STEINBUCH SERIES ON ADVANCES IN INFORMATION TECHNOLOGY

10

NICO ADLER

Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262

KIT Scientific Publishing

Nico Adler

Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262

Band 10
Steinbuch Series on Advances in Information Technology

Karlsruher Institut für Technologie (KIT)
Institut für Technik der Informationsverarbeitung

Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262

von
Nico Adler

Scientific Publishing

Karlsruher Institut für Technologie
Institut für Technik der Informationsverarbeitung

Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262

Zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
von der Fakultät für Elektrotechnik und Informationstechnik des
Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Dipl.-Ing. Nico Adler, geb. in: Freiburg im Breisgau

Tag der mündlichen Prüfung: 18. Dezember 2014
Hauptreferent: Prof. Dr.-Ing. Klaus D. Müller-Glaser
Korreferent: Prof. Dr.-Ing. Sören Hohmann

Impressum

KIT Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2015

ISSN 2191-4737
ISBN 978-3-7315-0442-9
DOI: 10.5445/KSP/1000049863

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als Mitarbeiter am FZI Forschungszentrum Informatik im Forschungsbereich Embedded Systems and Sensors Engineering (ESS).

Ganz besonders bedanke ich mich bei meinem betreuenden Doktorvater Herrn Prof. Dr.-Ing. Klaus D. Müller-Glaser für die Möglichkeit zur Promotion und das mir entgegengebrachte Vertrauen. Er unterstützte mich bei meiner Arbeit durch fachlich wertvolle Diskussionen und zahlreiche Anregungen. In gleicher Weise danke ich Herrn Prof. Dr.-Ing. Sören Hohmann für die Übernahme des Korreferats und das große Interesse an meiner Dissertation. Herrn Prof. Dr.-Ing. Eric Sax als neuem Direktor am FZI möchte ich ebenfalls meinen Dank für sein Interesse an meiner wissenschaftlichen Arbeit aussprechen.

Ein großer Dank gilt meinen Arbeitskollegen für die gemeinsame Zeit und interessante Fachgespräche, insbesondere Dr.-Ing. Christian Köllner, B. Sc. Fernando Sanchez Gil, M. Sc. Stefan Otten, Dipl.-Ing. Johannes Bach, Dipl.-Ing. Michael Dreschmann und Dr.-Ing. Lars Müller. Ich bedanke mich zudem bei allen Studenten, welche mich über die Zeit begleitet haben, für ihren Einsatz. Durch die interdisziplinäre Aufstellung und Zusammenarbeit konnten Thematiken aus unterschiedlichen Perspektiven betrachtet werden.

Im Rahmen des SAFE Projektes danke ich Dr. Alexander Rudolph und Philippe Cuenot recht herzlich für einen aufschlussreichen Austausch zum Thema funktionale Sicherheit sowie Dr. Stefan Voget für eine hervorragende Projektleitung. Im Kontext weiterer Projekte möchte ich mich bei Dr.-Ing. Clemens Reichmann, Dipl.-Ing. Daniel Gebauer, Dr.-Ing. Johannes Matheis, Dr. Eduard Metzker und Dr. Simon Burton für eine ergebnisreiche Zeit bedanken.

Nicht zuletzt danke ich meiner Familie und meinen Freunden für das aufgebrachte Verständnis und die Unterstützung während dieser Zeit.

Karlsruhe, Januar 2015

Nico Adler

Inhaltsverzeichnis

1 Einleitung

Die Lebensumstände unserer heutigen Zeit erfordern ein hohes Maß an Mobilität, zu der das Automobil einen großen Beitrag leistet. Diese Bedeutung zeichnet sich nach [1, S.16] dadurch aus, dass es sich beim Automobil einerseits um eine „universelle, individuelle und motorisierte“ Art der Fortbewegung handelt und es andererseits einen „flexible[n] und schnelle[n] Transport von Gütern“ ermöglicht. Die dadurch gebotene Freiheit äußert sich wiederum in der sozialen Wertschätzung für das Automobil. Allein auf Deutschland bezogen wurden nach dem Jahresbericht 2012 [2, S.21] des Kraftfahrt-Bundesamts (KBA) 3,08 Millionen Personenkraftfahrzeuge neu zugelassen, die Gesamtzahl stieg somit auf 43,43 Millionen [2, S.8]. Trotz des damit verbundenen höheren Verkehrsaufkommens konnte die Anzahl der tödlich Verletzten im deutschen Straßenverkehr deutlich verringert werden, vgl. [3, S.145] und [4, S.146].

Die deutsche Automobilindustrie verzeichnete 2009 einen Jahresumsatz von 263 Milliarden Euro [1, S.16]. Die Umsatzentwicklung zeigt nach [4, S.16] für 2013 bereits ein Wachstum auf über 360 Milliarden Euro, wovon 126 Milliarden auf den Inlandsumsatz entfallen. Dieser erhebliche Anteil von circa „20 Prozent des Gesamtumsatzes der deutschen Industrie“ [1, S.16] bezogen auf 2009 bildet die Automobilbranche als „größten Wirtschaftszweig Deutschlands“ ab. Auch die Anzahl an Beschäftigten mit über 750.000 nach [4, S.18] zeigt den wirtschaftlichen Stellenwert der Branche. Zudem wird die Bedeutung der deutschen Automobilindustrie auch im Exportumsatz ersichtlich, was sich nicht zuletzt auf den hohen Qualitätsstandard der Fahrzeuge zurückführen lässt.

Das steigende Bedürfnis des weltweiten Automobilmarktes und seiner Endkunden zeichnet sich nicht nur in der Forderung nach mehr Komfort und Qualität sondern auch in der Sicherheit ab [5], [122], [Adler2011a]. Zudem fordern die Kunden ein hohes Maß an Zuverlässigkeit der Fahrzeuge. Beispielsweise geht der Trend hinsichtlich der aktiven Fahrzeugsicherheit immer mehr in Richtung der Einführung von zahlreichen Fahrerassistenzsystemen, welche in die Fahrzeugdynamik eingreifen können [3, S.145]. Durch solche Systeme kann die Verkehrssicherheit gleichzeitig kontinuierlich verbessert werden. Im Zuge der weltweit zunehmenden Markteinführung von sowohl Hybrid- als auch Elektrofahrzeugen, welche mit Hochvoltkomponenten ausgestattet sind, gewinnt die Thematik Sicherheit weiter an Bedeutung. Dies betrifft zum einen den Insassenschutz im Falle eines Unfalles, zum anderen höhere Anforderungen an deren sicheren Betrieb [3, S.152]. Unerlässlich ist die Sicherstellung, dass auch Fahrzeuge mit neuen Antriebstechnologien ein hohes Sicherheitsniveau aufweisen.

Durch die damit verbundene Elektrifizierung von Fahrzeugen ist die zunehmende Anzahl an Systemen und Einführung neuer Technologien ein herausforderndes Unterfangen, da die Auswirkungen sich gleichzeitig auf die Komplexität der Elektrik-/Elektronik-Architekturen (EEAs) niederschlagen. Bedingt durch diese Entwicklung müssen die Automobilhersteller und Zulieferfirmen den neuen Fahrzeugtechnologien gerecht werden. Die Automobilindustrie ist durch ihren Ursprung im Maschinenbau geprägt, vgl. [1, S.86f], [123, S.42], [124, S.20]. Das damit verbundene traditionell beeinflusste Entwicklungsvorgehen wandelt sich seit mehreren Jahren, da der Einzug der Fachbereiche Elektrotechnik und Informatik drastisch zugenommen hat und mittlerweile das Alltagsgeschäft mitbestimmt. So erfolgt die Umsetzung innovativer Funktionalitäten zu 90 % durch die Verwendung von Elektrik-/Elektronik (EE)-Systemen [125, S.179].

Der Produktentstehungsprozess von aktuellen und kommenden Fahrzeuggenerationen ist von entscheidender Bedeutung und der Erfolgsfaktor, um sich am Markt zu behaupten. Orientierten sich frühere Vorgehensweisen teilweise noch an einer sequentiellen Abarbeitung der Entwicklungsaufgaben [125, S.310], so sind mittlerweile zahlreiche simultan arbeitende Abteilungen über Firmen verteilt an der Entwicklung von Fahrzeugen beteiligt. Die Komplexität solcher Architekturen und EE-Systeme bei gleichzeitiger gewünschter Variantenvielfalt ist nur schwer vollständig überschaubar und beherrschbar. Der Entwicklungsprozess erstreckt sich vom Automobilhersteller über die gesamte Zulieferpyramide. Hierbei gilt es für die Durchführung der Entwicklungstätigkeiten klar strukturierte und eindeutige Prozesse sowie Abläufe zu definieren, um nach der Systemintegration unter anderem ein funktional sicheres Fahrzeug dem Endkunden bereitzustellen.

Hinsichtlich der Absicherung von funktionaler Sicherheit konnte im Jahr 2011 mit dem Standard ISO 26262 „*Road vehicles - Functional safety*“ der Automobilbranche ein normatives Dokument an die Hand gegeben werden, welches eine domänenspezifische Adaption der IEC 61508 repräsentiert. Abgesehen von der Bewältigung der Umfänge einzuführender Systeme und dem Einzug neuer Technologien wird hierdurch gleichzeitig von allen Beteiligten die Einhaltung dieses Standards gefordert. Dieser erhebt nicht nur den Anspruch an eine Vielzahl von entwicklungsbegleitenden Nachweisdokumenten, sondern fordert unter anderem die Durchführung von Sicherheitsevaluationen, welche zu unterschiedlichen Designphasen angewandt werden müssen. Da die Auflagen verpflichtend sind, rückt der Aspekt der funktionalen Sicherheit in einen neuen Fokus. Im zeitgetriebenen Alltagsgeschäft vieler Beteiligter bei gleichzeitiger Einhaltung von Kostengrenzen ist das Etablieren und Stärken einer Sicherheitskultur eine große Herausforderung.

1.1 Motivation

Mit der Einführung der ISO 26262 „*Road vehicles - Functional safety*“ konnte der Automobilindustrie ein domänenspezifischer Standard zur Umsetzung von funktionaler Sicherheit über den gesamten Produktlebenszyklus zur Verfügung gestellt werden. Dies war ein bedeutendes Ereignis aus Sicht der Automobilindustrie, da nun Vorschriften für die Entwicklung vorliegen, auf welche man sich zudem berufen kann. Der Standard wird innerhalb der Automobilindustrie bereits gelebt, bietet allerdings hinsichtlich Verständnis bei den Prozessbeteiligten sowie der Unterstützung im Bereich von Entwicklungswerkzeugen noch große Potentiale, um sowohl Zeit, Aufwand als auch Ressourcen im alltäglichen Geschäft einzusparen. Zudem sind für die Komplexitätsbeherrschung zukünftiger Systeme klar definierte Prozesse unumgänglich.

Die von ISO 26262 behandelten Aspekte sind nicht nur umfangreich sondern auch vielfältig, da diese von Variantenbildung über definierte Schnittstellen zwischen beteiligten Firmen im Zuge der Kollaboration sämtliche Merkmale der Konzept- und Entwicklungsphase bis hin zur Produktion und dem Betrieb des Fahrzeuges aufgreifen. Dies wirkt sich somit nicht nur auf Entwickler, sondern auf alle Prozessbeteiligten aus, für welche die von ihnen zu berücksichtigenden normativen Vorgaben und sämtliche zu erbringenden Nachweisdokumente teilweise nur schwer überschaubar sind. Bereits die Aufschlüsselung der einzelnen geforderten Aktivitäten ist derartig umfangreich, dass der Inhalt des Standards in einer aufbereiteten Version den Beteiligten zur Verfügung gestellt werden muss. Somit muss sowohl eine gemeinsame Daten- als auch Wissensbasis bereitgestellt werden, um die im Standard festgehaltenen Prozesse zu verstehen, umzusetzen und nicht zuletzt als Firmenkultur zu leben. Am verständlichsten für kollaborative Entwicklungen mit unterschiedlichen Fachbereichen sind grafische Prozessbeschreibungen, welche zur Diskussion, zum Austausch und zum Management der Prozesse verwendet werden können.

Eines der umfangreichsten Schwerpunktthemen innerhalb der ISO 26262 ist die Produktentwicklung funktional sicherer Hardware. Hierbei gilt es durch eine iterative Vorgehensweise sowie einer frühzeitigen Evaluation und Optimierung von bereits vorläufigen Designs, die Auswirkungen von zufälligen Ausfällen der Hardware auf ein akzeptables Maß zu reduzieren. Angelehnt wird teilweise an altbewährte Methoden wie FTA und FMEA, welche jedoch in den Fokus der EEA-Komplexität und der aktuellen Entwicklungsvorgehen gesetzt werden müssen. Dazu muss zum einen die Beschreibung der Designs auf unterschiedlichen Abstraktionsebenen ermöglicht werden. Zum anderen, insbesondere unter dem Gesichtspunkt der funktionalen Sicherheit, sind sowohl frühzeitige Analysen als auch Evaluationen auf diesen Abstraktionsebenen zu unterschiedlichen Entwicklungsständen zwingend erforderlich. Somit können durch den Einsatz von gezielten Maßnahmen die geforderten Zielwerte erreicht werden, um dem Standard zu genügen. Diese auf den Designs aufsetzenden und geforderten Sicherheitsevaluationen sind maßgeblich entscheidend bei der Neu- oder Weiterentwicklung einer automobilen EEA,

deren Subsystemen oder Komponenten. Hierfür gilt es, eine integrierte werkzeugbasierte Beschreibung von in sich konsistenten *Hardwaredesigns* sowie ausführbare Sicherheitsevaluationen zu bieten, um den Anforderungen aus der ISO 26262 gerecht zu werden.

Durch den Standard bedingt sind, sowohl aus Nachweispflicht als auch für eine mögliche Zertifizierung, entsprechende Dokumente in Form von beispielsweise Spezifikationen und Berichten gefordert. Bezogen auf die Produktentwicklung von Hardware umfasst dies insbesondere die Designbeschreibungen sowie die Ergebnisse der Sicherheitsevaluationen. Durch einen Ansatz, welcher sowohl die Beschreibung der Designs als auch die Ausführung der Sicherheitsevaluationen ermöglicht, kann dies in sich konsistent bereitgestellt werden. Durch Akquise der entsprechenden Daten und Ergebnisse können somit die jeweiligen Reports zu jedem Entwicklungsstand zusammengestellt und generiert werden, was eine erhebliche Arbeitserleichterung und Fehlervermeidung bei der Entwicklung von funktional sicherer Hardware darstellt.

Um die Vielschichtigkeit zukünftiger komplexer und sicherheitsbezogener Systeme im Automobilbereich zu beherrschen, besteht einer der wesentlichen Bausteine aus verbesserten Methoden und Werkzeugen. Die Methodiken zur Absicherung funktionaler Sicherheit können insbesondere aufbereitet und werkzeuggestützt bereitgestellt werden, um in etablierten Umgebungen die Sicht der funktionalen Sicherheit als eine weitere Perspektive auf die Systeme einzubringen. Hierdurch wird eine altbewährte Sichtweise erweitert und ein optimiertes Entwicklungsvorgehen zur Verfügung gestellt, bei gleichzeitiger Verbesserung der Entwicklung über die kommenden Serien hinweg. Die modellbasierte Architekturentwicklung hat sich insbesondere in der Automobilindustrie etabliert und gewinnt stetig an Bedeutung, da durch Formalisierung eine konsistente Beschreibung von EEAs und deren Subkomponenten ermöglicht wird. Eine integrierte Beschreibung von Prozessen ist eine der geeignetsten Methoden, um relevante Abläufe sowie Schnittstellen zu definieren. Zudem bieten derartige Lösungen, sowohl für allgemeine Vorgehensweisen als auch für sicherheitsbezogene Entwicklungen nach ISO 26262, in diesen Werkzeugumgebungen die notwendige thematische Nähe für die Prozessbeteiligten. Weiterhin kann das Potential solcher Entwicklungsumgebungen genutzt werden, um diese hinsichtlich der Entwicklung von funktional sicherer Hardware zu erweitern und die durch den Standard geforderten Sicherheitsevaluationen zu unterstützen. Diese erweiterten Perspektiven ermöglichen eine zügige iterative Entwicklung bei gleichzeitiger Absicherung der funktionalen Sicherheit.

1.2 Zielsetzung und eigener Beitrag

Um eine bewährte Sicht auf sicherheitsbezogene EEAs, deren Subsysteme und Komponenten für insbesondere die Konzept- und frühe Entwicklungsphase zu bieten, muss ein modellbasierter und kollaborativer Ansatz gewählt werden. Diesen gilt es derartig zu erweitern, dass sowohl eine Beschreibung von Prozessen sowie deren Management als

auch eine Unterstützung hinsichtlich geforderter Sicherheitsevaluationen im Zuge der Produktentwicklung auf Hardwareebene geboten werden kann.

Zielsetzung im Rahmen dieser Arbeit war, die hinter der ISO 26262 stehenden Prozesse, welche miteinander hochgradig vernetzt sind, den Beteiligten in aufbereiteter Form zur Verfügung zu stellen. Zudem sollte darauf aufsetzend ein Prozessmanagement geboten werden, um den Prozessbeteiligten eine Analyse, Planung sowie Durchführung der Prozesse zu erleichtern, damit die geforderten Aktivitäten und Arbeitsprodukte zielgerichtet abgearbeitet werden können. Hierzu konnte die etablierte und weiter an Bedeutung gewinnende Prozessmodellierungssprache Business Process Model and Notation (BPMN) zunutze gemacht werden, um durch diese eine Prozessbeschreibung innerhalb von Architekturentwicklungsumgebungen bereitzustellen. Eine Analyse der aus BPMN relevanten Konstrukte wurde durchgeführt und an den Entwicklungskontext von EEAs in modellbasierten Werkzeugumgebungen angepasst, um eine weitere, zu den bestehenden Beschreibungen orthogonale, Ebene aufzuspannen. Gleichzeitig konnten Erweiterungen zur Anreicherung der Prozessbeschreibung hinsichtlich der Entwicklung von sicherheitsbezogenen Systemen im Kontext der ISO 26262 eingebracht werden. Die Anwendung der Prozessmodellierung auf ISO 26262 und das daraus hervorgehende Prozessmodell zeichnet sich insbesondere durch die gesamtheitliche Abbildung der geforderten und verankerten Prozesse hinsichtlich funktionaler Sicherheit ab.

Aufbauend auf den erarbeiteten Prozessbeschreibungen galt es Potentiale zu identifizieren, welche eine weitere Unterstützung durch modellbasierte Ansätze für die Umsetzung der ISO 26262 aufbringen. Die Analyse der ISO 26262-Prozessbeschreibung konnte durch eine vorgeschlagene Metamodell-Erweiterung und Implementierung in einer domänenspezifischen Entwicklungsumgebung durchgeführt werden. Hierbei ergab sich die mit ihren Aktivitäten und Arbeitsprodukten umfangreiche Produktentwicklung auf Hardwareebene als einer der wichtigsten Teile, welche zwingend sowohl durch Formalisierung und ausführbare Sicherheitsevaluationen als auch Generierung von Nachweisdokumenten unterstützt werden muss.

Für die Entwicklung von sicherheitsbezogener Hardware sollte eine Umgebung geschaffen werden, welche die Beschreibung auf unterschiedlichen Abstraktionsebenen ermöglicht und eine iterative Überarbeitung der Designs, basierend auf Analyse und Optimierung im Zuge der Verbesserung des Designs oder Einführung von Sicherheitsmechanismen, zulässt. Durch eine einschlägige Analyse insbesondere von Part 5 und Part 10 der ISO 26262 konnten vorgeschriebene Evaluationen herausgearbeitet und im Zuge einer abstraktionsebenenübergreifenden Anwendung spezifische Vorgehensweisen erarbeitet werden. Dies wurde durch einen modellbasierten Ansatz umgesetzt, um somit wiederum die Bezüge zu den geforderten Prozessen herzustellen. Hierzu wurde eine Metamodell-Erweiterung für die Hardware-Fehlerbeschreibung erarbeitet und die Konzepte und Methodiken zur Ausführung der Sicherheitsevaluationen in eine Entwicklungsumgebung integriert, um eine nahtlose Untersuchung zu ermöglichen.

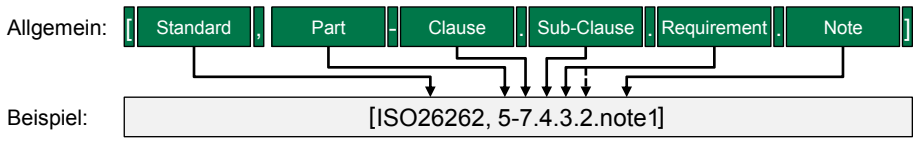
Um den vollzogenen Entwicklungsprozess für funktional sichere Hardware zu erfassen und die einzelnen getroffenen Maßnahmen festzuhalten, galt es eine Basis zur Dokumentation zu schaffen, welche nicht nur während der Entwicklung, sondern auch für eine nach der Finalisierung der Hardware gegebenenfalls anstehende Zertifizierung verwendet werden kann. Im Zuge der Nachweispflicht wurde eine Vorlage zur Dokumentation des *Hardwaredesigns* und dessen relevanten Informationen erarbeitet. Hierzu zählen unter anderem die dazugehörigen Sicherheitsziele, die eingebrachten Sicherheitsmechanismen sowie die Ergebnisse der Sicherheitsevaluationen. Somit wurde für die iterative Vorgehensweise innerhalb der Entwicklungsumgebung ein Umfeld geschaffen, welches zu jedem Entwicklungsstand die Dokumentation über alle geforderten Sicherheitsziele ermöglicht.

1.3 Anmerkungen

Terminologie: Englischsprachige Fachbegriffe, welche sich im deutschen Sprachgebrauch etabliert haben, werden unverändert beibehalten, um diese inhaltlich nicht zu verändern und Verwechslungen zu vermeiden. Dies trifft insbesondere auf Fachausdrücke aus den Standards ISO 26262 und ISO/IEC 19510 zu, um einen eindeutigen Bezug sicherzustellen.

Abkürzungen: Abkürzungen werden je nach Herkunft auf Deutsch oder Englisch verwendet. Bei erstmaliger Nutzung werden sie durch ihre ausgeschriebene Form eingeführt und ggf. eine Übersetzung sowie das dazugehörige Kürzel mit angegeben.

Referenzen auf Standards: Um Referenzen in Bezug auf verwendete Standards für den Leser zu vereinfachen, wird folgende Notation festgelegt:



Referenzen auf Abbildungen, Tabellen und Anhänge innerhalb eines Standards werden nach dem Verweis auf den Part entsprechend angegeben. Als Kennzeichnung werden Fig., Tab., Annex verwendet. Innerhalb von Kapiteln, welche im Kontext eines speziellen Standards stehen, werden zudem direkte Verweise, wie zum Beispiel Part 5 Clause 7, eingebracht. Die in dieser Arbeit aufgeführten Konzepte, Methodiken und Implementierungen basieren auf kontextbezogenen Interpretationen zur Anwendung des Standards, es sei daher immer auf die ursprünglichen Textpassagen innerhalb des Standards verwiesen.

Toolversionen: Die dargestellten prototypischen Implementierungen und Modelle wurden über den Zeitraum der wissenschaftlichen Arbeit auf unterschiedlichen Toolversionen umgesetzt und teilweise auf aktuellere Toolversionen migriert bzw. portiert. Dies trifft unter anderem auf die verwendete Entwicklungsumgebung PREEvision zu, in welcher zahlreiche Ansätze und Methodiken umgesetzt, verifiziert und validiert wurden.

Materialien: Einige im Rahmen dieser Arbeit vorgestellte und verwendete Materialien, wie beispielsweise Abbildungen, wurden bereits als eigene Beiträge in Projektberichten und Deliverables sowie eigenen Veröffentlichungen publiziert. Hierzu zählen unter anderem technische Paper, Journal-Artikel, Fachzeitschriften-Artikel, Poster und Präsentationen. Diese repräsentieren eigene Originalbeiträge und sind unter dem Verzeichnis der Quellennachweise aufgelistet.

1.4 Struktur der Arbeit

In Abb. 1.1 ist die Struktur der vorliegenden Dissertation dargestellt. Im folgenden Kapitel 2 wird einleitend auf die Fahrzeugentwicklung eingegangen und bewährte Vorgehensweisen dargestellt. Dies wird in Bezug zur modellbasierten Architekturentwicklung gesetzt und dafür verfügbare Architekturbeschreibungssprachen aufgegriffen. Da es sich bei ISO 26262 um ein prozessorientiertes Dokument handelt und hochgradig miteinander vernetzte Aktivitäten sowie Arbeitsprodukte enthalten sind, werden Prozessmodellierungssprachen vorgestellt. Eine nachfolgende Beschreibung der ISO 26262 zeigt sowohl strukturelle als auch inhaltliche Aspekte des Standards auf und vertieft diese im Bereich der Hardwareentwicklung.

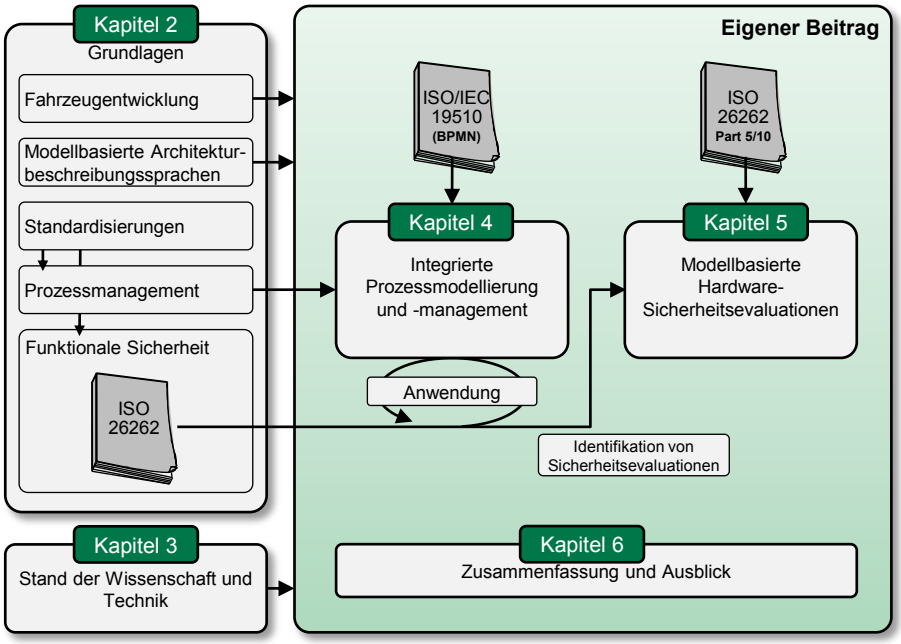


Abb. 1.1: Struktur der Dissertation

Das Kapitel 3 zum Stand der Wissenschaft und Technik ist zweiteilig aufgebaut. Im ersten Teil wird auf aktuelle Forschungstätigkeiten und Werkzeuge im Bereich der Prozessmodellierung und des -managements eingegangen. Hier steht unter anderem die Unterstützung von Prozessen im Kontext von funktionaler Sicherheit im Fokus. Der zweite Teil beleuchtet den aktuellen Stand hinsichtlich der Produktentwicklung auf Hardwareebene im Kontext der ISO 26262. Hierbei wird insbesondere die Unterstützung von Hardware-Sicherheitsevaluationen aufgegriffen.

Kapitel 4 stellt ein Konzept für eine integrierte Prozessmodellierung einschließlich der Beschreibung von Organisationsstrukturen und Zuweisung von Prozessressourcen bereit. Ein darauf aufsetzendes Prozessmanagement bezieht die Prozessanalyse und Prozessplanung ein. Weiterhin wird ein erster Bezug zwischen Prozessmodell, Organisationsstrukturen und Architekturmodell bereitgestellt. Durch eine Integration der Konzepte in eine Architekturbeschreibungssprache kann die Methodik auf den umfangreichen Standard ISO 26262 angewandt und für die geforderten Aktivitäten und Arbeitsprodukte im Bereich der Produktentwicklung auf Hardware vertieft werden.

Kapitel 5 stellt ein Konzept und eine Methodik für die modellbasierte Entwicklung funktional sicherer Hardware vor. Hierfür wird zunächst ein Metamodell-Vorschlag für Hardware-Fehlerinformationen im Zuge der Erweiterung von Architekturbeschreibungssprachen bereitgestellt und die Anwendbarkeit der nach ISO 26262 geforderten Hardware-Sicherheitsevaluationen auf unterschiedlichen Abstraktionsebenen erarbeitet. Darauf basierend konnte eine Vorgehensweise zur abstraktionsebenenübergreifenden Beschreibung von Hardwaredesigns einschließlich der Anreicherung um Fehlerinformationen und der Durchführung von geforderten Sicherheitsevaluationen abgeleitet werden. Zudem sind unter anderem Aspekte einer verteilten Entwicklung und etablierte Vorgehensweisen im Fokus. Durch die Integration in eine Werkzeugumgebung kann anhand eines Beispiels für eine Ventilregelung die durchgängige Entwicklung funktional sicherer Hardware einschließlich der Evaluationen gezeigt werden.

Das abschließende Kapitel 6 fasst die Inhalte und Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf mögliche weiterführende Arbeiten basierend auf den erzielten Erkenntnissen.

2 Grundlagen

Die Automobilindustrie hat über Jahrzehnte bewährte Methoden entwickelt, um den Anforderungen der Märkte und dem Konkurrenzdruck gerecht zu werden. Im folgenden Kapitel wird auf den automobilen Entwicklungsprozess eingegangen. Darauf aufbauend werden Vorgehensweisen und Prozessbewertungsmodelle aus dem Automobilbereich vorgestellt. Nachfolgend werden modellbasierte Architekturbeschreibungssprachen für die Automobilindustrie aufgezeigt und ein Einblick in den Entwicklungsprozess von Standards sowie in das Management von Prozessen gegeben. Hinsichtlich funktionaler Sicherheit von Straßenfahrzeugen wird der Standard ISO 26262 aufgegriffen und in Bezug auf die Produktentwicklung von Hardware vertieft.

2.1 Fahrzeugentwicklung und Produktlebenszyklus

In der Einleitung wurde bereits auf die Veränderung der Elektrik-/Elektronik (EE) in Fahrzeugen aus verschiedenen Gründen eingegangen. Um zusätzliche Funktionalitäten bewältigen zu können, weisen Fahrzeuge einen immer größeren Anteil an EE-Komponenten auf. Exemplarisch sei hier die Anzahl der Steuergeräte, im Englischen Electronic Control Units (ECUs), genannt, die beispielsweise je nach Fahrzeugklasse bis zu 80 oder mehr ergibt, vgl. [6, S.43], [126, S.20], [127, S.644]. So sind in einem BMW der 5er oder 7er Baureihe ca. 70 ECUs [7] verbaut. Die Steuergeräte sind innerhalb eines Fahrzeugs durch verschiedene Bussysteme, wie unter anderem Local Interconnect Network (LIN), Controller Area Network (CAN), Media Oriented Systems Transport (MOST), FlexRay oder Ethernet miteinander verbunden, vgl. [127, S.644]. Diese Entwicklung wirkt sich unter anderem auch auf die zunehmende Leitungslänge der verbauten Kabel sowie die höhere Anzahl an Steckverbindern aus. Die Teilezahl für Einzelleitungen eines unteren Mittelklassenfahrzeugs beläuft sich auf einen Wert von ca. 600 nach [127, S.654], was bezogen auf die Gesamtheit von 11.000 der Elektrik zugeordneten Teilen die Dimensionen vor Augen führt.

Die Darstellung eines Bordnetzes mit den elektronischen Komponenten für einen VW Phaeton ist in Abb. 2.1 nach [127, S.654] zu finden. Unter dem Begriff des Bordnetzes werden alle „zur Verbindung und Versorgung der elektrischen Bauteile eines Fahrzeugs erforderlichen Komponenten, einschließlich der zu ihrer Integration (Befestigung, Einbau) ins Fahrzeug nötigen nichtelektrischen Bauteile“ [127, S.654] verstanden. Der Beitrag der EE-Komponenten solcher Fahrzeuge der Mittel-/Oberklasse ist dabei derartig

groß, dass durch Weglassen rein mechanischer Teile, wie beispielsweise der Karosserie, der Fahrzeughersteller bestimmt werden kann.

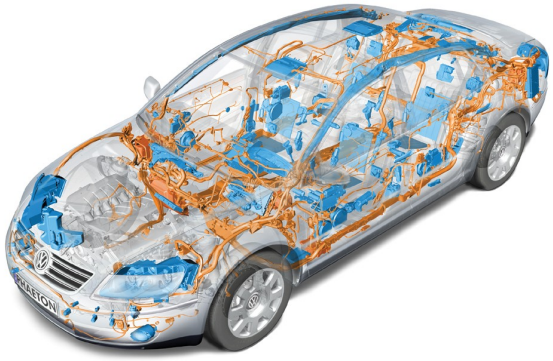


Abb. 2.1: Darstellung eines Bordnetzes in [127, S.654] - „Elektrikkomponenten (blau) und Bordnetz (braun) eines Oberklasse-Fahrzeuges“

Um solche komplexen Elektrik-/Elektronik-Architekturen (EEAs) zu überschauen und entwickeln zu können, wird nach einem Produktentstehungsprozess vorgegangen, welcher im Folgenden vereinfacht dargestellt wird und auf die Entwicklung der EE-Anteile ebenfalls zutrifft.

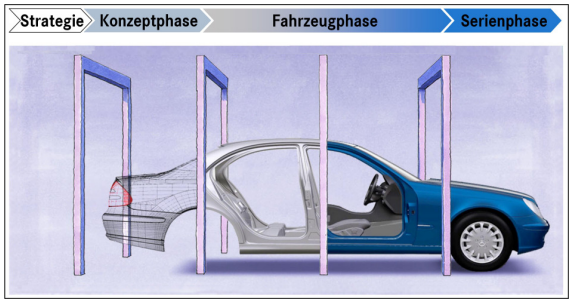


Abb. 2.2: „Fahrzeugentwicklung: Strategie- und Konzeptphase, Fahrzeug- und Serienphase“ in [5, S.513]

In [128, S.7] wird angemerkt, dass der Produktentstehungsprozess einen unterschiedlichen Verlauf und Dauer im Kontext verschiedener Automobilhersteller, im Englischen Original Equipment Manufacturers (OEMs), haben kann, jedoch zeichnen sich verbreitete Vorgehensweisen und strategische Überlegungen ab. Eine vereinfachte Darstellung

nach [128, S.8] setzt Entwicklungsphasen verschiedener Baureihen versetzt zueinander in Bezug und beschreibt die Produktentwicklung durch eine Aufteilung in Produktstrategie, Technologieentwicklung, Fahrzeugentwicklung und Produktions- sowie Verkaufsstart.

In Abb. 2.2 nach [5, S.513] sind die einzelnen Phasen der Produktentwicklung bis zur Serienproduktion sehr anschaulich aufbereitet. Der Entwicklungsprozess von Automobilen durchläuft die gleichen Phasen wie Produktentwicklungen aus anderen Domänen, vgl. [125, S.309f]. Die Produktentwicklung ist Teil des Produktlebenszyklus, dessen Zeitschiene vereinfacht aus folgenden Phasen besteht:

Strategiephase: Innerhalb der Strategiephase, alternativ auch als Phase der Produktstrategie bezeichnet, wird auf Managementebene, basierend auf Studien, Marktumfragen, etc. festgelegt, welche potenziellen Zielgruppen durch ein Produkt angesprochen werden könnten und ob hierfür ein „Business Case“ vorhanden ist, der eine Wirtschaftlichkeit bestätigt. Ist dies der Fall, können kundenerlebbare Funktionen definiert werden, gegebenenfalls durch Ableitung funktionaler Innovationen, angelehnt an [128, S.7].

Konzeptphase: Während der Konzeptphase, nach [128, S.7] auch als Technologieentwicklung bezeichnet, wird unter anderem die EEA abgeleitet. Hierbei wird meist auf einem basierenden Architekturkonzept aufgesetzt oder ein Plattform-Konzept neu entworfen. Von Bedeutung bei der Auswahl oder dem Aufbau des Architekturkonzeptes ist, dass die in der Strategiephase definierten funktionalen Anforderungen und Kundenwünsche aufgespannt werden können [128, S.7]. Um Innovationen oder neue Technologien einzubringen wird meist eine Baureihe aus dem hochpreisigen Fahrzeugsegment ausgewählt, um nicht unbedingt durch die Stückzahlen das Risiko zu verringern, wohl aber die Gesamtkosten im Falle von möglichen Rückrufaktionen [128, S.7].

Fahrzeugphase: Die Fahrzeugphase, auch als Entwicklungsphase oder nach [128, S.8] als Phase der Fahrzeugentwicklung bezeichnet, repräsentiert den Aufbau erster Prototypen bis hin zur Nullserie. Bei den Prototypen-Fahrzeugen stehen die Integration und das Zusammenspiel einzelner Komponenten sowie die ersten Applikationen des Gesamtsystems im Vordergrund. Bereits in der Fahrzeugentwicklungsphase werden die ersten Schritte hinsichtlich der Serienproduktion unternommen.

Serienphase: Die Serienphase arbeitet auf die Produktion und damit auf die Serienfertigung hin. Hierbei gilt es unter anderem sämtliche in das Fahrzeug eingebrachte Systeme und Funktionalitäten zu verifizieren und validieren und das Fahrzeug zur Produktreife zu bringen, um den Start Of Production (SOP) herbeizuführen und den Verkauf zu starten. Meist wird diese Phase bereits parallel aber zeitlich versetzt zur Fahrzeugphase begonnen, um die Werkzeugentwicklung voranzutreiben. Dies betrifft unter anderem auch die Vorbereitungen für Arbeitsabläufe der darauffolgenden Produktion im Sinne der Fertigungskapazitäten, vgl. [128, S.8].

Weitere Phasen des Produktlebenszyklus: Darüber hinaus sind im Rahmen des Produktlebenszyklus sowohl die Phasen der Produktion oder Serienfertigung, des Betriebs und

Services (Feld) als auch die Außerbetriebnahme im Sinne der Dekommissionierung von Bedeutung, vgl. den automobilen Sicherheitslebenszyklus nach [ISO26262, 1-V].

Nach [128, S.8] ist der Zeitraum zwischen Produktstrategie bis zum Verkaufsstart im Kontext einer neuen Fahrzeugbaureihe in einer typischen Größenordnung von 50 Monaten anzusiedeln. In [129, S.20] werden folgende Zeiträume eines automobilen Produktlebenszyklus angegeben: Die Entwicklung wird auf 3 Jahre eingestuft, der Produktionszeitraum beträgt 7 Jahre und für den Betrieb und Service wird eine Zeitspanne von 10-15 Jahren genannt.

Im Rahmen des Entwicklungsprozesses sind zahlreiche unterschiedliche Firmen beteiligt. Vereinfacht bildet der OEM das Dach für eine darunterliegende Zulieferpyramide, welche sich nach [130, S.28] basierend auf [131] in die folgenden Lieferstufen einteilen lässt: Kernlieferanten, die Systeme oder Module an den OEM liefern. Diese repräsentieren die sogenannte Tier 1. Die Komponentenlieferanten als Tier 2 stellen Bauteile oder Untergruppen dem Tier 1 oder direkt dem OEM zur Verfügung. Auf unterster Ebene sind die Rohmaterial-, Halbleiter- oder Normteilefabrikanten anzusiedeln, welche als Tier 3 die übergeordneten Komponentenlieferanten beliefern. Wird diese Pyramide nach den einzelnen Funktionen betrachtet, so zieht sich vom Fundament bis zur Spitze, der Volumenanbieter über den Spezialisten hin zum Systemintegrator. Sonderkonstellationen, welche sich über mehrere Ebenen erstrecken, sind dabei nicht ausgeschlossen. Jede einzelne Firma kann wiederum in Bereiche und Abteilungen partitioniert sein. Auf unterster Ebene können einzelne Rollen aufgeführt werden. Diese repräsentieren Mitarbeiter, welche bestimmte Verantwortlichkeiten und Zuständigkeiten innehaben. Inwiefern es sich bei ihnen um Prozessbeteiligte handelt, ist vom jeweiligen Projekt abhängig.

Funktionalität und Kosten im Kontext der Entwicklungszeit sind Hauptkriterien, welche dauerhaft während der automobilen Entwicklungsprozesse verfolgt werden. Um das Delta der Kostenentwicklung nicht nur über den Produktentwicklungszyklus sondern auch über den Produktlebenszyklus möglichst gering zu halten, müssen bereits zu sehr frühen Entwurfsphasen Methoden bereitgestellt werden, welche eine Bewertung und Absicherung der EEA ermöglichen. Dies tangiert auch die Gewährleistung von funktionaler Sicherheit, um gegebenenfalls Maßnahmen für ein gezieltes Einlenken treffen zu können. Daher ist für solche Produktentwicklungen die Konzeptphase eine der richtungsweisenden Phasen, da sie weitgreifende Auswirkungen auf den Verlauf hat und wegweisende Entscheidungen getroffen werden.

2.2 Automobile Vorgehensweisen und Prozessbewertungsmodelle

Der Entwicklungsprozess sollte sich nach [129, S.19] an bewährten Prinzipien orientieren. Zu den Vorgehensweisen zählt hierzu beispielsweise das Wasserfallmodell, das Spiralmodell sowie das V-Modell. Als Prozessbewertungsmodelle können CMMI und SPICE gelistet werden.

2.2.1 Wasserfallmodell

Im Beitrag „Managing the development of large software systems“ [8, S.338] wurde von Royce das spätere Wasserfallmodell geprägt. Bei [9, S.63] wird es als Nachfolger des „stagewise“ Modells angeführt und als ein häufig verwendetes Modell in der Softwareentwicklung genannt, da diese grundlegend aus zwei Stufen aufgebaut ist, der Analyse und Programmierung, welche durch Zusatzschritte sukzessive miteinander verbunden werden. Das Vorgehen ist dadurch in aufeinanderfolgende Phasen gegliedert, die bildhaft wasserfallartig angeordnet sind [8, S.329]. Jede Phase ist in sich abgeschlossen und liefert ein Ergebnis, welches als Ausgangsdokument für die folgende Phase dient, d.h. eine neue Phase kann nur beginnen, wenn die Vorausgegangene abgeschlossen ist. Von dieser ursprünglichen Version entwickelte sich eine erweiterte Form des Wasserfallmodells, die aufeinanderfolgenden Phasen eine Iteration bietet, wie bereits bei [8, S.330] aufgezeigt wurde. So können vor Prozessende Fehler behoben und Änderungen vorgenommen werden, die den weiteren Verlauf beeinflussen. Boehm bezeichnet dies als „Feedback loop“ [9, S.63]. Nach Ross [132, S.32] wird im Vergleich zum V-Modell durch das Wasserfallmodell eine höhere Abstraktionsebene beschrieben. Zugleich wird angemerkt, dass eine automobilspezifische Ableitung vermutlich Ähnlichkeiten zur ISO 26262 aufweisen würde.

2.2.2 Spiralmodell

Das Spiralmodell geht aus dem Beitrag von Boehm [9, S.64] ebenfalls als Vorgehensmodell für die Softwareentwicklung hervor. Unter anderem konnten Erfahrungen aus der Anwendung des Wasserfallmodells einfließen. In [132, S.32] wird es im Kontext der Automobildomäne als traditionelles Vorgehensmodell gesetzt. Die Namensgebung des Modells ergibt sich aus den sequentiellen Abläufen und den Iterationen, welche eine Spirale bilden. In der radialen Dimension werden die Kosten, die bis zum erreichten Schritt entstanden sind, abgebildet. Die Winkelmaße stellen den Fortschritt innerhalb eines Spiralkreises dar [9, S.65]. Jeder Kreis bahnt damit die nächste Ebene der Ausführung der zukünftigen Ziele, Einschränkungen und Alternativen des Systems an [10, S.59]. Im Zusammenhang mit dem Spiralmodell stehen auch die Musternamen des A-, B-, C- und D-Musters, welche bei den Automobilfirmen Verwendung finden. Nach [132,

S.33f] werden folgende Phasen genannt: „Prototypen Spezifikation“, „Prototypenbau“, „Experimentieren Test / Akzeptanz“ und „Änderung oder Ergänzungen“.

2.2.3 V-Modell 97 und V-Modell XT

Das Vorgehensmodell V-Modell XT[1] [VModellXT] ist eine Weiterentwicklung des V-Modells 97 [VModell97] und unterstützt durch eine ziel- und ergebnisorientierte Vorgehensweise [VModellXT, S.1-21] bei der Planung und Durchführung von Projekten, unter Berücksichtigung des gesamten Systemlebenszyklus [VModellXT, S.1-6]. Seit 2005 ist es ein international anerkannter Entwicklungsstandard für IT-Systeme. Das Modell definiert genaue Prozessergebnisse und legt die dazu notwendigen Schritte fest [VModellXT, S.1-6].

Das von Boehm 1979 [11] erstmals veröffentlichte Modell erhielt seinen Namen aufgrund der Darstellung in Form eines „Vs“ [133, S.574]. Das Projekt ist in einzelne, in sich abgeschlossene Vorgehensbausteine aufgeteilt [VModellXT, S.1-11], in denen alle Elemente, die zur Bearbeitung einer Aufgabe notwendig sind (Produkte, Aktivitäten, Rollen), enthalten sind [VModellXT, S.1-15]. Das V-Diagramm stellt neben der detaillierten, linearen Vorgehensweise auch den zeitlichen Aspekt dar. Von links oben werden durch das Ab- und Aufsteigen der Äste die Abstraktionsebenen sowie der zeitliche Verlauf veranschaulicht [134, S.13], [133, S.574].

Die erste Prozessphase, nach [134, S.13] auch als Entwurfsprozess bezeichnet, wird durch den abfallenden Ast symbolisiert und spezifiziert dabei die Prozessschritte, von der Anforderungsanalyse zur Implementierung (Top-Down Prozess) [135, S.64]. Auf dem aufsteigenden Ast wird die zweite Prozessphase (Verifikationsprozess) [134, S.13] dargestellt, in der die Integration der Elemente sowie deren Verifikation abgebildet werden (Bottom-Up Prozess) [135, S.64]. Jeder Information, die im linken Ast entsteht, kann eine Verifikation auf der rechten Seite zugeordnet werden [135, S.67],[128, S.68]. Der Anforderungsanalyse zu Prozessbeginn steht die Validierung zu Prozessende gegenüber [135, S.64], [128, S.67].

Im Beitrag [12, S.75] werden die Termini Verifikation und Validierung informell mit Fragen beschrieben. So wird für die Verifikation die Frage gestellt, ob das Produkt richtig gebaut wird und für die Validierung, ob das richtige Produkt gebaut wird. Somit ist die Validierung innerhalb des V-Modells auf oberster Ebene anzusiedeln, die Verifikation entsprechend darunter.

Das V-Modell gilt als Referenzmodell [132, S.25] für die im Folgenden beschriebenen Prozessbewertungsmodelle. Capability Maturity Model Integration (CMMI) sowie Software Process Improvement and Capability Determination (SPICE) zielen auf die Verbesserung und Bewertung von Prozessen ab.

[1] http://www.v-modell.iabg.de (Zugriff am 15.10.2014)

2.2.4 CMMI

CMMI[2] stellt Referenzmodelle zur Verfügung und basiert auf einer langen Historie, unter anderem auf SE-CMM [SECMM95] und EIA-731.1 [EIA731], [13]. Sie stellen eine Sammlung zur Prozessverbesserung dar. Alle CMMI-Modelle basieren auf dem CMMI-Framework mit Bestandteilen, welche in allen Modellen verwendbar sind (CMMI Model Foundation (CMF)). Jedes Modell wird durch eine Konstellation spezifischer Elemente ausgezeichnet und je nach Schwerpunkt mit einem Zusatz in der Namensgebung gekennzeichnet, vgl. hierzu die Dokumente [CMMIacq, S.8], [CMMIdev, S.7], [CMMIsvc, S.7].

Unterschieden wird in die drei veröffentlichten Modelle, welche als Technical Reports (TR) niedergeschrieben sind:

- CMMI® for Acquisition (CMMI-ACQ), Version 1.3 [CMMIacq] mit dem Untertitel „Improving processes for acquiring better products and services“
- CMMI® for Development (CMMI-DEV), Version 1.3 [CMMIdev] mit dem Untertitel „Improving processes for developing better products and services“
- CMMI® for Services (CMMI-SVC), Version 1.3 [CMMIsvc] mit dem Untertitel „Improving processes for providing better services“

Grundlegend sind die Dokumente durch ihren strukturellen Aufbau ähnlich, konzentrieren sich jedoch auf unterschiedliche Aspekte. Alle drei Dokumente und damit CMMI-Modelle setzen sich aus 16 Kernprozessen zusammen, welche Bereiche wie Prozess- und Projektmanagement sowie Unterstützungsprozesse abdecken. Zudem teilt jedes Modell einen weiteren Prozessbereich und enthält beim CMMI-ACQ und CMMI-DEV-Modell 5 bzw. beim CMMI-SCV-Modell 7 zusätzliche spezifische Prozessbereiche, vgl. [CMMIdev, S.3], [CMMIsvc, S.3].

CMMI-ACQ zielt auf eine Prozessverbesserung hinsichtlich des Erwerbs von Zukaufteilen und Zuliefererdiensten ab, um den Anforderungen der Kunden und Endnutzern gerecht zu werden [CMMIacq, S.i]. Das Modell bietet eine Unterstützung für die Prozessverbesserung des Auftraggebers (acquirer), weniger für Zulieferer. An diese adressiert sich die im Folgenden beschriebene CMMI-DEV [CMMIacq, S.4].

CMMI-DEV unterstützt die Entwicklung von Qualitätsprodukten und -dienstleistungen, um Kunden und Endanwender zufriedenzustellen [CMMIdev, S.i]. Dazu umfasst die CMMI-DEV Praktiken, die den ganzen Produktlebenszyklus einschließen. Besonders im Fokus stehen hierbei „die Entwicklung und Instandhaltung des Endprodukts“ [CMMIdev, S.3]. Beim CMMI-DEV sind 5 Prozessbereiche entwicklungsspezifisch, dazu gehört unter anderem die Anforderungsentwicklung, technische Umsetzung, Verifikation sowie Validierung, vgl. [CMMIdev, S.3].

Das CMMI-SVC adressiert sich an Anbieter von Dienstleistungen, um diese bei der Verbesserung ihres Services zu unterstützen [CMMIsvc, S.i]. In den spezifischen

[2] http://cmmiinstitute.com (Zugriff am 15.10.2014)

Prozessbereichen befinden sich dazu Themengebiete wie Kapazitäts- und Lieferbarkeitsmanagement, Leistungskontinuität, Leistungserbringung, Lösungen und Vorbeugung von Zwischenfällen, Wechsel von Dienstleistungen, die Entwicklung des Dienstleistungssystems sowie die strategischen Dienstleistungsmanagement-Prozesse, vgl. [CMMIsvc, S.3].

CMMI dient der Bewertung von Organisationen auf dem Weg der Prozessverbesserung. Unter einer Organisation kann ein ganzes Unternehmen aber auch eine Gruppe innerhalb eines Unternehmens verstanden werden [CMMIdev, S.21]. Verschiedene Organisationen können somit durch den Bewertungsgrad miteinander verglichen werden.

Durch die Vergabe von Graden wird die Umsetzung der gesetzten Ziele bewertet [CMMIdev, S.22]. Diese werden durch sogenannte „Appraisals“ vergeben. Eine Verbesserung der Grade kann über zwei Möglichkeiten erzielt werden, zum einen über die Steigerung in einzelnen Prozessbereichen, zum anderen über die Verbesserung durch sukzessive Auswahl von zusammenhängenden Prozessgebieten [CMMIdev, S.21].

Unterschieden wird zwischen dem Fähigkeitsgrad (Capability Level) und dem Reifegrad (Maturity Level). Mit Hilfe des Fähigkeitsgrades wird der „Zustand der Prozesse einer Organisation auf einem einzelnen Prozessgebiet“, nummeriert durch die Grade 0-3, dargestellt. Der Reifegrad definiert „den Gesamtzustand der Prozesse einer Organisation“, gegeben durch die Reifegrade 0-5 [CMMIdev, S.22-23]. Während CMMI die Bewertung von Prozessen zusammenfasst und den Reifegrad eines Unternehmens bestimmt, bewertet SPICE die einzelnen Prozesse [132, S. 24], [SPICEpamdt, S.7]. Es werden ebenfalls sogenannte Reifegradstufen verwendet [SPICEpamdt, S.7].

2.2.5 SPICE und Automotive SPICE

SPICE ist als Standard unter ISO/IEC 15504 [ISO15504] mit insgesamt 10 Parts verankert, hierbei sind die ersten 6 Parts normativ als Standard und die Restlichen entweder als ergänzende Technical Report (TR) oder Technical Specification (TS) veröffentlicht. SPICE repräsentiert eine Alternative zu CMMI und dient einer Bewertung von Prozessen. Innerhalb Part 10, welcher eine Technical Specification repräsentiert, wird auf eine Sicherheitserweiterung eingegangen. Diese definiert weitere Prozesse für „Safety Management process“, „Safety Engineering process“ und „Safety Qualification process“ [ISO15504, 10-4] und dient hiermit als Richtlinie einer Entwicklung funktionaler und nicht-funktionaler komplexer sicherheitsbezogener Systeme.

