses (*failureProbability*), [-]

OrGate Die Klasse *OrGate* stellt eine weitere Gatterklasse dar. Sie erbt ebenfalls von der als abstrakt anzusehenden Klasse *Gate*. Analog zur Klasse *AndGate* besitzt sie eine Eigenschaft für die spezielle Ausfallwahrscheinlichkeit des Gatters *orFailureProbability*, eine Gleichung zum Austausch der Eigenschaftswerte mit den Eigenschaften der Oberklasse und eine weitere Gleichung zur Berechnung der speziellen Ausfallwahrscheinlichkeit des Gatters aus den Zuverlässigkeiten der verknüpften Ereignisse:

$$F_{oder} = 1 - \prod^{i} R_{event,i}{}^{19} \tag{3.8}$$

F_{oder}: Ausfallwahrscheinlichkeit des Oder-Gatters (*orFailureProbability*), [-]; $R_{event,i}$: Zuverlässigkeit eines Ereignisses (*reliability*), [-]

19 Hier wird direkt die Zuverlässigkeit für die einzelnen Ereignisse verwendet, da in der verwendeten Version des Entwurfscompilers 43 lediglich Variablen, aber keine Terme ($R_{event,i} = 1 - F_{event,i}$) innerhalb der Produktfunktion intern verarbeitet werden können. Da die Klasse *OrGate* nur die Ausfallwahrscheinlichkeit als Eigenschaft besitzt, unterscheidet sich die Gleichung von Gleichung 3.2 für die Klasse *Serie*.

Die Instanzen der Klasse *OrGate* repräsentieren logische Verknüpfungen voneinander unabhängiger Ereignisse, die monokausal für das selbe Folgeereignis sind. Als Symbol im Fehlerbaum wird ein Schild mit einer nach oben abgerundeten Unterkante verwendet.

3.2.1 Fehlerbaumerstellung

Die in dieser Arbeit aufgezeigte Fehlerbaumerstellung verwendet die zuvor beschriebenen Vokabeln aus dem Klassendiagramm für die Zuverlässigkeitsberechung, welches für die Systemanalyse verwendet wird und dem Klassendiagramm der Fehlerbaumelemente, siehe Abbildung 3.2 und 3.4. Mithilfe dieses Vokabulars wird aus einem - unter Berücksichtigung der im Abschnitt 3.1 *Graphenbasierte Netzwerkdarstellung* beschriebenen Abgrenzungen und Annahmen - modellierten Netzwerk ein Fehlerbaum synthetisiert, an dem nachfolgend eine Fehlerbaumanalyse[20] erfolgen kann.
Ein entsprechend semantisch annotiertes Netzwerk wird dabei schrittweise automatisch über die Analyse des erweiterten Modellgraphen in für einen Fehlerbaum relevante Teilabschnitte algorithmisch zerlegt. Für diese Zerlegung in Teilabschnitte werden Graphenalgorithmen verwendet. Über (Graphen-)Suchalgorithmen, bestehend aus einer kombinierten Tiefen- und Breitensuche mit Abbruchkriterien, angewandt auf den Modellgraphen, werden Teilgraphen als serielle und parallele Abschnitte automatisch identifiziert. Die Identifikation der Teilgraphen im Rahmen der algorithmischen Zerlegung verfolgt das Ziel die Strukturfunktion des Netzwerkes zu ermitteln, um auf Grundlage dieser einen Fehlerbaum zu erstellen. Anhand des Schaltkreisbeispieles werden die Grundlagen der hier beschriebenen automatischen Fehlerbaumerstellung anschaulich verdeutlicht.

Prinzipielle Fehlerbaumerstellung
Im Ablauf einer Fehlerbaumanalyse wird zur Fehlerbaumerstellung zunächst das unerwünschte Ereignis festgelegt, welches die Spitze des Fehlerbaumes repräsentiert. Als nächster Schritt wird das System dahingehend untersucht, welche Ereignisse und Ereigniskombinationen zum unerwünschten Ereignis führen können. Sofern nicht alle Ereignisse Basisereignisse sind, werden die entsprechenden Ereignisse weiter entwickelt. Dieser Schritt wird solange wiederholt bis die unterste Ebene der gewünschten Detaillierung erreicht wurde und ausschließlich Basisereignisse weiter zu entwickeln wären.

20 Hier ist nicht die Methode gemeint, sondern die Analyse bzw. Auswertung als Schritt der Methode. In der Methode ist die semantische Annotation und die Erstellung des Fehlerbaumes Teil der Fehlerbaumanalyse.

Schematisches Beispiel

Mit diesem Beispiel soll verdeutlicht werden, wie, ausgehend von einem modellierten Netzwerk mit semantischer Annotation, ein zugehöriger Fehlerbaum synthetisiert wird. Die einzelnen Schritte dieser Fehlerbaumerstellung sind:

- Systembeschreibung
- Funktionsbeschreibung des Systems
- Wurzelereignis festlegen
- Identifikation kausaler Ereignisse und Abhängigkeiten
- Identifikation weiter zu entwickelnder Ereignisse
- Eventuelle Festlegung neuer interner Wurzelereignisse und Iteration (Abbruchkriterium: ausschließlich Basisereignisse)

System- und Funktionsbeschreibung Als Beispiel dient das in Abbildung 3.3a dargestellte Netzwerk, bestehend aus den vier Komponenten *Energiequelle*, *Schalter1*, *Schalter2* und *Lampe*. Die Funktion des Netzwerkes ist, dass eine Beleuchtung erfolgt. Diese wird erreicht, sofern die Lampe leuchtet. Dazu muss das Signal (Strom) eingehend bei der *Energiequelle* über einen Schalter und die *Lampe* wieder zurück zur *Energiequelle* laufen. Dabei wird das Signal von der *Energiequelle* sowohl an *Schalter1* als auch an *Schalter2* weitergegeben, die jeweils das Signal an die *Lampe* weitergeben. Hierbei ist es für die *Lampe* hinreichend, wenn das Signal von einer Quelle ankommt, um ein korrektes Ausgangssignal weiter zu geben. Jede Komponente wird als Zweizustandskomponente betrachtet, d.h. sie ist zu einem Zeitpunkt t entweder voll funktionsfähig oder komplett ausgefallen, im Sinne der Systemfunktion. Die Funktion einer Komponente ist im Bezug auf das Beispiel, dass das Eingangssignal das korrekte Ausgangssignal erzeugt und weitergibt, bei Erfüllen der internen Funktion. Beispielsweise, dass die Lampe leuchtet und das Signal weitergegeben wird.[21]

Systemzerlegung Nachdem die Funktionalität festgelegt wurde, wird im Rahmen der Fehlerbaumerstellung ein zu untersuchendes unerwünschtes Ereignis festgelegt, in diesem Fall der Ausfall des Systems durch Nichterfüllen der vorgegebenen Funktionalität. Nachfolgend werden die Ereignisse in der darunterliegenden Hierarchieebene bestimmt.

21 Freese [39] gibt in seinem Ansatz die Signalfehler als wichtige, zusätzliche Ereignisse im Fehlerbaum an. Durch die zur Verfügung gestellten Werkzeuge ist dies ebenfalls möglich, wird aber in den Beispielen nicht entsprechend extra ausgeführt, sondern implizit in den Ereignissen mit angenommen.

Hierfür werden im Sinne der Kausalität die Ursachen ermittelt, die zum unerwünschten Ereignis führen. Im Beispielfall sind dies das Nichterfüllen der Funktionalität der Komponenten *Energiequelle* und *Lampe* sowie der Komponentenkombination *Schalter1* und *Schalter2*. Bei den ersten zwei Ereignissen handelt es sich um Basisereignisse, die Komponentenausfallskombination muss allerdings in einer weiteren Iteration betrachtet werden. Das hierfür zu Grunde liegende unerwünschte Ereignis ist das Nichterfüllen der Funktionalität durch die Komponentenkombination. Die Ereignisse, die zu diesem Ereignis führen, sind das gleichzeitige Nichterfüllen der Funktionalität der Komponenten *Schalter1* und *Schalter2*. Das Nichterfüllen der geforderten Funktionalität der Komponenten *Schalter1* und *Schalter2* sind Basisereignisse, wodurch die Detaillierungsgrenze erreicht ist. Ein Fehlerbaum für das Wurzelereignis des Systemausfalls des Schaltkreises ist in Abbildung 3.5 dargestellt.

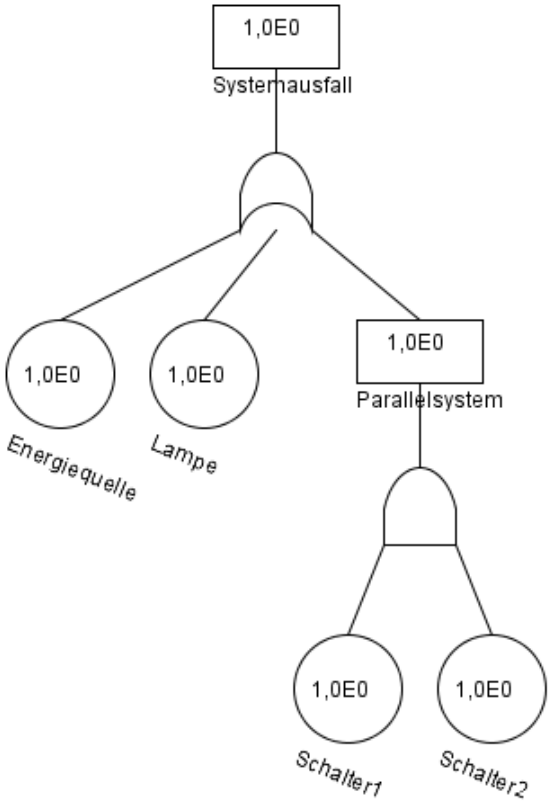

Abbildung 3.5: Fehlerbaum Schaltkreis[22]

22 Die Werte sind hier beispielhaft auf $F = 1$ gesetzt, da hier der Fokus auf der Struktur des erstellten Fehlerbaumes liegt.

Automatische Fehlerbaumerstellung Um den beschriebenen Ablauf zur Fehlerbaumerstellung zu automatisieren wird das semantisch annotierte System in folgenden Schritten hinsichtlich der Zuverlässigkeitsanalyse und Fehlerbaumerstellung erweitert, ohne das zu Grunde liegende Systemmodell zu verändern:[23]

- System in Seriensystem umwandeln
- System unterteilen
- Seriensysteme identifizieren
- Parallelpfade zusammenfassen
- Parallelsystemteile durch Ersatzobjekt darstellen
- Iteration durchführen (Abbruchkriterium: reines Seriensystem[24])

Bezogen auf das Beispiel wird zunächst das gesamte System algorithmisch dahingehend untersucht, ob parallele Abschnitte vorhanden sind. Im Beispiel handelt es sich um kein reines Seriensystem, also ein System ohne interne Parallelsysteme. Auf Grund dessen wird ein paralleler Abschnitt gefunden, so dass das erweiterte System für die Fehlerbaumerstellung in ein Seriensystem umgewandelt werden muss. Dazu wird das System zunächst in einzelne Abschnitte unterteilt, wie in Abbildung 3.3b zu sehen. Die Abschnitte, die bereits zu einem Seriensystem beitragen und die Abschnitte, die weiter untersucht werden müssen. Unter die weiter zu untersuchenden Abschnitte fallen im Beispiel die Komponenten *Schalter1* und *Schalter2*, die als Abschnitt zu einem Parallelsystem zusammengefasst werden. Innerhalb dieses Parallelsystems werden die einzelnen parallelen Pfade auf reine Seriensysteme überprüft, dargestellt in Abbildung 3.3c. Die im Beispiel dargestellten Pfade bestehen aus je einem Ereignis, das als reines Seriensystem angesehen wird. Die entsprechenden Parallelpfade, die aus reinen Seriensystemen bestehen, werden danach zusammengefasst zum Parallelsystem, ebenfalls zu sehen in Abbildung 3.3c. Dieses Parallelsystem wird durch ein Ersatzobjekt im darüber liegendem System dargestellt. Mit Hilfe dieses Ersatzobjekts wird das Gesamtsystem als reines Seriensystem interpretiert, wodurch dieses nun zusammengefasst betrachtet werden kann, wie auch in Abbildung 3.3c zu sehen ist.
Diese algorithmische Unterteilung und Bearbeitung eines Systems wird von Graphtransformationen begleitet, die auf das im vorherigen Abschnitt gezeigte Klassendiagramm

23 Die allgemeine Beschreibung ist in Abschnitt 3.1.2.2 *Netzwerkzerlegung* in Algorithmus 1 beschrieben.
24 System ohne interne Parallelsysteme

zur Systemanalyse zurückgreifen und das (Ausgangs-)Systemmodell - unter Beibehaltung der ursprünglichen Funktionalität - erweitern. In Abbildung 3.3d ist schematisch der erstellte Graph dargestellt. Dieser Graph bildet an der Spitze das definierte Wurzelereignis mit einer Instanz der entsprechenden Klasse *FTSystemMaster* ab. Darunter ist das Seriensystem als Instanz der Klasse *SeriesMaster* für das reduzierte Gesamtsystem aufbauend auf dem Ersatzobjekt in Abbildung 3.3c dargestellt. Gefolgt von den Komponenten *Energiequelle*, *Lampe* und dem Ersatzobjekt *Parallelsystem* jeweils als Instanzen der Klasse *FaultTreeObject*. Durch Ersetzung des Ersatzobjektes mit dem darunter liegenden Graphen, wird das Ersatzobjekt durch ein Parallelsystem der Klasse *RedundantMaster* ersetzt. Diesem Parallelsystem folgen die zwei Seriensysteme der Klasse *SeriesMaster* auf den Parallelpfaden. Diese Seriensysteme bestehen ihrerseits aus den Komponenten *Schalter1* bzw. *Schalter2* der Klasse *FaultTreeObject*.
Aufbauend auf diesem im Graphen dargestellten Modell kann mit Hilfe der Vokabeln aus dem Klassendiagramm für das Fehlerbaumprofil ein Fehlerbaum erstellt werden. Dabei werden zu den im Graphen erstellten Instanzen der Klasse *FaultTreeObject* Ereignisse der Klasse *BasicEvent* erstellt und mit diesen verknüpft. Instanzen der Klasse *SeriesMaster* werden zu Gattern der Klasse *AndGate*, sowie Instanzen der Klasse *RedundantMaster* zu Gattern der Klasse *OrGate*. Die entsprechenden Zwischenereignisse werden als Instanzen der Klasse *IntermediateEvent* zusammen mit dem zugehörigen Gatter generiert.

An diesem Beispiel sind die zwei prinzipiell, kausal unterschiedlichen Kombinationsmöglichkeiten von Ereignissen zu erkennen. Zum einen die Ereignisse einzelner Komponenten bzw. Komponentenkombinationen, die unabhängig voneinander zu einem übergeordneten Ereignis führen, die als serielle Bereiche angesehen werden. Und zum anderen Ereignisse, die im Zusammenspiel mit anderen Ereignissen zu übergeordneten Ereignissen führen, so genannte parallele Abschnitte.
Serielle Abschnitte Die drei seriellen Abschnitte im Beispiel bestehen auf der zweiten Ebene[25] aus den Basisereignissen *Energiequelle* und *Lampe*, sowie dem zusammengefasst als *Parallelsystem* bezeichneten Zwischenereignis. Auf unterster Ebene ergeben sich zwei serielle Abschnitte mit je einem Basisereignis, zum einen mit dem Basisereignis *Schalter1* und zum anderen mit dem Basisereignis *Schalter2*.
Parallele Abschnitte Im vorgestellten Beispiel ist ein paralleler Abschnitt zu finden, der aus den Ereignissen *Schalter1* und *Schalter2* besteht.

25 Die Ebenennummerierung erfolgt bei den hier dargestellten Fehlerbäumen beginnend beim Wurzelereignis, welches die erste Ebene des Fehlerbaumes darstellt. Eine Ebene besteht lediglich aus Ereignissen. Gatter sind keiner Ebene zugehörig.

3.2.2 Fehlerbaumanalyse

Aufbauend auf dem im Graphen repräsentierten Modell, das zur Zuverlässigkeitsberechnung erstellt wird, kann, neben der zuvor aufgezeigten Fehlerbaumerstellung, direkt eine Fehlerbaumauswertung erfolgen. Dies gilt sowohl für eine quantitative als auch eine qualitative Fehlerbaumanalyse, die nachfolgend genauer beschrieben werden.

Quantitative Fehlerbaumanalyse
Die quantitative Fehlerbaumanalyse wird durch die Verwendung der in den Klassen hinterlegten Gleichungen und dem sich somit aus Kopplungen der Instanzen ergebendem Gleichungssystem vorgenommen. Dieses Gleichungssystem wird mit Hilfe eines im Entwurfscompiler 43 integrierten Lösungspfadgenerators (LPG; englisch: solution path generator (SPG))[26] automatisch rechnergestützt in einem Computeralgebrasystem (CAS) gelöst, wobei die Ergebnisse als Attribute zurück in die Instanzen geschrieben werden. Auf diese im Modell hinterlegten Ergebnisse kann im weiteren Programmablauf zurückgegriffen werden.

Qualitative Fehlerbaumanalyse
In Rahmen der qualitativen Fehlerbaumanalyse wird das für die Zuverlässigkeitsberechnung erweiterte Modell hinsichtlich Pfaden und Schnitten ausgewertet. Insbesondere minimale Pfade und Schnitte sind für eine qualitative Auswertung[27] von hoher Bedeutung, da über diese Teilmengen kritische Stellen im Fehlerbaum, die kritische Stellen im System repräsentieren, identifiziert werden. Minimale Schnitte stellen für die Auswertung und Identifikation die wichtigere Menge dar, da sie eine Menge von Ereignissen enthalten, bei deren gleichzeitigen Auftreten das Auftreten des Wurzelereignisses kausal bedingt ist, während die Menge an Ereignissen eines minimalen Pfades bei deren gemeinsamen Ausbleiben das Ausbleiben des Wurzelereignisses bedeutet.
Minimale Pfade werden hierbei algorithmisch[28] basierend auf einer Tiefensuche ermittelt, allerdings ohne Vorgabe des Start- und Zielknotens, wie in den meisten Implementierungen. Diese Abwandlung ist nötig, da die Start- und Zielknoten automatisch ermittelt werden und zusätzlich alle minimalen Pfade gefunden werden müssen und nicht der minimale Pfad.[29]

26 Eine ausführliche Beschreibung über den integrierten Lösungspfadgenerator ist in Bölling [15] zu finden.
27 Die entsprechenden Definitionen sind in Kapitel 2.1.1 *Fehlerbaumanalyse (FTA)* zu finden.
28 basierend auf den von Kusiak [62] beschriebenen Algorithmen
29 „Der“ minimale Pfad verstanden als kürzester Pfad kann mit entsprechenden Algorithmen effizienter ermittelt werden. Kürzeste Pfadalgorithmen, wie von Dijkstra [23] oder Bellman-Ford [9], die von einer Quelle ausgehen (single-source shortest paths) oder wie von Floyd-Warshall [37] [114], die zwischen allen Kno-

Der Algorithmus beginnt bei einem Knoten in Form einer Instanz der Klasse *FaultTreeObject*, die mit einer Instanz der Klasse *FTSystemMaster* über die Verbindung *faultTreeObject* verbunden ist. Von jedem Knoten werden alle Pfade bis zum Endknoten ermittelt, indem die nachfolgenden Knoten, die über die Verbindung *nextObject* verbunden sind und ebenfalls der selben Instanz der Klasse *FTSystemMaster*, wie der zu untersuchende Knoten der Instanz der Klasse *FaultTreeObject*, über die Verbindung *faultTreeObject* zugeordnet sind. Falls für diesen nachfolgenden Knoten bereits die minimalen Pfade ermittelt sind, werden diese mit dem vorangegangen Knoten entsprechend zugeordnet, falls nicht wird von diesem Knoten ausgehend die beschriebene Prozedur wiederholt, bis alle nachfolgenden Knoten abgearbeitet sind. Hierbei wird für jeden Knoten der minimale Pfad gespeichert. Dieser Vorgang wird für alle Knoten entsprechend der oben beschriebenen Konstellation durchgeführt, wobei bereits bearbeitete Knoten - ausgehend von einer Instanz der Klasse *FTSystemMaster* - nicht mehrfach untersucht werden.
Nachdem alle Pfade für jeden Knoten bis zum Endknoten[30] ermittelt wurden, werden die Pfade miteinander verglichen. Dabei wird zunächst ein Pfad zufällig ausgewählt und die Knoten dieses Pfades mit Knoten der anderen Pfade verglichen. Wird ein Pfad gefunden, der zusätzlich zu den Knoten des Pfades weitere Knoten beinhaltet, wird dieser als nicht minimal verworfen. Ein Pfad der eine Teilmenge der Knoten des Pfades enthält wird als neuer zu untersuchender Pfad ausgewählt und der vorherige ebenfalls als nicht minimal verworfen. Nach einer Iteration über alle Pfade ist somit ein minimaler Pfad ermittelt worden. Danach wird weiter über die verbliebenen, also nicht verworfenen Pfade nach der selben Vorgehensweise iteriert bis lediglich minimale Pfade verbleiben. Algorithmus 2 stellt die beschriebene Vorgehensweise in einer vereinfachten Form dar.
Zur Bestimmung der *minimalen Schnitte*, ausgehend von jedem Knoten, wird auf die ermittelten minimalen Pfade zurückgegriffen.[31] Aus diesen werden alle enthaltenen Kno-

ten eines Graphen laufen (all-pairs shortest path), zielen darauf ab einen bezogen auf die Gewichtung der Kanten minimalen Pfad zwischen zwei Knoten zu ermitteln, also den (mit Abwandlung alle) kürzesten Pfad bei einheitlicher Gewichtung der Kanten. Wie in der Definition in Kapitel 2.1.1 *Fehlerbaumanalyse (FTA)* beschrieben, weicht die minimale Pfaddefinition für Fehlerbaumanalysen von dieser ab, da hier alle möglichen Pfade, die nicht andere Pfade beinhalten, definiert sind.

30 Mehrere Endknoten werden über einen Ersatzknoten zu einem Knoten zusammengefasst und nachfolgend für eine Berechnung sowie Fehlerbaumerstellung wieder aufgelöst.

31 Für die Ermittlung minimaler Schnitte eines Graphen sind verschiedene Algorithmen bekannt (auch unter der Bezeichnung maximaler Fluss nach dem Max-Flow-Min-Cut-Theorem), wie von Ford und Fulkerson [38] oder Dinic [30] bzw. Edmonds und Karp [33], aber auch aktuellere zurückgehend auf Goldberg [44], die die Laufzeit reduzieren. Diese sind - wie bei den minimalen Pfaden - darauf ausgelegt, den einen minimalen Schnitt bzw. maximalen Fluss eines Graphen zwischen zwei Knoten zu ermitteln und nicht die Menge an Schnitten, die, wie in der Definition in Kapitel 2.1.1 *Fehlerbaumanalyse (FTA)* beschrieben, für die Fehlerbaumanalyse benötigt wird, weshalb hier eine Lösung unter Verwendung der minimalen

Algorithmus 2: Bestimme Minimale Pfade (nach Kusiak [62])

Eingabe: Graph $G(V, E)$
Ausgabe: minimale Pfade $P_{min}(V)$

$P_{geordnet}(V) := P(V)$ nach Anzahl V geordnet von minimal zu maximal;
für alle *Pfade* $P(V)$ **tue**
  wenn $P_{min}(V) \subseteq P_i(V)$ **dann**
    überprüfe nächsten Pfad $P_{i+1}(V)$;
  Ende
  für alle *Pfade* $P_{geordnet}(V)$ **tue**
    wenn $P_{geordnet,i}(V) \subseteq P_i(V)$ **dann**
      überprüfe nächsten Pfad $P_{i+1}(V)$;
    Ende
  Ende
  $P_i(V)$ zu $P_{min}(V)$ hinzufügen;
Ende

ten für die Erstellung von Ausfallkombinationen extrahiert. Die generierten Ausfallkombinationen beginnen mit Einzelausfällen und werden bis zu einer maximalen Schnittgröße[32] erstellt. Mit einer Ausfallkombination[33] wird geprüft, ob Knoten dieser Kombination in allen minimalen Pfaden des zu untersuchenden Knotens gefunden werden und im Falle dessen die Kombination als minimaler Schnitt ausgegeben. In diesem iterativen Prozess werden generierte Kombinationen, die einen bereits ermittelten minimalen Schnitt beinhalten, von einer weiteren Untersuchung ausgeschlossen. Dieser Ablauf ist in Algorithmus 3 in einer vereinfachten Form dargestellt.
Mit dieser Vorgehensweise werden alle minimalen Pfade und Schnitte, ausgehend von jedem zugeordnetem Knoten ermittelt, aufbauend auf dem semantisch annotierten System. Die Ermittlung der minimalen Pfade und Schnitte direkt aus dem modellierten System heraus ist insbesondere für komplexe Systeme, wie Brückensysteme, von Bedeutung, da hier keine direkte Erstellung der Zuverlässigkeitsberechnung und des Fehlerbaumes durchgeführt werden kann.[34] Mit Hilfe von minimalen Schnitten und Pfaden ist eine Abschätzung der Zuverlässigkeit komplexer Systeme umsetzbar, wie im folgenden Abschnitt erläutert wird.

Pfade, wie von Kusiak [62], Rai [82] oder Rosenthal [89] beschrieben, aufgezeigt wird.

32 Anzahl Knoten, die in einer Ausfallkombination berücksichtigt werden. Ohne Vorgabe besteht die maximale Ausfallkombination aus allen ermittelten Knoten.

33 Eine Kombination kann hier auch ein einzelner Knoten sein.

34 Bei der direkten Erstellung eines Fehlerbaumes ist, gegebenenfalls auch mit entsprechender Fehlerbaumsoftware, die Bestimmung der minimalen Pfade und Schnitte effizienter direkt aus dem Fehlerbaum durchführbar.

Algorithmus 3: Bestimme Minimale Schnitte (nach Kusiak [62])

Eingabe: Graph $G(V, E)$
Ausgabe: minimale Schnitte $S_{min}(V)$

für alle V *aus minimalen Pfaden* $P_{min}(V)$ **tue**
  $P_{min*}(V) := P_{min}(V) \backslash V_i$;
  für alle V_* *aus verbleibenden* $P_{min*}(V)$ **tue**
    $S_{min,i}(V)$ erstellen;
    V_i zu $S_{min,i}(V)$ hinzufügen;
    wiederhole
      $V_{*,i}$ zu $S_{min,i}(V)$ hinzufügen;
      $P_{min*}(V) := P_{min*}(V) \backslash S_{min,i}(V)$;
    bis $P_{min*}(V) = \{\}$;
    $S_{min,i}(V)$ zu $S(V)$ hinzufügen;
  Ende
Ende

3.2.3 Brückensysteme

Netzwerke, die hinsichtlich der Zuverlässigkeitsberechnung[35] nicht direkt in serielle und parallele Teilsysteme unterteilt werden können, werden als komplexe Systeme, Nicht-Serien-Parallelsysteme oder Brückensysteme bzw. Brückennetzwerke [86], [108] bezeichnet. In Abbildung 3.6 ist ein solches Netzwerk schematisch dargestellt. Für diese Systeme ist im Allgemeinen - spezielle Brückensysteme können beispielsweise mit der Schlüsselwertmethode [13] berechnet werden - keine einfache exakte, analytische Berechnung der Zuverlässigkeit des Systems möglich. Um zu einer Abschätzung der Zuverlässigkeit dieser Systeme zu kommen, werden zugehörige Ersatzsysteme erstellt, von denen eine Zuverlässigkeitsberechnung sowie eine zugehörige Fehlerbaumerstellung möglich und automatisierbar umsetzbar ist.[36]

Die Erstellung der Ersatzsysteme verwendet als Grundlage minimale Schnitte und Pfade des Systems, die hier über die Algorithmen 2 und 3 bestimmt werden. Diese Ersatzsysteme haben hinsichtlich minimaler Schnitte und Pfade die selben wie das Ausgangssystem und bilden somit kausal bezogen auf die Zuverlässigkeitsanalyse (Strukturfunktion) ein äquivalentes System ab [57]. Im Anhang C ist die prinzipielle mathematische Äquivalenz der Ersatzsysteme beispielhaft ausgeführt.

35 Erstellung der Strukturfunktion.
36 In [86] ist neben der angesprochenen Schlüsselwertmethode und der hier verwendeten Ersatzsystemmethode, auch als Pfad-/Schnittrückverfolgungsmethode bezeichnet, noch die Event Space Methode beschrieben, welche, durch den sehr allgemeinen und umfassenden Ansatz durch Überprüfung jeglicher Kombinationen einen hohen Rechenaufwand verlangt, nicht weiter verfolgt wurde.

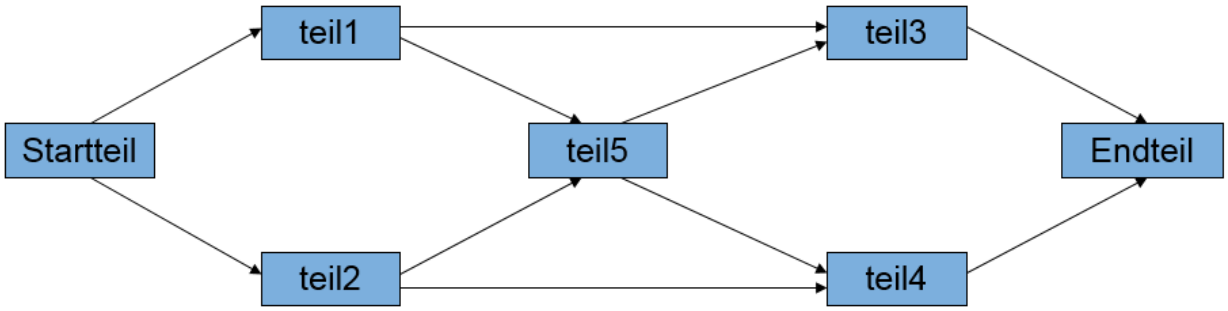

Abbildung 3.6: Brückensystem

Aus den minimalen Schnitten des Netzwerkes wird die zugehörige *reduzierte Serienform* erstellt. Hierbei werden die Elemente der einzelnen minimalen Schnitte parallel zueinander angeordnet und seriell mit den weiteren minimalen Schnitten verbunden. In Abbildung 3.7 ist das serielle Ersatzsystem zum Brückensystem in Abbildung 3.6 zu sehen.

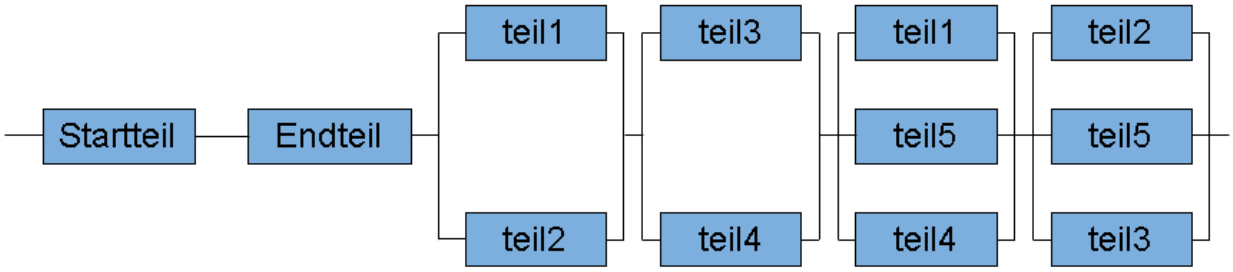

Abbildung 3.7: Serielles Ersatzsystem

Dementsprechend kann aus den minimalen Pfaden die *reduzierte Parallelform* des Netzwerkes erstellt werden. Die Objekte eines minimalen Pfades werden hierfür in Serie verbunden und diese Serien aus den minimalen Pfaden zu einem parallelen Ersatzsystem zusammengeschlossen. In Abbildung 3.8 ist das parallele Ersatzsystem zum Brückensystem aus Abbildung 3.6 dargestellt.

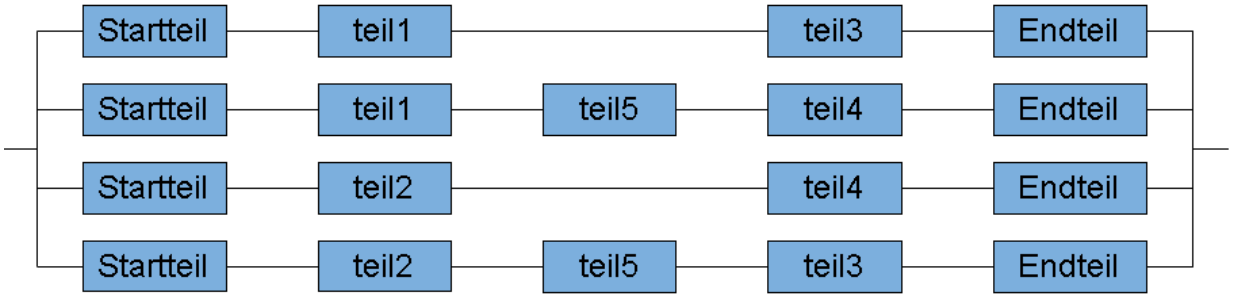

Abbildung 3.8: Paralleles Ersatzsystem

Die Berechnung der Zuverlässigkeit der Ersatzsysteme erlaubt Abschätzungen für das Ausgangssystem [57]. Dabei gibt die Zuverlässigkeitberechnung des seriellen Ersatzsystems eine untere Zuverlässigkeitsabschätzung (worst case) und die des parallelen Ersatzsystems eine obere Zuverlässigkeitsabschätzung des Systems an. Dementsprechend liegt die Zuverlässigkeit des Brückensystems R zwischen der Zuverlässigkeit des seriellen Ersatzsystems $R_{seriell}$ und der Zuverlässigkeit des parallelen Ersatzsystems $R_{parallel}$, wie in der nachfolgenden Gleichung dargestellt ist:[37]

$$R_{seriell} \leq R \leq R_{parallel} \tag{3.9}$$

Der Zuverlässigkeitswert des parallelen Ersatzsystems ist in der Praxis weniger relevant, wodurch das serielle Ersatzsystem als das interessantere anzusehen ist [57].
Auch anhand der Ersatzsysteme können automatisch Fehlerbäume erstellt werden. Der Fehlerbaum des seriellen Ersatzsystems ist in Abbildung 3.9 illustriert. Aus dem paralle-

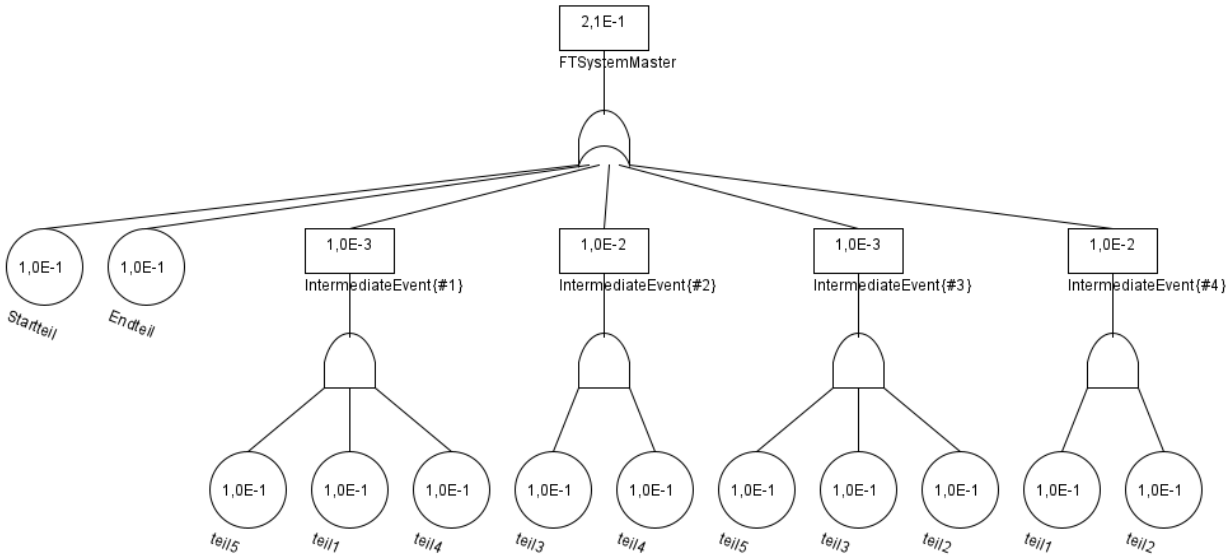

Abbildung 3.9: Fehlerbaum Brückensystem: serielles Ersatzsystem

len Ersatzsystem kann der in Abbildung 3.10 verbildlichte Fehlerbaum generiert werden. Die Zuverlässigkeit aller Komponenten wurde hier beispielhaft auf $R = 0.9$ gesetzt, in den dargestellten Fehlerbäumen ist der gerundete Wert für die Ausfallwahrscheinlichkeit zu sehen, welcher der vorherigen Formel entsprechend beim seriellen Ersatzsystem die

37 In Anhang C wird beispielhaft gezeigt, dass mit der angenommenen Unabhängigkeit der Komponenten die Ersatzsysteme mathematisch äquivalent sind und die analytische Berechnung der Zuverlässigkeit des Ausgangssystems ermöglichen. Diese mathematischen Umformungen sind noch nicht automatisiert umgesetzt, da neben einer partiellen Systemanalyse zusätzlich jederzeit eine vollständige Systemanalyse durchgeführt werden müsste und spezielle Umformungen für die Anwendung im CAS implementiert werden müssten.

untere Schranke der Zuverlässigkeit des Brückensystems angibt, während das parallele Ersatzsystem den oberen Wert angibt.[38]

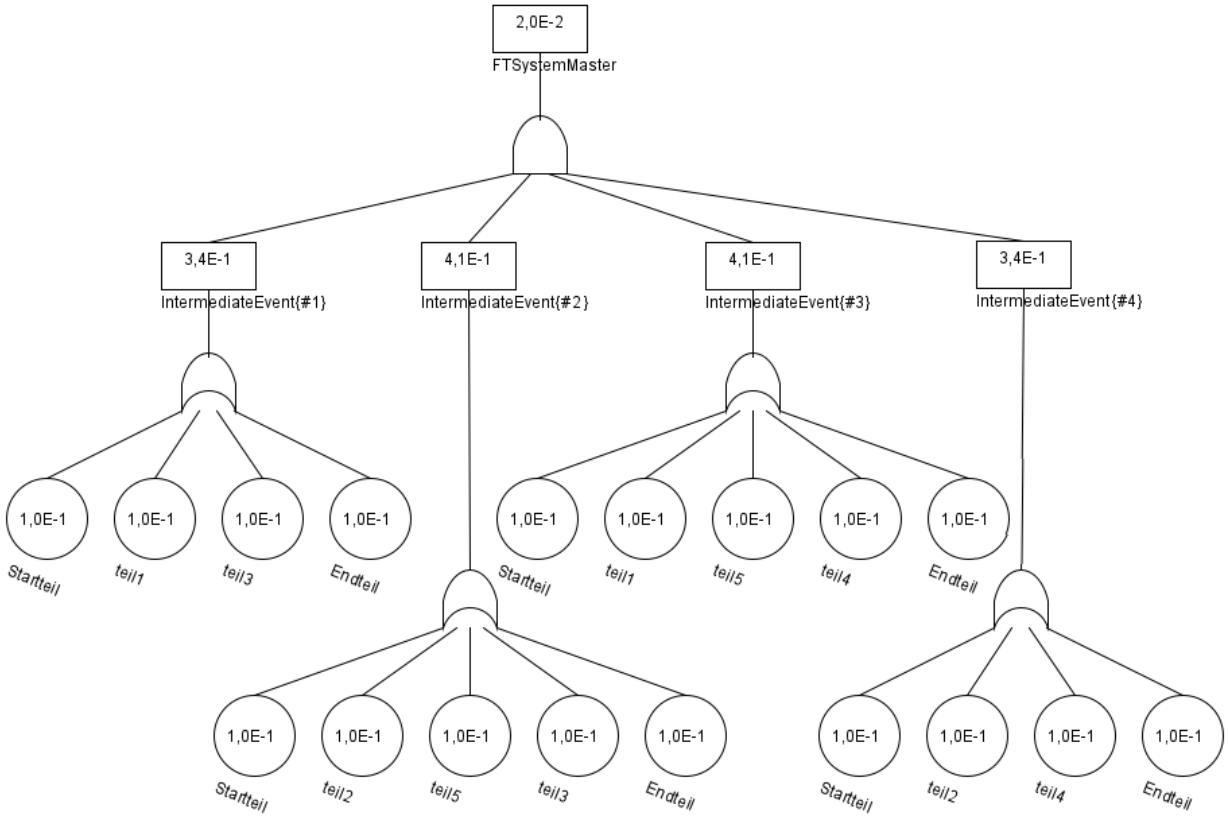

Abbildung 3.10: Fehlerbaum Brückensystem: paralleles Ersatzsystem

3.3 Einbindung in die Fehlermöglichkeits- und -einflussanalyse

In Kapitel 2.1 *Aktuelle Methoden der Zuverlässigkeitsanalyse* wird, neben der hier bereits behandelten Fehlerbaumanalyse, die Fehlermöglichkeits- und -einflußanalyse (FMEA) eingehender beschrieben. Im Rahmen der Automatisierung dieser Risikoanalysemethode wurde in diese die automatische Fehlerbaumanalyse integriert, wie in den folgenden Abschnitten beschrieben wird. Hierbei wird zunächst auf die Integration der FMEA in ein UML-Modell eingegangen und anschließend die Integration der Fehlerbaumanalyse in die FMEA erläutert.

38 Zu beachten ist, dass im Fehlerbaum die Ausfallwahrscheinlichkeit angegeben ist und nicht die Zuverlässigkeit, für die Ausfallwahrscheinlichkeit gilt:

$$F_{seriell} \geq F \geq F_{parallel} \tag{3.10}$$

3.3.1 Automatische Fehlermöglichkeits- und -einflussanalyse

Um eine automatische FMEA auf einem Modell mit den in Kapitel 3.1 beschriebenen Voraussetzungen auszuführen, wird das Modell zunächst wieder semantisch erweitert. Diese Erweiterung des Modells erfolgt mit Vokabeln, die in einem Klassendiagramm für die FMEA, siehe Abbildung 3.11, festgelegt sind. Aus einem entsprechend semantisch erweitertem Modell wird dann die Ausgabe der FMEA erzeugt und beispielsweise in einem Tabellenkalkulationsprogramm dargestellt.[39] Das Klassendiagramm für die FMEA ist dabei nach den Kernelementen der FMEA aufgebaut. Beginnend bei der Funktion des Systems (*Function*) hat diese eine *FMEA*. Die *FMEA* hat Fehlermodi (*FailureMode*) und Fehlermodusobjekte (*FailureModeObject*). Ein Fehlermodusobjekt besteht dabei aus einem Fehlermodus (*FailureMode*), einer Konsequenz (*Consequence*) und einer Ursache (*RootCause*). Im Folgenden werden die einzelnen Klassen und ihre Verknüpfungen detaillierter beschrieben.

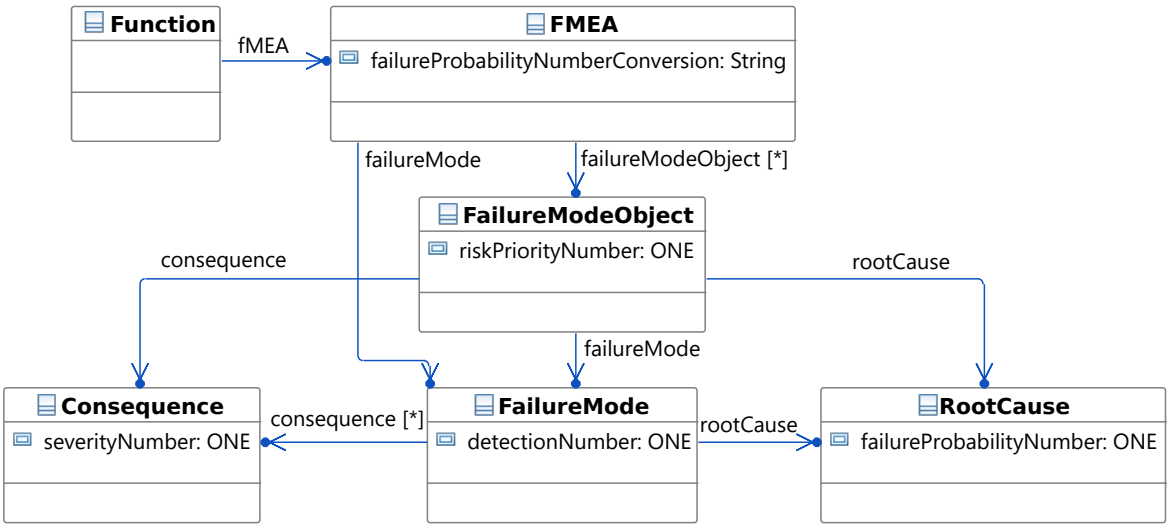

Abbildung 3.11: Klassendiagramm der Fehlermöglichkeits- und -einflussanalyse in UML

Funktion Ein System hat eine Funktion, die über die Klasse *Function* zugewiesen wird. Diese Funktion dient als Ausgangspunkt einer FMEA. Für diese Funktion werden im Rahmen einer FMEA ein Fehlermodus oder mehrere Fehlermodi ermittelt und diese untersucht. Im Klassendiagramm in Abbildung 3.11 ist die Zuordnung der Klasse *FMEA* zur Klasse *Function* dargestellt.