Das Automotive SPICE[3] ging als domänenspezifisches SPICE aus einem Zusammenschluss einer automobilen Interessensgemeinschaft von OEMs hervor. Es gliedert sich grundlegend in die zwei Dokumente „Process Assessment Model“ [SPICEpam] und das dazugehörige „Process Reference Model“ [SPICEprm].

[3] http://www.automotivespice.com (Zugriff am 15.10.2014)

2.3 Modellbasierte Architekturbeschreibungssprachen

Insbesondere im Rahmen der Konzeptphase sowie der frühen Entwicklungsphase werden vermehrt Architekturbeschreibungssprachen, im Englischen Architecture Description Languages (ADLs), für die modellbasierte Entwicklung von EEAs eingesetzt. Diese ermöglichen durch ein dahinterliegendes Metamodell eine konsistente und formale Beschreibung und damit eine Abstraktion von komplexen Sachverhalten, siehe Abb. 2.3.

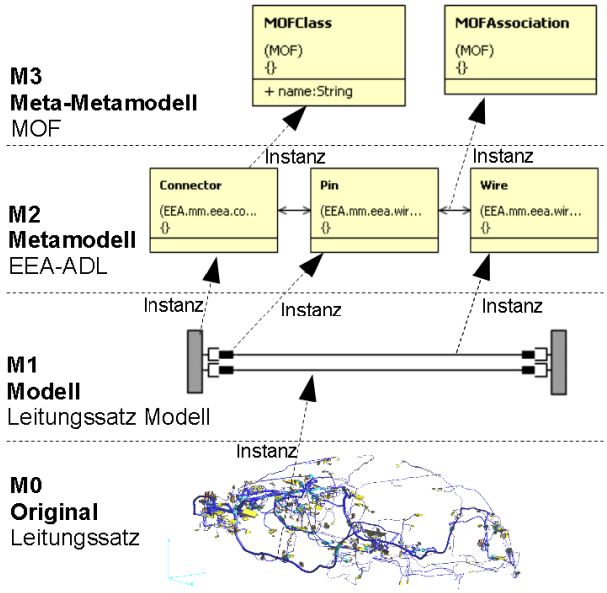


Abb. 2.3: „Beispiel Vier-Schichten-Modell - Leitungssatz“ in [122, S.16]

Im Folgenden wird auf die Beschreibungssprachen AADL, EAST-ADL und AUTOSAR sowie PREEvision eingegangen. Weitere Werkzeuge zur Beschreibung von EEAs wie Unified Modeling Language (UML) oder System Modeling Language (SysML) sind unter [122] beschrieben. Zudem werden in [122] die Grundlagen der Metamodellierung aufgegriffen.

2.3.1 AADL

Die Architecture Analysis and Design Language (AADL)[4], aktuelle Version V2.1, ermöglicht Architekturmodellierung und Analysen wie in [136, S.31] beschrieben.

[4]http://www.aadl.info (Zugriff am 15.10.2014)

Werkzeugumgebungen wie Open Source AADL Tool Environment (OSATE) 2[5] stellen diese zur Verfügung. Weitere Werkzeugumgebungen sind unter [14] gelistet.

Der Anfang der AADL wird auf das Jahr 1999 datiert, nachdem in den 90er-Jahren die Luft- und Raumfahrtdomäne erkannte, dass aufgrund der zunehmenden Bedeutung eingebetteter Systeme analysierbare Architekturentwicklungsmodelle benötigt wurden [136, S.10]. Die erste Version von AADL wurde von der SAE unter dem Aerospace-Standard AS5506 verankert und als internationaler Standard am 05.11.2004 herausgegeben, vgl. [15], [16]. Eine überarbeitete Version stellt AS5506A dar, publiziert am 20.01.2009 und mit Erweiterungen in den folgenden Jahren veröffentlicht [136, S.XVi, S.11]. Aktuelle Revision ist AS5506B vom 10.09.2012.

Bei der AADL handelt es sich um einen systematischen, architekturzentrierten Ansatz zur modellbasierten Entwicklung von Software-System-Architekturen [136, S.XV]. Als ein Rahmenwerk wird es verwendet, um die „statische modulare Software-Architektur“, die „Laufzeit-Architektur im Zusammenhang mit Kommunikationsaufgaben“, die „Architektur der Computerplattform, auf welcher die Software entwickelt wird“ und „sämtliche physikalische Umgebungen, mit denen das System in einer Interaktion steht“ zu erfassen [136, S.XVi]. Durch eine frühzeitige und kontinuierliche Analyse während des kompletten Lebenszyklus gilt es, Probleme der Systemebenen zu erkennen. Diese sind während der Integration- bzw. der Testphase nicht mehr ausfindig zu machen und verursachen höhere Kosten für die Systementwicklung und -erhaltung [136, S.XV].

AADL unterstützt durch Analysen das Systemdesign, die Systemintegration sowie die Systemsicherheit, indem sie die Software- und Hardwarekomponenten und deren Interaktionen abbildet [136, S.1]. Durch diese architekturzentrierte Analyse werden mehrere Dimensionen eines Systems betrachtet, wodurch ein lückenloseres Ergebnis zu erreichen ist, als bei analytischen Modellen, die auf Teilbereiche begrenzt sind [136, S.9].

Zusätzlich zum AADL Standard existieren 2 Volumes an Annexen [136, S.XVi]. 2006 wurde der Annex Volume 1 zum Standard SAE AS-5506/1 mit folgenden Inhalten veröffentlicht: Annex A: „Graphical AADL Notation“, Annex C: „AADL Meta-Model and Interchange Formats“, Annex D: „Language Compliance and Application Program Interface“, Annex E: „Error Model Annex“. Im Jahr 2011 wurde der Standard SAE AS-5506/2 als weiterer Annex Volume 2 definiert. Dieser umfasst folgende Inhalte: Annex B: „Data Modeling Annex“, Annex D: „Behavior Annex“ sowie den ARINC653 Annex.

Der „Behavior Annex“ erweitert den Kern der AADL, die Interaktionsverhalten der Komponenten genauer anzugeben, um damit die sicherheitskritischen Aspekte des Systems zu adressieren. Des Weiteren bietet der „Error Model Annex“ mehr Möglichkeiten, um Fehlerverhalten sowie Fehlerpropagation genauer zu hinterlegen, um damit den Zuverlässigkeitsaspekt des Systems miteinzubeziehen [136, S.11]. Der Annex E ergänzt den Hauptteil der AADL durch die Einführung spezifischer Fehlertypen, „error state machines“ sowie der Fehlerpropagation. Zusätzlich können Sicherheits- und Zuverlässig-

[5] https://wiki.sei.cmu.edu/aadl/index.php/Osate_2#Download (Zugriff am 15.10.2014)

keitsanalysen durchgeführt werden, wie zum Beispiel Functional Hazard Assessment (FHA), Failure Mode and Effects Analysis (FMEA) oder Mean Time To Failure (MTTF) Analysis [136, S.323].

In [17, S.15] wird auf Fehlermodelle im Kontext von Hardwarekomponenten eingegangen. Bei den Fehlermodellen handelt es sich um State Machines, welche mit AADL Komponenten oder Verbindungen verknüpft werden können [17, S.6]. Dies dient einer Beschreibung ihres Verhaltens im Zuge logischer Fehlerzustände. Unter anderem wird in permanente und transiente Fehler unterschieden und eine Anreicherung mit Wahrscheinlichkeiten vorgenommen.

Parallelen zwischen verschiedenen Architekturbeschreibungssprachen lassen sich in Bezug auf den Umgang mit der Komplexität der Software feststellen. Die Systemarchitektur wird jeweils in verschiedene Komponenten eingeteilt, wobei der Fokus je nach Sprache auf einem anderen System liegt. Während sich Simulink auf Kontrollsysteme bezieht, fokussiert sich SysML auf physikalische Systeme bzw. UML auf die konzeptionelle Architektur. Demgegenüber strukturiert AADL die Architektur eingebetteter Systeme [136, S.6].

Angemerkt sei, dass in [18] eine Erweiterung von SysML mit AADL-Konzepten vorgestellt wird. Basierend auf ausgewählten Aspekten unter anderem von AADL und SysML wurde der „Citrus“ Ansatz entwickelt und in [19] auf eine reale Cockpit-Applikation angewandt.

2.3.2 EAST-ADL und Eclipse-basierte Implementierung EATOP

Die Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL) ist eine Architekturbeschreibungssprache, welche ihre Anfänge im Projekt „Electronics Architecture and Software Technology - Embedded Electronic Architecture“ (EAST-EEA) nahm und als Erfolgsgeschichte hervorging [20, S.10]. Die aktuelle Version von EAST-ADL ist die Domain Model Specification Version 2.1.12 [21], welche unter anderem durch die Projekte MAENAD und TIMMO-2-USE getrieben wurde.

EAST-ADL wurde auf den Standard AUTomotive Open System ARchitecture (AUTOSAR) ausgerichtet. Der Ansatz beschreibt mit Hilfe eines Informationsmodells automobile elektronische Systeme, indem es Entwicklungsinformationen in standardisierter Form erfasst. Darunter befinden sich neben Fahrzeugfunktionen auch funktionale Architekturen sowie die Hardware-Architektur [22].

Das “System Model“ ist das übergeordnete Gebilde eines EAST-ADL Modells und stellt die EE-Systeme innerhalb eines Fahrzeuges dar [21, S.20]. Diese EE-Systeme sollen bis ins Detail erfasst und eine Modellierung für die Dokumentation, das Design, die Analyse sowie Synthese ermöglicht werden. Systembeschreibungen sind auf

verschiedenen Abstraktionsebenen notwendig [21, S.14], das bedeutet, dass das vollständige eingebettete System mit unterschiedlichem Detaillierungsgrad auf den verschiedenen Ebenen repräsentiert wird. Das V-Modell dient als Referenzmodell für den Ansatz von EAST-ADL, da jede Ebene durch die Verifikations- und Validierungsaspekte abgebildet wird [23, F.5].

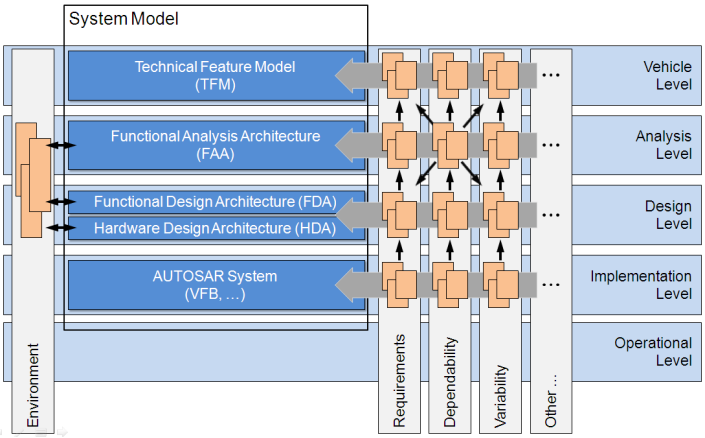


Abb. 2.4: Überblick EAST-ADL in [22]

Als Abstraktionsebenen, vgl. Abb. 2.4, werden das „Vehicle Level“, „Analysis Level“, „Design Level“ und das „Implementation Level“ [21, S.19], [24, S.22] gesehen. Das „Vehicle Level“ enthält technische Funktionsmodelle zur Darstellung der Fahrzeuginhalte („Technical Feature Model“). Sie sind so organisiert, dass sie die Fahrzeugkonfiguration mit unter anderem definierten Anforderungen reflektieren [21, S.23]. Das „Analysis Level“ bezieht sich auf die funktionalen Inhalte der EE-Systeme. Die funktionale Struktur wird durch die Functional Analysis Architecture (FAA) repräsentiert, wobei die Funktionen der vorangegangenen Phase realisiert werden [21, S.21]. Darauf aufbauend folgt das „Design Level“ auf dem die EE-Systeme auf einer zusätzlichen Ebene dargestellt werden. Integriert ist die Functional Design Architecture (FDA) sowie die Hardware Design Architecture (HDA). Die FDA repräsentiert eine Top-Level Funktion, d.h. sie unterstützt die Implementierung der Funktionen, die durch die FAA definiert wurden. Bei EAST-ADL beinhaltet die Design-Architektur auch die funktionale Spezifikation sowie die Hardware-Architektur eines EE-Systems. Zudem werden die nichttransparenten Funktionalitäten, wie Änderungsmodi oder der Umgang mit Fehlern, ebenfalls auf diesem Level repräsentiert, so dass ihre Auswirkungen abgeschätzt werden können [21, S.21]. Das „Implementation Level“ stellt die Software- und Hardwarearchitektur des EE-Systems eines Fahrzeuges dar. Definiert wird die Ebene durch Systemelemente von AUTOSAR [21, S.22]. Da das EAST-ADL Domänenmodell als Metamodell

dient und Konzepte von AUTOSAR verwendet, kann das EAST-ADL Metamodell in das AUTOSAR Metamodell importiert werden [21, S.14].

Die Modellstruktur unterstützt die Interaktion mit der Umgebung. Das „Environment Model“ gehört nicht zum „System Model“ und wird verwendet, um die Umgebung des EE-Systems zu beschreiben [21, S.64]. Es beinhaltet Funktionen, die beispielsweise dem Fahrzeugverhalten oder Teil des nicht-elektronischen Systems zugeordnet werden können. Diese „Umgebungsmodelle werden für die Validierung und Verifikation benötigt, von frühen Analysemodellen bis hin zu implementierten eingebetteten Systemen“ [21, S.19], [24, S.22]. Vergleiche der Architekturbeschreibungssprachen EAST-ADL und AADL sind unter anderem in [25], [26] und [27] zu finden.

2.3.2.1 Wissenschaftliche Projekte zur Weiterentwicklung von EAST-ADL – Verwandte Projekte und Technologien

Etliche wissenschaftliche Forschungsprojekte stehen im Bezug zu EAST-ADL. Vergangene und aktuell laufende Projekte sind in Abb. 2.5 ersichtlich, aufgeführt über die jeweiligen Versionen von EAST-ADL. Das Metamodell wird dadurch stetig weiterentwickelt. Dies kann unter anderem durch Change-Request in Form von Einspeisung von Projektergebnissen geschehen, um somit eine mögliche Erweiterung und Verbesserung zu unterstützen.

Der Einsatz und die Verwendung von EAST-ADL im produktiven industriellen Umfeld ist eher eingeschränkt. Begründet liegt dies darin, dass die EAST-ADL aus wissenschaftlicher Perspektive und Forschungssicht getrieben wird. Beteiligt an der Weiterentwicklung sind übergreifend sowohl OEMs, Zulieferfirmen, Toolhersteller als auch Forschungseinrichtung aus unterschiedlichen europäischen Ländern. Der Mehrwert von EAST-ADL zeigt sich insbesondere darin, dass allgegenwärtige Fragestellungen identifiziert und in einem gemeinsamen Umfeld vorangetrieben werden können, bei gleichzeitigem Aufbau eines gemeinsamen Verständnisses. Für spezielle Anwendungen im industriellen Umfeld werden häufig in EAST-ADL erarbeitete und verankerte Konzepte sowie Methoden genutzt, entsprechend portiert und adaptiert sowie in kommerziellen Produkten ausgeliefert. Die EAST-ADL bietet somit eine solide Basis, um insbesondere den Industriezweig Automobil voranzutreiben. Im Folgenden werden zugehörige und verwandte Projekte zu EAST-ADL nach [28] aufgeführt:

EAST-EEA und ATESST: Das EAST-EEA sowie das „Advancing Traffic Efficiency and Safety through Software Technology“ (ATESST) und das ATESST2 Projekt waren die treibenden Kräfte zur Entwicklung von EAST-ADL. So brachte EAST-EEA[6] die Spezifikation für EAST-ADL Version 1.02 hervor. Durch das Projekt ATESST[7] konnte die Spezifikation für EAST-ADL Version 2.0 entwickelt werden. Im Folgeprojekt

[6] https://itea3.org/project/east-eea.html (Zugriff am 15.10.2014)
[7] http://www.atesst.org (Zugriff am 15.10.2014)

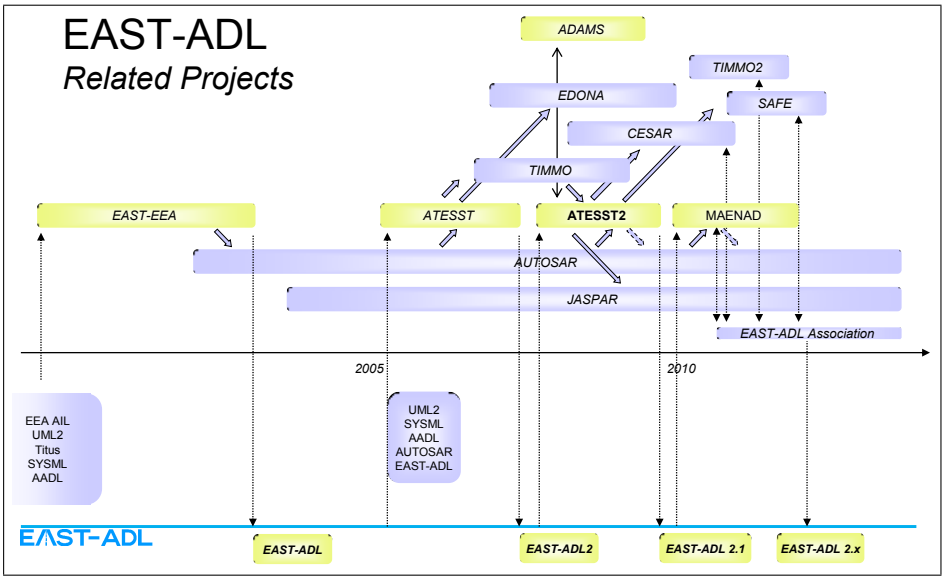


Abb. 2.5: „EAST-ADL Related Projects“ in [29, F.8]

„Advancing Traffic Efficiency and Safety through Software Technology 2“ (ATESST2)[7] wurde die Spezifikation für EAST-ADL Version 2.1 erstellt.

ADAMS: Das „Action for the Dissemination and Adoption of the MARTE and related Standards for component based middleware“ (ADAMS)[8] Projekt fokussierte sich im Rahmen des EU FP7 Programms, im Zeitraum von Mai 2008 bis April 2010[9], auf die industrielle Verwertung und Erweiterung insbesondere des Standards Modeling and Analysis of Real-Time and Embedded Systems (MARTE). Dies geschah sowohl durch modell- als auch komponentenbasierte Methoden, um die Entwicklung von Echtzeit- und eingebetteten Systemen voranzubringen. Ansatz war die Bildung einer Interessensgemeinschaft, die ihre Anliegen in die Standardisierungsbemühungen mit einbringt. Zielgruppen dieses Projektes waren der Automotive- sowie der Avionikbereich, die sich aktiv mit der Entwicklung und Integration von eingebetteten Systemen im Zuge von EAST-ADL, AUTOSAR und AADL befassen.

CESAR: Das „Cost-efficient methods and processes for safety relevant embedded systems“ (CESAR)[10] Projekt startete im März 2009 als ein ARTEMIS Joint Undertaking (JU) Projekt und endete im Oktober 2012. Ziele waren die Festigung der europäischen Vorreiterrolle im Transport- und Automationsbereich sowie der gestiegenen globalen Nach-

[8] http://www.adams-project.org (Zugriff am 15.10.2014)
[9] http://cordis.europa.eu/project/rcn/86635_en.html (Zugriff am 31.10.2014)
[10] http://www.cesarproject.eu (Zugriff am 15.10.2014)

frage in Bezug auf zunehmende Mobilität und Sicherheit nachzukommen. Die Weiterentwicklung der eingebetteten Systeme unter Sicherheitsaspekten war zudem motiviert durch eine Erhöhung der Effizienz bei gleichzeitiger Reduzierung der Kosten. CESAR verfolgte dazu den „multi-domain"-Ansatz, der die Interessen einer Vielzahl an Beteiligten aus den unterschiedlichen Domänen Avionik & Raumfahrt, Automobil, Zug und Automation integrierte. Es wurde ein geeigneter Entwurfsraum entwickelt und durch Formalisierungen die vielfältigen Sichtweisen und Kriterien vereint. Das CESAR Projekt sah vor, dass Firmen die Reference Technology Platform (RTP) basierend auf EAST-ADL mit der Interoperability Specification (IOS) verwenden, die eine Integration bereits bestehender Technologien mit Erneuerungen sowie eine übergreifende Nutzbarkeit ermöglicht. Die Ergebnisse aus CESAR werden seit Mai 2013 im „CRitical sYSTem engineering AcceLeration" (CRYSTAL) Projekt bzgl. der RTP und IOS weiter verfolgt.

EDONA: „Environnements de Développement Ouverts aux Normes de l'Automobile" (EDONA)[11] war ein auf den Zeitraum von September 2007 bis November 2010 angelegtes Projekt von Industrieakteuren aus 3 verschiedenen französischen Wettbewerbsregionen, darunter die „Systematic Paris-Area". Das Ziel des von Renault geleiteten Projektes war die Entwicklung einer Plattform, welche Kerntechnologien, kompatibel mit AUTOSAR, vereint und integriert, um somit für die Automobilindustrie verschiedene Entwicklungsketten zu erleichtern.

MAENAD: Beim „Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles" (MAENAD)[12] Projekt handelte es sich um ein unter FP7-ICT gefördertes Projekt. Laufzeit des Projektes war von September 2010 bis Februar 2014. Im Zusammenhang mit der zunehmenden Komplexität der vollelektronischen Fahrzeuge, im Englischen Fully Electric Vehicle (FEV), galt es Leistungsmanagement- und Optimierungsalgorithmen zu entwickeln, um eine hohe Performanz und Reichweite sowie einen niedrigen Energieverbrauch zu sichern. Hierzu wurde EAST-ADL 2 erweitert, um neben einer Unterstützung der ISO 26262 im Kontext automatischer Allokation von Sicherheitsanforderungen, modellbasierte Prognosen hinsichtlich Qualitätsattributen von FEVs sowie eine automatische Exploration von Designräumen zu ermöglichen.

TIMMO: Das Information Technology for European Advancement (ITEA) 2 Projekt „Timing Model" (TIMMO)[13] entwickelte im Zeitraum von April 2007 bis September 2009 eine standardisierte Infrastruktur für einen optimalen Umgang mit Timing-Informationen während des Designs von eingebetteten Echtzeit-Systemen in der Automobilindustrie. Die Notwendigkeit dafür wurde im Rahmen der Entwicklungspartnerschaft um AUTOSAR erkannt [30, S.23]. Innerhalb des Projektes wurde die formale UML-basierte Sprache „timing-augmented description language" (TADL) entwickelt, welche die EAST-ADL und AUTOSAR Modelle, bezogen auf zeitliche Aspekte, erweiterte. Zudem wurde eine Methodik für die Anwendung der Sprache sowie den Umgang

[11] http://www.edona.fr (Zugriff am 15.10.2014)
[12] http://www.maenad.eu (Zugriff am 15.10.2014)
[13] https://itea3.org/project/timmo.html (Zugriff am 15.10.2014)

mit zeitbezogenen Informationen entworfen. Die durch einen verkürzten Entwicklungskreislauf entstehende Kosteneinsparung sowie die gesteigerte Vorhersagbarkeit aufgrund frühzeitiger Analysen von zeitlichen Entwurfseinschränkungen gingen mit dieser Entwicklung einher. Die TIMMO-Sprache wurde in AUTOSAR 4.0 im Jahr 2009 integriert.

TIMMO-2-USE: Das auf TIMMO inhaltlich aufbauende „Timing Model - TOols, algorithms, languages, methodology, USE cases“ (TIMMO-2-USE)[14] Projekt war ebenfalls ein ITEA 2 Projekt mit einer Laufzeit von Oktober 2010 bis September 2012. Ziel war die Steigerung der Automatisierungen innerhalb der Entwicklungskreisläufe. Diese sollten dadurch abschätzbarer werden, um Entwicklungsrisiken bzw. Produkteinführungszeiten (time-to-market) zu reduzieren [31, S.24]. Durch die entwickelte Methodik verbindet TIMMO-2-USE die Kontroll- mit der Softwaredomäne. Zudem wurde die eXtensible Markup Language (XML)-basierte „Timing Augmented Description Language 2“ (TADL2) entwickelt.

SAFE: Das im Jahr 2011 gestartete ITEA 2 Projekt „Safe Automotive soFtware architEcture“ (SAFE) mit einer Laufzeit bis Ende 2014 erarbeitete Methoden und Lösungen für die integrierte Betrachtung und Analyse von funktionaler Sicherheit nach ISO 26262. Der Fokus liegt insbesondere auf einer modellbasierten Unterstützung bei der Entwicklung von Sicherheitsfunktionen im Automobil [32].

Hierzu wird ein erweitertes Metamodell zur Modellierung von Artefakten im Kontext von funktionaler Sicherheit im Rahmen der ISO 26262 basierend auf den existierenden Beschreibungssprachen EAST-ADL und AUTOSAR bereitgestellt. Das entwickelte Metamodell wird zudem in einer RTP von SAFE implementiert und mit verschiedenen Plug-Ins für beispielsweise Sicherheitsanalysen erweitert. Die SAFE Plattform basiert auf Sphinx[15], um somit die Integration mit den entsprechenden Implementierungen EAST-ADL Tool Platform (EATOP) und AUTOSAR Tool Platform (Artop) zu vereinfachen. Neben dem Metamodell sowie dessen Implementierung ist ein weiterer Schwerpunkt die Definition eines angepassten Entwicklungsprozesses konform zur ISO 26262, welcher mit realistischen industriellen Fallbeispielen evaluiert wird [32]. Weiterführende Informationen zum SAFE Projekt sind unter anderem in den Projektdeliverables und ITEA 2 Magazinen [33, S.8], [34, S.18ff] zu finden.

MBAT: Das „Combined Model-based Analysis and Testing of Embedded Systems“ (MBAT)[16] ARTEMIS-Projekt zielt auf eine Weiterentwicklung von Validierungs- und Verifikationstechnologien im Zusammenhang mit der Qualitätssicherung der im Transportwesen (Luftfahrt, Automobil und Schienenverkehr) verwendeten eingebetteten Systeme ab, um diese für die Industrie kosteneffizienter zu gestalten. Der entwickelte Ansatz kombiniert eine modellbasierte Testtechnologie mit statischen Analysetechniken und erweitert die V&V-Technologien in Form einer RTP. Laufzeit ist von November 2011 bis Dezember 2014.

[14] https://itea3.org/project/timmo-2-use.html (Zugriff am 15.10.2014)
[15] http://www.eclipse.org/sphinx (Zugriff am 15.10.2014)
[16] http://www.mbat-artemis.eu (Zugriff am 15.10.2014)

JasPar: Das „Japan automotive software Platform and architecture“ (JasPar) Projekt wurde im September 2004 mit dem Fokus auf einer zunehmenden Entwicklungseffizienz eingeführt. Das Konsortium wurde unter anderem durch japanische OEMs, wie Toyota, Nissan und Honda ins Leben gerufen. Das Ziel besteht darin, unter anderem durch Standardisierung und den allgemeinen Gebrauch von Software für elektronische Kontrollsysteme, eine zunehmende Entwicklungseffizienz sowie Zuverlässigkeit zu gewährleisten. JasPar reiht sich unter anderem in den Bereich der automobilen Entwicklungsvereinigungen für Standards wie AUTOSAR, Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug (OSEK) und Association for Standardisation of Automation and Measuring Systems (ASAM) ein, vgl. [129, S.19]. Arbeitsgruppen beschäftigen sich unter anderem mit AUTOSAR, Hochgeschwindigkeits-Netzwerken, Cyber Security, funktionaler Sicherheit, Mobilfunktelefon-Schnittstellen und Multimedia-Architekturen[17].

EASIS: Das „Electronic Architecture and System Engineering for Integrated Safety Systems“ (EASIS)[18] FP6-Projekt wurde von Januar 2004 bis März 2007 gefördert. Das Ziel war die Definition und Entwicklung von Technologien für integrierte Sicherheitssysteme (ISS). Dies wurde unter anderem durch das Bereitstellen einer Plattform für softwarebasierte Funktionalitäten in elektronischen Fahrzeugsystemen sowie durch eine elektronische Hardwareinfrastruktur unterstützt. Für die Anwendung auf ISS wurde ein dazugehöriger Entwicklungsprozess mit den notwendigen Werkzeugen definiert.

2.3.2.2 Eclipse-basierte Implementierung EATOP

Für EAST-ADL wurde eine dazugehörige Eclipse-basierte Implementierung namens EAST-ADL Tool Platform (EATOP) zur Verfügung gestellt, welche seit 2013 als Eclipse-Projekt online und Open Source ist. Die Implementierung von EATOP wurde wie bereits Artop auf das Sphinx Projekt[19] aufgesetzt, siehe Abb. 2.6. „Sphinx erleichtert die Erstellung von integrierten Modellierungs-Werkzeugumgebungen durch Unterstützung individueller oder mehrere[r] Modellierungssprachen“ [35, S.9]. Die EAST-ADL Metamodell-Implementierung basiert auf Java und dem Eclipse Modeling Framework (EMF)[20]. Unterstützt werden EAST-ADL Veröffentlichungen ab Version 2.1.10. Durch die Kombination mit Sphinx wird eine Serialisierung und Deserialisierung, welche konform zum XML-Schema des entsprechenden EAST-ADL Release ist, bereitgestellt. Durch den EAST-ADL Editor werden grundlegende Eclipse-Benutzerschnittstellen zur Verfügung gestellt, wie eine EATOP Perspektive und Eigenschaftsansicht. Weitere Funktionalitäten welche EATOP bietet, sind unter anderem Modell-zu-Modell-Transformationen, die Unterstützung von Variantenmanagement oder Refactorings. Als EATOP Technology Demonstrator wird eine selbstständige Anwendung unabhängig von Eclipse bereitgestellt.

[17] https://www.jaspar.jp/english (Zugriff am 15.10.2014)
[18] http://www.transport-research.info/web/projects/project_details.cfm?id=20366 (Zugriff am 15.10.2014)
[19] http://www.eclipse.org/sphinx (Zugriff am 15.10.2014)
[20] http://eclipse.org/proposals/modeling.eatop (Zugriff am 15.10.2014)

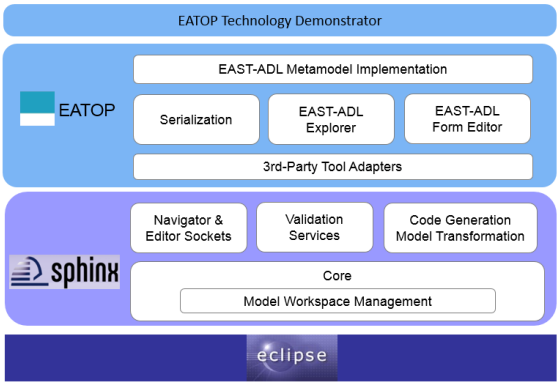


Abb. 2.6: EATOP Architektur aufbauend auf Eclipse Sphinx in [36]

In Abb. 2.7 sind die bedeutendsten Zusammenhänge zwischen EAST-ADL und EATOP ersichtlich. Einer der Hauptbestandteile ist die Übersetzung der EAST-ADL Metamodell-Definition in die EMF/Ecore-basierte Implementierung [35, S.10]. Durch die durchgängige Werkzeugkette bis zur EAST-ADL Implementierung und eine Programmierschnittstelle können unterschiedliche Anwendungen auf EAST-ADL durch Toolhersteller oder im Rahmen wissenschaftlicher Projekte darauf aufgesetzt werden.

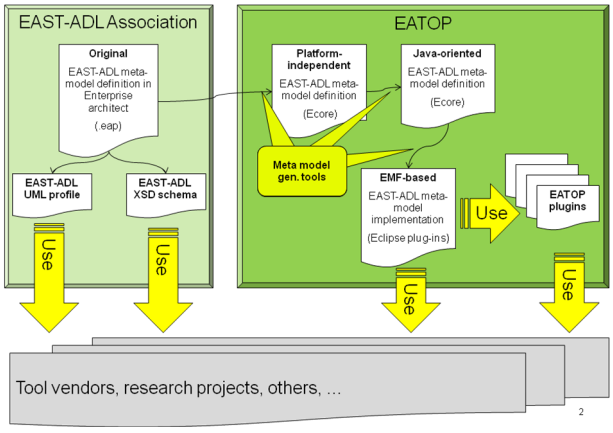


Abb. 2.7: „A tool chain for providing an EAST-ADL meta-model implementation [...]“ in [35, S.10]

Gleichsam zur ergänzenden Relation zwischen EAST-ADL und AUTOSAR sind auch deren Implementierung durch EATOP und Artop aufeinander abgestimmt, um durch die verwendete Sphinx Toolplattform eine maximale Interoperabilität zu gewährleisten.

2.3.3 AUTOSAR und Eclipse-basierte Implementierung Artop

Bei AUTomotive Open System ARchitecture (AUTOSAR) handelt es sich um eine Entwicklungspartnerschaft, welche 2002 mit einem technischen Team ins Leben gerufen und 2003 offiziell besiegelt wurde. Zusammengefunden haben sich sowohl Automobilhersteller, Zulieferer als auch Firmen aus den Bereichen Elektronik, Halbleiter und Entwicklungswerkzeuge. Diese sind innerhalb von AUTOSAR in „Core partners“, „Premium Partners“, „Development Partners“ und „Associate Partners“ sowie „Attendees“, je nach Beteiligung und Beitrag, klassifiziert.

Eines der Hauptziele des AUTOSAR-Konsortiums ist, den Austausch von Software auf verschiedenen Steuergeräten zu erleichtern. Hierzu konnte eine einheitliche Software-Architektur erarbeitet werden, welche unter anderem die Basic Software, AUTOSAR Runtime Environment und die AUTOSAR Software Component mit den AUTOSAR Interfaces definiert. Aktuelles Release hierzu ist 4.1 und innerhalb der entsprechenden Spezifikation festgehalten.

Motivation war es, die EE-Komplexität zu beherrschen, welche mit der steigenden Anzahl an Funktionalitäten einhergeht. Zudem soll flexibel auf Änderungen, Aufrüstungen und Aktualisierungen reagiert und eine Skalierbarkeit über Produktlinien hinweg ermöglicht werden. Nicht zuletzt sollen Qualität und Zuverlässigkeit der Systeme verbessert werden. Um dies zu erreichen, werden 9 Projektziele [AUTOSAR] verfolgt, welche in den Hauptarbeitsthemen „Software Architecture“, „Conformance Testing“, „Application Interfaces“ und „Methodology and Templates“ verankert sind: „Transferability of software, Scalability to different vehicle and platform variants, Different functional domains, Definition of an open architecture, Dependable systems, Sustainable utilization of natural resources, Collaboration between various partners, Standardization of basic software functionality of automotive ECUs, Applicable automotive international standards and state-of-the-art technologies“.

Artop[21] stellt die Eclipse-basierte Implementierung von AUTOSAR dar, welche Sphinx und EMF nutzt. Die Infrastruktur-Plattform bietet Basisfunktionalitäten für das Entwicklungstool, um somit AUTOSAR-konforme Systeme oder Steuergeräte zu entwerfen und zu konfigurieren. Unterprojekte von Artop sind ARText (An AUTOSAR Textual Language Framework)[22], welches als Framework das Aufsetzen einer textuellen Modellierungssprache für AUTOSAR ermöglicht, und ARUnit (Unit Testing of AUTOSAR Software Components)[23], welches eine leichtgewichtige Testumgebung für AUTOSAR Softwarekomponenten bereitstellt.

[21] http://www.artop.org (Zugriff am 15.10.2014)
[22] https://www.artop.org/artext/ (Zugriff am 15.10.2014)
[23] https://www.artop.org/arunit/ (Zugriff am 15.10.2014)

2.3.4 PREEvision

Das modellbasierte Entwicklungstool PREEvision [137] ermöglicht die Beschreibung von EEAs auf unterschiedlichen Abstraktionsebenen und Detaillierungsgraden. Die zugrundeliegende Beschreibungssprache ist die Electric Electronic Architecture - Analysis Design Language (EEA-ADL), welche mit dem EEA-Metamodell beschrieben wird [122, S.53]. Hierbei handelt es sich um ein „MOF-basiertes Modell, welches auf der M2-Ebene des Vier-Schichten-Modells der OMG angesiedelt ist“ [122, S.53]. Die Anwendung basiert auf der Eclipse Plattform [122, S.58]. Zahlreiche Konzepte konnten in das Tool einfließen, vgl. hierzu auch [37], [38], [39], [40], [41], [42], [43], [44]. Nachfolgend werden die Grundzüge der modellbasierten Entwicklungsunterstützung durch die Werkzeugumgebung vorgestellt.

2.3.4.1 Abstraktionsebenen

Im Folgenden werden die einzelnen Abstraktionsebenen, siehe Abb. 2.8, zur strukturierten Modellierung vorgestellt, welche innerhalb von PREEvision durch spezifische Diagramme unterstützt werden.

Ebene *Product Goals*: Die Produktziele (*Product Goals*) gliedern sich in die 3 unterschiedlichen Teilbereiche der Kundenmerkmale (*Customer Features*), der Anforderungen (*Requirements*) und dem Funktionalitäten-Netzwerk (*Feature Functionality Network*). Die ersten zwei Teilbereiche dienen hierbei einer systematischen Anforderungsentwicklung und einem -management, welches eine Grundvoraussetzung für die Produktentwicklung darstellt. Sie werden grundlegend über die Modellansicht aufgebaut und repräsentiert. Die textuellen Beschreibungen der beinhalteten Kundenmerkmale und Anforderungen können in Pakete weiter untergliedert, strukturiert und automatisch durchnummeriert werden. Eine Übersicht kann über tabellenbasierte Editoren gegeben werden.

Ebene *Logical Architecture*: Die logische Architekturebene unterstützt eine Abstraktion von der späteren technischen Realisierung in Hardware oder Software. Dies ist unter anderem dienlich, um beispielsweise eine Trennung zwischen Hardware und Software, basierend auf dem System-Funktionsnetzwerk, oder eine System-Dekomposition vornehmen zu können.

Ebene *Software Architecture*: Die Abstraktionsebene der Software-Architektur ermöglicht die Beschreibung von Softwarekomponenten und deren Kommunikation. Für komplexe Funktionalitäten kann eine strukturelle Dekomposition durchgeführt werden. Innerhalb dieser Ebene wird auch das Softwaredesign nach AUTOSAR durch die Modellierung und Erstellung von Softwarekomponenten und deren Schnittstellen unterstützt. Die Softwarekomponenten können mit der ausführenden Hardware verknüpft werden.

Ebene *Implementation*: Die Implementierungsebene verwaltet die Implementierungsartefakte der einzelnen Softwarekomponenten. Hierfür wird eine Verknüpfung zu einer

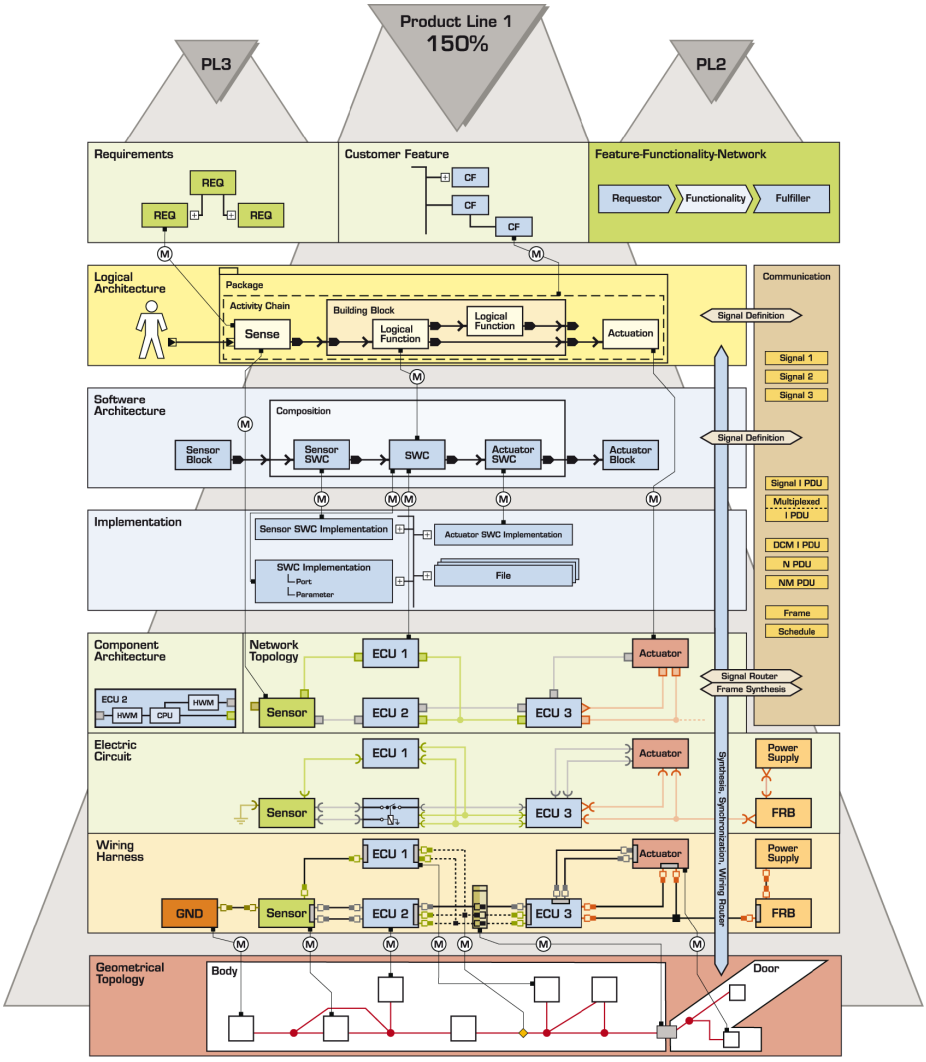


Abb. 2.8: Abstraktionsebenen von PREEvision in [137]

Verhaltensmodellierung in MATLAB/Simulink hergestellt. Dies unterstützt gleichzeitig konsistente Modelle in Bezug auf Ports und Schnittstellen.

Ebene *Hardware Architecture*: Zur Beschreibung von Hardware-Architekturen stehen unterschiedliche Diagrammtypen zur Verfügung, welche auf verschiedenen Granularitätsstufen und Darstellungsaspekten beruhen. So ermöglicht das Netzwerktopologie-Diagramm die Beschreibung der einzelnen Sensoren, Steuergeräte

und Aktuatoren sowie deren Netzwerk über Bussysteme und konventionelle Leitungen. Gleichzeitig werden Aspekte wie Spannungsversorgung und Massekonzepte berücksichtigt. Die Hardwarekomponenten-Architektur ermöglicht die Beschreibung von Aspekten innerhalb einer Hardwarekomponente. Hierzu sind unter anderem Mikrocontroller, Speicherbausteine und Hardwaremodule sowie interne logische Verbindungen verfügbar. Im Stromlaufplan werden die logischen Verbindungen zwischen den Hardwarekomponenten durch ihre physikalische Implementierung dargestellt. Im Kabelbaum kann die Darstellung durch Steckverbinder, Splices, Pins und Montagematerial angereichert werden.

Ebene *Geometrical Topology*: Mit Hilfe dieser Abstraktionsebene wird eine 2,5-dimensionale Beschreibung der Topologie ermöglicht. Die entsprechenden Daten können hierfür von 3D Computer-Aided Design (CAD) Tools importiert werden. Von Interesse sind insbesondere die Baubereiche, welche eine abgeschlossene Umgebung, wie beispielsweise den Motorraum beschreiben, und damit die Gesamttopologie in Bereiche aufteilt. Innerhalb dieser wird über Bauräume definiert, an welchen Koordinaten und über welche geometrischen Ausmaße bestimmte Einbaumöglichkeiten für Elektrik-/ Elektronik-Komponenten bestehen. Die einzelnen Bauräume wiederum können über Topologiesegmente miteinander verbunden werden, um ein späteres Routing der Leitungen zu ermöglichen.

2.3.4.2 Mappings

Über Verknüpfungen, sogenannte Mappings, können Artefakte aus unterschiedlichen Abstraktionsebenen miteinander in Bezug gesetzt werden. In der Anforderungsebene sind Verknüpfungen auch innerhalb der Ebene zulässig. Gleichzeitig können ebenenübergreifende Verknüpfungen von den Anforderungen ausgehend zu allen Artefakten der weiteren Abstraktionsebenen vorgenommen werden. Durch Verknüpfungen aus der logischen Architekturebene kann über Mappings eine Verfeinerung und Aufteilung in Software und Hardware erfolgen. Durch das Konstrukt der Mappings ist modellübergreifend eine eindeutige Nachverfolgbarkeit gegeben.

2.3.4.3 Regelwerk

PREEvision stellt Regelwerke für Modellabfragen sowie für Konsistenz- und Propagationsregeln bereit, die über eine grafische Modellierung durch Nutzung der Metamodell-Klassen aufgebaut werden können. Zudem kann gezielt das Vorliegen bestimmter Attributwerte überprüft werden. Über die Synchronisation der generierten Regeln mit dem EEA-Datenmodell lassen sich diese ausführen. Hierzu können im Rahmen des Metrikdiagramms beispielsweise Modellabfragen für ein Metrikartefakt hinterlegt werden. Alle im Datenmodell gefundenen Ergebnisse werden ausgegeben und stehen anschließend in der Metrik für weitere Berechnungen zur Verfügung.

2.3.4.4 Metrik-Framework

Das Metrik-Framework bietet einen grafischen Diagrammeditor und ermöglicht durch eine datenflussorientierte Metrikmodellierung die Ausführung von Berechnungen zur Analyse des aktuellen Datenmodells. Durch zur Verfügung gestellte Metrikartefakte wird eine Anreicherung von kundenspezifischer Intellectual Property (IP) unterstützt, indem Berechnungen als Java-Code hinterlegt werden können. Über Modelloperationen lassen sich zusätzlich Änderungen des Datenmodells für beispielsweise Optimierungen vornehmen. Die notwendigen Informationen aus dem Datenmodell können über Java-Implementierungen oder unter Nutzung der synchronisierten Modellabfrage-Regelwerke und deren generierten Regeln zur Verfügung gestellt werden.

2.3.4.5 Systemdokumentation

Für die Systemdokumentation wird ein Berichtgenerator bereitgestellt, welcher eine Ausgabe in Form verschiedener Dateiformate unterstützt, siehe Abb. 2.9. Zur Erstellung von strukturierten Dokumentationen können anpassbare Berichtvorlagen genutzt werden. Diese können sich sowohl aus statischen als auch dynamischen Anteilen, welche als Platzhalter eingebracht werden, zusammensetzen. Die Platzhalter werden bei der Generierung des Berichtes über Modellabfragen oder Metrikergebnisse aus dem aktuellen Datenmodell befüllt. Somit können unter anderem Diagramme und Beschreibungen aus dem Modell aufgegriffen werden.

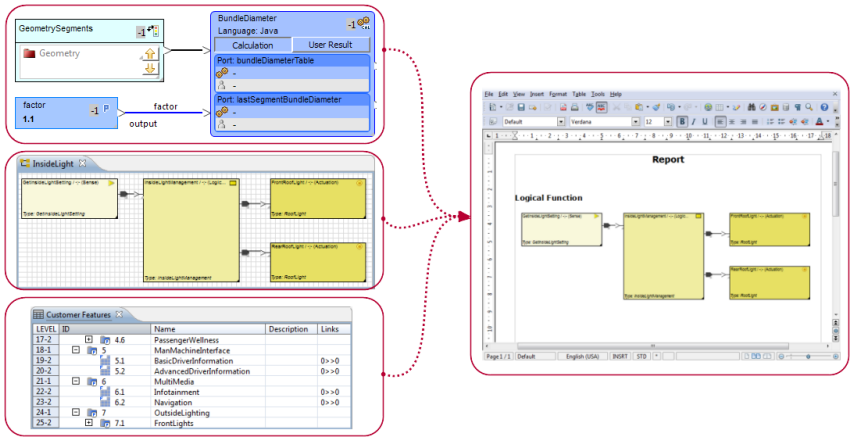


Abb. 2.9: Systemdokumentation mit PREEvision in [137, S.50]

2.3.4.6 Produktlinien- und Variantenmanagement

PREEvision ermöglicht ein Produktlinien- und Variantenmanagement. Das Variantenmanagement wird verwendet, um unterschiedliche Produkte innerhalb einer Produktlinie handhaben zu können. Hierzu wird die Produktlinie einer EEA als sogenanntes 150 % Modell aufgesetzt, um innerhalb eines einzelnen, in sich konsistenten Modells, alle Produkte vorliegen zu haben. Durch die Verwendung des integrierten Variantenmanagments werden die entsprechenden Produkte in unterschiedliche Teilmengen zugeschnitten.

2.3.4.7 Entwurf sicherheitsrelevanter Systeme

PREEvision Version 6.0 unterstützt funktionale Sicherheit im Zuge der ISO 26262 durch eine spezielle Perspektive, um somit Aspekte von sicherheitsrelevanten Systemen hervorzuheben. Hierbei wird die Item-Definition, die Durchführung eines Hazard Analysis and Risk Assessment (HARA), die Erstellung eines funktionalen und technischen Sicherheitskonzepts und die Durchführung einer FMEA ermöglicht [137, S.758]. In [45, S.332] werden FMEA und Fault Tree Analysis (FTA) als Sicherheitsanalysen zur Überprüfung des technischen Sicherheitskonzeptes im Kontext von PREEvision aufgeführt.

2.4 Standardisierungen

Trotz gegebenenfalls neu einzuhaltender Auflagen, welche im Zuge des Standes der Technik bei der Veröffentlichung eines neuen Standards gegeben sind, tragen Standardisierungen maßgeblich zur Weiterentwicklung von Industriezweigen bei. Dies ist zum einen bedingt durch die gemeinschaftliche Erarbeitung von Standards in Gremien, um somit unter anderem Methoden, Vorgehensweisen sowie Best-Practice festzuhalten; zum anderen, um nach der Veröffentlichung ein domänenweites Verständnis für die Thematiken aufzubauen und einen Austausch innerhalb der spezifischen Fachbereiche zu fördern.

Im Rahmen dieser Arbeit wurden unterschiedliche nationale und internationale Normen und Standards aufgegriffen, welche unter International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Object Management Group (OMG) oder Deutsches Institut für Normung e.V. (DIN) publiziert wurden.

Bedingt durch den geografischen Wirkungsbereich sind unterschiedliche Komitees in der Verantwortung, siehe Abb. 2.10. Für allgemeine Standards sind das DIN, das Comité Européen de Normalisation (CEN)[24] sowie die ISO zuständig. Standards im Fachgebiet Elektrotechnik werden ebenfalls auf diesen 3 Ebenen von der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (DKE)[25], Comité

[24] https://www.cen.eu (Zugriff am 15.10.2014)
[25] http://www.dke.de (Zugriff am 15.10.2014)

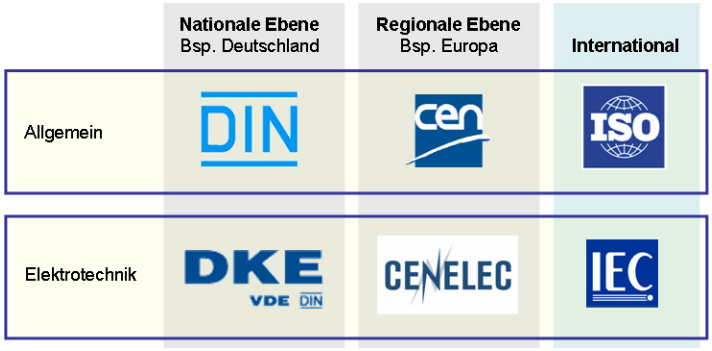


Abb. 2.10: Normungsorganisationen - angepasste Abbildung von DIN Deutsches Institut für Normung e. V. [46]

Européen de Normalisation Électrotechnique (CENELEC)[26] und IEC abgebildet und abgehandelt. Die Kooperation von CENELEC mit IEC ist im CENELEC Guide 13[27] festgehalten, auch bekannt unter „Dresden agreement". Eine weitere technische Kooperation zwischen ISO und CEN konnte im „Vienna agreement"[28] verankert werden. Der DKE ist ebenfalls bemüht, die nationalen Normen mit den europäischen und internationalen zu harmonisieren.

Normen und Standards erhalten bedingt durch deren geographisch normative Anwendung eine Kennzeichnung. Hierbei wird auf nationaler Ebene das Kürzel „DIN", auf regionaler Ebene „EN" für „Europäische Normen" und auf internationaler Ebene „ISO" verwendet. Aufgrund von nationalen Übersetzungen kann ein Standard in mehreren offiziellen Versionen zur Verfügung stehen. Originalsprache, weitere vorliegende Übersetzungen und der Bezug zum Ursprungsdokument sind jeweils vermerkt und bereits in der Dokumentnummer des Standards ersichtlich. Die einzelnen Phasen im Standardisierungsprozess werden durch die „International harmonized stage codes"[29] von der ISO dargestellt und branchenspezifisch beispielsweise durch die IEC auf die eigenen Codes[30] referenziert. Zusätzlich werden weitere Bezeichnungstypen, wie TS (technische Spezifikation) und TR (technischer Bericht), verwendet. Dabei handelt es sich um zusammengetragenes Wissen von Fachexperten nach dem Stand der Technik, jedoch ist deren Anwendung nicht normativ.