39 Es sei hier darauf hingewiesen, dass keine vollständige Integration von Elementen der FMEA vorgenommen, sondern eine reduzierte FMEA verwendet wurde, anhand derer die Integration der Fehlerbaumanalyse aufgezeigt werden kann.

FMEA Die Klasse *FMEA* beinhaltet die Rahmenbedingungen für die FMEA. Diese bestehen hier aus den Vorgaben für die Umrechnung von Ausfallwahrscheinlichkeiten in die Ausfallswahrscheinlichkeitszahl, wie in Abbildung 3.11 als Attribut *failureProbabilityNumberConversion* zu sehen. Der Unterschied besteht darin, dass eine Ausfallwahrscheinlichkeit für die FMEA in eine Kategorie eingeordnet werden muss, die einer Zahl zwischen 1 und 10 entspricht und für die Berechnung der Risikoprioritätszahl (siehe Fehlermodusobjekt) herangezogen wird. Darüber hinaus hat die FMEA ein Fehlermodusobjekt oder mehrere Fehlermodusobjekte.

Fehlermodusobjekt Das Fehlermodusobjekt, dargestellt durch die Klasse *FailureModeObject*, beschreibt einen Fehlermoduseintrag in der FMEA. Dazu hat ein Fehlermodusobjekt einen Fehlermodus, eine Konsequenz und eine Ursache. Auf Grundlage der Daten aus diesen Instanzen kann die Risikoprioritätszahl (RPZ), die als Attribut *riskPriorityNumber* hinterlegt ist, über die in der Klasse hinterlegte Gleichung

$$RPZ = A * B * E \tag{3.11}$$

berechnet werden. Diese Zahl kann Werte von 1 bis 1000 annehmen. Für die genauere Bedeutung der Risikoprioritätszahl sei auf Kapitel 2.1.2 *Fehlermöglichkeits- und -einflussanalyse (FMEA)* verwiesen.

Fehlermodus Die Klasse *FailureMode* beschreibt den Fehlermodus. Sie hat eine Konsequenz oder mehrere Konsequenzen und das Attribut *detectionNumber*, welches die Entdeckungswahrscheinlichkeit (E) des Fehlermodus angibt. Diese ist entsprechend der FMEA Kategorisierung ganzzahlig von 1 bis 10, wobei 1 als wird instantan entdeckt und 10 als keine Entdeckung als grobe Beschreibung dient. Des Weiteren hat der Fehlermodus eine Ursache oder mehrere Ursachen.

Konsequenz Die Auswirkung eines Fehlermodus wird durch die Klasse *Consequence* dargestellt. Diese Klasse hat das Attribut *severityNumber*, das die Bedeutung (B) der Konsequenz angibt. Entsprechend der FMEA Kategorisierung ist dies eine ganzzahlige Zahl von 1 bis 10, in der zur groben Beschreibung 1 als keine Auswirkung und 10 als katastrophale Auswirkung angesehen werden kann.

Ursache Die Ursache des Fehlermodus wird mit der Klasse *RootCause* beschrieben. Zur Beschreibung der Ausfallwahrscheinlichkeitszahl (A) besitzt diese Klasse das Attribut *failureProbabilityNumber*, welches ganzzahlig von 1 bis 10 gesetzt werden kann. 1 steht hierbei für kein Ausfall und 10 demgegenüber für eine hohe Ausfallwahrscheinlichkeit.

3.3.2 Integrierte Fehlerbaumanalyse

Für eine integrierte Lösung von FMEA und FTA wurden zunächst die Gemeinsamkeiten bzw. Schnittstellen der Methoden identifiziert. Im Rahmen dieser Untersuchung zeigte sich die Auswirkung in der FMEA als Entsprechung des Wurzelereignisses. Ebenfalls war der Zusammenhang zwischen einem Basisereignis und der Wurzelursache offensichtlich, auch im Bezug auf die zugehörigen Eigenschaften der Ausfallwahrscheinlichkeit und der Ausfallwahrscheinlichkeitszahl, die aus dieser ermittelt wird. Diese Verbindung der Klassendiagramme ist in Abbildung 3.12 dargestellt. Durch die Verbindung der Klassendiagramme ist es möglich über entsprechende Regeln nach Abschluss einer Fehlerbaumanalyse eine FMEA durchzuführen, die auf den Ergebnissen der FTA aufbaut und diese in die FMEA integriert. Diese Regeln erstellen zu einem Wurzelereignis - in der UML einer Instanz der Klasse *FTSystemMaster* - eine Auswirkung - in der UML einer Instanz der Klasse *Consequence* - und verbinden diese mit dem Wurzelereignis. Im Rahmen der automatischen Fehlerbaumanalyse wurden für das Wurzelereignis die Basisereignisse bzw. Ereigniskombinationen aus Basisereignissen (minimale Schnitte) bestimmt. Ein minimaler Schnitt entspricht hierbei einer Ursache im Sinne der FMEA. Dieser Zusammenhang wird genutzt, um über Regeln zu den entsprechenden Basisereignisse bzw. Ereigniskombinationen aus Basisereignissen - in der UML Instanzen der Klasse *FaultTreeObject* - Ursachen - in der UML Instanzen der Klasse *RootCause* - zu erstellen und diese miteinander zu verbinden. Zusätzlich wird die Verbindung zwischen der Auswirkung und der Ursache erstellt. Durch die (regelbasierte) Verbindung der Auswirkung zu einem Fehlermodus (Instanz der Klasse *FailureMode*) werden die Ergebnisse der FTA dann in eine FMEA integriert. Diese integrierte FTA in eine FMEA wird im Kapitel 4 *Anwendungsbeispiele* am Beispiel eines Schiffsmotors demonstriert.
Betrachtet man das Vorgehen der Erstellung der FMEA aus der FTA, ist der gegensätzliche Ansatz der Bottom-Up-Methode der FMEA gegenüber der Top-Down-Methode der FTA anhand des Systems für Einzelausfälle im sich überschneidenden Bereich[40] nicht zu unterscheiden. Dies ist mit der kontrafaktischen Kausalität erklärbar, die aus dem Ausbleiben der Wirkung das Ausbleiben der Ursache ableitet.

3.3.3 Unterschiede zur Fehlermöglichkeits- und -einflussanalyse

Die hier für die Modellbildung beschriebene FMEA beinhaltet alle wichtigen Kernelemente einer FMEA. Insbesondere wurde Wert darauf gelegt eine Anbindung zur Fehlerbau-

40 Dies bezieht sich auf die Verknüpfungen der Ursachen und Wirkungen in der jeweiligen Darstellung.

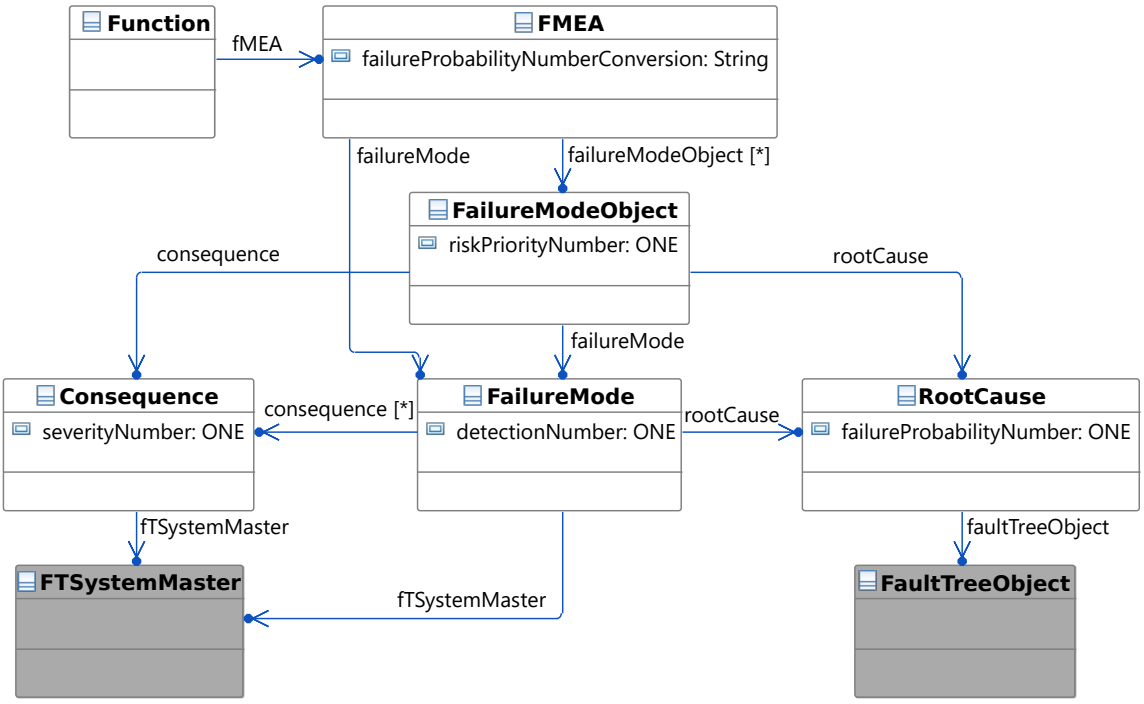

Abbildung 3.12: Klassendiagramm der Fehlermöglichkeits- und -einflussanalyse mit Integration der Fehlerbaumklassen in der UML

manalyse zu schaffen und die dafür als notwendig erachteten Elemente zu modellieren. Dadurch wurden speziell mathematisch verknüpfbare Elemente eingepflegt. Arbeitsanweisungen und Änderungsvorschläge wurden neben anderen bewusst ausgespart, da dies beispielsweise als topologische Änderung über den Einsatz von Entwurfssprachen direkt in diese mit integriert wäre und sich nicht in der FMEA Ausgabe zeigen würde.
Im vorherigen Abschnitt *Integrierte Fehlerbaumanalyse* wurde erwähnt, dass einer Ursache minimale Schnitte bestehend aus Basisereignissen zugeordnet werden. Ein minimaler Schnitt kann hierbei entweder ein oder mehrere Basisereignisse enthalten. Es widerspricht allerdings der FMEA einer Auswirkung mehrere Ursachen in einem Analyseschritt zuzuordnen. Somit ist im Sinne der FMEA die Zuordnung eines minimalen Schnittes nur dann möglich, sofern dieser genau ein Basisereignis beinhaltet. Beinhaltet der minimale Schnitt mehrere Basisereignisse, so ist dies in der FMEA nur dann zu betrachten, wenn diese Ereigniskombination als ein Ereignis dargestellt werden kann. Diese Möglichkeit der Zusammenführung der Ereignisse zu einem Ereignis wird hier angenommen.
Zur Verdeutlichung wird das System des einfachen Schaltkreises aus Abbildung 3.3 herangezogen. Über die quantitative Fehlerbaumanalyse werden für das modellierte System drei minimale Schnitte bestimmt: $S_1\{Energiequelle\}$, $S_2\{Lampe\}$ und $S_3\{Schalter1, Schal-$

ter2}. Die minimalen Schnitte S_1 und S_2 beinhalten jeweils lediglich ein einziges Basisereignis und sind deshalb direkt mit der FMEA vereinbar. Der minimale Schnitt S_3 hingegen beinhaltet zwei Basisereignisse, wodurch dieser in einer FMEA nicht direkt berücksichtigt wird.[41] Wird die Ereigniskombination des minimalen Schnittes S_3 als Parallelsystem dargestellt, so ist das Auftreten der Ursache, in diesem Fall der Ausfall des Parallelsystems, im Kontext des Detailgrades des Modells als eine einzige Ursache anzusehen, die wiederum mit der FMEA vereinbar ist.

3.4 Zusammenfassung und Zielsetzung

In diesem Kapitel wurden die grundlegenden Bausteine für eine automatische Zuverlässigkeitsanalyse beschrieben. Dazu gehört die entsprechende Wissensmodellierung durch Erstellung einer Ontologie - hier grafisch in Form von Klassendiagrammen dargestellt -, mit deren Hilfe Modelle semantisch annotiert werden können. Über die semantische Annotation eines Modells mit entsprechenden Modellelementen der Zuverlässigkeitsanalyse ist eine Zuverlässigkeitsberechnung und -abschätzung mit Fehlerbaumerstellung ermöglicht worden. Aufbauend auf einer automatischen qualitativen und quantitativen Fehlerbaumanalyse ist des Weiteren eine Risikoanalyse mit der Methode der Fehlermöglichkeits- und -einflussanalyse aufgezeigt worden.
Die Bausteine bis zur qualitativen und quantitativen Fehlerbaumanalyse sind für die Automatisierung der Fehlerbaumanalyse notwendig, zur vollständigen automatischen Fehlerbaumanalyse fehlen allerdings noch die nachfolgenden Schritte. Diese im Kapitel 2.1.1 *Fehlerbaumanalyse (FTA)* beschriebenen Schritte stellen Maßnahmen dar, die in iterativen Schritten das (Ausgangs-)Systemmodell verändern auf Grundlage der in der qualitativen und quantitativen Fehlerbaumanalyse ermittelten Ergebnisse. Diese iterativen auf den Ergebnissen aufbauenden Schritte werden direkt anhand eines Anwendungsbeispiels im folgenden Kapitel beschrieben.[42]

41 In der FMEA wird die Wirkung bei Eintreten einzelner Fehler (Ursachen) untersucht.
42 Der Begriff der *Fehlerbaumanalyse* kann - wie bereits beschrieben - verschiedene Bedeutungen haben, die hier nochmal differenziert werden: Steht der Begriff *Fehlerbaumanalyse* für sich, ist die Methode gemeint. Mit dem Zusatz eines Adjektivs, wie qualitativ oder quantitativ, ist die Analyse des Fehlerbaumes gemeint, die Teil der Methode ist.

3.5 Weiterführende Fragen

Über die in diesem Kapitel beschriebenen Teile der automatischen Fehlerbaumerstellung und -analyse hinaus, sind noch verschiedene, weitere, thematisch nahe Fragen von Interesse. Neben der Integration weiterer Elemente der Fehlerbaumanalyse, wie beispielsweise Transfergatter[43], die zu einer übersichtlicheren Darstellung beitragen, oder dem automatischen Erkennen für die Fehlerbaumanalyse irrelevanter Bereiche, die nicht weiter untersucht werden müssen, da beispielsweise die Ausfallwahrscheinlichkeit gegenüber der zu erwartenden Gesamtausfallwahrscheinlichkeit als vernachlässigbar eingeschätzt wird, ist vor allem für komplexe Strukturen wie Brückensysteme eine analytische Lösung, beispielsweise über die Schlüsselwertmethode[44], interessant. Über die hier erforderlichen, unidirektional gerichteten[45], zeitinvarianten[46] Systeme hinaus sind Erweiterungen für zeitabhängige Systeme, hin zu dynamischen Fehlerbäumen, sowie für Kreisschlüsse in Systemen, beispielsweise durch bidirektionale Verbindungen, weitere Ansatzpunkte zum Ausbau der automatischen Fehlerbaumanalyse. Auch ist der Ansatz der Verknüpfung der Fehlerbaumanalyse mit der Fehlermöglichkeits- und -einflussanalyse ein Aspekt, der zur weiteren Untersuchung einlädt. Es bieten sich sowohl kleinere als auch größere Fragestellungen an, mit denen die beschriebenen Werkzeuge erweitert werden können, um die Einsatzmöglichkeiten zu vergrößern.

43 Transfergatter erlauben es sich wiederholende (Teil-)Fehlerbäume darzustellen und diese entsprechend in den Fehlerbaum an unterschiedlichen Stellen schneller, übersichtlicher und konsistenter zu integrieren. Für die dargestellte Automatisierung bedeutet eine Integration lediglich den Vorteil der verbesserten Übersichtlichkeit.

44 Das System wird hierbei aufgeteilt in Teilsysteme in denen der *Schlüsselwert* (z.B. ein Ereignis, eine Komponente) einmal mit einer Zuverlässigkeit von $R = 1$ und einmal mit $R = 0$ berechnet wird.[13]

45 keine Kreisschlüsse

46 Systemfunktion unverändert

Kapitel 4

Anwendungsbeispiele

Eine Demonstration der Anwendung der erstellten automatischen Zuverlässigkeitsanalyse mittels Fehlerbäumen und der Fehlermöglichkeits- und -einflussanalyse erfolgt in diesem Kapitel an zwei Beispielen - einem Schifffahrtsmotor und verschiedenen Satellitenantriebssystemen. Diese Beispiele decken verschiedene, wichtige Aspekte der automatischen Zuverlässigkeitsanalyse - im speziellen der Fehlerbaumanalyse - ab.
Im Anwendungsbeispiel des Schifffahrtsmotors wird ein System mit verschiedenen Fehlermodi von Grund auf unter Berücksichtigung einer Zuverlässigkeitsberechnung in einer Entwurfssprache modelliert. Für dieses Systemmodell wird die Zuverlässigkeitsberechnung automatisch ausgeführt und ein zugehöriger Fehlerbaum erstellt. Des Weiteren wird an diesem Beispiel eine modulare Integration des Schiffsmotors in ein Schifffahrtsantriebssystem unter Berücksichtigung der Zuverlässigkeitsberechnung, sowie die Integration der automatischen Fehlerbaumanalyseergebnisse in die Fehlermöglichkeits- und -einflussanalyse demonstriert.
Das zweite Anwendungsbeispiel betrachtet ein mit verschiedenen generierten Satellitenantriebssystemen komplexeres System, das bereits in einer Entwurfssprache modelliert wurde und in das die Fehlerbaumanalyse nachträglich integriert wird. Diese Möglichkeit wird anhand verschiedener Antriebssysteme aufgezeigt und die Einbindung erläutert. Darüber hinaus wird eine automatische Verbesserung hinsichtlich der Zuverlässigkeit durch Anpassung der Parametrik und Topologie eines Satellitenantriebssystems demonstriert, welche die Integration und Anwendung einer automatischen Fehlerbaumanalyse in einen automatisierten Entwurfsprozess mit Entwurfssprachen zeigt. Zusätzlich wird anhand dieses Anwendungsbeispieles eine Entwurfsraumuntersuchung im Sinne einer Design of experiments (DOE) Studie demonstriert. Abschließend werden in diesem Kapitel die vorgestellten Beispiele in Bezug auf Vollständigkeit und Konsistenz diskutiert.

4.1 Schiffsmotor

Anhand des Beispiels des Schiffsmotors wird eine direkte Integration der Zuverlässigkeitsanalyse im Rahmen der Modellierung demonstriert. Dazu wird die automatische Fehlerbaumerstellung gezeigt und neben dieser die Integration einer qualitativen Fehlermöglichkeits- und -einflussanalyse erläutert. Die Teilmodellierungen zeigen zudem verschiedene Möglichkeiten zur Verknüpfung der Systemkomponenten, einen modularen Aufbau zur partiellen Auslegung, sowie eine Detaillierung bis zu einzelnen Fehlermodi durch eine entsprechende Erweiterung der Systemgrenzen.

4.1.1 Modellierung des Schiffsmotors

Ein Schiffsmotor besteht funktional aus einem Einspritzsystem, einem Zylinderblock, einem Kolbensystem und einer Kurbelwelle. Diese Teilsysteme bestehen ihrerseits aus weiteren Komponenten, deren Detaillierungsgrad die gewählte Systemgrenze festlegen. Im Folgenden wird die Detaillierung bis hin zu einzelnen Fehlermodi beispielhaft am Kolben als Komponente des Kolbensystems, das eine Komponente des Schiffsmotors ist, gezeigt. Die Modellierung des Schiffsmotors in einer Entwurfssprache spiegelt das beschriebene System entsprechend wider. Angefangen beim Klassendiagramm, zu sehen in Abbildung 4.1, das die Schiffsmotorklasse *Engine* zeigt, die eine Assoziation *enginePart* zur Schiffsmotorteileklasse *EnginePart*[1] aufweist. Die Schiffsmotorteileklasse erbt - hinterlegt in einem weiteren Klassendiagramm - von der Klasse *FaulTreeObject* aus dem Zu-

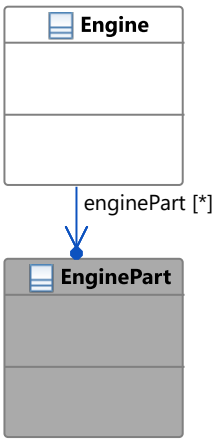

Abbildung 4.1: Klassendiagramm Schiffsmotor[2]

1 Die Teileklassen (Namensendung auf ...*Part*) stellen hier abstrakte Klassen dar, die zur besseren Erweiterbarkeit eingeführt wurden.
2 Die grau hinterlegten Klassen sind importierte Klassen aus anderen Klassendiagrammen.

verlässigkeitsklassendiagramm, welches im vorherigen Kapitel 3 *Automatische Fehlerbaumerstellung und -analyse* vorgestellt wurde[3]. Auf Grund dieser Vererbung ist jede Instanz der Klasse *EnginePart* ein Element der Zuverlässigkeitsanalyse und semantisch annotiert, wodurch diese Instanzen im Rahmen der Ausführung der automatischen Zuverlässigkeitsanalyse algorithmisch erkannt und berücksichtigt werden.

Das Aktivitätsdiagramm der Schiffsmotorentwurfssprache ist in Abbildung 4.2 dargestellt und zeigt den Programmablauf der Modellierung. Es besteht aus einer Regel *CreateEngine*, einem Unterprogramm *AddMainEngineReliabilityAnalysis*[4] und zwei Schnittstellenaufrufen für die Zuverlässigkeitsanalyse *Reliability Analysis* und die Fehlerbaumerstellung *Fault Tree*. Die Regel *CreateEngine* erstellt gemäß dem Namen eine Instanz der Schiffsmotorklasse *Engine*. Das Unterprogramm *AddMainEngineReliabilityAnalysis* beschreibt die zugehörigen Komponenten des Schiffsmotors, sowie deren funktionale Verknüpfungen.

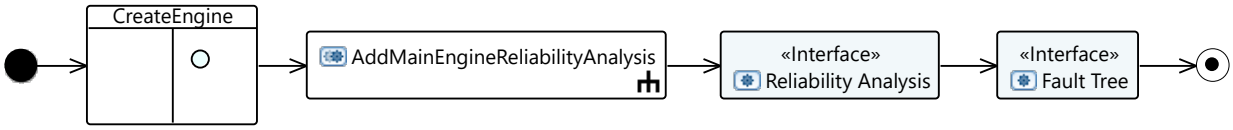

Abbildung 4.2: Aktivitätsdiagramm Schiffsmotor

In Abbildung 4.3 ist der Programmablauf des Unterprogramms *AddMainEngineReliabilityAnalysis* zu sehen. In diesem werden zunächst die Komponenten Einspritzsystem (fuel injection system), Zylinderblock (cylinder block), Kurbelwelle (crankshaft rod) und Kolbensystem (piston system) erstellt und mit der Instanz des Schiffsmotors verknüpft. Danach wird die funktionale Struktur der Komponenten im Unterprogramm *SetFunctionalFlow* modelliert und abschließend im Unterprogramm *AddSubsystemReliabilityAnalysis* eine weitere Detaillierung bezüglich der Zuverlässigkeit für die einzelne Komponenten des Schiffsmotors vorgenommen.

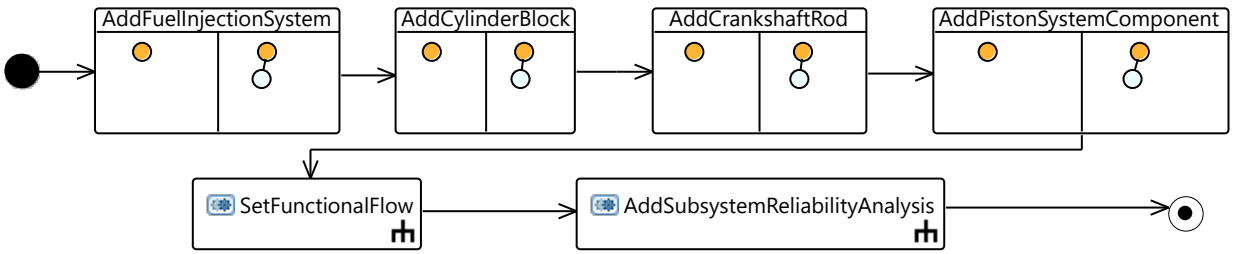

Abbildung 4.3: Unterprogramm Schiffsmotor *AddMainEngineReliabilityAnalysis*

3 siehe Abbildung 3.2
4 Der Schiffsmotor wird hier als Hauptmotor (main engine) bezeichnet, um eine Unterscheidung zu weiteren Motoren zu gewährleisten.

Abbildung 4.4 zeigt den Programmablauf (*SetFunctionFlow*) zur Erstellung des funktionalen Zusammenhangs der Komponenten untereinander. Die einzelnen Komponenten des Schiffsmotors werden durch die ersten drei Regeln funktional in einer Serie angeordnet. In der Regel *CreateEngineSystemFailure* wird zur Instanz des Schiffsmotors eine zugehörige Instanz der Klasse *FTSystemMaster* erstellt, die den Ausfall des Schiffsmotors im Modell abbildet. Abschließend werden die Komponenten mit dieser Instanz in der Regel *ConnectComponentsWithFTSystemMaster* verknüpft.

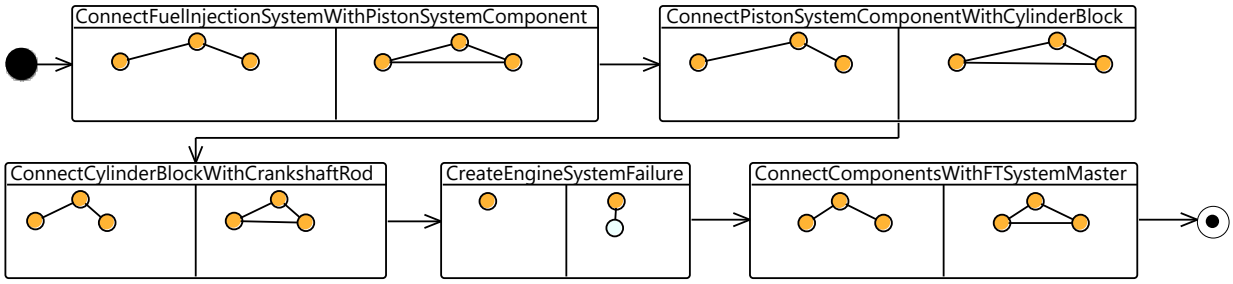

Abbildung 4.4: Unterprogramm Schiffsmotor *SetFunctionFlow*

Im Unterprogramm zur weiteren Detaillierung der Schiffsmotorkomponenten *AddSubsystemReliabilityAnalysis* werden die jeweiligen Programme innerhalb der Entwurfssprachen der Komponenten aufgerufen, welche analog - unter Berücksichtigung der internen Komponenten - des Programmablaufs des Unterprogramms *AddMainEngineReliabilityAnalysis* aufgebaut sind. In Abbildung 4.5 ist der Programmablauf dargestellt.

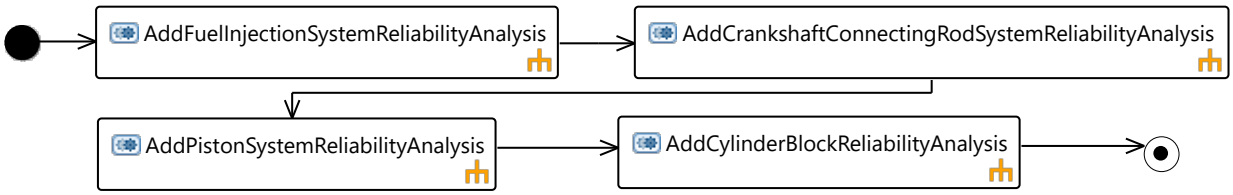

Abbildung 4.5: Unterprogramm Schiffsmotor *AddSubsystemReliabilityAnalysis*[5]

Kolbensystem

Die detaillierte Betrachtung eines Teilsystems des Schiffsmotors erfolgt am Kolbensystem. Dieses System besteht aus den Komponenten Kolbenring, Kolben und Zylinderrohr.

Im Klassendiagramm des Kolbensystem, dargestellt in Abbildung 4.6, ist die Kolbensystemklasse *PistonPistonRingCylinderLinerComponent* dargestellt. Diese Klasse ist ein Teil

5 Die mit einem orangen (in schwarz-weiß: grauen) Symbol unten rechts gekennzeichneten Unterprogramme in den Unterprogrammvisualisierungen zeigen analog zu den Klassen an, dass diese aus einer anderen Entwurfssprache importiert wurden.

des Schiffsmotors, umgesetzt durch die entsprechende Vererbung von der Klasse *EnginePart*, und hat Komponenten, die das Kolbensystem realisieren, im Modell durch die Assoziation zur Kolbensystemteileklasse *PistonSystemPart*[6] verwirklicht.

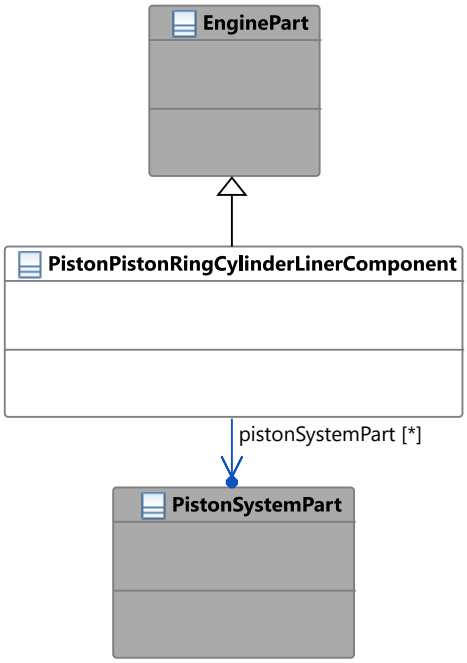

Abbildung 4.6: Klassendiagramm Kolbensystem

Im Programm *AddPistonSystemReliabilityAnalysis*, zu sehen in Abbildung 4.7, das durch das Programm *AddSubsystemReliabilityAnalysis* aus dem Programm der Entwurfssprache des Schiffsmotors aufgerufen wird (siehe Abbildung 4.5), werden die Komponenten des Kolbensystems in den *Add*-Regeln erstellt und mit der zuvor erstellten Instanz des Kolbensystems verknüpft. Die *Create*-Regel erstellt zur Instanz des Kolbensystems eine Instanz der Klasse *FTSystemMaster* und ordnet diese dieser zu. Die Instanz der Klasse *FTSystemMaster* wird in der *Connect*-Regel mit den Komponenten des Kolbensystems verknüpft.

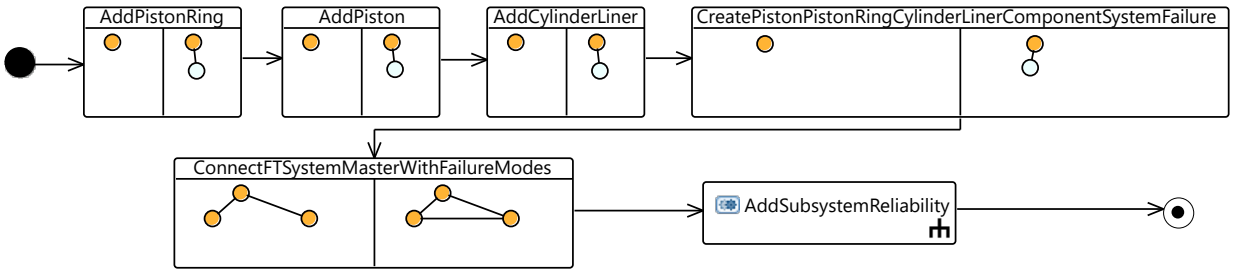

Abbildung 4.7: Aktivitätsdiagramm Kolbensystem *AddPistonSystemReliabilityAnalysis*

6 Die Klasse *PistonSystemPart* erbt ebenfalls von der Zuverlässigkeitsanalyseklasse *FaultTreeObject*.

Abschließend wird das Unterprogramm *AddSubsystemReliability* aufgerufen, welches analog zur übergeordneten Beschreibung des Schiffsmotors die weitere Detaillierung der Komponenten des Kolbensystems beinhaltet. Dieses ist in Abbildung 4.8 dargestellt.

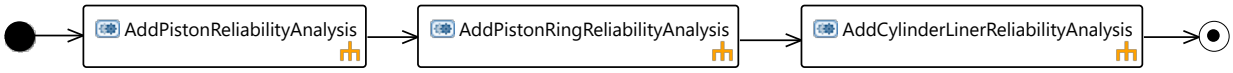

Abbildung 4.8: Aktivitätsdiagramm Kolbensystem *AddSubsystemReliability*

Im Gegensatz zum Programm in der Entwurfssprache des Schiffsmotors wird beim Programm der Entwurfssprache des Kolbensystems beispielhaft kein explizit über Verknüpfungen modellierter funktionaler Zusammenhang der Komponenten angegeben. Die Zuordnung der Komponenten, die Zuverlässigkeitsobjekte sind, zur Instanz der Klasse *FTSystemMaster*, die für diese funktionale Einheit erstellt wurde, fasst diese im Rahmen der automatischen Zuverlässigkeitsanalyse und Fehlerbaumerstellung zu einer modularen funktionalen Einheit zusammen. Da innerhalb dieser funktionalen Einheit für die Komponenten kein definierter funktionaler Zusammenhang erstellt wurde, werden diese Komponenten algorithmisch jeweils einzeln als Ereignis für je eigene, zugehörige Fehlerbäume betrachtet. Diese Fehlerbäume werden für die funktionale Einheit zu einem Fehlerbaum zusammengefasst. Die Zusammenfassung der einzelnen Fehlerbäume folgt der Handhabung wie bei Seriensystemen, sofern nicht explizit die Eigenschaft auf kausal abhängige bzw. parallele Berücksichtigung[7] gesetzt wurde.[8]
In der Entwurfssprache des Kolbensystems ist über den geometrischen Zusammenhang ein entsprechender funktionaler Zusammenhang gegeben, der in diesem Fall einem seriellen System entspricht. Eine diesbezügliche Erweiterung und (angepasste) logisch funktionale Verknüpfung würde durch die Integration und Erweiterung von Geometrie in abstrakter Form mit einer entsprechend erstellten Regel in die Zuverlässigkeitsanalyse und Fehlerbaumerstellung automatisch mit einbezogen werden und die generische oben beschriebene (serielle) Handhabung an den betroffenen Stellen ersetzen.

Kolben

Die weitere Detaillierung anhand der Komponente Kolben zeigt die hier gewählte Systemgrenze der Zuverlässigkeitsanalyse des Schiffsmotors auf. Hier wird für die Komponente Kolben ein Fehlermodus erstellt.

7 siehe Erläuterungen zu *FTSystemMaster* in Kapitel 3.2
8 Die standardmäßige Handhabung als Seriensystem ist auf Grund der niedrigeren Zuverlässigkeit von Serien- gegenüber Parallelsystemen gewählt worden, um hier eine untere Grenze abzubilden. Dass dies ein legitimer Ansatz ist, zeigt sich in der nachfolgenden, weiteren Detaillierung, die bis auf einzelne Fehlermodi herunterreicht.

Im Klassendiagramm der Entwurfssprache des Kolben, welches in Abbildung 4.9 zu sehen ist, erbt die Kolbenklasse *Piston* von der Komponentenklasse des Kolbensystems *PistonSystemPart*, womit es eine Komponente des Kolbensystems ist. Neben der Komponentenklasse ist eine Systemfehlerklasse *PistonSystemFailure* erstellt worden, die von der Systemfehlerklasse *FTSystemMaster* aus dem verknüpften Zuverlässigkeitsanalyseklassendiagramm erbt. Die Klasse *PistonFailureEvent* repräsentiert die Fehlermodi und erbt dementsprechend von der Fehlereventklasse *FailureEvent*, die von der Fehlerobjektklasse *FaultTreeObject* aus dem Klassendiagramm der Zuverlässigkeitsanalyse erbt und eine Namensbezeichnung als zusätzliche Eigenschaft hat, damit im Fehlerbaum explizit Namen erstellt werden. Im dargestellten Klassendiagramm sind keine Assoziationen zwischen den Klassen sichtbar, allerdings sind durch die Verknüpfungen innerhalb der verknüpften Klassendiagramme die Klassen miteinander verknüpft. Dies zeigt sich in den Regeln der Entwurfssprache.

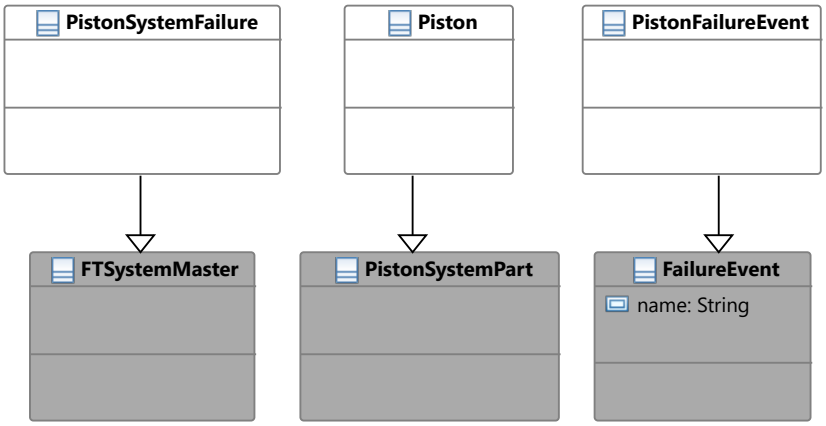

Abbildung 4.9: Klassendiagramm Kolben

In den Programmablauf der Entwurfssprache zur Modellierung des Kolbensystems ist das Programm *AddPistonReliabilityAnalysis* aus der Entwurfssprache zur Modellierung des Kolben integriert, wie in Abbildung 4.8 zu sehen. Das Aktivitätsdiagramm zum Programmablauf *AddPistonReliabilityAnalysis* ist in Abbildung 4.10 dargestellt. In dessen Ablauf wird zunächst der Systemfehler als Instanz der Klasse *PistonSystemFailure* in der Regel *CreatePistonSystemFailure* erstellt, nachfolgend wird ein Fehlermodus als Instanz der Klasse *PistonSystemFailure* durch die *Add*-Regel erstellt und mit diesem verknüpft. Anschließend wird die Ausfallwahrscheinlichkeit des Fehlermodus im abschließenden Unterprogramm *SetFailureProbability* gesetzt.

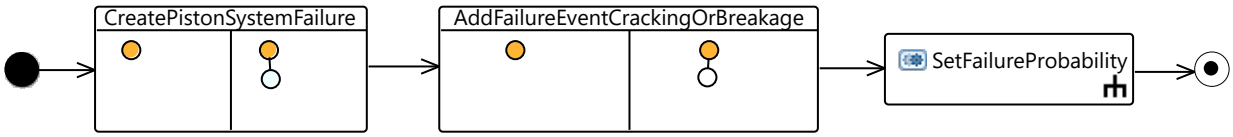

Abbildung 4.10: Aktivitätsdiagramm Kolben *AddPistonReliabilityAnalysis*

Abbildung 4.11 zeigt die Regel[9] *CreatePistonSystemFailure* zur Erstellung des Systemfehlers der Komponente Kolben. Die Instanz der Komponente Kolben wird mit einem untergeordnetem Fehlerbaum verknüpft.[10] Diese Verknüpfung ordnet einem Fehlerbaumobjekt einen Fehlerbaum zu, der modular in einen übergeordneten Fehlerbaum integriert werden kann.[11]

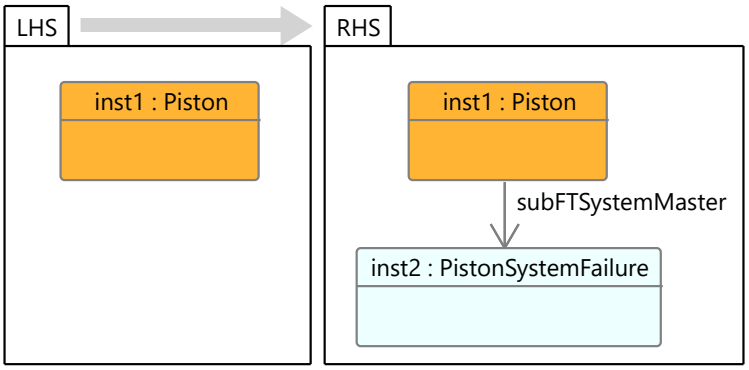

Abbildung 4.11: Regel Kolben *CreatePistonSystemFailure*[12]

In der Regel *AddFailureEventCrackingOrBreakage* wird zum Systemfehler ein Fehlermodus[13] *CrackingOrBreakage* erstellt und dieser Fehlermodus mit dem Systemfehler verknüpft. Die Regel ist in Abbildung 4.12 zu sehen.

Im abschließenden Unterprogramm *SetFailureProbability*, zu sehen in Abbildung 4.13, wird die Ausfallwahrscheinlichkeit des Fehlermodus entsprechend der Randbedingungen gesetzt. Die Randbedingung der Missionszeit bzw. Laufzeit des Motors oder des ent-

9 Prinzipielle Erläuterungen zu grafischen Regeln sind im Abschnitt 2.2.1 Graphgrammatik beschrieben.

10 Der Link *subFTSystemMaster* ist eine Instantiierung der Assoziation zwischen der Fehlerbaumobjektklasse *FaultTreeObject* und der Systemfehlerklasse *FTSystemMaster* aus dem Zuverlässigkeitsanalyseklassendiagramm. Die Kolbenklasse *Piston* ist ein Teil des Kolbensystems *PistonSystemPart*, welcher ein Fehlerbaumobjekt *FaultTreeObject* ist.

11 Diese Angabe zur Integration eines Moduls sollte allgemein von einem Domänenexperten begleitet werden, damit diese funktional und kausal korrekt ist. Hier sind die Komponenten unabhängig, wodurch die modulare Integration wie vorgestellt möglich ist.

12 Die Grundlagen zu den Regeln sind im Kapitel 2.2.1 Graphgrammatik ausführlicher beschrieben. Kurzanleitung zur Farbgebung: hell/hellblau -> neue Instanz (nur RHS), mittel dunkel/orange -> vorliegende Instanz (auf LHS und RHS), dunkel/rot -> entfernte Instanz (nur LHS)

13 In diesem Fall ist lediglich ein Fehlermodus erstellt worden, aber wie in nachfolgend dargestellten Fehlerbäumen zu sehen sind für andere Komponenten mehrere, unterschiedliche Fehlermodi umgesetzt.

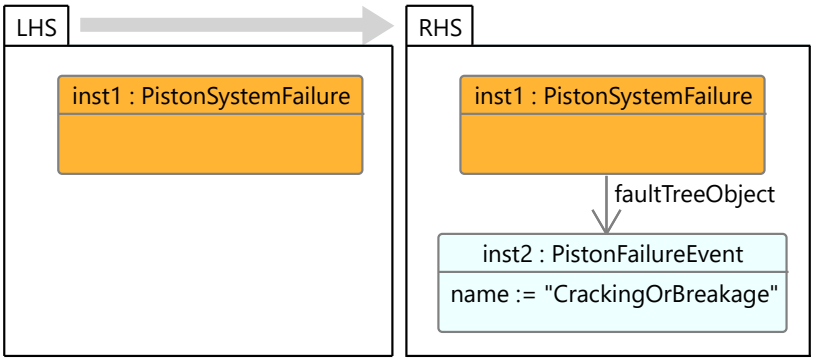

Abbildung 4.12: Regel Kolben *AddFailureEventCrackingOrBreakage*[14]

sprechend übergeordneten Systemfehlers wird in einer Regel überprüft und danach der entsprechende Programmablauf zur Festlegung der Ausfallwahrscheinlichkeit gewählt.

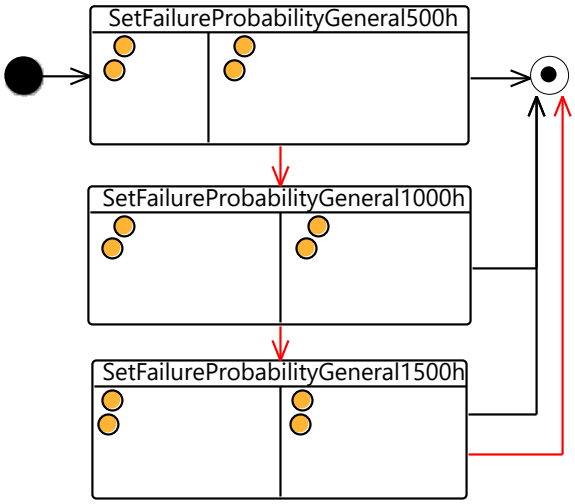

Abbildung 4.13: Aktivitätsdiagramm Kolben *SetFailureProbability*[15]

In Abbildung 4.14 ist beispielhaft die Regel *SetFailureProbabilityGeneral500h* für eine Missionszeit von 500 Stunden dargestellt, in der die Auftretenswahrscheinlichkeit[16] des Fehlermodus *CrackingOrBreakage* des Kolben mit der Eigenschaft *failureProbability* auf den Wert $F = 0.003$ gesetzt wird.[17] Im nachfolgenden Abschnitt wird aufgezeigt wie der

14 Das zugehöriges Klassendiagramm ist in Abbildung 4.9 zu sehen.
15 Rot/grau farbige Kontrollflüsse im Programmablauf werden verwendet, sofern die Regel nicht ausgeführt wird. Hier wenn die Eigenschaft der Missionszeit nicht mit dem gesuchten Wert übereinstimmt.
16 Bei einer Komponente wird die Ausfallwahrscheinlichkeit angegeben, ein Fehlermodus bzw. ein Fehlerereignis hat eine Auftretenswahrscheinlichkeit. Im Rahmen der automatischen Zuverlässigkeitsanalyse werden diese beide mit der Eigenschaft *failureProbability* gesetzt und in die Berechnung integriert.
17 Die Zahlenwerte sind dem Paper von Dong [31] entnommen und werden in diesem Fall trotz der Vorgabe einer Einsatzzeit nicht über eine Fehlerrate berechnet, da hier experimentelle Werte für bestimmte Einsatzzeiten als Datengrundlage verwendet werden.

Fehlerbaum erstellt wird und die gesetzten Werte für die Berechnung der Ausfallwahrscheinlichkeit des Schiffsmotors in diese integriert sind.

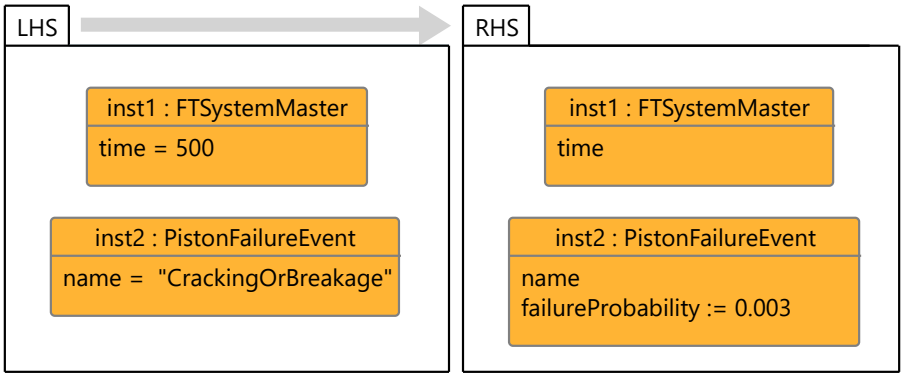

Abbildung 4.14: Regel Kolben *SetFailureProbabilityGeneral500h*[18]

4.1.2 Fehlerbaumerstellung für den Schiffsmotor

In der Entwurfssprache des Schiffsmotors kann über eine Regel die Randbedingung der Einsatzzeit des Schiffsmotors festgelegt werden. Hier werden abhängig von der Einsatzzeit (500, 1000 oder 1500 Stunden) unterschiedliche Entwurfsgraphen erstellt, die in diesem Fall zu quantitativ unterschiedlichen Fehlerbäumen führen. Abbildung 4.15 zeigt einen automatisch generierten Fehlerbaum für den Schiffsmotor für eine Einsatzzeit von 500 Stunden. Der Fehlerbaum setzt sich aus den einzelnen Fehlerbäumen der Komponenten zusammen. Beginnend beim Wurzelereignis, welches der Systemfehler des Schiffsmotors ist, sind in der zweiten Ebene des Fehlerbaumes die Systemausfälle der Komponenten des Schiffsmotors zu finden. In den darauffolgenden Ebenen sind die weiteren Detaillierungen bis zur Systemgrenze dargestellt. Das Fehlerbaumelement des zuvor detaillierter beschriebenen Kolbensystems betrachtend, welches auf der rechten Bildseite zu sehen ist, folgen die Ausfälle der Komponenten des Systems Kolbenring, Kolben oder Zylinderrohr. Diese Ausfälle werden jeweils durch einen Fehlermodus ausgelöst, im Falle des Kolben durch einen Bruch oder Riss des Kolben.

Die Auswertung des in Abbildung 4.15 dargestellten Fehlerbaumes ergibt qualitativ, dass jedes Basisereignis ein minimaler Schnitt und somit ein Einzelfehler ist also kausal für das Auftreten des Systemfehlers. Dies ist basierend auf dem zu Grunde liegenden reinen

18 Eigenschaften (*UML properties* die auf der Suchseite (LHS) verwendet werden (hier: Gleichzeichen „=“), sind ebenfalls auf der Änderungsseite (RHS) dargestellt, allerdings dort ohne Wert, wenn dieser nicht neu gesetzt werden soll (hier: Zuweisung „:=“). Dies ist aus Gründen der Übersicht, um z.B. neu gesetzte von unveränderten (aber gesuchten) Werten zu unterscheiden, dementsprechend umgesetzt.

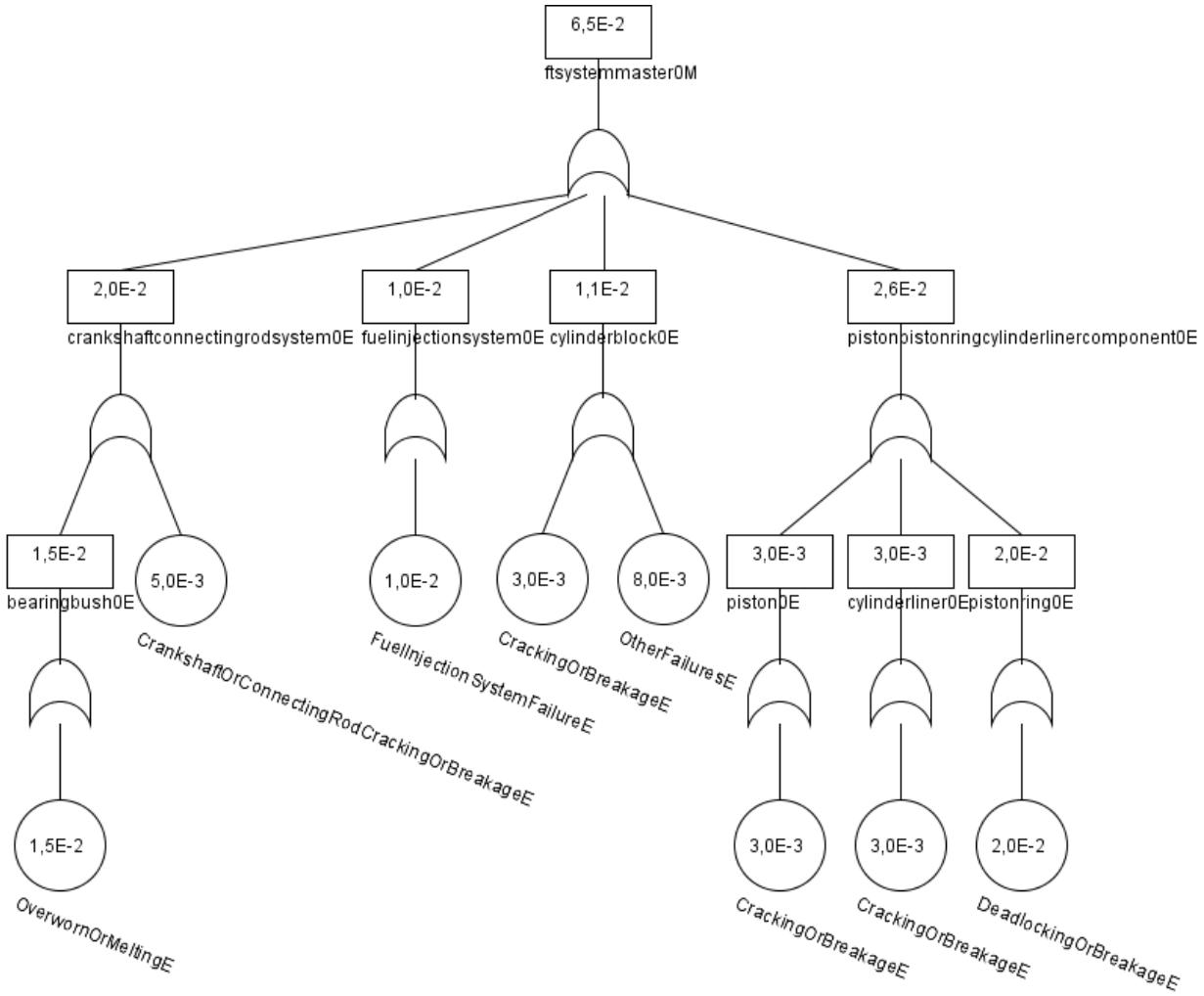

Abbildung 4.15: Fehlerbaum Schiffsmotor 500 Stunden[19]

Seriensystem einfach nachvollziehbar und erwartbar. Die quantitative Berechnung mit den Werten aus [31] ergibt eine (gerundete) Ausfallwahrscheinlichkeit des Schiffsmotors von $F \approx 6.5 \cdot 10^{-2}$ für 500 Stunden Laufzeit. Weitere automatisch generierte Fehlerbäume für geänderte Einsatzzeiten, die durch die Änderung der Randbedingungen erstellt werden, unterscheiden sich in diesem Fall lediglich quantitativ. Diese weiteren generierten Fehlerbäume sind im Anhang B abgebildet.

19 Die Werte, die im Fehlerbaum dargestellt sind, sind jeweils auf eine Vor- und eine Nachkommastelle mit entsprechender Zehnerpotenz zur ersten zählenden Stelle gerundet.
In den hier dargestellten automatisch generierten Fehlerbäumen sind teilweise unnötig erscheinende Gatter zu sehen, die nur ein nachfolgendes Ereignis haben. Dies liegt an der Erstellung aus dem zu Grunde liegenden modular aufgebauten Modell, sowie der Vorgabe, dass nur Gatter mit Ereignissen und umgekehrt verbunden werden können, aber nie direkt Ereignisse mit Ereignissen (oder Gatter mit Gatter). In manuell erstellten Fehlerbäumen wird ein solcher Fehler beispielsweise sprachlich verbunden als Ereignis in den Fehlerbaum eingepflegt (Beispiel hier: *CrackingOrBreakageE* über Gatter zu *pistonE* wird zu *CrackingOrBreakageE of pistonE*). Mit einer Nachbearbeitung des Modells könnten diese entfernt werden.

4.1.3 Fehlermöglichkeits- und -einflussanalyse des Schiffsmotors

Neben der Fehlerbaumerstellung und Zuverlässigkeitsberechnung des Schiffsmotors ist eine einfache Integration der Fehlermöglichkeits- und -einflussanalyse umgesetzt. Diese Integration erfolgt in einer Entwurfssprache, die die Entwurfssprache des Schiffsmotors verwendet und, durch die Einbindung des Klassendiagramms der Fehlermöglichkeits- und -einflussanalyse[20] ins Klassendiagramm, erweitert, um die erforderlichen Klassen in der Entwurfssprache verwenden zu können. Das Aktivitätsdiagramm beginnt mit dem ursprünglichen Programm des Schiffsmotors *EngineRootProgram*, wie im vorherigen Unterkapitel beschrieben. Danach wird, durch die Einbindung einer Regel *AddFailureModeAndConsequence* zur Erstellung der Fehlermodusdaten der Fehlermöglichkeits- und -einflussanalyse in das Aktivitätsdiagramm, das Modell entsprechend erweitert. Zusätzlich wird ein Schnittstellenaufruf *Failure Mode and Effect Analysis* zur Erstellung der Fehlermöglichkeits- und -einflussanalyse integriert. In Abbildung 4.16 ist das erweiterte Aktivitätsdiagramm zu sehen.

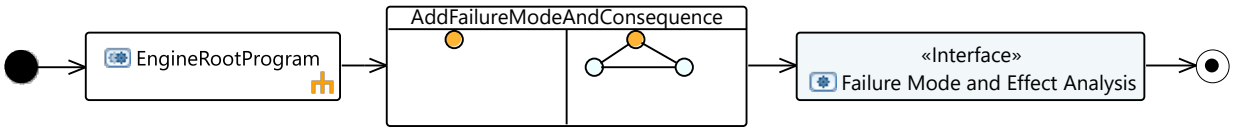

Abbildung 4.16: Erweitertes Aktivitätsdiagramm des Schiffsmotors

Zur einfachen Integration der Fehlermöglichkeits- und -einflussanalyse ist die Regel *AddFailureModeAndConsequence* allgemein gehalten, die exemplarisch am Schiffsmotor gezeigt wird. Dargestellt ist die Regel *AddFailureModeAndConsequence* in Abbildung 4.17. In dieser Regel wird zu einer Instanz der Klasse Systemfehler der Schiffsmotorkomponente *EngineSystemFailure* eine Konsequenz des Fehlermodus mit einem zugehörigen Parameter der Bedeutung (*severityNumber*) dieser für das zu untersuchende System erstellt. Die Konsequenz ist hierbei der Systemfehler der Schiffsmotorkomponente. Des Weiteren wird in dieser Regel eine Instanz des Fehlermodus erstellt, mit dem Systemfehler der Schiffsmotorkomponente und der Konsequenz verknüpft und die Entdeckungswahrscheinlichkeit (*detectionNumber*) gesetzt.

20 Das Klassendiagramm der Fehlermöglichkeits- und -einflussanalyse ist in Kapitel 3.3.1 ausführlich beschrieben.

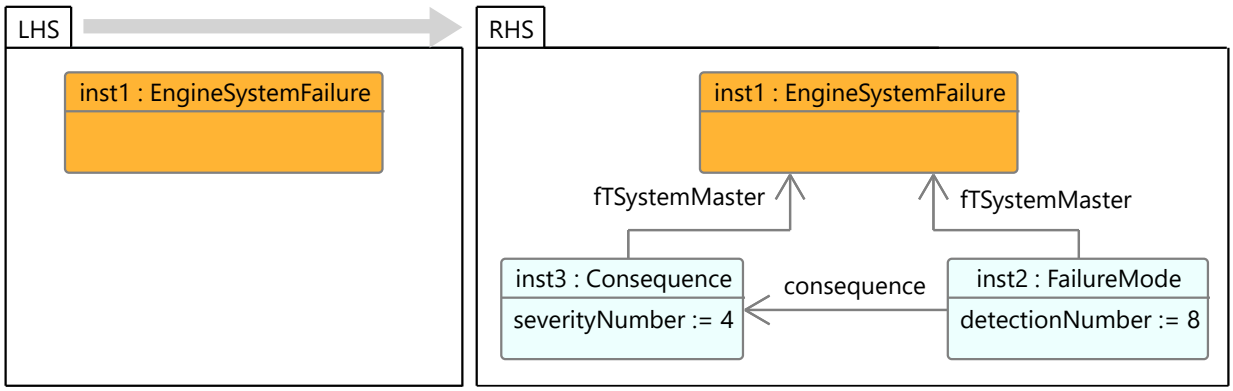

Abbildung 4.17: Beispielregel *AddFailureModeAndConsequence*[21]

Die gezeigte einfache Erweiterung in einer Entwurfssprache mit der Integration[22] der Fehlermöglichkeits- und -einflussanalyse ermöglicht über die Schnittstelle der Fehlermöglichkeits- und -einflussanalyse die Berechnung zugehöriger Kenngrößen, sowie die Bestimmung der Ursachen eines Fehlermodus. Die Ergebnisse können in einem Formblatt der Fehlermöglichkeits- und -einflussanalyse ausgegeben werden, welches die Zahlenwerte der Regeln zu Bedeutung und Entdeckungswahrscheinlichkeit sowie die berechneten Zahlenwerte zur Auftretenswahrscheinlichkeit, entsprechend einer vorgegebenen Vorschrift, wie im Kapitel 3.3.1 *Automatische Fehlermöglichkeits- und -einflussanalyse* beschrieben, und der Risikoprioritätszahl enthält. In Abbildung 4.18 ist ein generiertes Formblatt für eine Laufzeit von 500 Stunden abgebildet.

21 In dieser Beispielregel ist eine spezifizierte Klasse *EngineSystemFailure* dargestellt, allgemein wird die Oberklasse *FTSystemMaster* verwendet, damit für die einfache vorgestellte Integration eine Regel ausreichend ist, um für jeden (Teil-)Fehlerbaum einen zugehörigen Fehlermodus und eine zugehörige Konsequenz zu erstellen.

22 Einbau einer Regel in Programmablauf, sowie Schnittstellenaufruf

Funktion	Fehlermodus	Auswirkung	Ursache	Auftretenswahrscheinlichkiet	Bedeutung für das System	Entdeckungswahscheinlichkeit	RPZ
System function	failuremode2	consequence2	CrackingOrBreakageRC	6.0	4	8	192
System function	failuremode4	consequence4	pistonpistonringcylinderlinercomponent0RC	7.0	4	8	224
System function	failuremode4	consequence4	cylinderblock0RC	6.0	4	8	192
System function	failuremode3	consequence3	CrackingOrBreakageRC	6.0	4	8	192
System function	failuremode7	consequence7	pistonring0RC	7.0	4	8	224
System function	failuremode4	consequence4	crankshaftconnectingrodsystem0RC	7.0	4	8	224
System function	failuremode7	consequence7	cylinderliner0RC	6.0	4	8	192
System function	failuremode0	consequence0	bearingbush0RC	7.0	4	8	224
System function	failuremode6	consequence6	OtherFailuresRC	6.0	4	8	192
System function	failuremode6	consequence6	CrackingOrBreakageRC	6.0	4	8	192
System function	failuremode5	consequence5	DeadlockingOrBreakageRC	7.0	4	8	224
System function	failuremode4	consequence4	fuelinjectionsystem0RC	6.0	4	8	192
System function	failuremode7	consequence7	piston0RC	6.0	4	8	192
System function	failuremode1	consequence1	OverwornOrMeltingRC	7.0	4	8	224
System function	failuremode0	consequence0	CrankshaftOrConnectingRodCrackingOrBreakageRC	6.0	4	8	192
System function	failuremode8	consequence8	FuelInjectionSystemFailureRC	6.0	4	8	192

Abbildung 4.18: FMEA-Formblatt für einen Schiffsmotor bei einer Laufzeit von 500 Stunden[23]

23 Die Zahlenwerte sind durch die einfache Regelintegration für die Bedeutung und Entdeckungswahrscheinlichkeit gleich und dienen lediglich zur Veranschaulichung.

4.1.4 Integration: Schiffsantriebssystem

Der modulare Aufbau durch einzelne Entwurfssprachen für die Komponenten und Unterkomponenten des Schiffsmotors zur detaillierten Zuverlässigkeitsanalyse einzelner Komponenten mit der Möglichkeit der Erstellung einer übergeordneten Zuverlässigkeitsanalyse wurde in den vorherigen Abschnitten beschrieben. Zur Verdeutlichung der Anwendbarkeit und der Anwendungsmöglichkeiten des modularen Aufbaus wird nachfolgend im Gegensatz zum vorgestellten Top-Down-Ansatz zur Erstellung der Entwurfssprachen, von Systemmotor bis zum Kolben, der vorgestellte Schiffsmotor in ein übergeordnetes Schiffsantriebssystem Bottom-Up integriert.

4.1.4.1 Modellierung des Schiffsantriebssystems

Der funktionale Aufbau eines Schiffsantriebssystems ist in Abbildung 4.19 zu sehen. Das Schiffsantriebssystem besteht hier funktional aus den Komponenten Schiffsmotor (main engine), Übertragungsregler (gear box), Energieübertragung (shaft, bearing, stern tube, sealing) und Propeller (propeller).

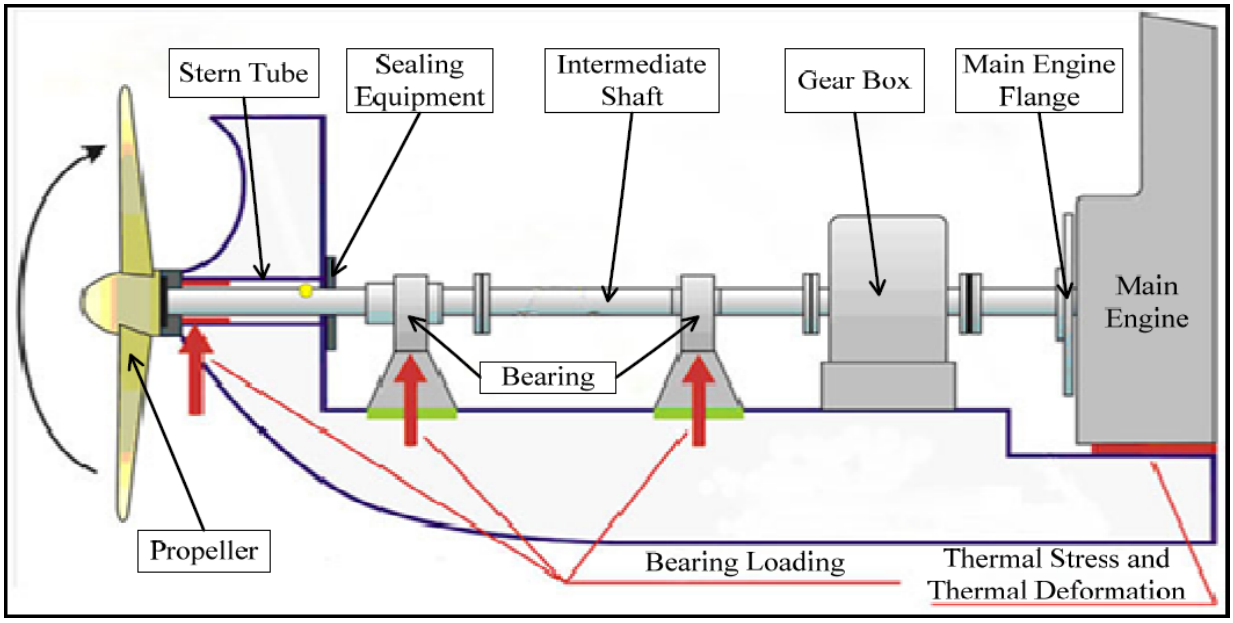

Abbildung 4.19: Skizze eines Schiffsantriebssystems aus [31]

Die Komponente Schiffsmotor ist ein Teil des Schiffsantriebssystems. Dieser Zusammenhang drückt sich in der Erweiterung des Klassendiagramms des Schiffsmotors, zu sehen in Abbildung 4.20, durch ein Erben der Klasse *Engine* von der Klasse *PropulsionSystemPart* aus. Diese Erweiterung ist ausreichend, um die Entwurfssprache des Schiffsmotors

für die automatische Zuverlässigkeitsanalyse in die Entwurfssprache des Schiffsantriebssystems integrieren zu können.

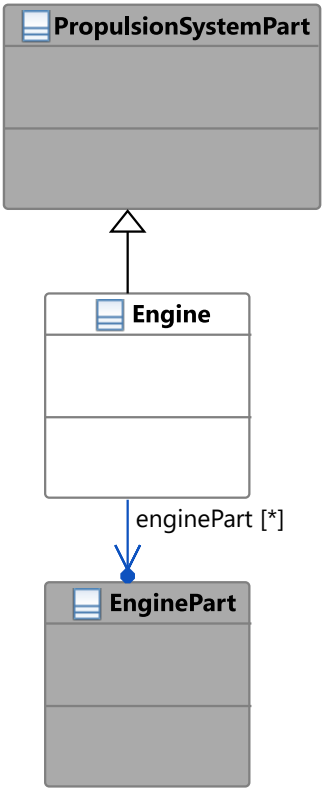

Abbildung 4.20: Erweitertes Klassendiagramm Schiffsmotor

Die Entwurfssprache des Schiffsantriebssystems ist abweichend von den vorherigen Entwurfssprachen von Systemen, wie dem Schiffsmotor oder dem Kolbensystem, aufgebaut, um die mögliche funktionale Abgrenzung im modularen Aufbau zu demonstrieren. Dazu ist im Klassendiagramm des Schiffsantriebssystems, statt der direkten Assoziation von der Klasse *PropulsionSystem* zur Schiffsantriebssystemteileklasse *PropulsionSystemPart*, eine zusätzliche (Zwischen-)Klasse *PropulsionSystemFunctionalPart* zur speziellen Modellierung des funktionalen Schiffsantriebssystemteils integriert, wie in Abbildung 4.21 zu sehen. Diese erbt von der Fehlerbaumobjektklasse *FaultTreeObject* und hat eine Assoziation zur Schiffsantriebssystemteileklasse *PropulsionSystemPart*, wodurch das konkrete Teil vom Fehlerbaumobjekt in der Modellierung getrennt wird.[24] Darüber hinaus ist abweichend von den vorherigen Klassendiagrammen für den Schiffsmotor eine Assoziation

24 Um keine unnötig redundanten Elemente in den Modellen zu haben, werden durch die Trennung Änderungen in den vorher vorgestellten Entwurfssprachen (Klassendiagramme, Aktivitätsdiagramme und Regeln) nötig. Allerdings ist dies für die gewünschte Ausführung nicht notwendig, sondern eine optionale Anpassung zur Verbesserung der Modellierung.

propulsionSystemSystemFailure von der Klasse *PropulsionSystem* zur zugehörigen Systemfehlerklasse *PropulsionSystemSystemFailure* modelliert, um diesen dieser zuzuordnen.[25]

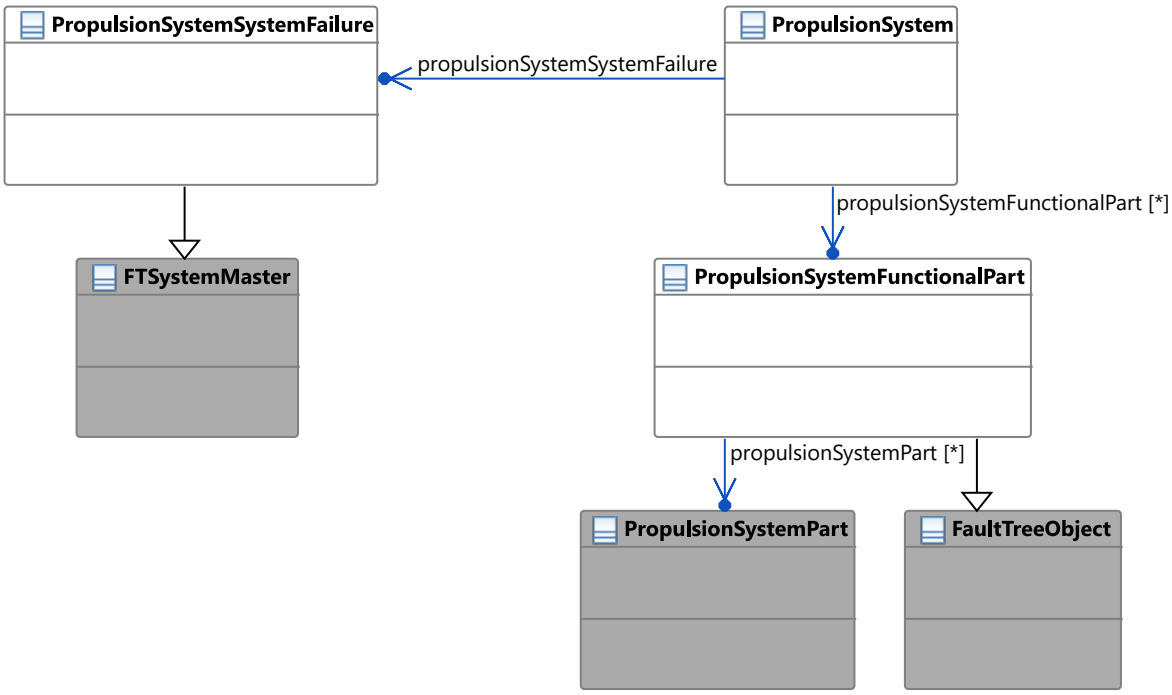

Abbildung 4.21: Klassendiagramm Schiffsantriebssystem

In Abbildung 4.22 ist das Aktivitätsdiagramm der Entwurfssprache des Schiffsantriebssystems dargestellt. Es werden hier zu Beginn in der Regel *CreateBoundaryConditions* die Randbedingungen gesetzt, die die redundante Anzahl der einzelnen Untersysteme festlegt. Der weitere Programmablauf ist analog zu den vorherigen Programmen der einzelnen Systeme. In der Regel *CreatePropulsionSystem* wird eine Instanz des Schiffsantriebssystems erstellt. Zu dieser Instanz werden die funktionalen Komponenten mit *Add*-Regeln in den entsprechenden Unterprogrammen erstellt und mit dieser verknüpft. Anschließend wird der funktionale Zusammenhang im Unterprogramm *SetFunctionFlow* erstellt, die Zuverlässigkeitsanalyseprogrammabläufe der funktionalen Komponenten im Unterprogramm *AddMarinePropulsionSystemReliabilityAnalysis* integriert, der Systemfehler in der Regel *CreatePropulsionSystemSystemFailure* erstellt und die funktionalen Teile in der folgenden *Add*-Regel mit diesem verknüpft und abschließend die Zuverlässigkeitsanalyse und die Fehlerbaumerstellung über Schnittstellenaufrufe durchgeführt.

25 Die Zuordnung erfolgte in den vorherigen Entwurfssprachen über die Verbindung, die sich durch die Vererbungen zu *FTSystemMaster* und *FaultTreeObject* aus dem Klassendiagramm zur Zuverlässigkeitsberechnung ergibt. Da durch die Trennung von konkretem Teil und Fehlerbaumobjekt auf oberster Ebene der Modellierung die Klasse *PropulsionSystem* kein Fehlerbaumobjekt *FaultTreeObject* ist, ist hier eine spezifische Assoziation notwendig.

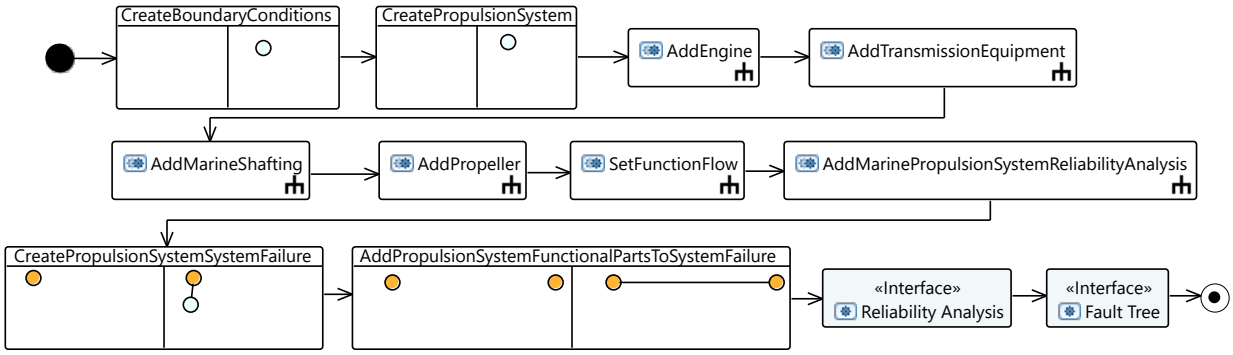

Abbildung 4.22: Aktivitätsdiagramm Schiffsantriebssystem

Das Hinzufügen der Teilsysteme des Schiffsantriebssystems in den *Add*-Unterprogrammen über die in ein konkretes Objekt und ein Fehlerbaumobjekt aufgeteilten Instanzen, erfordert zusätzliche Regeln, die beispielhaft im Aktivitätsdiagramm zum Hinzufügen des Schiffsmotors *AddEngine* in Abbildung 4.23 zu sehen sind. Zunächst wird eine funktionale Teileinstanz in der Regel *AddEngineFunctionalPart* zum Antriebssystem hinzugefügt. Danach werden in einem iterativen Unterprogramm *AddEngineObjects* die in den Randbedingungen vorgegebene Anzahl redundanter Schiffsmotorinstanzen erzeugt und der funktionalen Teileinstanz zugeordnet. Nachfolgend wird eine Instanz des Systemfehlers (des Teilsystems) erzeugt (*CreateEngineFunctionalPartSystemFailure*), wie in Abbildung 4.24 dargestellt, in der der Parameter *parallel* des erzeugten Systemfehlers auf wahr gesetzt wird, um die verknüpften Fehlerbaumobjekte als redundant bzw. kausal abhängig für die Zuverlässigkeitsberechnung und den Fehlerbaum zu kennzeichnen. Abschließend wird die Instanz des Systemfehlers mit den Schiffsmotorinstanzen verknüpft *AddEngineFunctionalPartObjects2SystemFailure*.

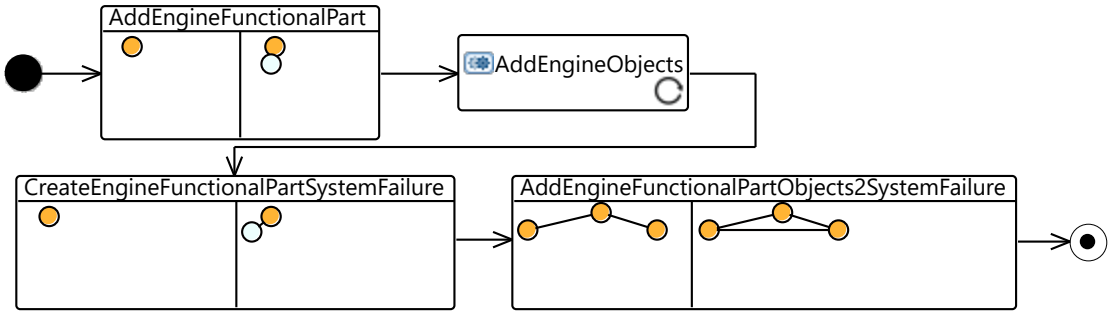

Abbildung 4.23: Aktivitätsdiagramm *AddEngine*[26]

26 Das Symbol des kreisförmigen Pfeiles in der Unterprogrammdarstellung rechts unten kennzeichnet ein iteratives Unterprogramm.

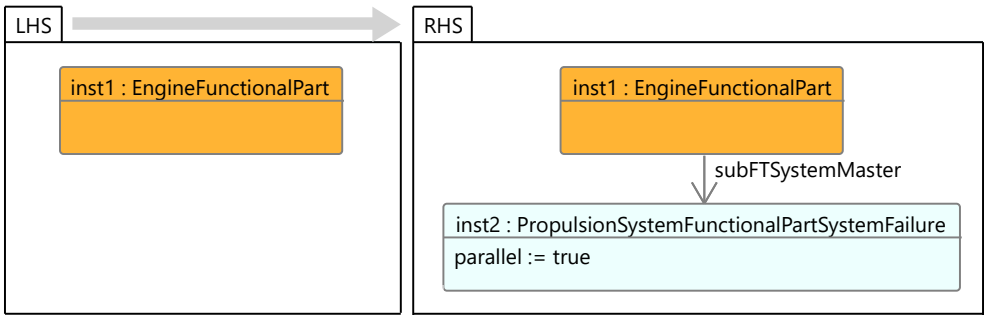

Abbildung 4.24: Regel *CreateEngineFunctionalPartSystemFailure*

Im Unterprogramm *SetFunctionFlow* wird entsprechend der vorherig gezeigten Entwurfssprachen der funktionale Fluss modelliert, hier allerdings zwischen den funktionalen Instanzen anstatt der konkreten Komponenten. Dies zeigt der Programmablauf in Abbildung 4.25, in dem über *Connect*-Regeln die funktionale Instanz des Schiffsmotor (main engine) mit der funktionalen Instanz des Übertragungsreglers (transmission equipment), diese mit der funktionalen Instanz der Energieübertragung (marine shafting) und diese abschließend mit der funktionalen Instanz des Propellers (propeller) verknüpft wird.

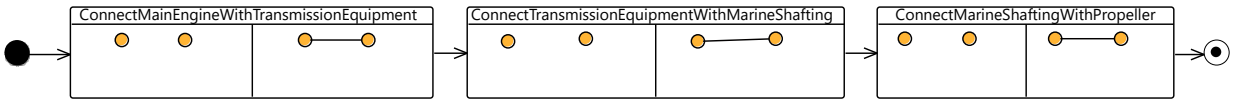

Abbildung 4.25: Unterprogramm *SetFunctionFlow*

Das nachfolgende Unterprogramm *AddMarinePropulsionSystemReliabilityAnalysis* besteht aus den Aufrufen der einzelnen Aktivitätsdiagramme der Entwurfssprachen der Komponenten, die die für die Zuverlässigkeitsanalyse relevanten Regeln enthalten. Das zugehörige Aktivitätsdiagramm ist in Abbildung 4.26 zu sehen.

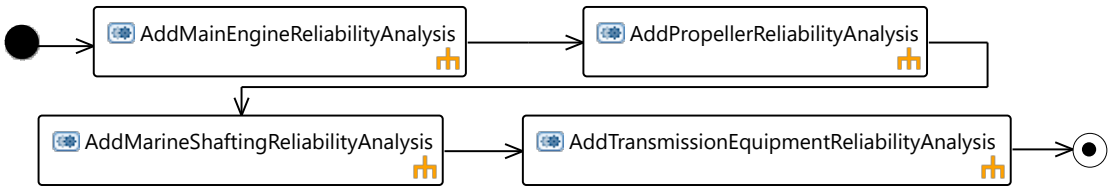

Abbildung 4.26: Unterprogramm *AddMarinePropulsionSystemReliabilityAnalysis*[27]

27 Der Programmablauf der Entwurfssprache des Schiffsmotors wurde bereits detailliert beschrieben, die Aktivitätsdiagramme der anderen Komponenten des Schiffsantriebssystems sind analog aufgebaut und werden nicht weiter im Detail beschrieben.

4.1.4.2 Fehlerbaumerstellung für das Schiffsantriebssystem

Das Ergebnis eines automatisch generierten Fehlerbaums für ein Schiffsantriebssystem ohne Redundanz nach Ausführung der Entwurfssprache unter Verwendung der untergeordneten Entwurfssprachen ist in Abbildung 4.28 zu sehen. Dabei ist offensichtlich, dass ein einzelner Fehlerbaum ab einer gewissen Größe den Vorteil einer grafisch übersichtlichen Darstellung verliert. Durch den modularen Aufbau können jedoch die einzelnen Fehlerbäume bzw. Äste eines Fehlerbaumes[28] ausgegeben werden, um sie zur (visuellen) Analyse zu verwenden. Der qualitative Fehlerbaum des Schiffsantriebssystems mit einer Detaillierung von einer Ebene ist in Abbildung 4.27 zu sehen. Für einen Ast sei auf den Fehlerbaum des Schiffsmotors in Abbildung 4.15 hingewiesen.[29]

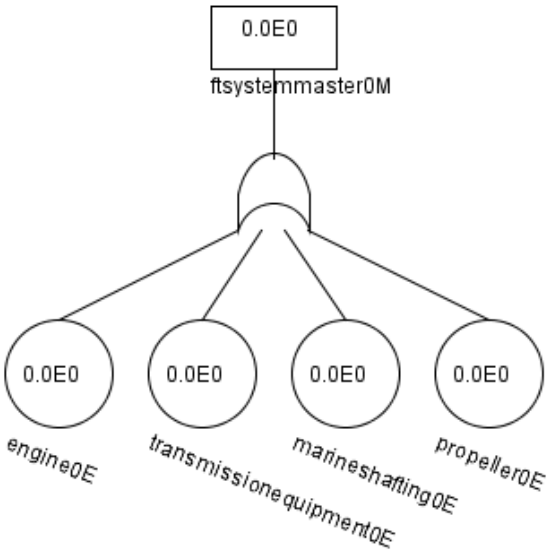

Abbildung 4.27: Teilfehlerbaum Schiffsantriebssystem[30]

28 Dies ist vergleichbar mit der Verwendung von Transfergattern in Fehlerbäumen, die modular einzelne Fehlerbäume für Teilsysteme repräsentieren. Allerdings könnten in (manuell) frei erstellten Transfergattern globale Fehler lokal hinterlegt werden, wodurch der lokale Fehlerbaum nicht einzeln unabhängig - ohne kausalen Zusammenhang - zum globalen Fehlerbaum richtig ist.

29 Hier allerdings mit quantitativ hinterlegten Werten, die beim vollständigen Fehlerbaum des Schiffsantriebssystems nicht verwendet wurden, da nicht für alle Komponenten Daten vorlagen.

30 In den Abbildungen 4.27 und 4.28 sind lediglich qualitative Fehlerbäume abgebildet, da nur für den Schiffsmotor (*engine*) bzw. dessen (Teil-)Komponenten Werte vorlagen und eine Mischung mit angenommenen Werten innerhalb eines Modells nicht als sinnvoll erachtet wird.

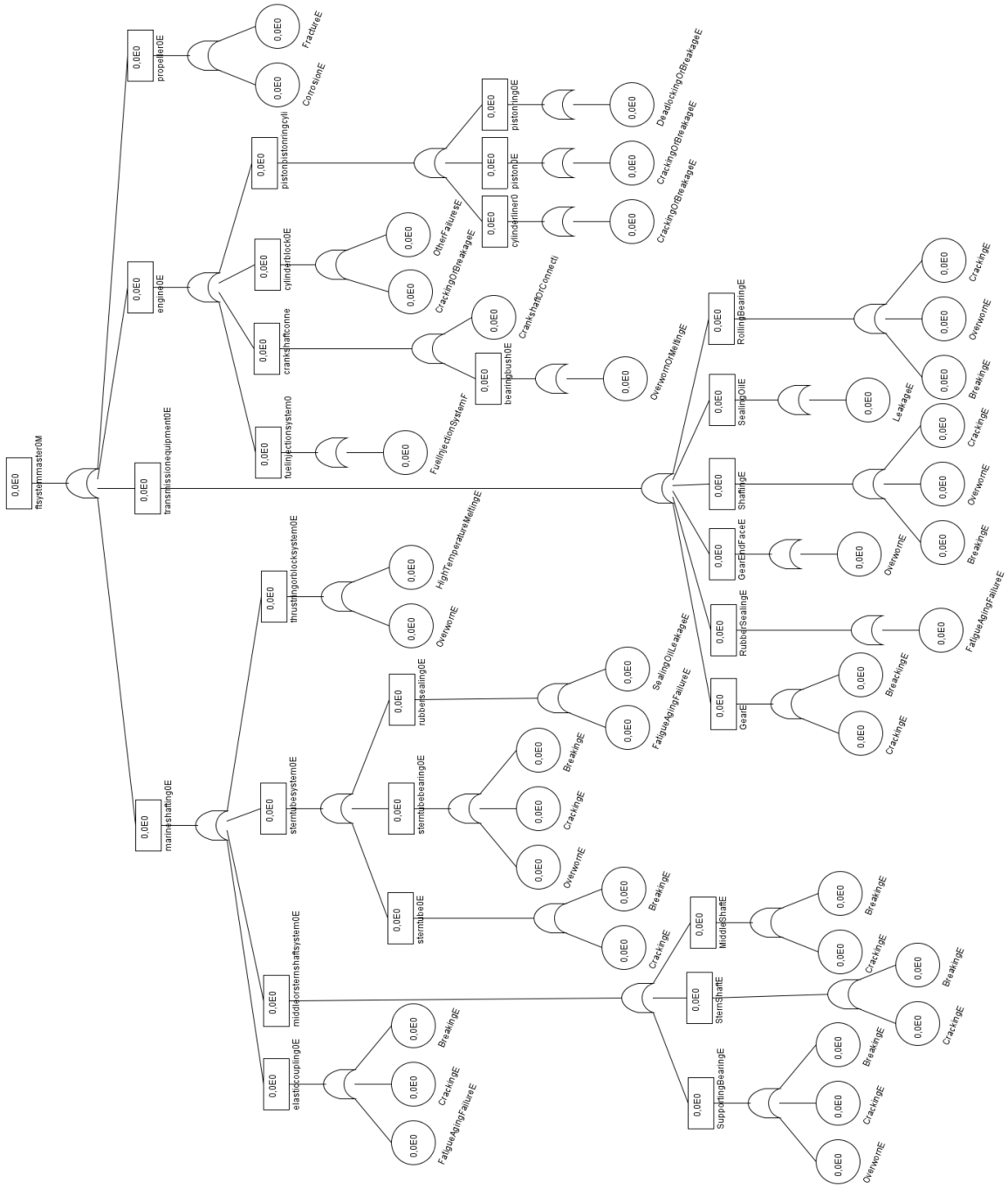

Abbildung 4.28: Fehlerbaum Schiffsantriebssystem

4.1.4.3 Fehlermöglichkeits- und -einflussanalyse des Schiffsantriebssystems

In Abbildung 4.29 ist die Integration der Fehlermöglichkeits- und -einflussanalyse unter Verwendung des Aktivitätsdiagramms des Schiffsantriebssystem in einer erweiterten Entwurfssprache dargestellt. Nach dem vorgestellten Aktivitätsdiagramm des Schiffsantriebssystem, welches hier als Unterprogramm *PropulsionSystemProgram* eingebunden und in Abbildung 4.22 zu sehen ist, folgt ein Unterprogramm und ein Schnittstellenaufruf. Im Unterprogramm *AddMarinePropulsionSystemFailureModeData* werden analog zur Zuverlässigkeitsanalyse die relevanten Programme der einzelnen Entwurfssprachen des Schiffsantriebssystems mit den Regeln für die Fehlermöglichkeits- und -einflussanalyse eingebunden und über den folgenden Schnittstellenaufruf *Failure Mode and Effect Analysis* die automatische Fehlermodusobjekterstellung, wie im Kapitel 3.3.1 *Automatische Fehlermöglichkeits- und -einflussanalyse* beschrieben, durchgeführt. Die Ausgabe in einem FMEA-Formblatt kann aus dem erweiterten Modell heraus durchgeführt werden und ist im Anhang B zu finden.