[26] http://www.cenelec.eu (Zugriff am 15.10.2014)
[27] http://www.cenelec.eu/aboutcenelec/whoweare/globalpartners/iec.html (Zugriff am 15.10.2014)
[28] http://www.din.de/sixcms_upload/media/2896/Vienna_Agreement.pdf (Zugriff am 15.10.2014)
[29] http://www.iso.org/iso/home/standards_development/resources-for-technical-work/stages_table.htm (Zugriff am 15.10.2014)
[30] http://www.iec.ch/standardsdev/how/processes/stage_codes.htm (Zugriff am 15.10.2014)

Im Folgenden wird der Standardisierungsprozess der ISO anhand eines vereinfachten Modells erläutert, vgl. [47, S.27]. Die ISO spricht von einem Projekt. Hierfür werden die einzelnen Phasen aufgegriffen, welche unter anderem auch einer Absicherung des Inhalts und dem Aufbau eines gleichen Verständnisses dienen. Grundlegend ist der Prozess für das Projekt in 7 bzw. 8 Phasen eingeteilt, die auch auf die einzelnen Stages abgebildet werden können, siehe Abb. 2.11.

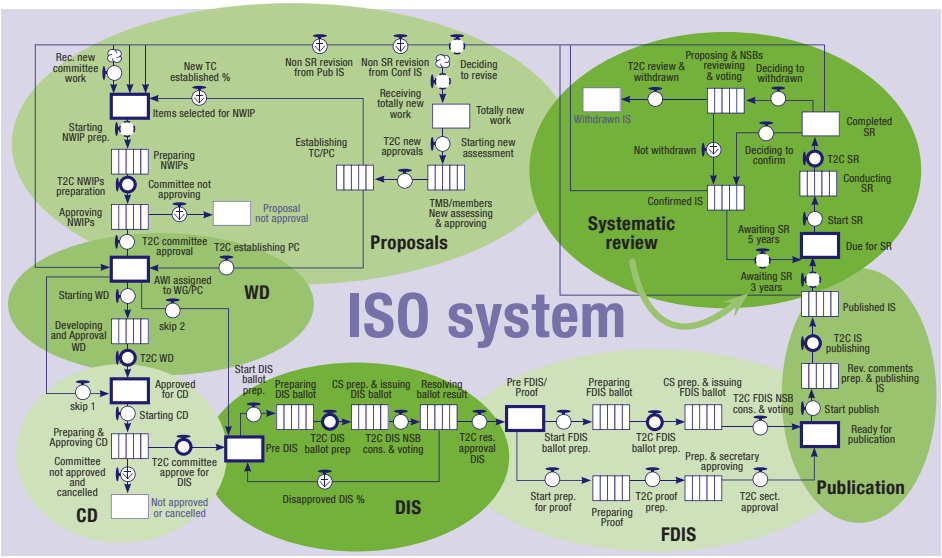


Abb. 2.11: „A model of ISO's current processes“ in [47, S.27]

Bei vorliegendem Bedarf für einen neuen Standard, der keinen Zusammenhang zu einem bisherigen Internationalen Standard hat bzw. auch keine Revision darstellt, wird ein neuer Prozess der Standard-Veröffentlichung begonnen. Die erste Phase ist die „Proposal Stage“ (Kennung 10), in welcher der Vorschlag vom Technical Management Board (TMB) zugelassen und einem eingerichteten Technical Committee (TC) vorgelegt wird. Das TC wählt die Items für das New Work Item Proposal (NWIP) aus. Das Approved Work Item (AWI) wird einer Arbeitsgruppe, Working Group (WG), aus Experten übergeben, die diesen Vorschlag als Working Draft (WD) bearbeitet. Dies wird unter der verbindlichen „Preparatory Stage“ (Kennung 20) aufgeführt. Das Parent Committee (PC) entscheidet darüber, ob der Vorschlag in der Komiteephase zum Committee Draft (CD) zugelassen wird. Hierbei handelt es sich um die optionale „Committee Stage“ (Kennung 30), in welcher das TC unter anderem über den weiteren Verlauf bzw. Prozessabbruch entscheidet. Bei erfolgreichem Verlauf wird der CD im nächsten Schritt als Draft International Standard (DIS) zugelassen und allen National Standard Bodys (NSBs) zur Abstimmung vorgelegt. Alternativ kann die „Committee Stage“ dadurch umgangen werden,

dass der WD direkt als DIS vorgelegt wird und damit in die verbindliche „Enquiry Stage“ (Kennung 40) übergeht. Nachgeschaltet ist die optionale „Approval Stage“ (Kennung 50), welche eine zusätzliche Prüfung des Projektes in Form des Final Draft International Standard (FDIS) ermöglicht. Bei erfolgreichem Mehrheitsvotum erfolgt die Veröffentlichung des International Standard (IS) durch das Central Secretariat (CS) im Rahmen der „Publication Stage“ (Kennung 60). Durch ein Systematic Review (SR) im dritten bzw. fünften Jahr nach der Veröffentlichung wird der Standard überprüft und ggf. zurückgezogen oder in seiner Berechtigung bestätigt. Hierzu sind die zwei Phasen der „Review stage“ (Kennung 90) und „Withdrawal stage“ (Kennung 95) gegeben. Weitere Informationen können aus den Direktiven [ISOdir-1] und [ISOdir-2] entnommen werden.

Hinsichtlich der zeitlichen Rahmenbedingungen kann auf Abb. 2.12 aus [48, S.4] des Verbandes der Automobilindustrie e. V. (VDA) verwiesen werden. Die 3 unterschiedlichen Zeitrahmen nach [ISOdir-1, 2.1.6.1] sind hier den einzelnen Phasen zugeordnet. Ausgangspunkt ist das „New project approved“ mit der Kennung 10.99, von dem aus die entsprechenden Zeiten in Monaten gelistet sind. Die Durchlaufzeit beträgt somit bis zum „International Standard published“ mit der Kennung 60.60 insgesamt 36 Monate für den Standardzeitrahmen, 48 Monate für den erweiterten Zeitrahmen und 24 für den beschleunigten Zeitrahmen.

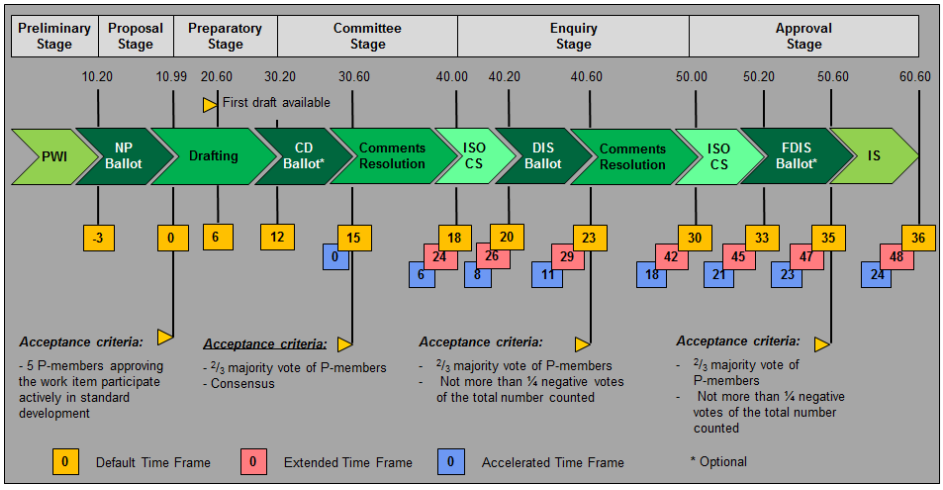


Abb. 2.12: „Aktueller ISO-Normen-Entwicklungsprozess“ in [48, S.4]

Kontroverse Diskussionen werden geführt, wenn es inhaltlich sowohl um die Frage geht, ob auf übergreifende Standards zurückgegriffen werden muss, falls kein domänenspezifischer Standard vorliegt als auch inwiefern Normenkonformität gleichbedeutend mit Rechtskonformität ist, vgl. hierzu unter anderem den Beitrag [49].

2.5 Prozessmanagement

Nach [138, S.3] hat das Geschäftsprozessmanagement, im Englischen Business Process Management (BPM), zwei geistige Vorfahren. Zum einen Shewhart und Deming im Bereich der statistischen Prozesskontrolle, zum anderen Hammer [50] im Bereich der Geschäftsprozess-Umgestaltung, im Englischen Business Process Reengineering (BPR).

In [138, S.37] spricht Harmon von drei sogenannten großen Traditionen, welche in einzelnen Organisationen vorliegen: „Management-Tradition“, „Qualitätssicherungs-Tradition“ und „Informationstechnologie-Tradition“. Bezogen auf die „Qualitätssicherungs-Tradition“ wird bereits auf den Beitrag „Principles of Scientific Management“ aus dem Jahre 1911 von Frederick Winslow Taylor [139] verwiesen. Beschrieben werden Schlüsselideen im Kontext von Arbeitserleichterung, Zeitstudium und systematischem Experimentieren, welche Manager unterstützen sollen. Hierunter werden sowohl „TQM“, „Lean“ und „Six Sigma“ als auch das „Capability Maturity Model“ eingeordnet. Hinsichtlich der „Management-Tradition“ liegt der Fokus auf der Gesamtperformance des Unternehmens. Die Ausrichtung der Strategie ist ein Schwerpunkt, um die Unternehmensziele zu erreichen. Unter anderem zählt zu diesem Bereich das BPR. Die „Informationstechnologie-Tradition“ bezieht sich auf die Verknüpfung von Prozessen mit Computer und Softwareanwendungen. Das Augenmerk lag auf der Automatisierung von Arbeitsprozessen und nahm seinen Anfang in den späten 1960er Jahren im Bereich der Abrechnungsstellen, wie Buchhaltungen. Mittlerweile hat sich dies auf eine Vielzahl an Tätigkeitsbereichen ausgeweitet, um Mitarbeitern ihre alltägliche Arbeit zu erleichtern. Hierunter fallen beispielsweise das BPR, Prozessmodellierungs-Tools und die Schnittstelle zwischen Geschäftsangelegenheiten und der Informationstechnologie.

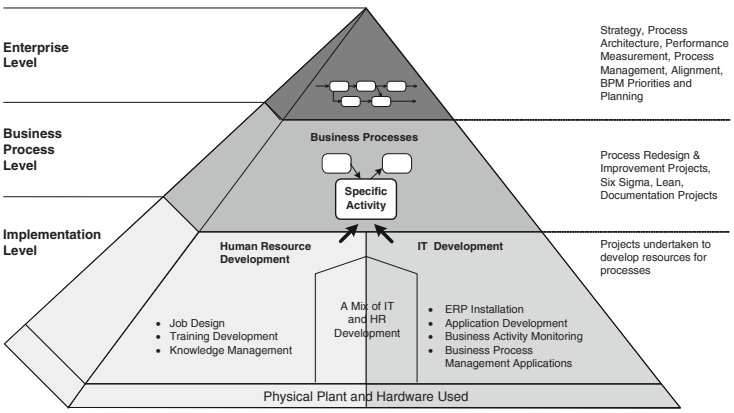


Abb. 2.13: „The business process trends pyramid“ in [138, S.54]

Inzwischen verschmelzen die einzelnen Traditionen und Synergien entstehen, um auf unterschiedlichen Ebenen gleichzeitig zu arbeiten. Auf oberster Ebene, wie in Abb. 2.13 ersichtlich, wird das „Enterprise Level“ angeordnet [138, S.54]. Dies zielt auf das gesamte Unternehmen ab, um Prozesse mit Strategien auszurichten und sowohl Steuerungs- als auch Messsysteme einzuführen. Das mittig angesiedelte „Business Process Level“ fokussiert sich auf „die Erstellung, die Umgestaltung oder Verbesserung von Geschäftsprozesse[n]“ [138, S.70]. Insbesondere stehen die Methodiken und Tools im Mittelpunkt. Das „Implementation Level“ bezieht sich auf die Umsetzung von spezifischen Lösungen für Geschäftsprozesse. An dieser Stelle wird von Business Process Management Systems (BPMSs) gesprochen, welche unter anderem Arbeitsabläufe und Regelwerke in einem Softwaretool vereinen.

Allgemein können betriebliche Prozesse in verschiedene Teilmengen strukturiert werden, so bilden die Geschäftsprozesse eine Untermenge der betrieblichen Prozesse. Alternativ kann anstelle von einem Geschäftsprozess auch von einem Kernprozess, Leistungsprozess oder Unternehmensprozess gesprochen werden. Unter diesen Begriff werden alle Aktivitäten gefasst, die einem Erreichen der Organisations- und Unternehmensziele dienlich sind und welche die Wettbewerbsfähigkeit eines Unternehmens sichern, vgl. [140, S.1f]. Prozesse können anhand von Kriterien klassifiziert werden, darunter die Einheiten, die am Geschäftsprozess beteiligt sind, die Objekte, die transformiert werden bzw. die Art der Tätigkeit, die durchgeführt wird, vgl. [140, S.5f]. Je nach Ausführung werden Prozesse neben den Leistungsprozessen in Führungsprozesse sowie Unterstützungsprozesse eingeteilt. Führungsprozesse haben vor allem Leitungsfunktion in Bezug auf die Prozessarchitektur, Unterstützungsprozesse dienen dem Support der Leistungsprozesse [140, S.8].

Im Zusammenhang mit dem Geschäftsprozessmanagement werden insbesondere Prozessmodelle verwendet, welche unter anderem der Dokumentation, Analyse und Automatisierung zu unterschiedlichen Phasen des Prozesslebenszyklus dienen [141, S.262]. Als verschiedene Perspektiven auf den Prozess werden nach [141, S.263] die am häufigsten gelistet: Kontrollfluss-, Organisations- und Datenperspektive sowie die funktionale und die operative Perspektive. Diese stehen im Zusammenhang des Verhaltens, der Ressourcen, Informationen, Aufgaben und Anwendungen.

2.5.1 Prozessmodellierung

Ein Teilbereich des Geschäftsprozessmanagements ist unter anderem die Prozessmodellierung. In [140, S.47] werden Prozessmodelle als vereinfachte Repräsentationen von komplexen Prozessen dargestellt. Im Artikel von Curtis et al. [51, S.76] wird geschildert, dass „ein Prozessmodell eine abstrakte Beschreibung eines aktuellen oder vorgeschlagenen Prozesses“ repräsentiert. Dargestellte Prozesselemente können im Folgenden Menschen oder Maschinen hinsichtlich der Ausführung zugeordnet werden. Nach [138, S.171] wird der Beitrag [51] als „ungefähres Geburtsdatum der modernen Geschäftsprozessmodellierungs-Disziplin“ verwendet.

Prozessmodelle geben einen Überblick über den Prozessfluss und erlauben die Zuweisung von Ressourcen. Insgesamt ermöglicht dies das Verständnis von Prozessen und formt die Basis für nachfolgende Analysen und Planung der Prozesse. Vorteile der Prozessmodellierung und des prozessorientierten Wissensmanagements werden nach [DINsp91281, Anhang A] anhand von Fallstudien aus unterschiedlichen Branchen aufgezeigt. Es wird erwähnt, dass neue Mitarbeiter beispielsweise im zeitintensiven Tagesgeschäft leichter eingearbeitet werden können. Ein weiterer Nutzen besteht darin, fachbezogenes Wissen zu vermitteln, so dass Qualitätsdefizite zugleich verhindert werden können. Weiterhin kann es für das Einbinden von Projekterfahrungen in standardisierte Geschäftsabläufe hilfreich sein. Zudem kann eine „Verteilung bzw. Aktualisierung und Umsetzung der kunden- bzw. branchenspezifischen Anforderungen“ ermöglicht werden, wenn die „vorliegenden Informationsstrukturen“ nicht mehr ausreichend sind. Die Prozessmodellierung unterstützt die Transparenz über Prozesse und Strukturen in Unternehmen, so dass Aufgaben schneller bearbeitet werden können und dem Fehlen klarer Handlungsanweisungen für zentrale Prozesse entgegengewirkt werden kann. Des Weiteren gilt es durch das prozessorientierte Wissensmanagement die wachsenden Wissensbestände zu beherrschen und einem großen Mitarbeiterkreis zur Verfügung zu stellen, um eine redundante Datenhaltung zu verhindern. Somit ist vorhandenes Fach- und Methodenwissen schnell zugreifbar, systematisch sowie strukturiert aufbereitet und verknüpft die einzelnen Fachgebiete. Nach [140, S.47] werden somit durch Prozessmodellierung und -management unter anderem folgende Ziele angestrebt: Transparente Abläufe, Fehlervermeidung, Kostenreduktion, Dokumentation/personenunabhängige Verfügbarkeit des Wissens, Schulungsmaterial, Mitarbeitermotivation, Auswertungsmöglichkeiten, Prozessoptimierung, Simulationen, Zertifizierung.

Hinsichtlich der Prozessmodellierung werden im Rahmen des Beitrags von [52] Qualitätseigenschaften an Prozessmodellierungstechniken gestellt. Zur Beurteilung werden drei Eigenschaften des FRISCO Berichts [53] in den Fokus gerückt, welche sich auf die Metamodelle der Modellierungssprachen beziehen. Dazu gehört neben der Ausdrucksfähigkeit, die Willkür sowie die Angemessenheit. Um den Fokus auszuweiten stellt Hommes weitere aus der Literatur gesammelte Aspekte dar, welche eine Modellierungstechnik ausmachen. Dazu gehört neben der Verständlichkeit, die Kohärenz, die Vollständigkeit, die Effizienz sowie die Effektivität.

2.5.2 Prozessmodellierungssprachen

Einen Überblick über Standards im Bereich des Geschäftsprozessmanagement und deren Evolution ist in [138, S.513] gegeben. Bezüglich der Prozessmodellierung wird dies insbesondere an den graphenbasierten und anwenderbasierten Ansätzen vorgenommen, welche als die zwei Grundlegendsten gelten. In [142, S.15ff] werden als Prozessmodellierungssprachen die Ereignisgesteuerte Prozesskette (EPK), die Petri-Netze und die Business Process Model and Notation (BPMN) erwähnt.

Diese gehören nach [143, S.64] zu den kontrollflussorientierten Methoden. Die verschiedenen diagrammbasierten Methoden, die zur Darstellung von Geschäftsprozessen verwendet werden, können neben dem kontrollflussorientierten Ansatz in zwei weitere Ansätze, datenfluss- und objektorientiert, eingeteilt werden. Die datenflussorientierten Methoden beinhalten beispielsweise Darstellungen, wie die Struktogrammtechnik der Softwareentwicklung, und werden immer seltener genutzt [143, S.64]. Im Zusammenhang mit Geschäftsprozessen wird daher nicht näher auf diesen Ansatz eingegangen. Die objektorientierten Methoden verbreiten sich dagegen zunehmend und werden unter anderem durch Anwendungsfall- sowie Aktivitätsdiagramme aus UML unterstützt. In den folgenden Unterkapiteln werden daher die Methoden aus UML, die Petri-Netze, EPK und die BPMN genauer vorgestellt.

2.5.2.1 UML im Kontext der Prozessmodellierung

Die Unified Modeling Language (UML) hat sich aus den drei vorausgegangenen objektorientierten Methoden OOSE, OMT und Booch entwickelt [UMLinf, S.1]. 1997 wurde sie durch die OMG als internationaler Standard veröffentlicht [143, S.90]. UML, aktuelle Version 2.4.1, war ursprünglich vor allem der Softwareentwicklung und dem Systemdesign angegliedert. Der Standard stellt nun aber auch die Modellierung von Geschäftsprozessen oder vergleichbaren Prozessen zur Verfügung [UMLinf, S.1], insbesondere wird speziell die Softwareprozess-Modellierung genannt [UMLinf, S.14]. Erwähnt wird, dass Aktivitäten zur organisatorischen Modellierung von Geschäftsprozessplanungen und Arbeitsabläufen verwendet sowie zur Modellierung von Prozessen zur Informationsgewinnung genutzt werden können [UMLsup, S.324]. Prozesse werden dadurch unter anderem auf Systemebene spezifiziert. Ereignisse können sich sowohl innerhalb des Systems im Zuge von finalisierten Aufgaben ereignen als auch außerhalb des Systems auftreten.

In [143] werden zwei verschiedene Diagrammtypen genannt, die aus UML für die Prozessmodellierung verwendet werden können: Anwendungsfalldiagramm (Use-Case Diagram) sowie Aktivitätsdiagramm (Activity Diagram). Das Anwendungsfalldiagramm wird vor allem in der Softwareentwicklung, mitunter jedoch auch bei der Analyse von Geschäftsprozessen verwendet, indem Geschäftsvorfälle beschrieben werden [143, S.90]. Durch das Anwendungsfalldiagramm wird der Bezug zwischen Funktion und Organisationseinheit veranschaulicht. Das Aktivitätsdiagramm stellt Prozesse auf hohem Abstraktionsniveau [143, S.64] sowie einzelne Verarbeitungsschritte eines Geschäftsvorfalls dar. In den Grundzügen ist es den seit vielen Jahren verwendeten Programmablaufplänen ähnlich. Beim Aktivitätsdiagramm lassen sich zudem Parallelen zu den kontrollflussorientierten Methoden der Prozessmodellierung erkennen [143, S.91].

2.5.2.2 Petri-Netze

Petri-Netze gehören zu den kontrollflussorientierten Methoden und bieten eine weitreichende Einsatzmöglichkeit, vgl. [142, S.20]. Basierend auf der mathematischen Fundierung eignen sie sich unter anderem zur Modellierung und Simulation von Prozessen, weshalb sie auch im Kontext der Geschäftsprozessmodellierung sowie für Workflows verwendet werden [143, S.73]. Grundstein hierfür legte Carl Adam Petri im Zusammenhang mit seiner im Jahre 1962 veröffentlichten Dissertation [144].

Unter einem Petri-Netz versteht man einen gerichteten bipartiten Graphen, welcher sich somit aus zwei unterschiedlichen Arten von Knoten zusammensetzt. Zum einen gibt es sogenannte Stellen, die durch Kreise symbolisiert werden und im Kontext der Prozessmodellierung statische Zustände von Dokumenten, Daten oder Ressourcen darstellen, vgl. [143, S.73]. Zum anderen werden Transitionen verwendet, welche die Umformungen von Informationen im Sinne von Funktionen, Prozessen oder Aktivitäten repräsentieren und damit die Zustandsübergänge verdeutlichen. Diese werden durch Rechtecke abgebildet. Mit Hilfe von Kanten, die grafisch durch Pfeile ausgedrückt werden, wird der Kontrollfluss zwischen den einzelnen Stellen und Transitionen hergestellt, vgl. [143, S.73], [145, S.22]. Dynamisches Netzverhalten wird durch sogenannte Marken, mit denen die Stellen belegt werden, dargestellt [143, S.73]. Diese durchlaufen entsprechend der Schaltregeln das Modell.

Einfache Petri-Netze wurden zu Höheren weiterentwickelt, welche in einer Vielzahl an Variationen vorliegen. In den Ausführung von [143] werden die gängigsten Typen gelistet und erläutert. So gehören die Kanal/Instanzen-Netze (K/I-Netze), Bedingungs/Ereignis-Netze (B/E-Netze) sowie die Stellen/Transitions-Netze (S/T-Netze) zu den einfachen Petri-Netzen, während die Prädikat/Transitionen-Netze (Pr/T-Netze), die zeitbewerteten Petri-Netze und die hierarchischen Petri-Netze dem höheren Grundtyp zugeordnet werden [143, S.75].

Der Vorzug gegenüber anderen Darstellungsformen liegt in der leicht verständlichen Darstellung der Petri-Netze, da sie sich aus nur wenigen unterschiedlichen Elementen aufbauen lassen. Zugleich werden vorliegende Analysemethoden als Vorteile betrachtet. Die Einfachheit, bedingt durch die Elementtypen, erfordert jedoch oft eine Erweiterung zu höheren Petri-Netzen, um reale Sachverhalte abbilden zu können. Hierbei sind insbesondere Modellhierarchien und Zeitangaben zu nennen.

In der Praxis erscheinen diese erweiterten Formen jedoch teilweise als zu komplex und schwer verständlich. Aufgrund der vielen Variationen ergaben sich im Laufe der Zeit uneinheitliche Notationen. Nach [143, S.76] wird dies als Grund aufgeführt, dass sich Petri-Netze wenig für betriebswirtschaftlich orientierte Geschäftsprozessmodelle und deren Diskussion im Kontext einer Ist-Analyse und Sollkonzeption von Prozessen eignen. Der Stand der Technik zeigt jedoch, dass dies nicht unbedingt der Fall sein muss.

Mit ISO/IEC 15909 wurde für die Software- und Systementwicklung ein Standard für höhere Petri-Netze definiert. Dieser stellt eine semi-grafische Modellierungssprache für die Spezifikation, Design und Analyse von Systemen zur Verfügung. In Part 1 wird unter anderem die mathematische Definition festgelegt und als eines der generischen Anwendungsgebiete auf den Bereich der Geschäftsprozessmodellierung verwiesen. Part 2 beschreibt das Petri Net Markup Language (PNML) Core Model und ein XML-Format, um den Austausch von höheren Petri-Netzen zwischen unterschiedlichen Werkzeugen zu ermöglichen. Das Transferformat ist derartig gestaltet, dass es auch für zukünftige Erweiterungen genutzt werden kann.

2.5.2.3 Ereignisgesteuerte Prozesskette (EPK) und erweiterte EPK

Die Ereignisgesteuerte Prozesskette (EPK), im Englischen mit Event-driven Process Chain (EPC) bezeichnet, wird insbesondere im Bereich des industriellen Geschäftsprozessmanagements verwendet [142, S.15]. Im Rahmen von [54] wurde die EPK durch Keller, Nüttgens und Scheer im Jahr 1992 vorgestellt. Es handelt sich um eine grafische Modellierungssprache, welche innerhalb des Rahmenkonzeptes Architecture of Integrated Information Systems (ARIS) entwickelt wurde. Grundlage der EPK sind stochastische Netzplanverfahren sowie Petri-Netze [146, S.125].

Als gerichteter Graph enthält die EPK die grafischen Hauptelemente Funktion, Ereignis, Kontrollfluss und Konnektoren. Die Funktion ist mit einer Aktivität innerhalb eines Prozesses gleichzusetzen und wird in Form eines abgerundeten Rechtecks repräsentiert. Ereignisse stellen Prozesszustände dar und werden durch Hexagone veranschaulicht. Diese beiden Hauptelemente werden durch den Kontrollfluss verbunden. Zudem wird es durch Konnektoren ermöglicht, logische Operatoren auszudrücken [142, S.15f], hierfür existieren verschiedene Symbole [146, S.125]. Hierdurch können die Abläufe aufgespalten oder wieder zusammengeführt werden.

Der Prozessablauf kann als eine Folge von Funktionen und Ereignissen im Wechsel beschrieben werden [142, S.15]. Vorgegebene Regeln sind bei der Modellierung zu berücksichtigen. Ereignisse stehen am Anfang eines Prozesses und beschreiben Zustände, welche für das Auslösen einer Funktion zuständig sind [54, S.10]. Es besteht die Möglichkeit, dass auf ein Ereignis mehrere Funktionen folgen, demgegenüber kann ein Ereignis erst durch den Abschluss mehrerer Funktionen hervorgebracht werden [146, S.125].

Beinhaltet das Prozessmodell neben den Hauptelementen der EPK weitere Elemente, so wird von der erweiterten EPK (eEPK) gesprochen. Zu den Erweiterungen zählen unter anderem der Ein- und Ausgang von Prozessen. Dies kann zum Beispiel ein Dokument oder eine Ware als physischer Ausgang sein. Zusätzlich sind Informationsobjekte bzw. Daten vorhanden, welche innerhalb der Funktionen beispielsweise erstellt oder verändert werden. Des Weiteren stehen Organisationseinheiten zur Verfügung, um die Zuständigkeit der Durchführung zuordnen zu können [142, S.17].

Nach [143, S.65] gehört die erweiterte EPK (eEPK) zu den am meisten verwendeten Methoden. Da diese durch die Anschaulichkeit und Nachvollziehbarkeit schnell verständlich ist, erlaubt sie eine gute Kommunikation zwischen verschiedenen Fachbereichen, die am Geschäftsprozess beteiligt sind. Zudem ist die eEPK durch die abstrakt gewählten Elemente der Funktion und des Ereignisses ein flexibles und domänenunspezifisches Modellierungsinstrument [142, S.19].

2.5.2.4 BPMN

Ebenfalls zu den kontrollflussorientierten Methoden gehört die Business Process Model and Notation (BPMN). Die ehemals unter der Bezeichnung „Business Process Modeling Notation (BPMN) Version 1.0“ in 2004 herausgegebene Spezifikation wurde von Stephen A. White als IBM Corporation Mitarbeiter erarbeitet und durch die Business Process Management Initiative (BPMI) veröffentlicht. Im Juni 2005 verkündeten die BPMI und OMG ihre strategische Fusionierung hinsichtlich der BPM-Aktivitäten. 2011 wurde die Version BPMN 2.0 mit einer umfangreicheren Sprache sowie einem Workflow-Management veröffentlicht, welche nachfolgend bei der ISO und IEC eingespeist und als Standard unter diesen herausgebracht werden konnte. BPMN verbindet als standardisierte Methode sowohl die betriebswirtschaftliche als auch technische Sicht auf Prozesse und bezieht zudem organisatorische und datenorientierte Aspekte ein [143, S.85ff].

Im Beitrag [55] von White wurde 2004 ein Vergleich der Modellierungsnotationen für die Aktivitätsdiagramme aus UML mit BPMN anhand von 21 ausgewählten Pattern durchgeführt. Ziel war, die Stärken und Schwächen beider Ansätze aufzuzeigen. Mit dem Ergebnis, dass beide Notationssprachen die meisten Pattern erfüllen, können somit ähnliche Lösungen erzielt werden. Es existieren jedoch Unterschiede beispielsweise in der Terminologie. Eine Analyse in [56] ergab 2008, dass die BPMN-Notation nicht nur einen bedeutenden Trend darstellt, sondern auch von Werkzeugen dieses Bereiches unterstützt werden muss. Daher wird im folgenden Kapitel auf den im Zuge von BPMN herausgegebenen Standard ISO/IEC 19510 detaillierter eingegangen.

2.5.3 Prozessmodellierung nach ISO/IEC 19510 - BPMN

Die BPMN[31], aktuelle Version 2.0.1 [BPMN], wurde durch die OMG standardisiert und formte die Basis für die ISO/IEC 19510:2013 [ISO19510] mit dem Titel „*Information technology - Object Management Group Business Process Model and Notation*“, publiziert im Juli 2013.

[31] http://www.bpmn.org/ (Zugriff am 15.10.2014)

2.5.3.1 Hintergrund und Zielsetzung von BPMN

Ziel der BPMN war, allen Beteiligten eine Notation einfach verständlich darzubieten und durch die Standardisierung das *business process design* mit der *process implementation* zu verbinden [ISO19510, 1.1]. Als zweites Ziel wird aufgeführt, dass XML-Sprachen zur Ausführung von *business processes* in einer geschäftsorientierten Notation dargestellt werden. Dies kann beispielsweise die Web Services Business Process Execution Language (WS-BPEL) sein.

Berücksichtigt werden in BPMN unter anderem die *Collaboration diagrams*, *Process diagrams*, und *Choreography diagrams*. Somit konnten unterschiedliche Modellierungsnotationen und Gesichtspunkte standardisiert und miteinander verschmolzen werden. Hierdurch wird eine einfache Kommunikationsbasis zwischen *business users* ermöglicht. Dazu wurden teilweise divergierende Notationen und Methoden untersucht, beispielsweise „UML Activity Diagram, UML EDOC Business Processes, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM, and Event-Process Chains (EPCs)“.

2.5.3.2 Überblick und Inhalte der einzelnen Kapitel

In Abb. 2.14 ist ein Überblick über die Dokumentenstruktur für ISO/IEC 19510 gegeben. Der Standard setzt sich aus insgesamt 15 Parts zusammen, wobei die Parts 8-11 den Kernteil bilden. Im Folgenden wird auf die einzelnen Kapitel kurz eingegangen und die inhaltlichen Grundzüge werden erläutert.

Part 1 *„Scope“*: Part 1 beschreibt die Herkunft und Ziele von BPMN und geht auf untersuchte Notationen und Methoden ein.

Part 2 *„Conformance“*: Konformität zu BPMN 2.0 darf angegeben werden, wenn alle Konformitätspunkte nach dem Standard erfüllt sind. Sollte dies nicht der Fall sein, so kann der Standard lediglich als Basis angegeben werden, nicht jedoch die vollständige Konformität. Hierzu werden vier unterschiedliche Konformitäten aufgelistet: *Process Modeling Conformance*, *Process Execution Conformance*, *BPEL Process Execution Conformance* und *Choreography Modeling Conformance*. Auf die Anforderungen hinsichtlich der Konformität wird in den einzelnen Abschnitten [ISO19510, 2.2, 2.3, 2.4, 2.5] eingegangen und diese in [ISO19510, 2.6] zusammengefasst.

Part 3 *„Normative References“*: Als normative Referenzen werden unter anderem die OMG UML und die OMG Meta Object Facility (MOF) aufgeführt. Eine Vielzahl an nichtnormativen Referenzen wie WS-BPEL sind zudem gelistet.

Part 4 *„Terms and Definitions“*: Hier wird darauf verwiesen, dass Begriffe und Definitionen im Anhang C [ISO19510, AnnexC] unter dem Glossar aufgelistet sind.

Part 5 *„Symbols“*: Part 5 existiert, ist allerdings nicht befüllt. Somit werden Symbole innerhalb des Standards [ISO19510] nicht festgelegt.

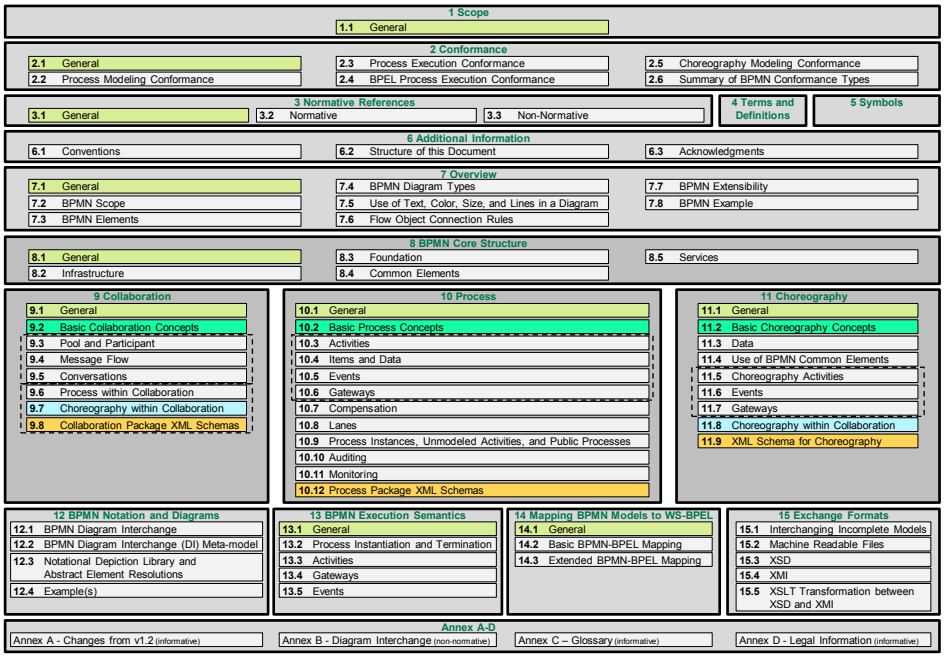


Abb. 2.14: Überblick über Dokumentenstruktur der ISO/IEC 19510 - eigene Darstellung nach Inhaltsverzeichnis und [ISO19510, 6.2]

Part 6 *„Additional Information“*: Als zusätzliche Informationen werden „typografische und sprachliche Konventionen und Styles“, die Struktur des Standards und Danksagungen gelistet.

Part 7 *„Overview“*: Part 7 gibt einen Überblick über die BPMN-Entwicklung der vergangenen Jahre und deren Anwendungsmöglichkeiten. Die Anwendung von BPMN 2.0.1 wird erläutert und die 3 grundlegenden Typen der Sub-Modelle genannt, darunter *Processes*, *Choreographies* sowie *Collaborations*. In Abschnitt [ISO19510, 7.3] werden die 5 Kategorien der BPMN-Elemente aufgelistet. Dazu gehören *Flow-Objects*, *Data*, *Connecting Objects*, *Swimlanes* und *Artifacts*. Zudem werden die Basic BPMN Modeling Elements und die Extended BPMN Modeling Elements aufgeführt, [ISO19510, 7.3.1] und [ISO19510, 7.3.2]. In den darauf folgenden Abschnitten [ISO19510, 7.4] und [ISO19510, 7.5] werden Alternativen der Diagrammtypen sowie für die Verwendung von Text, Farbe und Größe der Diagramme angegeben. Die *Flow Connection Rules* werden in [ISO19510, 7.6] erläutert. Part 7 stellt in den letzten beiden Abschnitten 7.7 und 7.8 die Erweiterungsfähigkeit von BPMN sowie ein Beispiel für die Anwendung vor.

Part 8 *„BPMN Core Structure“*: Die in Part 8 beschriebene *BPMN Core Structure* ist für alle Konformitätstypen erforderlich. Technisch gesehen wird ein Konzept von Erweiterungs-

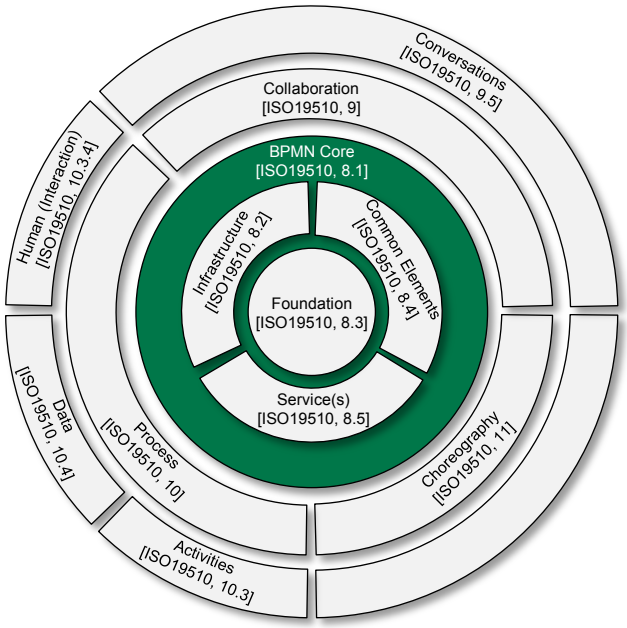


Abb. 2.15: Darstellung des BPMN Core und der Ebenenstruktur, angelehnt an [ISO19510, 8-Fig.1]

ebenen auf eine Menge von *Core Elements*, welche im *BPMN Core* verankert sind, angewandt. Der *Core* wird vorgesehen als „einfach, präzise und erweiterbar mit einem genau definierten Verhalten“ [ISO19510, 8.1]. Der *Core* beinhaltet die vier Sub-Pakete *Infrastructure* [ISO19510, 8.2], *Foundation* [ISO19510, 8.3], *Common Elements* [ISO19510, 8.4] und *Service* [ISO19510, 8.5], wie in Abb. 2.15 ersichtlich. *Infrastructure* beschreibt die Klassen *Definitions* und *Import*, welche als Elemente für abstrakte Syntax-Modelle und Diagrammmodelle verwendet werden. *Foundation* beinhaltet die grundlegenden Konstrukte für die Modellierung als Unterbau. Interessante Konstrukte werden im Folgenden dargestellt. Die abstrakte Superklasse *BaseElement* stellt ein Attribut *id* und die Klasse *Documentation* bereit, welche an alle Elemente vererbt werden [ISO19510, 8.3.1]. Eine *Extensibility* [ISO19510, 8.3.3] wird grundlegend über vier Klassen zur Verfügung gestellt: *Extension*, *ExtensionDefinition*, *ExtensionAttributeDefinition* und *ExtensionAttributeValue*. Erweiterungen des spezifizierten Metamodells durch Verwendung dieser, belässt es BPMN konform. Die abstrakte Superklasse *RootElement* [ISO19510, 8.3.5] erbt alle Attribute und Modell-Assoziationen von *BaseElement*. *Service* stellt die grundlegenden Modellierungskonstrukte für *services* und *interfaces* bereit. *Common Elements* stellt die Elemente zur Verfügung, welche unter anderem für *Process* [ISO19510, 10], *Choreography* [ISO19510, 11] und *Collaboration* [ISO19510, 9] genutzt werden können. Die Klasse

Artifact [ISO19510, 8.4.1] stellt eine *Association*, *Group* und *TextAnnotation* zur Verfügung. Die Klasse *Events* [ISO19510, 8.4.5] definiert ein Ereignis mit Auswirkung auf den Prozessfluss. Die Klasse *FlowElement* [ISO19510, 8.4.7] ist die abstrakte Superklasse für alle Prozesselemente und beschreibt mit der Klasse *FlowElementsContainer* [ISO19510, 8.4.8] eine Übermenge an Elementen. Zwei von vier Typen sind hierbei *Process* und *Sub-Process*. Des Weiteren werden *Gateways* [ISO19510, 8.4.9], *Item Definition* [ISO19510, 8.4.10], *Messages* [ISO19510, 8.4.11] sowie *Resources* [ISO19510, 8.4.12] beschrieben. Über die Klasse *SequenceFlow* [ISO19510, 8.4.13] kann die Reihenfolge von *Flow Elements* festgelegt werden.

Part 9 „*Collaboration*“: Durch Verwendung des Pakets *Collaborations* [ISO19510, 9.1] kann über *Pools* eine Interaktion von *Participants* unterstützt werden. Hierbei können die *Pools Prozesse* beinhalten. Wichtige Konstrukte und Klassen sind *Pool*, *Participant* [ISO19510, 9.3], *MessageFlow* [ISO19510, 9.4] und *Conversation* [ISO19510, 9.5].

Part 10 „*Process*“: Part 10 ist, bezogen auf die Seitenzahl, der umfangreichste Part. Hier wird auf die Beschreibung eines Prozessflusses eingegangen und insbesondere auf wesentliche Konstrukte für die Prozessmodellierung. Dies wird im Zuge von *BPMN Process Choreography Conformance*, *BPMN Process Execution Conformance* oder *BPMN BPEL Process Execution Conformance* jedoch nicht gefordert. Der Prozess setzt sich aus einem Graphen von *Flow Elements* zusammen und repräsentiert die Sequenz oder den Fluss von Aktivitäten. Die Flusselemente definieren eine endliche Ausführungssemantik. Folgende Arten von BPMN-Prozessen werden unterschieden: „Private Non-executable (internal) Business Processes; Private Executable (internal) Business Processes, Public Processes“ [ISO19510, 10.2.1]. Interessant für die vorliegende Arbeit ist die erste Kategorie. Diese kann über das Prozesselement *Process* durch die Attribute *processType* mit den Enumerationsliteralen (none, private, public) und *isExecutable* als boolscher Wert spezifiziert werden. Die privaten Prozesse werden üblicherweise mit Workflow oder BPM-Prozess bezeichnet. Als weiteres Synonym wird aus dem Bereich der Webservices die Orchestrierung als Service verwendet. Ein öffentlicher Prozess zeigt die Interaktion zwischen einem privaten und einem weiteren Prozess oder Beteiligten. Grafische Kernelemente sind die *Activities*, *Events* und *Gateways*. Für die ersten beiden werden die entsprechenden Definitionen aus dem Standard gegeben:

Definition 1 (Activities) *„An Activity is work that is performed within a Business Process. An Activity can be atomic or non-atomic (compound). The types of Activities that are a part of a Process are: Task, Sub-Process, and Call Activity, which allows the inclusion of re-usable Tasks and Processes in the diagram. However, a Process is not a specific graphical object. Instead, it is a set of graphical objects.“ [ISO19510, 10.3]*

Definition 2 (Events) *„An Event is something that „happens“ during the course of a Process. These Events affect the flow of the Process and usually have a cause or an impact and in general require or allow for a reaction. The term „event“ is general enough to cover many things in a*

Process. The start of an Activity, the end of an Activity, the change of state of a document, a Message that arrives, etc., all could be considered Events."[ISO19510, 10.5]

Part 11 „*Choreography*": Im Part 11 wird auf die Choreographie eingegangen, wobei es sich hier um einen speziellen Prozesstyp handelt [ISO19510, 7.2.1], der sich von dem Standard-Prozess abhebt und auf einen anderen Nutzen abzielt. Choreographien werden zwischen Pools angeordnet und repräsentieren beispielsweise ein Verfahrensvertrag zwischen den Beteiligten [ISO19510, 7.2.1]. Somit werden mit Hilfe von Aktivitäten die Interaktionen zwischen den Beteiligten dargestellt. Eine Interaktion kann einen mehrfachen Informationsaustausch repräsentieren und mehrere Beteiligte hinzubeziehen [ISO19510, 7.2.1]. Nach [ISO19510, 11.1] wird der Choreographie die Formalisierung der Art, wie Geschäftsbeteiligte ihre Interaktionen koordinieren, zugeschrieben. Somit kann dies auch aus der Sicht einer Art Geschäftsvertrag betrachtet werden. Choreographien sind derartig ausgelegt, dass sie für sich stehen können, aber auch innerhalb von Kollaborations-Diagrammen eingebracht werden können [ISO19510, 11.1]. Sie haben eine Beziehung und einen Zusammenhang zu den Kollaborationen, wobei es sich hierbei um das grundlegende Konzept handelt [ISO19510, 11.2]. Innerhalb der Choreographien werden keine *Data Objects* verwendet [ISO19510, 11.3]. Jedoch werden Elemente genutzt, welche auch beim Prozessdiagramm vorkommen, wie der *Sequence Flow*, *Text Annotations* und *Groups* [ISO19510, 11.4.1]. Durch ein *Participant Band* werden die Beteiligten dargestellt und der initiierende Beteiligte festgelegt [ISO19510, 11.5.1]. Ein Container kann zur Bildung einer Hierarchie eingesetzt werden. Gleichfalls können Ereignisse genutzt und Gateways eingebracht werden, um beispielsweise sequentielle oder parallele Markierungen zu setzen [ISO19510, 11.7]. Die Verwendung von Choreographien innerhalb von Kollaborationen ist möglich [ISO19510, 11.8], um somit auch *Swimlanes* (*Pools* und *Lanes*) dargestellt zu bekommen.

Part 12 „*BPMN Notation and Diagrams*": Dieser Part fokussiert sich auf den Austausch von BPMN-Diagrammen zwischen unterschiedlichen Tools. Hierzu wird das MOF-basierte BPMN Diagram Interchange (BPMN-DI) Metamodell bereitgestellt. Das BPMN-DI ermittelt nicht, ob das Diagramm eine syntaktische und semantische Korrektheit aufweist [ISO19510, 12.1]. Die Serialisierung und der Diagrammaustausch kann mit Hilfe der XML Metadata Interchange (XMI) durchgeführt werden. Zudem ist ein XML Schema Definition (XSD) für BPMN-DI festgelegt.

Part 13 „*BPMN Execution Semantics*": In diesem Part werden die Definitionen zur Orchestrierung von Prozessen dargestellt. Unterschieden wird zwischen operativen und nicht-operativen Elementen. Prozesse können durch ein *Start Event* instanziiert werden. Zudem wird ein sogenanntes Token auf den ausgehenden *Sequence Flows* erstellt [ISO19510, 13.2]. Durch das *End Event* erfolgt die Terminierung. Bedeutend für die Orchestrierung sind die einzeln beschriebenen Bereiche für die *Activities*, *Gateways* und *Events*.

Part 14 „*Mapping BPMN Models to WS-BPEL*": Eine Verknüpfung von BPMN-Modellen mit der WS-BPEL wird in Part 14 beschrieben. Hierzu werden Orchestrierungs-Prozesse

auf individuelle WS-BPEL-Prozesse abgebildet [ISO19510, 14.1]. Allerdings lassen sich nicht alle abbilden, so lässt beispielsweise WS-BPEL keine unstrukturierten Schleifen zu. Zudem dürfen unter anderem keine „Deadlocks“ oder ein „Mangel an Synchronisierung“ vorliegen.

Part 15 *„Exchange Formats“*: Durch XMI können auch unvollständige Modelle, welche bei einer iterativen Modellierungs-Vorgehensweise entstehen können, ausgetauscht werden. Dies ist bei der Implementierung zu berücksichtigen [ISO19510, 15]. Außerdem werden im Zuge von maschinenlesbaren Dateien die Formate XSD, XMI und XSLT, welche als „Extensible Stylesheet Language Transformation“ für die Umwandlung von XSD in XMI sowie umgekehrt dient, aufgelistet.

Anhänge: Im Anhang A (informativ) *„Changes from v1.2“*, Anhang B (nicht-normativ) *„Diagram Interchange“*, Anhang C (informativ) *„Glossary“*, Anhang D (informativ) *„Legal Information“* werden Informationen zu Notationsänderungen, technischen Änderungen, Diagramm Austausch (DI), Glossar und rechtliche Hinweise aufgeführt.

2.5.3.3 Beispiel einer Modellierung unter Verwendung von BPMN

Ein sehr anschauliches Beispiel für eine Modellierung unter Verwendung von BPMN wird für den Nobelpreisträger-Prozess für Medizin in einem von der OMG herausgegebenen nicht-normativen Dokument mit BPMN 2.0 konformen Beispielen dargestellt. Auf ausgewählte Inhalte der Prozessbeschreibung wird im Folgenden kurz eingegangen, vgl. [BPMNex, S.25]. Wie in Abb. 2.16 dargestellt, handelt es sich bei diesem Beispiel um einen zeitlich gesehen langwierigen Prozess. Dieser beginnt wie ersichtlich bereits im September des Vorjahres und schließt mit dem festgelegten Prozessende, der Nobelpreis-Verleihung. Der Festakt findet alljährlich am Todestag von Alfred Nobel, dem 10. Dezember, in Stockholm statt. Bedingt durch die zeitlichen Gegebenheiten von Prozessende und -anfang überlagern sich jeweils zwei angestoßene Prozesse. Zudem gilt es den vorgegebenen Prozessfluss strikt durchzuführen und zeitlich einzuhalten, bedingt durch den Stichtag der Preisverleihung.

Innerhalb des Gesamtprozesses sind insgesamt 5 unterschiedliche Akteure beteiligt, welche in den einzelnen Pools dargestellt sind: Nobelpreis-Komitee für Medizin (Nobel Committee for Medicine), Ernennende (Nominators), speziell ernannte Experten (Specially appointed experts), Nobelpreisversammlung (Nobel Assembly) und Nobelpreisträger (Nobel Laureates).

Das Nobelpreis-Komitee stellt das entscheidende Gremium dar und ist für die Initiierung des Prozesses verantwortlich. Es steht in einer vielfältigen Interaktion mit allen Prozessbeteiligten, beispielsweise durch den Versand von Nominierungsformularen an die Gruppe der Ernennenden und deren Rückantworten. Exemplarisch wird innerhalb des Prozesses auch die Verwendung von Entscheidungspunkten dargestellt. Bezogen auf das Beispiel kann optional ein assistierendes Expertenteam als weitere Instanz zu Rate ge-

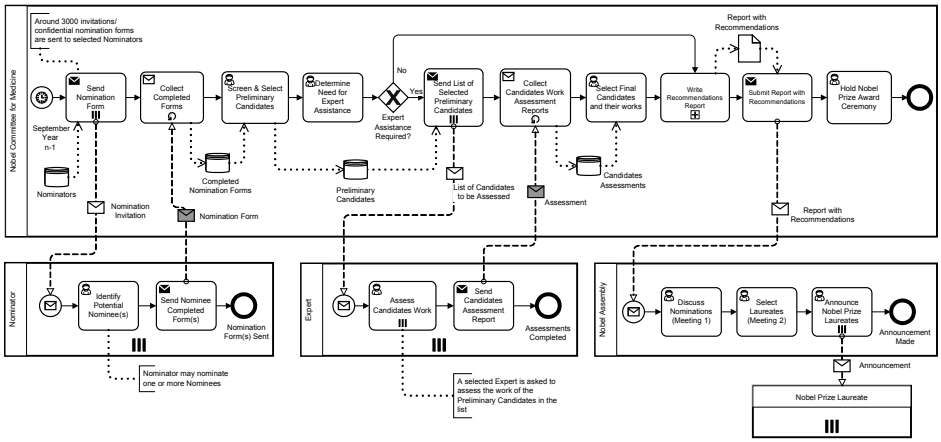


Abb. 2.16: Exemplarisches Prozessszenario für den Nobelpreis in Medizin, dargestellt in [BPMNex, S.26]

zogen werden. Die konsequente Dokumentation in Form von Berichten ist Grundlage für die prozessweite sowie übergreifende Kommunikation zwischen den einzelnen Akteuren. Endpunkte der einzelnen Prozesse sind jeweils gekennzeichnet. Die letzte Aktivität hat das Komitee bei der Vergabe des Nobelpreises inne.

2.5.3.4 BPMN-Modellaustausch

Die BPMN-Arbeitsgruppe „BPMN Model Interchange Working Group“ (BPMN MIWG) beschäftigt sich mit dem Austausch von BPMN-Modellen. Hierbei liegt das Augenmerk auf der Unterstützung von Werkzeuganbietern bei der Implementierung der Austauschschnittstelle, um Fehlinterpretationen aus dem Standard zu verhindern. Zugleich sollen Fehler im BPMN-Modellaustausch identifiziert und berichtet werden. Mit der Schaffung einer Demonstrationsinfrastruktur sollen Demonstrationsprozesse, Leitfäden und Validierungswerkzeuge eingeführt werden. Eine aktuelle Auflistung von untersuchten kommerziellen sowie Open Source Werkzeugen kann unter [57] eingesehen werden.

2.5.3.5 Business Process Execution Language (BPEL)

Business Process Execution Language (BPEL)[32] ist eine XML-basierte Ausführungssprache und unter Organization for the Advancement of Structured Information Standards (OASIS)[33] als gemeinnütziges Konsortium verankert [WSBPEL]. Als „domänen-

[32] http://bpel.xml.org (Zugriff am 15.10.2014)
[33] https://www.oasis-open.org (Zugriff am 15.10.2014)

spezifische, imperative Programmiersprache“ eignet sich WS-BPEL, um Aktivitäten von Geschäftsprozessen als Web-Services zu beschreiben. BPEL wird den Serviceorchestrierungs-Sprachen zugeordnet und bietet sich deswegen für Backend-Prozesse an, d.h. Geschäftsprozesse können automatisiert und ohne manuelle Unterstützung ausgeführt werden („Dunkelverarbeitung“). Ein BPEL-Prozess instrumentiert die Interaktionen zwischen verschiedenen Web-Services. Mit BPEL4People [BPEL4People], welche auf WS-HumanTask [WS-HT] basiert, werden Erweiterungen zur Verfügung gestellt, die den Fokus auf die menschliche Interaktion setzen, wodurch eine sogenannte „Hellverarbeitung“ möglich wird. Dies bezieht sich sowohl auf ausführbare als auch abstrakte BPEL-Prozesse.

2.6 Funktionale Sicherheit für Straßenfahrzeuge nach ISO 26262

Nach [132, S.9] wird die Funktionssicherheit „allgemein als eine korrekte technische Reaktion eines technischen Systems in einem definierten Umfeld, bei gegebener definierter Stimulation am Eingang des technischen Systems“ angesehen. Nach ISO 26262 ist funktionale Sicherheit wie folgt definiert: „absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems“ [ISO26262, 1-1.51]. Dies wird nach [132, S.9] mit der „Freiheit von unakzeptablen Risiken basierend auf Gefahren, die durch Fehlfunktion von E/E-Systemen verursacht werden“ übersetzt.

Die ISO 26262 konnte durch das Technische Komitee ISO/TC 22 für „Road vehicles“ im Sub-Komitee SC 3 für „Electrical and electronic equipment“ in der Arbeitsgruppe WG 16 für „Functional Safety“[34] auf internationaler Ebene erarbeitet werden, siehe Abb. 2.17. Der Standard hat zwei Klassifikationen im Zuge der International Classification for Standards (ICS) erhalten. Das aktive ISO-Gremium SC3/WG16 für Funktionssicherheit steht unter deutscher Federführung (Vorsitz und Sekretariat), siehe [58, S.4] und [59, S.32].

Die ISO 26262 ist eine domänenspezifische Adaption der IEC 61508 [IEC61508] und betrachtet funktionale Sicherheit für den Anwendungssektor von Straßenfahrzeugen [ISO26262]. Die Anwendung bezieht sich auf alle Aktivitäten innerhalb des Sicherheitslebenszyklus von sicherheitsbezogenen Systemen, welche Elektrik-, Elektronik- und Softwarekomponenten enthalten. Begründet wird dies mit der Bedeutung von Sicherheit in zukünftigen Fahrzeugen. Durch kommende neue Funktionalitäten für Fahrdynamikreglung und Fahrerassistenz ist das Fahrzeug zunehmend im Fokus der Systemsicherheit. Somit sind auch die dahinterliegenden Prozesse betroffen und die Erbringung eines Nachweises, dass sämtliche Sicherheitsziele erfüllt werden. Durch die technologische Komplexität, umfassende Softwareumfänge und das Einbringen zahlreicher mechatronischer Systeme erhöht sich dabei das Risiko von sowohl systematischen Fehlern als auch von zufälligen Ausfällen der Hardware. Die Absicherung wird durch die ISO 26262

[34] http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/iso_technical_committee.htm?commid=46706 (Zugriff am 15.10.2014)

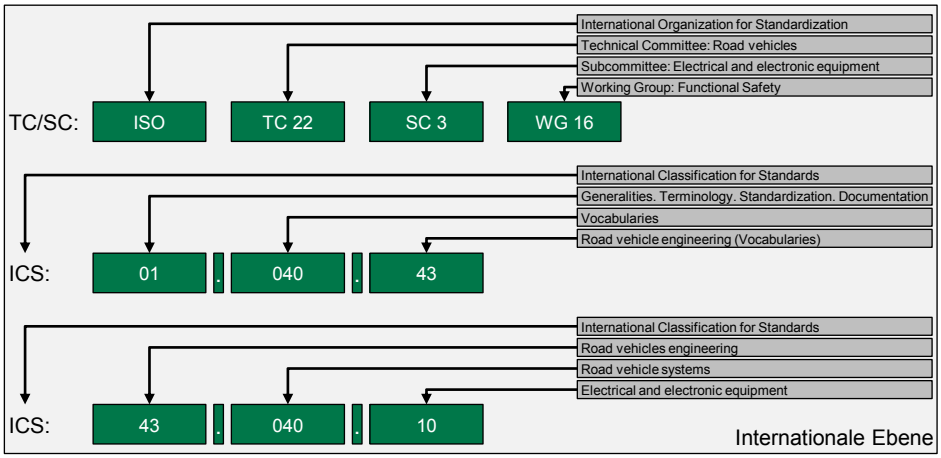


Abb. 2.17: Einordnung der ISO 26262 nach Fachgremium und ICS auf internationaler Ebene

mittels angemessenen Anforderungen und Prozessen geleitet. Erzielt wird diese durch Sicherheitsmaßnahmen, welche zu unterschiedlichen Zeitpunkten des Entwicklungsprozesses angewandt werden. Zugleich bietet die ISO 26262 ein Rahmenwerk in dem auch sicherheitsbezogene Systeme anderer Technologien betrachtet werden können. Dieses umfasst einen automobilen Sicherheitslebenszyklus und das maßgerechte Zuschneiden auf die einzelnen Phasen. Des Weiteren wird ein automobilspezifischer und risikobasierter Ansatz zur Bestimmung von Integritätsleveln bereitgestellt sowie Anforderungen für die Validierung und Bestätigungen von Maßnahmen zur Absicherung einer akzeptablen Sicherheit. Nicht zuletzt wird die Beziehung und Interaktion mit Zulieferern aufgegriffen. Funktionale Sicherheit wird durch den Entwicklungsprozess, Produktion und Kundendienstprozess sowie durch den Managementprozess bestimmt. Zugleich sind die Angelegenheiten hinsichtlich funktionaler Sicherheit sowohl mit den funktions- und qualitätsorientierten Entwicklungsaktivitäten als auch Arbeitsprodukten vernetzt. Die Strukturierung der ISO 26262 basiert auf dem Referenzprozessmodell des V-Modells bezüglich der Produktentwicklung.

Der Gültigkeitsbereich des Standards ISO 26262 ist für eine Anwendung auf sicherheitsbezogene Systeme im Automobilbereich für Serienkraftfahrzeuge kleiner 3,5 Tonnen Gesamtfahrzeuggewicht, die ein oder mehrere EE-Systeme beinhalten, festgelegt [ISO26262, 1]. Ausnahme bilden EE-Systeme welche in Spezialfahrzeugen, beispielsweise für Fahrer mit Behinderung, zum Einsatz kommen. Bei Systemen oder Komponenten, welche bereits vor der Publikation des Standards entwickelt oder freigegeben wurden, ist die Anwendung nicht verpflichtend. Selbiges gilt für Systemanteile, die aus vorherigen Serien wiederverwendet wurden. Hierbei ist lediglich eine Anwendung auf

die Änderungen verpflichtend. Inhaltlich werden mögliche Gefahren betrachtet, welche durch Funktionsstörungen der sicherheitsbezogenen Systeme oder deren Interaktion zustande kommen können, allerdings nicht Gefahren, bedingt durch beispielsweise UV-Strahlung, Schadstoffgehalt (Toxizität), Korrosion oder ähnliche Gefahren, solange diese nicht aus möglichen Störungen hervorgehen. „Nominal Performance“ von EE-Systemen steht ebenfalls nicht im Fokus, wenngleich dedizierte funktionale Performance-Standards vorliegen. Dies betrifft unter anderem aktive/passive Sicherheitssysteme, Bremssysteme oder Adaptive Cruise Control (ACC).

Bei der ISO 26262 handelt es sich um ein durch viele Beteiligte gemeinschaftlich erarbeitetes Dokument bei dem Kompromisse bei der Beschreibung eingegangen werden mussten, vgl. [132, S.1]. Für die Anwendung ist daher nicht nur eine Analyse der relevanten Parts notwendig, sondern eine entwicklungsspezifische Interpretation und Deutung muss ebenfalls angegangen werden. Dies muss gegebenenfalls unter den Gesichtspunkten der jeweiligen Firma betrachtet werden.

2.6.1 Historie und offizielle Bekanntmachungen durch den Normenausschuss

Der Normenausschuss Automobiltechnik (NAAutomobil), ehemals FAKRA, ist eine Institution des VDA und des DIN. Sowohl organisatorisch, finanziell als auch personell ist dieser dem VDA zuzuordnen [48, S.6]. Der Hauptfokus liegt auf der Mithilfe bei der Erarbeitung internationaler Standards [60]. Die Normungsaktivitäten zur ISO 26262 sind auf nationaler Ebene innerhalb des Normenausschusses (NA) Automobil verankert, wie in Abb. 2.18 ersichtlich.

Die Historie von Standards sowie aktuelle Aktivitäten des Normenausschusses (NA) und der Arbeitskreise (AKs) innerhalb der Arbeitsausschüsse (AAs) können aus den Berichten der Arbeitsgremien in den jeweiligen Jahresberichten „Auto & Normung“, publiziert vom VDA, nachgelesen werden.

Jahresbericht 2006: Im Rahmen des Jahresberichtes von 2006 ist ersichtlich, dass der Arbeitskreis 16 darauf abzielte, einen für die Automobilindustrie „tauglichen, handhabbaren und international abgestimmten Sicherheitsstandard als anwendungsspezifische Ableitung aus der DIN EN 61508“ hervorzubringen [58, S.24]. Hierbei wurden für den angestrebten Standard die Fahrzeugklassen M, N und O[35] angedacht. Eine Auflistung der unterschiedlichen EG-Fahrzeugklassen kann aus [61, L 263/62], [62, S.931] und [63, S.6] entnommen werden.

Zum damaligen Stand, Januar 2007, befand sich „das Projekt im Stadium eines vorläufig registrierten Work-Items (PWI) bei ISO TC22“ und war kurz vor der Antragsstellung

[35] M: „Für die Personenbeförderung ausgelegte und gebaute Kraftfahrzeuge mit mindestens vier Rädern.“
N: „Für die Güterbeförderung ausgelegte und gebaute Kraftfahrzeuge mit mindestens vier Rädern.“
O: „Anhänger (einschließlich Sattelanhänger).“ [61, L 263/62]

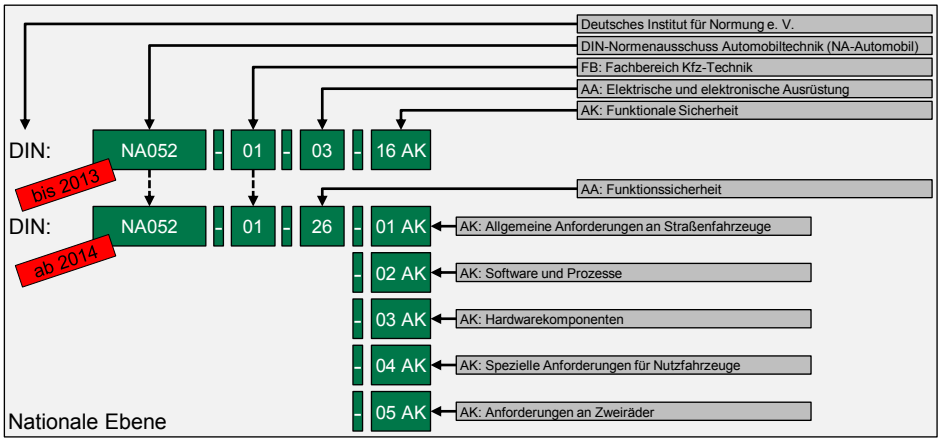


Abb. 2.18: Einordnung der ISO 26262 im Normenausschuss Automobiltechnik auf nationaler Ebene

einer „Erteilung eines internationalen Normungsvorhabens“, um somit nachfolgend die „Eröffnung der CD-Umfrage“ einzuleiten. Angemerkt sei, dass die geplante Struktur von Part 1 bis Part 8 bis zur späteren Veröffentlichung beibehalten wurde. Part 9 und Part 10 wurden als Erweiterungen zur ursprünglichen Planung von informativen Annexen eingebracht.

Jahresbericht 2011: Im Rahmen des Jahresberichtes von 2011 wurde mitgeteilt, dass nach einer „zweijährigen vorläufigen Bearbeitung“ das Projekt im Juni 2007 „auf ISO-Ebene“ „bei ISO/TC22 als Normenprojekt angenommen“ wurde. „Anfängliche[n] Widerstände[n] aus diversen Ländern“ konnten bereinigt werden und eine große Beteiligung erfolgte. Für den Anwendungsbereich werden „Straßenfahrzeuge bis 3,5 t Gesamtgewicht“ genannt, wobei eine Anwendung auf andere Kraftfahrzeugklassen nicht ausgeschlossen wird. Zu diesem Zeitpunkt konnte das Projekt auf insgesamt 10 Parts ausgelegt werden, welches sich am Sicherheitslebenszyklus orientiert. Die FDIS-Umfrage wurde im April 2011 eingereicht und im Juni 2011 angenommen. Mit einer Veröffentlichung wurde somit für Ende 2011 gerechnet [59, S.32ff].

Hinsichtlich kommender Aktivitäten wurde mitgeteilt, dass die Arbeiten bis Mitte 2012 ruhen würden, um im Anschluss an die voraussichtliche Veröffentlichung mit einer Überarbeitung in Form einer Revision zu beginnen, welche eine Erweiterung der Anwendung auf schwere Fahrzeuge vorsieht. Für den Beginn der Revision in 2014 sollen bereits erste Erkenntnisse aus den Erfahrungen der ISO 26262 einfließen.

Jahresbericht 2012-2013: Im Rahmen des Jahresberichtes von 2012/2013 wird von der Publikation Ende 2011 der 9 normativen Parts der ISO 26262 und dem Anfang 2012 publizierten Part 10 als „Leitfaden“ „mit informellem Charakter“ berichtet. Eine Anwendung

des Standards ISO 26262 kann auch für andere Kraftfahrzeuge genutzt werden. Es wird darauf hingewiesen, dass bereits nach der ersten Ausgabe der ISO 26262 der Arbeitskreis mit der „Erweiterung des Anwendungsbereiches auf schwere Nutzfahrzeuge größer 3,5 t Gesamtgewicht und auf einspurige Kraftfahrzeuge ausgerichtet ist“ [64, S.35].

Jahresbericht 2014: Im aktuellen Jahresbericht 2014 [48, S.57] wird insbesondere der „Austausch von Erfahrungen und Erkenntnissen bei der Anwendung des Standards“ und die Vorbereitung einer Revision, welche auch von internationaler Seite gewünscht wird, hervorgehoben. Als ein zentrales Thema wird die Ausweitung des Anwendungsbereiches auf „Motorräder und schwere Nutzfahrzeuge“ gesehen. Ein weiteres Kernthema ist „die Einführung von Festlegungen für Halbleiterbauelemente“. Des Weiteren werden inhaltlich Änderungen auf den Erfahrungen der bisherigen Anwendung des Standards basieren.

Bedingt durch die zukünftigen Erweiterungen der Anwendungsgebiete, wurde die Thematik aus dem „NA052-01-03 herausgelöst“ und ein „eigener Ausschuss, der NA052-01-26 gegründet“, siehe Abb. 2.18. Dieser setzt sich aus insgesamt 5 Arbeitskreisen zusammen, siehe hierzu auch die Übersicht der Arbeitsgruppen unter [48, S.5]: Der „AK26-01: Allgemeine Anforderungen an Straßenfahrzeuge“ beschäftigt sich unter anderem mit den Erfahrungen aus der ersten Ausgabe, um gegebenenfalls Änderungen für die zweite Ausgabe einzubringen. Zudem werden Diskussionen hinsichtlich noch nicht erfasster Themen geführt. Im Rahmen des „AK26-02: Software und Prozesse“ werden „spezielle Anforderungen“ für „Software und software-gesteuerte Prozesse“ erarbeitet. Der Arbeitskreis basiert auf einem Unterarbeitskreis von Part 6. Der Arbeitskreis „AK26-03: Hardwarekomponenten“ beschäftigt sich mit der Erarbeitung der ISO PAS 19451. Dies spiegelt die „Anforderungen an Halbleitersysteme“ wieder, welche in die zweite Revision der ISO 26262 übernommen werden sollen. Angemerkt sei, dass bereits ein neues Projekt im TC22/SC3 Arbeitsprogramm gelistet ist, welches unter der Bezeichnung „ISO/AWI PAS 19451“ läuft und den Titel „Application of ISO 26262:2011-2012 to Semiconductors“ trägt. Innerhalb von „AK26-04: Spezielle Anforderungen für Nutzfahrzeuge“ werden schwere Nutzfahrzeuge, wie beispielsweise „Autodrehkran bis hin zum Müllfahrzeug“, in den Fokus gesetzt und identifiziert, an welchen Stellen Änderungen oder Erweiterungen innerhalb der Norm für diese Variationen erforderlich sind. Der Arbeitskreis „AK26-05: Anforderungen an Zweiräder“ erarbeitet gemeinsam mit ISO TC22/SC27 eine Publicly Available Specification (PAS) für die Anwendung auf Motorräder, welche ebenfalls in die zweite Revision der ISO 26262 einfließen soll.

2.6.2 Struktureller Aufbau

Die ISO 26262 ist in 10 Parts aufgeteilt, wobei Part 1-9 bereits im November 2011 den Status „International Standard published“ erhalten haben, hingegen Part 10 erst im August 2012 diesen Status bekommen hat. Somit sind alle Parts normativ. Die Strukturierung des Gesamtdokuments erfolgt auf oberster Ebene durch die einzelnen Parts.

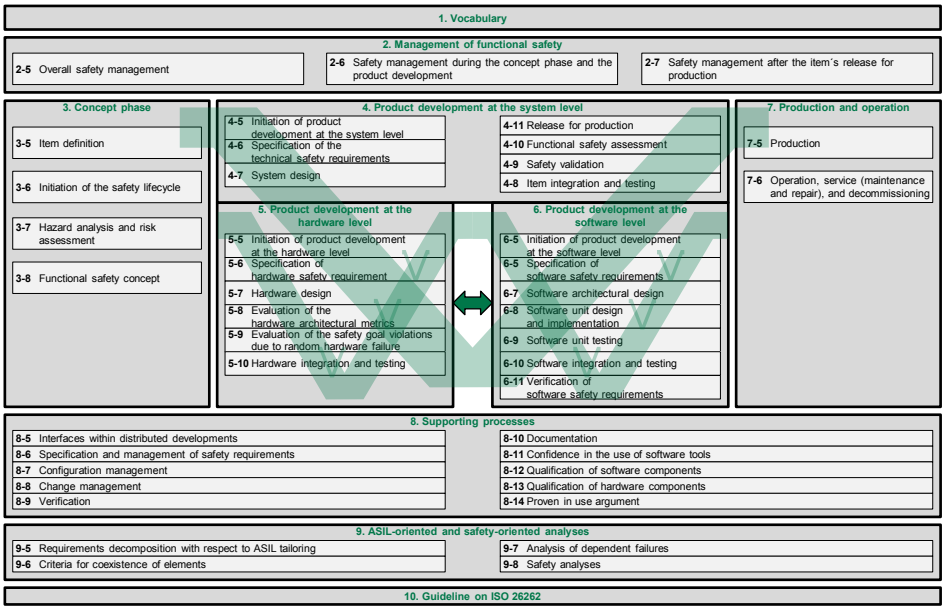


Abb. 2.19: Überblick über Dokumentenstruktur der ISO 26262 - eigene Darstellung von [ISO26262, 10-Fig.1]

2.6.3 Inhalte der ISO 26262

Im Folgenden wird auf für diese Arbeit wichtige Inhalte der ISO 26262 eingegangen und die Inhalte kurz anhand der einzelnen Parts beschrieben. Insbesondere Part 5 und Part 10 stehen im Fokus dieser Arbeit.

Part 1 *„Vocabulary“*: In Part 1 sind für sämtliche Parts der ISO 26262 Begriffe, deren Definitionen und Abkürzungen festgehalten.

Part 2 *„Management of functional safety“*: In Part 2 werden sowohl projektunabhängige als auch projektspezifische Anforderungen an das Management von funktionaler Sicherheit spezifiziert. Hierbei werden die einzelnen Phasen und Sub-Phasen als Überblick für den Sicherheitslebenszyklus beschrieben. Auf oberster Ebene erfolgt die Einteilung in „Concept Phase“ (abgedeckt durch Part 3), „Product development“ (abgedeckt durch Part 4-6 und Teilen von Part 7) sowie „After the release for production“ (abgedeckt durch Part 7). Für die ersten beiden Phasen werden die Sicherheitskultur (safety culture) [ISO26262, 1-1.107], Kompetenz- und Qualitätsmanagement sowie projektunabhängige Aufteilungen des Sicherheitslebenszyklus aufgegriffen, um somit zu organisationsspezifischen Regeln und Prozessen zu gelangen. Zusätzlich wird auf Rollen, Verantwortlichkeiten sowie Planung und Koordination der Sicherheitsaktivitäten und deren Aufteilung

(Tailoring) eingegangen. Der „Safety Case“ als Sicherheitsnachweis wird eingeführt sowie das Audit und die Beurteilung der funktionalen Sicherheit. Für die letzte Phase werden diese in den Kontext der Feldüberwachung gesetzt.

Part 3 *„Concept phase“*: In diesem Part werden die Anforderungen bezogen auf die Konzeptphase dargestellt. Dies betrifft die „Item definition“, um die Abhängigkeiten und Interaktionen mit der Umwelt zu beschreiben und zusätzlich ein Verständnis für das Fahrzeugsystem, im Englischen Item, zu erarbeiten, um nachfolgende Schritte entsprechend einleiten zu können. Bei der „Initiation of the safety lifecycle“ ist die Unterscheidung zwischen einem neuen Item und der Modifikation eines Existierenden entscheidend. Durch Anwendung einer HARA können Gefährdungen identifiziert und in Automotive SIL (ASIL) eingestuft werden. Daraus lassen sich wiederum die Sicherheitsziele, im Englischen Safety Goals (SGs), ableiten und definieren, um somit durch Maßnahmen die Gefährdungen gezielt abzuschwächen oder zu verhindern. Aus den SGs werden entsprechend die funktionalen Sicherheitsanforderungen, im Englischen Functional Safety Requirements (FSRs), abgeleitet und diese mit den vorläufigen Architekturelementen verknüpft, um nun zu einem sogenannten funktionalen Sicherheitskonzept, im Englischen Functional Safety Concept (FSC), zu gelangen.

Part 4 *„Product development at the system level“*: Der Part 4 ist der erste und übergeordnete Teil, welcher die Produktentwicklung aufgreift. Dabei wird als erstes die Systemebene betrachtet und die entsprechenden Aktivitäten auf dieser Ebene definiert. Nachfolgend werden die technischen Sicherheitsanforderungen, im Englischen Technical Safety Requirements (TSRs), spezifiziert, welche die FSRs erfüllen müssen und dadurch das FSC verfeinert wird. Damit wird das Systemdesign und das technische Sicherheitskonzept, im Englischen Technical Safety Concept (TSC), erstellt, gegebenenfalls durch ein iteratives Vorgehen. Hierbei werden unter anderem Maßnahmen zur Verhinderung von systematischen Fehlern und zufälligen Ausfällen der Hardware, Allokation auf Hardware und Software sowie die Hardware-Software Schnittstelle, im Englischen Hardware-Software Interface (HSI), berücksichtigt. Bei der Integration werden drei Phasen dargestellt. Die Erste integriert Hardware und Software zu einem Element, die zweite Phase Elemente zu einem Item und die Dritte das Item mit weiteren Systemen und dem Fahrzeug. Bei der Prüfung müssen alle Sicherheitsanforderungen abgedeckt und erfüllt sein. Bei der Sicherheitsvalidierung muss der Nachweis erbracht werden, dass die SGs korrekt, vollständig und erfüllt sind sowie das FSC entsprechend angemessen ist. Nachfolgend kann eine Beurteilung der funktionalen Sicherheit stattfinden und bei erfolgreichem Ergebnis die Produktion freigegeben werden.

Part 5 *„Product development at the hardware level“*: Durch eine Initiierung der Produktentwicklung auf Hardwareebene werden die entsprechenden Aktivitäten für funktionale Sicherheit zunächst erfasst und im Sicherheitsplan, im Englischen Safety Plan, festgehalten. Die Hardware-Sicherheitsanforderungen werden abgeleitet und mit dem TSC verifiziert. Zudem wird das HSI detailliert. Im Folgenden kann das *Hardwaredesign* erstellt werden, bei dem zwischen dem *Hardware-Architekturdesign* und dem *detaillierten*

Hardwaredesign unterschieden wird. Für die zwei Designphasen sind sich ergänzende Sicherheitsevaluationen durchzuführen, um nach Erfüllung der Anforderungen eine Hardware-Integration und -Überprüfung vorzunehmen. Angemerkt sei, dass auf Grundlagen im Kontext von Hardware-Sicherheitsevaluationen zusätzlich im Kapitel 2.7 eingegangen wird.

Part 6 *„Product development at the software level“*: Auch hier müssen die Aktivitäten für funktionale Sicherheit initiiert, die Software-Sicherheitsanforderungen erstellt und das HSI detailliert werden. Nach der Erstellung eines *„Software Architectural Designs“* kann das *„Software Unit Design“* und dessen Implementierung vorgenommen werden. Schließlich folgt die Unit-Überprüfung sowie die Software-Integration und -Überprüfung. Durch die Verifikation gegen die Software-Sicherheitsanforderungen kann zur Systemintegration übergegangen werden.

Part 7 *„Production and operation“*: Part 7 beschreibt die Phase „After the release for production“. Ausgearbeitet werden muss ein Produktionsprozess für die sicherheitsbezogenen Elemente, welcher entsprechend gepflegt werden muss. Für den Prozess müssen auch alle Produktionsbeteiligte wie OEM und Zulieferer berücksichtigt werden. Des Weiteren müssen Kundeninformationen, Wartungs- und Reparaturanweisungen sowie Anweisungen zur Demontage ausgearbeitet werden.

Part 8 *„Supporting processes“*: Im Rahmen von Part 8 wird auf Schnittstellen hinsichtlich verteilter Entwicklungen Bezug genommen. Zudem werden Anforderungen an die Spezifikation und das Management von Sicherheitsanforderungen gestellt, welche sich über den gesamten Produktlebenszyklus erstrecken. Dies sind beispielsweise eine hierarchische Struktur, Ableitung der Anforderungen, Festhalten der Abhängigkeiten und Beziehungen sowie Hinterlegen von Attributen und Charakteristiken. Für das Konfigurations- und Änderungsmanagement wird unter anderem eine klare Nachvollziehbarkeit und Reproduktion gefordert. Die Verifikation dient einer Absicherung, dass die entstandenen Arbeitsprodukte mit den erstellten Anforderungen übereinstimmen. Auch hinsichtlich der Dokumentation wird eine Strategie für eine klare Vorgehensweise gefordert. Schließlich wird noch auf die Qualifikation von sowohl Software- als auch *Hardware-Komponenten* eingegangen und über „Proven in use argument“ der Umgang und die Anforderungen an existierende Items festgehalten.

Part 9 *„Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses“*: Regeln und Anleitungen werden hinsichtlich der Dekomposition von Anforderungen im Zuge des ASIL-Tailoring gegeben. Zudem werden „Criteria for coexistence of elements“, welche den Umgang zwischen sicherheitsbezogenen Sub-Elementen mit Sub-Elementen, die keinen oder einen anderen ASIL besitzen, dargestellt. Außerdem wird auf die Analyse von abhängigen Ausfällen eingegangen. Ein weiterer Teil sind die Sicherheitsanalysen, welche die Konsequenzen von Abweichungen und Ausfällen untersuchen, und gegebenenfalls bisher nicht identifizierte Gefährdungen lokalisieren.

Part 10 *„Guideline on ISO 26262“*: Part 10 dient als Richtlinie und es wird insbesondere darauf hingewiesen, dass bei Inkonsistenzen zu den anderen Parts die Anforderungen aus diesen Vorrang haben. Aufgegriffen werden die Schlüsselkonzepte der ISO 26262, auch in Verbindung mit der IEC 61508, und ausgewählten Themen betreffend des Sicherheitsmanagements. Des Weiteren werden die Inhalte der Konzept- und Systementwicklungsphase anhand von Beispielen dargestellt. Im Fokus stehen außerdem die Sicherheitsanforderungen sowie die Hardwareentwicklung. Zu guter Letzt werden das „Safety element out of context“, das „Proven in use argument“ und die „ASIL decomposition“ erneut aufgegriffen.

2.6.4 Herausforderung bei der Anwendung des Standards

Die Vielschichtigkeit der ISO 26262 spiegelt sich bei der Anwendung und den damit verbundenen Herausforderungen wider, welche in Fachkreisen häufig diskutiert werden. Ein Auszug an Diskussionsthemen wird in den nachfolgenden Abschnitten gegeben.

Ein gemeinsames Verständnis für den Standard ist eine der Grundvoraussetzungen, um konzeptionell und methodisch einen Fortschritt zu erzielen. Durch das Aufgreifen des gesamten Produktlebenszyklus von einer elementaren Komponente bis zum gesamten Fahrzeug, von Software über Hardware bis hin zum System, ist dies durch die Vielzahl an Fachbereichen nicht leicht zu bewerkstelligen. Bedingt dadurch, dass sich häufig durch unternehmensinterne Methoden oder Prozesse ein anderes Verständnis von bestimmten Sachverhalten aufgebaut hat, ist das zudem nicht selbstverständlich zu realisieren. Schwierigkeiten können bereits bei der Auslegung von Definitionen auftreten, beispielhaft sei hier die Definition des „Items“ genannt, bei dem je nach Unternehmen auch unterschiedliche Sichten vorliegen, da deren Komponenten oder Systemgrenzen anders festgelegt werden. Hierbei stellt sich wiederum die Frage, an welchen Stellen die Grenzen gezogen werden dürfen, um durch deren Festlegung den Wirkungsbereich für den Standard einzugrenzen.

Eine Integration der vorgeschriebenen Anforderungen und Prozesse aus der ISO 26262 in bestehende Unternehmensprozesse, welche sich über Jahre etabliert haben, stellt eine große Herausforderung dar, insbesondere wenn das Alltagsgeschäft fließend weiterlaufen muss. Eine der offensichtlichsten Fragen ist in diesem Zusammenhang, wie bereits in Unternehmen abgedeckte Prozesse betreffend der ISO 26262 identifiziert und gegebenenfalls angepasst werden können.

Die Nachvollziehbarkeit ist einer der grundlegendsten Aspekte von funktionaler Sicherheit. Es stellt sich die Frage, wie diese konsistent gewährleistet werden kann, wenn sie in den Kontext des Alltagsgeschäftes eines umfangreichen Projektes gesetzt und beispielsweise hinsichtlich Sicherheitsanforderungen betrachtet wird. Nach Anlaufen der Produktion ist der Wirkbereich der ISO 26262 noch nicht abgeschlossen, weshalb der Umgang mit dem Standard während des gesamten Produktlebenszyklus betrachtet werden muss.

Eine weitere Herausforderungen ergibt sich aus der Kollaboration zwischen OEM und Zulieferern. Es muss definiert werden, welche Vereinbarungen getroffen werden müssen und festgelegt werden, was gegebenenfalls offengelegt werden muss. Zu klären ist, inwiefern an bestimmten Stellen die Entwicklungen verschmelzen müssen. Dies betrifft die Schnittstellen zwischen Unternehmen, vgl. die Entwicklungsschnittstellen-Vereinbarung, im Englischen Development Interface Agreement (DIA). Dieses Unterfangen ist unter anderem aber auch eine Frage der unternehmenseigenen IP.

Weitreichend ist auch die Frage nach der Produkthaftung und wie aus rechtlicher Sicht der Standard gehandhabt wird. Grundsätzlich ist von Interesse, wer die Verantwortung trägt, ob diese beim Auftragnehmer oder Auftraggeber liegt, beziehungsweise diese anteilhaftig ist. Ein wichtiger Aspekt ergibt sich aus der Frage, wie ein Zulieferer die funktionale Sicherheit weiterreichen kann, insbesondere wenn er selbst Einkäufer zum Beispiel bei einem Halbleiterhersteller ist und auf dessen Informationen angewiesen ist, beziehungsweise auf diese vertrauen muss.

Eine weitere zu klärende Fragestellung ergibt sich aus einer Wiederverwendung von Fahrzeugsystemen für nachfolgende Projekte. Hier gilt es zu klären, welche Teile aus Vorserien übernommen werden dürfen, beziehungsweise inwiefern sich der Umgang mit Änderungen eines Produktes im Rahmen einer Wiederverwendung gestaltet. Hier wird auch das Variantenmanagement in den Fokus gerückt.

Eine abschließende Herausforderung ist, den Entwicklern eine bessere Toolunterstützung darzureichen und gleichzeitig lange Toolketten zu eliminieren oder durch bessere Kopplungen zu harmonisieren. Hierbei ist auch von Interesse, wie in einem Zertifizierungsprozess die bereits vorhandenen Entwicklungsdaten als Nachweisdokumente erbracht werden können, um zu einem Zertifikat zu gelangen. Aber auch hier ist der Weg bis zum „ISO 26262 Zertifikat“ nicht direkt offensichtlich.

2.7 Funktionale Sicherheit im Kontext Hardware

Innerhalb dieses Kapitels werden hinsichtlich der Produktentwicklung von funktional sicherer Hardware sowohl Grundlagen aus der ISO 26262 als auch weiterreichende Informationen dargestellt. Der Fokus liegt auf den von der ISO 26262 geforderten Hardware-Sicherheitsevaluationen und wird dementsprechend abgegrenzt.

2.7.1 Hardwaredesignprozess nach ISO 26262

In der vereinfachten Darstellung nach Abb. 2.20 sind die Aktivitäten und Arbeitsprodukte ersichtlich, welche für die Produktentwicklung auf Hardwareebene nach Part 5 gefordert sind.

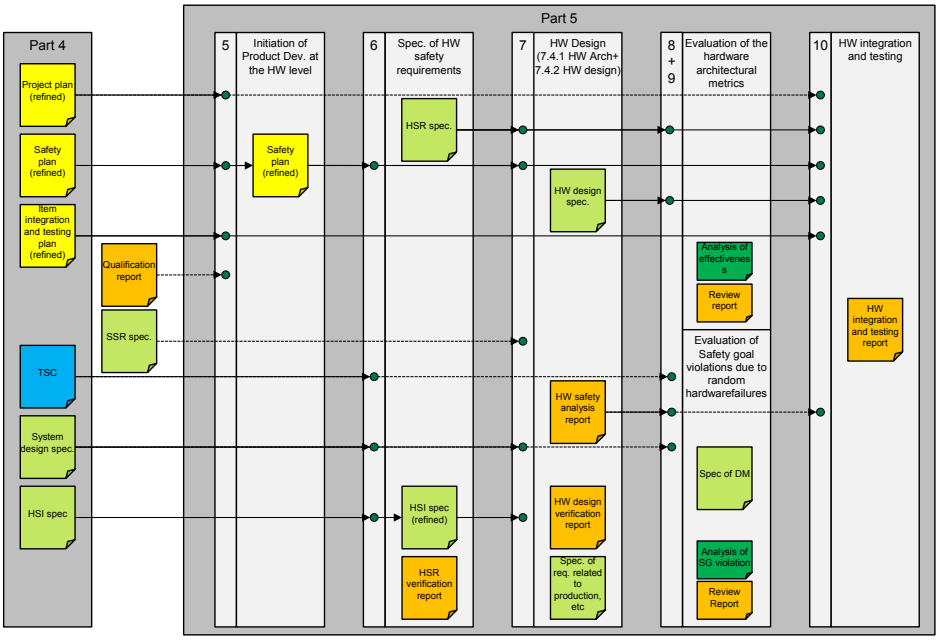


Abb. 2.20: Geforderte Arbeitsprodukte im Kontext der einzelnen Aktivitäten aus Part 5

Im Fokus stehen die Spezifikation der Hardware-Sicherheitsanforderungen nach Clause 6, die Entwicklung des *Hardwaredesigns* nach Clause 7 sowie die Absicherung gegenüber zufälligen Ausfällen der Hardware im Rahmen der Hardware-Sicherheitsevaluationen nach Clause 8 und Clause 9.

2.7.1.1 Sicherheitsanforderungen

Ein Sicherheitsziel, im Englischen Safety Goal (SG) [ISO26262, 1-1.108], definiert eine Top-Level Sicherheitsanforderung für das Fahrzeugsystem [ISO26262, 3-7.4.4.3.note], welche aus einer im Rahmen der HARA identifizierten Gefährdung abgeleitet wird. Für jede Gefährdung wird eine Klassifizierung in ein ASIL vorgenommen, welche dem abgeleiteten Sicherheitsziel zugewiesen wird [ISO26262, 3-7.4.4.3]. Diese wird aus folgenden Parametern bestimmt: Schwere eines potentiellen Schadens, im Englischen Severity of potential harm (S); Eintrittswahrscheinlichkeit, im Englischen probability of Exposure (E); Beherrschbarkeit, im Englischen Controllability (C). Für die einzelnen Parameter sind verschiedene Klassen zur Einstufung festgelegt, beispielsweise S0 bis S3 für die Schwere eines potentiellen Schadens. Nach individueller Einstufung aller Parameter kann aus der Tabelle nach [ISO26262, 3-Tab.4] der entsprechende ASIL abgeleitet werden. Nach

[ISO26262, 3-7.4.4.1] sind insgesamt vier ASILs definiert, ASIL-A bis ASIL-D, wobei ASIL-A das niedrigste Integritätslevel repräsentiert, ASIL-D das Höchste. Zusätzlich wird eine Klasse des Qualitätsmanagements (QM) eingeführt. Durch diese wird gekennzeichnet, dass keine besonderen Anforderungen in Bezug auf funktionale Sicherheit erfüllt werden müssen.

Sicherheitsziele werden im Laufe der Entwicklung des Systems hierarchisch verfeinert. Hierbei werden beispielsweise funktionale Sicherheitsanforderungen, im Englischen Functional Safety Requirements (FSRs) [ISO26262, 1-1.53], für das System abgeleitet. Diese spezifizieren das implementierungsunabhängige Sicherheitsverhalten oder notwendige Sicherheitsmaßnahmen, um die übergeordneten Sicherheitsziele zu erfüllen. Abgeleitet aus den Sicherheitszielen wird die entsprechende Klassifizierung des ASIL übernommen. Für jedes Sicherheitsziel soll mindestens eine funktionale Sicherheitsanforderung spezifiziert sein [ISO26262, 3-8.4.2.2].

Eine Verfeinerung der FSR stellen die technischen Sicherheitsanforderungen dar, im Englischen Technical Safety Requirement (TSR) [ISO26262, 1-1.133], welche aus den funktionalen Sicherheitsanforderungen abgeleitet werden und implementierungsspezifische Inhalte aufgreifen. Diese können anschließend beispielsweise in Hardware- oder Software-Sicherheitsanforderungen heruntergebrochen werden.

2.7.1.2 Hardware-Abstraktionsebenen

Im Kontext der ISO 26262 werden in Part 5 zwei verschiedene Abstraktionsebenen für *Hardwaredesigns* vorgestellt [ISO26262, 5-7.4]. Eine vereinfachte Darstellung der unterschiedlichen Abstraktionsebenen ist in Abb. 2.21 abgebildet. Das *Hardwaredesign* mit seinen *Hardware-Elementen* bildet die übergeordneten Termini. Das *Hardware-Architekturdesign* wird durch *Hardware-Komponenten* und deren Interaktion beschrieben. In diesem Zusammenhang ist insbesondere die Beschreibung von Wirkketten in Form von Input-Output-Beziehungen von Interesse. Das *Hardware-Architekturdesign* stellt somit eine höhere Abstraktionsebene gegenüber dem *detaillierten Hardwaredesign* dar, welches sich auf der Ebene von elektronischen Schaltplänen befindet und aus *Hardware-Bauteilen* zusammengesetzt ist. Bei den *Hardware-Bauteilen* handelt es sich auf der detaillierten Ebene um einzelne elementare Bausteine wie Widerstände und Kondensatoren, aber auch um komplexe Bausteine wie Application-Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) oder Microcontrollers (uCs). In [147, S.303] wird im Kontext von Softwareentwicklung analog von Architekturdesign und Feindesign gesprochen.

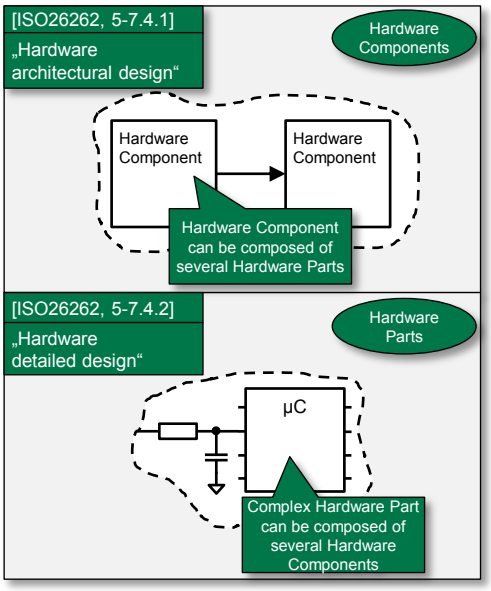


Abb. 2.21: Vereinfachte Darstellung der zwei Abstraktionsebenen für Hardwaredesigns

2.7.2 Sicherheitsnachweis

Der in ISO 26262 beschriebene Begriff des Sicherheitsnachweises, im Englischen Safety Case [ISO26262, 1-1.106], repräsentiert ein Argument und bezieht sich auf die Sicherheitsanforderungen eines Fahrzeugsystems. Der Sicherheitsnachweis soll entwicklungsbegleitend alle während des Sicherheitslebenszyklus generierten Arbeitsprodukte zusammenführen [ISO26262, 2-6.4.6.2]. Diese müssen vollständig erfasst sein und der Nachweis muss erbracht werden, dass durch die Vereinigung sämtlicher Arbeitsprodukte alle Sicherheitsanforderungen erfüllt sind. Hierbei kann der Sicherheitsnachweis um weitere Nachweise im Kontext von „Safety“, welche sich außerhalb des Bereichs der ISO 26262 befinden, erweitert werden.

Aktuelle Ansätze für eine grafische Unterstützung des Sicherheitsnachweises verfolgen die Goal Structuring Notation (GSN)[36] oder „Claims-Argument-Evidence“ [ISO26262, 10-5.3.1]. Hinsichtlich der GSN kann insbesondere auf Arbeiten von Kelly[37] [148], [65], [66], [67] verwiesen werden, welche auch in der ISO 26262 referenziert werden.

[36] http://www.goalstructuringnotation.info/ (Zugriff am 15.10.2014)
[37] http://www-users.cs.york.ac.uk/tpk/ (Zugriff am 15.10.2014)

2.7.3 Sicherheitsanalysen

Als Methoden zur Systemanalyse zählen nach [132, S.131] die FMEA, FTA, Zuverlässigkeitsblockdiagramme, im Englischen reliability block diagrams, Ereignisbaumanalyse, im Englischen Event Tree Analysis (ETA), Markov-Analyse oder HAZard and OPerability analysis (HAZOP). Im Kontext der ISO 26262 [ISO26262, 9-8.2] werden diese aufgegriffen und wie in Tab. 2.1 dargestellt eingeordnet. Die Sicherheitsanalysen werden auf geeigneter Abstraktionsebene während der Konzept- und Entwicklungsphase angewendet.

Qualitative Analysis	Quantitative Analysis
Qualitative FMEA	Quantitative FMEA
Qualitative FTA	Quantitative FTA
HAZOP	Quantitative ETA
Qualitative ETA	Markov Models
	Reliability Block Diagrams

Inductive analysis method
Deductive analysis method

Tab. 2.1: Erwähnte qualitative und quantitative Analysemethoden nach ISO 26262

Qualitative Analysen identifizieren Fehlerursachen bzw. Auswirkungen während quantitative Analysen die Häufigkeit dieser untersuchen. Weiterhin kann eine Unterscheidung in deduktive und induktive Analysemethoden vorgenommen werden. Deduktiv beschreibt das Vorgehen nach einem Top-Down Prinzip, beginnend von bekannten Auswirkungen nach möglichen Fehlerursachen zu suchen. Im Gegensatz dazu ist im Rahmen einer induktiven Vorgehensweise nach dem Bottom-Up Prinzip, ausgehend von bekannten Fehlerursachen auf mögliche Auswirkungen zu schließen. Im Folgenden werden die innerhalb der ISO 26262 aufgeführten und im Kontext dieser Arbeit relevanten Sicherheitsanalysen FTA und FMEA vorgestellt.

2.7.3.1 Fehlerbaumanalyse (FTA)

Die Fehlerbaumanalyse ist eine bewährte Methode sowohl zur qualitativen als auch quantitativen Analyse und wird in Standards von unterschiedlichen Domänen und Anwendungsgebieten aufgegriffen. Bereits 1981 wurde von der *„United States Nuclear Regulatory Commission“* das *„NUREG-0492: Fault Tree Handbook“* [NUREG0492] herausgegeben. Ein allgemeiner und branchenübergreifender Standard für die FTA wurde publiziert durch die IEC in IEC 61025. Als deutsche Norm sind die korrespondierende *„DIN 61025 Fehlzustandsbaumanalyse“* [DIN61025] sowie die in zwei Teilen beschriebene *„DIN 25424-1“* [DIN25424-1] für „Fehlerbaumanalyse - Methode und Bildzeichen“ und *„DIN 25424-2“* [DIN25424-2] für „Fehlerbaumanalyse - Handrechenverfahren zur Auswertung eines Fehlerbaumes“ zu nennen.

FTA ist eine deduktive Systemanalysetechnik, um potentielle Fehlerursachen in einem System zu identifizieren. Basierend auf einer qualitativen und quantitativen Analyse [68] können Maßnahmen getroffen werden, um potentielle Abweichungen zu korrigieren oder zu verhindern. Ein Auszug der Hauptziele einer FTA gemäß [DIN61025, 5.2] ist nachfolgend aufgeführt:

- Identifizierung von Ursachen und Kombinationen von Ursachen, die zum Eintritt des Hauptereignisses führen
- Beurteilung, ob eine bestimmte Zuverlässigkeitsmaßgröße eine festgelegte Forderung erfüllt
- Identifizierung von potentiellen Ausfallarten oder Einflussgrößen, welche am meisten zur Ausfallwahrscheinlichkeit oder Nichtverfügbarkeit eines instandsetzbaren Systems beitragen
- Analyse und Vergleich von verschiedenen Entwurfs-/Konstruktionsalternativen für eine bessere Systemzuverlässigkeit

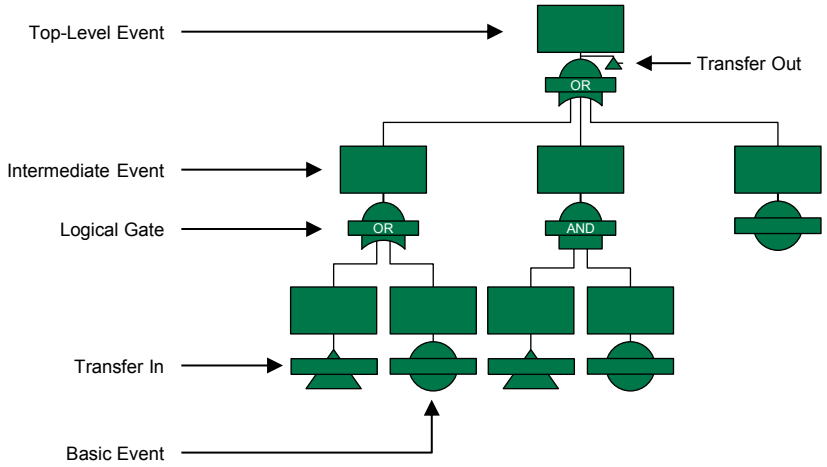


Abb. 2.22: Vereinfachte Darstellung eines Fehlerbaums

Ein exemplarischer Fehlerbaum ist in Abb. 2.22 dargestellt. Ein Fehlerbaum ist aufgebaut aus einem Hauptereignis, im Englischen Top-Level Event, als Wurzel des Fehlerbaumes und seiner Verbindung zu einem oder mehreren Grundereignissen, im Englischen Basic Events, als Blätter des Baumes. Zwischen dem Hauptereignis und den Grundereignissen können Zwischenereignisse, im Englischen Intermediate Events, vorhanden sein. Die Verbindung der Ereignisse erfolgt über logische Gatter, wie beispielsweise AND sowie OR, siehe [DIN61025, Tab.A.3] für statische Gatter. Zusätzlich können Transfer-Gatter genutzt werden, um für komplexe Systeme Teil-Fehlerbäume miteinander zu verknüpfen.

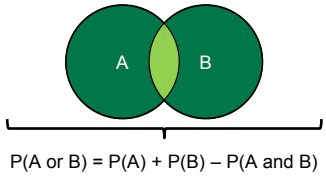
Die Symbole und Beschreibungen der verschiedenen Ereignistypen sind in [DIN61025, Tab.A.2] aufgeführt.

Im Rahmen der qualitativen Analyse des Fehlerbaums können weiterhin Minimalschnitte, im Englischen minimal cut sets, abgeleitet werden, welche in [NUREG0492, S.VII-15] beschrieben sind. Ein Minimalschnitt ist definiert als kleinste Kombination von Grundereignissen, welche zum Eintritt des Hauptereignisses führt. Hierbei kann dem Minimalschnitt eine entsprechende Ordnung zugewiesen werden, je nach Anzahl der enthaltenen Grundereignisse.

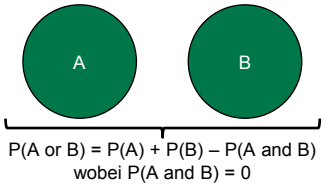
Operation	Wahrscheinlichkeit	Mathematik	Logik	Ingenieurwissenschaft
Vereinigung von A und B	A oder B	$A \cup B$	$A \vee B$	$A + B$
Durchschnitt von A und B	A und B	$A \cap B$	$A \wedge B$	$A \cdot B$ oder AB
Komplement von A	nicht A	A' oder $\overline{A}$	$\neg A$	A' oder $\overline{A}$

Tab. 2.2: Boolesche Algebra in [NUREG0492, S.VI-16]

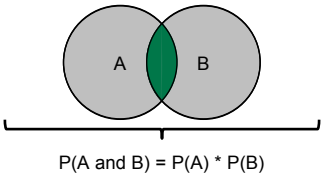
In [NUREG0492, S.VI-16] wird auf die Symbolik für die Notation der logischen Verknüpfungen eingegangen, siehe Tab. 2.2. Im Folgenden wird die Symbolik der Ingenieur-Notation, welche sich nach [NUREG0492, S.VI-16] weit verbreitet hat, verwendet. Somit werden im Rahmen der Fehlerbaumanalyse die Symbole „$+$“ und „$\cdot$“ überschrieben.



(a) Berechnungsvorschrift OR



(b) Berechnungsvorschrift OR (mutually exclusive events)



(c) Berechnungsvorschrift AND

Abb. 2.23: Grafische Veranschaulichung für die Berechnungsvorschriften der logischen Gatter AND sowie OR

Im Rahmen einer quantitativen Analyse des Fehlerbaums können für die einzelnen Ereignisse Eintrittswahrscheinlichkeiten angegeben bzw. bestimmt werden. Für die Grund-

ereignisse sind die Wahrscheinlichkeiten zuzuweisen, anschließend können für die Zwischenereignisse und das Hauptereignis mit Hilfe der logischen Gatter sowie der Wahrscheinlichkeitsrechnung die Eintrittswahrscheinlichkeiten bestimmt werden. Die Berechnungsvorschriften hierfür sind in Abb. 2.23 für die logischen Gatter AND sowie OR abgebildet. In [NUREG0492, VI-3] werden die „Algebraischen Operationen mit Wahrscheinlichkeiten“ beschrieben.