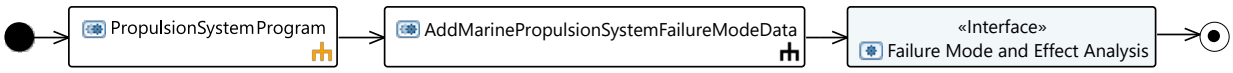

Abbildung 4.29: Aktivitätdiagramm Schiffsantriebssystem mit Integration der FMEA

4.2 Satellitenantriebssysteme

Die Integration der automatischen Fehlerbaumanalyse in ein vorab modelliertes System wird am Beispiel eines Modells für Satellitenantriebssysteme demonstriert. Diese Satellitenantriebssystemmodelle werden über eine Entwurfssprache, wie von Schmidt [100] beschrieben, erstellt. Mit dieser Entwurfssprache können unterschiedliche topologische und parametrische Ausprägungen dreier unterschiedlicher Satellitenantriebssystemarten - monergol, diergol und Kaltgas - generiert werden. Aus einem solchen generierten Modell kann eine schematische Flussdarstellung des Systems automatisch abgeleitet werden, wie in Abbildung 4.30 für ein Kaltgassystem dargestellt. Aufbauend auf den erstellten Modellen wird die Integration der automatischen Fehlerbaumanalyse durch Erweiterung der Entwurfssprache im Folgenden vorgestellt.

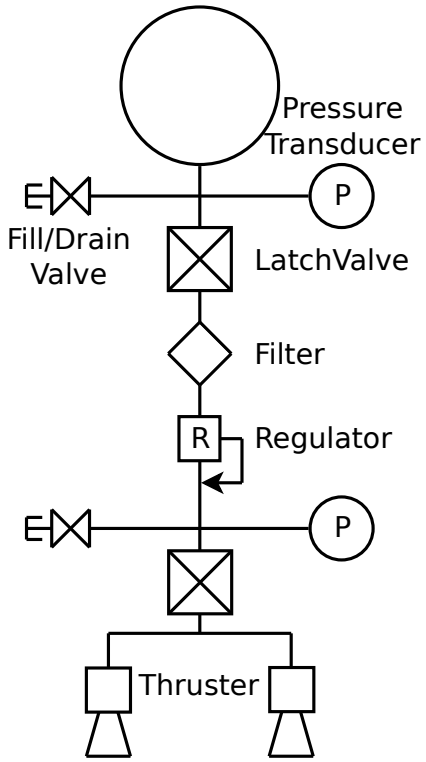

Abbildung 4.30: Flussdarstellung eines Kaltgassatellitenantriebssystems [98]

4.2.1 Modellierung der Satellitenantriebssysteme

In den ersten Schritten einer Fehlerbaumanalyse werden die Systemfunktion, die Systemfunktionsausfälle, die Systemkomponenten und die kausalen Verknüpfungen dieser untereinander ermittelt. Angewandt auf das Satellitenantriebssystem ist die zu erfüllende Systemfunktion die Erzeugung des geforderten Schubs. Dafür muss der korrekte[31] Treibstofffluss innerhalb des Systems gewährleistet sein. Ein Systemfunktionsausfall ist dementsprechend der nicht gewährleistete korrekte Treibstofffluss innerhalb des Systems. Die für diesen Systemfunktionsausfall relevanten Systemkomponenten sind die, durch die Treibstoff fließt bzw. die am Treibstofffluss beteiligt sind. Die kausalen Verknüpfungen der Systemkomponenten untereinander entsprechen den Durchflussverbindungen innerhalb des Satellitenantriebssystems. Diese Schritte der Fehlerbaumanalyse werden mit einer Erweiterung der Entwurfssprache der Satellitenantriebssysteme im Rahmen der Modellierung umgesetzt.

Ausgehend vom existierenden Klassendiagramm des Satellitenantriebssystems wird dieses entsprechend der ermittelten Vorgaben erweitert, um die Fehlerbaumerstellung zu

31 Geforderter Treibstofffluss unter geforderten Randbedingungen zu geforderter Zeit an geforderter Stelle mit geforderter Geschwindigkeit im Rahmen der geforderten Toleranz.

ermöglichen. Dazu wird das Klassendiagramm für die Satellitenantriebssysteme mit dem Klassendiagramm der Zuverlässigkeitsanalyse verknüpft. Diese Verknüpfung ist schematisch in Abbildung 4.31 zu sehen. Hierbei erbt die Klasse *FlowElement* aus der Satellitenantriebsentwurfssprache, die die Komponenten des Satellitenantriebssystems repräsentiert durch die der Treibstoff fließt und die zentrale Grundklasse der Komponenten darstellt, von der Klasse *FaultTreeObject* aus dem Zuverlässigkeitsanalyseklassendiagramm.

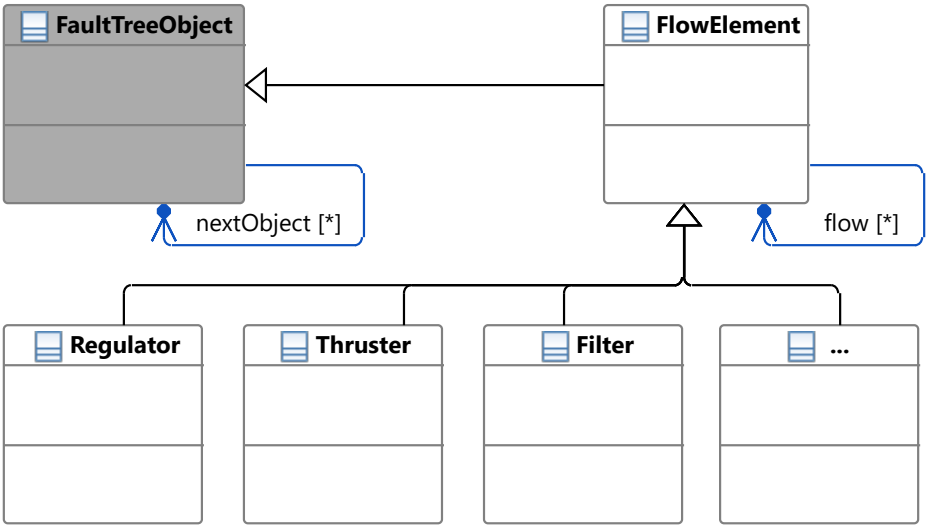

Abbildung 4.31: Integration des Fehlerbaumobjekts in das Klassendiagramm der Satellitenantriebsentwurfssprache

Neben der Einbindung des Klassendiagramms der Zuverlässigkeitsanalyse wird das Aktivitätsdiagramm der Satellitenantriebsentwurfssprache für die automatische Fehlerbaumanalyse erweitert, wie in Abbildung 4.32 zu sehen. Das Aktivitätsdiagramm der Satellitenentwurfssprache beginnt mit einer *Axiom*-Regel, die den Typ des Satellitenantriebssystems und die Randbedingungen (z.B. Schubanforderungen) an das System festlegt. In den Unterprogrammen *Topology* und *Parametrics* wird das System ausgelegt, erstellt und berechnet.[32] Die Erweiterung für die Zuverlässigkeitsanalyse erfolgt im Unterprogramm *SubReliability*, auf welches im Folgenden genauer eingegangen wird.

4.2.2 Fehlerbaumerstellung für ein Satellitenantriebssystem

Für die automatische Fehlerbaumerstellung werden im Programm *SubReliability* das Wurzelereignis, die Verknüpfung zu zugehörigen Komponenten und deren funktional kausa-

32 Für detailliertere Ausführungen zur Satellitenantriebsentwurfssprache siehe Schmidt [98].

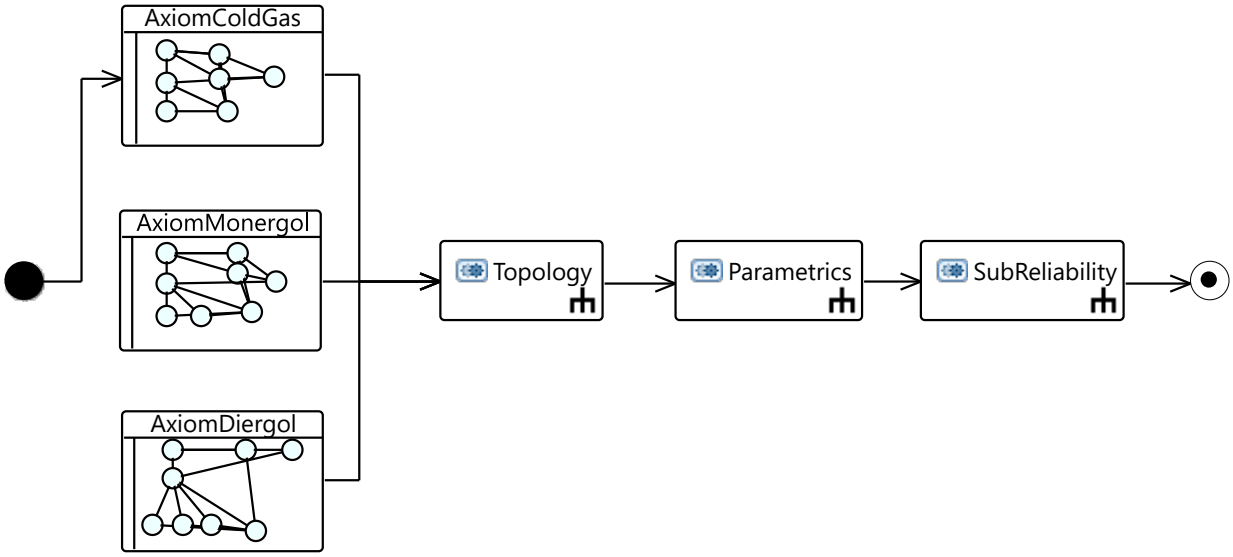

Abbildung 4.32: Erweitertes Aktivitätsdiagramm der Satellitenantriebsentwurfssprache[33]

le Verknüpfungen untereinander erstellt. In Abbildung 4.33 ist das Aktivitätsdiagramm zu sehen. Die Regel *CreateFTSystemMaster* erstellt eine Instanz der Klasse *FTSystemMaster*, die das Wurzelereignis im Fehlerbaum darstellt. Im nachfolgenden Unterprogramm *ConnectWithFTSystemMaster* werden die Komponenten, die zur Systemfunktion gehören, mit der erstellten Instanz der Klasse *FTSystemMaster* verknüpft. Darauf folgend werden die Komponenten untereinander im Unterprogramm *SetFunctionFlow* entsprechend der funktional kausalen Struktur verknüpft. Im Unterprogramm *SetReliability* werden die Zuverlässigkeitswerte der Komponenten[34] gesetzt. Über die Schnittstellenaufrufe wird die Zuverlässigkeitsanalyse (*Reliability Analysis*) durchgeführt, sowie ein zugehöriger Fehlerbaum (*Fault Tree*) erstellt.

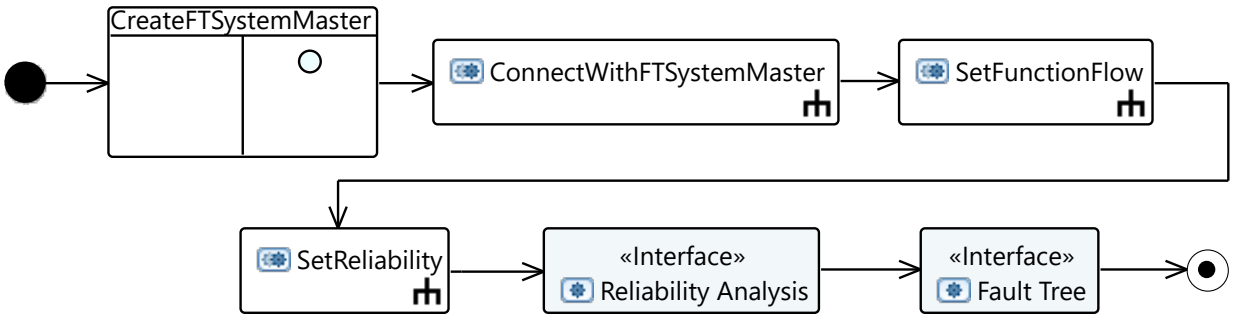

Abbildung 4.33: Aktivitätsdiagramm *SubReliability*

33 Der Kontrollfluss führt in diesem Fall zur Erstellung eines Kaltgassatellitenantriebssystemmodells. Eine Anpassung des Kontrollflusses ermöglicht die Erstellung von monergolen oder diergolen Satellitenantriebssystemmodellen.

34 nach [122]

Die geforderte Systemfunktion wird im Rahmen der Systemanalyse mit dem Austritt des korrekt verarbeiteten Treibstoffes, der aus dem Tank kam, aus einem Triebwerk definiert. Die Komponenten, die an dieser Funktion beteiligt sind, sind die Instanzen der Klasse *FlowObject*, durch die der Treibstoff fließt. Im, in Abbildung 4.34 gezeigten, Programm *SetFunctionFlow* wird der funktionale Zusammenhang für diese Funktion entsprechend des gewählten Satellitenantriebssystems festgelegt.

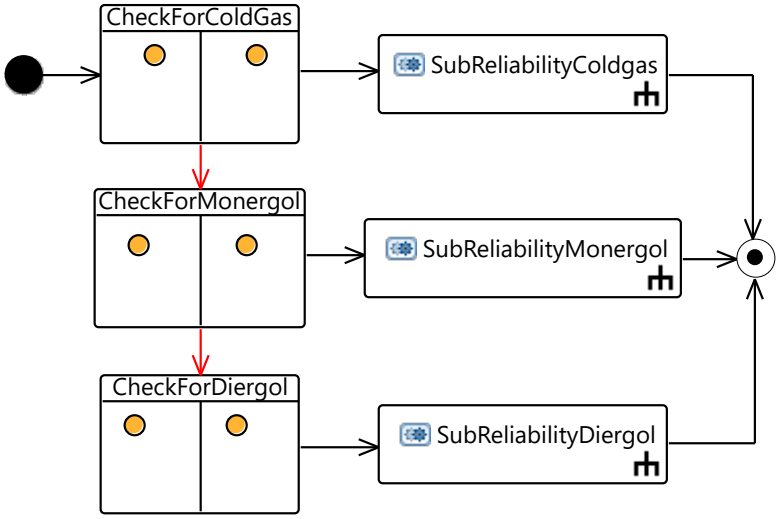

Abbildung 4.34: Aktivitätsdiagramm *SetFunctionFlow*[35]

Nachdem das Wurzelereignis, die zugehörigen Komponenten und die funktional kausalen Verknüpfungen dieser Komponenten untereinander bestimmt wurden, kann die Zuverlässigkeitsberechnung und Fehlerbaumerstellung mit Auswertung durchgeführt werden. Für das Kaltgassatellitenantriebssystem, welches in Abbildung 4.30 dargestellt ist, ist der automatisch erzeugte Fehlerbaum in Abbildung 4.35 zu sehen.

Qualitative Fehlerbaumanalyse[36] Die qualitative Fehlerbaumanalyse des Fehlerbaumes des Kaltgassatellitenantriebssystems zeigt - wie direkt im Fehlerbaum ersichtlich[37], dass - bis auf die redundanten Komponenten der Triebwerke - durch die fehlende Redundanz jeder Ausfall einer Komponente kausal für den Ausfall des Gesamtsystems ist, sprich einen Einzelfehler (single point failure) darstellt.

35 Falls rot farbige Signalflüsse im Programmablauf sind, werden diese verwendet, sofern die Regel nicht ausgeführt wird. Hier wenn keine entsprechende Instanz des Satellitenantriebssystemtyps gefunden wird.
36 Erläuterungen zur Auswertung von Fehlerbäumen in Kapitel 2.1.1 Fehlerbaumanalyse (FTA)
37 Ein Basisereignis mit einer direkten Verbindung zum Wurzelereignis über ein *Oder-Gatter*, bedeutet, dass das Basisereignis kausal für das Wurzelereignis ist.

Quantitative Fehlerbaumanalyse Über die gesetzten Ausfallwahrscheinlichkeiten der Komponenten, ergibt sich für die Ausfallwahrscheinlichkeit der vorgegeben Systemfunktion des Kaltgassatellitenantriebssystems $F \approx 1,6 \cdot 10^{-2}$.

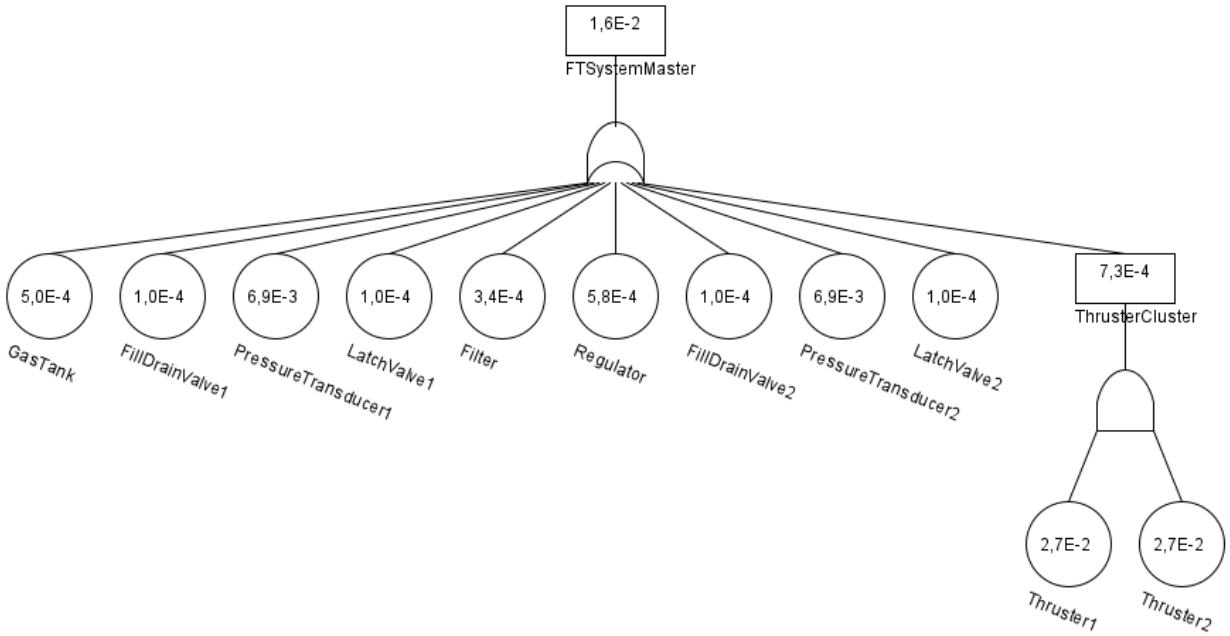

Abbildung 4.35: Fehlerbaum des Kaltgassatellitenantriebssystems

4.2.3 Automatische Topologievariation eines Satellitenantriebssystems

Der letzte, iterative Schritt einer Fehlerbaumanalyse ist die Verbesserung eines Systems anhand der Resultate aus der Fehlerbaumauswertung hinsichtlich der Zuverlässigkeit. Diese Verbesserung ist spezifisch in einer zusätzlichen Entwurfssprache für das Kaltgassatellitenantriebssystem umgesetzt, dessen Aktivitätsdiagramm in Abbildung 4.36 zu sehen ist. Die Entwurfssprache zur Verbesserung des Satellitenantriebssystems hinsichtlich der Zuverlässigkeit verwendet die Satellitenantriebsentwurfssprache. Zunächst wird das Axiom für das Kaltgassatellitenantriebssystem im Unterprogramm *AxiomColdGas*[38] erstellt. Darauf folgend wird im Unterprogramm *CreatePropulsionSystem* das nicht erweiterte[39] Aktivitätsdiagramm der Satellitenantriebsentwurfssprache nach dem Axiom ausgeführt. In der Regel *CreateFTSystemMaster* wird eine Instanz der Klasse *FTSystemMaster* erstellt und die Eigenschaft für eine minimale Zuverlässigkeit (als Ziel) gesetzt. Im nachfolgenden Unterprogramm *SetUpReliabilityNetwork* werden die kausalen Verknüpfungen

38 Im Unterprogramm ist lediglich die *Axiom*-Regel. Dass hier nicht die Regel direkt importiert wurde, liegt daran, dass nur (Unter-)Programme von einer anderen Entwurfssprache importiert verwendet werden können, aber keine Regeln

39 Ohne das Unterprogramm *SubReliability* (siehe Abbildung 4.32)

erstellt, sowie (Anfangs-)Zuverlässigkeitswerte für die Komponenten gesetzt, analog zum Ablauf im Unterprogramm *SubReliability*. Danach erfolgt die Zuverlässigkeitsanalyse im nachfolgend beschriebenen Unterprogramm *ReliabilityAnalysis*. Im folgenden Entscheidungsknoten wird geprüft, ob die Zuverlässigkeitsanforderung mit der Topologie des Systems erfüllt wurde. Ist dies nicht der Fall, wird das Satellitenantriebssystem unter Verwendung der identifizierten Topologieänderung neu erstellt und ausgewertet.

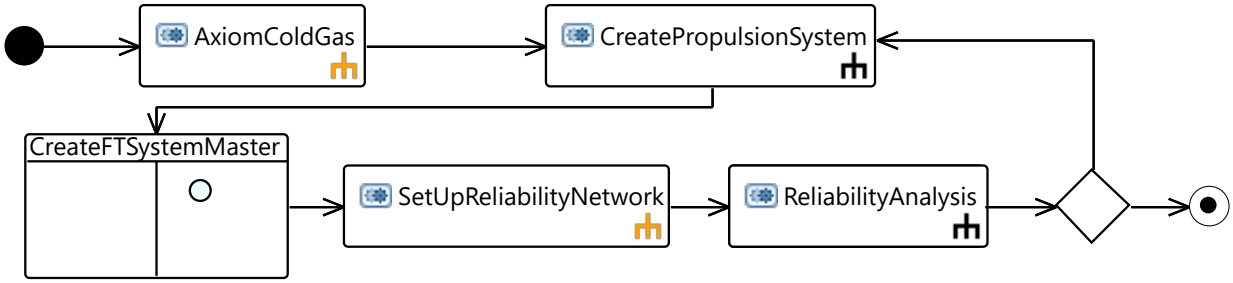

Abbildung 4.36: Aktivitätsdiagramm zur Erstellung eines (Kaltgas-)Satellitenantriebssystems mit Zuverlässigkeitsanpassung[40]

Prinzipiell entspricht dieser iterative Ablauf der Methode der Fehlerbaumanalyse, wie im Kapitel 2.1.1 beschrieben, mit automatischer Modellanpassung:

Funktionsdefinition → *Teiledefinition* → *Analyse* → *Maßnahmen*

Der Ablauf des Unterprogramms *ReliabilityAnalysis* zur Verbesserung der Zuverlässigkeit ähnelt sich bei den unterschiedlichen Satellitenantriebssystemtypen. Prinzipiell wird das System hinsichtlich der Zuverlässigkeit geprüft und gegebenenfalls verbessert. Dazu werden in einem iterativen Prozess die automatische Zuverlässigkeitsberechnung durchgeführt, die Ergebnisse mit der vorgegebenen Zielzuverlässigkeit verglichen und gegebenenfalls das System hinsichtlich der Zuverlässigkeit parametrisch verbessert oder topologische Änderungen zurückgegeben.[41] Abbildung 4.37 zeigt diesen Ablauf.

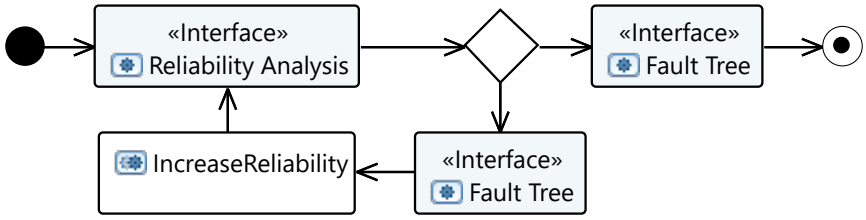

Abbildung 4.37: Aktivitätsdiagramm *ReliabilityAnalysis*

40 Ein oranges/helles Symbol rechts unten in einem Unterprogrammsymbol bedeutet, dass dieses Unterprogramm Teil einer anderen Entwurfssprache ist.
41 Die Rückgabe erfolgt durch veränderte Eigenschaften in den Instanzen.

Die Verbesserung erfolgt durch eine automatische Fehlerbaumanalyse. Dazu wird zunächst der Fehlerbaum des Systems erstellt und anschließend im Programm *IncreaseReliability* die Verbesserung über eine Heuristik durchgeführt. Das zugehörige Aktivitätsdiagramm *IncreaseReliability* ist in Abbildung 4.38 zu sehen. Innerhalb dieses Unterprogramms wird zunächst in der Javaregel[42] *identifyComponentsWithLeastReliability* anhand des automatisch erstellten Fehlerbaumes das Ereignis identifiziert, welches die niedrigste Zuverlässigkeit[43] auf der zweiten Ebene[44] des Fehlerbaumes[45] aufweist. Wenn ein Zwischenereignis identifiziert wurde, wird entsprechend der darunterliegende Fehlerbaum untersucht. Diese Prozedur wiederholt sich, bis ein Basisereignis identifiziert wurde. Nach der Identifikation des Basisereignisses wird das Unterprogramm *IncreaseFunctionalElementsReliability* aufgerufen, um die Zuverlässigkeit der identifizierten (funktionalen) Komponente zu erhöhen.

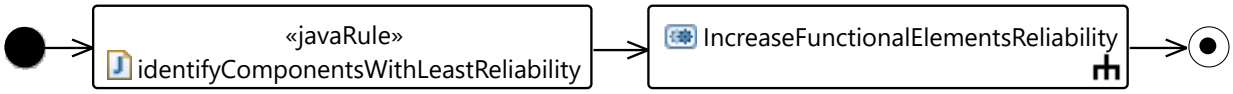

Abbildung 4.38: Aktivitätsdiagramm *IncreaseReliability*

Abbildung 4.39 zeigt das Aktivitätsdiagramm *IncreaseFunctionalElementsReliability*. In diesem wird über Entscheidungsknoten, auf Grundlage des identifizierten Basisereignisses im Fehlerbaum, bestimmt, welche (funktionale) Komponente hinsichtlich der Zuverlässigkeit verbessert werden soll, um die Zuverlässigkeit des Systems zu erhöhen.

Innerhalb dieser Unterprogramme für die einzelnen Komponenten werden zwei prinzipielle Vorgehensweisen zur Verbesserung der Zuverlässigkeit angewandt. Eine Änderung der Parametrik[46] oder der Topologie. Dabei wird eine Änderung der Parametrik einer Änderung der Topologie vorgezogen, weil diese als günstiger betrachtet wird bzw. weit weniger Auswirkungen auf den Entwurf hat, da die wesentliche Produktarchitektur unverändert bleibt. Beispielhaft ist dieser Ablauf für die Funktion der Druckregelung (*LimitPressure* mit der Komponente des *Regulator*s in Abbildung 4.40 dargestellt. Zunächst wird

42 Eine Javaregel unterscheidet sich von den grafischen Regeln durch eine rein textuelle Darstellung.
43 Bei identischen Zuverlässigkeitswerten entscheidet die (zufällige) Reihenfolge beim algorithmischen Abgleich, welches Ereignis gewählt wird.
44 Die Ebenennummerierung erfolgt - wie bereits erwähnt - bei den hier dargestellten Fehlerbäumen beginnend beim Wurzelereignis, welches die erste Ebene des Fehlerbaumes darstellt und besteht lediglich aus Ereignissen. Gatter sind keiner Ebene zugehörig.
45 Die niedrigste Zuverlässigkeit eines Ereignisses auf einer Ebene ist gleichbedeutend mit der höchsten Importanz (siehe Kapitel 2.1.1 Fehlerbaumanalyse (FTA) bzw. [12]) des Ereignisses auf dieser Ebene.
46 Parametrik bedeutet hier, dass die Zuverlässigkeitsparameter verändert werden, aber die Topologie erhalten bleibt. Diese parametrische Änderung kann aber ebenso den Austausch einer Komponente bedeuten.

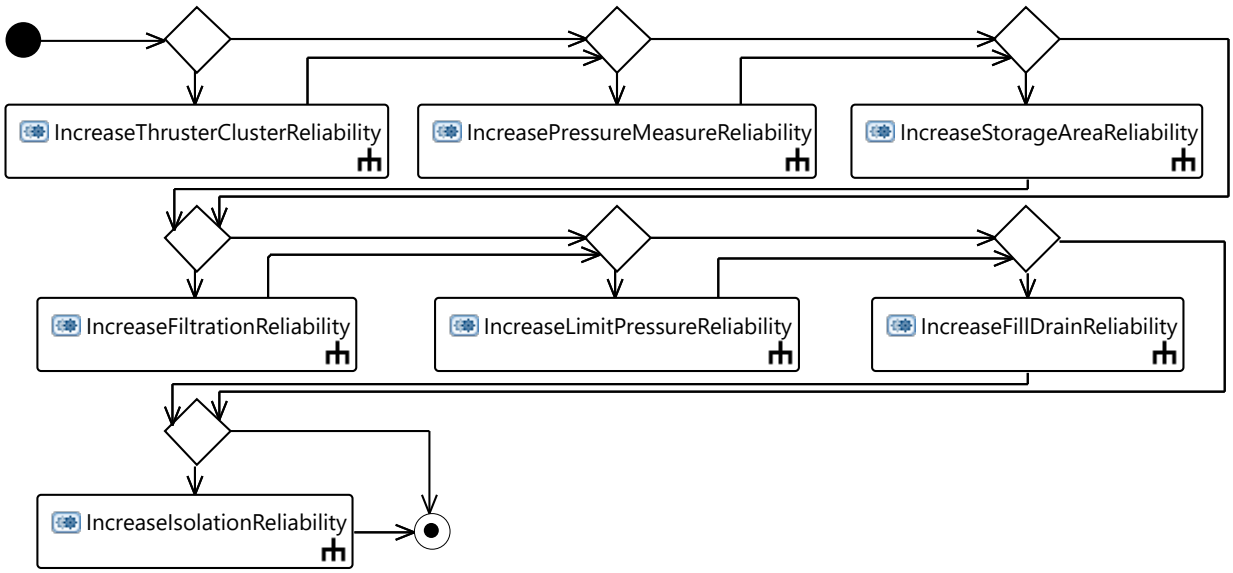

Abbildung 4.39: Aktivitätsdiagramm *IncreaseFunctionalElementsReliability*

in der Regel *ChangeRegulator* versucht durch Verwendung zuverlässigerer Komponenten, also einer Änderung der Parametrik, eine Erhöhung der Zuverlässigkeit zu erreichen. In diesem Fall umgesetzt durch eine Änderung der gesetzten Ausfallwahrscheinlichkeit der Komponente, wie in Abbildung 4.41 zu sehen. Falls keine (weitere) parametrische Erhöhung durchgeführt werden kann, wird die Redundanz der Funktion[47] erhöht, welche dann in der Satellitenantriebssystemerstellung berücksichtigt wird. Schematisch ist eine Topologieänderung, die das System redundant um einen zusätzlichen Regulator als Komponente der Funktion der Druckregelung erweitert, in Abbildung 4.42 dargestellt.

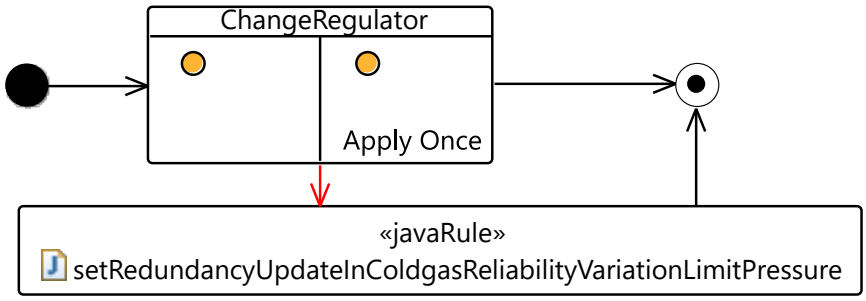

Abbildung 4.40: Aktivitätsdiagramm *IncreaseLimitPressureReliability*[48]

47 Funktionen werden durch (physikalische) Komponenten umgesetzt, die innerhalb der Funktion sozusagen in einer Blackbox sind, was die Umsetzung einer Redundanz bedeutet, da diese Redundanz hier rein auf funktionaler Ebene angegeben wird. Siehe auch Kapitel 2.3.2 Funktionale Äquivalenz

48 *Apply Once* in einer grafischen Regel bedeutet, dass diese Regel nur auf ein Suchergebnis angewandt wird und nicht auf alle. Dies hat hier den Zweck, dass bei mehreren Regulatoren, beispielsweise durch Redundanz, jeweils bei jeder Iteration nur einer verändert wird und danach geprüft wird, ob die Zielzuverlässigkeit erreicht wurde. Bei weiteren Iterationen werden weitere Regulatoren einzeln angepasst.

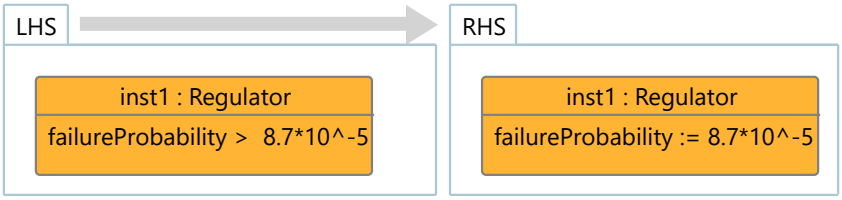

Abbildung 4.41: Regel *ChangeRegulator*

Nach dem Ablauf des Unterprogramms *IncreaseReliability* wird die Zuverlässigkeit des geänderten Systems berechnet und geprüft, ob die Zielzuverlässigkeit erreicht wurde oder topologische Systemänderungen als Maßnahmen zur Verbesserung der Zuverlässigkeit nötig sind.[49] Falls topologische Systemänderungen als Maßnahmen notwendig sind, werden diese im Rahmen der Erstellung des Satellitenantriebssystems umgesetzt. Der Ablauf der Erstellung des Satellitenantriebssystems im Unterprogramm *CreatePropulsionSystem* ist in Abbildung 4.43 dargestellt. Das Aktivitätsdiagramm besteht aus Unterprogrammen aus der Satellitenantriebsentwurfssprache. Das Unterprogramm *Topology* ist hierbei in den einzelnen internen Unterprogrammen *InitialPropulsionSystem*, *BranchesAndMerges* und *CreateFlowElements* dargestellt, das Unterprogramm *Parametrics* ist unverändert. In den Ablauf des Unterprogramms *Topology* ist ein zusätzliches Unterprogramm *SetRedundancy* integriert, in dem die Redundanzeigenschaftswerte der (Teil-)Funktionen[50] des Satellitenantriebssystems nach den Ergebnissen der Fehlerbaumanalyse gesetzt werden.[51]

Im Falle topologischer Änderungen wird das Modell unter Berücksichtigung der angepassten Redundanzen mit der Satellitenantriebsentwurfssprache neu aufgebaut, ausgelegt

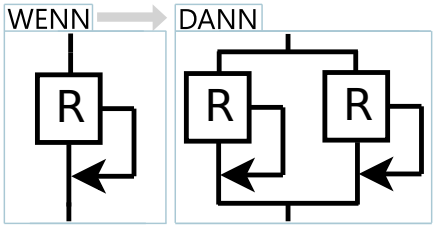

Abbildung 4.42: Schematische redundante Regulatorerweiterung

49 Ein Unterprogramm hat nur einen (Ausgangs-)Kontrollfluss, weshalb hier auch bei einer topologischen Änderung zunächst die Zuverlässigkeitsanalyse durchgeführt wird, obwohl zu diesem Zeitpunkt keine topologische Änderung am System vorgenommen wurde. Hier sind alternative Programmabläufe denkbar.

50 Die entsprechende Klasse in der Satellitenantriebsentwurfssprache hat eine Eigenschaft *redundancy* für die Anzahl der Redundanz, sowie eine Eigenschaft *redundancyType* für den Redundanztyp (parallel oder seriell).

51 In der initialen Einstellung ist die Redundanz aller Funktionen auf 0 gesetzt. Durch Änderung der Satellitenantriebsentwurfssprache können gewünschte Redundanzen im automatischen Ablauf direkt berücksichtigt werden.

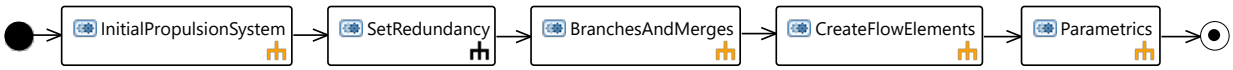

Abbildung 4.43: Aktivitätsdiagramm *CreatePropulsionSystem*

und berechnet. Nachfolgend wird die automatische Fehlerbaumanalyse des veränderten Systems durchgeführt, mit zunächst den unzuverlässigsten Komponenten[52]. Sobald am System durch die Fehlerbaumanalyse keine topologische Veränderung als Maßnahme zur Erreichung der Zuverlässigkeit als notwendig identifiziert wurde, endet die Topologievariation des Satellitenantriebssystems.

Dieser Ablauf der automatischen Fehlerbaumanalyse der hier anhand des Kaltgassatellitenantriebssystem gezeigt wurde, ist mit entsprechenden Modifikationen auch für die anderen Arten von Satellitenantriebssystemen - monergol und diergol - umgesetzt.

4.2.4 Fehlerbäume von Satellitenantriebssystemen

Die vorgestellte automatische Fehlerbaumanalyse ermöglicht die Generierung verschiedener Satellitenantriebssystemtopologien abhängig von einer Zuverlässigkeitsanforderung. Für zwei unterschiedliche Zuverlässigkeitsanforderungen[53] an Kaltgassatellitenantriebssysteme sind die Fehlerbäume mit dargestellter Zwischenebene für die funktionalen Komponenten in den Abbildungen 4.44 und 4.45 zu finden.
Das Kaltgassatellitenantriebssystem zum Fehlerbaum in Abbildung 4.44 wurde mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 5 \cdot 10^{-3}$ erstellt. Gegenüber dem Ausgangssystem, welches keine Funktion - auch nicht die Triebwerkseinheit (*ThrusterCluster*), wie in Abbildung 4.30 dargestellt - redundant ausgelegt hat, wurde ein redundantes Triebwerk (*Thruster*) hinzugefügt und beide (Ausgangs-)Druckregler (*PressureMeasure*) durch einen zuverlässigeren Druckregler ersetzt.
Für die Systemerstellung zum Fehlerbaum in Abbildung 4.45 wurde eine geforderte maximale Ausfallwahrscheinlichkeit von $F = 2,5 \cdot 10^{-3}$ gesetzt. Dieses Kaltgassatellitenantriebssystem wurde weitergehend zu den vorherigen Anpassungen gegenüber dem Ausgangssystem um ein weiteres redundantes Triebwerk erweitert und die Funktion einer Druckmessung (*PressureMeasure*) redundant durch einen weiteren Druckregler des unzuverlässigeren Ausgangstyps ausgelegt.

52 Dies basiert auf der Annahme, dass diese die billigsten Komponenten repräsentieren und deshalb vorzugsweise verwendet werden sollen.
53 Die Werte wurden exemplarisch genommen und dienen zur Verdeutlichung der erfolgten parametrischen und topologischen Änderungen. Nachfolgende Paretodiagramme zeigen die Veränderung der Systemtopologie in einer DOE (Design of experiments) Studie.

Neben der Möglichkeit der reinen Vorgabe einer Zielausfallwahrscheinlichkeit, sind beispielhaft verschiedene Zuverlässigkeitsverbesserungsstrategien[54] umgesetzt. In den Abbildungen 4.45 und 4.46 sind die Lösungen für zwei unterschiedliche Strategien bei einer vorgegebenen Zielausfallwahrscheinlichkeit ($F = 2,5 \cdot 10^{-3}$) dargestellt. In Abbildung 4.45 wird die funktionale Komponente[55] mit der höchsten Ausfallwahrscheinlichkeit angepasst, in Abbildung 4.46 ist die Redundanz mit einer niedrigeren Priorität versehen, wodurch sich die Satellitenantriebssysteme in der Topologie und Parametrik unterscheiden, wie in den Fehlerbaumdarstellungen zu sehen ist.
Bei erstgenanntem Kaltgassatellitenantriebssystem wurden gegenüber dem Ausgangssystem die Druckregler ausgetauscht, sowie eine Funktion der Druckmessung redundant ausgelegt. Zusätzlich wurde die Funktion der Triebwerkseinheit doppelt redundant ausgelegt, um die Zielausfallwahrscheinlichkeit zu erreichen.
Die zweite Strategie veränderte die Topologie gegenüber dem Ausgangssystem nur durch ein redundantes Triebwerk in der Funktion der Triebwerkseinheit. Um die Zielausfallwahrscheinlichkeit zu erreichen, wurden aber neben den Druckreglern, für den Tank (*GasTank*), den Regulator, den Filter und ein Befüll-/Entleerventil (*FillDrainValve*) parametrisch hinsichtlich der Zuverlässigkeit bessere Komponenten erstellt.

Weitere Fehlerbäume von automatisch parametrisch und topologisch für andere Zuverlässigkeitsanforderungen angepasste Kaltgassatellitenantriebssysteme sowie für monergole Satellitenantriebssysteme mit unterschiedlichen Randbedingungen sind im Anhang B zu finden, da die maschinell generierten Systeme teilweise größer werden und die resultierenden Fehlerbaumdarstellungen über für eine DIN-A4 Seitengröße ansehnliche Verhältnisse hinausgehen und nichts prinzipiell Neues darstellen.

54 Neben der Zuverlässigkeitsverbesserung der Komponente mit der höchsten Ausfallwahrscheinlichkeit, kann bei den Strategien beispielsweise zusätzlich festgelegt werden, ob Redundanz mit einer niedrigen Priorität belegt wird, so dass zunächst parametrische Zuverlässigkeitsverbesserungen an Komponenten vorgenommen werden, die nicht die höchste Ausfallwahrscheinlichkeit im momentanen Fehlerbaum auf der entsprechenden Ebene haben und Topologieänderungen erst nach Ausschöpfen dieser Möglichkeiten genutzt werden. Eine weitere Strategie ist die entsprechend umgekehrte Priorität.

55 Der Ausdruck funktionale Komponente bedeutet hier, dass bei parametrischen Änderungen direkt die Komponente, bei topologischen Änderungen die Funktion, die hier mit einer Komponente umgesetzt wird, angepasst wird.