Für eine logische OR-Verknüpfung gilt allgemein die Berechnungsvorschrift 2.1 nach [NUREG0492, VI-4], siehe Abb. 2.23(a). In Abb. 2.23(b) sind die beiden Ereignisse A und B sich gegenseitig ausschließend, wie beispielsweise bei einem Münzwurf. Hierdurch ergibt sich ein Sonderfall der allgemeinen Gleichung, bei dem die Schnittmenge der beiden Ereignisse leer ist, siehe Gleichung 2.2 nach [NUREG0492, VI-2]. Angemerkt sei, dass der Sonderfall immer eine obere Schranke für die Wahrscheinlichkeit darstellt.

$$P(A\ oder\ B) = P(A) + P(B) - P(A\ und\ B) \tag{2.1}$$

$$P(A\ oder\ B) = P(A) + P(B), falls\ P(A\ und\ B) = 0 \tag{2.2}$$

Für die logische UND-Verknüpfung zweier Ereignisse ist die Berechnungsvorschrift für die Wahrscheinlichkeit nach [NUREG0492, VI-8] in Gleichung 2.3 dargestellt, siehe Abb. 2.23(c). Hierbei ist die Unabhängigkeit der Ereignisse vorausgesetzt, siehe auch [ISO26262, 5-AnnexC.1].

$$P(A\ und\ B) = P(A) \cdot P(B) \tag{2.3}$$

2.7.3.2 Failure Mode and Effects Analysis (FMEA)

Bei der FMEA handelt es sich um eine induktive Methode, welche sich insbesondere in der Automobilindustrie als Analysemethode etabliert hat, vgl. [132, S.153]. ISO 26262 greift die drei unterschiedlichen Ausprägungen der System-, Design- und Prozess-FMEAs nach [ISO26262, 9-8.2] auf. Produkt- und Prozess-FMEA werden im VDA Band 4 „Sicherung der Qualität in der Prozesslandschaft“ [69] unter den genannten Risikoanalysen dargestellt.

Die FMEA fokussiert sich auf eine strukturierte qualitative Analyse von Systemelementen, um die Effekte von Fehlerursachen auf das Systemverhalten zu identifizieren. Im Laufe der Zeit haben sich verschiedene erweiterte Methoden basierend auf der FMEA ausgeprägt, welche im Folgenden erläutert werden.

Die Failure Mode, Effects and Criticality Analysis (FMECA) adressiert als Erweiterung der klassischen FMEA zusätzlich quantitative Aspekte, indem Informationen über Schwere und Wahrscheinlichkeit von Ausfällen mittels einer Kritikalitätsmetrik eingebracht werden [70]. Initial formalisiert wurde die Prozedur zur Durchführung einer FMECA in MIL-STD-1629, vgl. überarbeitete Revision [MIL1629A, 4.2].

Die Failure Mode, Effects and Diagnostic Analysis (FMEDA) betrachtet zusätzlich quantitative Fehlerinformationen und den Diagnoseabdeckungsgrad, beispielsweise eines Sicherheitsmechanismus. FMEDA wurde basierend auf dem Beitrag [71] eingeführt und wurde namentlich erstmals in [72] verwendet. Die FMEDA-Methodik wurde insbesondere in IEC 61508 verfeinert, um zusätzliche quantitative Ergebnisse für die Safe Failure Fraction (SFF) bereitzustellen. FMEDA ist eine allgemeine Praxis für Sicherheitsanalysen von elektrischen/elektronischen Systemen [70].

ISO 26262 spricht im Allgemeinen nur von FMEA und erwähnt die FMEDA nicht direkt, allerdings sind die Tabellen innerhalb [ISO26262, 5-AnnexE] an eine solche angelehnt. Nach [149, S.168] erweitert ISO 26262 die klassische FMEDA um die Betrachtung der Multiple-Point-Faults (MPFs).

2.7.4 Fehlerinformationen und Zuverlässigkeit von Hardware

Ausfälle können im Rahmen der ISO 26262 in systematische Ausfälle, im Englischen systematic failure, [ISO26262, 1-1.130] sowie zufällige Ausfälle der Hardware, im Englischen random hardware failure, [ISO26262, 1-1.92] eingeordnet werden. Systematische Ausfälle basieren auf einer bestimmten Ursache und können nur durch Änderungen im Design- oder Herstellungsprozess, der Betriebsart oder weiteren Prozeduren vermieden werden, siehe auch [147, S.419]. Fehler in der Software beschreiben systematische Fehler, Ausfälle der Hardware können zum Teil ebenfalls systematisch sein [ISO26262, 10-4.3]. Zufällige Ausfälle der Hardware können durch physikalische Prozesse wie Alterung unvorhersagbar für ein *Hardware-Element* auftreten [ISO26262, 10-4.3]. Im Kontext der Zuverlässigkeit von *Hardwaredesigns* sind insbesondere diese mit ihren zugehörigen Charakteristika relevant.

2.7.4.1 Fehlerursache (Fault) - Fehlerzustand (Error) - Fehlerauswirkung (Failure)

Die deutsche Terminologie für die Beschreibung von Fehlern ist nicht so umfangreich wie die Englische. Zudem werden englische Begrifflichkeiten minimal unterschiedlich übersetzt. Da diese Arbeit im Fokus der ISO 26262 steht, werden die Begrifflichkeiten Fault, Error und Failure und deren Definitionen als Grundlage verwendet.

Der „Fault“ kann nach [132, S.5] mit Abweichung oder mit Fehlerursache [150, S.48] übersetzt werden. Nach [ISO26262, 1-1.42] handelt es sich um „einen abweichenden Zustand, der dazu führen kann, dass ein Element [ISO26262, 1-1.32] oder ein Item [ISO26262, 1-1.69] ausfällt“.

Der „Error“ kann nach [132, S.5] mit Fehler oder mit Fehlerzustand [150, S.48] übersetzt werden. Nach [ISO26262, 1-1.36] handelt es sich um eine „Diskrepanz zwischen einem berechneten, beobachteten oder gemessenen Wert oder Zustand und dem richtigen, spezifizierten oder theoretisch korrekten Wert oder Zustand“.

Der „Failure“ kann nach [132, S.5] mit Ausfall oder mit Fehlerauswirkung [150, S.48] übersetzt werden. Nach [ISO26262, 1-1.39] handelt es sich um die „Beendigung der Fähigkeit eines Elements [ISO26262, 1-1.32] eine Funktion wie erforderlich durchzuführen“.

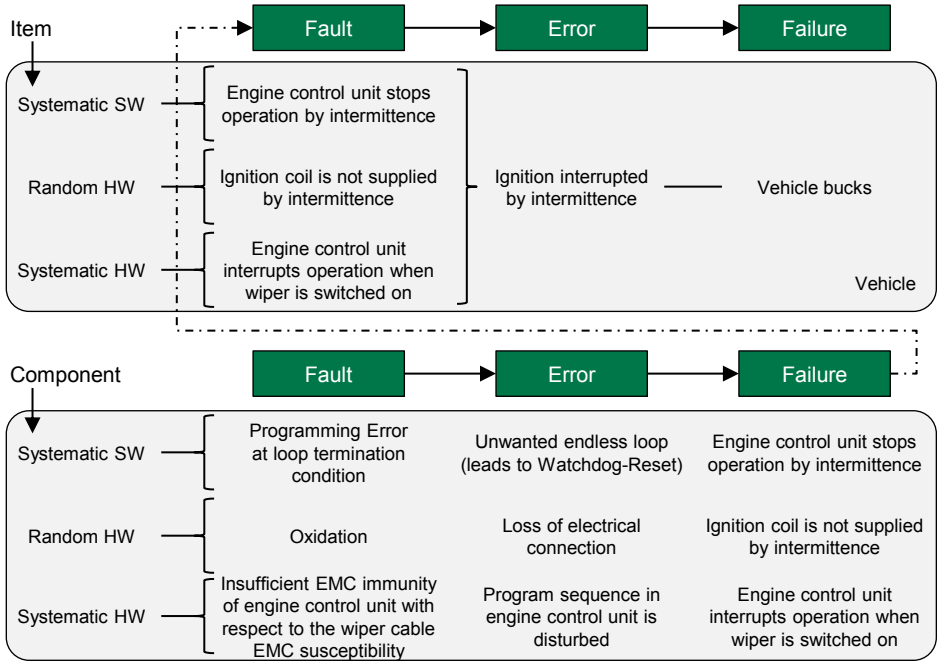


Abb. 2.24: Fault - Error - Failure auf unterschiedlichen Abstraktionsebenen nach [ISO26262, 10-Fig.5]

Die Zusammenhänge zwischen Fault, Error und Failure auch im Kontext der vorgestellten Einstufungen von Fehlern in systematische bzw. zufällige sind in Abb. 2.24 dargestellt. Weitere Definitionen können aus den Dokumenten [73, S.3] oder [74] entnommen werden.

2.7.4.2 Ausfallrate (Failure Rate)

Zur Beschreibung der Zuverlässigkeit von *Hardware-Elementen* im Kontext von zufälligen Ausfällen der Hardware werden unterschiedliche Kenngrößen genutzt. Einer der wichtigsten Begriffe ist die Ausfallrate, im Englischen failure rate. ISO 26262 beschreibt in [ISO26262, 1-1.41] die Ausfallrate als „Wahrscheinlichkeitsdichte eines Ausfalls geteilt durch die Wahrscheinlichkeit des Überlebens eines *Hardware-Elements* “. Für gewöhnlich wird sie durch den griechischen Buchstaben λ angegeben. Die Ausfallrate beschreibt den Ausfall von *Hardware-Elementen* in einer bestimmten Zeit. Die Ausfallrate kann im Allgemeinen nur statistisch erhoben und damit experimentell ermittelt werden [147, S.77].

Diese wird üblicherweise in der Einheit Failures In Time (FIT) angegeben, wobei ein FIT einem Ausfall in 10^9 h entspricht, siehe Gleichung 2.4.

$$1\ FIT = 1 \cdot 10^{-9} \frac{1}{h} \tag{2.4}$$

Die Ausfallrate λ eines *Hardware-Elements* ist veränderlich über die Zeit, $\lambda(t)$. Dieser Zusammenhang wird meist über eine Weibull-Verteilung mit einem Formparameter b, häufig auch unter dem Namen „Badewannenkurve“ bekannt, abgebildet [147, S.76, S.334, S.497]. Diese ist exemplarisch in Abb. 2.25 dargestellt, wobei hier drei verschiedene Bereiche auf der Zeitachse unterschieden werden.

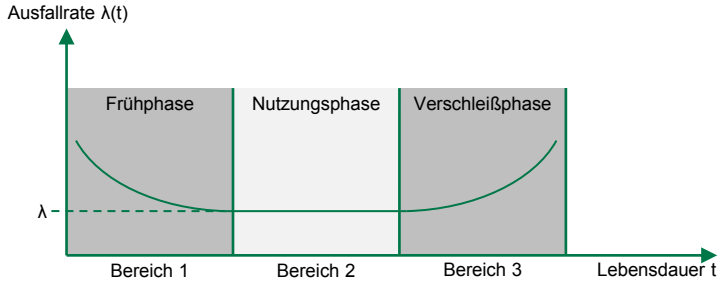


Abb. 2.25: „Weibull-Verteilung der Hardware-Zuverlässigkeit“, angelehnt an [147, S.76, S.334, S.497]

Bereich 1 „Frühphase“: Die Ausfallrate ist bedingt durch Frühausfälle von Bauelementen zu Beginn sehr hoch. Eine stetige Abnahme durch Überarbeitung und Verbesserung des Systems kennzeichnet anschließend die Ausfallrate.

Bereich 2 „Nutzungsphase“: In der Phase der Zufallsausfälle „erreicht die Ausfallrate $\lambda(t)$ ihr Minimum“ und „kann als konstant angenommen werden“.

Bereich 3 „Verschleißphase“: Mit „der Lebensdauer der Bauelemente steigt“ die Ausfallrate bedingt durch beispielsweise Alterungseffekte bis zur Unbrauchbarkeit des Elementes an.

Bezogen auf den typischen Verlauf der Badewannenkurve ist im mittleren der drei Bereiche über der Zeit t die Ausfallrate λ als konstant angenommen. Im Rahmen der ISO 26262 wird nach Gleichung 2.5 immer eine näherungsweise konstante Ausfallrate angenommen [ISO26262, 1-1.41.note] und somit lediglich die Nutzungsphase des Elementes betrachtet, siehe hierzu auch [147, S.75].

$$\lambda(t) \approx konst. \tag{2.5}$$

Statt der Ausfallrate als Kenngröße kann eine mittlere ausfallfreie Arbeitszeit, im Englischen Mean Time To Failure (MTTF), angegeben werden. Im Falle einer konstanten Ausfallrate stellt die MTTF den reziproken Wert der Ausfallrate dar, siehe Gleichung 2.6 [147, S.73].

$$MTTF = \frac{1}{\lambda} \tag{2.6}$$

Bei reparierbaren Systemen kann nun eine mittlere Instandsetzungszeit, im Englischen Mean Time To Repair (MTTR), angegeben werden. Mit dieser MTTR kann die mittlere Brauchbarkeitsdauer, im Englischen Mean Time Between Failures (MTBF), als Zeit zwischen zwei Ausfällen nach Gleichung 2.7 angegeben werden [147, S.74].

$$MTBF = MTTF + MTTR \tag{2.7}$$

Falls kein reparierbares Element vorliegt oder $MTTF >> MTTR$ gilt, kann bei konstanten Ausfallraten Gleichung 2.7 zu Gleichung 2.8 vereinfacht werden.

$$MTBF = MTTF = \frac{1}{\lambda} \tag{2.8}$$

Die Zusammenhänge von MTTF, MTTR und MTBF sind zusammenfassend in Abb. 2.26 dargestellt. Bei der Mean Time To First Failure (MTTFF) handelt es sich um die mittlere Lebensdauer bis zum ersten Ausfall.

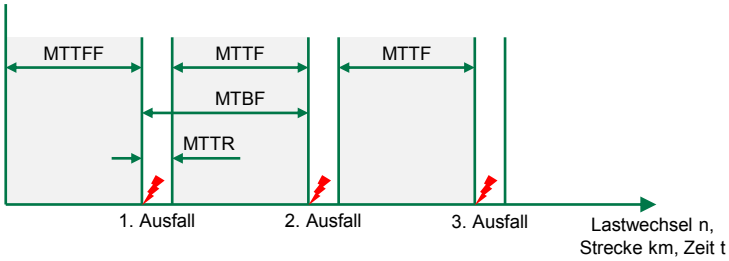


Abb. 2.26: Zusammenhang zwischen MTTFF, MTTF, MTTR und MTBF nach [147, S.503]

Im Kontext der Annahme von konstanten Ausfallraten im Bereich der Nutzungsphase lässt sich eine Exponentialverteilung als Sonderfall der Weibullverteilung mit Formparameter b=1 ableiten [147, S.499]. Die Exponentialverteilung wird in der Elektrotechnik häufig genutzt, da diese „charakteristisch für elektronische Bauteile“ ist [147, S.498]. Eine monoton fallende Dichtefunktion sowie eine konstante Ausfallrate sind die wesentlichen Eigenschaften. Die Ausfalldichte einer Exponentialverteilung lässt sich nach Gleichung 2.9 darstellen. Im Rahmen der ISO 26262 wird eine Exponentialverteilung auch

im Kontext der Annahme einer konstanten Ausfallrate vorgeschlagen, siehe [ISO26262, 5-AnnexC.1.2], [ISO26262, 10-AnnexB.4].

$$f(t) = \frac{dF(t)}{dt} = \lambda \cdot e^{-\lambda \cdot t} \tag{2.9}$$

Die Ausfallwahrscheinlichkeit $F(t)$ einer Exponentialverteilung mit konstanter Ausfallrate λ ergibt sich nach Gleichung 2.10.

$$F(t) = 1 - e^{-\lambda \cdot t} \tag{2.10}$$

Die Zuverlässigkeit $R(t)$ im Umkehrschluss zur Ausfallwahrscheinlichkeit lässt sich nach Gleichung 2.11 beschreiben.

$$R(t) = 1 - F(t) = e^{-\lambda \cdot t} \tag{2.11}$$

2.7.4.3 Ausfallart (Failure Mode)

Ausfallarten, im Englischen Failure Modes, beschreiben nach [ISO26262, 1-1.40] die Art, in welcher ein Element oder Fahrzeugsystem (Item) ausfallen kann. Diese sind im Kontext zufälliger Ausfälle der Hardware ergänzend zur Angabe einer Ausfallrate. Bezogen auf *Hardware-Bauteile* sind gebräuchliche Ausfallarten beispielsweise ein Kurzschluss, im Englischen short circuit, oder ein offener Schaltkreis, im Englischen open circuit.

Für jede Ausfallart wird weiterhin eine Ausfallratenverteilung spezifiziert. Die Ausfallratenverteilung gibt die prozentuale Verteilung der Ausfallrate des *Hardware-Elements* bezogen auf die einzelnen Ausfallarten an. Der Wertebereich für die Ausfallratenverteilung liegt zwischen 0 % und 100 %. Die Summe über die Ausfallratenverteilungen aller Ausfallarten eines *Hardware-Elements* muss dabei immer 100 % ergeben. Die Ausfallarten mit ihrer Ausfallratenverteilung sind ebenfalls statistisch erhobene bzw. experimentell ermittelte Werte.

Der Standard MIL-HDBK-338B gibt beispielsweise für einen drahtgebundenen Widerstand die Ausfallratenverteilung 9 % für die Ausfallart „short circuit“, 65 % für die Ausfallart „open circuit“ sowie 26 % für die Ausfallart „parameter change“ an [MIL338B, 7-198].

2.7.4.4 Klassifikation der Ausfallarten

Der Ausfall eines *Hardware-Elements* im Kontext einer spezifischen Ausfallart kann verschiedene Auswirkungen auf das Verhalten des Fahrzeugsystems (Item) haben, unter Berücksichtigung eines spezifischen Sicherheitsziels. Daher können diese Ausfallarten

bzw. ihre verursachten Abweichungen durch die potentielle Verletzung im Kontext des betrachteten Sicherheitsziels klassifiziert werden. Hierbei sind die Klassifikationen bezogen auf verschiedene Standards unterschiedlich vorgeschlagen. Im Rahmen der IEC 61508 ist eine Einstufung in sichere bzw. gefährliche Fehler sowie für gefährliche Fehler tiefergehend in entdeckbare bzw. nicht entdeckbare Fehler vorgeschlagen.

Im Kontext der ISO 26262 wird nach [ISO26262, 5-7.4.3.2] eine Klassifikation wie folgt vorgenommen: Wenn die Ausfallart keine Auswirkung auf die Verletzung des Sicherheitsziels hat, wird dies als ein sicherer Fehler, im Englischen Safe-Fault (SF), spezifiziert. Ein Einzelfehler, im Englischen Single-Point-Fault (SPF), oder Restfehler, im Englischen Residual-Fault (RF), führt direkt zur Verletzung des Sicherheitsziels. Ausfallarten, welche zur Verletzung des Sicherheitsziels in Kombination mit anderen unabhängigen Ausfallarten führen, werden als Mehrfachfehler, im Englischen Multiple-Point-Fault (MPF), eingestuft. Die Zusammenhänge der Klassifikationen sind übergreifend in Abb. 2.27 dargestellt.

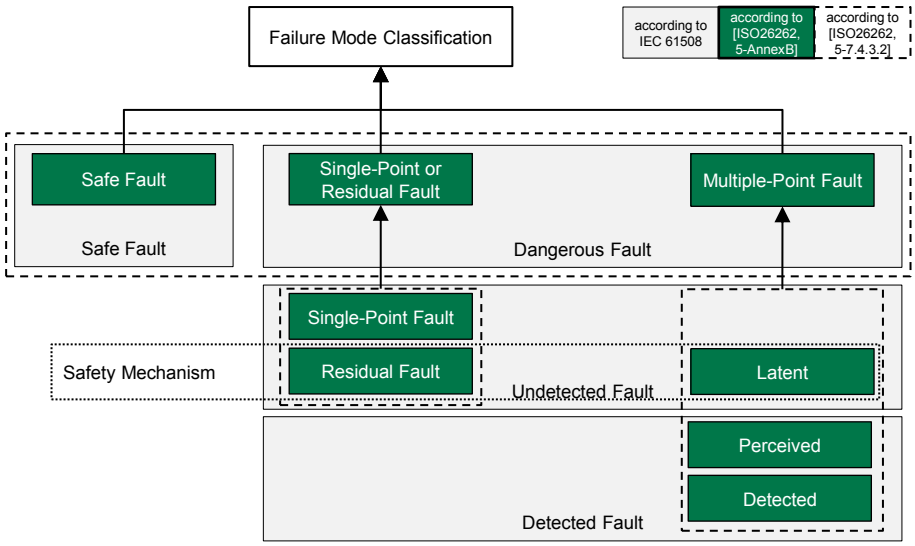


Abb. 2.27: Klassifikation der Ausfallart in Bezug auf ISO 26262 und IEC 61508

Ein SPF führt ohne Abdeckung durch einen Sicherheitsmechanismus zur direkten Verletzung des Sicherheitsziels [ISO26262, 1-1.121], [ISO26262, 1-1.122]. Wenn ein Sicherheitsmechanismus die direkte Verletzung abdeckt, entspricht der Anteil der Abweichung, welcher nicht abgedeckt wird, dem RF [ISO26262, 1-1.96]. Somit ist ein SPF mit einem RF, dessen Abdeckung gleich 0 % ist, gleichzusetzen [ISO26262, 1-1.121.note1]. MPFs können entweder durch einen Sicherheitsmechanismus erkannt (detected), durch den Fahrer aufgrund unterschiedlichen Systemverhaltens wahrgenommen (perceived) oder

nicht erkannt bzw. wahrgenommen werden (latent). MPFs mit einer Distanz von 2 werden im Rahmen der ISO 26262 auch als Zweifachfehler, im Englischen Dual-Point-Fault (DPF), bezeichnet [ISO26262, 1-1.29], [ISO26262, 1-1.30]. SF erhöhen die Wahrscheinlichkeit einer Verletzung des Sicherheitsziels nicht maßgeblich [ISO26262, 1-1.101]. Hierbei können Fehler sowohl von sicherheitsbezogenen als auch nicht-sicherheitsbezogenen *Hardware-Elementen* einen SF darstellen [ISO26262, 5-Fig.B.1].

2.7.4.5 Sicherheitsmaßnahmen und Sicherheitsmechanismen

Sicherheitsmaßnahmen, im Englischen Safety Measures [ISO26262, 1-1.110], beinhalten als Oberbegriff die Sicherheitsmechanismen, im Englischen Safety Mechanisms (SMs), [ISO26262, 1-1.111]. Bei Sicherheitsmechanismen handelt es sich um technische Lösungen, um einen sicheren Zustand, im Englischen Safe State, zu erreichen oder beizubehalten. Ein implementierter Sicherheitsmechanismus soll SPFs verhindern, RFs reduzieren oder verhindern, dass MPFs latent werden.

Die Effektivität eines Sicherheitsmechanismus ist gekennzeichnet durch seinen Diagnoseabdeckungsgrad K_{DC}, im Englischen Diagnostic Coverage (DC), [ISO26262, 1-1.25][38]. In Bezug auf einen SM wird hiermit der prozentuale Wert angegeben, um den eine Ausfallart abgedeckt wird. Die zum SM zugehörigen Werte für die Diagnoseabdeckungsgrade beziehen sich entweder auf die Abdeckung in Bezug auf Restfehler oder in Bezug auf latente Mehrfachfehler, womit zwei unterschiedliche Diagnoseabdeckungsgrade angegeben werden [ISO26262, 1-1.25.note1]. Sicherheitsmechanismen werden im Rahmen von Part 5 der ISO 26262 auf Ebene der *Hardware-Bauteile* abgebildet, in Part 10 für Mikrocontroller auf Ebene der Ausfallart.

2.7.4.6 Hardware-Sicherheitsevaluationen

Im Rahmen der Produktentwicklung auf Hardwareebene nach Part 5 werden explizit zwei sich ergänzende quantitative Evaluationen gefordert, um das *Hardwaredesign* gegenüber zufälligen Ausfällen abzusichern. Dies betrifft die „*Evaluation of the hardware architectural metrics*“ [ISO26262, 5-8] und die „*Evaluation of safety goal violations due to random hardware failures*“ [ISO26262, 5-9]. Diese Hardware-Sicherheitsevaluationen basieren auf den vorgestellten Ausfallraten, Ausfallarten, der Absicherung durch Sicherheitsmechanismen sowie der ASIL-Klassifizierung des zugehörigen Sicherheitsziels. Durchgeführte Sicherheitsanalysen können zur Unterstützung der Evaluationen dienen [ISO26262, 9-8.2].

[38] Der Begriff der Diagnostic Coverage (DC) wird im Kontext von Sicherheitsmechanismen sowie der Ausfallratenklassen-Methode mit einer unterschiedlichen Bedeutung verwendet.

2.7.4.7 Anerkannte Industriesammlungen

Die vorgestellten statistischen Fehlerinformationen für die Beschreibung von zufälligen Hardware-Fehlern sind in unterschiedlichen anerkannten Industriesammlungen festgehalten. Die ISO 26262 listet in [ISO26262, 5-8.4.3] beispielhaft zehn „allgemein anerkannte Industriequellen zur Bestimmung von *Hardware-Bauteil* Ausfallraten und Ausfallratenverteilungen " auf:

IEC/TR 62380: Dieser TR [IECTR62380], ursprünglich „RDF 2000 (UTE C 80-810)", wurde von der Union Technique de l'Electricite (UTE)[39] als französischer Telekommunikationsstandard herausgegeben. Der Titel lautet „Reliability data handbook - Universal model for reliability prediction of electronics components, PCBs and equipment". Der Standard beschreibt verschiedene mathematische Modelle zur Berechnung von Ausfallraten und listet Ausfallarten für unterschiedliche *Hardware-Bauteile* auf. Interessant sind insbesondere auch die mathematischen Modelle für bestückte printed circuit boards (PCBs) und Hybrid-Schaltungen.

IEC 61709: Dieser Standard trägt den Titel „Electric components - Reliability - Reference conditions for failure rates and stress models for conversion" und in der deutschen Norm DIN EN 61709:2012-01 „Elektrische Bauelemente - Zuverlässigkeit - Referenzbedingungen für Ausfallraten und Beanspruchungsmodelle zur Umrechnung". Hierbei ist ein generischer Ansatz für Ausfallraten unter Referenzbedingungen gewählt. Die tatsächliche Beanspruchung kann mit einem Beanspruchungsmodell dargestellt werden.

MIL HDBK 217 F notice 2: Das „Military Handbook" für „Reliability Prediction of Electronic Equipment" [MIL217F], veröffentlicht vom Department of Defense, stellt eines der ältesten Sammlungen für statistischen Fehlerinformationen dar. Die aktuelle Version notice 2 stammt aus dem Jahr 1995. Sie beschreibt Ausfallratenmodelle für unterschiedliche elektronische *Hardware-Bauteile* insbesondere zur Nutzung in frühen Designphasen.

UTE C80-811: Diese Norm entspricht dem FIDES[40] guide 2004 issue A, aktuelle Version ist der FIDES guide 2009 [FIDES]. Der FIDES guide ist aus einem europäischen Konsortium von acht Industriepartnern hervorgegangen und wurde anschließend von der französischen Standardisierungsorganisation UTE mit der Referenz UTE C80-811 akzeptiert. Der Standard ist frei verfügbar. In diesem werden neben den Zuverlässigkeitsdaten auch eine Anleitung für Prozesskontrolle und Audits präsentiert.

EN 50129:2003, Annex C: Die DIN EN 50129 mit dem Titel „Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Sicherheitsrelevante elektronische Systeme für Signaltechnik" stellt den Standard bzgl. funktionaler Sicherheit für die Domäne der Bahnanwendungen dar. Annex C beschreibt die Identifikation von Ausfallarten im Kontext von Hardware.

[39] http://www.ute-fr.com (Zugriff am 15.10.2014)
[40] http://fides-reliability.org (Zugriff am 15.10.2014)

IEC 62061:2005, Annex D: IEC 62061:2005 trägt den Titel „Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems“. Dieser beschreibt Anforderungen bzgl. funktionaler Sicherheit für Maschinen-Steuerungen und enthält Fehlerinformationen für die Zuverlässigkeit von Hardware.

RIAC NPRD-95: Die „Nonelectronic Parts Reliability Data (NPRD)“ stammt von der Reliability Information Analysis Center (RIAC)[41]. In der NPRD-2011 werden Ausfallraten für mechanische, elektro-mechanische und elektronische Baugruppen betrachtet. Ergänzend hierzu wird die Electronic Parts Reliability Data (EPRD)-97 zur Verfügung gestellt.

RIAC FMD-97: Die Failure Mode / Mechanism Distribution (FMD) fokussiert sich auf Ausfallarten sowie deren Ausfallratenverteilungen von Hardware. Sie ist somit als Ergänzung zu den NPRD bzw. EPRD zu sehen, welche Ausfallraten betrachten. Die aktuelle Version ist bezeichnet mit FMD-2013.

RIAC HDBK-217Plus: Das „Handbook of 217Plus^{TM} Reliability Prediction Models“, veröffentlicht von RIAC, ist ein Handbuch für Zuverlässigkeits-Vorhersagemodelle. Es beschreibt die Verwendung der Modellierungsparameter und -gleichungen der angelehnten Methodik 217Plus^{TM}.

MIL HDBK 338: Das „Electronic Reliability Design Handbook“, veröffentlicht vom Department of Defense, beschreibt grundsätzliche Ansätze, Methodiken und Aktivitäten in Bezug auf die Entwicklung im Kontext von funktionaler Sicherheit. Hierbei werden neben den theoretischen Modellen auch Richtlinien und Analysemethoden aufgegriffen.

Eine weitere interessante sowie die in der Industrie häufig verwendete Sammlung ist die SN 29500 von Siemens.

[41] http://www.theriac.org/productsandservices/products/ReliabilityData.php (Zugriff am 15.10.2014)

3 Stand der Wissenschaft und Technik

Innerhalb dieses Kapitels wird der Stand der Wissenschaft und Technik für Prozessmodellierung und -management sowie für Hardware-Sicherheitsevaluationen im Kontext von ISO 26262 gesondert aufgeführt.

3.1 Prozessmodellierung und -management

Die Beschreibung von Prozessen und deren Kommunikation ist in allen Wirtschaftszweigen entscheidend, um eine produktive und effiziente Arbeit zu gewährleisten sowie eine strukturierte und festgelegte Vorgehensweise einzuhalten. In [56, S.142] werden Geschäftsprozessmanagement-Aktivitäten betrachtet und eine Evaluation von Geschäftsprozess-Modellierungswerkzeugen aus dem Bereich der wissenschaftlichen und kommerziellen Tools vorgestellt. Hierzu werden die Aspekte der Geschäftsprozess-Analyse, Geschäftsprozess-Modellierung, Geschäftsprozess-Optimierung, Simulation von Geschäftsprozessen, Automation und Integration sowie des Geschäftsprozess-Monitorings für BPM-Ansätze gelistet, vgl. hierzu auch [151]. Diese werden insbesondere bei der Bereitstellung von Notationen berücksichtigt, jedoch werden die vielzähligen geforderten Aspekte teilweise unterschiedlich ausgeprägt adressiert, was sich wiederum in der Popularität und Nutzung der darauf aufbauenden Werkzeuge niederschlägt.

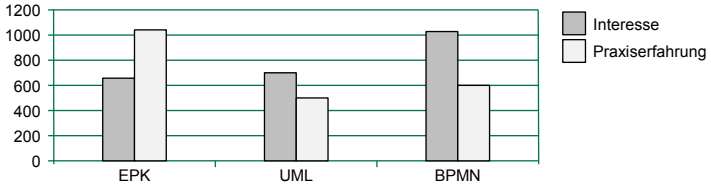


Abb. 3.1: „Popularität von Prozessnotationen“ nach [152, S.XIII] (Stand 2010)

In [152, S.XIII] wird das Interesse und die Praxiserfahrung von den Prozessmodellierungs-Notationen EPK, UML und BPMN gegenübergestellt. Im Vorfeld wurden im Jahr 2009 die Daten aus einem deutschsprachigen BPM-Netzwerk mit über 6.000 BPM-Professionals erhoben, von denen 2.400 hinterlegte Detailprofile ausgewertet werden konnten. Die Anzahl der Mitglieder des Netzwerks stieg mittlerweile auf über 7.000 an und die Datensätze konnten erneut für eine Auswertung nach Abb.3.1 verwendet werden. Nach [152,

S.XIII] hat die Praxiserfahrung von BPMN im Vergleich zu der vorherigen Auswertung um ca. 45 % zugenommen. Zudem lässt sich festhalten, dass das Interesse an BPMN am größten ist, jedoch befinden sich alle drei Notationen aktuell im Einsatz, was sich aus der zunehmenden Praxiserfahrung über alle Notationen hinweg folgern lässt.

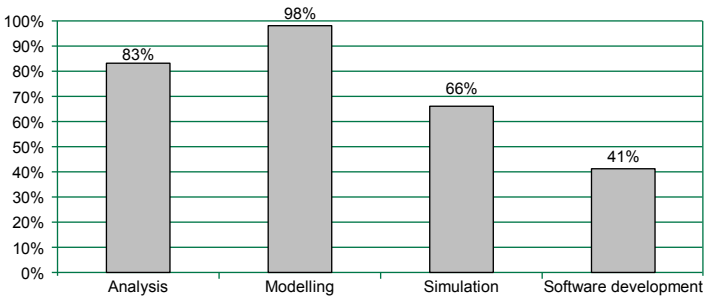


Abb. 3.2: Einsatzgebiete der BPM-Werkzeuge nach [56, Abb.6]

Innerhalb von [56] werden die Ergebnisse einer empirischen Analyse von 41 BPM-Werkzeugen dargestellt, welche unter anderem die Marktrelevanz, die unterstützten BPM-Aspekte sowie die unterschiedlichen Notationen berücksichtigt. In Abb. 3.2 wird ersichtlich, dass insbesondere der Modellierungsaspekt, gefolgt von der Prozessanalyse, im Hauptfokus der Nutzer steht. Somit wird gefolgert, dass Simulation und die modellgetriebene Softwareentwicklung nicht ausreichend sind. Zudem wird angemerkt, dass durch die Trennung der Prozesslogik und der ausführenden Anwendungen, noch immer keine integrierte Sicht auf die Geschäftsprozesse und Informationssysteme gegeben werden kann.

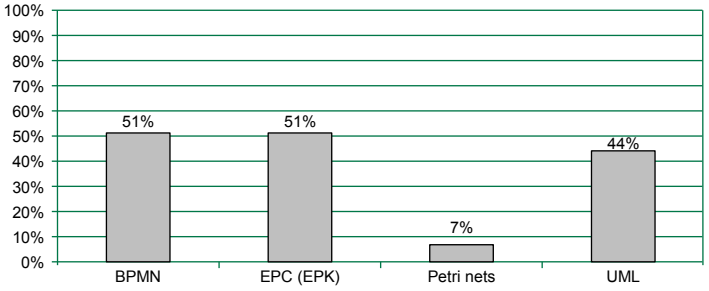


Abb. 3.3: Werkzeugunterstützung von BPM-Notationen nach [56, Abb.7]

Für das in Abb. 3.3 dargestellte Diagramm wird hervorgehoben, dass von 2007 bis 2008 die Unterstützung von BPMN als Notation von 44 % auf 51 % angestiegen ist. Weiterhin

wird dargestellt, dass die EPK insbesondere innerhalb der deutschen Interessensgruppen gefördert wird. Nicht zuletzt wird angemerkt, dass innerhalb der Werkzeuge meist nur eine ausgewählte Umsetzung der UML-Diagramme erfolgt und damit die Unterstützung sehr spezifisch ist. Der Beitrag [56, S.152] zieht das Fazit, dass die Werkzeuge übergreifend eine Nutzung von BPMN unterstützen müssen, um somit dem Trend dieser Notation zu folgen. Unter Berücksichtigung einer ganzheitlichen Unterstützung der genannten Aspekte spiegelt dies einen großen Aufwand für die Umsetzung wider.

3.1.1 Stand der Wissenschaft

In dieser Arbeit wird der BPMN-Ansatz für eine Prozessbeschreibung innerhalb einer integrierten Entwicklungsumgebung für EEAs verfolgt, so dass als zusätzlicher Aspekt eine andere Anwendungsdomäne als gewöhnlich eingebracht wird. Daher werden zunächst allgemeine Methoden und Rahmenwerke im Zuge des Stands der Wissenschaft dargestellt, um nachfolgend den Bogen für die Verwendung von BPMN für automobile Entwicklungsprozesse zu spannen.

3.1.1.1 Software & Systems Process Engineering Meta-Model (SPEM)

Das Software & Systems Process Engineering Meta-Model (SPEM) v2.0[42], ehemalige Bezeichnung „Software Process Engineering Metamodel“, ist in [SPEM] unter der OMG verankert. SPEM basiert auf MOF und verwendet die UML 2 Infrastruktur Bibliothek. Zum einen wird es als Metamodell, zum anderen als UML2-Profil definiert.

SPEM ist selbst keine eigene generische Prozessmodellierungssprache. Das Ziel besteht darin, Software- und Systementwicklungsprozesse zu formulieren indem eine Bandbreite unterschiedlicher Ansätze innerhalb eines Konzept-Rahmenwerks zur Verfügung gestellt und der Entwickler bei der Auswahl des geeignetsten Ansatzes unterstützt wird. In [SPEM] werden die sieben Hauptpakete des Metamodells beschrieben, die dem Anwender für eine standardisierte und strukturierte Arbeit, sowohl in Bezug auf den methodischen Inhalt als auch hinsichtlich des Prozesses zur Verfügung stehen. Für die Inkraftsetzung der Prozesse können unterschiedliche Wege eingeschlagen werden. Erwähnt werden hierbei die Verknüpfung der beschriebenen Prozesse mit Projektplänen oder mit einem Geschäftsfluss bzw. einer Ausführungssprache, um das Ergebnis im Anschluss mit einer beispielsweise BPEL-basierten Workflow-Engine auszuführen [SPEM, S.147]. Es wird der Zusammenhang zu Eclipse Process Framework (EPF) beschrieben, welche auf SPEM basiert und auf einen Großteil der Spezifikationen zurückgreift, indem es Inhalte aus verschiedenen Quellen vereint.

Im Beitrag [75] werden sechs unterschiedliche UML-basierte Sprachen im Kontext von Software-Prozessmodellierung miteinander verglichen. Hierzu werden die Anfor-

[42] http://www.omg.org/spec/SPEM/2.0/ (Zugriff am 15.10.2014)

derungen der semantischen Fülle, Konformität zum UML Standard, grafische Darstellungen und die Unterstützung von verschiedenen Ansichten, Ausführbarkeit, Modularität, Formalität sowie Werkzeugunterstützung in den Fokus gerückt. Im Hinblick auf die derzeitige Praxis in der Softwareentwicklung werden hinsichtlich Letzterer insbesondere die industriellen Standards wie SPEM 1.1 und SPEM 2.0 hervorgehoben und auf eine Unterstützung durch das Eclipse Process Framework (EPF) hingewiesen.

Prozessmodellierung im Kontext von SPEM und ISO 26262 ist Gegenstand der Beiträge [76] und [77], vgl. auch Kapitel 2.3.2.1 im Zusammenhang mit dem Projekt CESAR. Anforderungen aus ISO 26262 wurden in konkrete Aktivitäten übersetzt und Rollen zugeordnet. Um Informationen gemeinsam mit den Eingängen und Ausgängen der Aktivitäten bereitzustellen wurde ein Framework für ein Tabellenkalkulationsprogramm beschrieben. Zusätzlich wurden SPEM [SPEM] und EPF Composer genutzt, um ein Prozess-Framework für sicherheitsbezogene Projekte bereitzustellen.

3.1.1.2 Eclipse Process Framework (EPF)

Das EPF ist ein Open Source Projekt, organisiert innerhalb der Eclipse Foundation, welches 2006 initiiert wurde. Das erste Ziel war ein erweiterbares Rahmenwerk sowie exemplarische Werkzeuge für die Softwareprozessentwicklung und -erhaltung bereitzustellen, welche für viele Projekttypen und Entwicklungsstile geeignet sind [78]. Das EPF unterstützt das Erstellen von Methoden und Prozessen, ein Bibliotheksmanagement sowie die Konfiguration bzw. Veröffentlichung von Prozessen. Ein zweites Ziel bestand in der Bereitstellung von exemplarischen und ausbaufähigen Prozessinhalten, die für verschiedene Softwareentwicklungen und Managementprozesse anwendbar sind, flexible und iterative Entwicklungen unterstützen sowie auf andere Entwicklungsplattformen und Anwendungen übertragbar sind. Der Rahmen des EPF besteht aus einem formalen Metamodell. Durch dieses wird der methodische Inhalt sowie der Prozess strukturiert und durch den Gebrauch von MOF, UML-Diagrammen sowie einem zugehörigen XML-Schema dokumentiert. Das initiale Metamodell wurde aus der „IBM's Unified Method Architecture" (UMA) abgeleitet, welches eine Erweiterung von SPEM v1.1 darstellt und Elemente des „IBM Rational Unified Process", „IBM Global Services" und „IBM Rational Summit Ascendant" integriert. Die aktuelle Version EPF 1.5.1.6 wurde im Januar 2014 veröffentlicht.

In [153] wurde im Kontext von Softwareentwicklung ein integriertes und interpretierbares Prozessmodell vorgeschlagen, um die „Lücke zwischen informeller Dokumentation und formalen Metamodell-Spezifikationen" zu verkleinern. Für die prototypische Implementierung wurde auf EPF sowie das Rahmenwerk Java Workflow Tooling (JWT)[43] zurückgegriffen. EPF wurde hierbei zur Modellierung der Softwareentwicklungsprozesse verwendet, um existierende Referenzprozesse aus anderen Projekten der automobilen

[43] http://www.eclipse.org/jwt/ (Zugriff am 15.10.2014)

Softwareentwicklungsdomäne, wie beispielsweise AUTOSAR und MAENAD, nutzen zu können [153, S.233].

3.1.1.3 Architecture of Integrated Information Systems (ARIS)

In [79, S.366] wird eine allgemeine Architektur für Geschäftsprozesse beschrieben, welche auf ARIS basiert. Hauptbestandteil von ARIS ist die EPK [142, S.15] und wird zur Beschreibung der Prozessschicht eingesetzt [154, S.21]. ARIS stellt zum einen ein Konzept zur „Beschreibung von Unternehmen und betriebswirtschaftlichen Anwendungssystemen“ [154, S.3] dar und wird zum anderen als ein Softwaretool umgesetzt. Das gleichnamige Tool wird im Rahmen des Stands der Technik in Kapitel 3.1.2.2 beschrieben.

ARIS wird zur Unterstützung der Unternehmen bei der „Modellierung, Analyse und Optimierung von Prozessen“ verwendet [154, S.1]. Das „ARIS-Haus“ der Geschäftsentwicklung wird dazu in vier Ebenen eingeteilt. Neben der Prozessentwicklung gehören die Prozessplanung und -kontrolle sowie die Kontrolle des Arbeitsflusses und das Anwendungssystem dazu [79, S.368].

Im Modell werden zudem verschiedene Entwicklungsstufen dargestellt, von den betriebswirtschaftlichen Anforderungen bis hin zur technischen Implementierung. Die ARIS-Architektur gliedert sich in folgende Schichten: Daten-, Funktions- und Organisationsschicht, vgl. dazu [54, S.4]. Durch die Steuerungs- bzw. Prozessschicht wird eine Verbindung zwischen den genannten Schichten hergestellt [154, S.14]. Die Abstraktionsebenen werden in das Fachkonzept, das Datenverarbeitungskonzept sowie die Implementierung aufgeteilt [54, S.5].

3.1.1.4 PICTURE-Methode

Die Arbeit [80] beschreibt, dass Prozessmodellierung bedeutend wichtiger wird für andere Anwendungen neben der Softwareentwicklung. Daher werden allgemeine Techniken zur Anpassung von Prozessmodellen in Bezug auf spezifische Verwendungszwecke präsentiert.

Der DIN-Fachbericht 158 [DINfb158] bezieht sich auf prozessorientiertes Vorgehen in der öffentlichen Verwaltung. Es werden die Prozessmodellierungsmethoden BPMN und PICTURE aufgegriffen. Aus der PICTURE-Methode, welche am Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster entwickelt wurde, ging die webbasierte PICTURE-Prozessplattform der Firma PICTURE GmbH[44] hervor. Hierbei werden unter anderem visuelle und wiederverwendbare Prozessbausteine verwendet. Besonderes Augenmerk liegt auf dem Prozessmanagement, der Prozessdokumentation

[44] http://www.picture-gmbh.de (Zugriff am 15.10.2014)

und Prozessverbesserung. PICTURE wurde speziell für die öffentliche Verwaltung entwickelt und stellt eine auf die Domäne angepasste Beschreibung, Analyse und Optimierung zur Verfügung. Die PICTURE-Methode stellt „Geschäftsprozesse aus einer fachlich-organisatorischen Perspektive“ dar. Beide Modellierungsmethoden werden im informativen Anhang D/E exemplarisch dargestellt. Bezogen auf BPMN wird zusätzlich der Kontext von BPEL eingebracht. Im DIN-Fachbericht 158 zeigt sich deutlich, dass sowohl spezifische Lösungen eingesetzt werden können, jedoch auch die etablierte BPMN genutzt werden kann.

3.1.1.5 Design-Methodik nach MAENAD

Im Rahmen von MAENAD wurde hinsichtlich der Design-Methodik nach [81] zur Modellierung von automobilen Prozessen die BPMN 2.0 genutzt. Hierzu wurde das Eclipse-basierte Tool ADONIS eingesetzt. Für den Nutzer wird die MAENAD-Methodik als Dateisatz im Hypertext Markup Language (HTML)-Format zur Verfügung gestellt. Hierdurch wird ein einfaches Navigieren in den einzelnen Prozessen ermöglicht. Es gab auch die Überlegung, SPEM im Rahmen von EPF einzusetzen.

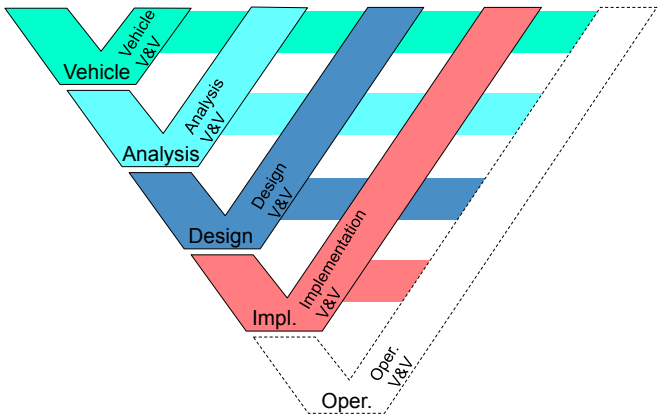


Abb. 3.4: V-Modell als Referenz für die Ebenen nach EAST-ADL in [23, F.6]

Die dahinterstehende Methodik berücksichtigt über vier Phasen die Prozessbeschreibung im Kontext von EAST-ADL und FEV, welche sich an den einzelnen Abstraktionsebenen von EAST-ADL orientieren und als Pools angelegt wurden, siehe hierzu auch Abb. 3.4. Zudem wurden für jede Phase mehrere Swimlanes angelegt, welche entsprechend den Rahmenbedingungen zugeschaltet werden können. Folgende Swimlanes werden berücksichtigt und repräsentieren individuelle Aspekte: Core, Functional Safety, Timing und Fully Electric Vehicle. Jede der Swimlanes wird nach einem generischen Muster strukturiert [81, S.10]: 1. Verfeinerung, Einführung und Validierung von Anforderungen;

2. Erstellung einer Lösung; 3. Zuordnung von Anforderungen auf Lösung; 4. Erstellung unterstützender Modelle; 5. Analyse des erweiterten Modells; 6. Verifikation der Lösung gegen die Anforderungen; 7. Spezifikation und Validierung der Anforderungen.

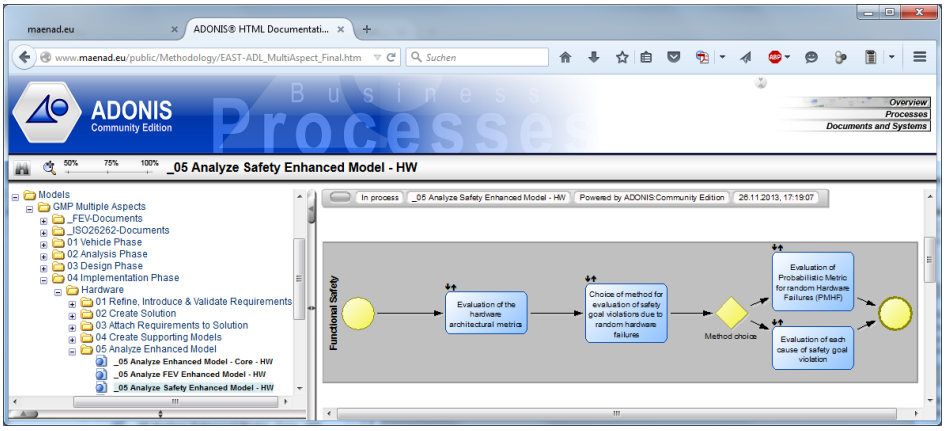


Abb. 3.5: Auszug eines Prozessmodells aus dem MAENAD Projekt, umgesetzt mit ADONIS [82]

In Abb. 3.5 ist ein Auszug aus den Prozessmodellen des MAENAD Projektes ersichtlich. Dies zeigt sowohl Aktivitäten als auch Start- und Endereignisse. Zudem sind Informationen in Form von Beschreibungen hinterlegt. Pools und Swimlanes wurden anstelle von Verantwortlichkeiten im Sinne der BPMN für einzelne zu berücksichtigende Aspekte verwendet. Die Granularität des Modells ist auf einer sehr hohen Ebene angesetzt.

3.1.1.6 SAFE Engineering Process

Im Rahmen des Projektes SAFE konnte in [D6a, S.63ff] eine erste allgemein gehaltene Prozessstruktur hinsichtlich dem Umgang von Sicherheitsaktivitäten und Arbeitsprodukten dargestellt werden, siehe Abb. 3.6. Diese wurde in den Kontext der Veränderung von Anforderungen, Design oder Architektur und zudem in Bezug zum Standard-Entwicklungsprozess gesetzt. Die Darstellung erfolgte in Enterprise Architect als BPMN-Diagramm. Durch den aktuell sich in Arbeit befindenden SAFE Engineering Process (SEP) sollen die geforderten Prozesse detailliert aufgeschlüsselt werden.

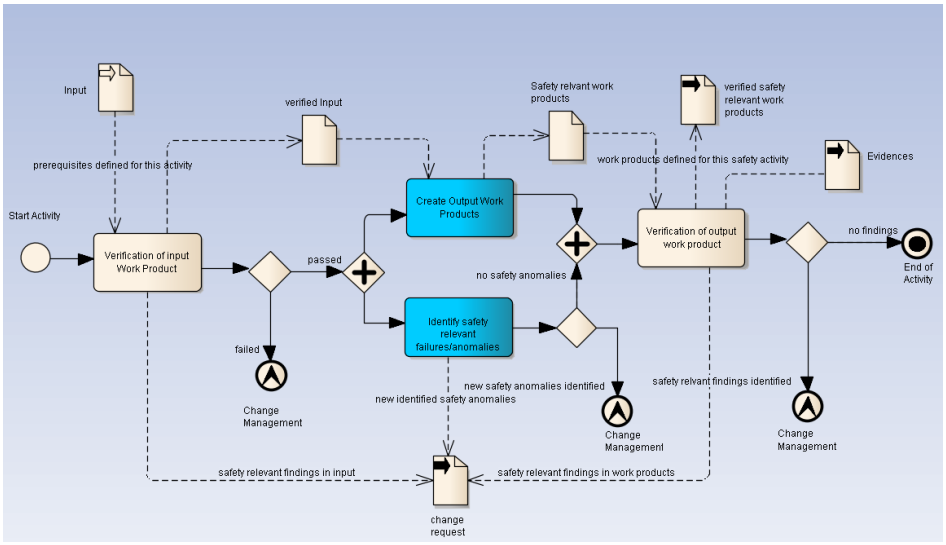


Abb. 3.6: Grundlegende identifizierte Prozessstruktur für Aktivitäten nach ISO 26262 [D6a, Fig.50]

3.1.1.7 Verwendung von BPMN-Konstrukten

Aus dem in Abb. 3.7 dargestellten Diagramm, welches auch in [138, S.227] aufgegriffen wird, lässt sich herauslesen, dass die Flusselemente zusammen mit den Ereignissen von höchster Bedeutung für die Modellierer im Kontext von BPMN sind. Im ergänzenden Beitrag [83] wird die Vermeidung bestimmter BPMN-Konstrukte insbesondere darauf zurückgeführt, die Verwirrung bei der Interpretation von Prozessmodellen zu begrenzen.