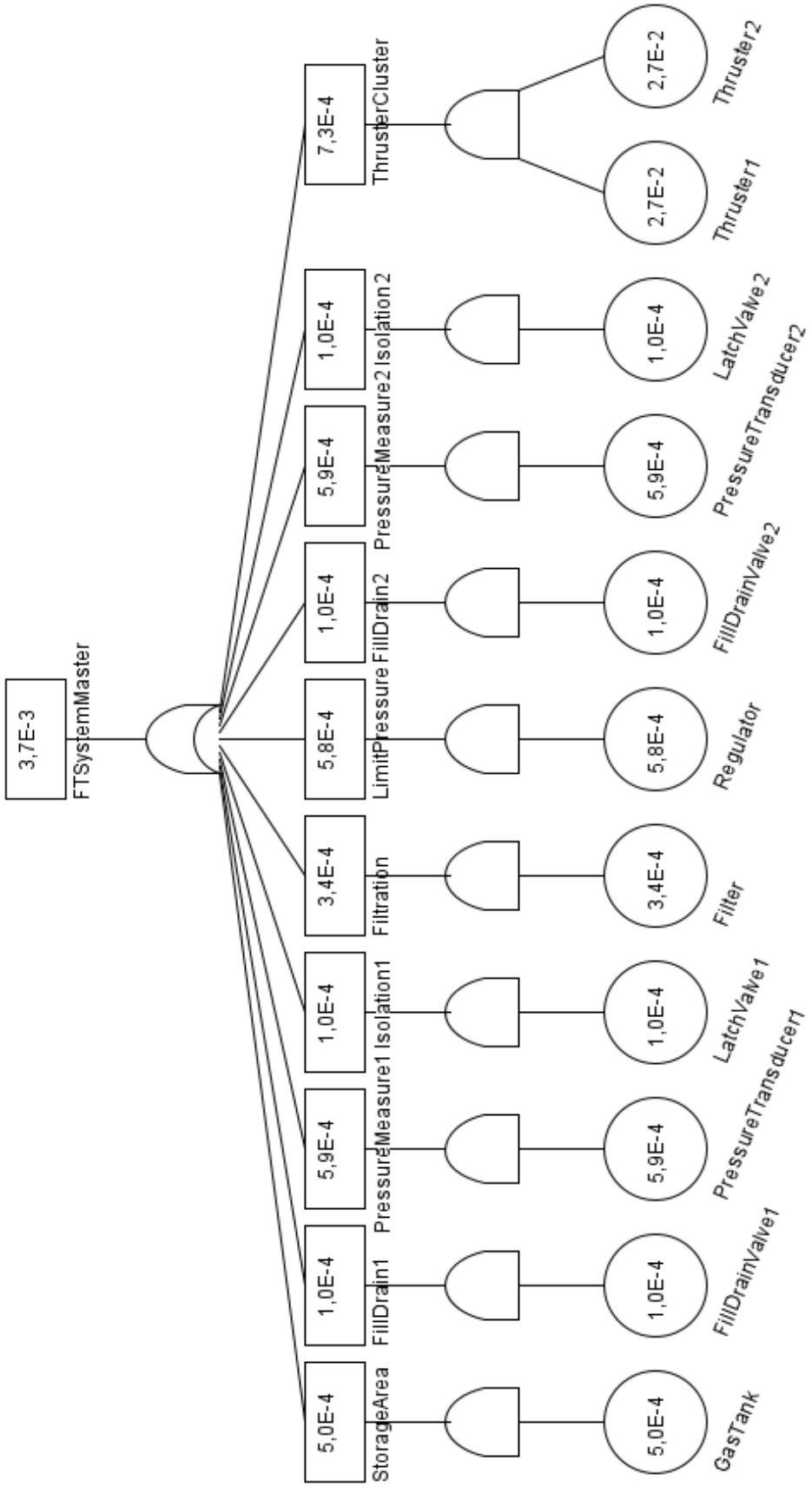

Abbildung 4.44: Fehlerbaum des Kaltgassatellitenantriebssystems mit $F \leq 5 \cdot 10^{-3}$

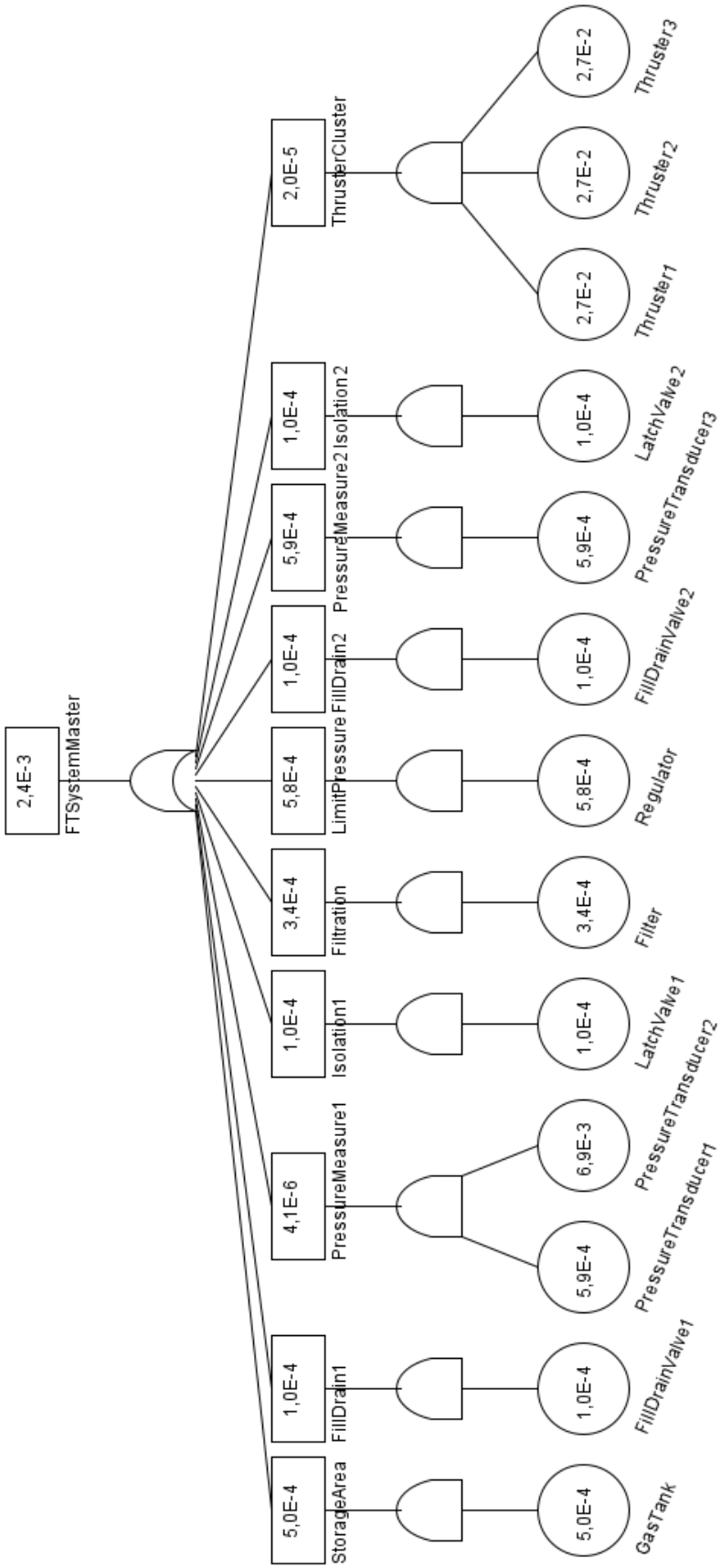

Abbildung 4.45: Fehlerbaum des Kaltgassatellitenantriebssystems mit $F \leq 2,5 \cdot 10^{-3}$

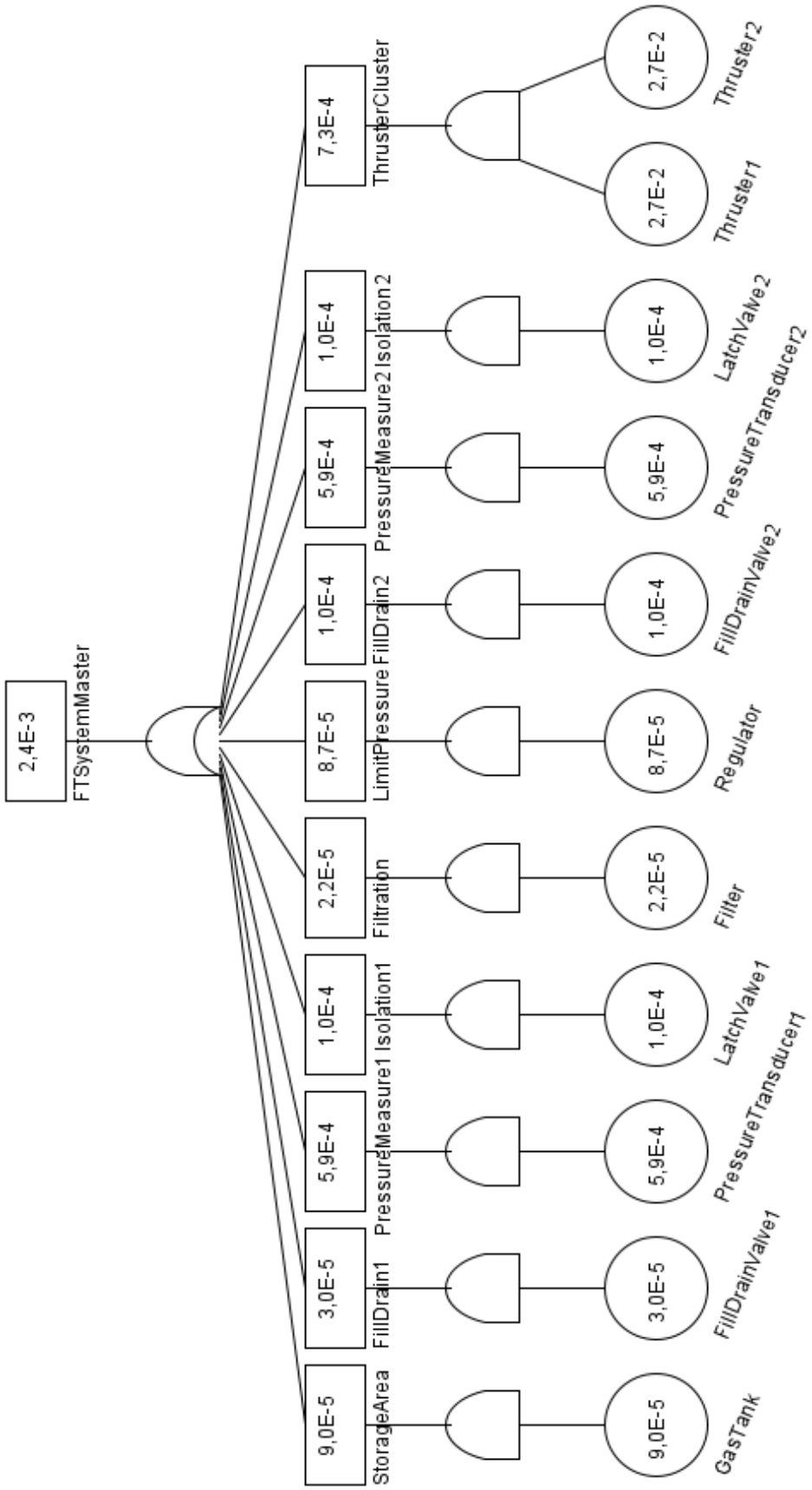

Abbildung 4.46: Fehlerbaum des Kaltgassatellitenantriebssystems mit $F \leq 2,5 \cdot 10^{-3}$ und Priorisierung von Parameteränderungen

Das Problem der visuellen Darstellung und Analyse einer Vielzahl von Fehlerbäumen, die insbesondere bei ergebnisoffeneren Vorgaben automatisch untersucht werden, wird hier durch die Darstellung der Ergebnisse - in einer Design of experiments Studie (DOE) - in Diagrammen angegangen. Diese Diagramme setzen Leistungsparameter von Systemtopologien und -parametrisierungen in Zusammenhang, wie in Abbildung 4.47 für die Zuverlässigkeit über der Masse des Kaltgassatellitenantriebssystems zu sehen.
Jeder Datenpunkt stellt in diesem Diagramm ein einzelnes Satellitenantriebssystem dar, welches während der Ausführung der Entwurfssprache erstellt und hinsichtlich der Zuverlässigkeit ausgewertet wurde. Topologieänderungen des Systems erfolgten, wie vorherig beschrieben, durch Änderungen der Redundanz der (Teil-)Funktionen[56] der Satellitenantriebsentwurfssprache. Die selben Farben für Datenpunkte im Diagramm bedeuten die selbe Topologie mit unterschiedlichen Parametern für die Ausfallwahrscheinlichkeit und dementsprechend den zugehörigen Massen der Komponenten. Ein Farbwechsel im Diagramm stellt eine Topologieänderung des Satellitenantriebssystems dar, im Fall des vorgestellten Satellitenantriebssystems ist dies das redundante und funktional äquivalent vorausgesetzte Hinzufügen der selben Funktion erzielt über das Hinzufügen einer die Funktion erfüllenden Komponente.
Im dargestellten Diagramm wurde eine maximale Ausfallwahrscheinlichkeit von $F = 5 \cdot 10^{-6}$ gefordert, die mit unterschiedlichen Zuverlässigkeitsverbesserungsstrategien jeweils vom Ausgangssystem beginnend erreicht wurde.[57] Das Ausgangssatellitenantriebssystem für das Diagramm hatte ein Triebwerk, weshalb der erste Topologiewechsel[58] ein redundantes Triebwerk hinzufügt.[87]
Auffällig im Diagramm sind zwei größere Sprünge zwischen einzelnen Topologieänderungen. Der große Sprung in der Zuverlässigkeitsachse beim ersten Hinzufügen eines redundanten Triebwerks entsteht auf Grund der relativ zu den anderen Komponenten hohen Ausfallwahrscheinlichkeit des zuverlässigsten Triebwerkstyp. Der Sprung in der Masse des Satellitenantriebssystems ist auf die Integration eines zusätzlichen Tanks zurückzuführen, der entsprechend mit Treibstoff gefüllt ist.

56 Ausführungen zur Voraussetzung funktionaler Äquivalenz sind in Kapitel 2.3.2 zu finden.
57 Im dargestellten Diagramm in Abbildung 4.47 sind 262 Satellitenantriebssysteme als Datenpunkte dargestellt, die Laufzeit der Entwurfssprache betrug insgesamt circa sechs Minuten.
58 Betrachtungsreihenfolge ist hierbei von der niedrigsten Zuverlässigkeit mit der niedrigsten Masse ausgehend, in Farben: braun, ocker, grün, hellgrau, usw.

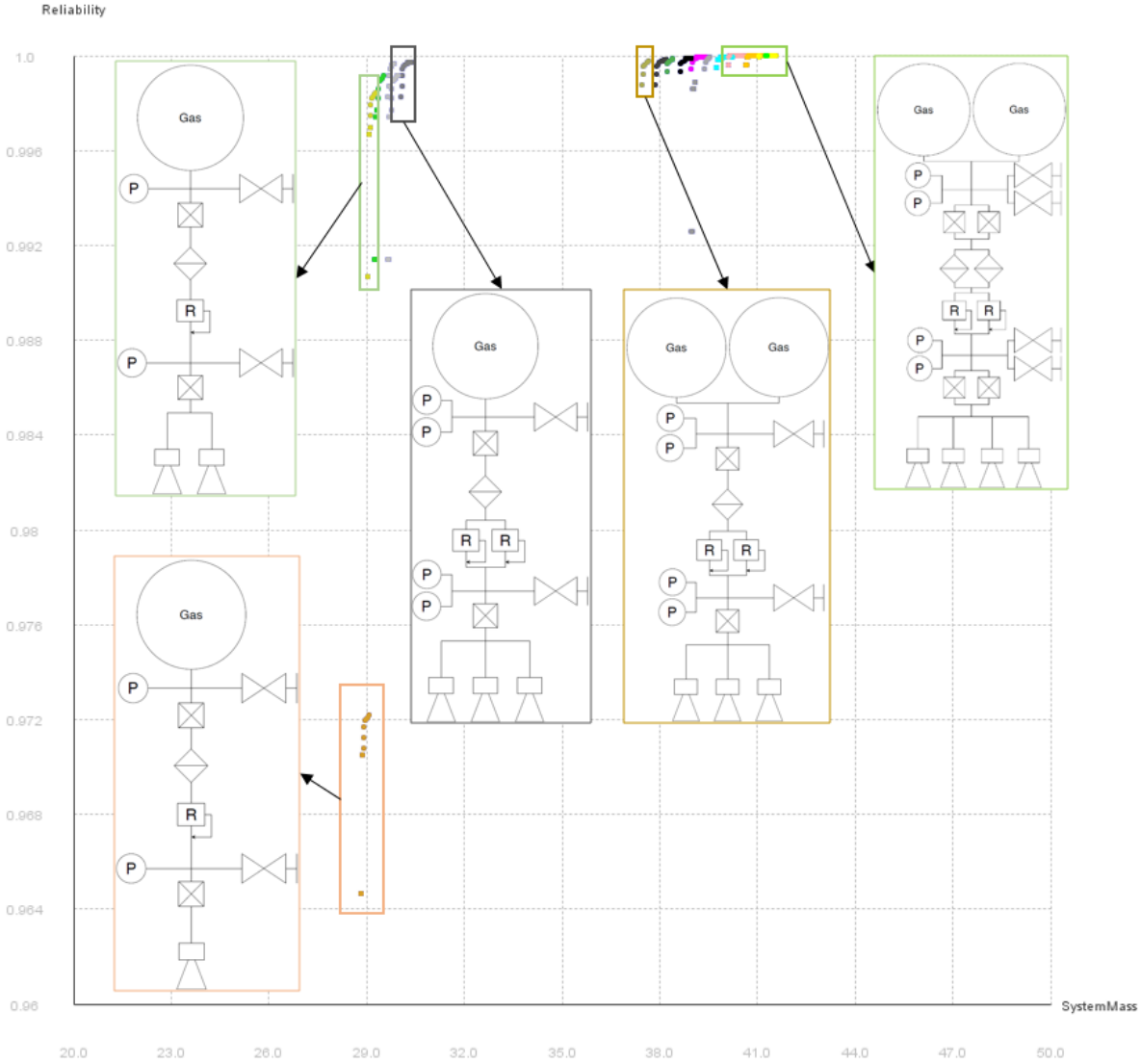

Abbildung 4.47: Zuverlässigkeit über Masse Diagramm der Kaltgassatellitenantriebssystemauswertungen bis zu einem Zielwert von $F \leq 5 \cdot 10^{-6}$

Die Integration der Fehlerbaumerstellung in die komplexere Satellitenantriebssystementwurfssprache ist in diesem Beispiel gezeigt worden. Über eine einfache Verknüpfung der Klassendiagramme kann eine automatische Fehlerbaumerstellung für ein bereits modelliertes System ermöglicht werden. Danach muss das Expertenwissen für die Fehlerbaumanalyse über Regeln im Rahmen der Systemanalyse eingepflegt werden. Das System kann anschließend hinsichtlich der Verbesserung der Zuverlässigkeit sowohl in der Konfiguration als auch in der Topologie - unter Beibehaltung der Systemfunktion - verändert und direkt eine angepasste Fehlerbaumanalyse durchgeführt werden. Hiermit ist sowohl eine Erstellung einer Design of experiments Studie möglich, als auch eine automatische Anpassung eines Systems hinsichtlich (Zuverlässigkeits-)Vorgaben.

4.3 Diskussion

Anhand der beschriebenen Anwendungsbeispiele wurde die automatische Fehlerbaumerstellung- und -analyse mit unterschiedlichen, interessanten Teilaspekten und Einsatzmöglichkeiten vorgestellt. Die Besonderheiten der Beispiele werden nachfolgend diskutiert und eine Einordnung anhand der Begriffe Vollständigkeit und Konsistenz vorgenommen.

4.3.1 Anwendungsbeispielvergleich

Schiffsmotor Das Beispiel des Schiffsmotors fokussiert sich auf die automatische Fehlerbaumerstellung. In der Modellierung des Systems wird eine Fehlerbaumerstellung von Beginn an mit berücksichtigt. Auf Grund dieser frühen Berücksichtigung wird ein modularer Aufbau der einzelnen Systeme und Komponenten erleichtert. Dieser modulare Aufbau zeigt sich besonders in der Einbindung des Schiffsmotormodells in ein übergeordnetes Schiffsantriebssystem. Neben der automatischen Fehlerbaumerstellung wurde das Modell um eine einfache Fehlermöglichkeits- und -einflussanalyse (FMEA) erweitert, die die Ergebnisse der qualitativen Fehlerbaumanalyse zur Grundlage hat. Diese Verknüpfung der Methoden demonstriert eine Möglichkeit der Verbindung zwischen einer Zuverlässigkeits- und einer Risikoanalysemethode.

Satellitenantriebssystem Das Beispiel der Satellitenantriebssysteme zeigt die automatische Fehlerbaumanalyse. Dafür wurde eine Fehlerbaumerstellung in ein existierendes Modell integriert. Diese Integration erfolgt durch eine Modellerweiterung sowie eine Verknüpfung mit dem Klassendiagramm der Zuverlässigkeitsberechnung. Der Schritt von einer automatischen Fehlerbaumerstellung mit Auswertung zur Fehlerbaumanalyse wird über eine zusätzliche Entwurfssprache, die die Satellitenantriebsentwurfssprache verwendet, durch Einbindung einer Heuristik erreicht. Über diese Heuristik wird das System hinsichtlich der Zuverlässigkeit verbessert, indem automatisch Schwachstellen des Systems durch eine quantitative und qualitative Fehlerbaumanalyse identifiziert werden und gezielt an diesen Verbesserungen vorgenommen werden. Dieses Beispiel zeigt, wie Zuverlässigkeit als Bewertungskriterium für ein System herangezogen und auf Grundlage dieses Bewertungskriteriums das System parametrisch und topologisch angepasst werden kann, um einen vorgegebenen Bewertungswert zu überschreiten. Eine solche Einbindung in den Produktentwurf mit eingebundener Optimierung, auch als Round-Trip-Enginneering bezeichnet, ist ein wichtiger Schritt, um eine automatische Produkterstellung auf Basis der Randbedingungen zu realisieren. Darüber hinaus wurde die Mög-

lichkeit einer Entwurfsraumuntersuchung mit einer DOE-Studie (design of experiments study) vorgestellt, die durch die automatische Erzeugung verschiedener Produkttopologien und -parametriken mit automatischer Zuverlässigkeitsberechnung einen schnellen Überblick in Bezug auf Zuverlässigkeit gegenüber anderen automatisch berechneten Entwurfsparametern für Satellitenantriebssysteme erlauben.

4.3.2 Vollständigkeit

Die Vollständigkeit der automatischen Fehlerbaumanalyse ist durch die Modellierung [105] bedingt.[59] Folglich wird aus der Modellierung ein vollständiger, zugehöriger Fehlerbaum hinsichtlich der für eine Fehlerbaumanalyse semantisch als relevant annotierten und kausal verknüpften Entitäten erstellt. Der Unterschied zu manuell erstellten Fehlerbäumen, bei denen Vereinfachungen für erwartbar vernachlässigbare Fehlerbaumäste vorgenommen werden, liegt hierbei darin, dass durch die Automatisierung der Fehlerbaum an allen Stellen vollständig entwickelt wird. Diese Vereinfachung im manuellen Prozess bietet, neben weniger Arbeitsaufwand, eine bessere Übersichtlichkeit in der Fehlerbaumdarstellung, birgt aber die Gefahr, dass bei sich verändernden Systemen vernachlässigte Äste, die einen signifikanten Einfluss durch die bzw. nach der Änderung bekommen, nicht beachtet werden.

Ein weiterer Vorteil der Vollständigkeit des Fehlerbaumes liegt darin, dass über einen vollständigen Fehlerbaum im Falle einer Verknüpfung mit einer Fehlermöglichkeits- und -einflussanalyse (FMEA) direkt eine aussagekräftigere FMEA möglich ist. Dies ist darin begründet, dass es sich bei der FMEA um eine Risikoanalyse handelt, in der für die Zuverlässigkeit des Gesamtsystems vernachlässigbare Elemente, die in einer manuellen Fehlerbaumanalyse aus dem Modell entfernt wurden, eine signifikante Auswirkung haben können[60] und somit zu Änderungen am Modell führen.

Ebenfalls unter dem Aspekt der Kausalität ist die Vollständigkeit eines Fehlerbaumes in Bezug auf das Modell erstrebenswert, da das Wirkungsereignis in einem unvollständigen Fehlerbaum durch das Auslassen von Ursachenereignissen auftreten kann, ohne dass eine Ursache im Fehlerbaum hinterlegt ist. Die Vollständigkeit der Fehlerbaumanalyse in Bezug auf das Modell verhindert einen sich in der kontrafaktischen Kausalität entsprechend ergebenden Widerspruch. Der Fehlerbaum ist für das Modell dementsprechend hinsichtlich Kausalität vollständig und aussagekräftig[61].

59 D.h. die Fehlerbaumanalyse ist vollständig für die zu Grunde liegende Modellierung des Systems.
60 Hohe Risikoprioritätszahl auf Grund der Bedeutung und der Entdeckungswahscheinlichkeit bei niedriger Auftretenswahrscheinlichkeit.
61 Ereignis A ist kausal für Wirkung B im zugehörigen Modell.

4.3.3 Konsistenz

Im multidisziplinären Produktentwurf ist in Bezug auf unterschiedliche Aspekte, wie Zuverlässigkeit, Kosten, Nachverfolgbarkeit etc., Konsistenz zwischen den einzelnen Domänen wichtig. Die Zuverlässigkeitsanalyse ist eine dieser Domänen, die bei einem geänderten Modell konsistent gehalten werden muss, um auf Grund der Änderungen entstehende Probleme frühzeitig zu identifizieren und Übertragungsfehler zu vermeiden. Durch die Automatisierung der Zuverlässigkeitsanalyse ist die einfache Erstellung zugehöriger, konsistenter Zuverlässigkeitsmodelle auf Grundlage des Systemmodells nahezu zum Änderungszeitpunkt möglich.
Im Beispiel des Schiffsmotors wurde die Integration des (Motor-)Modells in ein Modell eines Schiffsantriebssystems gezeigt, was den modularen Charakter der Zuverlässigkeitsanalyse demonstriert. Dieser modulare Charakter erlaubt die Erstellung einzelner Fehlerbäume für die Komponenten, anhand derer jeweils entkoppelte Zuverlässigkeitsanalysen durchgeführt werden können.[62] Gleichzeitig ist die Zusammenführung in einem übergeordneten Fehlerbaum eines Systems, das die Teilsysteme, in diesem Fall Komponenten, integriert, möglich. Hierbei ist die Konsistenz[63] innerhalb der einzelnen Zuverlässigkeitsanalysemodelle gegeben, die automatisch aus dem Systemmodell erstellt werden können. Zusätzlich wurde der Zusammenhang mit einer Risikoanalyse mit der Verknüpfung zwischen der Fehlermöglichkeits- und -einflussanalyse und der Fehlerbaumanalyse gezeigt. Über diese Verknüpfung werden generierte Analysen aus diesen Bereichen im zentralen Modell konsistent zueinander gehalten.
Beim Satellitenantriebssystembeispiel findet die Konsistenz des Systemmodells mit dem Modell der Zuverlässigkeitsanalyse unmittelbar in der Verbesserung des Systemmodells ihre Verwendung. Auf Basis des Systemmodells wird hier ein zugehöriges, konsistentes Zuverlässigkeitsmodell erstellt, welches zu einer Modelländerung führen kann, die wiederum zu einer Anpassung des Zuverlässigkeitsmodells führt. In diesem Fall wird die Systemmodelländerung durch die Zuverlässigkeitsanalyse angestoßen, aber es wird eine Systemmodelländerung vorgenommen, die ebenfalls durch andere Domänen angestoßen werden kann, wie beispielsweise in der vorherigen Auslegung des Satellitenantriebssystems mit einer geforderten Schubvorgabe.

62 Dekomposition des Gesamtsystems bei funktionaler Äquivalenz, siehe Kapitel 1.3 Problemstellung und Gliederung und Kapitel 2.3.2 Funktionale Äquivalenz sowie Anhang C Mathematische Ausführungen: Brückensystem

63 Die modularen (Teil-)Systeme sind konsistent zum Gesamtsystem und umgekehrt, d.h. eine Änderung eines (Teil-)Systems bedingt eine Änderung des Gesamtsystems und eine Änderung des Gesamtsystems bedingt die Änderung von (Teil-)Systemen.

Kapitel 5

Zusammenfassung

Aus der Problemstellung des Front-Loadings zur dringenden Verbesserung der Einsetzbarkeit der Zuverlässigkeitsanalyse im Produktentwurf heraus wurde eine für den konzeptionellen Produktentwurf mit vertretbarem Arbeitsaufwand begleitende, frühzeitige Zuverlässigkeitsanalyse als Lösung abgeleitet. Diese Zielstellung wurde mit einer Automatisierung der Zuverlässigkeitsanalyse, in Form der Fehlerbaumanalyse, umgesetzt. Diese Zuverlässigkeitsanalyse ermöglicht es, bei Bedarf parametrische und topologische Änderungen der Produktarchitektur vorzunehmen. Damit wird die Fehlerbaumanalyse als prägende Entwurfsfunktion im konzeptionellen Vorentwurf verfügbar.
Für die Automatisierung wurde dabei zunächst eine Ontologie erstellt, die die verschiedenen Zuverlässigkeitsvokabeln beinhaltet. Aufbauend auf dieser Ontologie können Systeme semantisch annotiert und anschließend im Rahmen der Zuverlässigkeitsanalyse automatisch analysiert werden. Diese Analyse wurde mit der in der Zuverlässigkeitsanalyse weit verbreiteten Methode der Fehlerbaumanalyse verwirklicht. Dabei wird im Rahmen der Systemanalyse das System semantisch annotiert und kausal verknüpft. Aus dem semantisch annotierten System lassen sich dann zugehörige Fehlerbäume erstellen und diese auswerten. Mit den Ergebnissen der Fehlerbaumauswertung wird nachfolgend das System über den Einsatz von Entwurfssprachen auf Basis der Randbedingungen unter Verwendung einer Zielvorgabe automatisch parametrisch und topologisch angepasst.

In diesem Kapitel werden die Ergebnisse zusammengefasst und abschließend ein Ausblick auf zukünftige interessante Forschungsbereiche, die zur Erweiterung der dargestellten Arbeit dienen, gegeben.

5.1 Ergebnisse

Durch die domänenspezifische Wissensmodellierung über eine abstrakte Ontologie im Bereich der Zuverlässigkeitsanalyse wurde die Möglichkeit geschaffen, Systeme in frühen

Phasen des Produktentwurfes hinsichtlich ihrer Zuverlässigkeit mit reduziertem Arbeits- und Zeitaufwand zu untersuchen. Dies dient der Kostenreduzierung und dem effektiveren Produktentwurf, da kritische Systemstellen in Bezug auf die Zuverlässigkeit früher erkannt und diese entsprechend angepasst werden können und nicht nachträglich Verbesserungen an das System angefügt werden, die bei früherer Erkennung nicht so gelöst worden wären.
Darüber hinaus wurde es ermöglicht aus einem semantisch annotieren und kausal verknüpften System direkt Fehlerbäume zu erstellen, sofern die entsprechenden Daten im Modell hinterlegt wurden. Diese Modellierung in einem zentralen Modell führt zu einem konsistenten Fehlerbaum gegenüber den anderen Modellansichten, die im Rahmen des Produktentwurfes aus diesem zentralen Modell generiert werden, wie beispielsweise ein Geometrie- oder Thermalmodell. Neben der Konsistenz zu anderen domänenspezifischen Modellen ermöglicht die automatische Fehlerbaumerstellung auch eine einfachere Anpassung des Fehlerbaummodells bei Modellmodifikationen, die im Produktentwurf häufig nötig sind. Dank der einfacheren automatischen Anpassung verliert die Problematik, dass ein Fehlerbaum erst später im Entwurfszyklus erstellt wird, weil durch eine Änderung dieser an Gültigkeit verloren hat und ein hoher Aufwand für die erneute Fehlerbaumerstellung von Nöten ist, an Gewichtung. Zusätzlich ermöglicht dies z.B. im Rahmen einer Design of experiments (DOE) Studie[1] eine Untersuchung des Entwurfsraums unter Berücksichtigung der Zuverlässigkeit.

Die automatische Zuverlässigkeitsanalyse wurde in unterschiedlicher Anwendung anhand von zwei Beispielen vorgestellt. Im ersten Beispiel wurde ein Schiffsmotor bis auf die einzelnen Fehlermodi der Komponenten modelliert und durch eine Integration in ein übergeordnetes Schiffsantriebssystem die Modularisierung einzelner Zuverlässigkeitsanalysen mittels Fehlerbäumen gezeigt. Des Weiteren wurde eine Verbindung der Zuverlässigkeitsanalyse zu einer automatischen Fehlermöglichkeits- und -einflussanalyse demonstriert. Im zweiten Beispiel wurde ein komplexeres System mit den Satellitenantriebssystemen betrachtet. In eine entsprechende Entwurfssprache wurde die automatische Fehlerbaumerstellung und -auswertung nachträglich integriert, was wiederum die modulare Einsatzfähigkeit der Ontologie und hier im Speziellen der Entwurfssprachen zeigt. Außerdem wurde die prinzipielle Möglichkeit einer ebenfalls automatischen parametrischen und topologischen Veränderung des Systems auf Grundlage der Analyseer-

1 Abbildung 4.47

gebnisse gezeigt, womit die vollständige[2], automatisierte Methode der Fehlerbaumanalyse demonstriert wurde. Darüber hinaus wird die Fehlerbaumanalyse somit als zunächst konzeptuelle Entwurfsfunktion im Vorentwurf komplexer Systeme verfügbar.

5.2 Ausblick

Neben den positiven Ergebnissen geht der Einsatz der Ontologie durch die Wissensmodellierung zunächst mit einem relativ großen Aufwand einher, der am Anfang getätigt werden muss, um die Vorteile der Automatisierung mit Entwurfssprachen[3] nutzen zu können. Eine Lösung zur Verbesserung dieser Problematik ist eine einheitliche Wissensmodellierung allgemein im Bereich der Zuverlässigkeit und speziell für die Erstellung von Fehlerbäumen, beispielsweise in der Form der Erstellung von Datenbanken, die für entsprechende Komponenten Zuverlässigkeits- und Fehlerbaumdaten enthält. Neben diesen Datensätzen sind auch Datensätze für Regelsammlungen für die semantische Annotation und kausale Verknüpfungen von Systemen oder die topologische Umgestaltung dieser ein weiteres zukünftiges Ziel, das den Einsatz der automatischen Zuverlässigkeitsberechnung und Fehlerbaumerstellung vereinfacht und sinnvoll erweitern wird.
Ein weiterer, zukünftiger Aspekt ist, abseits der Erweiterung und Detaillierung der Fehlerbaum- und Fehlermöglichkeits- und -einflussanalyse beispielsweise in Richtung der eingrenzenden, geforderten Zeitinvarianz hin zu einer dynamischen Zuverlässigkeitsanalyse, die Beschäftigung mit weiteren Zuverlässigkeitsanalysemethoden, um sie hinsichtlich einer Automatisierung als sinnvolle Ergänzung oder Alternative einstufen zu können. Darüber hinaus ist, da in dieser Arbeit vornehmlich auf den Entwurfsparameter der Zuverlässigkeit eingegangen wurde, parallel dazu eine Untersuchung von ableitbaren Größen, wie der Importanz oder der Durchschnittszeit bis zum Fehlerauftritt (mean time to failure (MTTF)), interessant. Zudem sind die Integration und Automatisierung von sowie das Zusammenspiel mit angrenzenden Themenbereichen, wie Wartbarkeitsanalyse, Verfügbarkeitsanalyse, Risikoanalyse, Sicherheitsanalyse und funktionale Sicherheit, weitere, interessante Schritte, um die Digitalisierung im Produktentwurf voranzutreiben.

2 In Bezug auf die Methode der Fehlerbaumanalyse, also die einzelnen Schritte der Methode, wie in Kapitel 2.1.1 Fehlerbaumanalyse (FTA) beschrieben.

3 Dieser Mehraufwand ist kennzeichnend für die Mehrheit graphenbasierter Entwurfssprachen, da die gesamte Domäne vorab in einem Klassendiagramm dargestellt werden muss. Dafür sind alle weiteren, aus der Entwurfssprache heraus generierten Modelle relativ günstig zu erzeugen.

Anhang A

Beispiel: Beleuchtungssystem

Anhand dieses Beispiels wird demonstriert, wie die Erstellung einer Entwurfssprache auf Grundlage einer zu erreichenden Funktionsanforderung umgesetzt wird. Zusätzlich wird schrittweise die Einbindung einer Zuverlässigkeits- und Risikoanalyse mit automatischer Systemanpassung dargestellt.
Ausgangspunkt ist die Funktion *Beleuchtung*. Für diese Funktion wird als Lösung ein *Beleuchtungssystem* erstellt. In Abbildung A.1 ist der entsprechende elektrische Schaltkreis der umgesetzten Lösung des Beleuchtungssystems als Schaltplan dargestellt. Der Schaltkreis besteht dabei aus den Komponenten *Schalter*, *Energiequelle* und *Lampe*.
Um eine automatische Zuverlässigkeitsanalyse mit Fehlerbaumanalyse durchzuführen wird das System zunächst modelliert. Danach folgt auf Grundlage der Modellierung mit Hilfe semantischer Annotation die Einbindung der automatischen Fehlerbaumanalyse. Darauf folgend wird das System unter dem Gesichtspunkt der Fehlermöglichkeits- und -einflussanalyse (FMEA) semantisch annotiert und die Integration der automatischen Fehlerbaumanalyse in die FMEA gezeigt. Abschließend wird eine automatische Anpassung des Beleuchtungssystems anhand der Ergebnisse der Fehlerbaumanalyse hinsichtlich der Zuverlässigkeit über Variation der Parametrik und Topologie vorgestellt.

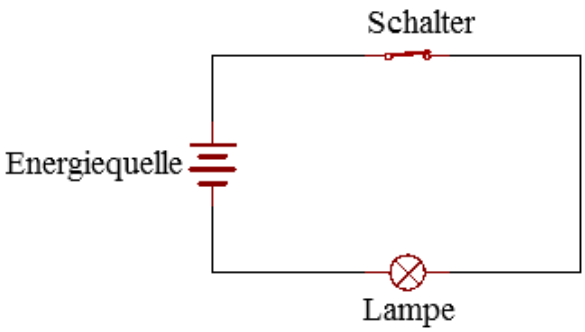

Abbildung A.1: Beleuchtungssystem

A.1 Modellierung

Das Beleuchtungssystem - Klasse *Beleuchtungssystem* - besteht aus Komponenten des Beleuchtungssystems, dargestellt als Klasse *KomponentenBeleuchtungssystem*. Die spezifischen Komponentenklassen, die die Klasse *KomponentenBeleuchtungssystem* beerben, ergeben sich aus den Komponenten Energiequelle, Lampe und Schalter, die in der Modellbildung als Klassen mit den entsprechenden Namen *Energiequelle*, *Lampe* und *Schalter* realisiert wurden. Das zugehörige UML-Klassendiagramm zeigt Abbildung A.2.

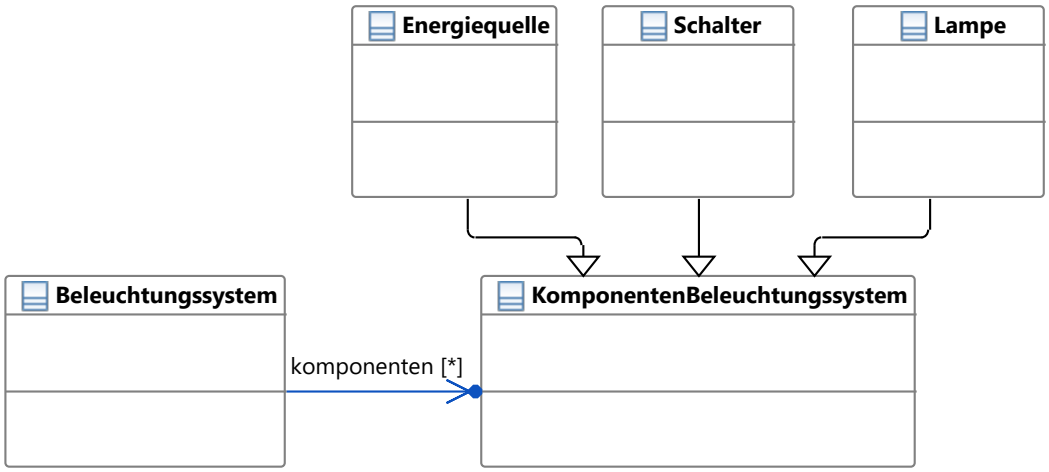

Abbildung A.2: Klassendiagramm des Beleuchtungssystems in der UML

Aus diesen Klassen, die als Vokabeln der Entwurfssprache dienen, kann über die Regel *ErstelleBeleuchtungssystem*, zu sehen in Abbildung A.3, die im Aktivitätsdiagramm (Abbildung A.4) integriert ist, ein Modell des Beleuchtungssystems erstellt werden.

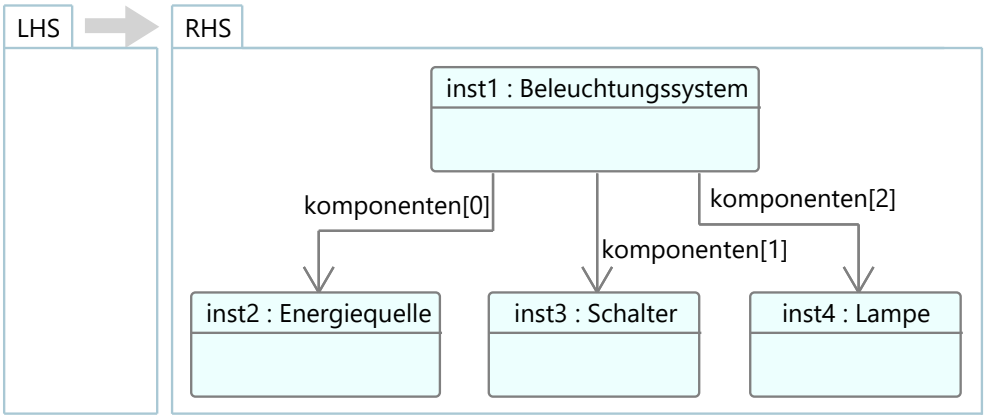

Abbildung A.3: Regel *ErstelleBeleuchtungssystem*

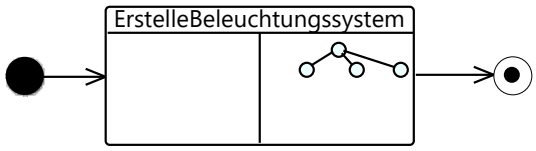

Abbildung A.4: Aktivitätsdiagramm zur Erstellung des Beleuchtungssystems

Modellerweiterung zur Fehlerbaumerstellung Aufbauend auf dem modellierten System muss für eine Zuverlässigkeitsanalyse das System mit entsprechendem Vokabular erweitert werden, wie im um die Zuverlässigkeitsanalyse erweiterten Klassendiagramm des Beleuchtungssystems in Abbildung A.5 dargestellt. Für diese Erweiterung wird das Klassendiagramm des Beleuchtungssystems mit dem Klassendiagramm der Zuverlässigkeitsanalyse, welches in Kapitel 3.2 vorgestellt wurde, verknüpft. Diese Verknüpfung ermöglicht die Verwendung der Vokabeln aus dem Klassendiagramm der Zuverlässigkeitsanalyse in der Modellierung des Beleuchtungssystems. Im nächsten Modellierungsschritt wird eine Klasse *Defekt* erstellt, welche von der Klasse *KomponentenBeleuchtungssystem*, aus dem Klassendiagramm des Beleuchtungssystems, assoziiert[1] wird. Diese Klasse *Defekt* erbt von der Klasse *FaultTreeObject*, aus dem verknüpften Klassendiagramm der Zuverläs-

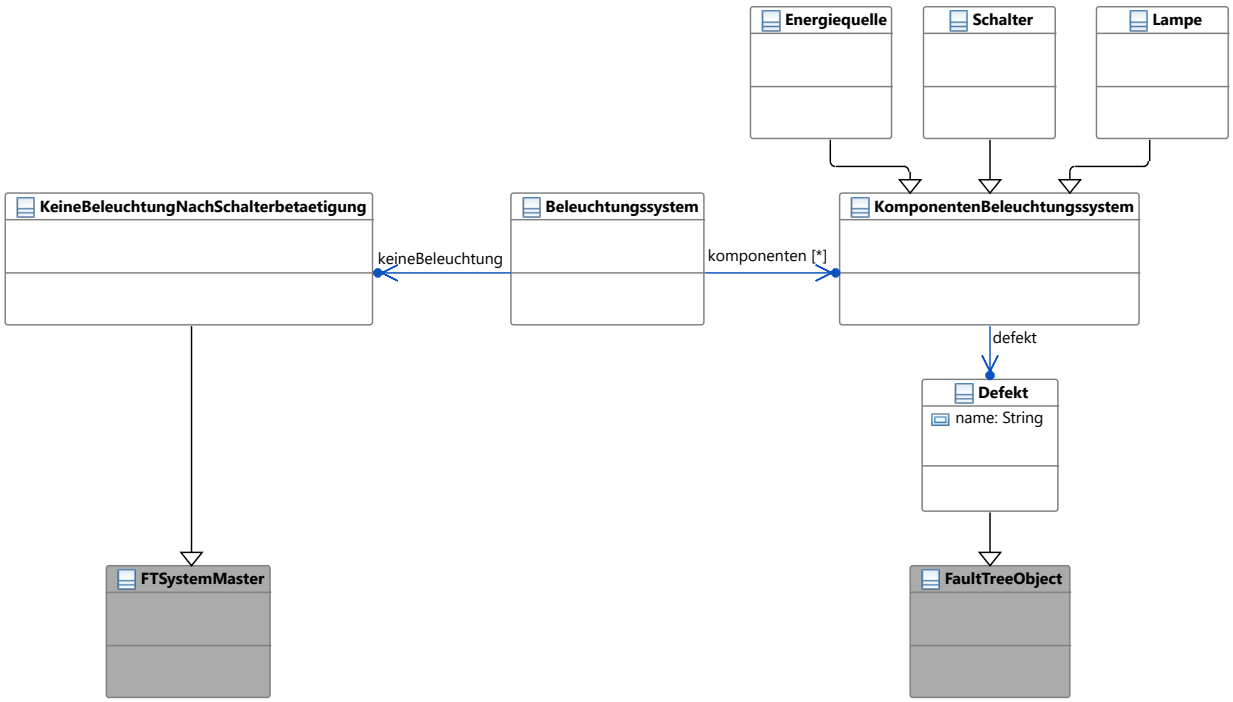

Abbildung A.5: Verknüpfung des Beleuchtungssystemklassendiagramms mit dem Zuverlässigkeitsanalyseklassendiagramm

1 Dies bedeutet, dass eine Komponente des Beleuchtungssystem einen Defekt haben kann, wie nachfolgend in den Regeln umgesetzt.

sigkeitsanalyse. Dadurch ist jede Instanz der Klasse *Defekt* ein Fehlerbaumobjekt, welches von einer Komponente des Beleuchtungssystems verknüpft werden kann.
Des Weiteren wird die Klasse *Beleuchtungssystem* über eine Assoziation mit der Klasse *FTSystemMaster* verknüpft und kann somit einen Funktionsfehler haben. Im Klassendiagramm ist diese Assoziation direkt zur konkreteren, neu erstellen Klasse *KeineBeleuchtungNachSchalterbetaetigung* gezogen worden. Diese Zusammenhänge sind im um die Zuverlässigkeitsanalyse erweiterten Klassendiagramm des Beleuchtungssystems in Abbildung A.5 zu sehen.

Neben dem Klassendiagramm wird das Aktivitätsdiagramm des Beleuchtungssystems für die Integration der Fehlerbaumanalyse erweitert. In Abbildung A.6 ist das entsprechend erweiterte Aktivitätsdiagramm abgebildet. In der zusätzlichen Regel *ErstelleFunktionsfehler* wird ein Funktionsfehler *KeineBeleuchtungNachSchalterbetaetigung* erstellt, im darauffolgenden hinzugekommenen Unterprogramm *Komponentenfehler*, welches in Abbildung A.7 dargestellt ist, werden die einzelnen Komponentenfehler in Form von Instanzen der Klasse *Defekt* in den *ErstelleKomponentenfehler*-Regeln festgelegt, und in der weiteren Regel *OrdneFunktionsfehlerZu* werden die Fehlerbaumobjekte dem Funktionsfehler zugeordnet, also im Beispiel die Instanzen der Klasse *Defekt*, die nach dem erweiterten Klassendiagramm Fehlerbaumobjekte sind.

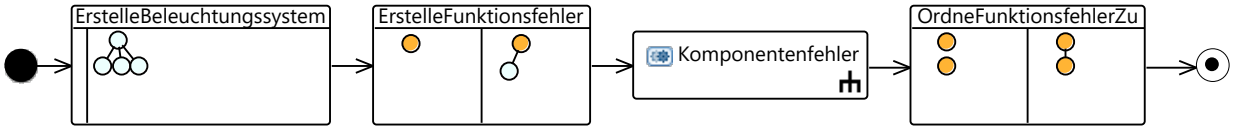

Abbildung A.6: Aktivitätsdiagramm zur Erstellung des Beleuchtungssystems mit Zuverlässigkeitsanalyseerweiterung

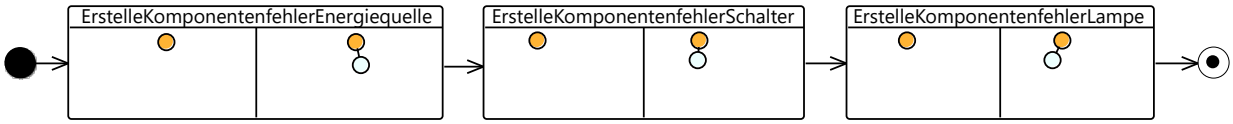

Abbildung A.7: Aktivitätsdiagramm zur Erstellung der Komponentenfehler

Die Modellbildung des Beleuchtungssystems hinsichtlich der Fehlerbaumanalyse beinhaltet bereits die ersten Schritte einer Fehlerbaumanalyse, wie sie in Kapitel 2.1.1 beschrieben wird. Es wurden im Rahmen einer Systemanalyse die Ausfallmöglichkeiten der Komponenten festgelegt, deren kausalen Verknüpfungen untereinander modelliert, sowie ein unerwünschtes Ereignis definiert, das untersucht werden soll.

Basisereignisse Instanzen der Klasse *Defekt*, die von der Klasse *FaultTreeObject* erbt und von der Klasse *KomponentenBeleuchtungssystem* assoziiert wird, welche die Komponenten des Beleuchtungssystems darstellt, sind nach der gezeigten Modellierung Fehlerbaum(basis)ereignisse.

Verknüpfungen Die kausal seriellen Verknüpfungen sind durch die Verknüpfung der Komponentenfehler zum Funktionsfehler durch die Vorgabe einer seriellen Handhabung über die Eigenschaft *parallel* (gesetzt auf *false*) im Modell vorgegeben.[2]

Wurzelereignis Als Wurzelereignis ist der in der Regel *ErstelleFunktionsfehler* erstellte Funktionsfehler, der aus der Klasse *KeineBeleuchtungNachSchalterbetaetigung* instanziiert wird, festgelegt.

A.2 Fehlerbaumanalyse

Die Zuverlässigkeitsberechnung und Fehlerbaumanalyse des semantisch annotierten Modells des Beleuchtungssystems wird nun automatisch über einen Pluginaufruf ausgeführt. Hierbei sei nochmal erwähnt, dass die Fehlerbaumanalyse hier dem Namen entsprechend, aber nicht der Definition folgend, siehe Kapitel 2 und 3, angewendet wird. Die ersten Schritte einer Fehlerbaumanalyse per Definition sind bereits in der Modellierung geschehen und in diesem Abschnitt wird auf die tatsächliche Erstellung und Analyse eines zugehörigen Fehlerbaumes eingegangen.
Der automatisch generierte Fehlerbaum zum Wurzelereignis *KeineBeleuchtungNachSchalterbetaetigung*, das die Nichterfüllung der geforderten Funktionalität des Beleuchtungssystems darstellt, ist in Abbildung A.8 dargestellt. Anhand dieses Fehlerbaumes wird nun sowohl eine qualitative als auch eine quantitative Fehlerbaumanalyse durchgeführt.

Qualitative Fehlerbaumanalyse Die qualitative Fehlerbaumanalyse des Fehlerbaumes erfolgt über die Ausgabe der minimalen Schnitte des Fehlerbaumes. Im betrachteten Beleuchtungssystem bestehen die minimalen Schnitte aus den Ereignissen *EnergiequelleDefekt*, *SchalterDefekt* und *LampeDefekt*. Die aufgezeigten minimalen Schnitte beinhalten jeweils ein Ereignis, wodurch sie als Einzelfehler (single point failure) bezeichnet werden. Auf Grundlage dieser Ergebnisse kann beispielsweise, bei Vorgabe des Ausschlusses von Einzelfehlern, das System redundant so erweitert werden, dass keine Einzelfehler mehr zum Auftreten des Wurzelereignisses führen.

2 Siehe dazu auch Kapitel 3.2 Implementierung der automatischen Fehlerbaumanalyse.

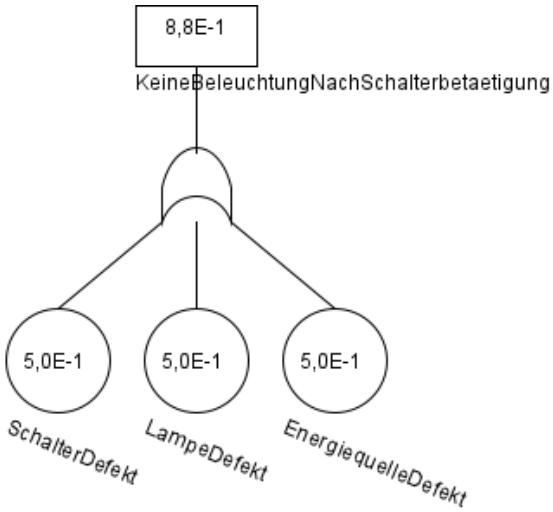

Abbildung A.8: Generierter Fehlerbaum des Beleuchtungssystem

Quantitative Fehlerbaumanalyse Die Quantitative Fehlerbaumanalyse ermittelt, aus den Auftretenswahrscheinlichkeiten der Ereignisse auf Grundlage der in Kapitel 2.1.1 beschriebenen Formeln, die Auftretenswahrscheinlichkeit des Wurzelereignisses. Diese Auftretenswahrscheinlichkeit ist für das Wurzelereignis *KeineBeleuchtungNachSchalterbetaetigung* $F \approx 8,8 * 10^{-1}$, wobei die Einzelausfallwahrscheinlichkeiten der Komponenten auf Grund der Instanzen der Klasse *Defekt*, die dem Systemfehler zugeordnet sind, in den Regeln des Unterprogramms *Komponentenfehler*, auf jeweils $F = 5,0 * 10^{-1}$ gesetzt wurden.[3] Unter Zuhilfenahme dieses Wertes kann das System bewertet und nachfolgend, hinsichtlich einer vorgegebenen Ausfallwahrscheinlichkeit das Wurzelereignis betreffend, automatisch angepasst und den Zuverlässigkeitswert betreffend verbessert werden. Im Abschnitt A.4 wird eine Verbesserung der Zuverlässigkeit des Systems durch heuristische Variation auf Grundlage der Analyseergebnisse für das Beleuchtungssystem erläutert.

A.3 Fehlermöglichkeits- und -einflussanalyse

Neben der Fehlerbaumanalyse wird anhand dieses Beispiels zusätzlich die Kombination der automatischen Fehlerbaumanalyse mit der Fehlermöglichkeits- und -einflussanalyse demonstriert. Dafür wird das Klassendiagramm des Beleuchtungssystems mit dem Klassendiagramm der Fehlermöglichkeits- und -einflussanalyse aus Kapitel 3.3.1 verknüpft. Diese Verknüpfung zeigt sich im erweiterten Klassendiagramm des Beleuchtungssystems

3 Der gesetzte Wert ist absichtlich nicht auf einen realistischen Wert gesetzt, um plakativere Werte in diesem akademischen Beispiel zu haben.

in Abbildung A.9 durch die Einbindung der Klasse *Function*, die die Klasse *Beleuchtungssystem* über eine Assoziation verknüpft. Diese Verknüpfung mit der Bezeichnung *beleuchtung* erfolgt über die Klasse *Beleuchtung* eine für das Beispiel passende, konkretere Ausprägung der Klasse *Function*. Einfacher ausgedrückt: Ein *Beleuchtungssystem* kann eine Funktion *Beleuchtung* haben.[4]

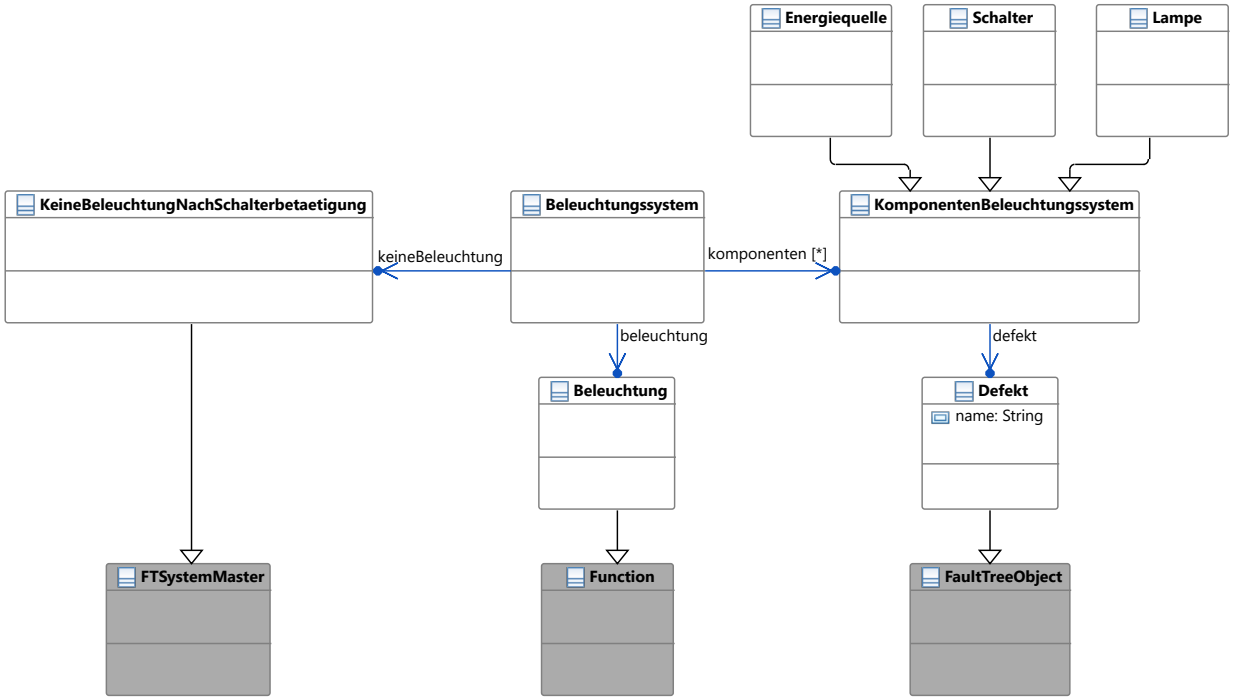

Abbildung A.9: Klassendiagramm des Beleuchtungssystems mit integrierten FMEA-Klassen[5]

Nachdem über die Erweiterung des Klassendiagramms die notwendigen Vokabeln definiert wurden, wurden die für die automatische Fehlermöglichkeits- und -einflussanalyse nötigen Regeln erstellt und in den Programmablauf eingepflegt. Analog zu einer konventionellen FMEA beginnt das erweiterte Aktivitätsdiagramm mit dem Festlegen der Funktion des zu untersuchenden Systems (Regel *ErstelleSystemfunktion*). Das im Beispiel modellierte Beleuchtungssystem soll die Funktion *Beleuchtung* implementieren, die über das Leuchten einer Lampe realisiert wird (Regel *ErstelleLoesung*). Der nachfolgende Ablauf entspricht dem vorherigen in Abbildung A.6 dargestellten Ablauf mit einer Regelan-

4 Prinzipiell kann die Richtung der Assoziation umgedreht werden, bei entsprechender Modellanpassung, damit eine Funktion eine Lösung hat und nicht eine Lösung eine Funktion. Hier wurde, aus Implementationsgründen und dem Fokus auf das Beleuchtungssystem als Lösung für die Funktion, die Modellierung wie dargestellt umgesetzt.

5 Hier ist lediglich die Klasse *Function* aus dem FMEA Klassendiagramm direkt verknüpft sichtbar, allerdings sind dadurch die weiteren Klassen, auch durch die Verknüpfung des Zuverlässigkeitsanalyseklassendiagramms mit dem FMEA Klassendiagramm, integriert.

passung für die Komponentenerstellung (Regel *ErstelleBeleuchtungssystemkomponenten*), die nicht in der selben Regel wie die Instanz der Klasse *Beleuchtungssystem* erstellt werden, da diese in der vorherigen Regel bereits als Lösung erstellt wurde und hier nur gesucht wird. Darauf folgend soll von der umgesetzten Funktion ausgehend eine FMEA erstellt werden (Regel *ErstelleFMEA*). Für diese FMEA wird im Beispiel ein Fehlermodus als Instanz der Klasse *FailureMode* mit der Namenseigenschaft *KeineBeleuchtungNachSchalterbetaetigung* erstellt, der mit der Instanz der Klasse *FMEA* verknüpft wird. Eine Auswirkung des Fehlermodus ist, dass das System nicht nachttauglich ist. Die Konsequenz ist mit der Instanz der Klasse *Consequence* mit der Namenseigenschaft *NichtNachttauglich* modelliert (Regel *ErstelleFehlermodus*). Die Konsequenz wird abschließend mit dem Funktionsfehler in Form der Instanz der Klasse *KeineBeleuchtungNachSchalterbetaetigung* verknüpft (Regel *VerbindeKonsequenzMitSystemfehler*). Diese Erweiterung um die entsprechenden Regeln ist im - in Abbildung A.10 dargestellten - Aktivitätsdiagramm zu sehen.

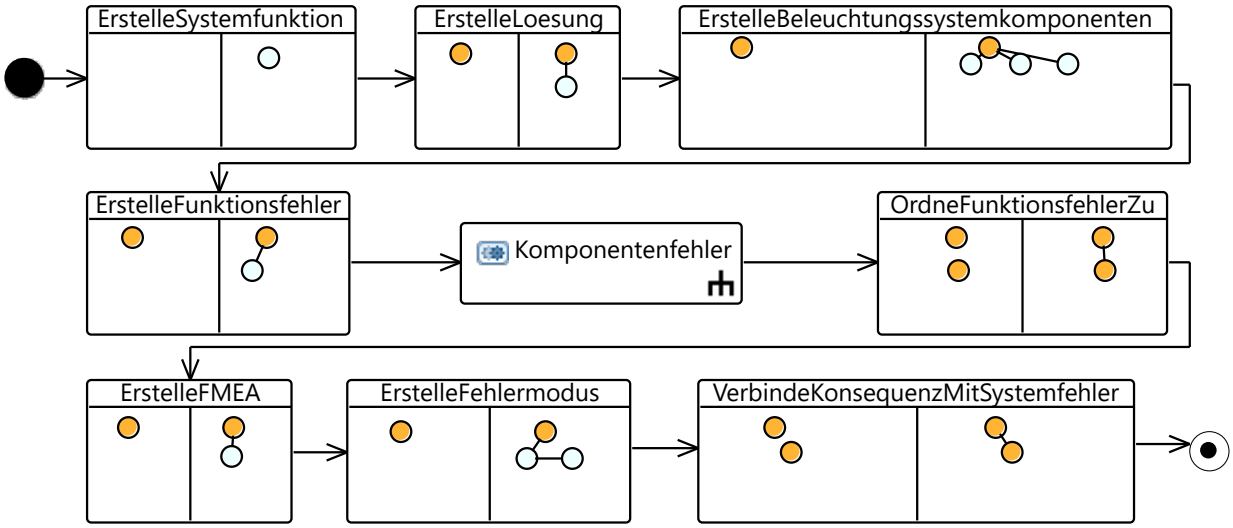

Abbildung A.10: Aktivitätsdiagramm für die Systemerstellung mit FMEA-Integration

Aufbauend auf diesem Modell wird die automatische Fehlermöglichkeits- und -einflussanalyseerstellung für den modellierten Fehlermodus ausgeführt. Dabei werden die entsprechend zugehörigen Ursachen als Instanzen der Klasse *RootCause* aus dem Klassendiagramm der FMEA erstellt und diese mit den Ereignissen der minimalen Schnitte (Instanzen der Klasse *FaultTreeObject*) verknüpft. Diese minimalen Schnitte sind wie oben beschrieben die Funktionsausfälle der Komponenten *Energiequelle*, *Schalter* und *Lampe* in Form der jeweiligen Instanz der Klasse *Defekt*.

Um eine quantitative FMEA durchzuführen wurde für den Fehlermodus die Entdeckungswahrscheinlichkeit (*detectionNumber*) mit $E = 8$ angegeben, die Bedeutung der Konsequenz für das System (*severityNumber*) mit $B = 5$ und für die Berechnungsvorschrift der Auftretenswahrscheinlichkeitszahl der Ursachen (*failureProbabilityNumber*) in der Instanz *FMEA* der gleichnamigen Klasse die Vorgaben der amerikanischen Automobilindustrie hinterlegt [5]. Die somit grafisch modellierte FMEA wurde dann über ein erstelltes Plugin exportiert, dessen Ergebnis ein reduziertes Formblatt der FMEA für das modellierte System ist, zu sehen in Abbildung A.11. In diesem Formblatt sind ebenfalls die automatisch berechneten Werte für die Risikoprioritätszahl eingetragen, die zur weiteren FMEA-Untersuchung als wichtiges Kriterium herangezogen werden.

Funktion	Fehlermodus	Auswirkung	Ursache	A	B	E	RPZ
Beleuchtung	KeineBeleuchtungNachSchalterbetaetigung	NichtNachttauglich	schalter0RC	4	5	8	160
Beleuchtung	KeineBeleuchtungNachSchalterbetaetigung	NichtNachttauglich	energiequelle0RC	4	5	8	160
Beleuchtung	KeineBeleuchtungNachSchalterbetaetigung	NichtNachttauglich	lampe0RC	4	5	8	160

Abbildung A.11: Generiertes Formblatt der FMEA des Beleuchtungssystems (A: Auftretenswahrscheinlichkeit; B: Bedeutung für das System; E: Entdeckungswahrscheinlichkeit; RPZ: Risikoprioritätszahl)[6]

A.4 Topologievariation

Die letzten Schritte einer Fehlerbaumanalyse beinhalten, neben einer Dokumentation der Ergebnisse, Vorschläge bzw. Maßnahmen zur Verbesserung der Zuverlässigkeit des zu untersuchenden Systems. Die Durchführung entsprechender Maßnahmen führt zu Modelländerungen, die Einfluss auf die Zuverlässigkeit des Systems haben. Diese Modelländerungen können im Rahmen einer gezielten Variation hinsichtlich verbesserter Zuverlässigkeit automatisch ausgeführt und anschließend analysiert werden.

Im Folgenden wird anhand des Beleuchtungssystembeispiels eine solche automatische Systemanpassung über eine Heuristik vorgestellt. Diese demonstriert die dynamische Verwendung der automatischen Zuverlässigkeitsberechnung und Fehlerbaumanalyse. Das Beleuchtungssystem wird hierfür hinsichtlich Parametrik und Topologie automatisch angepasst, um eine vorgegebene Zuverlässigkeit gegenüber einem Funktionsausfall des Systems zu gewährleisten.

Anmerkung: Die Zahlenwerte für die Ausfallwahrscheinlichkeit bzw. Zuverlässigkeit der Ereignisse sind zur besseren Veranschaulichung realitätsfern hoch bzw. niedrig gesetzt.

6 Das RC am Ende der Ursachenbezeichnung steht für *RootCause*, der Namensteil *Defekt* für die Ursachen wurde hier gestrichen.

Modellierung

Für die automatische Systemanpassung wurde das Aktivitätsdiagramms angepasst, wie in Abbildung A.12 zu sehen ist. Abweichend vom zuvor beschriebenen Ablauf für die FMEA sind die abschließenden Regeln entfernt worden. Auch die Regel *OrdneFunktionsfehlerZu* aus dem ursprünglichen Ablauf mit integrierter Zuverlässigkeitsanalyse wurde entfernt, findet sich aber in geänderter Form im Unterprogramm *Systemvariation* wieder. Neu hinzugekommen ist die Regel *ErstelleAnforderung*, in der der Parameter *minSystemReliability*, als minimale Systemzuverlässigkeit in Bezug auf den Funktionsfehler, für die Instanz der Klasse *KeineBeleuchtungNachSchalterbetaetigung* gesetzt wird. Die Anpassung des Systems erfolgt im Unterprogramm *Systemvariation*.

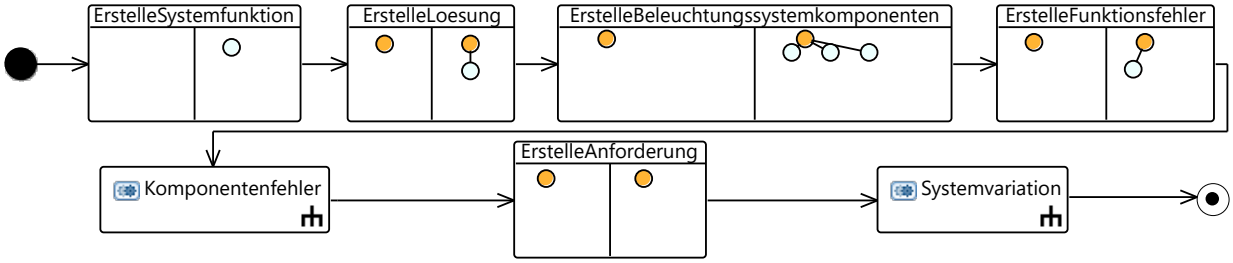

Abbildung A.12: Aktivitätsdiagramm Beleuchtungssystem mit Topologievariation

Im Unterprogramm *Systemvariation*, abgebildet in Abbildung A.13, werden zunächst funktionale Objekte als Instanzen der Klasse *Defekt* für die einzelnen Komponenten mit zugehöriger Instanz der Klasse *FTSystemMaster* erstellt, die eine abstrakten Zusammenhang der Komponenten beschreiben und vom Systemfehler der Instanz der Klasse *KeineBeleuchtungNachSchalterbetaetigung* assoziiert werden (Unterprogramm *ErstellungFunktionalerObjekte*). Diesen funktionalen Objekten werden die Komponentenfehler der zugehörigen Komponenten zugeordnet, analog zur bisherigen Zuordnung nur in einer abstrakten (funktionalen) Zwischenebene (Unterprogramm *ZuordnungFunktionalerObjekte*).[7] Die automatische Systemanpassung erfolgt im gleichnamigen Unterprogramm *Systemanpassung*, dessen Ablauf ist in Abbildung A.14 dargestellt.

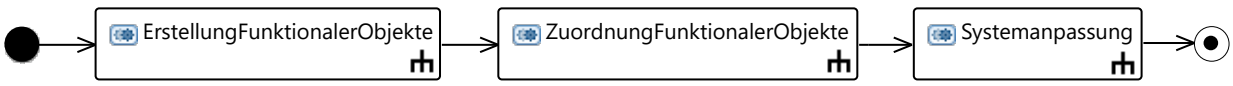

Abbildung A.13: Aktivitätsdiagramm *Systemvariation*

7 Es sind nicht die tatsächlichen Komponenten kausal miteinander verknüpft, sondern eine funktionale Komponente, die die tatsächlichen Komponenten intern - wie in einer *black box* - enthält und deren Zusammenhang untereinander beschreibt - hier kausal redundante Ausführung.

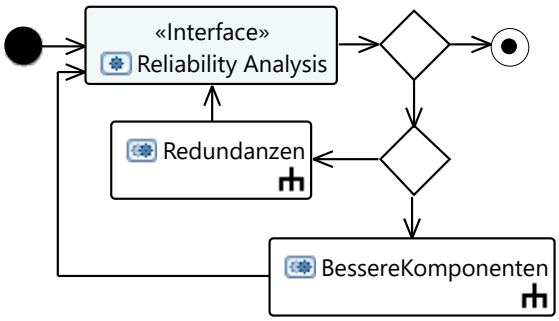

Abbildung A.14: Aktivitätsdiagramm *Systemanpassung*

Automatische Systemanpassung Für das Modell des Beleuchtungssystems wird zunächst über den Schnittstellenaufruf der Zuverlässigkeitsanalyse (*Reliability Analysis*) die Zuverlässigkeit gegenüber dem Funktionsfehler *KeineBeleuchtungNachSchalterbetaetigung* berechnet. Im folgenden Entscheidungsknoten wird dahingehend überprüft, ob die geforderte Zuverlässigkeit durch das System gegenüber dem Funktionsfehler gegeben ist. Ist dies nicht der Fall, wird im anschließenden Entscheidungsknoten überprüft, ob das System durch zuverlässigere Komponenten hinsichtlich der Zuverlässigkeit weiter verbessert werden kann oder ob bereits alle Komponenten die höchstmögliche Zuverlässigkeit, der verwendbaren Komponenten aufweisen, und redundante Komponenten integriert werden müssen, um die Zuverlässigkeit zu erhöhen. Sofern zuverlässigere Komponenten eingebaut werden können wird das Unterprogramm *BessereKomponenten* aufgerufen, zur Erhöhung der Zuverlässigkeit durch Redundanz das Unterprogramm *Redundanzen*.

In Abbildung A.15 ist das Unterprogramm *BessereKomponenten* dargestellt. Die Regeln in diesem Aktivitätsdiagramm ändern die Zuverlässigkeit einer Komponente (*Energiequelle*, *Schalter* oder *Lampe*) durch Änderung der Ausfallwahrscheinlichkeit des zugeordneten Komponentenfehlers, sofern diese einen in den Regeln festgelegten Ausfallwahrscheinlichkeitswert überschreiten. Konkret für die Regel *VerwendeKomponente07*, zu sehen in Abbildung A.16, bedeutet dies, dass für eine Instanz der Klasse *Defekt*, die eine Aus-

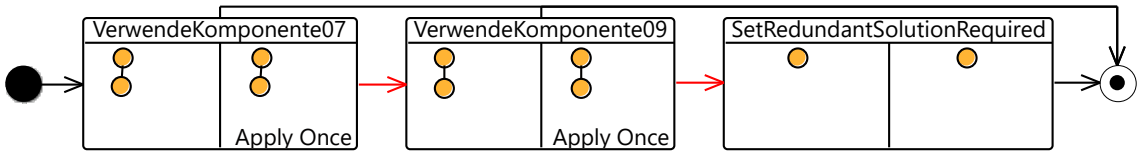

Abbildung A.15: Aktivitätsdiagramm *BessereKomponenten* für Parametrikänderungen[8]

8 Der Text *Apply Once* in den Regeln bedeutet, dass die Regel(-Transformation) nur einmal auf das Modell (Entwurfsgraph) angewendet wird und nicht auf jedes gefundene Objekt.
Rote/hellgraue Pfeile sind Abläufe, welche bei Nichtausführung der Regeln ablaufen.

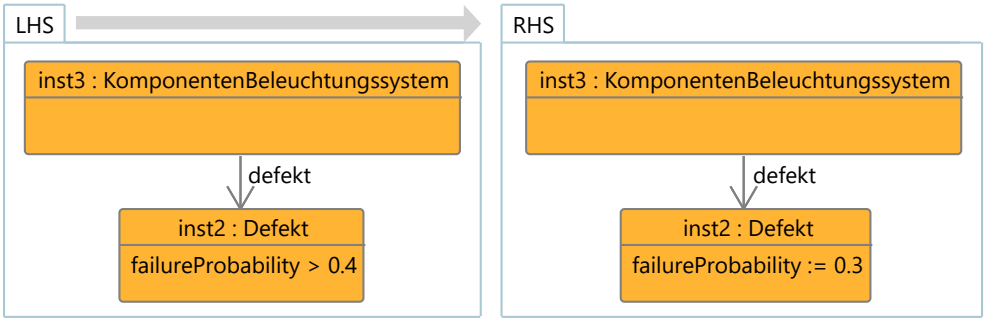

Abbildung A.16: Regel zum Komponententausch (hier: *VerwendeKomponente07*)

fallwahrscheinlichkeit von $F > 0,4$ aufweist, die Eigenschaft der Ausfallwahrscheinlichkeit auf $F = 0,3$ gesetzt wird. Sobald eine Regel des Unterprogramms erfolgreich ausgeführt werden kann, endet das Unterprogramm und die Zuverlässigkeitsanalyse des neuen Modells wird durchgeführt. Kann eine Regel durch eine nicht im Entwurfsgraphen des Modells repräsentierte Suchseite nicht ausgeführt werden, dann wird die nächste Ersetzungsregel angestoßen. In der Abbildung A.15 des Aktivitätsdiagramms ist diese Unterscheidung durch die unterschiedlich farbigen Kontrollflüsse dargestellt. Liefert eine Regel einen positiven Rückgabewert, der angibt, dass die Regel erfolgreich ausgeführt werden konnte, wird der schwarze Kontrollfluss weiterverfolgt, andernfalls der rote bzw. hellgraue Kontrollfluss.
Wird die letzte Regel des Unterprogramms erreicht, haben alle Komponentenfehler die maximal im Modell verfügbare Zuverlässigkeit erreicht. In der Regel *SetRedundantSolutionRequired* wird ein Boolescher Wert gesetzt, der angibt, dass in der nächsten Iteration eine redundante Lösung erforderlich ist, um das System hinsichtlich der Zuverlässigkeit weiter zu verbessern.

Der alternative Pfad nach dem zweiten Entscheidungsknoten führt zum Unterprogramm *Redundanzen*, welches in Abbildung A.17 zu sehen ist. Dieses Unterprogramm beginnt mit einem Unterprogramm *ResetToMostUnreliableComponents*, in dem die Ausfallwahrscheinlichkeiten der Komponentenfehler auf die Ausgangswerte zurückgesetzt werden, damit eventuell unzuverlässigere, aber den Zielwert erreichende (Parameter-)Kombinationen mit einer geänderten Topologie überprüft werden. Des Weiteren folgen für die Redundanz der einzelnen Komponenten wiederum Unterprogramme, sowie Entscheidungsknoten, die auf bereits durchgeführte Toplogieänderungen in den jeweiligen Untersystemen prüfen und diese miteinander vergleichen. Es wird die funktionale Komponente mit der minimalen Redundanz redundant erweitert, sofern es keine Unterschiede in der

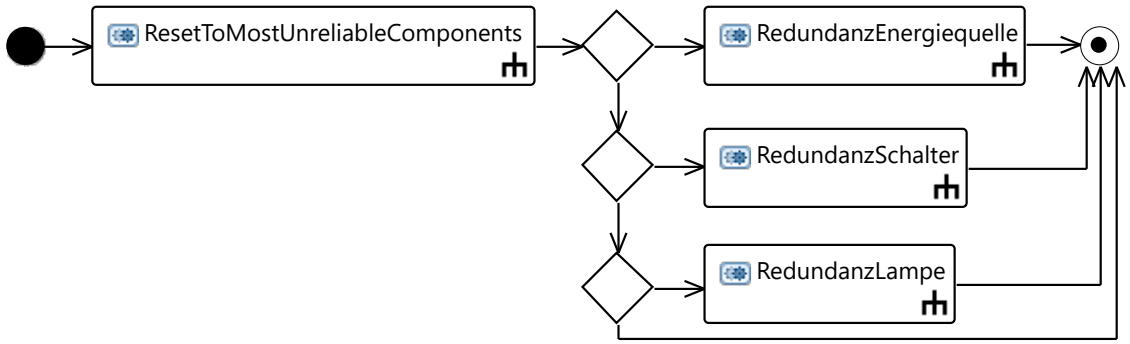

Abbildung A.17: Aktivitätsdiagramm *Redundanzen* für die Topologieanpassung

momentanen Redundanz gibt, wird die zuerst im Ablauf befindliche funktionale Komponente redundant erweitert. Exemplarisch wird auf das Unterprogramm *RedundanzSchalter* genauer eingegangen. Dieses beinhaltet lediglich die in Abbildung A.18 dargestellte Regel *ErstelleSchalterMitFunktionsfehler*, die das System um einen Schalter mit zugehörigem Fehlerereignis als Instanz der Klasse *Defekt* erweitert und diesem dem funktionalen Schalter zuordnet.

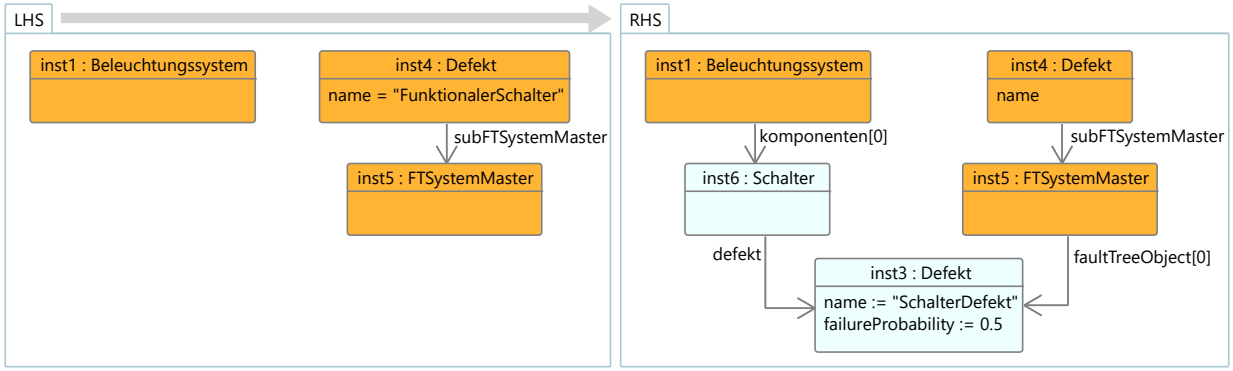

Abbildung A.18: Regel zum redundanten Schalter (*ErstelleSchalterMitFunktionsfehler*)

Fehlerbäume

Die Ausführung des Programms des erweiterten Beleuchtungssystems führt je nach Anforderung an das System zu unterschiedlichen Parametriken und Topologien des Beleuchtungssystems. Aus diesen verschiedenen Modellen können die zugehörigen Fehlerbäume, entweder zur Laufzeit oder nach Programmende erstellt und analysiert werden.

In Abbildung A.19 ist ein Fehlerbaum für ein Beleuchtungssystem zu sehen, das lediglich Parametrikänderungen enthält. Die Anforderungen an dieses System waren eine geforderte Mindestzuverlässigkeit von $R = 0,7$ bzw. Ausfallwahrscheinlichkeit von $F = 0,3$. Dafür wurden die Komponentenfehler des Systems mit einer Ausgangszuverlässigkeit

von je $R = 0,5$ mit zuverlässigeren Komponentenfehlern ausgetauscht, bis die Anforderung erfüllt wurde. Diese Erfüllung der Anforderung wurde mit Komponentenfehlern erreicht, die jeweils eine Zuverlässigkeit von $R = 0,9$ aufweisen.

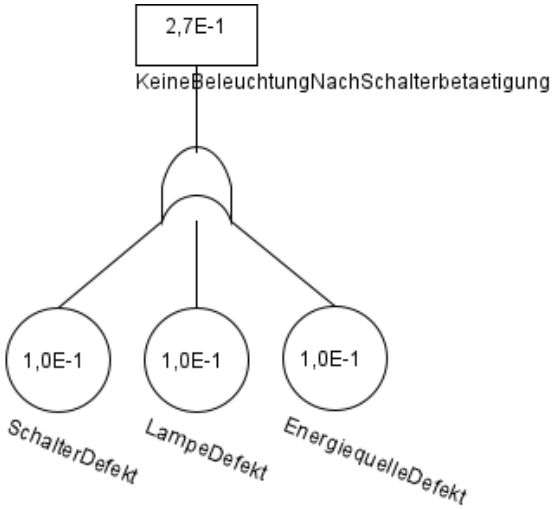

Abbildung A.19: Fehlerbaum mit Parametrikanpassung

Ein weiterer Fehlerbaum, zu sehen in Abbildung A.20, gehört zu einem Beleuchtungssystem, dessen Topologie durch redundante Komponenten verändert wurde. Die Anforderungen an dieses System waren eine Mindestzuverlässigkeit von $R = 0,25$ unter Verwendung von Komponentenfehlern der Zuverlässigkeit $R = 0,5$. Das System wurde dabei um eine Instanz der Klasse *Energiequelle* und eine Instanz der Klasse *Schalter* erweitert, um die geforderten Bedingungen zu erfüllen, was sich im automatisch erstellten Fehlerbaum widerspiegelt. In diesem Fehlerbaum sind auch die oben beschriebenen, entsprechenden, funktionalen Objekte mit dargestellt.

Dieses Beispiel zeigt anhand eines relativ simplen Systems die Integration der automatischen Fehlerbaumanalyse in einen modellbasierten Entwurfsprozess. Diese Einbindung in frühe Phasen des Entwurfsprozesses wird durch die Erweiterung um die Fehlermöglichkeits- und -einflussanalyse zusätzlich in Verbindung mit einer weit verbreiteten Risikoanalysemethode gestellt. Die Fehlermöglichkeits- und -einflussanalyse baut dabei auf den Ergebnissen der automatischen Fehlerbaumanalyse auf und verwendet diese für die eigene Analyse. In diesem Beispiel wird deutlich, dass umfangreiches Wissen über ein System modelliert werden muss, um eine zufriedenstellende Fehlerbaumanalyse und gegebenenfalls Fehlermöglichkeits- und -einflussanalyse durchführen zu können.

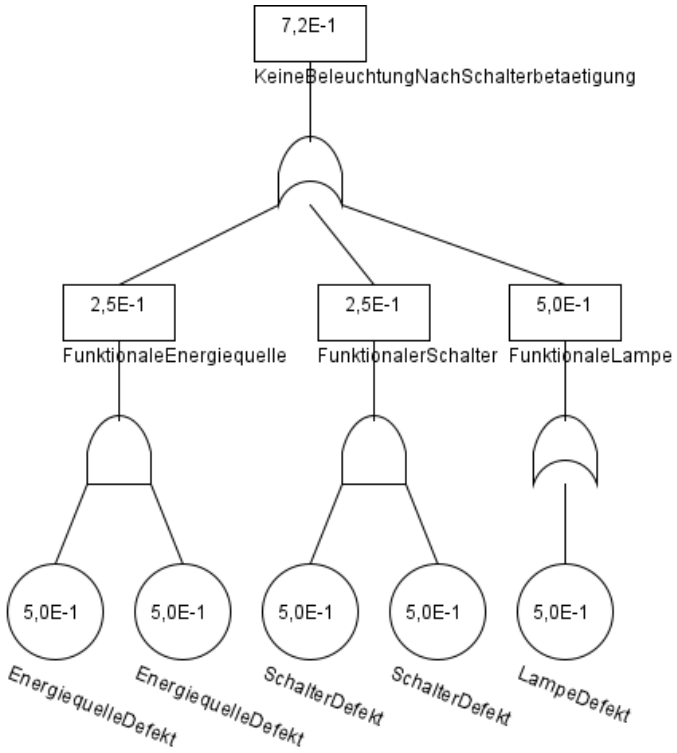

Abbildung A.20: Fehlerbaum mit Redundanz

Zusammengefasst wurde anhand des Beispieles des Beleuchtungssystems zunächst die prinzipielle Modellierung und automatische Fehlerbaumerstellung und -analyse gezeigt. Des Weiteren wurde ein Zusammenspiel zwischen der Fehlerbaumanalyse und der Fehlermöglichkeits- und -einflussanalyse vorgestellt, wobei die FMEA die Ergebnisse der FTA verwendet, um eine Analyse durchzuführen. Abschließend demonstriert dieses Beispiel den dynamischen Einsatz der Zuverlässigkeitsanalyse und Fehlerbaumerstellung an einem sich veränderndem System, sowie eine Anwendung zur Verbesserung des Systems hinsichtlich der Zuverlässigkeit durch automatische Systemanpassung über den Einsatz von Entwurfssprachen, wodurch die komplette Methode der Fehlerbaumanalyse automatisiert umgesetzt wurde. Vor allem die dynamische Fehlerbaumerstellung bietet großes Potential zur Zeitersparnis in frühen Phasen eines Entwurfsprozesses, die häufig sich ändernde Systemparametriken und -topologien beinhalten. Eben diese automatische und dynamische Fehlerbaumerstellung ist im Rahmen einer Optimierung hinsichtlich Parametrik oder Topologie erstrebenswert, um den Aufwand einer Änderung in Bezug auf begleitende Modelle nicht als hemmenden Faktor zuzulassen.

Anhang B

Fehlerbäume

Im Folgenden sind weitere automatisch erzeugte Fehlerbäume zu den Anwendungsbeispielen aus Kapitel 4 dargestellt. Beim Schiffmotor bzw. Schiffantriebssystem werden Fehlerbäume für verschiedene Betriebsdauern des Systems gezeigt, sowie Ausgabeergebnisse einer angewandten automatischen Fehlermöglichkeits- und -einflussanalyse vorgestellt. Für die Satellitenantriebssysteme sind verschiedene Fehlerbäume je nach Zuverlässigkeitanforderung sowie für verschiedene Satellitenantriebssystemarten abgebildet.

B.1 Schiffsmotor

Für den Schiffsmotor sind in den Abbildungen B.1, B.2 und B.3 Fehlerbäume dargestellt, die Zuverlässigkeitswerte beinhalten, die für die vorgegebene Betriebsdauer ermittelt wurden. Eine formelmäßige Berechnung der Zuverlässigkeitswerte in Abhängigkeit der Betriebsdauer ist, neben fixen Werten, eine weitere Möglichkeit, die bei entsprechend ermitteltem Verhalten, in der Anwendung hilfreich ist.