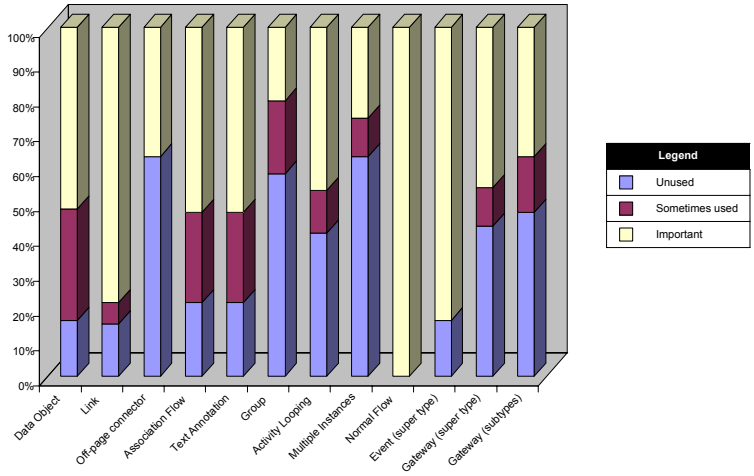


Abb. 3.7: Verwendung von ausgewählten BPMN-Symbolen in [84, S.7]

3.1.1.8 Zukunftsthema Geschäftsprozessmanagement

Im Rahmen einer Studie zu Geschäftsprozessmanagement, durchgeführt von der PricewaterhouseCoopers AG (PwC) gemeinsam mit der Bayerischen Julius-Maximilians-Universität Würzburg, konnte die Wichtigkeit des Zukunftsthemas Geschäftsprozessmanagement erarbeitet werden [85]. Kriterien nach denen aufgeschlüsselt wurde, waren die „Branche, der Unternehmensumsatz, die Position des Befragten im Unternehmen und das Land“. Die Studie, welche Ende 2010 durchgeführt wurde, basiert auf Befragungen von insgesamt 239 Führungskräften. Bezogen auf die Selbsteinschätzung der Firmen auf den Entwicklungsstand ihres eigenen Geschäftsprozessmanagements liegt die Branche der Automobilindustrie und Zulieferer unter dem Durchschnitt, verglichen zu beispielsweise Handel, Chemie und Pharma. Aus den Umfrageergebnissen geht hervor, dass sich die Unternehmen aktuell in der „Übergangsphase zwischen der Definition einzelner ausgewählter Geschäftsprozesse hin zu einer systematischen Dokumentation (Prozessmodellierung) aller relevanten Prozesse befinden“. Gedeutet wird dies damit,

dass Unternehmen derzeit ein „standardisiertes Vorgehen entdecken“, um sich „zunächst einen Überblick über ihre Prozesslandschaft zu verschaffen“. Jedoch sind sie „noch weit von einem systematischen, nachhaltigen und damit unternehmensweiten Vorgehen entfernt“. Dieser Trend zeichnet sich durch die rechnergestützte „Modellierung der Geschäftsprozesse und die vermehrte Einführung von Standardnotationen wie zum Beispiel der BPMN“ ab. Folglich sind integrierte Geschäftsprozesse einer der wichtigsten Bausteine, um sich „flexibel an die laufenden Veränderungen“ anzupassen. Somit müssen die Geschäftsprozessstandardisierung und -harmonisierung weiter vorangetrieben werden, da das Geschäftsprozessmanagement bei den „befragten Unternehmen, unabhängig von ihrer Größe und Branche, ganz oben auf der Agenda zur Organisationsentwicklung“ steht. Zudem wird ein direkter Zusammenhang zwischen den Aktivitäten im Geschäftsprozessmanagement und dem Unternehmenserfolg gesehen.

3.1.2 Stand der Technik

Die Modellierung und das Management von Prozessen wird zunehmend angewandt und stetig vorangetrieben. Nachfolgend ist ein Auszug sowohl freier Werkzeuge als auch kommerziell verfügbarer Toolumgebungen aufgelistet. Diese werden in Kürze beschrieben und stehen hauptsächlich im Fokus von Petri-Netzen und BPMN.

3.1.2.1 Horus

Das Horus®-Konzept der Horus software GmbH stellt eine Methode zur Modellierung, Analyse, Simulation und Optimierung von Geschäftsprozessen zur Verfügung, vgl. [155]. Der Ansatz gliedert sich in die Bausteine der Horus Methode, Horus Enterprise als professionelles Werkzeug und Horus Endeavor, unter anderem für die wissenschaftliche Weiterentwicklung [86]. Die Horus Methode umfasst grundlegend zum einen die Phase der Strategie und Architektur, zum anderen die Phase der Geschäftsprozessanalyse. Horus Enterprise dient als Softwarewerkzeug zur Anwendung der Horus Methode. Dazu gehört der Horus Business Modeler[45] sowie Referenzmodelle. Als Modellierungssprachen werden unter anderem Petri-Netze (XML-Netze), Organigramme sowie semantisch-hierarchische Strukturen bereitgestellt. Weiterhin werden automatisierte Schnittstellen zu BPMN 2.0 und BPEL zur Verfügung gestellt. Über die sogenannte Horus Cloud wird ein Zugang zu den Horus Tools bereitgestellt. Horus Endeavor stellt den Horus Business Modeler als Freeware zur Verfügung.

Angemerkt sei, dass in der betreuten Arbeit [Schwaer2013] Horus für eine erste Prozessbeschreibung der ISO 26262 verwendet werden konnte.

[45] http://www.horus.biz (Zugriff am 15.10.2014)

3.1.2.2 ARIS

ARIS[46] [47] wurde durch die Software AG von der IDS Scheer übernommen. Der Toolname des Geschäftsprozessanalyse-Werkzeugs stammt vom gleichnamigen Konzept der Architecture of Integrated Information Systems (ARIS) ab. Die Software AG stellt unter der ARIS Business Process Analysis Platform verschiedene Produkte für die Prozessoptimierung bereit. Aktuelle Version der Software ist 9.6. Zu den Neuerungen gehören unter anderem das Bereitstellen einer Cloud-Anwendung sowie die Nutzung sozialer Medien. ARIS unterstützt die Modellierung nach BPMN 2.0 einschließlich von Simulationen. Auch eine Interoperabilität mit SAP als Tool der SAP AG ist vorhanden. Des Weiteren existiert eine freie Modellierungssoftware, ARIS Express, aktuelle Version 2.4. In [156] finden sich verschiedene Anwendungsbeispiele der ARIS-Software.

3.1.2.3 Signavio

Signavio Process Editor[48] von der Signavio GmbH ist ein webbasiertes Prozessmodellierungs-Werkzeug und konform zum BPMN 2.0 Standard. Dieser bietet unter anderem das Anlegen eigener Attribute für Prozesselemente, Simulation von Prozessdurchläufen sowie Import- und Exportschnittstellen, darunter auch den Import für ARIS Markup Language (AML). Des Weiteren bietet das Reporting die Möglichkeit automatisch ein Prozesshandbuch zu erstellen oder Kostenrechnungen sowie Ressourcenbedarfsrechnungen durchzuführen. Über sogenannte Prozessmodellmetriken können zudem Statistiken über die einzelnen Diagramme erhoben werden.

3.1.2.4 Enterprise Architect

Enterprise Architect, aktuelle Version 11, von SparxSystems Software GmbH ist ein Modellierungswerkzeug zur Unterstützung der Softwareentwicklung sowie der Entwicklung von eingebetteten und Echtzeitsystemen. Es unterstützt unter anderem UML und SysML. Ab Version 9 wurde auch BPMN 2.0 [87] für die Prozessmodellierung und aus deren Modellen die Erstellung von ausführbaren BPEL-Skripten unterstützt. Verfügbar sind in Enterprise Architect Version 9.3 unter der Technologie „Core Extensions“ im Bereich der UML auch sogenannte „Business Process“ Diagramme [157]. Hierbei lassen sich über Geschäftsprozessmodelle Verhaltens- und Informationsflüsse darstellen. Dies gilt sowohl für Organisationen als auch Systeme. Weiterhin werden verschiedene Profile bereitgestellt, welche beispielsweise SPEM-Diagramme bzw. das Einbinden von eEPK-Symbolen ermöglichen.

[46] http://www.softwareag.com/de/ (Zugriff am 15.10.2014)
[47] http://www.ariscommunity.com/aris-express (Zugriff am 15.10.2014)
[48] http://www.signavio.com (Zugriff am 15.10.2014)

Für eine erste Beschreibung des SAFE Engineering Process (SEP) [D6a] wurde unter anderem Enterprise Architect verwendet, vgl. Abb. 3.6.

3.1.2.5 ADONIS

Das Tool ADONIS von BOC Information Technologies Consulting AG ist Teil der Produktfamilie BOC Management Office und bietet einen Rahmen für ein Geschäftsprozessmanagement. Die aktuelle Version 6.0 wurde im Juli 2014 herausgegeben [88]. Das Werkzeug verwendet den fachlichen Ansatz der BPMN 2.0. Neben der Unterstützung der informationstechnischen Umsetzung werden zudem Metamodellierungskonzepte bereitgestellt [89]. Es ermöglicht einen kontinuierlichen Verbesserungsprozess und bietet vielfältige Funktionen zur Prozessanalyse und -simulation [90]. Auf Grundlage der Analysen werden zudem Funktionalitäten zur Prozesskostenrechnung bzw. Personalbedarfsplanung ermöglicht. Zur Prozesssimulation werden unterschiedliche Algorithmen bereitgestellt, die je nach Szenario verwendet werden können. Hierbei stehen exemplarisch die Pfadanalyse, Belastungsanalyse und Auslastungsanalyse im Fokus. Über ein sogenanntes ADONIS Prozessportal können die Prozessmodelle webbasiert zur Verfügung gestellt werden. ADONIS unterstützt verschiedene Schnittstellen, beispielsweise XML, die ADONIS Definition Language (ADL), API-basierte Schnittstellen sowie im Zusammenhang mit der Prozessumsetzung unterschiedliche Standards wie BPMN-DI, BPEL, XMI bzw. XPDL. ADONIS wurde zur Modellierung der MAENAD-Methodik [81] verwendet, siehe Abb. 3.5.

3.1.2.6 Atego Process Director

Die Atego Process Director Family[49] stellt den ategoProcessDirector™ und ategoProcessConsumer™ zur Verfügung. Atego bietet mit dem ategoProcessDirector™ über ein Bibliothekskonzept die Möglichkeit, auf vorhandene Prozessbeschreibungen, welche beispielsweise aus Standards abgeleitet sind, zurückzugreifen oder diese neu zu erstellen. Hierbei können auch spezifische Informationen oder Diskussionen hinterlegt werden. Basierend auf dieser Bibliothek zur Prozessdefinition können anhand einer Übersicht der Prozessstruktur neue Prozesse erstellt und per drag-and-drop über eine grafische Benutzeroberfläche projektspezifisch angelegt werden. Verantwortlichkeiten können eingebracht und zugewiesen werden. Eine automatische Erstellung eines Microsoft Project Plans mit Verknüfungen zu den entsprechenden Prozessteilen aus dem ategoProcessDirector™, wird zur Verfügung gestellt. Im Kontext der ISO 26262 wird eine erste Modellierung mit dem ategoProcessDirector™ in [91] vorgestellt.

[49] http://www.atego.com/products/atego-process/ (Zugriff am 15.10.2014)

3.1.3 Abgrenzung

Der Stand der Wissenschaft und Technik sowie die in den Grundlagen vorgestellten Notationen zeigen deutlich auf, dass bereits sowohl Formalisierungen und zahlreiche Methoden für BPM vorliegen als auch die entsprechenden Implementierungen innerhalb von Werkzeugen bereitgestellt werden. Dies erstreckt sich von der eigentlichen Modellierung über die Analyse und Simulation von Prozessen sowie deren grafischen Darstellungen bis hin zur Bereitstellung von Bibliothekskonzepten. Jedoch ist eine integrierte Sicht auf die Prozesse sowie der Bezug zum Anwendungssystem meist nicht gegeben, vgl. hierzu auch [56].

Bezogen auf die automobile EEA-Entwicklung und insbesondere im Kontext der Produktentwicklung von sicherheitsbezogenen Systemen ist der prozessorientierte Standard ISO 26262 normativ. Hierzu sind erste Ansätze einer Prozessmodellierung vorhanden, jedoch ist nirgends ein vollständiges Prozessmodell mit etablierten Darstellungen sowie Möglichkeiten zur Planung und Analyse gegeben. Somit wird sowohl den Sicherheitsverantwortlichen als auch den beteiligten Entwicklern bisher keine integrierte Sichtweise auf die Entwicklungsprozesse einschließlich der einzuhaltenden Prozesse zu funktionaler Sicherheit ermöglicht.

Des Weiteren bieten die vorgestellten Ansätze keine Möglichkeit, die Aktivitäten und Arbeitsprodukte im Kontext funktionaler Sicherheit mit Modellelementen im Rahmen einer produktiven Entwicklungstätigkeit in Bezug zu setzen. Dies könnte auch eine automatisierte Ausführung von Teilprozessen, wie beispielsweise geforderte Sicherheitsevaluationen, unterstützen.

3.2 Hardware-Sicherheitsevaluationen

Modellbasierte Ansätze und Entwicklungsumgebungen für EEAs wie EAST-ADL, EEA-ADL und AADL ermöglichen insbesondere die strukturelle Beschreibung von *Hardwaredesigns*. Die aktuellen Ansätze der Wissenschaft und Technik zur Durchführung von Hardware-Sicherheitsevaluationen in diesem Kontext sind im Folgenden dargestellt.

3.2.1 Stand der Wissenschaft

3.2.1.1 Entwicklungsprozess für integrierte Sicherheitssysteme

Die Arbeit [126] beschreibt Anforderungen und Konzepte zukünftiger Elektronik-Architekturen im Kontext von integrierten Sicherheitssystemen. In der Arbeit wird ein Entwicklungsprozess mit einer dazugehörigen Umsetzung zur Entwicklung einer Hardwarearchitektur sowie einer Softwareplattform vorgestellt. Frühzeitige Konzeptabsicherungen sowie eine verlässliche Hardware (HW)-Architektur werden durch diesen Ansatz ermöglicht.

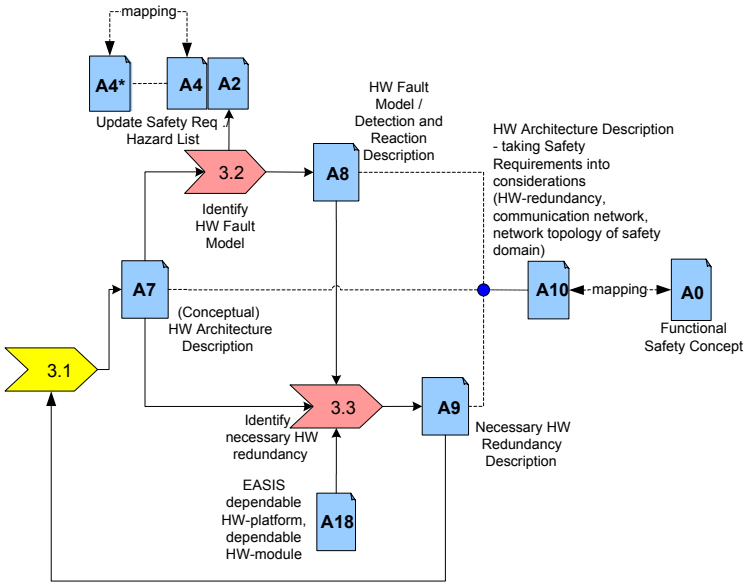


Abb. 3.8: „Entwicklungsschritte für die Hardware-Architektur“ als Teil des Entwicklungsprozesses für integrierte Sicherheitssysteme in [126, S.78]

Die Entwicklungsschritte für die HW-Architektur können Abb. 3.8 entnommen werden. Hier wird bereits ein Bezug zu den Anforderungen für Hardware aus der ISO 26262 her-

gestellt und diese aufgelistet sowie gegenübergestellt [126, S.94]. Die Analyse der Auswirkung von Ausfällen der Hardware sowie die Einführung entsprechender Sicherheitsmaßnahmen wird ebenfalls aufgegriffen. Im Zusammenhang mit integrierten Sicherheitssystemen wird auf das EASIS Projekt verwiesen, das sich ebenfalls mit der Entwicklung von fehlertoleranten elektronischen Architekturen in Fahrzeugen sowie Ansätzen für standardisierte Systeme befasste [126, S.53], vgl. Kapitel 2.3.2.1.

3.2.1.2 Modellbasierte Unterstützung für Sicherheitsanalysen

Der Beitrag [92] stellt einen Ansatz zur ECU-Entwicklung vor, da die technische Realisierung, Dokumentation sowie die Konformität zu ISO 26262, unter Berücksichtigung der geforderten Kriterien einer effizienten Planung, Design und Implementierung, neue und effiziente Lösungen erfordern. Der Weg zu einer erfolgreichen ECU-Entwicklung wird in Form eines systematischen Ansatzes gesehen, der durch die Model-Based Systems Engineering (MBSE)-Methode unterstützt werden soll. Ziel ist die Steigerung der Effizienz der Produktentwicklung, indem aus einer Vielzahl an Designalternativen die ideale Umsetzung identifiziert wird, um Herausforderungen, die sich auf den Erfolg dieser Entwicklung auswirken, zu bewältigen. Dies betrifft unter anderem den Umgang mit Komplexität in Funktionalität, Technologie und Produktvariabilität sowie Sicherheitsaspekte und Kosten.

Hierbei sollen insbesondere die Möglichkeiten einer Top-Down gesteuerten Entwicklung unter Berücksichtigung von Anforderungen, semi-formaler Architekturbeschreibung und früher Unterstützung der ISO 26262 im Kontext von ASIL-D ausgenutzt werden. Der Kerninhalt des TRW-Ansatzes ist es, die Design-Informationen (Systemstruktur und Verhalten) eines ECU-Modells in SysML zusätzlich für die Entwicklung des ECU-Sicherheitskonzeptes zu verwenden, um Sicherheitsaspekte direkt im Systemmodell zu integrieren und dadurch Änderungen im Design mit hoher Effizienz erneut analysieren zu können.

Der MBSE Ansatz basiert auf drei grundlegenden Prinzipien:

1. Modellbasiertes Design inklusive Anforderungsmanagement: Im Vergleich zu dokumentenbasierten Vorgehensweisen, bei denen Anforderungen jeder Abteilung in Dokumenten notiert werden, die aufgrund fehlender strukturierter Verifizierungsprozesse jedoch fehleranfällig sind, beugt der modellbasierte Ansatz durch eine Sicherung der Informationen auf Modellebene vor. Spezifikationen werden in einem Tool für Requirements Management (RM) festgehalten. Dieses besitzt eine Schnittstelle zum Modellierungstool, in welchem Modelle über Beschreibungssprachen wie SysML oder UML angefertigt werden. Das RM sichert die Nachvollziehbarkeit, während die Modellierung unterschiedliche Darstellungsformen bietet und im Modell die einzelnen Anforderungsebenen zusammengefügt werden.

2. Integrative und iterative Sicherheitsanalysen: Der modellbasierte Design-Ansatz ermöglicht eine iterative Vorgehensweise, die aufgrund der sich ständig ändernden Anforderungen, in der Summe ca. 2 % pro Monat, von Nöten ist. Eine einzige Analysephase, wie in V- bzw. Wasserfallmodellen, ist in diesem Kontext nicht anwendbar. Beim modellbasierten Design werden die Anforderungen mit den Modellelementen verbunden und dadurch bei Änderungen die Auswirkungen auf das Design einfacher erkannt. Die Änderungsinformationen werden an das Sicherheitsanalysetool weitergeleitet, so dass festgestellt werden kann, ob der Sicherheitsnachweis beeinträchtigt ist.

3. Unterstützung der Varianten durch hochgradig wiederverwertbare Artefakte: Um redundante Arbeit zu vermeiden, besteht die Herausforderung darin, durch nur einen Sicherheitsnachweis eine komplette Produktfamilie oder Plattform abzudecken. Als Lösung werden geschickt gewählte Abstufungen der Abstraktionsebenen, Variationen in Bezug auf die Modellelemente sowie Variationspunkte auf einer niederen Ebene hinsichtlich der Analyse der zufälligen Ausfälle der Hardware angesehen.

Der Design-Prozess basiert auf den drei Werkzeugarten: RM-Tool, SysML-Modellierungstool sowie ein Sicherheitsanalysetool. Ihre Interaktion unterstützt den kompletten Entwicklungslebenszyklus. Im ersten Schritt sind neben den gestellten Anforderungen, Quellen, Stand der Analyse sowie die Verantwortlichkeiten zu identifizieren und im RM-Tool zu hinterlegen. Diese werden mit Hilfe einer Schnittstelle (z.B. DOORS Gateway) mit dem Modellierungstool verbunden, um eine Überprüfung auf Vollständigkeit zu vereinfachen. Über eine Regel kann die Verknüpfung zwischen jedem Architekturelement und mindestens einer Anforderung überprüft werden. Durch MBSE wird von der konzeptuellen Designphase bis zu den späteren Phasen des Lebenszyklus eine formalisierte Anwendung der Modellierung geboten. Dies ermöglicht die Erhebung der Systemanforderungen, die Designanalyse sowie Verifikations- und Validierungsaktivitäten.

Das Modell beruht auf einer engen Interaktion zwischen RM-Tool und dem Modellierungstool. Die Ergebnisse der Architekturdesign-Phase werden an weitere Domänen wie Hardware oder Software zur Implementierung weitergegeben. Die Sicherheitsanalyse überlappt mit den Anforderungen und der Modellierung, so dass die beiden anderen Tools mit einer Schnittstelle zum Sicherheitstool ausgestattet sind. Das System muss somit nicht erneut in Bezug auf die Sicherheitsanalyse modelliert werden und es kann vermieden werden, dass Verknüpfungen zwischen der Sicherheits- und Designentwicklung fehlen.

Das Modell ist in zwei hierarchische Ebenen strukturiert, die logische Ebene („Problemdomäne“) sowie die physikalische Ebene („Lösungsdomäne“). In der logischen Ebene wird das System mit den grundlegenden Funktionen ausgestattet, die wiederverwendet werden können. Je nach Funktionsansprüchen können aufbauend Highend-Systeme mit weiteren Funktionsinhalten ausgestattet werden. Der Ansatz ist durch diese stabile Grundlage auf unterschiedliche physikalische Lösungen anwendbar. Diese Stabilität hat

Auswirkungen auf die Sicherheitsanalyse, da die funktionalen Netze an die FMEA und FTA weitergereicht werden. Zur Modellierung der funktionalen Netze werden in SysML Aktivitätsdiagramme verwendet, welche durch Swimlanes strukturierbar sind.

Diese Vorgehensweise bietet sich auch für die Evaluation von verschiedenen Architekturen an. Dadurch, dass modellierte Operationen der logischen Ebene mit der physikalischen Ebene, z.B. Blöcke und Parts, in Bezug gebracht werden, können Änderung der Architektur zuverlässig erkannt und die Auswirkungen evaluiert werden.

Um die Effizienz der Sicherheitsanalyse zu erhöhen, schlagen die Autoren vor, die Schnittstelle zwischen den Sicherheitsanalysemethoden, wie FTA und den Hardware-Architekturmetriken (z.B. SPFM/LFM), mit dem SysML-Modell zu verbessern. Zwei Aspekte, die Toolschnittstelle und Prozessschnittstelle, sollen dabei berücksichtigt werden.

In diesem Zusammenhang muss sichergestellt werden, dass alle beteiligten Entwicklungsbereiche hinsichtlich der Modellierung und Sicherheitsanalyse gleich ausgerichtet sind. Je nach Harmonisierung sind verschiedene Analysen möglich. Im Falle eines ausführbaren Modells kann eine (semi-)automatische FTA durchgeführt werden. Bezogen auf den vorliegenden Vorschlag der ECU-Architektur stehen die statischen und dynamischen Aspekte im Vordergrund und eine Ausführbarkeit des Modells wird nicht gefordert.

Der iterative Prozess zur Verbesserung einer ECU-Architektur beruht auf einer Weitergabe der SysML-Informationen in das Sicherheitsanalysetool, in dem eine FTA und die Hardware-Architekturmetriken durchgeführt sowie die Anforderungen unter Berücksichtigung von z.B. ASIL modelliert werden. Die erhaltenen Sicherheitsanforderungen werden dann an das RM-Tool zurückgegeben. Es wird vorgeschlagen den Top-Down Ansatz zu beenden, wenn alle Hardware (HW)- und Software (SW)-Architekturkomponenten erfasst sind und ihnen entsprechende Sicherheitsanforderungen zugeordnet wurden. Auf die erste Iteration des detaillierten Designs der EE-Architektur und die Weitergabe der EE-Komponentenliste an das Sicherheitsanalysetool, folgt eine Bottom-Up Analyse, wobei systematische und zufällige Ausfälle der Hardware für alle EE-Komponenten definiert werden. Das Sicherheitsanalysetool unterstützt die Methoden nach ISO 26262 zur Evaluation der zufälligen Ausfälle der Hardware. Die systematischen Ausfälle werden durch eine Design-FMEA bewertet. Zudem wird die Sicherheitsanalyse durch Modellprüfungen auf Konsistenz und Vollständigkeit sowie einem integrierten Fehlerreport unterstützt.

Ziel des Ansatzes ist es, durch eine verbesserte Schnittstelle zwischen der Sicherheitsanalyse und dem Modell der Systemarchitektur den Änderungsmanagement-Prozess zu unterstützen. Die Auswirkungen von Veränderungen in der Systemarchitektur können durch eine toolbasierte Unterstützung zwischen den Modellelementen und der Sicherheitsanalyse sicher nachvollzogen werden. Durch den Ansatz kann dem Anspruch der ISO 26262, Sicherheitsaspekte während einer Entwicklung frühzeitig zu berück-

sichtigen, nachgekommen werden. Der Ansatz minimiert das Risiko, dass Sicherheitsprobleme erst zu einem späten Zeitpunkt entdeckt werden.

Die Einführung der MBSE als semi-grafische Notationsform innerhalb eines Unternehmens bedeutet, dass alle beteiligten Abteilungen geschult werden müssen und über ein sukzessives Ausrollen nachgedacht werden muss. Im Beitrag wurden alle Erfahrungen, die im Zusammenhang mit der Einführung des MBSE im Rahmen einer ECU-Entwicklung bei TRW Braking gemacht wurden, zusammengetragen und bewertet. Diese zeigen, dass die Effizienz der ECU-Entwicklung sowie der EE-Sicherheitsanalyse durch das Einführen einer Methode und einer Tool-Schnittstelle zwischen dem RM-Tool, dem ECU-Architekturmodell in SysML und dem Sicherheitsanalysemodell gesteigert werden könnte. Die nächsten Schritte von TRW bestehen darin, Anstrengungen zu unternehmen, um die Formalisierung des Modells zu erhöhen und eine Modellausführung zu erreichen. Es soll ebenfalls untersucht werden, wie die Modellierungseffizienz verbessert werden kann, indem semi-formale grafische Notationen wie SysML mit formalisierten textuellen Notationen kombiniert werden.

3.2.1.3 Anwendung der FRC-Methode

Im Beitrag [93], welcher den Beitrag [94] erweiternd aufgreift, wird auf die Anwendung der Failure Rate Class (FRC)-Methode während der Entwicklung am Fallbeispiel eines „Electric Power Steering (EPS)“ eingegangen, welches ASIL-D eingestuft ist. Erwähnt wird, dass für die Evaluation von Hardware eine Zuverlässigkeitsanalyse eine der entscheidenden Grundvoraussetzung ist. Hierbei werden bewährte Praktiken für die Abschätzung von Ausfallraten der *Hardware-Bauteile* aufgezeigt. Für die Skalierung von Ausfallraten wird auf [ISO26262, 5-Tab.F.1] oder auf die Arrhenius-Gleichung verwiesen. Für die Ausfallratenverteilung wird [IECTR62380] als Standardwerk genannt, für Konnektoren [MIL338B]. Ein Vergleich der beiden Methoden zur „*Evaluation of safety goal violations due to random hardware failures*“ nach Clause 9 aus Part 5 ergibt nach Ansicht von [93] Vorteile bei der Nutzung der FRC-Methode. Diese bietet unter anderem eine eher robuste Analyse. Auch im Zuge verteilter Entwicklungen lässt sich die FRC-Methode besser und effizienter anwenden, da diese keinen Mehraufwand für die Zuweisung von Zielwerten aufweist. Zudem wird eine individuelle Untersuchung jedes einzelnen *Hardware-Bauteils* in Bezug zu einer FMEA gesehen und daher wird eine weitere Untersuchung auf Item-Ebene im Rahmen einer Probabilistic Metric for random Hardware Failures (PMHF) nicht mehr als notwendig erachtet. Eine Werkzeugunterstützung zur weiteren Kombination von FMEA und der FRC-Methode innerhalb einer durchgängigen und konsistenten Methodik ist von den Autoren gewünscht, da die existierenden Analysetools Einschränkungen aufweisen.

3.2.1.4 Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS)

Die Methode Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) erlaubt eine integrierte Bewertung von komplexen Systemen durch Verwendung der etablierten Methoden FMEA und FTA [158]. Die HiP-HOPS-Untersuchung wird in drei Hauptphasen beschrieben [68], bei denen die letzten zwei Phasen automatisiert sind: Während der ersten Phase wird das Systemdesign modelliert und Fehlerinformationen werden in der Systembeschreibung annotiert. Dies wird manuell in einem Modellierungstool durchgeführt, wie beispielsweise Matlab® Simulink®, Eclipse-basierte UML-Tools oder SimulationX® von der ITI GmbH . Während der zweiten Phase, die Fehlerbaum-Synthese, verfolgt HiP-HOPS automatisch den Pfad der Fehler-Propagation im Systemmodell und baut ein Netzwerk von zusammengeschalteten Fehlerbäumen auf. Dieser Prozess arbeitet rückwärts von den zu untersuchenden Hauptereignissen. Die letzte Phase bestimmt die Minimalschnitte und berechnet die Wahrscheinlichkeit eines Systemfehlers als eine quantitative Analyse [68]. Weitere Arbeiten im Kontext von HiP-HOPS sind in Bezug auf FMEA in [95] und im Rahmen einer automatischen Zuordnung von Sicherheits-Integritätsleveln (SIL) als auch EAST-ADL in [96] zu finden.

3.2.1.5 Projektergebnisse aus MAENAD

Im Deliverable [81] von MAENAD wird ein erster Bezug zwischen den einzelnen Parts der ISO 26262 zu den Ebenen von EAST-ADL hergestellt. Dies steht insbesondere im Kontext von FEV. So werden Inhalte von Part 3 der ISO 26262 dem „Analysis Level Modeling“ und „Vehicle Level Modeling“ zugeordnet. Part 4 steht in Bezug zum „Design Level Modeling“ und Part 5 sowie Part 6 zum „Implementation Level Modeling“. Als orthogonale Angelegenheiten und damit anwendbar auf alle Modellierungsebenen werden die Inhalte aus Part 2, Part 8 und Part 9 angesehen. Diese Zuordnung wird durch das Aufgreifen einzelner Anforderungen aus der ISO 26262 und daraus abgeleiteten Empfehlungen unterstützt. Im Zusammenhang mit der Produktentwicklung auf Hardwareebene stehen insbesondere die Hardware-Sicherheitsanforderungen im Fokus. Bezogen auf die Hardware-Sicherheitsevaluationen werden exemplarisch Anforderungen im Kontext der Zielwerte für PMHF und die FRC-Methode aufgegriffen.

3.2.1.6 Projektergebnisse aus SAFE

Im Rahmen des SAFE Projekt-Deliverables [97, S.13] konnten unterschiedliche firmenspezifische Sichten auf die Durchführung von Sicherheitsanalysen im Kontext von Hardware zusammengeführt und im Rahmen von Entwicklungsphasen eingeordnet werden. Als Resultat ergab sich ein globaler Ansatz, welcher sich in zwei Alternativen unterteilen lässt, siehe Abb. 3.9. Während Alternative 2 den meist angewandten Ansatz repräsentiert,

bei dem eine FMEDA auf *Hardware-Bauteil*-Ebene durchgeführt wird, zeichnet sich Alternative 1 durch die Sonderheit aus, dass eine FMEDA bereits auf *Hardware-Komponenten*-Ebene angewandt wird.

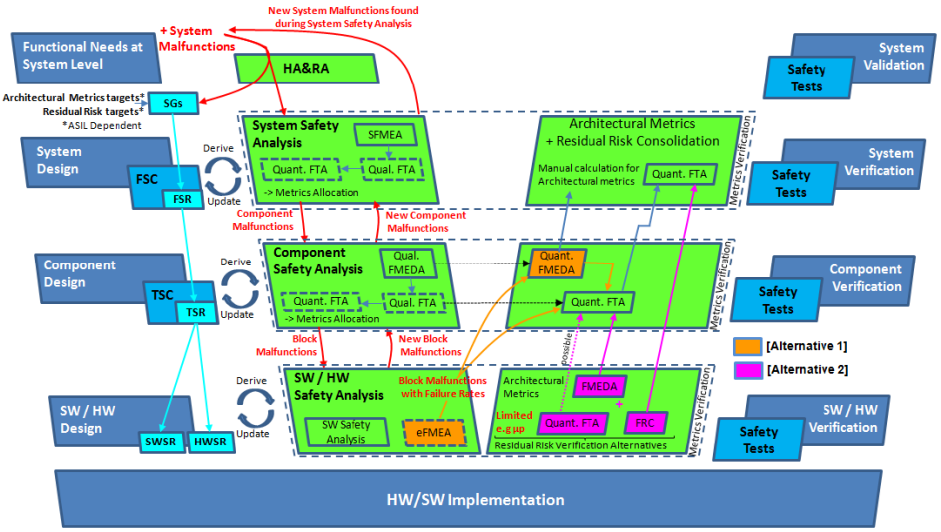


Abb. 3.9: Vorschlag für die Einordnung von Sicherheitsanalysen in [97, S.13]

3.2.1.7 Sicherheitsevaluationen für komplexe Hardware-Bauteile

Im Artikel [98] wird die Evaluation von Hardware im Kontext von Mikroprozessoren und Mikrocontrollern erwähnt. Dieser gibt einen Überblick über die Auswirkung von funktionalen Sicherheitsanforderungen für das Design von Mikroprozessoren. Im Beitrag [99] wird auf zusätzliche Fragestellungen von FPGAs in Bezug auf hohe Zuverlässigkeit und Sicherheitsintegrität eingegangen. Hierzu wird ein Vergleich zu uCs gezogen und exemplarisch quantitative Werte für Ausfallraten im Kontext von Strahlungsrisiken gegeben.

Das Unternehmen Yogitech Spa[50] beschäftigt sich insbesondere mit der Untersuchung von funktionaler Sicherheit auf Ebene von System-on-Chips (SoCs). Hierzu wurde die Methode „fRMethodology“ entwickelt, bei der es sich um einen white-box Ansatz handelt [100]. Diese teilt sich grundlegend in drei Phasen auf: Zunächst wird der integrierte Schaltkreis in sensitive Zonen auf Ebene von beispielsweise Registern aufgeteilt. In der zweiten Phase wird eine FMEDA basierend auf den vorliegenden Informationen und optional auf einem Profil der Arbeitsbelastung vorbereitet. Diese umfasst zusätzlich eine

[50] http://www.yogitech.com/ (Zugriff am 15.10.2014)

Berechnung der Ausfallraten für die sensitiven Zonen innerhalb des integrierten Schaltkreises. In der abschließenden Phase wird die erstellte FMEDA-Datenbank validiert. Dies erfolgt durch eine Mischung von Fehlerinjektion und -simulation. Ein Überblick über die Methode zur Durchführung der FMEA einschließlich Validierung für SoCs ist in [101] dargestellt. Weitere ergänzende Beiträge im Kontext von Sicherheitsevaluationen sowie Sicherheitsmechanismen sind unter anderem in [102], [103] zu finden. Angemerkt sei, dass nach [104] Yogitech der federführende Autor für den Annex A „ISO 26262 and microcontrollers“ aus ISO 26262 Part 10 ist.

In [105] wird auf eine Vorgehensweise zur Bewertung von Mikroprozessoren im Kontext der Produktentwicklung auf Hardwareebene nach ISO 26262 eingegangen. Neben einem dahinterstehenden Prozess wird zudem auf eine Ausfallraten-Abschätzung im Kontext von Mikroprozessoren und die Problematik der PMHF bei einer zunehmenden Designgröße eingegangen.

In [106] wird auf Zuverlässigkeitsmodelle auf Ebene von Hardware Description Language (HDL)-Beschreibungen eingegangen. Angemerkt wird, dass im Zuge der IEC 61508 hinsichtlich der Erreichung eines spezifischen Safety Integrity Level (SIL) nicht die Zuverlässigkeit von verwendeten HDL-Quellcodes berücksichtigt wird. Daher wird ein Konzept vorgestellt, welches eine Abschätzung und Berechnung von Zuverlässigkeitswerten für HDL-Designs im Kontext von konventionellen Software-Zuverlässigkeitsmodellen ermöglicht. Hinsichtlich der Fehlerintensität wird für FPGAs nach [106, F.1] eine Trennung in $\lambda_{FPGA} = \lambda_{HW} + \lambda_{HDL}$ vorgenommen. Hierbei repräsentiert λ_{HW} die Ausfallrate für den FPGA an sich. λ_{HDL} ergibt sich aus der Funktionalität, welche mit Hilfe der HDL beschrieben wird. Informationen zum Design von sicherheitsbezogenen ASICs und SoCs unter Berücksichtigung von IEC 61508 sind unter [107] und [108] zu finden.

Im Artikel [109] wird eine komponentenbasierte Sicherheitsanalyse für FPGAs vorgestellt. Hierzu werden die drei Perspektiven der Sicherheit, des Architekturdesigns und der Netzliste betrachtet. Um die Sicherheitsanalysen zu vervollständigen, wird eine Top-Down Analyse auf Komponentenebene betrachtet und als Bottom-Up Analyse die Fault Propagation and Transformation Calculus (FPTC) eingeführt, um Fehlercharakteristiken für die einzelnen *Hardware-Komponenten* zu identifizieren.

3.2.1.8 Weitere Ansätze für Sicherheitsevaluationen im Rahmen von ISO 26262

Der Beitrag [110], welcher vor der Veröffentlichung der ISO 26262 publiziert wurde, konzentriert sich vordergründig auf den Prozess der Durchführung von Hardware-Architekturmetriken. Allerdings gibt es weder eine semi-formale oder formale Methode noch eine prototypische Implementierung. Die Sicherheitsanalyse eines vorgeschlagenen Brake-by-Wire-Systems aus der Perspektive von funktionaler Sicherheit nach ISO 26262 wird im Artikel [111] beschrieben. Im Zuge von elektrischen Fahrzeugen wird in [112]

ein Ansatz vorgestellt, welcher die Entwicklung von Architekturalternativen auf Systemebene in den Kontext von ISO 26262 sowie zufälligen Ausfällen der Hardware setzt.

QuantUM ist ein UML-basierter Ansatz und beschreibt die Unterstützung des Designprozesses für insbesondere automobile sicherheitsbezogene Systeme, durch Verwendung von semi-formalen und formalen Methoden. In [113] wird aufgezeigt, dass der QuantUM-Ansatz im Kontext von Part 5 der ISO 26262 nicht für die Beschreibung von Hardwarearchitekturen anwendbar ist, jedoch das QuantUM-Tool die Durchführung der Berechnungen von probabilistischen Metriken unterstützen kann. Hinsichtlich quantitativen Sicherheitsanalysen wird im Beitrag [114] ein Bezug zwischen UML-Modellen und QuantUM hergestellt.

3.2.1.9 Vergleich von Automobil- und Medizintechnik im Kontext funktionaler Sicherheit

Im Artikel [115] wird der Standard funktionale Sicherheit für Straßenfahrzeuge ISO 26262 den Standards aus der Medizintechnik-Domäne wie beispielsweise ISO 13485 gegenübergestellt. Angemerkt wird, dass auch diese sich aus einer Vielzahl an Aktivitäten zusammensetzen. Zudem werden Lebenszyklus-Anforderungen in den Standards beschrieben. Hintergrund ist, dass bedingt durch den Entwicklungsprozess für ein Produkt ein Schaden an Personen verursacht werden kann. Dies ist insbesondere auf den Trend softwareintensiver Produkte zurückzuführen. Eine Gegenüberstellung in Bezug auf die unterstützenden Prozesse ausgehend von ISO 26262 ist in [115, Tab.4] dargestellt. Als Fazit wird geschildert, dass zwar beide Domänen einer gemeinsamen Philosophie nachgehen, jedoch ein domänenübergreifender Austausch nur schwer durchführbar ist, bedingt durch unterschiedliches Vokabular und Interpretationen der Standards. Jedoch könnten durch standardisierte Ansätze große Synergien hervorgebracht werden. Daher wäre ein Austausch von Wissen bezüglich Erfahrungen und Methoden über sicherheitskritische Domänen hinweg sehr förderlich.

3.2.2 Stand der Technik

Da die Vorgehensweisen und Anwendung der Sicherheitsevaluationen meist firmenspezifisch sind, vgl. [97, S.13], und teilweise gewachsene Individuallösungen darstellen, vgl. Beitrag [92] und [93] mit der Forderung nach einer kombinierten Lösung, wird ersichtlich, dass aktuell wenige Werkzeuge am Markt sind. Im Folgenden werden die zur Zeit verfügbaren Werkzeuge im Kontext von Hardware-Sicherheitsevaluationen dargestellt.

medini^{TM} analyze[51] von der ikv++ technologies ag bietet eine konsistente Anwendung von Sicherheitsanalysen unter Verwendung von SysML-Architekturmodellen

[51] http://www.ikv.de/index.php/de/products/functional-safety (Zugriff am 15.10.2014)

[116]. Im Informationsblatt [159] zu medini^{TM} analyze von Mai 2014 wird unter anderem dargestellt, dass FTA, FMEA, probabilistische Analysen sowie die Hardware-Architekturmetriken unterstützt werden. Hierbei ist auch eine Generierung von Arbeitsprodukten aufgeführt. Eine Schnittstelle zu Tools wie beispielsweise MATLAB Simulink und Stateflow wird bereitgestellt. Für die initiale Definition des Items und eine erste Systemmodellierung kann ein grafischer SysML-Editor genutzt werden. Die Anreicherung mit Fehlerinformationen umfasst unter anderem Ausfallarten und Ausfallraten für Elemente auf Systemebene. Interessant sind die Ausfallraten-Bestimmung aus Katalogen und Handbüchern sowie Bill of materials (BOM)-Import. Als Sicherheitsanalysen werden FTA, FMEA, FMEDA und die Hardware-Architekturmetriken aufgeführt. Für die FTA wird ein grafischer Editor zur Verfügung gestellt, welcher eine qualitative und quantitative FTA unterstützt. In Bezug auf Variantenmanagement wird eine Kombination von medini^{TM} analyze und pure::variants von pure-systems GmbH im Beitrag [117, S.2] aufgegriffen.

Der RAM Commander[52] von der ALD Reliability Engineering Ltd. stellt einen Werkzeugsatz für RAMS (Reliability, Availability, Maintainability and Safety) bereit. Dieser unterstützt unterschiedliche Sicherheitsanalysen wie beispielsweise FMECA und FTA sowie Zuverlässigkeits-Blockdiagramme. Der ALD MTBF Calculator[53] ermöglicht die Bestimmung von MTBF und Ausfallraten durch Auswahl einer Komponentenfamilie sowie des Typs, der Zuverlässigkeitsvorhersage-Methode, der Umgebung sowie Temperatur und der Eingabe von Komponentenparametern. Bei der Komponentenfamilie wird grundlegend in elektronische und mechanische Komponenten unterschieden. Bei Ersterem werden Zuverlässigkeitsvorhersage-Methoden beispielsweise basierend auf MIL-217E/F, FIDES 2009, HDBK-217Plus sowie Siemens SN 29500 bereitgestellt. Bei der mechanischen Komponentenfamilie wird beispielsweise die NPRD-95 aufgegriffen.

Der Software-Assistent SISTEMA[54] vom Institut für Arbeitsschutz (IFA) der Deutschen Gesetzlichen Unfallversicherung (DGUV) e.V. bietet Unterstützung bei der Sicherheitsbewertung von sicherheitsbezogenen Maschinensteuerungen nach DIN 13849 [DIN13849]. Sicherheitsbezogene Steuerungsteile können auf Basis von vorgesehenen Architekturen abgebildet werden. Anschließend lassen sich Zuverlässigkeitswerte im Rahmen der Sicherheitsevaluationen nach DIN 13849 berechnen.

Ein Auszug von kommerziellen Werkzeugen, welche eine FTA bzw. FMEA mit unterschiedlichen Charakteristiken unterstützen, ist im Folgenden aufgeführt: IQ-FMEA von APIS Informationstechnologien GmbH[55], medini^{TM} analyze von ikv++ technologies ag[56], SCIO^{TM} von PLATO AG[57], Reliability Workbench incorporating FaultTree+ von Isograph

[52] http://aldservice.com/en/reliability-products/rams-software.html (Zugriff am 15.10.2014)
[53] http://www.aldservice.com/en/reliability-software/free-mtbf-calculator.html (Zugriff am 15.10.2014)
[54] http://www.dguv.de/dguv/ifa/Praxishilfen/Software/SISTEMA/index.jsp (Zugriff am 15.10.2014)
[55] http://www.apis.de/ (Zugriff am 15.10.2014)
[56] http://www.ikv.de/ (Zugriff am 15.10.2014)
[57] http://www.plato.de/home-en.html (Zugriff am 15.10.2014)

Ltd[58], SafetyOffice X2 (SOX2) von Engineers Consulting GmbH[59], ITEM ToolKit von ITEM Software[60].

3.2.3 Abgrenzung

Die vorgestellten Ansätze und Werkzeuge bieten keine vollständige bzw. durchgängige Unterstützung der innerhalb von ISO 26262 geforderten Sicherheitsevaluationen. Dies schränkt die Nutzung im Kontext unterschiedlicher Entwicklungsvorgehen ein, da bei den Evaluationen aus Alternativen ausgewählt werden kann. Weiterhin stehen die Werkzeuge für sich und erfordern immer eine Anbindung an weitere Werkzeuge, sei es hinsichtlich dem Anforderungsmanagement, der strukturellen Modellierung, der Anreicherung um Fehlerinformationen oder der Durchführung der Sicherheitsevaluationen. Die dadurch notwendigen Werkzeugketten führen gegebenenfalls zu möglichen Inkonsistenzen bedingt durch die Werkzeugwechsel oder sogar zu Werkzeugbrüchen. Dies kann insbesondere auch bei der Verwendung von Bibliotheksansätzen eine mögliche Herausforderung in Bezug auf die Synchronisation sein.

Gleichzeitig muss sowohl eine Rückverfolgbarkeit als auch Nachvollziehbarkeit von Sicherheitsanforderungen bis hin zu den Verfeinerungen im Zuge des *Hardware-Architekturdesigns* sowie des *detaillierten Hardwaredesigns* vorliegen, um die einzelnen Entwicklungsphasen durchgängig zu unterstützen. Hierbei muss insbesondere an weitverbreitete Architekturbeschreibungssprachen angekoppelt werden, um eine modellbasierte Unterstützung auf unterschiedlichen Abstraktionsebenen zu bieten. Somit können die Sicherheitsevaluationen aufbauend innerhalb solcher Umgebungen im Rahmen der Ankopplung an das EE-Architekturmodell integriert unterstützt werden.

Ein wichtiger Gesichtspunkt in der Automobilindustrie ist die verteilte Entwicklung. Um einen produktiven und zielgerichteten Einsatz zu gewährleisten, muss besonderer Wert auf die Unterstützung von großformatigen Architekturmodellen in einem benutzerfreundlichen Frontend und eine Zusammenführung in ein stabiles Backend als single-source gelegt werden, um somit eine spätere Integration auf Systemebene unter funktionalen Sicherheitsaspekten zu berücksichtigen und zu ermöglichen. Dies ist weiterhin in Bezug auf ein Änderungsmanagement und iterative Überarbeitungen des Systems essentiell. Auch im Kontext der unterschiedlichen Abstraktionsebenen ist die einheitliche und konsistente Durchführung der Sicherheitsevaluationen in Bezug auf verteilte Entwicklungen relevant.

Nicht zuletzt gilt es die im Zuge der ISO 26262 zu erbringenden Arbeitsprodukte in Form von Nachweisdokumenten zu unterstützen. Hierfür müssen insbesondere Vorlagen bereitgestellt werden, welche an entwicklungsspezifische Gegebenheiten angepasst werden können. Dies ist in den dargestellten Lösungen und Werkzeugen im Kontext der

[58] http://www.isograph.com/ (Zugriff am 15.10.2014)
[59] http://www.enco-software.com/ (Zugriff am 15.10.2014)
[60] http://www.itemsoftware.com/ (Zugriff am 15.10.2014)

Hardware-Sicherheitsevaluationen nur sehr eingeschränkt möglich. Insbesondere aufgrund der Werkzeugketten liegen nicht alle relevanten Informationen in einer Umgebung vor. Dies kann hinsichtlich einer Vollständigkeit der Darstellung sowie den unterschiedlichen Revisionsständen von Arbeitsprodukten zu Inkonsistenzen führen.

4 Integrierte Prozessmodellierung und -management

Prozessmodellierung und -management wird nicht nur insgesamt zunehmend bedeutender, sondern insbesondere für die Produktentwicklung von sicherheitskritischen sowie sicherheitsbezogenen Systemen in der Automobil-, Avionik-, Eisenbahn-, und Medizinproduktdomäne. Dies ist bedingt durch zunehmende Systemkomplexität, lange Zulieferketten und einer Vielzahl an Beteiligten. Neben der eigentlichen Absicherung von hoher Produktqualität gilt es zusätzlich sowohl gesetzliche Vorschriften als auch nationale sowie internationale Standards einzuhalten. Einer der ausschlaggebendsten Gründe für die Einführung eines Prozessmanagements ist oft die Nachweispflicht in Form von entwicklungsbegleitenden Dokumentationen, welche unter anderem Begründungen für Entscheidungen, getroffene Maßnahmen und Analyseergebnisse festhalten.

Standards definieren häufig über Anforderungen, welche Arbeitsprodukte gefordert und damit umzusetzen sind. Die dahinterstehenden Prozesse sind aus den entsprechenden Quellen zu übernehmen oder abzuleiten. Eine Herausforderung ist deren Abbildung, gegebenenfalls Einbettung bzw. Integration in bereits existierende domänen- und firmenspezifische Prozesse. Über ein Prozessmodell kann die spätere Umsetzung und Einhaltung der Prozesse maßgeblich unterstützt und erleichtert werden. Zudem können Schnittstellen zwischen Auftragnehmern und Auftraggebern klar definiert sowie kommuniziert werden, um hierdurch Prozesse mit vielen beteiligten Firmen und Mitarbeitern zu beherrschen. Weiterhin kann der Bezug zu anderen möglicherweise vorhandenen Zertifizierungen, wie beispielsweise nach ISO 9001 [ISO9001], hergestellt werden.

Mit Fokus auf funktionale Sicherheit in der Automobilentwicklung ist der umfangreiche Standard ISO 26262 [ISO26262] normativ und wird bereits auf zahlreiche Industrieprojekte angewandt. Durch die von Kunden gewünschten Ausstattungsmerkmale, zum Beispiel im Bereich der Fahrerassistenzsysteme, kann hinsichtlich expandierender Systemkomplexität und Auswirkungen von funktionalen Sicherheitsaspekten tiefergehend aufgezeigt werden, inwiefern die Bedeutung eines gemeinsamen Prozessverständnisses zunimmt. Um diese komplexe Thematik überschaubar, handhabbar und zugleich kontrollierbar zu machen, ist eine domänenspezifische Integration von Prozessmodellierung und -management eine der geeignetsten Herangehensweisen. Damit können die geforderten Prozesse zudem als positiver Seiteneffekt den fachfremden Beteiligten ohne Hintergrundwissen zu funktionaler Sicherheit einfach zugänglich und aufbereitet zur Verfügung gestellt werden.

Im Folgenden wird hierzu eine Metamodell-Erweiterung für modellbasierte Architekturbeschreibungssprachen hinsichtlich Prozessbeschreibungen, Organisationsstrukturen und Ressourcenzuweisungen bereitgestellt. Diese basiert auf der etablierten Notation BPMN gemäß ISO/IEC 19510 [ISO19510] und wurde in Bezug auf domänenspezifische Gegebenheiten erweitert. Die Integration in eine Entwicklungsumgebung für großformatige EEAs ermöglicht eine grafische Prozessmodellierung. Darüber hinaus können die Prozesselemente mit den Elementen des EEA-Datenmodells verknüpft werden. Zudem kann über eingebrachte Ressourcen eine Zuweisung der Verantwortlichkeiten gegenüber beschriebenen Prozessinhalten erfolgen. Zur Unterstützung eines Prozessmanagements wird basierend auf dem Modell eine Möglichkeit der Prozessanalyse und -planung geboten. Als wissenschaftliches Fallbeispiel erfolgte die Anwendung auf die innerhalb der ISO 26262 beschriebenen Prozesse für funktionale Sicherheit. Teile dieser Arbeit konnten bereits im Rahmen eines Journals als eigener Beitrag [Adler2014a] veröffentlicht werden.

4.1 Konzept einer integrierten Prozessmodellierung und -management

Während der Konzept- und Entwicklungsphase von komplexen Systemen werden unterschiedliche Abstraktionsebenen und Perspektiven für die Beschreibung einer EEA benötigt, vgl. [21], [137], [160]. Innerhalb von modellbasierten Architekturbeschreibungssprachen nach Kapitel 2.3 können hierzu Spezifikationen, einschließlich der Anforderungen im Sinne eines Lasten- und Pflichtenheftes, hinterlegt werden, siehe Abb. 4.1. Die Spezifikation des Gesamtsystems dient der Erstellung einer logischen Architekturbeschreibung. Aufbauend auf dieser kann das System in eine Software- und Hardwarearchitektur verfeinert werden. Dies kann um die Beschreibung der geometrischen Eigenschaften ergänzt werden. Eine Nachverfolgbarkeit über unterschiedliche Abstraktionsebenen hinweg kann über Verknüpfungen der Architekturelemente sichergestellt werden. Zusammen mit der Überprüfung auf Konsistenz wird hierdurch ein solider Ausgangspunkt für die Entwicklung von sicherheitsbezogenen Systemen bereitgestellt.

Die im Stand der Wissenschaft nach Kapitel 3.1 vorgestellten Konzepte stellen wesentliche Konstrukte zur Verfügung, um sowohl einer Prozessbeschreibung durch Modellierung als auch einem Management zu genügen. Für die Modellierung von Prozessen kann auf eine Vielzahl an Werkzeugumgebungen zurückgegriffen werden. Diese erfüllen erste Anforderungen im Kontext der Beschreibung von sicherheitsbezogenen Systementwicklungsprozessen. Jedoch stehen diese für sich selbst und lediglich über eine übergreifende Kombination von Werkzeugen können entwicklungsspezifische Prozessbeschreibungen in Bezug zu den tatsächlichen Aktivitäten und Arbeitsprodukten gebracht werden. Daher ist eine Integration der domänenspezifischen Anforderungen im Rahmen des Entwicklungsprozesses nicht gegeben. Die Verknüpfung mit den während des Prozesses genutzten Entwicklungsumgebungen birgt zudem, bedingt durch Werkzeugwechsel und -brüche, mögliche Inkonsistenzen. Somit ist die Verwendung von BPMN oder vergleich-

baren Notationen für die Modellierung von Prozessen äußerst angemessen, jedoch wird kein direkter Bezug zu Architekturbeschreibungssprachen zur Verfügung gestellt.

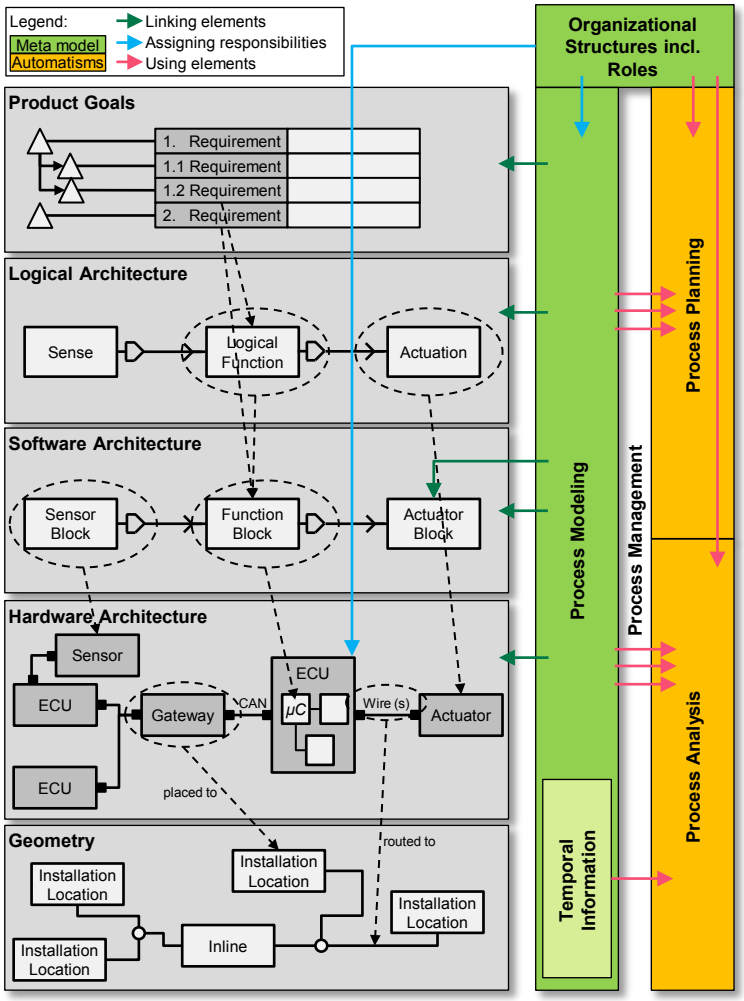


Abb. 4.1: Konzeptüberblick für integrierte Prozessmodellierung und -management innerhalb einer Beschreibungssprache für EEAs

Daher zielt eine Integration in eine domänenspezifische Entwicklungsumgebung darauf ab, eine Erweiterung hinsichtlich Prozessmodellierung und -management in etablierten Werkzeugen bereitzustellen. Gleichermaßen sind bewährte Werkzeugcharakteristiken, welche durch die Entwicklungsumgebung zur Verfügung gestellt werden, unangetastet

beizubehalten. Auf diesen kann gegebenenfalls unter dem Gesichtspunkt einer Konzeptverschmelzung aufgesetzt werden.

Das Konzept und die dahinterstehende Methodik einer integrierten Prozessmodellierung und eines Prozessmanagements setzt sich aus zwei unterschiedlichen Bereichen zusammen. Zum einen aus der Erweiterung sowie Anpassung von Metamodell-Konstrukten, welche der eigentlichen Prozessbeschreibung und Hinterlegung von Organisationsstrukturen innerhalb von Architekturbeschreibungssprachen dienen. Zum anderen aus der Bereitstellung eines Prozessmanagements im Kontext der Unterstützung von ausgewählten Prozessanalysen und -planungen.

Der Ansatz sieht hierzu vor, eine zusätzliche Ebene für die Prozessbeschreibungen aufzuspannen, welche orthogonal zu den beispielhaft dargestellten Abstraktionsebenen in Abb. 4.1 ist. Hierzu soll ein Metamodell-Vorschlag zur Verfügung gestellt, um die entsprechenden Erweiterungen einzubringen. Dieser soll auf der etablierten Notation BPMN basieren, da innerhalb des Standards ISO/IEC 19510 grundlegende Konstrukte beschrieben sind und erforderliche Erweiterungsmöglichkeiten geboten werden, vgl. Kapitel 2.5.3. Um einen Bezug mit den Elementen eines EEA-Datenmodells herzustellen, können Prozesselemente entsprechend mit diesen verknüpft werden. Eine darauf aufsetzende Beschreibung von Organisationsstrukturen ermöglicht die Zuweisung von Zuständigkeiten sowie Verantwortlichkeiten in Richtung der Prozess- und EEA-Datenelemente. Das Prozessmanagement repräsentiert keine zusätzliche Ebene, sondern greift unter anderem auf die Prozessmodelle und Beschreibungen der Organisationsstrukturen zurück. Es wird in die Teilbereiche der Prozessanalyse und -planung aufgeteilt.

4.1.1 Prozessmodellierung

Eine integrierte Prozessmodellierung soll unterschiedlichen Ansprüchen genügen. Zum einen sollen allseits bekannte Geschäftsprozessmodelle erstellt und grafisch dargestellt, zum anderen firmenspezifische Vorgehensmodelle hinterlegt werden können. Dies ermöglicht sowohl eine Erhebung von aktuell vorliegenden Prozessen nach dem Prinzip der Aufnahme des Ist-Zustands als auch die Erarbeitung von umzugestaltenden oder neu einzuführenden Prozessen nach einem gewünschten Soll-Zustand. Des Weiteren sollen Prozesse, welche innerhalb von umzusetzenden Standards beschrieben oder verankert sind, eingebracht oder nach einer erarbeiteten Ableitung modelliert werden können. Im Rahmen dieser Arbeit steht dies im Hauptfokus von ISO 26262 und den damit verbundenen Aktivitäten sowie Arbeitsprodukten. Etablierte automobile Entwicklungsprozesse können zudem der Prozessbeschreibung der ISO 26262 überlagert werden. Darauf aufbauend sollen erste Schritte in Richtung eines prozessorientierten und prozessgetriebenen Entwicklungsvorgehens berücksichtigt werden.

Die Prozessmodellierung soll eine vertikale Ebene zu den in Kapitel 2.3 beschriebenen horizontalen Abstraktionsebenen aufspannen. Dies ermöglicht eine Verknüpfung der

Prozesselemente mit den EEA-Datenelementen bei gleichzeitigen ebenenübergreifenden Verknüpfungen innerhalb des EEA-Datenmodells. Somit kann ein Bezug von Prozessen zu den produktiven Entwicklungstätigkeiten hergestellt werden. Weitergehend können hierdurch Prozesse hinsichtlich des Produktlebenszyklus bis zu den zur Verfügung stehenden Detaillierungsgraden und Modellierungsmöglichkeiten der modellbasierten Entwicklungsumgebung bedient werden. Für die Bereitstellung einer Prozessmodellierung sollen anerkannte Vorgehensweisen und Hilfsmittel einbezogen werden. Dies ermöglicht unter anderem Nutzern aus unterschiedlichen Fachbereichen einen bewährten Blick auf die Thematik.

Der Beitrag erhebt somit den Anspruch an einen ersten integrierten Ansatz in einer domänenspezifischen modellbasierten Werkzeugumgebung. Hierzu gilt es, die etablierten und relevanten BPMN-Konstrukte für die Prozessmodellierung zu identifizieren, diese maßgeschneidert zu erweitern und übergreifend mit den vorhandenen Konstrukten von Architekturbeschreibungssprachen zu harmonisieren. Hierdurch kann eine Möglichkeit der Integration in eine vorhandene EEA-Entwicklungsumgebung geboten werden, um somit eine Anwendung im Kontext von Architekturmodellen bereitzustellen. Dies erweitert die existierenden Abstraktionsebenen innerhalb der Architekturbeschreibungssprache und unterstützt die Interaktion sowie nahtlose Architekturentwicklung.

4.1.2 Organisationsstrukturen und Prozessressourcen

Für die Beschreibung von Organisationsstrukturen im Kontext der modellierten Prozesse ist eine Metamodell-Erweiterung vorzuschlagen. Sowohl unterschiedliche interne und externe Organisationsstrukturen und Hierarchien als auch einzelne Rollen sollen mit Schwerpunkt auf ihrer Interaktion mit dem Prozess modelliert werden können. Daher müssen Konstrukte für die Zuordnung zu spezifischen Prozesselementen bereitgestellt werden. Eine Zuordnung zu Elementen aus dem EEA-Datenmodell in Bezug auf Verantwortlichkeiten für bestimmte Teilbereiche steht ebenfalls im Fokus. Dies ist im Zuge einer internen und externen Kommunikation dienlich. Zudem werden eindeutige Schnittstellen definiert und Ansprechpartner dokumentiert. Um Rollen sowie deren Verantwortlichkeiten und Zuständigkeiten im Kontext des Prozesses beschreiben zu können, dürfen nicht nur für die Entwicklungsumgebung vorgesehene und eingetragene Benutzer betrachtet werden, sondern alle Parteien mit einer Prozessbeteiligung bis herunter auf einzelne Mitarbeiter müssen berücksichtigt werden.

4.1.3 Prozessmanagement

Basierend auf dem Prozessmodell, den Organisationsstrukturen und deren Verknüpfungen zum EEA-Datenmodell können unterschiedliche Methoden für ein Prozessmanagement zur Verfügung gestellt werden, welche sich in Prozessanalyse und Prozessplanung aufteilen. Beide Teilbereiche basieren auf den modellierten Prozessbeschrei-

bungen und werden auf diese angewandt sowie mit den hinterlegten Rollen und Verantwortlichkeiten in Bezug gesetzt. Somit wird keine Metamodell-Erweiterung benötigt, jedoch erfordert dies zum einen ein Rahmenwerk, auf dem eine Ausführung aufgesetzt werden kann, zum anderen Konstrukte, welche eine Gruppierung unabhängig und wiederum orthogonal zu den Prozessbeschreibungen ermöglichen. Die Verwendung einer solchen Schnittstelle unterstützt die Erstellung einer Vielzahl unterschiedlicher Automatismen und Metriken unter dem Gesichtspunkt einer Prozessanalyse und -planung. Für komplexe Prozesse und involvierte Personen, oft nicht auf einzelne Unternehmen beschränkt, müssen unterschiedliche Perspektiven bereitgestellt werden, um dem gesamten Fokus sowie den Teilinteressen gerecht zu werden.

4.2 Erweiterte BPMN-Konstrukte für integrierte EEA-Entwicklungsumgebungen

Für die Beschreibung von Prozessen in einer integrierten modellbasierten EEA-Entwicklungsumgebung wird das Hauptaugenmerk auf die Verwendung der grundlegenden BPMN-Konstrukte gelegt, vgl. unter anderem Kapitel 3.1, welche an die spezifischen Gegebenheiten angepasst und zusätzlich erweitert werden. Somit können die benötigten Modellierungsartefakte bereitgestellt und die erforderliche Semantik zur Verfügung gestellt werden. Der Standard ISO/IEC 19510 beschreibt insgesamt fünf Basiskategorien für Elemente wie folgt: *Flow Objects*, *Data*, *Connecting Objects*, *Swimlanes* und *Artifacts* [ISO19510, 7.3]. Für die Modellierung im Kontext dieses Ansatzes sind die Elemente *Flow Objects*, *Data* und *Connecting Objects* relevant.

Im Folgenden werden die Konstrukte zur Modellierung von Prozessen sowie firmenspezifischen Strukturen und deren Verantwortlichkeiten in Bezug auf die Prozesse vorgestellt. Die vorgeschlagenen UML-Klassendiagramme, als Grundlage für die Modellierung von Prozessartefakten, sind angelehnt an die Metamodell-Beschreibung von BPMN, wurden jedoch mit zusätzlichen Klassen und Attributen gemäß der *BPMN Extensibility* [ISO19510, 7.7] angereichert, um somit spezifische Bedürfnisse zu erfüllen. Ausschlaggebend ist, insbesondere die BPMN-Kernelemente aus [ISO19510, 8.4] zu berücksichtigen. Die erarbeiteten Konstrukte werden in den vier Klassendiagrammen *FlowElements*, *IntegratedProcessModeling*, *OrganizationalStructure* und *ProcessResource* beschrieben.

4.2.1 Konstrukte zur Beschreibung von Prozessmodellen

Für die Beschreibung von Prozessen bezogen auf die Prozess-Flusselemente wird eine Formalisierung im Rahmen eines Metamodells benötigt. Dies wird in den folgenden Abschnitten anhand von Klassendiagrammen sowie der Beschreibung der einzelnen Klassen vorgestellt.

4.2.1.1 Klassendiagramm FlowElements

Eine Superklasse für alle Elemente, welche in einem Prozessfluss enthalten sind, wird durch die Klasse *FlowElement* beschrieben, wie in Abb. 4.2 ersichtlich. Dieses Konzept wird basierend auf [ISO19510, 8.4.7] erweitert, indem unter anderem die Klassen *AbstractPort* und *Connection* eingeführt werden. Somit kann das BPMN-Konstrukt aus *FlowNode* und *SequenceFlow* ersetzt und sowohl die *Activity* als auch das *Event* dem *FlowElement* direkt zugeordnet werden.

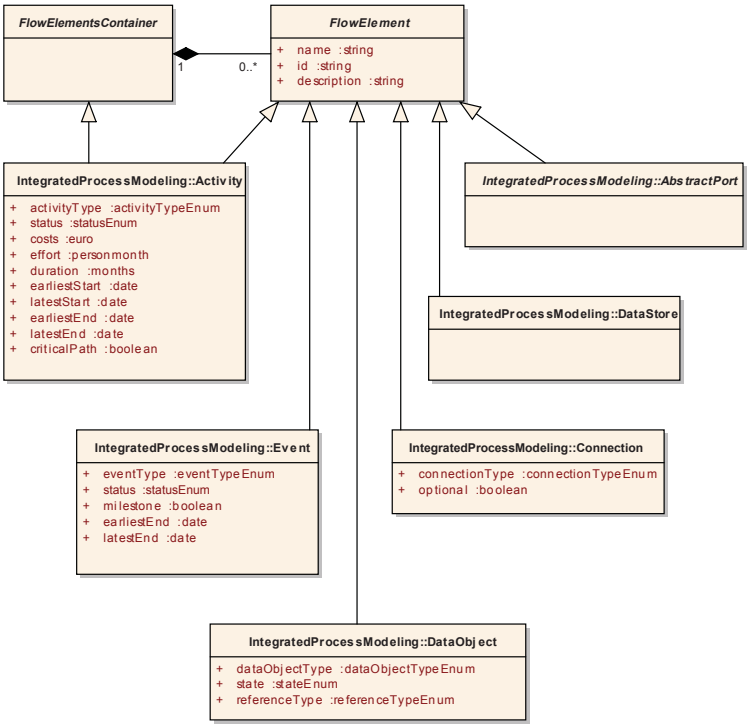


Abb. 4.2: Klassendiagramm *FlowElements* - vorgeschlagenes Metamodell nach [Adler2014a]

FlowElement stellt die öffentlichen Attribute *name*, *id* und *description* zur Verfügung. Der Datentyp dieser Attribute ist vom Typ *string*, um somit einen Namen sowie eine eindeutige ID und gegebenenfalls eine Beschreibung hinterlegen zu können. Im Kontext eines Prozessflusses erben die folgenden Klassen sämtliche öffentliche Attribute der Superklasse *FlowElement*: *Activity*, *Event*, *DataObject*, *DataStore*, *AbstractPort* und *Connection*. Die aufgelisteten Flusselemente werden zur späteren Modellierung der Prozesse benötigt und spiegeln die Prozessartefakte in den Diagrammen wider.

Die abstrakte Klasse *FlowElementsContainer* basiert auf [ISO19510, 8.4.8] und repräsentiert einen Container. Dieser stellt die Bündelung von mehreren Flusselementen über die Multiplizität „0..*“ bereit und besitzt als eingeführte direkte Unterklasse die Klasse *Activity*. Durch die Definition von Untermengen im Kontext von Aktivitäten kann somit durch die Klasse *FlowElementsContainer* eine Hierarchisierung vorgenommen werden. Dadurch kann eine Menge von Flusselementen selbst wiederum eine Aktivität in Form von Prozess und Unterprozess repräsentieren.

4.2.1.2 Klassendiagramm IntegratedProcessModeling

Das in Abb. 4.3 vorgeschlagene Klassendiagramm für die integrierte Prozessmodellierung stellt die Metamodell-Konstrukte zur späteren Beschreibung von Prozessen und damit der Arbeitsabläufe bereit. Der Fokus liegt unter anderem auf der Modellierung von in Standards festgehaltenen Prozessen. Für die eindeutige Anwendung soll eine einfache Modellierung der Prozesse ermöglicht werden, ohne beispielsweise das Einbringen von logischen Bedingungen im Rahmen des Prozessflusses.

Nach [ISO19510, 7.3] sind *Flow Qbjects* die grundlegenden grafischen Elemente, um das Verhalten eines Prozesses zu beschreiben. Bei den drei gelisteten Flussobjekten handelt es sich um *Events*, *Activities* und *Gateways*. Im Zuge dieses Ansatzes werden die Aktivitäten und Ereignisse aufgegriffen, da diese die entscheidensten Elemente in der Beschreibung eines Prozesses widerspiegeln. Der Ansatz fokussiert sich somit nicht direkt auf die Verwendung von Gateways, welche jedoch in vereinfachter Form mit dem Konstrukt der eingehenden und ausgehenden Ports beschrieben werden können.

Klasse *Activity*: Aktivitäten beschreiben im Allgemeinen produktive Arbeit. Diese werden nach [ISO19510, 10.3] innerhalb eines Prozesses durchgeführt. Eine Aktivität kann wiederum weitere Elemente eines Prozessflusses beinhalten und repräsentiert daher in Anlehnung an BPMN auf der einen Seite einen Gesamtprozess oder Unterprozess sowie auf der anderen Seite einen elementaren Arbeitsschritt, der als Aufgabe verstanden wird.

Die Unterscheidung erfolgt über die Enumeration *activityTypeEnum*, welche die entsprechenden zulässigen Literale aufführt und somit nach der Einstufung eine Einordnung in der Hierarchie ermöglicht. Die Beschreibung von Aktivitäten als Prozesse oder Unterprozesse wird von der abstrakten Klasse *FlowElementsContainer* abgeleitet. Die Klasse *FlowElementsContainer* beschreibt in Übereinstimmung mit [ISO19510, 8.4.8] die Klasse zur Kapselung von mehreren spezifischen *FlowElements* wie Aktivitäten, Ereignisse und Verbindungen über eine Komposition. Für einen *Task* kann somit lediglich ein Bezug zu einem übergeordneten *FlowElementsContainer* bestehen, jedoch repräsentiert dieser selbst als elementarer Baustein keinen Container.

Über das Attribut *status* wird der Status einer Aktivität gekennzeichnet, beispielsweise ob diese gestartet oder bereits gelöst ist. Dieser kann durch einen Beteiligten oder Verantwortlichen gesetzt werden. Über den Status der Teilprozesse können die einzelnen

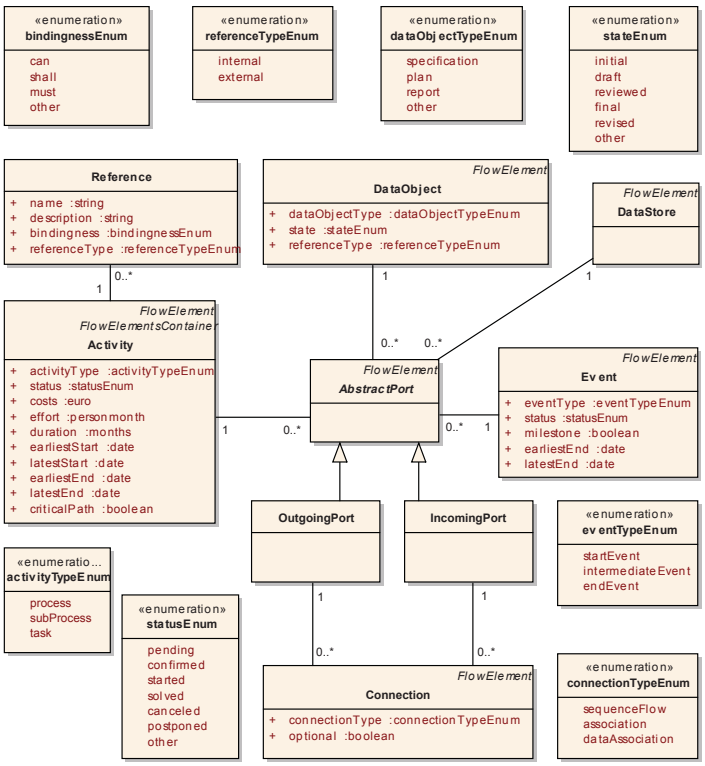


Abb. 4.3: Klassendiagramm *IntegratedProcessModeling* - vorgeschlagenes Metamodell nach [Adler2014a]

Rollen den aktuellen Stand des Prozesses nachvollziehen. Im Kontext der Hierarchisierung schließt eine auf höherer Ebene angesiedelte Einstufung einer Aktivität den Status der untergeordneten Flusselemente gegebenenfalls ein.

Zur Hinterlegung von betriebswirtschaftlichen Informationen werden als Erweiterung die Attribute *costs*, *effort* und *duration* für die Aktivitäten bereitgestellt. Als spezifische Datentypen sind exemplarisch die Einheiten *euro*, *person month* und *months* als eine Erweiterung zu den primitiven Typen eingeführt und werden für die entsprechenden Attribute verwendet. Ferner wird die Aktivität um die Attribute *earliestStart*, *latestStart*, *earliestEnd* und *latestEnd* mit dem Datentyp *date* erweitert, um somit zeitliche Informationen und Einschränkungen aufzugreifen. Deren Attributwerte lassen sich im Modell sowohl manuell eintragen als auch über Berechnungen automatisiert bestimmen. Hinterlegte Werte ermöglichen aufbauend eine Prozessanalyse und -planung. Zusätzlich können Aktivitäten, welche dem kritischen Pfad zugehörig sind, über das boolesche Attribut *criticalPath* spezifiziert werden.

Klasse *Reference*: Die Klasse *Reference* ist mit der Klasse *Activity* assoziiert und repräsentiert allgemeingültig ein Konstrukt für eine Referenz. Dies ist notwendig, um Verweise und Relationen auf beispielsweise Standards aber auch weiterreichende Dokumente mit unterstützenden Informationen darzubieten. Um unterschiedliche Verbindlichkeiten bezogen auf die jeweils verknüpfte Aktivität auszudrücken, wird das Attribut *bindingness* verwendet, welches über den eingeführten Datentyp *bindingnessEnum* unter anderem die Enumerationsliterale *can*, *shall* und *must* zur Verfügung stellt. Bei der Referenz handelt es sich um kein Flusselement, weshalb die Attribute *name* und *description* in der Klasse selbst enthalten sind und nicht über die abstrakte Klasse *FlowElement* vererbt werden. Weiterreichend kann mit Hilfe des Attributs *referenceType* eine Einstufung der Referenzquelle in intern oder extern aus Sicht des betrachteten Prozesses vorgenommen werden. Da für eine einzelne Aktivität mehrere unterschiedliche Referenzen vergeben werden können, wird eine Multiplizität von „0..*“ zugewiesen.

Klasse *Event*: Über die Klasse *Event* wird das Flusselement Ereignis zur Verfügung gestellt. Nach [ISO19510, 10.5] entspricht ein Ereignis „etwas, das sich während des Verlaufs eines Prozesses ereignet“ und „sich auf den Fluss eines Prozesses auswirkt“. Eine Unterscheidung zwischen den drei unterschiedlichen Ereignistypen wird mit Hilfe des Attributs *eventType* über die Verwendung der Enumerationsliterale *startEvent*, *intermediateEvent* und *endEvent* kenntlich gemacht. Diese Möglichkeit der Einordnung ist in Anlehnung an BPMN nach [ISO19510, 10.5] gegeben. Vergleichbar zur Aktivität wird der aktuelle Status des Ereignisses über das Attribut *status* hinterlegt, welches den selben Datentyp zugewiesen hat.

Zudem kann ein Ereignis, wenn es von einer besonderen Bedeutung für den Prozess ist, optional als Meilenstein eingestuft werden. Hierzu wird das dazugehörige Attribut *milestone* vom Typ *boolean* verwendet. Da Ereignisse zeitlich gesehen keine Dauer vorweisen, wird die Erweiterung hinsichtlich zeitlicher Informationen auf die Attribute *earliestEnd* und *latestEnd* zugeschnitten und beschränkt. Hierdurch wird die Beschreibung der frühesten und spätesten Enddaten ermöglicht. Die beabsichtigte Beschränkung wirkt sich auf betriebswirtschaftlich relevante Aspekte, wie Aufwand und Zeitdauer, aus.

Klasse *DataObject*: Während eines Prozessflusses werden Arbeitsprodukte, wie beispielsweise Dokumentationen in Form von Berichten, ausgearbeitet. Einige dienen als Voraussetzung für weitere anstehende Aktivitäten und repräsentieren somit bereitzustellende Eingangsdokumente. Um diese innerhalb der Prozessbeschreibung bereitzustellen, werden die Klassen *DataObject* und *DataStore* gemäß [ISO19510, 10.4.1] eingebracht. Ein Datenobjekt stellt das wesentliche Konstrukt zur Modellierung von Daten innerhalb einer Prozessbeschreibung dar. Der Typ eines jeweiligen Datenobjektes kann über das Attribut *dataObjectType* spezifiziert und zugewiesen werden. Als Enumerationsliterale stehen exemplarisch *specification* oder *report* zur Verfügung. Hinsichtlich anstehender oder abgearbeiteter Arbeitsprodukte kann eine Verfügbarkeit über die Erweiterung durch das Attribut *state* für den aktuellen Stand herausgestellt werden. Hierfür können durch die Verwendung der Enumeration *stateEnum* beispielsweise *initial*, *draft* oder *revised* für die

Version des Datenobjekts im Zuge eines Arbeitsproduktes annotiert werden. Zugleich beschreibt das Attribut *referenceType*, ob das Datenobjekt in direktem Bezug zum beschriebenen Prozess steht oder auf eine externe Quelle referenziert wird, aus welcher dieses zu beziehen ist. Die Konstrukte zur Beschreibung von Daten und deren Verlauf innerhalb des Prozessflusses ermöglichen eine klare Darstellung und Nachvollziehbarkeit für beispielsweise über Dokumentationen nachweispflichtige Projekte. Nach [ISO19510, 10.4.1] ist die Lebensdauer eines Datenobjekts auf den Prozess oder Unterprozess beschränkt. Dies bedingt die Einführung eines Datenspeichers.

Klasse *DataStore*: Der Datenspeicher beschreibt nach [ISO19510, 10.4.1] ein Konstrukt zur Einholung oder Speicherung von Informationen für eine Aktivität. Die Lebensdauer geht nach BPMN im Vergleich zum Datenobjekt über die zugehörigen Prozesse hinaus. Die Verwendung des Datenspeichers wird insbesondere in der Kombination mit dem Datenobjekt gesehen. Im speziellen Kontext besteht kein Bedarf an der Erweiterung hinsichtlich zusätzlicher Attribute für die Klasse.

Klasse *AbstractPort*: Für Verbindungen zwischen Flussobjekten sieht BPMN insgesamt vier unterschiedliche Verknüpfungstypen vor. Gemäß [ISO19510, 7.3] sind die *Connecting Objects* beschrieben als *Sequence Flows*, *Message Flows*, *Associations* oder *Data Associations*, wobei im Rahmen dieser Arbeit hauptsächlich der Sequenzfluss im Fokus steht. Nach [ISO19510, 8.4.13] sieht BPMN die Verknüpfung von Flusselementen über einen Sequenzfluss gemeinsam mit einer Richtungsangabe vor, um somit eine Prozessabfolge einzubringen. Dieses Konstrukt wird abgewandelt, indem eine Trennung zwischen der eigentlichen Verbindung und der Richtungsangabe vorgenommen wird. Die somit zusätzlich eingeführte abstrakte Klasse *AbstractPort* repräsentiert einen abstrakten Port, welcher im Rahmen des Prozessmodells als ein *IncomingPort* oder *OutgoingPort* instanziiert werden kann. Somit kann die Richtung des Flusses durch die Verwendung zweier unterschiedlicher Porttypen spezifiziert werden. Die Klasse *AbstractPort* ist als Flusselement definiert und kann daher in einem *FlowElementsContainer* enthalten sein. Jeder Aktivität, jedem Ereignis, Datenobjekt und Datenspeicher kann eine beliebige Anzahl an Eingangs- und Ausgangsports zugewiesen werden, daher die Multiplizität von „0..*“.

Klasse *Connection*: *Connection* ist die ergänzende Klasse zum Port. Sie stellt das Konstrukt zur gegenseitigen Verbindung von Eingangs- mit Ausgangsports bereit, um dadurch den Prozessfluss zu komplementieren. Jeder Eingangs- oder Ausgangsport lässt mehrere zugehörige Verbindungen zu, was durch die Multiplizität von „0..*“ repräsentiert wird. Eine Spezifikation der Verbindung kann über das Attribut *connectionType* vorgenommen werden, welches die Typen gemäß [ISO19510, 7.3] abbildet. Da der *Message Flow* zwar im Kontext der *Collaboration* aber nicht im Fokus der Arbeit steht, wird dieser innerhalb der dazugehörigen Enumeration *connectionTypeEnum* nicht berücksichtigt. Über ein weiteres Attribut *optional* vom Typ *boolean* wird eine Charakterisierung von optionalen Verbindungen ermöglicht.

4.2.2 Konstrukte zur Beschreibung von Organisationsstrukturen und Prozessressourcen

In der Entwicklung von komplexen elektrischen und elektronischen Systemen sind zumeist mehrere Parteien involviert. Durch diese verteilte Entwicklung, welche sowohl firmenintern als auch in Bezug zu externen Zulieferfirmen auftaucht, sind die Kollaboration und die Interaktion zwischen den einzelnen Parteien innerhalb des Prozessflusses von großer Bedeutung. Erschwert wird dies jedoch durch unterschiedliche vorliegende Organisationsformen innerhalb der Firmen.

Um nach vorgeschriebenen und gefestigten Prozessen zu arbeiten, ist eine Kombination von Prozessbeschreibungen mit den dazugehörigen Prozessbeteiligten von Nöten. Dies kann weiterführend durch die Zuweisung bestimmter Rollen und Verantwortlichkeiten unterstützt und gestärkt werden. Somit werden verständliche sowie eindeutige Kommunikationspfade ermöglicht und sowohl interne als auch externe Schnittstellen zwischen den Beteiligten definiert. Im Folgenden wird daher zum einen auf ein Metamodell zur Abbildung von Organisationsstrukturen, einschließlich Zugehörigkeiten und Verantwortlichkeiten, und zum anderen auf die Zuweisung von Prozessressourcen eingegangen.

4.2.2.1 Klassendiagramm OrganizationalStructure

BPMN definiert in [ISO19510, 10.3] eine sogenannte *ResourceRole* als Ressource, welche eine Aktivität ausführen kann oder für diese verantwortlich sein wird. Die Ressource kann in Form einer „speziellen Einzelperson, einer Gruppe, einer Organisationsrolle bzw. Position, oder einer Organisation“ spezifiziert sein. Zudem wird in [ISO19510, 10.1] die *ResourceRole* in den Kontext des Prozesses gesetzt. Dieses alleinstehende Konstrukt ist unter dem Gesichtspunkt von verteilten Entwicklungen sehr grobgranular und wird daher verfeinert. Die Erweiterung sieht die Modellierung von Organisationsstrukturen einschließlich Firmen, Fachbereichen, Abteilungen bis herunter auf einzelne Rollen, sprich Mitarbeiter, vor. Zudem soll eine Trennung zwischen Zugehörigkeiten und Verantwortlichkeiten bezogen auf die Organisationsstruktur eingebracht werden.

In Abb. 4.4 ist das Konzept für die entsprechenden Modellierungskonstrukte gegeben, welche bei Bedarf eine Abbildung der kompletten Zulieferpyramide ermöglichen. Im Folgenden sind die Klassen und deren Attribute näher erläutert. Übergreifend erhalten die Klassen, welche von *Resource* abgeleitet sind, die Attribute *name* und *description*.

Klasse *OrganizationalStructure*: Über die Klasse *OrganizationalStructure* kann die Organisationsstruktur im Sinne einer Zulieferpyramide beschrieben werden. Dies erfolgt in Form einer Zusammenstellung über die Klassen *Company*, *Division* und *Department*. Eine Einordnung dieser Klassen unter einer Organisationsstruktur wird als optional angesehen, daher die vorgesehene Multiplizität von „0..*“. Über die weitere Aggregation kann

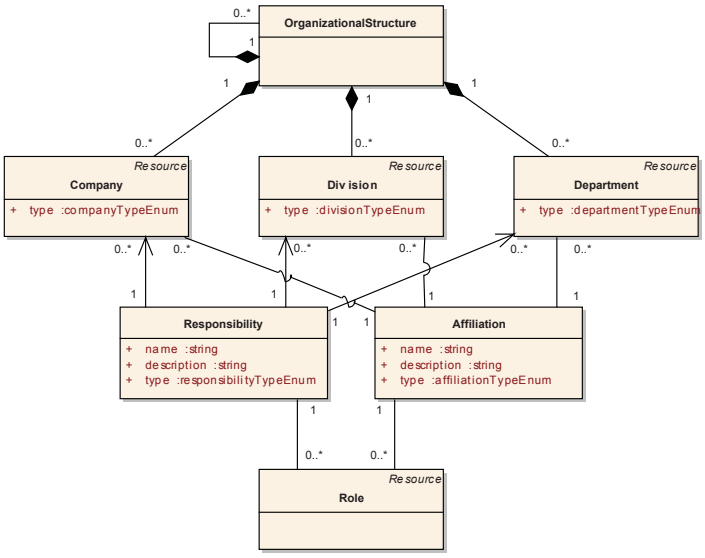


Abb. 4.4: Klassendiagramm *OrganizationalStructure* - vorgeschlagenes Metamodell nach [Adler2014a]

die Klasse *OrganizationalStructure* als Container für eine bzw. mehrere Firmen beispielsweise im Kontext der Zulieferpyramide dienen.

Klasse *Company*, *Division* und *Department*: Die Klassen *Company*, *Division* und *Department* werden zur Beschreibung der Struktur angesehen. Die Klasse *Company* ermöglicht das Hinterlegen von Firmen, welche am Prozess beteiligt sind oder beteiligt werden können. Die Klasse *Division* erlaubt das optionale Hinterlegen von Bereichen innerhalb einer Firma im Rahmen des Organigramms, sollten diese vorhanden sein. Die Klasse *Department* stellt eine Verfeinerung in Abteilungen zur Verfügung. Diese drei Klassen enthalten ein Attribut *type* mit einer jeweils speziell hinterlegten Enumeration, um eine Spezifikation hinsichtlich ihres Typs vorzunehmen. Die Enumeration kann mit entsprechenden Literalen zur Typisierung individuell angelegt werden.

Über das Attribut *type* der Klasse *Company* kann eine spezifische Zuordnung in der Zulieferpyramide erfolgen. Die dazugehörige Enumeration *companyTypeEnum* ermöglicht die Einstufung exemplarisch in einen Zulieferer oder OEM im Kontext des Gesamtprozesses. Über *divisionTypeEnum* sind unterschiedliche Bereichstypen abhängig von der internen Struktur einer Firma beschreibbar, beispielsweise Geschäftsbereiche. Die *departmentTypeEnum* ermöglicht die spezifische Beschreibung einer Abteilung, wie beispielsweise Forschung und Entwicklung.

Klasse *Role*, *Affiliation* und *Responsibility*: Komplementär zu den strukturellen Klassen zur Beschreibung der Unternehmensstruktur ermöglicht die Klasse *Role* das Hinterlegen von einzelnen elementaren Rollen im Sinne einer Einzelperson, beispielsweise als Arbeitnehmer. Dies bildet die unterste Ebene ab und kann durch informative Beschreibungen, wie des Titels der Person, ergänzt werden. Jede einzelne Rolle innerhalb eines Unternehmens besitzt eine oder mehrere Zugehörigkeiten und kann bestimmte Verantwortlichkeiten innehaben. Da diese sich auf unterschiedliche Ebenen im Organigramm niederschlagen können, sind Assoziationen ausgehend von der Klasse *Role* zu den Klassen *Company*, *Division*, *Department* notwendig. Hierdurch kann sowohl ein klassisches pyramidenförmiges Organigramm als auch eine Matrix-Struktur im Unternehmen abgebildet werden. Hierzu lässt sich über die Klasse *Affiliation* die Zugehörigkeit einer Rolle zu einem Organisationsteil spezifizieren. Zudem ermöglicht die Klasse *Responsibility* eine Beschreibung von Verantwortlichkeiten, indem eine Beziehung zwischen der Rolle und bestimmten Ebenen innerhalb der Organisationsstruktur beschrieben wird. Beide Klassen enthalten die Attribute *name*, *description* sowie *type*, um hierdurch optional zusätzliche Informationen zu hinterlegen. Somit kann über die Konstrukte für die Organisationsstruktur eine vollständige Beschreibung der internen Organisation einer Firma erfolgen.

4.2.2.2 Klassendiagramm ProcessResource

Für eine Zuweisung von Prozessverantwortlichkeiten zu konkreten Prozess-Flusselementen kann auf das Konstrukt der Ressource nach [ISO19510, 8.4.12] zurückgegriffen werden. Dieses wird dahingehend erweitert, dass es die Verknüpfung zwischen dem Prozessmodell und dem Organisationsstruktur-Modell ermöglicht. Hierzu verbindet das Konstrukt über eine Assoziation die Prozessartefakte und stellt den Bezug zu den unterschiedlichen Organisationsteilen oder Rollen her, um somit eine feingranulare Zuweisung zu ermöglichen. Das Klassendiagramm ProcessResource in Abb. 4.5 zeigt bereits vorgestellte Klassen im Kontext der Zuordnungen entsprechender Ressourcen.

Bei der Klasse *Resource* handelt es sich um eine Superklasse, welche ausführende Einheiten als unterschiedliche Organisationsteile oder Einzelpersonen abstrakt repräsentiert. Hierzu generalisiert die Ressource sowohl die Klassen *Company*, *Division* und *Department* als auch einzelne Rollen, welche den Prozess-Flusselementen *Activity* und *Event* zugewiesen werden können. Da jede Aktivität oder jedes Ereignis mehrere Ressourcen zugeordnet bekommen kann oder umgekehrt, wird die Multiplizität der Assoziation auf „0..*“ festgelegt. Die Attribute *name* und *description* der Klasse *Resource* sind vom Typ *string* und werden den Sub-Klassen entsprechend vererbt, siehe auch Kapitel 4.2.2.1.

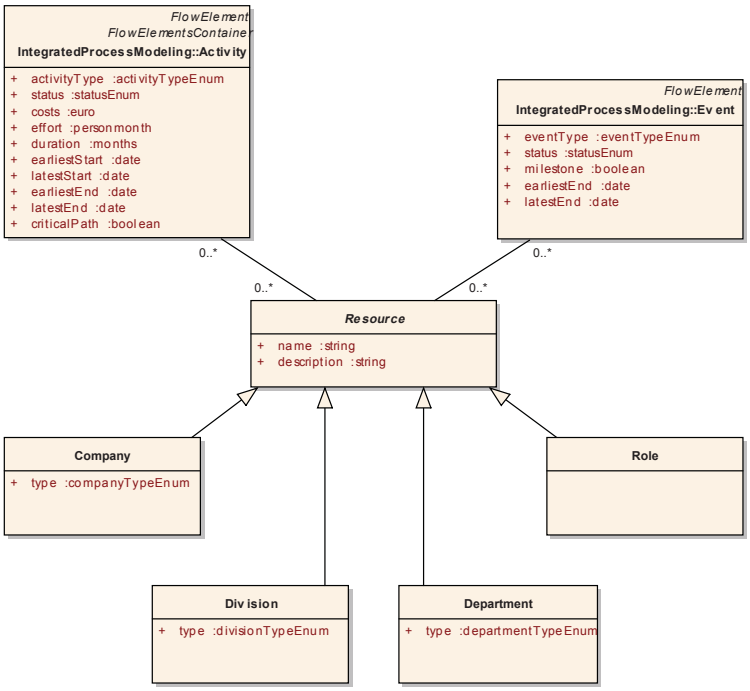


Abb. 4.5: Klassendiagramm *ProcessResource* - vorgeschlagenes Metamodell nach [Adler2014a]

4.3 Bezug zwischen Prozessmodell, Organisationsstrukturen, EEA-Datenmodell und Dokumentenmodell

Durch eine Integration der vorgeschlagenen Metamodelle in eine Entwicklungsumgebung kann aufbauend auf erstellten Prozessbeschreibungen sowie Organisationsstrukturen eine Verknüpfung zwischen den EEA-Daten-, Organisationsstruktur- und Prozesselementen hergestellt werden. Hierbei werden den Verknüpfungen beabsichtigt Freiheitsgrade in Bezug auf Richtung und Multiplizität gelassen, um entwicklungsspezifischen Gegebenheiten Genüge leisten zu können. Ein Überblick über die Verknüpfungen der Modellelemente ist abstrahiert in Abb. 4.6 gegeben. Aus einer prozesstechnischen Sicht wird eine Verknüpfung ausgehend von den Prozesselementen vorgeschlagen. Dies ermöglicht eine Abarbeitung, welche sich gezielt an dem vorliegenden Prozessmodell orientiert. Liegt der Fokus auf dem EEA-Datenmodell und es gilt beispielsweise bestimmte normative Standards einzuhalten, so ist gegebenenfalls eine Verknüpfung ausgehend von Datenelementen zweckmäßig. Eine Verknüpfung ausgehend von den Organisationsstrukturen ermöglicht weiterführend spezifische Blickwinkel für beispielsweise

einzelne Rollen. Aus implementierungstechnischer Sicht ist bei sämtlichen Ansätzen eine Nachverfolgbarkeit unabhängig von der Verknüpfungsrichtung möglich. In Bezug auf die Multiplizität kann eine m:n Verknüpfung zwischen den jeweiligen Elementen vorgenommen werden.

Insbesondere im Kontext von Standards mit geforderten Arbeitsprodukten ist es weiterhin denkbar, ein Dokumentenmodell aufzusetzen, welches sämtliche Arbeitsprodukte inklusive unterschiedlicher Revisionsstände für einen Prozess erfasst und strukturiert, vgl. [Weischer2014]. Dieses Dokumentenmodell kann wiederum mit den Aktivitäten und Ereignissen im Prozess verbunden werden. Somit können sowohl indirekt aus den Datenmodellen als auch ausgehend von den Organisationsstrukturen Dokumente nachverfolgt werden.

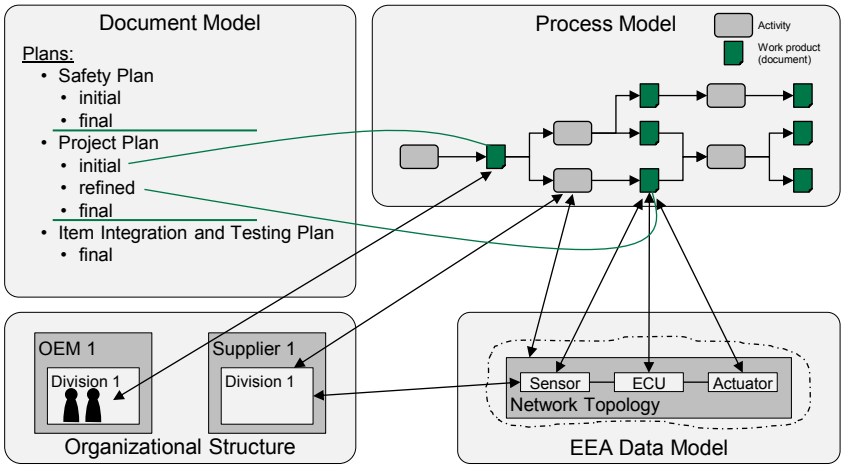


Abb. 4.6: Verknüpfungen der Elemente aus Prozess-, Daten- und Dokumentenmodell sowie der Organisationsstruktur

Eine Verknüpfung der Prozesselemente in Richtung des EEA-Datenmodells kann für die einzelnen Diagramme bzw. auch übergreifend für die entsprechenden Abstraktionsebenen vorgenommen werden. Zusätzlich ist es möglich, sehr feingranular auf einzelne Elemente im Datenmodell zu verweisen. Durch das hierarchische Konzept ist eine exakte Zuordnung von Prozessen, Unterprozessen oder einzelnen elementaren Arbeitsschritten möglich. Hierbei ist insbesondere auf redundante Verknüpfungen, welche sich unter anderem durch die unterschiedlichen Abstraktionsebenen ergeben können, zu achten.

Aufbauend könnte eine automatische Zuordnung eines EEA-Datenelementes zu einem spezifischen Teilprozess erfolgen. Hierfür müssten jedoch spezifische Informationen hinterlegt sein, um eine eindeutige Verknüpfung in Richtung des dazugehörigen Prozesselementes zu ermöglichen. Inwiefern beispielsweise ein Datenelement dem Hardware-

oder Softwareentwicklungsprozess zugeordnet wird, könnte über die dazugehörige Metamodell-Klasse bzw. vorhandene oder zusätzlich eingebrachte Attribute identifiziert werden.

In [44] wird eine asynchrone und synchrone Modellierung innerhalb einer EEA-Entwicklungsumgebung im Kontext von kollaborativer Arbeit vorgestellt. Hierdurch könnte im Rahmen des Prozessmodells die Abarbeitung und damit der Status der einzelnen Prozessschritte von allen Beteiligten zeitgleich verfolgt werden. Dies lässt sich wiederum mit einer speziellen grafischen Aufbereitung des Prozessmodells für den produktiven Einsatz hervorheben.

4.4 Prozessmanagement

Das Management von Prozessen ist von großer Bedeutung für eine zielgerichtete Produktentwicklung. Unter dem Aspekt von sicherheitsbezogenen Produkten spricht ISO 26262 von „Projekt-unabhängigen Anforderungen in Hinblick auf die involvierten Organisationen“ im Sinne eines Gesamt-Sicherheitsmanagements. Zudem werden „Projektspezifische Anforderungen in Bezug auf Managementaktivitäten im Sicherheitslebenszyklus“ erhoben [ISO26262, 2-1]. Diese greifen beispielsweise das „Management während der Konzeptphase, Produktentwicklung und nach der Produktionsfreigabe“ auf. Um diesen Forderungen an das Management in einem ersten Schritt zu genügen, wird im Folgenden ein Konzept dargestellt, welches diesbezüglich eine Unterstützung leistet.

Der Ansatz für das Prozessmanagement basiert auf dem beschriebenen Prozessmodell, welches in der integrierten Umgebung hinterlegt ist, und kann auf diesem ausgeführt werden. Unterschiedliche Analyse- und Planungsaspekte können bereitgestellt werden, um die Handhabung von komplexen Produktentwicklungen zu unterstützen. Diese basieren auf der Integration der adaptierten und erweiterten Metamodell-Konstrukte, dem Vorteil Elemente des Prozessmodells mit denen des EEA-Datenmodells zu koppeln sowie dem Ressourcen-Konstrukt. Zusätzlich können klassische Funktionalitäten von modellbasierten Werkzeugumgebungen kombiniert und für ein erleichtertes Tagesgeschäft genutzt werden.

Darauf aufbauend kann in Hinsicht auf die Darstellung von Prozessmodellen eine grafische Aufbereitung erfolgen und diese zusätzlich mit Funktionalitäten für eine Ausblendung von Prozesselementen, welche nicht im aktuellen Fokus eines spezifischen Arbeitskontextes stehen, kombiniert werden. Neben der grafischen Unterstützung müssen Algorithmen für die Berechnung von zeitlichen Aspekten sowie der Identifikation des kritischen Pfades bereitgestellt werden, um die Analyse und Planung von Prozessen zu unterstützen.

Im Folgenden wird zunächst allgemein auf die Adaption von Konstrukten aus dem Variantenmanagement zur Unterstützung des Prozessmanagements eingegangen. Anschließend erfolgt die Beschreibung von ausgewählten spezifischen Konzepten bezüglich Prozessanalyse und -planung, welche sich zum Teil auf die Konstrukte aus dem angepassten Variantenmanagement beziehen.

4.4.1 Adaption von Konzepten des Variantenmanagements

Für die Unterstützung des Prozessmanagements sind Ansätze eines Variantenmanagements angemessen, um zum einen eine Strukturierung und zum anderen eine grafische Repräsentation der Ergebnisse zu ermöglichen. Hierzu können die im Rahmen von [137] präsentierten Konstrukte für ein Variantenmanagement herangezogen und adaptiert werden.

Daher repräsentiert das Prozessmodell selbst einen *Concept Space* für Analyse und Planung in Hinsicht auf den modellierten Gesamtprozess, wie in Abb. 4.7 gezeigt. Innerhalb des *Concept Space* sind unterschiedliche potentielle *Process Analysis/Planning Variants* verfügbar. Diese Varianten werden für eine Strukturierung und grafische Repräsentation von Analyse- sowie Planungsergebnissen genutzt. Eine *Process Analysis/Planning Variant* repräsentiert einen spezifischen Analyse- oder Planungskontext, beispielsweise die Hervorhebung von allen Prozesselementen, welche in die Änderung eines spezifischen Dokuments involviert sind.