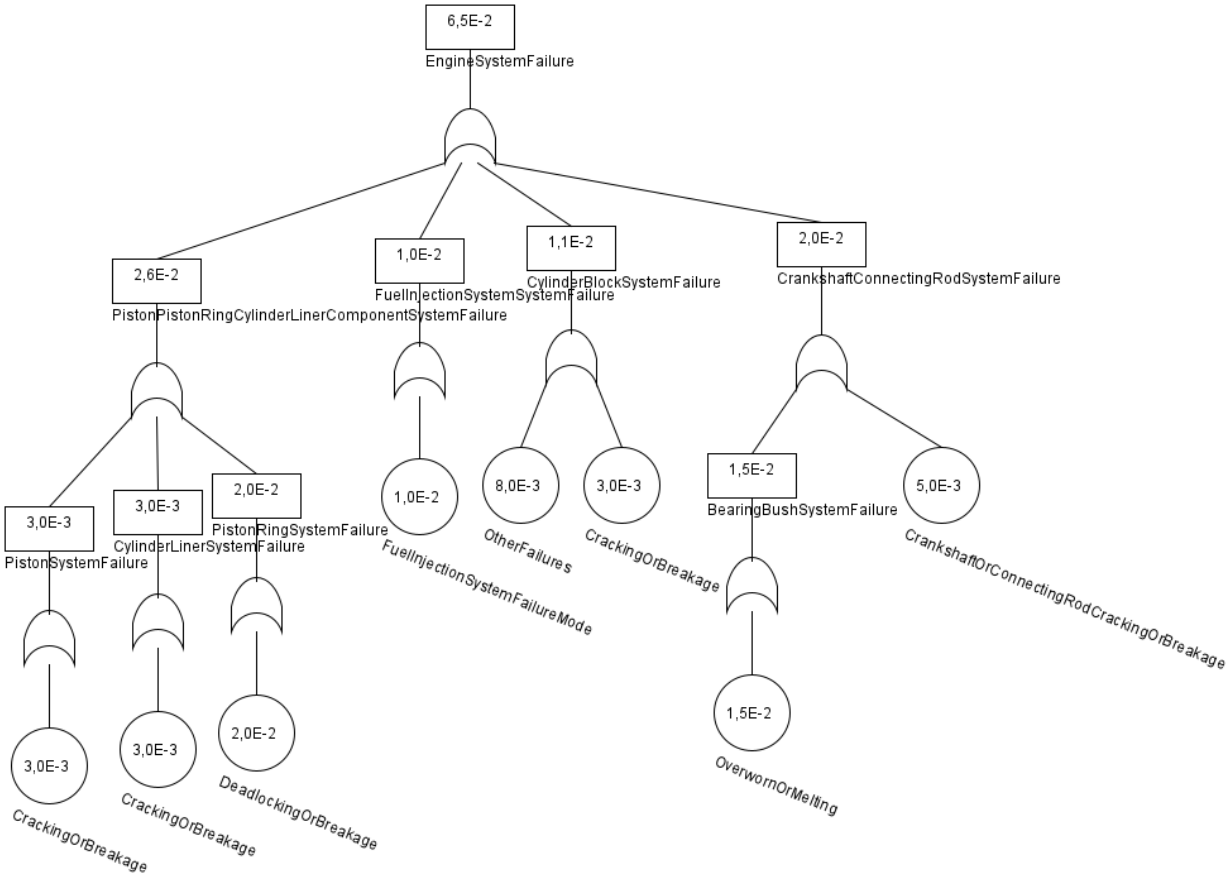

Abbildung B.1: Fehlerbaum Schiffsmotor 500 Stunden

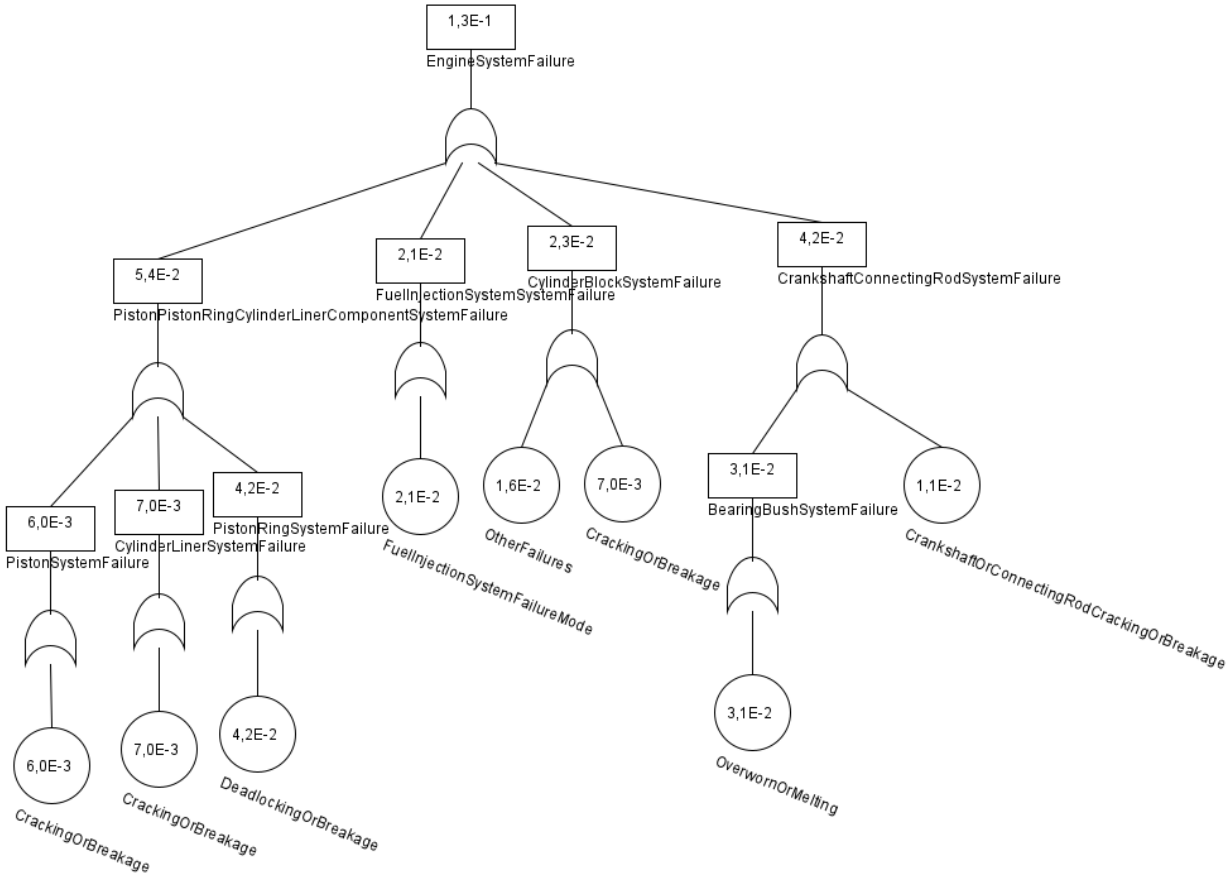

Abbildung B.2: Fehlerbaum Schiffsmotor 1000 Stunden

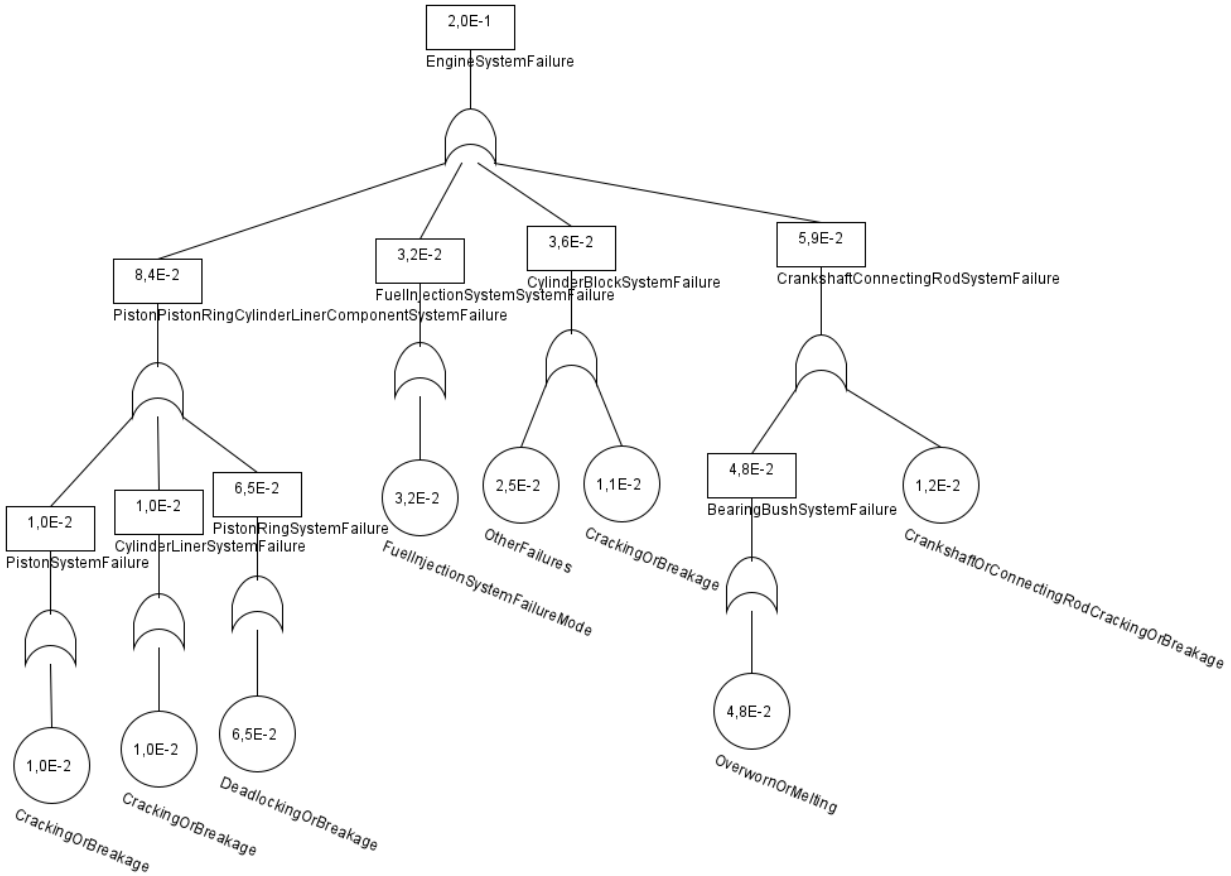

Abbildung B.3: Fehlerbaum Schiffsmotor 1500 Stunden

B.2 Schiffsantriebssystem

Neben einem rein zur Expansionsanschauung dienenden Fehlerbaum des Schiffsantriebssystems ohne Redundanz (und Zahlenwerte) in den Untersystemen in Abbildung B.4, ist noch ein generiertes Formblatt der FMEA aus den Modelldaten des Schiffsantriebssystems abgebildet in den Abbildungen B.5 und B.6. Die Zahlenwerte in den einzelnen Zeilen und Spalten entsprechen sich jeweils, da hier eine allgemeine Regel im Rahmen der Modellerweiterung verwendet wurde, die jedem Ereignis die selbe Konsequenz mit der entsprechenden Bedeutung sowie eine einheitliche Entdeckungswahrscheinlichkeit zuordnet. Die Werte der Ausfallwahrscheinlichkeit sind nicht gesetzt und dementsprechend wird die Auftretenswahrscheinlichkeit durch eine Zuordnung in die unterste Kategorie, sprich keinem Ausfall, auf den Wert 1 gesetzt. Die Tabelle der einzelnen FMEA-Einträge beinhaltet 69 Einträge für das Schiffsantriebssystemmodell ohne Redundanz.

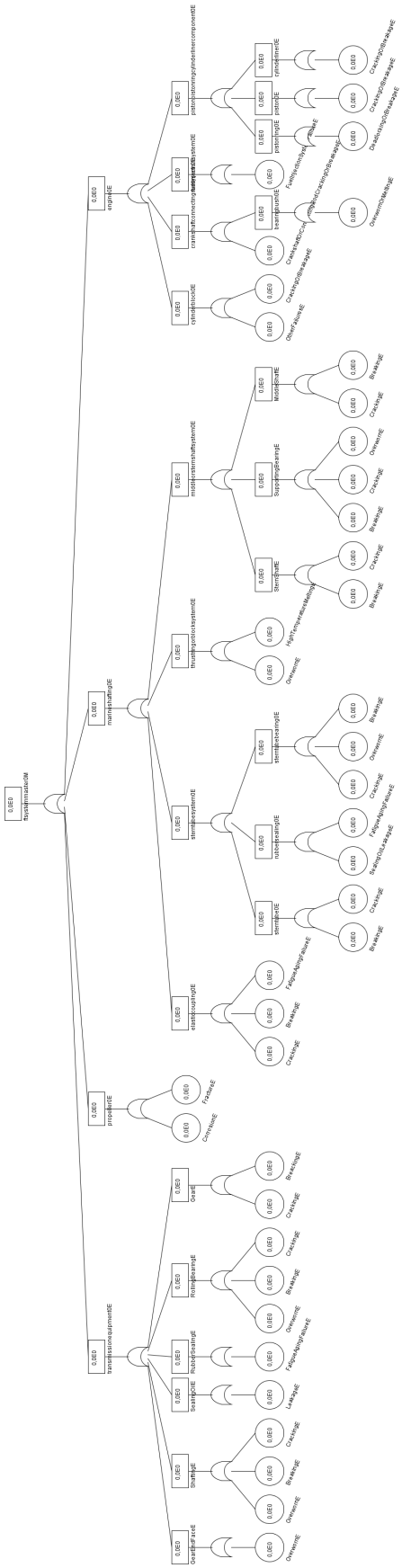

Abbildung B.4: Fehlerbaum Schiffsantriebssystem ohne Redundanz

	A	B	C	D	E	F	G	H
1	Funktion	Fehlermodus	Auswirkung	Ursache	Auftretenswahrscheinlichkeit	Bedeutung für das System	Entdeckungswahrscheinlichkeit	RPZ
2	System function	failuremode16	consequence16	rubbersealing0RC	1.0	4	8	32
3	System function	failuremode15	consequence15	FuelInjectionSystemFailureRC	1.0	4	8	32
4	System function	failuremode4	consequence4	elasticcoupling0RC	1.0	4	8	32
5	System function	failuremode19	consequence19	FatigueAgingFailureRC	1.0	4	8	32
6	System function	failuremode28	consequence28	RollingBearingRC	1.0	4	8	32
7	System function	failuremode28	consequence28	RubberSealingRC	1.0	4	8	32
8	System function	failuremode16	consequence16	sterntube0RC	1.0	4	8	32
9	System function	failuremode14	consequence14	BreakingRC	1.0	4	8	32
10	System function	failuremode6	consequence6	piston0RC	1.0	4	8	32
11	System function	failuremode28	consequence28	GearRC	1.0	4	8	32
12	System function	failuremode6	consequence6	pistonring0RC	1.0	4	8	32
13	System function	failuremode12	consequence12	pistonpistonringcylinderlinercomponent0RC	1.0	4	8	32
14	System function	failuremode23	consequence23	CrackingRC	1.0	4	8	32
15	System function	failuremode23	consequence23	OverwornRC	1.0	4	8	32
16	System function	failuremode4	consequence4	sterntubesystem0RC	1.0	4	8	32
17	System function	failuremode13	consequence13	BreakingRC	1.0	4	8	32
18	System function	failuremode26	consequence26	transmissionequipment0RC	1.0	4	8	32
19	System function	failuremode2	consequence2	CrackingRC	1.0	4	8	32
20	System function	failuremode17	consequence17	SealingOilLeakageRC	1.0	4	8	32
21	System function	failuremode16	consequence16	sterntubebearing0RC	1.0	4	8	32
22	System function	failuremode5	consequence5	OverwornRC	1.0	4	8	32
23	System function	failuremode20	consequence20	CorrosionRC	1.0	4	8	32
24	System function	failuremode11	consequence11	CrackingOrBreakageRC	1.0	4	8	32
25	System function	failuremode8	consequence8	OverwornRC	1.0	4	8	32
26	System function	failuremode4	consequence4	thrustringorblocksystem0RC	1.0	4	8	32
27	System function	failuremode23	consequence23	BreakingRC	1.0	4	8	32
28	System function	failuremode21	consequence21	DeadlockingOrBreakageRC	1.0	4	8	32
29	System function	failuremode12	consequence12	crankshaftconnectingrodsystem0RC	1.0	4	8	32
30	System function	failuremode20	consequence20	FractureRC	1.0	4	8	32
31	System function	failuremode27	consequence27	SternShaftRC	1.0	4	8	32
32	System function	failuremode12	consequence12	fuelinjectionsystem0RC	1.0	4	8	32
33	System function	failuremode1	consequence1	BreakingRC	1.0	4	8	32
34	System function	failuremode27	consequence27	MiddleShaftRC	1.0	4	8	32
35	System function	failuremode12	consequence12	cylinderblock0RC	1.0	4	8	32
36	System function	failuremode5	consequence5	CrackingRC	1.0	4	8	32
37	System function	failuremode0	consequence0	CrackingRC	1.0	4	8	32
38	System function	failuremode24	consequence24	bearingbush0RC	1.0	4	8	32
39	System function	failuremode28	consequence28	SealingOilRC	1.0	4	8	32
40	System function	failuremode0	consequence0	BreakingRC	1.0	4	8	32
41	System function	failuremode28	consequence28	GearEndFaceRC	1.0	4	8	32

Abbildung B.5: FMEA-Formblatt 1

42	System function	failuremode13	consequence13	OverwornRC	1.0	4	8	32
43	System function	failuremode1	consequence1	CrackingRC	1.0	4	8	32
44	System function	failuremode26	consequence26	propeller0RC	1.0	4	8	32
45	System function	failuremode17	consequence17	FatigueAgingFailureRC	1.0	4	8	32
46	System function	failuremode22	consequence22	FatigueAgingFailureRC	1.0	4	8	32
47	System function	failuremode7	consequence7	HighTemperatureMeltingRC	1.0	4	8	32
48	System function	failuremode25	consequence25	OtherFailuresRC	1.0	4	8	32
49	System function	failuremode9	consequence9	CrackingRC	1.0	4	8	32
50	System function	failuremode3	consequence3	CrackingOrBreakageRC	1.0	4	8	32
51	System function	failuremode9	consequence9	BreakingRC	1.0	4	8	32
52	System function	failuremode26	consequence26	engine0RC	1.0	4	8	32
53	System function	failuremode7	consequence7	OverwornRC	1.0	4	8	32
54	System function	failuremode5	consequence5	BreakingRC	1.0	4	8	32
55	System function	failuremode14	consequence14	CrackingRC	1.0	4	8	32
56	System function	failuremode18	consequence18	OverwornOrMeltingRC	1.0	4	8	32
57	System function	failuremode22	consequence22	CrackingRC	1.0	4	8	32
58	System function	failuremode25	consequence25	CrackingOrBreakageRC	1.0	4	8	32
59	System function	failuremode4	consequence4	middleorsternshaftsystem0RC	1.0	4	8	32
60	System function	failuremode22	consequence22	BreakingRC	1.0	4	8	32
61	System function	failuremode26	consequence26	marineshafting0RC	1.0	4	8	32
62	System function	failuremode28	consequence28	ShaftingRC	1.0	4	8	32
63	System function	failuremode14	consequence14	OverwornRC	1.0	4	8	32
64	System function	failuremode27	consequence27	SupportingBearingRC	1.0	4	8	32
65	System function	failuremode6	consequence6	cylinderliner0RC	1.0	4	8	32
66	System function	failuremode13	consequence13	CrackingRC	1.0	4	8	32
67	System function	failuremode2	consequence2	BreackingRC	1.0	4	8	32
68	System function	failuremode10	consequence10	LeakageRC	1.0	4	8	32
69	System function	failuremode24	consequence24	CrankshaftOrConnectingRodCrackingOrBreakageRC	1.0	4	8	32

Abbildung B.6: FMEA-Formblatt 2

B.3 Satellitenantriebssysteme

In den folgenden Abbildungen (B.7, B.8, B.9, B.10 und B.11) sind zunächst Fehlerbäume eines Kaltgassatellitenantriebssystem zu sehen. Dabei wird die Veränderung der Parametrik und Topologie des Systems bei steigenden Anforderungen an die Ausfallwahrscheinlichkeit im Zuge der automatischen Fehlerbaumanalyse in den automatisch erzeugten Fehlerbäumen ersichtlich. Darauf folgend ist in Abbildung B.12 ein Fehlerbaum des Kaltgassatellitenantriebsystems unter Vorgabe einer geänderten Strategie, die Parameteränderungen Topologieänderungen vorzieht, zu sehen. Nachfolgend sind generierte Fehlerbäume für ein monergoles Satellitenantriebssystem mit und ohne Redundanz in den Abbildungen B.13 und B.14 dargestellt. Abschließend ist ein Fehlerbaum für ein diergoles Satellitenantriebssystem in Abbildung B.15 abgebildet.

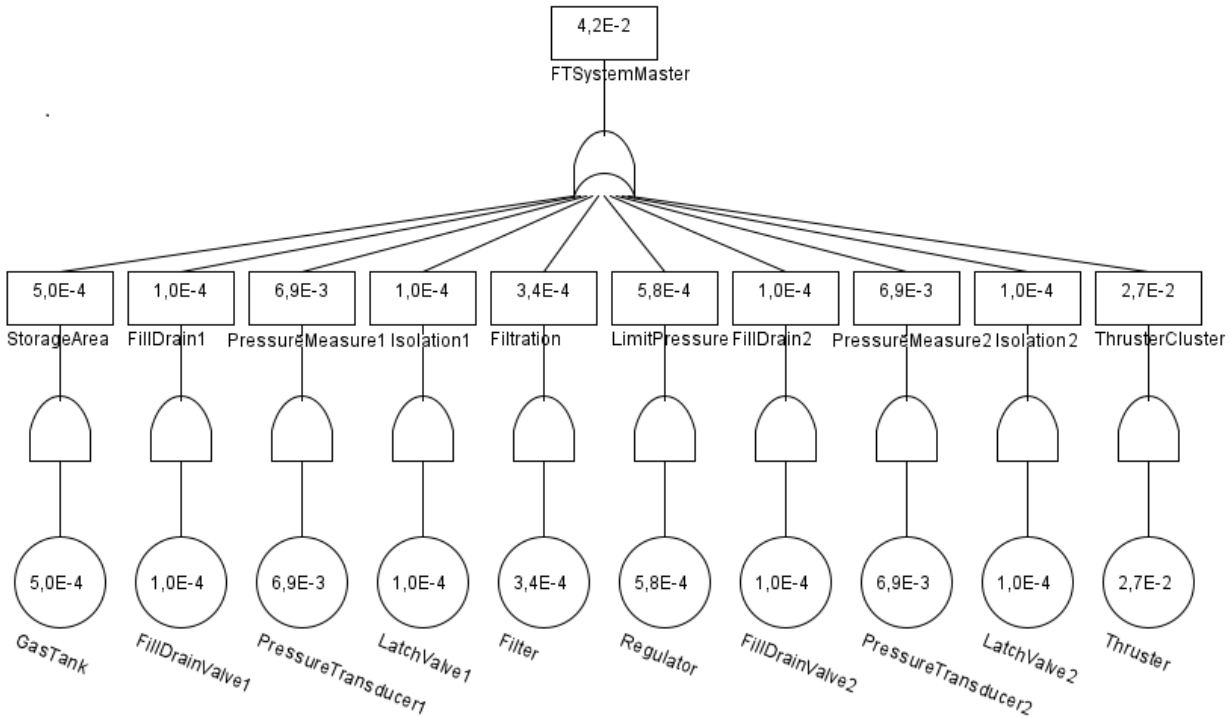

Abbildung B.7: Fehlerbaum des Kaltgassatellitenantriebssystems ohne Zuverlässigkeitsanforderung

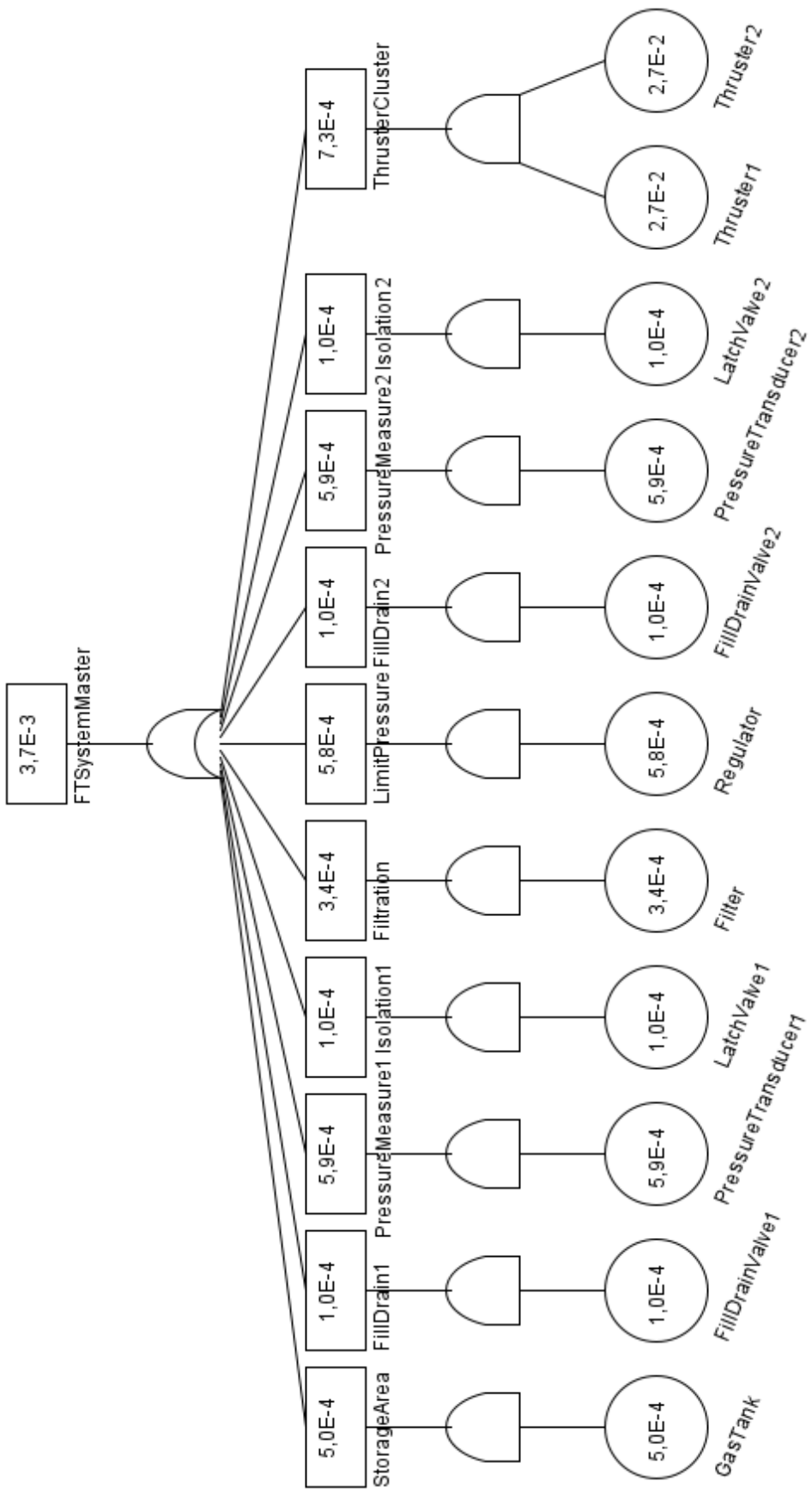

Abbildung B.8: Fehlerbaum des Kaltgassatellitenantriebssystems mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 1 \cdot 10^{-2}$

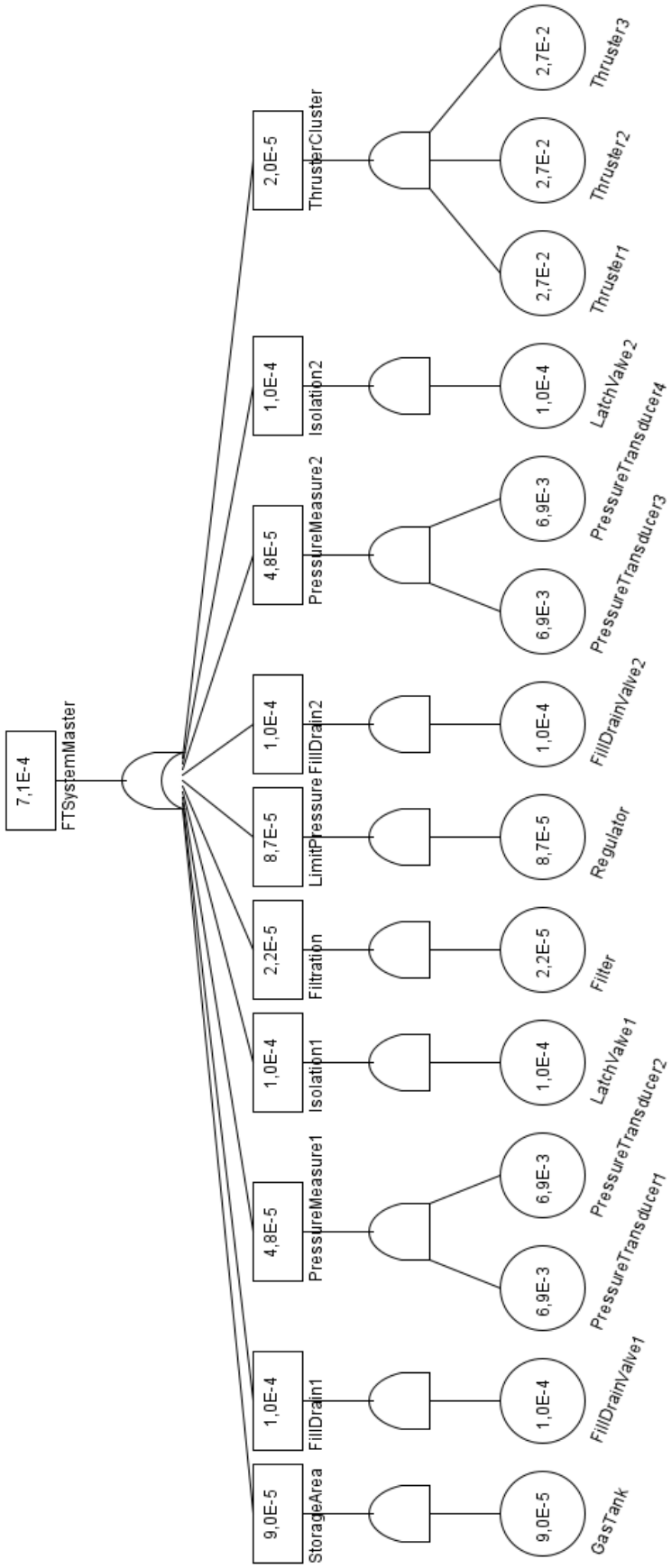

Abbildung B.9: Fehlerbaum des Kaltgassatellitenantriebssystems mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 1 \cdot 10^{-3}$

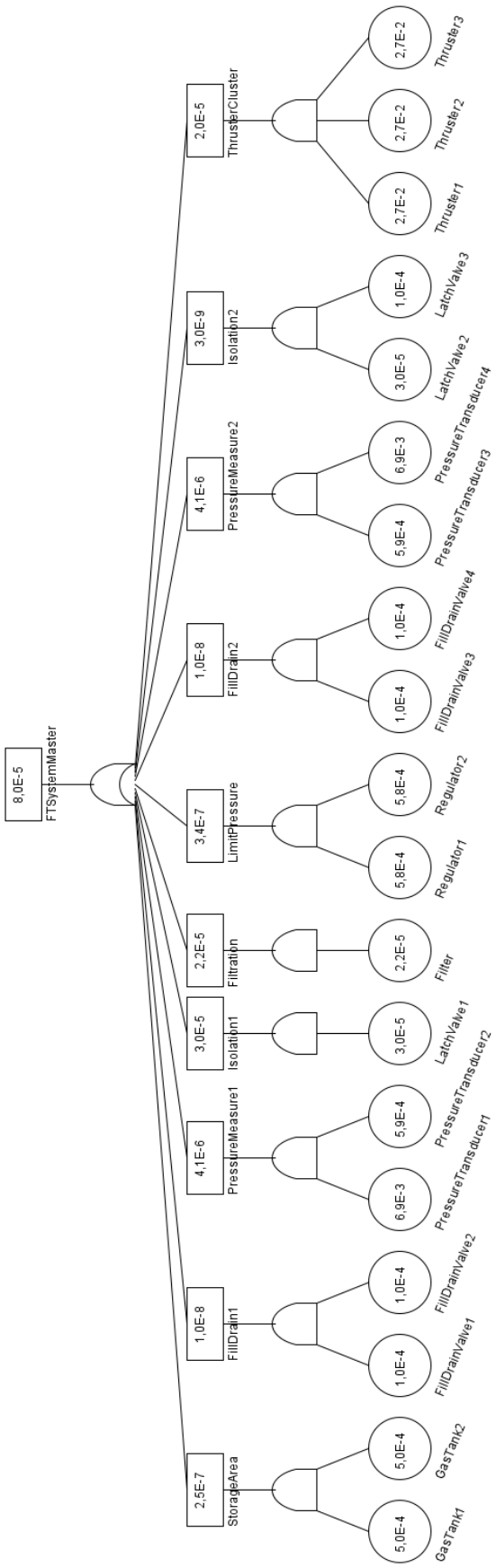

Abbildung B.10: Fehlerbaum des Kaltgassatellitenantriebssystems mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 1 \cdot 10^{-4}$

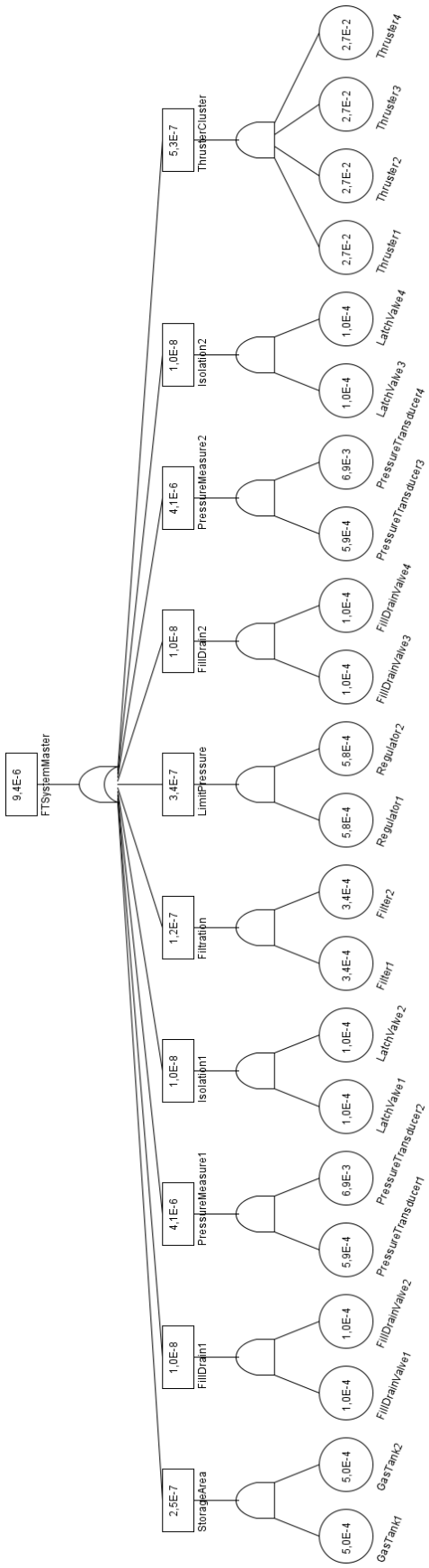

Abbildung B.11: Fehlerbaum des Kaltgassatellitenantriebssystems mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 1 \cdot 10^{-5}$

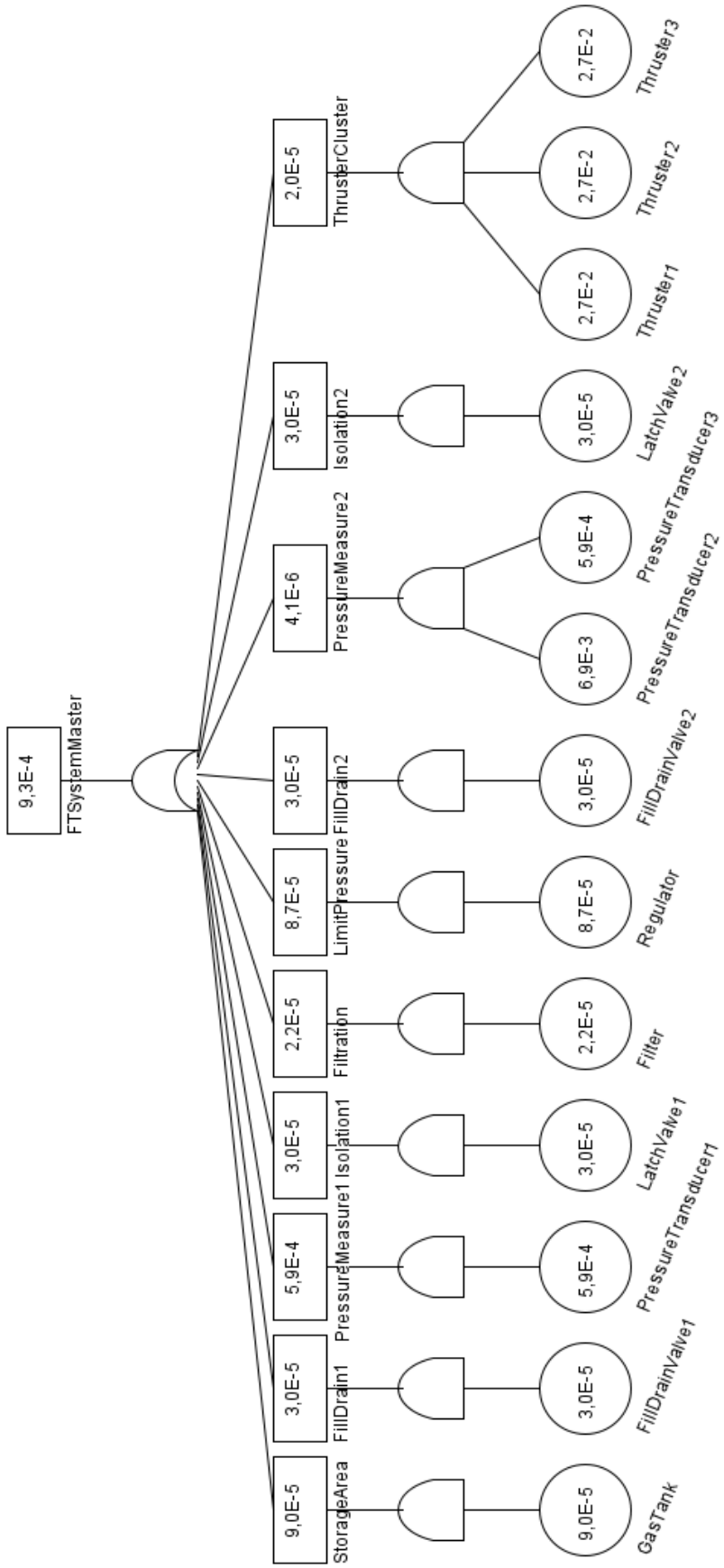

Abbildung B.12: Fehlerbaum des Kaltgassatellitenantriebssystems mit einer geforderten maximalen Ausfallwahrscheinlichkeit von $F = 1 \cdot 10^{-3}$ und Bevorzugung von Parameteränderungen

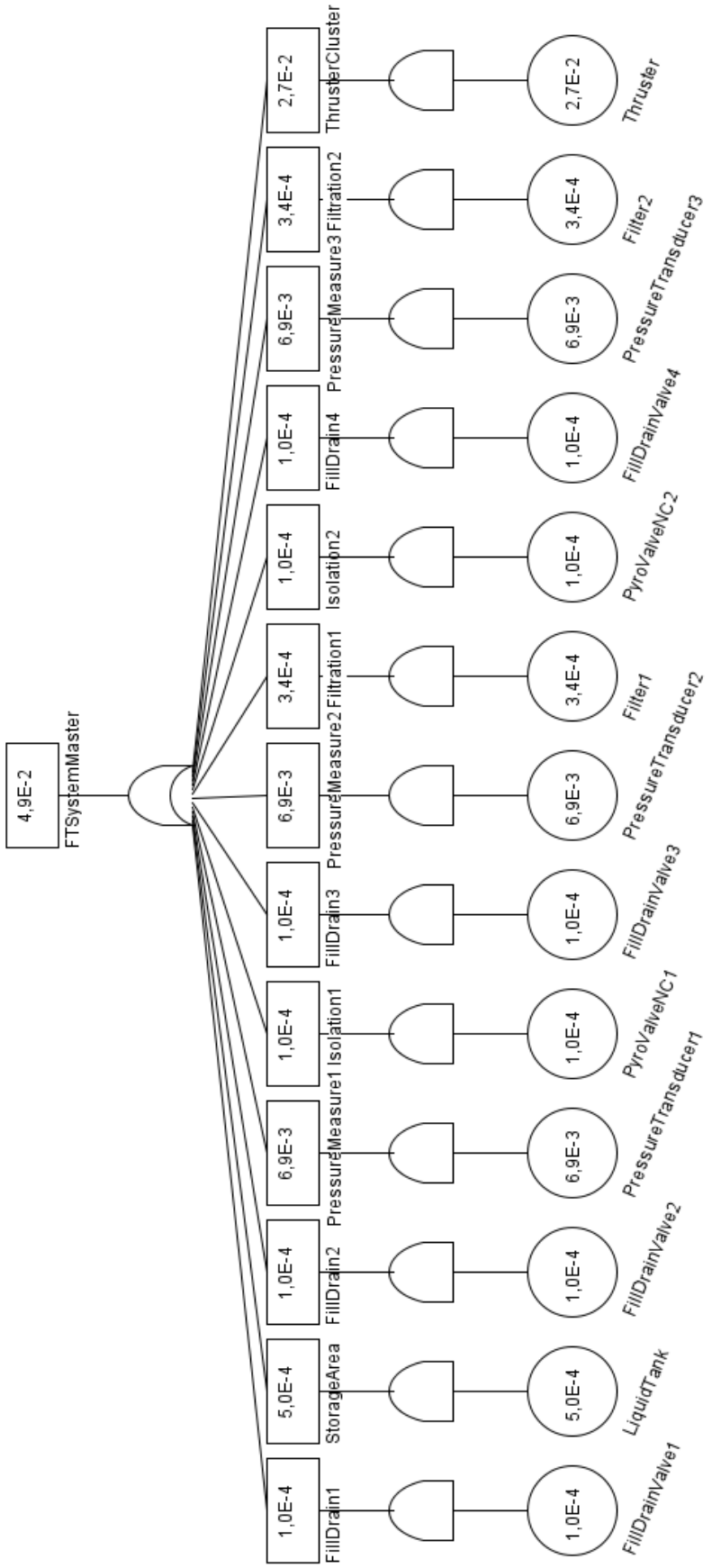

Abbildung B.13: Fehlerbaum eines monergolen Satellitenantriebssystems ohne Zuverlässigkeitsanforderung

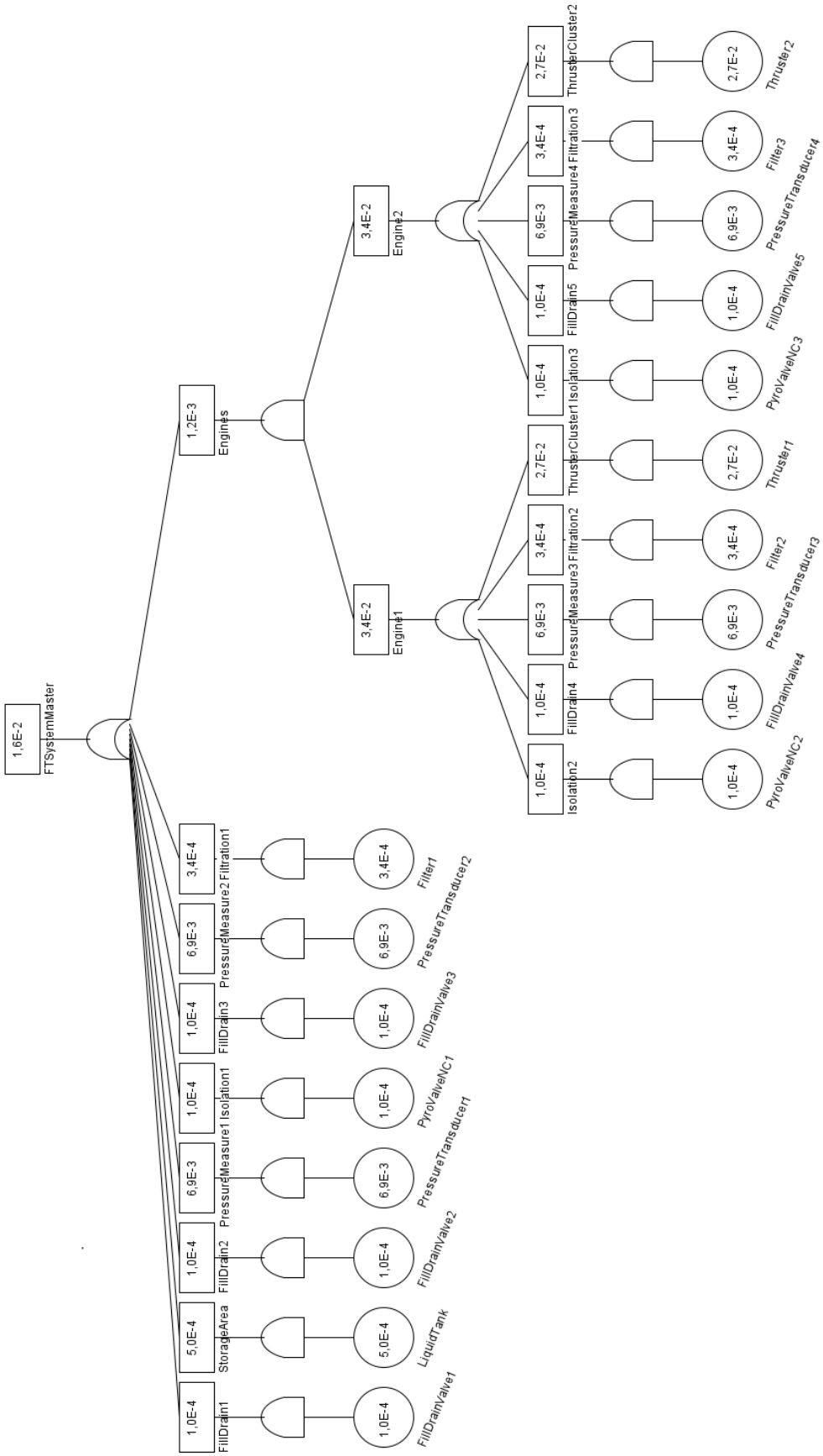

Abbildung B.14: Fehlerbaum eines monergolen Satellitenantriebssystems mit einem redundanten Antriebsstrang (Engine)

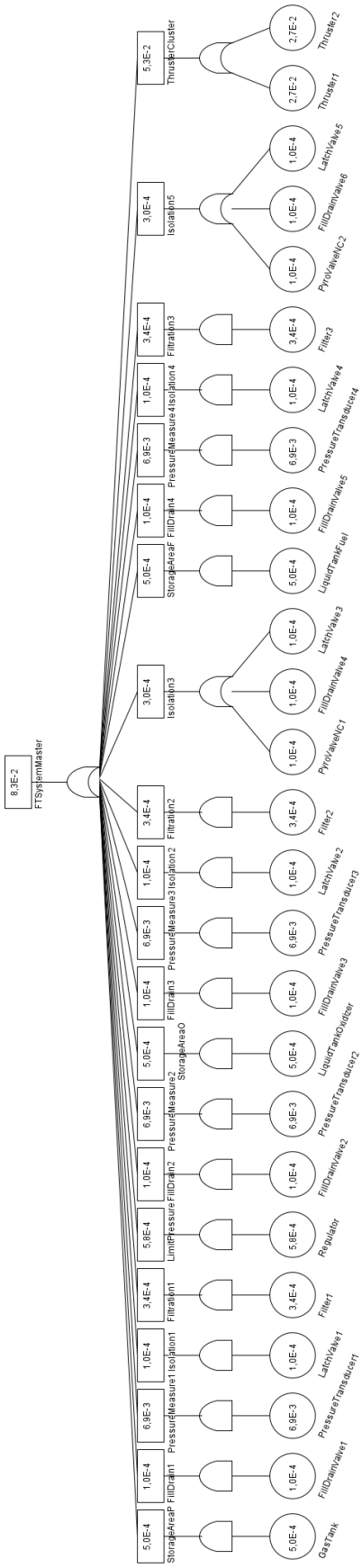

Abbildung B.15: Fehlerbaum eines diergolen Satellitenantriebssystems mit zwei nicht redundanten Triebwerken und Isolationen aus mehreren Komponenten

Anhang C

Mathematische Ausführungen: Brückensystem

In diesem Anhang wird anhand der Ersatzsysteme des Brückensystems aus Kapitel 3.2.3 Brückensysteme die Gleichheit der Strukturfunktionen für die Berechnung der Zuverlässigkeit dieser Ersatzsysteme bei unabhängigen Elementen ausgeführt. Hierbei wurden zur besseren Übersichtlichkeit *Startteil* und *Endteil* aus den Formeln gestrichen.
Die Strukturfunktion für das **parallele** Ersatzsystem, erstellt aus den minimalen Pfaden des Brückensystems, lautet:

$$R = 1-(1-R_1*R_2)*(1-R_1*R_5*R_4)*(1-R_3*R_4)*(1-R_3*R_5*R_2) \tag{C.1}$$

Gleichung mit aufgelösten Klammern:

$$\begin{aligned} R = &-R_1^2R_3^2R_2^2R_4^2R_5^2 + R_1^2R_3^2R_2R_4R_5^2 + R_1^2R_3R_2R_4^2R_5 - R_1^2R_3R_4R_5 \\ &+ R_1R_3^2R_2^2R_4R_5 - R_1R_3^2R_2R_5 + R_1R_3R_2^2R_4^2R_5^2 - R_1R_3R_2R_4R_5^2 \\ &- R_1R_3R_2R_4 + R_1R_3 - R_1R_2R_4^2R_5 + R_1R_4R_5 - R_3R_2^2R_4R_5 + R_3R_2R_5 + R_2R_4 \end{aligned} \tag{C.2}$$

Analytische Lösung für unabhängige Elemente mit $R_i * R_i = R_i$:

$$\begin{aligned} R = &1*R_1R_3 + 1*R_2R_4 + 1*R_1R_4R_5 + 1*R_3R_2R_5 - 1*R_1R_3R_2R_4 - 1*R_1R_3R_2R_5 \\ &- 1*R_1R_3R_4R_5 - 1*R_1R_2R_4R_5 - 1*R_3R_2R_4R_5 + 2*R_1R_3R_2R_4R_5 \end{aligned} \tag{C.3}$$

Die Strukturfunktion für das **serielle** Ersatzsystem, erstellt aus den minimalen Schnitten des Brückensystems, lautet:

$$\begin{aligned} R = &(1-(1-R_1)*(1-R_3))*(1-(1-R_2)*(1-R_4))*(1-(1-R_1) \\ &*(1-R_5)*(1-R_4))*(1-(1-R_3)*(1-R_5)*(1-R_2)) \end{aligned} \tag{C.4}$$

Gleichung mit aufgelösten Klammern:

$$
\begin{aligned}
R = {} & R_3^2R_1^2 + R_3^2R_2^2R_1^2 - R_3R_2^2R_1^2 + R_3^2R_4^2R_1^2 + R_3^2R_2^2R_4^2R_1^2 - 2R_3R_2^2R_4^2R_1^2 + R_2^2R_4^2R_1^2 \\
& - R_3R_4^2R_1^2 - 2R_3^2R_2R_4^2R_1^2 + 3R_3R_2R_4^2R_1^2 - R_2R_4^2R_1^2 + R_3^2R_5^2R_1^2 + R_3^2R_2^2R_5^2R_1^2 \\
& - R_3R_2^2R_5^2R_1^2 + R_3^2R_4^2R_5^2R_1^2 + R_3^2R_2^2R_4^2R_5^2R_1^2 - 2R_3R_2^2R_4^2R_5^2R_1^2 + R_2^2R_4^2R_5^2R_1^2 \\
& - 2R_3R_4^2R_5^2R_1^2 - 2R_3^2R_2R_4^2R_5^2R_1^2 + 4R_3R_2R_4^2R_5^2R_1^2 - 2R_2R_4^2R_5^2R_1^2 + R_4^2R_5^2R_1^2 - R_3R_5^2R_1^2 \\
& - 2R_3^2R_2R_5^2R_1^2 + 2R_3R_2R_5^2R_1^2 - 2R_3^2R_4R_5^2R_1^2 - 2R_3^2R_2^2R_4R_5^2R_1^2 + 3R_3R_2^2R_4R_5^2R_1^2 \\
& - R_2^2R_4R_5^2R_1^2 + 3R_3R_4R_5^2R_1^2 + 4R_3^2R_2R_4R_5^2R_1^2 - 6R_3R_2R_4R_5^2R_1^2 + 2R_2R_4R_5^2R_1^2 \\
& - R_4R_5^2R_1^2 - 2R_3^2R_2R_1^2 + R_3R_2R_1^2 - 2R_3^2R_4R_1^2 - 2R_3^2R_2^2R_4R_1^2 + 3R_3R_2^2R_4R_1^2 - R_2^2R_4R_1^2 \\
& + R_3R_4R_1^2 + 4R_3^2R_2R_4R_1^2 - 4R_3R_2R_4R_1^2 + R_2R_4R_1^2 - 2R_3^2R_5R_1^2 - 2R_3^2R_2^2R_5R_1^2 \\
& + 2R_3R_2^2R_5R_1^2 - 2R_3^2R_4^2R_5R_1^2 - 2R_3^2R_2^2R_4^2R_5R_1^2 + 4R_3R_2^2R_4^2R_5R_1^2 - 2R_2^2R_4^2R_5R_1^2 \\
& + 3R_3R_4^2R_5R_1^2 + 4R_3^2R_2R_4^2R_5R_1^2 - 7R_3R_2R_4^2R_5R_1^2 + 3R_2R_4^2R_5R_1^2 - R_4^2R_5R_1^2 + R_3R_5R_1^2 \\
& + 4R_3^2R_2R_5R_1^2 - 3R_3R_2R_5R_1^2 + 4R_3^2R_4R_5R_1^2 + 4R_3^2R_2^2R_4R_5R_1^2 - 6R_3R_2^2R_4R_5R_1^2 \\
& + 2R_2^2R_4R_5R_1^2 - 4R_3R_4R_5R_1^2 - 8R_3^2R_2R_4R_5R_1^2 + 10R_3R_2R_4R_5R_1^2 - 3R_2R_4R_5R_1^2 + R_4R_5R_1^2 \\
& - R_3^2R_2^2R_1 + R_3R_2^2R_1 - R_3^2R_4^2R_1 - 2R_3^2R_2^2R_4^2R_1 + 4R_3R_2^2R_4^2R_1 - 2R_2^2R_4^2R_1 + R_3R_4^2R_1 \\
& + 3R_3^2R_2R_4^2R_1 - 4R_3R_2R_4^2R_1 + R_2R_4^2R_1 - R_3^2R_5^2R_1 - 2R_3^2R_2^2R_5^2R_1 + 2R_3R_2^2R_5^2R_1 \\
& - R_3^2R_4^2R_5^2R_1 - 2R_3^2R_2^2R_4^2R_5^2R_1 + 4R_3R_2^2R_4^2R_5^2R_1 - 2R_2^2R_4^2R_5^2R_1 + 2R_3R_4^2R_5^2R_1 \\
& + 3R_3^2R_2R_4^2R_5^2R_1 - 6R_3R_2R_4^2R_5^2R_1 + 3R_2R_4^2R_5^2R_1 - R_4^2R_5^2R_1 + R_3R_5^2R_1 + 3R_3^2R_2R_5^2R_1 \\
& - 3R_3R_2R_5^2R_1 + 2R_3^2R_4R_5^2R_1 + 4R_3^2R_2^2R_4R_5^2R_1 - 6R_3R_2^2R_4R_5^2R_1 + 2R_2^2R_4R_5^2R_1 \\
& - 3R_3R_4R_5^2R_1 - 6R_3^2R_2R_4R_5^2R_1 + 9R_3R_2R_4R_5^2R_1 - 3R_2R_4R_5^2R_1 + R_4R_5^2R_1 + R_3^2R_2R_1 \\
& + R_3^2R_4R_1 + 3R_3^2R_2^2R_4R_1 - 4R_3R_2^2R_4R_1 + R_2^2R_4R_1 - 4R_3^2R_2R_4R_1 + 2R_3R_2R_4R_1 + R_3^2R_5R_1 \\
& + 3R_3^2R_2^2R_5R_1 - 3R_3R_2^2R_5R_1 + 2R_3^2R_4^2R_5R_1 + 4R_3^2R_2^2R_4^2R_5R_1 - 8R_3R_2^2R_4^2R_5R_1 \\
& + 4R_2^2R_4^2R_5R_1 - 3R_3R_4^2R_5R_1 - 6R_3^2R_2R_4^2R_5R_1 + 10R_3R_2R_4^2R_5R_1 - 4R_2R_4^2R_5R_1 + R_4^2R_5R_1 \\
& - 4R_3^2R_2R_5R_1 + 2R_3R_2R_5R_1 - 3R_3^2R_4R_5R_1 - 7R_3^2R_2^2R_4R_5R_1 + 10R_3R_2^2R_4R_5R_1 \\
& - 3R_2^2R_4R_5R_1 + 2R_3R_4R_5R_1 + 10R_3^2R_2R_4R_5R_1 - 10R_3R_2R_4R_5R_1 + 2R_2R_4R_5R_1 + R_3^2R_2^2R_4^2 \\
& - 2R_3R_2^2R_4^2 + R_2^2R_4^2 - R_3^2R_2R_4^2 + R_3R_2R_4^2 + R_3^2R_2^2R_5^2 - R_3R_2^2R_5^2 + R_3^2R_2^2R_4^2R_5^2 - 2R_3R_2^2R_4^2R_5^2 \\
& + R_2^2R_4^2R_5^2 - R_3^2R_2R_4^2R_5^2 + 2R_3R_2R_4^2R_5^2 - R_2R_4^2R_5^2 - R_3^2R_2R_5^2 + R_3R_2R_5^2 - 2R_3^2R_2^2R_4R_5^2 \\
& + 3R_3R_2^2R_4R_5^2 - R_2^2R_4R_5^2 + 2R_3^2R_2R_4R_5^2 - 3R_3R_2R_4R_5^2 + R_2R_4R_5^2 - R_3^2R_2^2R_4 + R_3R_2^2R_4 \\
& + R_3^2R_2R_4 - R_3^2R_2^2R_5 + R_3R_2^2R_5 - 2R_3^2R_2^2R_4^2R_5 + 4R_3R_2^2R_4^2R_5 - 2R_2^2R_4^2R_5 + 2R_3^2R_2R_4^2R_5 \\
& - 3R_3R_2R_4^2R_5 + R_2R_4^2R_5 + R_3^2R_2R_5 + 3R_3^2R_2^2R_4R_5 - 4R_3R_2^2R_4R_5 + R_2^2R_4R_5 \\
& - 3R_3^2R_2R_4R_5 + 2R_3R_2R_4R_5
\end{aligned}
\tag{C.5}
$$

für unabhängige Elemente gilt $R_i * R_i = R_i$

$$
\begin{aligned}
R = {} & R_3R_1 + R_3R_2R_1 - R_3R_2R_1 + R_3R_4R_1 + R_3R_2R_4R_1 - 2R_3R_2R_4R_1 + R_2R_4R_1 \\
& - R_3R_4R_1 - 2R_3R_2R_4R_1 + 3R_3R_2R_4R_1 - R_2R_4R_1 + R_3R_5R_1 + R_3R_2R_5R_1 \\
& - R_3R_2R_5R_1 + R_3R_4R_5R_1 + R_3R_2R_4R_5R_1 - 2R_3R_2R_4R_5R_1 + R_2R_4R_5R_1 \\
& - 2R_3R_4R_5R_1 - 2R_3R_2R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 2R_2R_4R_5R_1 + R_4R_5R_1 - R_3R_5R_1 \\
& - 2R_3R_2R_5R_1 + 2R_3R_2R_5R_1 - 2R_3R_4R_5R_1 - 2R_3R_2R_4R_5R_1 + 3R_3R_2R_4R_5R_1 \\
& - R_2R_4R_5R_1 + 3R_3R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 6R_3R_2R_4R_5R_1 + 2R_2R_4R_5R_1 \\
& - R_4R_5R_1 - 2R_3R_2R_1 + R_3R_2R_1 - 2R_3R_4R_1 - 2R_3R_2R_4R_1 + 3R_3R_2R_4R_1 - R_2R_4R_1 \\
& + R_3R_4R_1 + 4R_3R_2R_4R_1 - 4R_3R_2R_4R_1 + R_2R_4R_1 - 2R_3R_5R_1 - 2R_3R_2R_5R_1 \\
& + 2R_3R_2R_5R_1 - 2R_3R_4R_5R_1 - 2R_3R_2R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 2R_2R_4R_5R_1 \\
& + 3R_3R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 7R_3R_2R_4R_5R_1 + 3R_2R_4R_5R_1 - R_4R_5R_1 + R_3R_5R_1 \\
& + 4R_3R_2R_5R_1 - 3R_3R_2R_5R_1 + 4R_3R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 6R_3R_2R_4R_5R_1 \\
& + 2R_2R_4R_5R_1 - 4R_3R_4R_5R_1 - 8R_3R_2R_4R_5R_1 + 10R_3R_2R_4R_5R_1 - 3R_2R_4R_5R_1 + R_4R_5R_1 \\
& - R_3R_2R_1 + R_3R_2R_1 - R_3R_4R_1 - 2R_3R_2R_4R_1 + 4R_3R_2R_4R_1 - 2R_2R_4R_1 + R_3R_4R_1 \\
& + 3R_3R_2R_4R_1 - 4R_3R_2R_4R_1 + R_2R_4R_1 - R_3R_5R_1 - 2R_3R_2R_5R_1 + 2R_3R_2R_5R_1 \\
& - R_3R_4R_5R_1 - 2R_3R_2R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 2R_2R_4R_5R_1 + 2R_3R_4R_5R_1 \\
& + 3R_3R_2R_4R_5R_1 - 6R_3R_2R_4R_5R_1 + 3R_2R_4R_5R_1 - R_4R_5R_1 + R_3R_5R_1 + 3R_3R_2R_5R_1 \\
& - 3R_3R_2R_5R_1 + 2R_3R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 6R_3R_2R_4R_5R_1 + 2R_2R_4R_5R_1 \\
& - 3R_3R_4R_5R_1 - 6R_3R_2R_4R_5R_1 + 9R_3R_2R_4R_5R_1 - 3R_2R_4R_5R_1 + R_4R_5R_1 + R_3R_2R_1 \\
& + R_3R_4R_1 + 3R_3R_2R_4R_1 - 4R_3R_2R_4R_1 + R_2R_4R_1 - 4R_3R_2R_4R_1 + 2R_3R_2R_4R_1 + R_3R_5R_1 \\
& + 3R_3R_2R_5R_1 - 3R_3R_2R_5R_1 + 2R_3R_4R_5R_1 + 4R_3R_2R_4R_5R_1 - 8R_3R_2R_4R_5R_1 \\
& + 4R_2R_4R_5R_1 - 3R_3R_4R_5R_1 - 6R_3R_2R_4R_5R_1 + 10R_3R_2R_4R_5R_1 - 4R_2R_4R_5R_1 + R_4R_5R_1 \\
& - 4R_3R_2R_5R_1 + 2R_3R_2R_5R_1 - 3R_3R_4R_5R_1 - 7R_3R_2R_4R_5R_1 + 10R_3R_2R_4R_5R_1 \\
& - 3R_2R_4R_5R_1 + 2R_3R_4R_5R_1 + 10R_3R_2R_4R_5R_1 - 10R_3R_2R_4R_5R_1 + 2R_2R_4R_5R_1 + R_3R_2R_4 \\
& - 2R_3R_2R_4 + R_2R_4 - R_3R_2R_4 + R_3R_2R_4 + R_3R_2R_5 - R_3R_2R_5 + R_3R_2R_4R_5 - 2R_3R_2R_4R_5 \\
& + R_2R_4R_5 - R_3R_2R_4R_5 + 2R_3R_2R_4R_5 - R_2R_4R_5 - R_3R_2R_5 + R_3R_2R_5 - 2R_3R_2R_4R_5 \\
& + 3R_3R_2R_4R_5 - R_2R_4R_5 + 2R_3R_2R_4R_5 - 3R_3R_2R_4R_5 + R_2R_4R_5 - R_3R_2R_4 + R_3R_2R_4 \\
& + R_3R_2R_4 - R_3R_2R_5 + R_3R_2R_5 - 2R_3R_2R_4R_5 + 4R_3R_2R_4R_5 - 2R_2R_4R_5 + 2R_3R_2R_4R_5 \\
& - 3R_3R_2R_4R_5 + R_2R_4R_5 + R_3R_2R_5 + 3R_3R_2R_4R_5 - 4R_3R_2R_4R_5 + R_2R_4R_5 \\
& - 3R_3R_2R_4R_5 + 2R_3R_2R_4R_5
\end{aligned}
\tag{C.6}
$$

Analytische Lösung für unabhängige Elemente:

$$\begin{aligned} R = {} & 1 * R_1R_3 + 1 * R_2R_4 + 1 * R_1R_4R_5 + 1 * R_3R_2R_5 - 1 * R_1R_3R_2R_4 - 1 * R_1R_3R_2R_5 \\ & - 1 * R_1R_3R_4R_5 - 1 * R_1R_2R_4R_5 - 1 * R_3R_2R_4R_5 + 2 * R_1R_3R_2R_4R_5 \end{aligned} \tag{C.7}$$

Die Strukturfunktionen für das parallele (Gleichung C.1) und das serielle Ersatzsystem (Gleichung C.4) des Brückensystems zur Berechnung der Zuverlässigkeit sind für unabhängige Elemente identisch, wie in den umgestellten Formeln der analytischen Lösungen in den Gleichungen C.3 und C.7 zu sehen ist.

Literaturverzeichnis

[1] Alber, R.; Rudolph, S. (2003): *"43" - A Generic Approach for Engineering Design Grammars.* Proceedings of American Association for Artificial Intelligence, Spring Symposium, Stanford, Kalifornien, USA.

[2] Andow, P. K. (1980): *Difficulties in Fault-Tree Synthesis for Process Plant*. Transactions on Reliability, Volume R-29, Nummer 1, Institute of Electrical and Electronics Engineers.

[3] Andrews, John (2012): *Introduction to Fault Tree Analysis*. Annual Reliability and Maintainability Symposium, Nottingham, Großbritannien.

[4] Arnett, J.; Gindorf, T. (1990): *Reliability Analyses Handbook*. Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Kalifornien, USA.

[5] Automotive Industry Action Group (2008): *FMEA Reference Manual*. 4. Auflage.

[6] Avenhaus, Rudolf (2003): *Zuverlässigkeitstheorie*. Universität der Bundeswehr München, München, Deutschland.

[7] Barlow, R. E.; Proschan, F. (1975): *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Winston, New York, USA.

[8] Barlow, R. E.; Proschan, F. (1981): *Statistical theory of reliability and life testing: probability models*. McArdle Press, Inc., Silver-Spring, MD, USA.

[9] Bellman, R. E. (1958): *On a routing problem*. Quarterly of Applied Mathematics, 16(1), Brown University, Providence, USA. (87-90)

[10] Billinton, Roy; Allan, Ronald N. (1996): *Reliability Evaluation of Power Systems*. 2. Auflage, Plenum Press.

[11] Billinton, R.; Bobyn, J. E.; Krasnodebski; J. (1973): *Reliability and Maintainability in the Design Phase of a Generating Station*. Canadian Electrical Association, Teil III, Bericht Nr. 73-SP-152, Toronto, Kanada, März 1973.

[12] Birnbaum, Z. W. (1968): *On the Importance of different Components in a multicomponent System*. Technischer Bericht Nr. 54, University of Washington, Seattle, Washington, USA.

[13] Birolini, Alessandro (1985): *Qualität und Zuverlässigkeit technischer Systeme: Theorie, Praxis, Management*. Springer-Verlag, Berlin Heidelberg.

[14] Bittner, B.; Bozzano, M.; Cavada, R.; Cimatti, A.; Gario, M.; Griggio, A.; Mattarei, C.; Micheli, A.; Zampedri,G. (2016): *The xSAP Safety Analysis Platform*. Proceedings of 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Eindhoven, Niederlande, 2. - 8. April 2016.

[15] Bölling, Michael (2015): *Lösungspfadbasierte Analysen im Entwurf komplexer Systeme*. Dissertation, Universität Stuttgart, Stuttgart, Deutschland.

[16] Böhnke, Daniel; Reichwein, Axel; Rudolph, Stephan (2009): *Design Language for Airplane Geometries using the Unified Modeling Language*. ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 5, 35th Design Automation Conference, Parts A and B, San Diego, Kalifornien, USA, 30. August - 2. September 2009. (18)

[17] Bossche, Andre (1988): *Fault Tree Analysis and Synthesis*. Dissertation, Technische Universiteit Delft, Delft, Niederlande.

[18] Bunge, Mario (1958): *Causality: The Place of the Causal Principle in Modern Science.*. Buenos Aires, Argentinien.

[19] CEI/IEC 61025 (2006): *Analyse par arbre de panne (AAP); Fault tree analysis (FTA)*. International Electrotechnical Commission, Referenznummer CEI/IEC 61025:2006, Schweiz.

[20] Cepkin, Marko; Mavko, Borut (2002): *A dynamic fault tree*. Reliability Engineering and System Safety, Volume 75, Elsevier. (83-91)

[21] Chekuri, Chandra S.; Goldberg, Andrew V.; Karger, David R.; Levine, Mattew S.; Stein, Cliff (1997): *Experimental Study of Minimum Cut Algorithms*. SODA '97, Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, Louisiana, USA, 05. - 07. Januar 1997, Institute of Electrical and Electronics Engineers. (324-333)

[22] Deutsche Gesellschaft für Qualität e.V. (2012): *FMEA - Fehlermöglichkeits- und Einflussanalyse*. DGQ-Band 13-11, 5. Auflage, Beuth Verlag GmbH, Frankfurt, Deutschland. (23)

[23] Dijkstra, Edsger W. (1959): *A note on two problems in connexion with graphs*. Numerische Mathematik 1. (269-271)

[24] DIN 25424 Teil 1 (1981) und Teil 2 (1990): *Fehlerbaumanalyse Teil 1: Methode und Bildzeichen, Teil 2: Handrechenverfahren zur Auswertung eines Fehlerbaumes*. Deutsche Norm, Beuth Verlag GmbH, Berlin, Deutschland.

[25] DIN 25448 (1990): *Ausfalleffektanalyse (Fehler-Möglichkeits- und -Einfluß-Analyse)*. Deutsche Norm, Beuth Verlag GmbH, Berlin, Deutschland.

[26] DIN EN 60812 (2006): *Analysetechniken für die Funktionsfähigkeit von Systemen - Verfahren für die Fehlzustandsart- und -auswirkungsanalyse (FMEA)*. Deutsche Kommission Elektrotechnik Elektronik Informationstechnik, Deutsche Norm, Beuth Verlag GmbH, Berlin, Deutschland.

[27] DIN EN 61025 (2007): *Fehlzustandsbaumanalyse*. Deutsche Kommission Elektrotechnik Elektronik Informationstechnik, Deutsche Norm, Beuth Verlag GmbH, Berlin, Deutschland.

[28] DIN EN 61078 (2015): *Techniken für die Analyse der Zuverlässigkeit - Zuverlässigkeitsblockdiagramm und BooleŽsche Verfahren*. Deutsche Kommission Elektrotechnik Elektronik Informationstechnik, Norm-Entwurf, Beuth Verlag GmbH, Berlin, Deutschland.

[29] DIN EN ISO 13849-1 (2016): *Sicherheit von Maschinen - Sicherheitsbezogene Teile von Steuerungen - Teil 1: Allgemeine Gestaltungsleitsätze*. Deutsche Kommission Elektrotechnik Elektronik Informationstechnik, Deutsche Norm, Beuth Verlag GmbH, Berlin, Deutschland.

[30] Dinic, E. A. (1970): *Algorithm for solution of a problem of maximum flow in networks with power estimation*. Soviet Mathematics - Doklady, Volume 11, No. 5. (1277-1280)

[31] Dong Conglin; Yuan Chengqing; Liu Zhenglin; Yan Xinping (2013): *Marine Propulsion System Reliability Research Based on Fault Tree Analysis*. Advanced Shipping and Ocean Engineering, Volume 2, Issue 1, März 2013. (27-33)

[32] Du Plessis, Alfred; Frank, Karen; Saglimbene, Mark; Ozarin, Nathaniel (2014): *The Thirty Greatest Reliability Challenges*. 2014 Annual Reliability and Maintainability Symposium, Colorado Springs, Colorado, USA, 27. - 30. Januar 2014, Institute of Electrical and Electronics Engineers.

[33] Edmonds, J.; Karp, R. M. (1972): *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*. Journal of ACM, Volume 19. (248-264)

[34] Ericson, Clif (1999): *Fault Tree Analysis - A History*. Proceedings of The 17th International System Safety Conference, Orlando, Florida, USA, 16. - 21. August 1999.

[35] Feiler, Peter H.; Lewis, Bruce A.; Vestal, Steve (2006): *The SAE Architecture Analysis & Design Language (AADL) - A Standard for Engineering Performance Critical Systems*. Proceedings of the Conference on Computer Aided Control Systems Design, München, Deutschland, 04. - 06. Oktober 2006, Institute of Electrical and Electronics Engineers.

[36] Fischer, T.; Niere, J.; Torunski, L.; Zündorf, A. (2000): *Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java*. Theory and Application of Graph Transformations, Volume 1764, Lecture Notes in Computer Science, Springer Berlin Heidelberg. (296-309)

[37] Floyd, Robert W. (1962): *Algorithm 97 (SHORTEST PATH)*. Communications of the ACM 5(6). (345)

[38] Ford Jr., L. R.; Fulkerson, D. R. (1956): *Maximal flow through a network*. Canadian Journal of Mathematics, Volume 8. (399-404)

[39] Freese, Wolfgang (2015): *Erstellung von Fehlerbäumen*. Carl Hanser Verlag, München, Deutschland.

[40] Gang Wang; Xiaoqin Su; Xuhua Pan (2012): *Computer Aided Visualized Fault Tree Analysis*. Fifth International Symposium on Computational Intelligence and Design, Hangzhou, China, 28. - 29. Oktober 2012, Institute of Electrical and Electronics Engineers.

[41] Garro, Alfredo; Groß, Johannes; Riestenpatt genannt Richter, Marius; Tundis, Andrea (2012): *Experimenting the RAMSAS method in the reliability analysis of an attitude determination and control system (ADCS)*. Proceedings of the International Workshop on Applied Modeling and Simulation (WAMS 2012), jointly held with the NATO CAX FORUM, Rom, Italien, 24. - 27. September 2012.

[42] Garro, Alfredo; Groß, Johannes; Riestenpatt genannt Richter, Marius; Tundis, Andrea (2014): *Reliability analysis of an attitude determination and control system (ADCS) through the RAMSAS method*. Journal of Computational Science, Volume 5, Nummer 3, Elsevier B.V., Amsterdam, Niederlande. (439-449)

[43] Garro, Alfredo; Tundis, Andrea (2012): *Modeling and Simulation for System Reliability Analysis: The RAMSAS Method*. Proceedings of the 7th IEEE International Conference on System of Systems Engineering, Genua, Italien, 16. - 19. Juli 2012, Institute of Electrical and Electronics Engineers.

[44] Goldberg, A. V.; Tarjan, R. E. (1986): *A new approach to the maximum flow problem*. Proceedings of the 18th ACM Symposium on Theory of Computing. ACM, New York. (136-146)

[45] Göttler, Herbert (1988): *Graphgrammatiken in der Softwaretechnik: Theorie und Anwendungen*. Springer-Verlag, Berlin, Deutschland.

[46] Greulich, Walter (1999): *Lexikon der Physik : in sechs Bänden. - 3. Band Ha bis Mh.*. Heidelberg: Spektrum Akademischer Verlag, Berlin, Deutschland.

[47] Groß, Johannes; Rudolph, Stephan (2012): *Dependency Analysis in Complex System Design using the FireSat example*. INCOSE International Symposium, Rom, Italien. (14)

[48] Halpern, Joseph Y. (2015): *A modification of the Halpern-Pearl definition of causality*. Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentinien, 25. - 31. Juli 2015. (3022-3033)

[49] Halpern, Joseph Y. (2016): *Actual Causality*. The MIT Press, Cambridge, Massachusetts, USA.

[50] Halpern, Joseph Y. (2008): *Defaults and normality in causal structures*. In Principles of Knowledge Representation and Reasoning: Proceedings of The Eleventh International Conference (KR). (198-208)

[51] Halpern, Joseph Y.; Pearl, Judea (2005): *Causes and Explanations: A Structural-Model Approach. Part II: Explanations*. British Journal for Philosophy of Science, Volume 56, Nummer 4. (843-887)

[52] Härtung, Joachim; Elpelt, Bärbel; Klösener, Karl-Heinz (1989): *Lehr- und Handbuch der angewandten Statistik*. Oldenbourg Verlag München/Wien.

[53] Hillenbrand, Martin (2011): *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen*. KIT Scientific Publishing, Karlsruhe, Deutschland.

[54] Hume, David (1748): *An Enquiry Concerning Human Understanding - Untersuchung in Betreff des menschlichen Verstandes*. Übersetzung: von Kirchmann, J. H. (1869).

[55] IILS mbH (2017): *Entwurfscompiler 43*. URL www.iils.de, Stand: 02.09.17, Ingenieurgesellschaft für intelligente Lösungen und Systeme mbH.

[56] Jakob, Frank; Schweizer, Volker; Bertsche, Bernd; Dobry, ALexander (2014): *Comprehensive Approach for the Reliability Prediction of Complex Systems*. Proceedings of Reliability and Maintainability Symposium (RAMS), Colorado Springs, Colorado, USA, 27. - 30. Januar 2014, Institute of Electrical and Electronics Engineers.

[57] Jensen, Paul A.; Bard, Jonathan F. (2001): *Operations Research Models and Methods*. John Wiley and Sons, USA.

[58] Johansson, Cristina (2013): *On System Safety and Reliability in Early Design Phases*. Linköping Studies in Science and Technology, Thesis No. 1600, Division of Machine Design, Department of Management and Engineering, Linköping University, Linköping, Schweden.

[59] Kaiser, Bernhard (2003): *A Fault-Tree Semantics to model Software-Controlled Systems*. Hasso-Plattner-Institut für Software Systems Engineering, Universität Potsdam, Potsdam, Deutschland.

[60] Kocza, Gabor; Bossche, Andre (1997): *Automatic Fault Tree Synthesis and Real-Time Tree Trimming based on Computer Models*. Technische Universiteit Delft, Delft, Niederlande.

[61] Kumamoto, Hiromitsu; Henley, Ernest G. (1996): *Probabilistic Risk Assessment and Management for Engineers and Scientists*. Institute of Electrical and Electronics Engineers, Inc., New York, New York, USA.

[62] Kusiak, A. (1999): *Engineering Design: Products, Processes, and Systems*. Academic Press, San Diego, CA, USA.

[63] Lapp, Steven A.; Powers, Gary J. (1977): *Computer-aided Synthesis of Fault-trees*. Transactions on Reliability, Institute of Electrical and Electronics Engineers.

[64] Leitner-Fischer, Florian; Leue, Stefan (2013): *Probabilistic fault tree synthesis using causality computation*. Technical Report soft-13-03, Chair for Software Engineering, Universität Konstanz, Konstanz, Deutschland.

[65] Leitner-Fischer, Florian; Leue, Stefan (2011): *QuantUM: Quantitative Safety Analysis of UML Models*. Proceedings of the 9th Workshop

on Quantitative Aspects of Programming Languages (QAPL), Saarbrücken, Deutschland, 1. - 3. April 2011.

[66] Leitner-Fischer, Florian; Leue, Stefan (2014): *SpinCause: A Tool for Causality Checking*. Proceedings of the International SPIN Symposium on Model Checking of Software (SPIN), San Jose, Kalifornien, USA.

[67] Leveson, Nancy G. (1995): *Safeware*. Addison-Wesley Publishing Company, University of Washington, Washington, USA.

[68] Lewis, David (1973): *Causation*. Journal of Philosophy, Volume 70, New York, USA. (556-567)

[69] Limnios, N.; Ziani, R. (1986): *An Algorithm For Reducing Cut Sets in Fault-Tree Analysis*. Transactions on Reliability, Volume R-35, Nummer 5, Institute of Electrical and Electronics Engineers.

[70] Limnios, Nikolaos (2005): *Fault Trees*. ISTE Ltd, Antony Rowe Ltd, Chippenham, Wiltshire, Großbritannien.

[71] Mackie, John L. (1974): *The Cement of the Universe: A Study of Causation*. Oxford University Press, Oxford, Großbritannien.

[72] Mahmud, Nidhal (2012): *Dynamic Model-based Safety Analysis-From State Machines to Temporal Fault Trees*. Dissertation, University of Hull, Hull, Großbritannien.

[73] National Aeronautics and Space Administration (2000): *Failure Modes and Effects Analysis (FMEA): A Bibliography*. Technische Berichte, NASA Scientific and Technical Information Program, USA.

[74] Noser, H-R.; Rudolph, S.; Stucki, P. (2001): *Physics-Enhanced L-Systems*. Proceedings 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2001), Pilsen, Tschechien, 5. - 9. Februar 2001.

[75] Paakko, Matti; Myllymäki, Petri; Holsti, Niklas; Tirri, Henry: *Modern AI Approaches for Building Robust FDIR Components*. Helsinki, Finnland.

[76] Paganin, Lucas, Borsato, Milton (2017): *A critical review of Design for Reliability - A bibliometric analysis and identification of research oppor-*

tunities. FAIM2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, Modena, Italy, 27. - 30. Juni 2017.

[77] Pai, Ganesh J.; Dugan, Joanna Bechta (2002): *Automatic Synthesis of Dynamic Fault Trees from UML System Models*. Proceedings of the 13th International Symposium on Software Reliability Engineering, Annapolis, Maryland, USA, 12. - 15. November 2002, Institute of Electrical and Electronics Engineers.

[78] Papadopoulos, Yiannis; Parker, David; Grante, Christian (2004): *A Method and Tool Support for Model-based Semi-automated Failure Modes and Effects Analysis of Engineering Designs*. SCS '04, Proceedings of the 9th Australian workshop on Safety critical systems and software, Volume 47. (89-95)

[79] Papadopoulos, Yannis; Maruhn, Matthias (2001): *Model-Based Synthesis of Fault Trees from Matlab-Simulink models*. International Conference on Dependable Systems and Networks Göteborg, Schweden, 01. - 04. Juli 2001, Institute of Electrical and Electronics Engineers.

[80] Perl, J. (1981): *Graphentheorie, Grundlagen und Anwendungen*. Akademische Verlagsgesellschaft, Wiesbaden, Deutschland.

[81] Pickard, Karsten (2009): *Erweiterte qualitative Zuverlässigkeitsanalyse mit Ausfallprognose von Systemen*. Disseration, Universität Stuttgart, Stuttgart, Deutschland.

[82] Rai, S.; Aggarwal, K. K. (1978): *An efficient method for reliability evaluation of a general network*. IEEE Transactions on Reliability, Volume 27. (206-211)

[83] Rasmuson, D. H.; Marshall N. H. (1978): *FATRAM - A Core Efficient Cut-Set Algorithm*. Transactions on Reliability, Volume R-27, Nummer 4, Institute of Electrical and Electronics Engineers.

[84] Rauzy, Antoine (1993): *New Algorithms for Fault Tree Analysis*. Reliability Engineering and System Safety, Volume 40, Elsevier. (203-211)

[85] Redmill, Felix, Chudleigh, Morris, Catmur, James (1999): *System Safety: HAZOP and Software HAZOP*. John Wiley & Sons Ltd., Chichester, Großbritannien.

[86] ReliaSoft Corporation (2015): *System Analysis Reference*. Tucson, Arizona, USA.

[87] Riestenpatt genannt Richter, Marius; Rudolph, Stephan (2019): *A scientific discourse on creativity and innovation in the formal context of graph-based design languages*. Angenommenes Manuskript zur 13th Computational and Cognitive Models of Creative Design, Heron Island, Queensland, Australia, 15. - 18. Dezember 2019. (19)

[88] Riestenpatt genannt Richter, Marius; Schmidt, Jens; Rudolph, Stephan (2014): *Automated fault tree analysis for satellite propulsion systems*. Proceedings of the 6th International Systems & Concurrent Engineering for Space Applications Conference (SECESA), Stuttgart, Deutschland, 8. - 10. Oktober 2014. (8)

[89] Rosenthal, A. (1979): *Approach to comparing cut-set enumeration algorithms*. IEEE Transactions on Reliability, Volume 28. (62-65)

[90] Rozenberg, Grzegorz (1997): *Handbook of Graph Grammars and Computing by Graph Transformation*. 3 Bände, World Scientific Publishing Co., Inc., River Edge, New Jersey, USA.

[91] Rudolph, Stephan (2003): *Aufbau und Einsatz von Entwurfssprachen für den wissensbasierten Ingenieurentwurf*. 3. Forum Knowledge Based-Engineering, CAT-PRO, Stuttgart, Deutschland.

[92] Rudolph, Stephan (2011): *On design process modelling aspects in complex systems*. 13th NASA-ESA Workshop on Product Data Exchange (PDE), Cypress, Kalifornien, USA, 11. - 12. Mai 2011.

[93] Rudolph, Stephan (2008): *Übersichtsvortrag Ähnlichkeitsmechanik*. Vorlesung Ähnlichkeitsmechnik, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, Stuttgart, Deutschland.

[94] Rudolph, Stephan (2002): *Übertragung von Ähnlichkeitsbegriffen*. Habilitationsschrift, Universität Stuttgart, Stuttgart, Deutschland.

[95] Rudolph, Stephan; Kröplin Bernd (1994): *Über die systematische Bewertung von Konstruktionen*. Bauingenieur, Volume 69 (1), Stuttgart, Deutschland. (3-11)

[96] Ruppert, Heiko (2002): *CAD-integrierte Zuverlässigkeitsanalyse und -optimierung*. Dissertation, Universität Stuttgart, Stuttgart, Deutschland.

[97] Schaefer, Jörg; Rudolph, Stephan (2004): *Satellitenentwurf mit Entwurfsgrammatiken*. (11)

[98] Schmidt, Jens (2013): *Entwurfssprache für Satellitenantriebssysteme*. Diplomarbeit, Universität Stuttgart, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Deutschland.

[99] Schmidt, Jens (2017): *Total Engineering Automation*. White Paper, Ingenieurgesellschaft für intelligente Lösungen und Systeme mbH, Leinfelden-Echterdingen, Deutschland, Version: 06.10.2017. (20)

[100] Schmidt, Jens; Rudolph, Stephan (2016): *Automation Opportunities in the Conceptual Design of Satellite Propulsion Systems*. Proceedings of the 7th International Systems & Concurrent Engineering for Space Applications Conference (SECESA), Madrid, Spanien, 5. - 7. Oktober 2016. (8)

[101] Schneeweiss, Winfrid G. (1999): *Die Fehlerbaum-Methode*. LiLoLe-Verlag, Hagen, Deutschland.

[102] Schilling, Simon (2009): *Beitrag zur dynamischen Fehlerbaumanalyse ohne Modulbildung und zustandsbasierte Erweiterungen*. Dissertation, Bergische Universität Wuppertal, Wuppertal, Deutschland.

[103] Schwindt, Eike (2004): *Gefahrenanalyse mittels Fehlerbaumanalyse*. Seminararbeit, Universität Paderborn, Paderborn, Deutschland.

[104] Sinnamon, Roslyn M.; Andrews, John D. (1996): *Fault Tree Analysis and Binary Decision Diagrams*. Proceedings of the Annual Reliability and Maintainability Symposium, Las Vegas, Nevada, USA, 22. - 25.

Januar 1996, Institute of Electrical and Electronics Engineers. (215-222)

[105] Stachowiak, H. (1973): *Allgemeine Modelltheorie.* Springer Verlag, Wien, Österreich.

[106] Stoer, Mechthild; Wagner, Frank (1997): *A Simple Min-Cut Algorithm*. Journal of the ACM, Volume 44, Nummer 4. (585-591)

[107] Suiran Yu; Qingyan Yang; Jiwen Liu; Minxian Pan (2011): *A Comparison of FMEA, AFMEA and FTA*. Proceedings of 9th International Conference on Reliability, Maintainability and Safety, Guiyang, China, 12. - 15. Juni 2011, Institute of Electrical and Electronics Engineers. (954-960)

[108] Terruggia, Roberta (2010): *Reliability analysis of probalistic networks*. Doktorarbeit, Universita degli Studi di Torino, Turin, Italien.

[109] Thums, Andreas (2004): *Formale Fehlerbaumanalyse*. Dissertation, Universität Augsburg, Augsburg, Deutschland.

[110] Tonhäuser, Claudia; Rudolph, Stephan (2016): *Individual Coffee Maker Design Using Graph-Based Design Languages*. Proceedings of the 7th Design Computing and Cognition Conference, Northwestern University, Evanston (Chicago), USA, 27. - 29. Juni 2016.

[111] U.S. Department of Defense (1949): *MIL-P-1629: Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. Militärische Vorgehensweise, U.S. Department of Defense, USA.

[112] Vemuri, Kiran Kumar; Dugan, Joanne Bechta; Sullivan, Kevin J. (1999): *A Design Language for Automatic Synthesis of Fault Trees*. Proceedings of the annual Reliability and Maintainability Symposium, Washington, Washington, USA, 18. - 21. Januar 1999, Institute of Electrical and Electronics Engineers. (91-96)

[113] Vemuri, Kiran Kumar; Dugan, Joanne Bechta; Sullivan, Kevin J. (1999): *Automatic Synthesis of Fault Trees for Computer-Based Systems*. Transactions on Reliability, Volume 48, Nummer 4, Institute of Electrical and Electronics Engineers. (394-402)

[114] Warshall, Stephen (1962): *A theorem on boolean matrices*. Journal of the ACM 9(1). (11-12)

[115] Watson, Hugh A. (1961): *Launch Control Safety Study*. Bell Telephone Laboratories, Murray Hill, New Jersey, USA.

[116] Wendai Wang; Loman, James; Vassiliou, Pantelis (2004): *Reliability Importance of Components in a Complex System*. Proceedings of the Annual Reliability and Maintainability Symposium, Los Angeles, Kalifornien, USA, 26. - 29. Januar 2004.

[117] Wieber, Martin S.(2015): *Qualitätssicherung von Modelltransformationen - Über das dynamische Testen programmierter Graphersetzungssysteme*. Dissertation, Technische Universität Darmstadt, Darmstadt, Deutschland.

[118] Wissenschaftliche Dienste des Deutschen Bundestages (2016): *Zur Diskussion um die Verkürzung von Produktlebenszyklen*. Fachbereich für Wirtschaft und Technologie, Ernährung, Landwirtschaft und Verbraucherschutz, Tourismus, Deutscher Bundestag, Deutschland, Aktenzeichen: WD 5 - 3000 - 053/16.

[119] Yanjun Wang (2004): *Development of a Computer-Aided Fault Tree Synthesis Mehodology for Quantitative Risk Analysis in the Chemical Process Industry*. Dissertation, Texas A&M University, Texas, USA.

[120] Yuchang Mo; Farong Zhong; Huawen Liu; Quansheng Yang; Gang Cui (2012): *Efficient Ordering Heuristics in Binary Decision Diagram-based Fault Tree Analysis*. Wiley Online Library, Quality and Reliability Engineering International 2013, Volume 29. (307-315)

[121] Vesley, W. E.; Goldberg, F. F.; Roberts, N. H.; Haasl, D. F. (1981): *Fault Tree Handbook*. Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, D.C., USA.

[122] Zandbergen, Barry: *Typical failure rate data of propulsion subsystem components*. Technische Universiteit Delft, Delft, Niederlande.

[123] Zio, E. (2009): *Reliability engineering: Old problems and new challenges*. Reliability Engineering and System Safety Volume 94, Elsevier. (125-141)