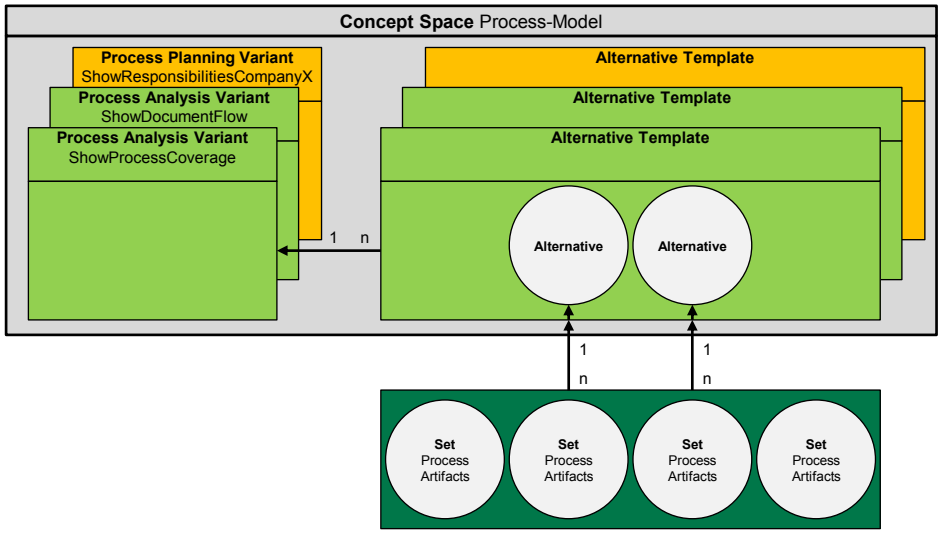


Abb. 4.7: Verwendung von Konzepten des Variantenmanagements zur Analyse und Planung von Prozessen

Sets repräsentieren die elementaren Konstrukte zur Strukturierung und werden genutzt, um Prozessartefakte hinsichtlich eines spezifischen Kontextes, welcher analysiert oder geplant werden muss, zu bündeln oder zu gruppieren. Eine angemessene Granularität muss abgeleitet werden, um diese in mehreren *Alternatives* wiederverwenden zu können. Ein *Set* kann beispielsweise als alle vorkommenden Prozesselemente, für welche eine spezifische Rolle verantwortlich ist, definiert werden. Basierend auf der Beschreibung von spezifischen *Sets* auf angemessenem Granularitätslevel, kann eine Konjunktion von kontextbezogenen Prozessartefakten ermöglicht werden.

Alternatives wiederum können unterschiedliche *Sets* von Prozessartefakten bündeln. Für Analysezwecke muss nur eine *Alternative* im Kontext einer *Process Analysis Variant* bereitgestellt werden. Im Sinne einer Planung müssen unterschiedliche Alternativen vorhanden sein.

Alternative Templates können eine Disjunktion von unterschiedlichen *Alternatives* repräsentieren. Dies erleichtert die Entscheidung für die am besten geeignete *Alternative* im Rahmen einer Prozessplanung. Die einzelnen *Alternative Templates* müssen mit den *Process Analysis/Planning Variants* verknüpft werden.

Um die grafische Repräsentation des Prozessmodells im Zuge einer Verblassung von nicht-relevanten Prozesselementen in einem spezifischen Kontext zu ermöglichen, wird eine Kopplung der Darstellung sowohl über die *Process Analysis/Planning Variants* als auch die *Alternatives* hergestellt. Dies ist im Umkehrschluss auch für eine Hervorhebung relevanter Prozesselemente möglich.

4.4.2 Prozessanalyse

Zur Analyse von Prozessen kann eine erste Identifikation von offensichtlichen Inkonsistenzen und Lücken im Prozessfluss durch den Prozessmodellierer während der Prozessbeschreibung erfolgen. Abgesehen davon kann eine Konsistenzanalyse durch implementierte Abfragen auf dem modellierten Prozess ermöglicht werden. Hierzu kann eine Überprüfung sowohl auf syntaktische Fehler in der Modellierung als auch für individuell eingebrachte Inhalte und Einschränkungen erfolgen. Diese Abfragen können nebenläufig während der Erstellung des Prozessmodells ausgeführt oder auf das aktuelle Prozessmodell angewendet werden.

Um eine bessere Nutzerfreundlichkeit des Prozessmodells zu ermöglichen, kann eine Analyse auf den verschiedenen Prozessartefakten hinsichtlich ihrer hinterlegten Attribute durchgeführt werden. Dies kann für eine anschließende grafische Aufbereitung genutzt werden, um beispielsweise die Einfärbung des Hintergrunds für Objekte der Klasse *DataObjects* nach einem benutzerdefinierten Schema automatisch, basierend auf den unterschiedlichen Werten des Attributs *dataObjectType*, bereitzustellen.

Eine weitere Analyse ist die Bestimmung, in welchem Unterprozess sich spezifische Objekte der Klasse *DataObjects* bezogen auf das Attribut *state* beispielsweise im initialen oder finalen Status befinden. Dies repräsentiert eine Art Dokumentenfluss durch den Prozess. Dieser Fluss der *DataObjects* und ihren dazugehörenden Aktivitäten, welche eine Änderung des Status herbeiführen, unterstützt eine klare Nachvollziehbarkeit. Hierfür kann das im vorherigen Kapitel vorgestellte adaptierte Konzept aus dem Variantenmanagement verwendet werden. Die Abb. 4.8 zeigt dies anhand des spezifischen *DataObjects* „SafetyPlan“. Wie ersichtlich wird hierfür exakt ein *Set* einer *Alternative* und diese wiederum einem *Alternative Template* zugeordnet. Durch dieses Konstrukt kann eine Hervorhebung des Dokumentenflusses im Prozessmodell erfolgen. Analog können auch Gruppen von *DataObjects*, beispielsweise alle Reports, in einem *Set* gebündelt und in den Prozessdiagrammen hervorgehoben werden.

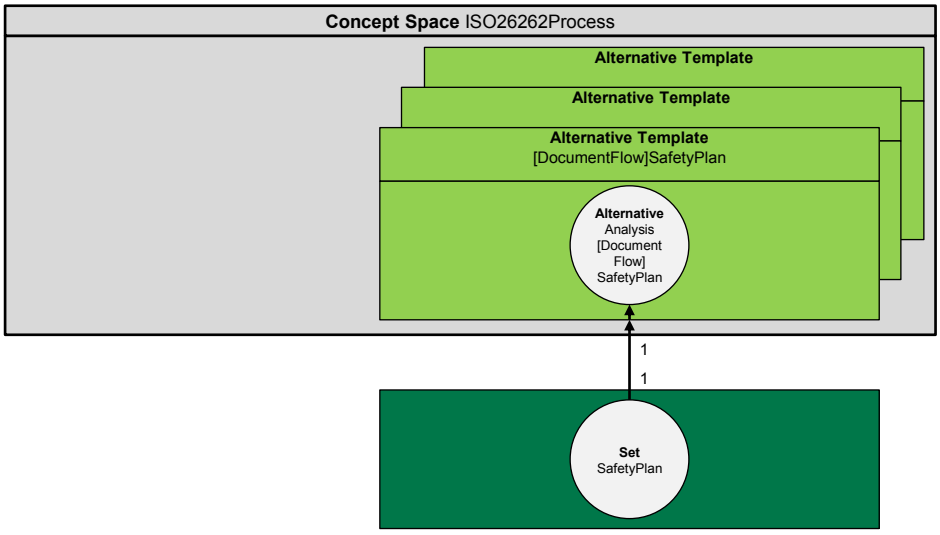


Abb. 4.8: Prozessanalyse für einen Dokumentenfluss

Im Hinblick auf interdisziplinäre, innerbetriebliche und firmenübergreifende Arbeit dient das Prozessmodell einschließlich der zugewiesenen Ressourcen als eine Repräsentation von Prozessverantwortlichkeiten auf unterschiedlichen Organisationsebenen. Dies ermöglicht ein Tailoring des Prozesses in Hinsicht auf einzelne Verantwortlichkeiten. Die Umgebung neben dem zu untersuchenden Teilprozess kann somit ausgeblendet werden, um sich auf die relevanten Schritte für unterschiedliche Verantwortlichkeiten zu fokussieren. Zusätzlich kann es die interne und externe Kommunikation unterstützen und konstituiert eindeutige Schnittstellen zwischen den Beteiligten, welche in den Prozess involviert sind.

Hierzu ist in Abb. 4.9 die Anwendung auf einzelne Ebenen innerhalb der Zulieferpyramide ersichtlich. Somit kann als Mehrwert eine klare Trennung zwischen den Verantwortlichkeiten in Bezug auf OEM und alle Zulieferer herausgearbeitet werden. Weiterhin ist eine getrennte Darstellung der Verantwortlichkeiten einzelner Firmen aus der Zulieferpyramide möglich.

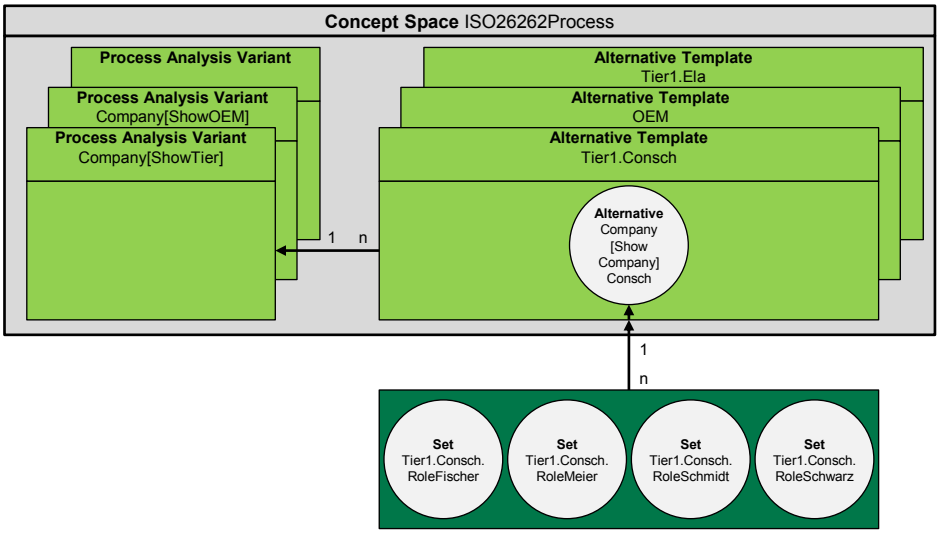


Abb. 4.9: Prozessanalyse für Verantwortlichkeiten im Kontext einer Zulieferpyramide

Hierfür werden die Zuständigkeiten aller zu einer Firma gehörigen Rollen über *Sets* einer jeweiligen *Alternative* zugeordnet. Basierend auf dem Firmentyp kann eine Einordnung in die Zulieferpyramide im Rahmen einer *Process Analysis Variant* vorgenommen werden. Diese ermöglichen eine getrennte Darstellung für einzelne Ebenen innerhalb einer Zulieferpyramide. Die *Process Analysis Variant* bündelt somit exemplarisch die Verantwortlichkeiten mehrerer Tier1. Da die Informationen zu einzelnen Firmen bereits in die *Alternative* eingeflossen sind, wird auch eine gesonderte Darstellung innerhalb des Prozessmodells für einzelne Firmen ermöglicht.

Die Analyse von Verantwortlichkeiten innerhalb einer einzelnen Firma wird nach Abb. 4.10 vorgenommen. Dies dient dem Aufbau übergreifender Verantwortlichkeiten für bestimmte Bereiche bzw. Abteilungen über die Zusammenführung mehrerer *Sets* im Zuge einzelner Rollen. Ein *Alternative Template* hat hierbei exakt eine *Alternative* zugewiesen.

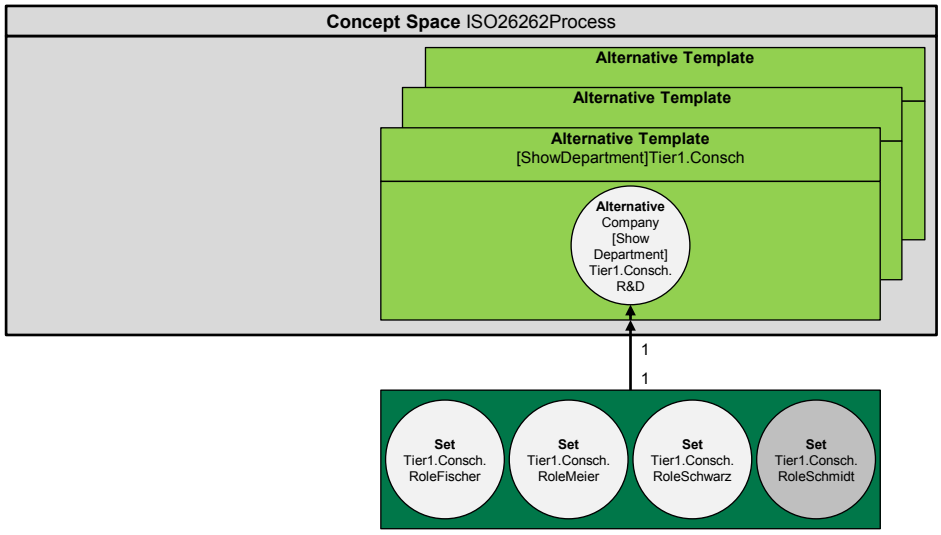


Abb. 4.10: Prozessanalyse für Verantwortlichkeiten im Kontext einzelner Firmen

Um einzelnen Rollen jederzeit eine Analyse der jeweiligen Verantwortlichkeiten im Prozess zu ermöglichen, wird dies über die Zuordnung nach Abb. 4.11 bis auf unterste Ebene heruntergebrochen. Hierzu wird dem *Alternative Template* die gleiche Anzahl an *Alternatives* wie vorliegende Rollen zugeordnet. Jeder der *Alternatives* wird eine Rolle über ein entsprechendes *Set* zugewiesen. Somit erfolgt eine eindeutige 1:1 Abbildung.

Basierend auf dem Prozessmodell und der Ressourcenzuweisung kann ebenfalls die Abdeckung des Gesamtprozesses überprüft werden. Das grafische Verblassen ermöglicht eine sofortige Darstellung von nicht-abgedeckten jedoch relevanten Prozesselementen. Dies kann unter anderem auch für eine Prozessplanung dienlich sein.

Durch Verwendung des integrierten Prozessmodells können weiterhin Statusänderungen von Aktivitäten und Ereignissen im Kontext des Prozess- und Arbeitsfortschritts über das Attribut *status* nachvollzogen werden. Dies unterstützt ein eindeutiges Verständnis vom aktuellen Status innerhalb des Prozessflusses in Hinsicht auf die unterschiedlichen Literale der Enumeration *statusEnum*.

Auf Grundlage eines konsistenten Prozessmodells, kann ein mathematisches Graphenmodell konstruiert werden. Dieses ermöglicht eine Anwendung der Methode des kritischen Pfades, im Englischen Critical Path Method (CPM), um zudem die frühesten/spätesten Start-/Enddaten von Objekten der Klasse *Activity*, basierend auf der Dauer der verbundenen *Activities* und einem spezifizierten Prozess-Startdatum, zu berechnen. Dies kann unter Verwendung der Vorwärts-/Rückwärtsterminierung, im Englischen forward and backward pass, als eine CPM-Technik durchgeführt werden.

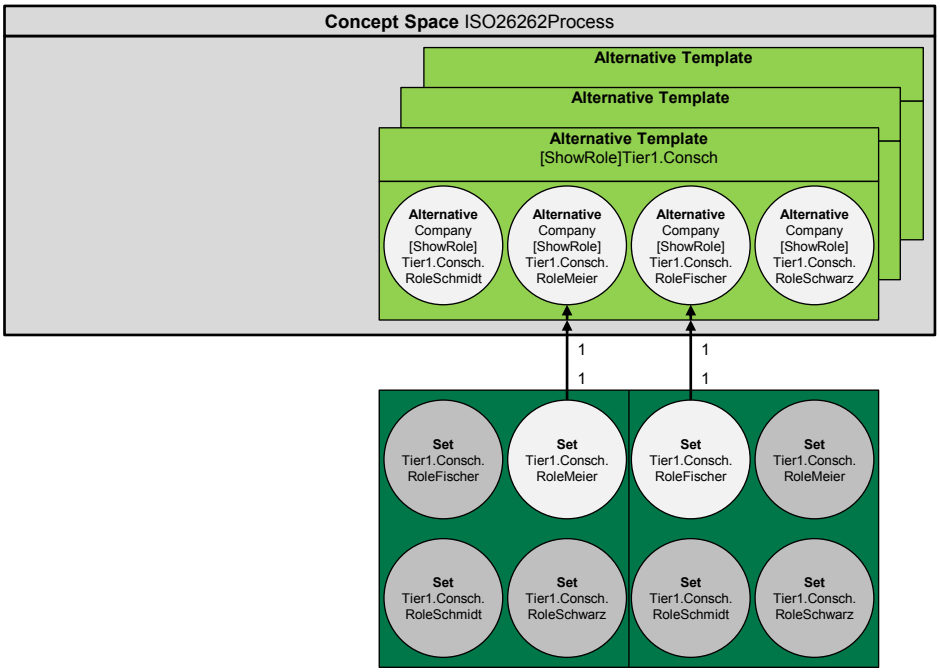


Abb. 4.11: Prozessanalyse für Verantwortlichkeiten im Kontext einzelner Rollen

Im Rahmen der verankerten Attribute *costs*, *effort* und *duration* kann weiterhin eine Analyse hinsichtlich der Wirtschaftlichkeit auf dem Prozessmodell durchgeführt werden, um Management-Entscheidungen zu unterstützen. Diese Methode ist ebenfalls im Fokus einer Prozessplanung und kann durch entsprechend implementierte Berechnungsvorschriften, welche die relevanten Daten aus dem Prozessmodell verwenden, weiterführend umgesetzt werden.

4.4.3 Prozessplanung

Die bereitgestellten Konstrukte für die Beschreibung von Organisationsstrukturen und deren Zuweisung auf spezifische Prozesselemente unterstützen auf direkte Weise die Planung von notwendigen Ressourcen, welche aus einer spezifischen Prozesssicht involviert sein müssen.

Weiterhin muss für die Planung von firmenübergreifenden Arbeiten eine Abdeckung des Gesamtprozesses gewährleistet sein. Diese Analyse kann wiederum durch die Verwendung des adaptierten Variantenmanagements unterstützt werden, indem zwischen Alternativen, beispielsweise im Zuge unterschiedlicher konkurrierender Zulieferfirmen oder

Dienstleister, gewählt und entschieden werden kann. Somit kann über unterschiedliche Konstellationen geplant werden, von wem die geforderten Aktivitäten produktiv durchgeführt und damit bedient werden. Durch Verwendung der grafischen Aufbereitung für die Abdeckung von Teilprozessen können Entscheidungen, beispielsweise basierend auf dem Lieferumfang, ermöglicht werden.

Hierfür wird nach Abb. 4.12 jeder *Alternative* exakt ein *Set* zugewiesen, womit die Anzahl der *Sets* der Anzahl an *Alternatives* gleicht. Das *Set* beinhaltet alle Aktivitäten, welche von der Zulieferfirma übernommen werden und bildet daher die entsprechenden Teilprozesse ab. Aus Sicht unterschiedlicher Aufgabenbereiche werden dazugehörige *Alternative Templates* angelegt, exemplarisch dargestellt für die Unterstützung von Teilprozessen im Bereich Hardware und Software. Die angelegten *Alternative Templates* werden in der *Process Planning Variant* zusammengeführt.

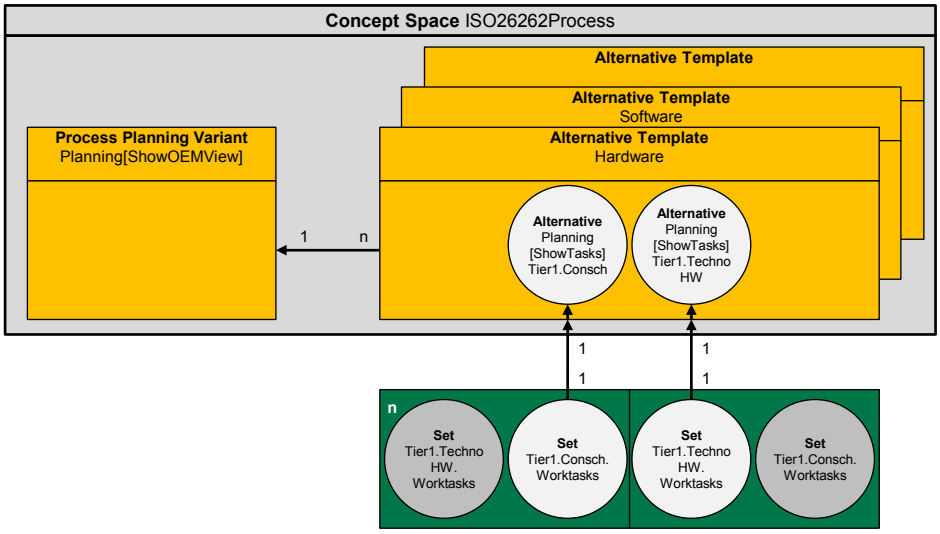


Abb. 4.12: Prozessplanung für die Abdeckung von Prozessen

Eine weitere Unterstützung der Prozessplanung steht unter dem Gesichtspunkt von zeitlichen Aspekten im Zuge der CPM, welche bereits in der Prozessanalyse aufgegriffen wurde. Hierfür muss das in der Metamodell-Klasse *Activity* eingeführte Attribut *duration* genutzt werden. Basierend auf dem Prozessmodell kann ausgehend von der zeitlich gesehen ersten Aktivität ein mathematisches Graphenmodell aufgespannt werden, welches sämtliche Prozesselemente in den Unterprozessen miteinbezieht. Die darauf angewandte Vorwärtsterminierung berechnet die frühesten Start- sowie Endzeitpunkte der Aktivitäten und hinterlegt diese in den Attributen *earliestStart* und *earliestEnd*. Bei Ereignissen betrifft dies das Attribut *earliestEnd*. In einem weiteren Schritt wird die

Rückwärtsterminierung angewandt, um somit die spätesten Start- und Endzeitpunkte für die Aktivitäten zu erhalten. Für die Ereignisse betrifft dies den Wert des Attributs *latestEnd*. Diese zeitlichen Informationen unterstützen die Planung des Prozesses in Hinsicht auf die Ressourcenzuweisung oder die Definition von Meilensteinen. Zur Analyse aus projekttechnischer Sicht kann die CPM zur Identifikation des kritischen Pfades genutzt werden. Eine Zugehörigkeit von Aktivitäten zum kritischen Pfad kann über das boolesche Attribut *criticalPath* gesetzt werden. Dieses ermöglicht wiederum über einen eingebrachten Automatismus die grafische Hervorhebung des kritischen Pfades im Diagramm.

Die aufgeführten Analysen für das Prozessmodell und die Bestimmung von zeitlichen Informationen unterstützen einen ersten konzeptionellen Ansatz für eine Prozessoptimierung, da sie unter anderem einer Identifikation von Parallelzweigen oder Einführung von neuen Prozesszweigen dienen. Zusätzlich können Prozessbeschreibungen aus anderen Arbeitsgebieten und Fachrichtungen verknüpft oder eventuell verschmolzen werden.

4.5 Integration in domänenspezifische Entwicklungsumgebung

In Kapitel 2.3 wurden exemplarisch Architekturbeschreibungssprachen für die modellbasierte Entwicklung von automobilen Fahrzeugsystemen vorgestellt. Die Entwicklungsumgebung PREEvision stellt für die Beschreibung von großformatigen EEAs entsprechende Abstraktionsebenen sowie geeignete Modellierungskonzepte basierend auf einem eigenen Metamodell zur Verfügung. Bedingt durch das vorhandene Metrik-Framework mit welchem sowohl benutzerdefinierte Berechnungen als auch Modelloperationen auf den hinterlegten Modellen ausgeführt werden können und der Möglichkeit von spezifischen Erweiterungen durch beispielsweise generische Attribute, wurde es für einen ersten integrierten Ansatz in einer domänenspezifischen Entwicklungsumgebung ausgewählt. Die wissenschaftlich-prototypische Umsetzung erfolgte in PREEvision Version 6.0 SP1, gleichwohl ist dies in anderen Architekturbeschreibungssprachen nach Kapitel 2.3 ebenfalls denkbar.

4.5.1 Prozessmodellierung

Für die prototypische Umsetzung des Metamodell-Vorschlags im Kontext der Prozessmodellierung wurden die bestehenden Konstrukte innerhalb von PREEvision untersucht. Hierbei erwies sich der entsprechende Diagrammtyp aus der logischen Architekturebene zur Modellierung von Prozessbeschreibungen als geeignet, da dieser zum einen ein Konstrukt zur Hierarchisierung beinhaltet und sich zum anderen das dahinterliegende Verknüpfungskonzept in Richtung fast aller weiteren Architekturebenen aufspannen lässt. Dies dient aus wissenschaftlicher Sicht einer ersten orthogonalen Beschreibung, gefordert nach Kapitel 4.1.1. Um eine integrierte Prozessmodellierung zur Verfügung zu stellen, wurden im Rahmen der prototypischen Implementierung geeignete und bereits

vorhandene Konstrukte mit dem vorgeschlagenen Metamodell verglichen und mit einer konsistenten Semantik versehen. Somit wird die logische Architekturebene prototypisch mit Konstrukten für die Prozessmodellierung und deren spezifischen Attributen gemäß dem vorgeschlagenen Metamodell angereichert.

Das in Kapitel 4.2.1.1 beschriebene Konstrukt im Kontext des *FlowElementsContainer* als Container für Prozess-Flusselemente kann durch die Klasse *LogicalBuildingBlock* aus PREEvision umgesetzt werden. Hierüber kann eine Hierarchisierung von Aktivitäten in Form von Prozessen, Unterprozessen und elementaren Arbeitsschritten vorgenommen werden. Eine grafische Beschreibung des Prozesses auf den jeweiligen Hierarchieebenen wird durch ein zum *LogicalBuildingBlock* zugehöriges Diagramm ermöglicht. Im Rahmen von Prozess-Flusselementen wurde der Metamodell-Vorschlag hinsichtlich der Klasse *Activity* auf *LogicalBuildingBlock* und für *Event* auf *LogicalFunction* abgebildet. Sämtliche weitere Klassen im Kontext der Beschreibung von Prozessmodellen nach Kapitel 4.2.1 konnten auf vorhandene Klassen in PREEvision abgebildet werden.

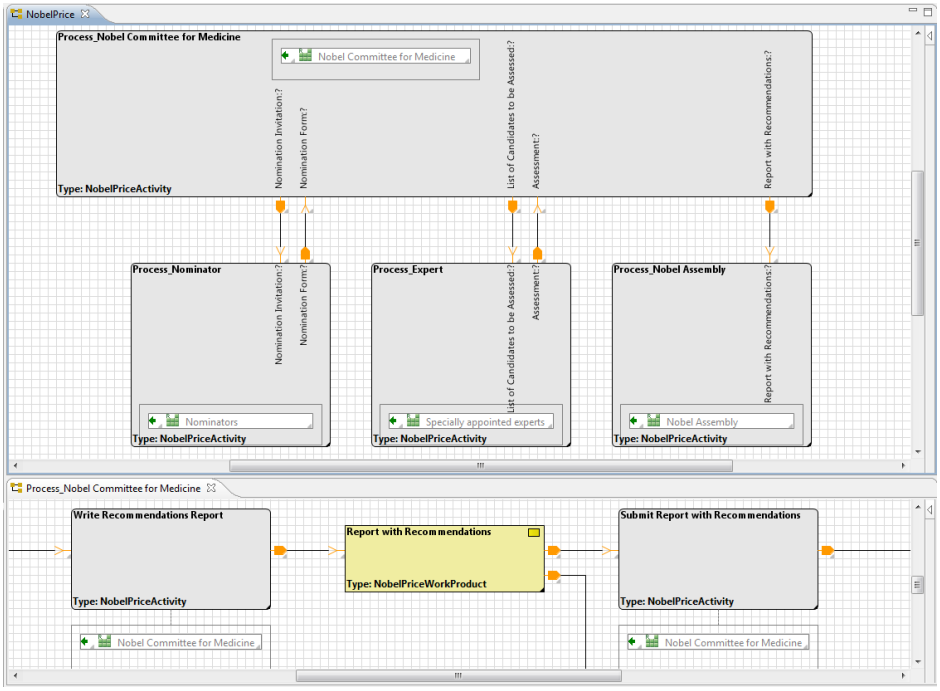


Abb. 4.13: Screenshot: Exemplarisches Prozessszenario für den Nobelpreis in Medizin nach [BPMNex, S.26]

In Abb. 4.13 wurde die integrierte Prozessmodellierung auf das im Grundlagenkapitel 2.5.3.3 exemplarisch beschriebene BPMN-Prozessszenario in [BPMNex, S.26] angewandt. Neben dem Hierarchisierungskonzept sind vorausgreifend auch die Ressourcenzuweisungen für die jeweiligen Aktivitäten ersichtlich.

Um insbesondere den Umgang mit umfangreichen Prozessmodellen zu erleichtern und ein darauf aufsetzendes Prozessmanagement zu unterstützen, wird im Folgenden unter anderem die Anreicherung mit generischen Attributen über eine in PREEvision implementierte Metrik, der Aufbau der Organisationsstruktur einschließlich Ressourcenzuweisung und eine automatische grafische Aufbereitung des Prozessmodells detaillierter erläutert.

4.5.2 Anreicherung mit generischen Attributen

Um die im Rahmen des Metamodell-Vorschlags vorgestellten Attribute einschließlich ihrer Datentypen in die modellierte Prozessbeschreibung einzubringen, werden innerhalb von PREEvision sowohl die spezifizierten Datentypen als auch die entsprechenden Enumerationen sowie deren dazugehörige Literale in der Bibliothek hinterlegt, siehe Abb. 4.14.

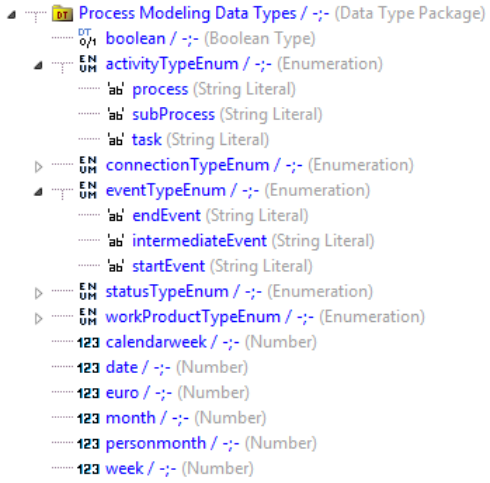


Abb. 4.14: Auszug der angelegten Datentypen nach dem Metamodell-Vorschlag aus Kapitel 4.2

Für die Erstellung der Attribute einschließlich der Zuweisung der zugehörigen Datentypen wird die in Abb. 4.15 dargestellte Metrik verwendet. Diese hat Veränderungen am Modell vorzunehmen, somit werden die enthaltenen Berechnungsblöcke als Modelloperationen ausgeführt. Hierbei muss der Metrik ein Kontext für die Ausführung in

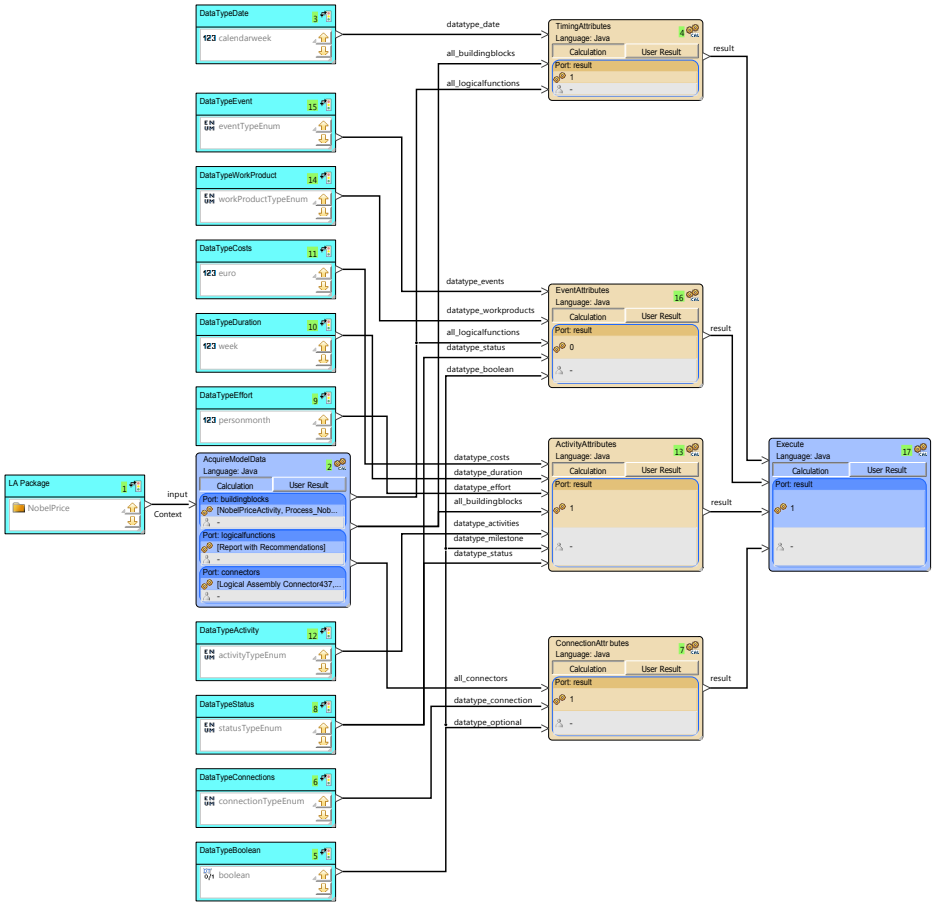
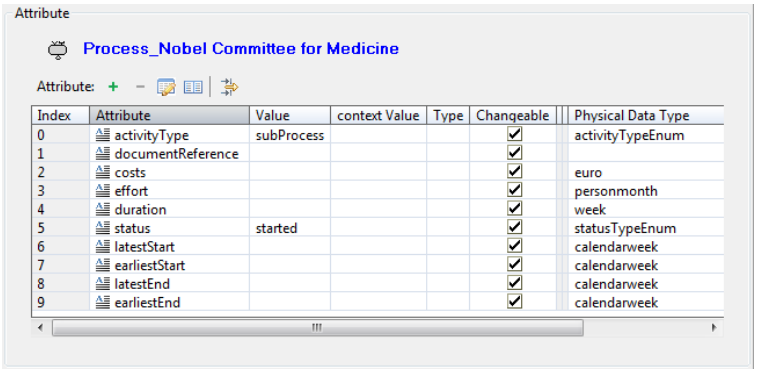


Abb. 4.15: Metrik zur automatisierten Erstellung der Attribute nach dem Metamodell-Vorschlag aus Kapitel 4.2

Form eines logischen Architekturpaketes übergeben werden, in welchem ein integriertes Prozessmodell nach den vorgestellten Konzepten vorliegt. Die Erstellung der Attribute erfolgt automatisiert und übergreifend für alle zum Paket gehörenden Aktivitäten, Ereignisse und Verbindungen. Aufgrund der differierenden Attribute für die drei aufgegriffenen Arten von Prozess-Flusselementen werden diese getrennt voneinander behandelt. Zudem wird das Anlegen der Attribute bzgl. zeitlicher Aspekte für Aktivitäten und Ereignisse gekapselt von den restlichen Attributen vorgenommen. Die relevanten Datentypen werden über Kontextblöcke bereitgestellt, um eine direkte Zuweisung zu den anzulegenden Attributen zu ermöglichen. Die Metrik berücksichtigt bei erneuter Ausführung

bereits erstellte Attribute, somit kann eine Erweiterung bzw. Überarbeitung des Prozesses jederzeit erfolgen.

In Abb. 4.16 sind die von der Metrik erstellten Attribute für eine Aktivität in der Eigenschaftsansicht von PREEvision ersichtlich. Exemplarisch wurde aus den zulässigen Werten der Enumeration *activityTypeEnum* eine entsprechende Einordnung als Unterprozess für die Aktivität vorgenommen und der Status auf gestartet gesetzt. Angemerkt sei, dass die Attributwerte für die zeitlichen Aspekte durch den Benutzer eingetragen oder über einen Automatismus berechnet werden können, siehe Kapitel 4.6.6.5.



Index	Attribute	Value	context Value	Type	Changeable	Physical Data Type
0	activityType	subProcess			☑	activityTypeEnum
1	documentReference				☑	
2	costs				☑	euro
3	effort				☑	personmonth
4	duration				☑	week
5	status	started			☑	statusTypeEnum
6	latestStart				☑	calendarweek
7	earliestStart				☑	calendarweek
8	latestEnd				☑	calendarweek
9	earliestEnd				☑	calendarweek

Abb. 4.16: Erzeugte generische Attribute in der Eigenschaftsansicht

4.5.3 Modellierung der Organisationsstruktur

Für die Modellierung von sowohl Organisationsstrukturen als auch der Zulieferpyramide wurde im Rahmen einer ersten prototypischen Durchgängigkeit die Anforderungsebene und deren Konstrukte mit einer anderen Semantik versehen. Dies ermöglicht das Mappingkonzept von PREEvision umfassend zu nutzen, nachdem verschiedene beteiligte Bereiche, Abteilungen und Rollen mit dem Konstrukt einer Anforderung zweckentfremdet dargestellt wurden. Die Klasse der Anforderungspakete repräsentiert die im Metamodell dargestellten Firmen und die Klasse Anforderungen die Bereiche, Abteilungen und einzelne Rollen. Somit wird eine eindeutig ersichtliche Baumstruktur im Modellbaum hinsichtlich der Zugehörigkeiten im Sinne der *Affiliation* bereitgestellt. Verantwortlichkeiten innerhalb des Unternehmens werden über ebeneninterne Verknüpfungen hergestellt, welche sich mit zusätzlichen Informationen anreichern lassen. Dies kann für unterschiedliche Ebenen generisch angewandt werden. Ermöglicht wird sowohl die Beschreibung von klassischen Firmenstrukturen als auch von Matrix-Strukturen sowie Sonderformen. Eine Verknüpfung mit den eingetragenen PREEvision-Nutzern muss nicht zwingend vorgenommen werden, da dies eine Einschränkung der

modellierten Rollen zur Folge hätte. Es wird somit im ersten Schritt eine klare Trennung zwischen den Prozessbeteiligten und den Nutzern der Entwicklungsumgebung erzielt. Eine Modellierung der Organisationsstrukturen für das exemplarische Prozessszenario aus BPMN lässt sich ebenfalls abbilden und ist in Abb. 4.17 dargestellt.

Abb. 4.17: Screenshot der Modellierung einzelner Rollen im Kontext des Nobelpreises 2014 für Medizin nach [118], [119]

4.5.4 Verantwortlichkeiten gegenüber dem Prozess und Bezug zum EEA-Datenmodell

Basierend auf der modellierten Organisationsstruktur ist die Zuweisung von Ressourcen zu den innerhalb des Prozesses vorkommenden Flusselementen vorzunehmen, um somit Prozessschritt-Verantwortlichkeiten nach der *Resource* zu definieren. Hierzu werden Verknüpfungen eingesetzt, welche auf die unterschiedlichen Ebenen der Prozesse bis herunter auf elementare Prozesselemente angewandt werden können. Diese Verknüpfungen können in unterschiedliche Pakete strukturiert und über eine Typisierung spezifiziert werden. Auszugsweise ist die Zuweisung der Ressourcen für das exemplarische Prozessszenario in Abb. 4.18 ersichtlich.

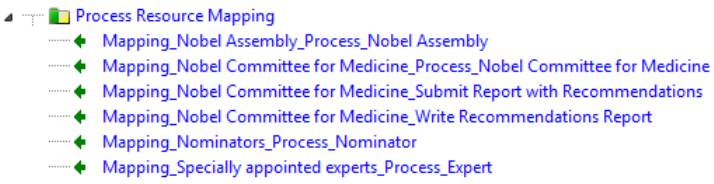


Abb. 4.18: Modellbaumansicht für die Ressourcenzuweisung auf Prozess-Flusselemente

Für den Bezug zwischen Prozess-Flusselementen und dem EEA-Datenmodell kann aufgrund des integrierten Ansatzes ebenfalls die vorangegangene Methodik genutzt werden. Hierbei kann sowohl der Bezug zu einzelnen Ebenen als auch zu elementaren Datenmodell-Artefakten hergestellt werden. Weiterhin können Elemente aus den Organisationsstrukturen mit Elementen aus dem EEA-Datenmodell verknüpft werden. Die Differenzierung der beschriebenen Verknüpfungsarten erfolgt über entsprechende Typen und Pakete.

4.5.5 Grafische Aufbereitung des Prozessmodells

Prozessbeschreibungen können im industriellen Umfeld bedingt durch zahlreiche Unterprozesse sowie Querverbindungen sehr umfangreich sein und dadurch unübersichtlich werden. Die Lesbarkeit eines Prozessmodells ist jedoch für die Prozessbeteiligten von großer Bedeutung. Um eine Unterstützung für die Erfassung der Prozesse zu bieten, sind diese in geeigneter Art und Weise aufzubereiten. Dies kann in erster Linie erreicht werden, indem zum einen das innerhalb von BPMN beschriebene Erscheinungsbild für Flusselemente gegeben ist, zum anderen über gezielte Änderungen der grafischen Darstellungen. Letzteres ist angelehnt an [ISO19510, 7.3], worin insbesondere erwähnt wird, dass zusätzliche Variationen unter dem Gesichtspunkt der Komplexität hinzugefügt werden dürfen, jedoch das äußere Erscheinungsbild der Diagramme in den Grundzügen beibehalten werden muss.

Daher wird eine Aufbereitung über Typisierungen und Gruppierungen von bestimmten Prozess-Flusselementen bereitgestellt. Als insbesondere für wichtig erachtet werden die Aktivitäten, Ereignisse, Datenobjekte und Verbindungen. Anhand der in Kapitel 4.2 eingebrachten Attribute betrifft dies *activityType*, *eventType*, *dataObjectType* und *optional*. Die darauf aufbauende Einfärbung soll einem individuell einstellbaren Farbschema und gleichzeitig den Empfehlungen von BPMN genügen, um somit die Lesbarkeit im Rahmen von vorgegebenen grafischen Merkmalen sowie eigenen Erweiterungen zu erleichtern.

Um dies zu unterstützen, wird eine ausführbare Metrik nach Abb. 4.19 zur Verfügung gestellt, welche die grafische Repräsentation über die Modellobjekt-Konfiguration verändert. Für die Konfiguration werden die entsprechenden Attributwerte der Prozess-Flusselemente hinzugezogen, um Änderungen im Kontext der Hintergrundfarbe, Rahmenfarbe, Rahmenbreite und Rahmenart vorzunehmen.

Zunächst erfolgt über einen Modellkontext die Übergabe des Paketes, auf welches die grafische Aufbereitung angewandt werden soll. Anschließend werden die Aktivitäten, Ereignisse, Datenobjekte und Verbindungen einschließlich einer vorhandenen Hierarchisierung erfasst. Die Modelloperationen zum Setzen der entsprechenden Modellobjekt-Konfiguration sind für die Flusselemente einzeln aufgeführt. Die vorgenommene grafische Aufbereitung ist nachfolgend exemplarisch dargestellt und kann auf individuelle Bedürfnisse angepasst werden

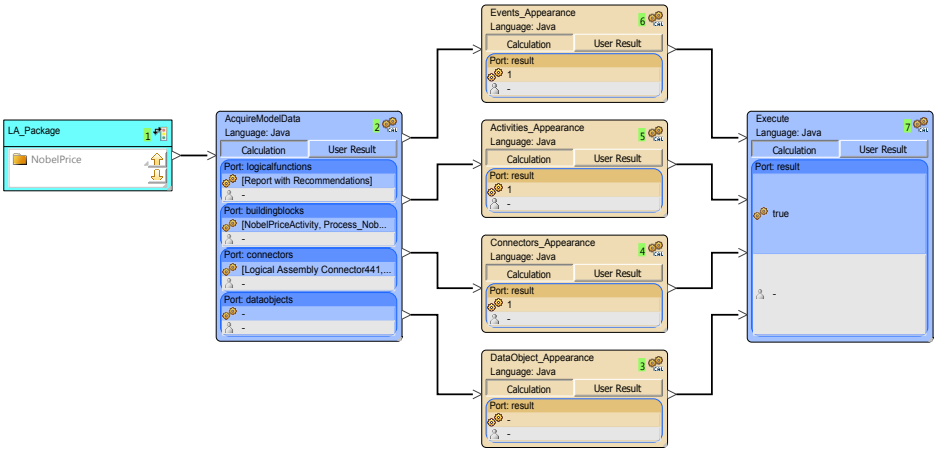


Abb. 4.19: Metrik zur grafischen Aufbereitung von Prozess-Flusselementen

Im Kontext von Ereignissen wird basierend auf dem Attribut *eventType* eine Aufbereitung hinsichtlich der Literale startEvent, intermediateEvent und endEvent vorgenommen. Verändert werden die <Rahmenfarbe, Rahmenbreite, Rahmenart>. Somit erhalten die startEvents <blau, 3, durchgängig>, intermediateEvents <schwarz, 5, gestrichelt> und endEvents <grün, 7, durchgängig>. Hinsichtlich Datenobjekten wird die Hintergrundfarbe genutzt, um entsprechend ihres Typs über *dataObjectTypeEnum* eine Hervorhebung zu erzielen. So erhält die Einstufung als Spezifikation die Hintergrundfarbe grün, der Plan die Farbe gelb, der Report wird durch orange gekennzeichnet und Other durch hellblau. Optionale Verbindungen werden durch Einfärbung in der Farbe lila ersichtlich. Auf Aktivitäten wird vorerst kein Farbschema angewandt, jedoch sind die entsprechenden Vorbereitungen in der Metrik bereits eingepflegt.

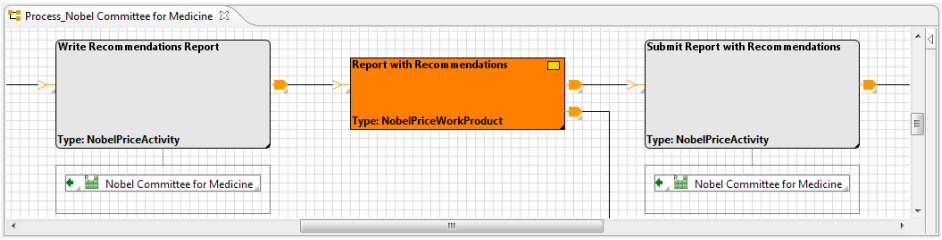


Abb. 4.20: Grafisch aufbereiteter Prozessauszug für das exemplarische Szenario

In Abb. 4.20 ist exemplarisch die Aufbereitung für das Prozessszenario ersichtlich. Durch die Ausführung der Metrik wurde das Datenobjekt „Report with Recommendations“ über die Modellobjekt-Konfiguration entsprechend den Vorgaben automatisch eingefärbt.

4.6 Beispielhafte Anwendung der Prozessmodellierung auf ISO 26262

Der internationale Standard ISO 26262 *„Road vehicles - Functional safety“* fordert Prozesse für den kompletten Produktlebenszyklus von funktional sicheren automobilen EE-Systemen. Der durch das Dokument beschriebene Gesamtprozess, bestehend aus einer Vielzahl an geforderten Teilprozessen, bezieht dahinterstehende Anforderungen von der Konzeptphase bis zur Außerbetriebnahme der Systeme ein. Somit sind neben den klassischen Entwicklungsprozessen die Inhalte von Standards ebenfalls im Fokus einer Prozessmodellierung.

Die geforderten Aktivitäten und Arbeitsprodukte innerhalb des Dokuments ISO 26262 sind nicht nur inhaltlich umfangreich, sondern zudem hochgradig miteinander vernetzt. Somit sind die Abläufe und zu definierenden Schnittstellen nicht direkt offensichtlich und müssen teilweise mühsam identifiziert sowie abgeleitet werden. Zusätzlich sind Referenzen zu beispielsweise externen Qualitätsmanagement-Standards oder allgemein anerkannten Industriequellen gegeben. Unter Berücksichtigung bereits vorliegender firmenspezifischer Vorgehensweisen für die Produktentwicklung, vgl. hierzu auch Kapitel 3.2.1.6, ist eine Verknüpfung mit den geforderten Prozessen hinsichtlich funktionaler Sicherheit äußerst anspruchsvoll. Zudem ist die Anwendung von Prozessen zur funktionalen Sicherheit insbesondere in verteilten Entwicklungen in Bezug auf Beschreibung, Verständnis, Analyse und Planung von Prozessen herausfordernd. Daher wird im Folgenden ein Prozessmodell für ISO 26262 vorgestellt, welches auf den zuvor beschriebenen Konzepten und Implementierungen basiert.

4.6.1 Analyse des strukturellen Dokumentenaufbaus der ISO 26262

Die Erstellung eines Prozessmodells für ISO 26262 erfordert zunächst eine strukturelle Analyse des Standards. Ein erster Überblick hierzu kann aus der grundlegenden Struktur direkt aus dem Inhaltsverzeichnis der einzelnen Parts gewonnen werden. Für eine Prozessbeschreibung ist dies jedoch nicht ausreichend und muss bis auf die Ebene von elementaren Arbeitsschritten und Arbeitsprodukten heruntergebrochen werden.

Die Umfänge des Standards zeigen sich bereits in der Netto-Seitenanzahl für das Gesamtdokument von 390 Seiten[61]. Aus Prozesssicht sind nicht alle Parts des Standards relevant, somit hat eine Aufschlüsselung ausgehend von den Parts zu erfolgen. Die Strukturierung des Gesamtdokumentes erfolgte in insgesamt 10 Parts. Kernelemente sind Part 3 für die Konzeptphase, Part 4 bis Part 6 für die Produktentwicklung auf System-/Hardware-/Softwareebene sowie Part 7 für die Produktion und den Betrieb des Fahrzeuges. Angemerkt sei an dieser Stelle nochmals, dass die Gesamtstruktur auf dem V-Modell als Referenz-Prozessmodell basiert und das übergeordnete „V“ die Verknüpfung zwischen den Parts 3 bis 7 darstellt [ISO26262, 1-V]. Die Parts hinsichtlich

[61] Die Aufsummierung erfolgte über die ausgewiesenen, zu bezahlenden Seiten der einzelnen ISO 26262 Parts.

Entwicklung von Hardware und Software beinhalten wiederum ihre eigenen untergeordneten V-Modelle.

Tabelle 4.1 gibt Aufschluss über quantitativ erhobene Kennzahlen zum Dokument ISO 26262. Die Produktentwicklung auf Hardwareebene ist hierbei von der Seitenanzahl am umfangreichsten und stellt einen beachtlichen Beitrag zur Erfüllung von funktionaler Sicherheit dar. Hinsichtlich der Anzahl an Clauses weist Part 8 bedingt durch die Darstellung übergreifender Thematiken den höchsten Wert auf.

			Core Parts								
	Vocabulary	Management of functional safety	Concept phase	Product development at the system level	Product development at the hardware level	Product development at the software level	Production and operation	Supporting processes	Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses	Guideline on ISO 26262	
Part	1	2	3	4	5	6	7	8	9	10	ISO 26262 overall document
# Pages	23	26	25	36	76	40	11	48	16	89	390 Pages
# Terms and Defintions	142	---	---	---	---	---	---	---	---	---	142 Terms and Definitions
# Clauses	2	7	8	11	10	11	6	14	8	11	88 Clauses
# Sub-Clauses	0	18	23	38	33	38	13	53	23	23	262 Sub-Clauses
# Work products	0	9	8	21	14	27	11	27	5	0	95 Work products[1]
# Annexe	---	5	2	2	6	4	1	2	1	2	25 Annexe
Determination of Sub-Clauses for											
Clause 1	0	0	0	0	0	0	0	0	0	0	[1] only unique work products
Clause 2	0	0	0	0	0	0	0	0	0	0	
Clause 3	---	0	0	0	0	0	0	0	0	0	
Clause 4	---	3	3	3	3	3	3	3	3	3	
Clause 5	---	5	5	5	5	5	5	5	5	3	
Clause 6	---	5	5	5	5	5	5	5	5	5	
Clause 7	---	5	5	5	5	5	---	5	5	0	
Clause 8	---	---	5	5	5	5	---	5	5	3	
Clause 9	---	---	---	5	5	5	---	5	---	2	
Clause 10	---	---	---	5	5	5	---	5	---	4	
Clause 11	---	---	---	5	---	5	---	5	---	3	
Clause 12	---	---	---	---	---	---	---	5	---	---	
Clause 13	---	---	---	---	---	---	---	5	---	---	
Clause 14	---	---	---	---	---	---	---	5	---	---	ISO 26262 overall document
	0	18	23	38	33	38	13	53	23	23	262 Sub-Clauses

Tab. 4.1: Datenblatt zur ISO 26262 als quantitative Erfassung

Bezogen auf eine inhaltliche Untersuchung auf Part-Ebene können folgende Aussagen getroffen werden: Für Part 1 kann kein Prozessmodell erstellt werden, da hier nur fachspezifisches Vokabular beschrieben und Abkürzungen definiert werden. Selbiges gilt prinzipiell auch für Part 2, da innerhalb diesem das Management von funktionaler Sicherheit bezüglich der einzelnen Entwicklungsphasen im Fokus steht. Jedoch wird insbesondere der Bezug zu übergreifenden Arbeitsprodukten im Zuge der Konzept- und Produktentwicklungsphase hergestellt. Durch Part 8 werden Informationen bereitgestellt, welche unterstützenden Prozesse für die Produktentwicklung gefordert sind. Diese werden unter die Aspekte der jeweiligen Phase innerhalb des Sicherheitslebenszyklus gesetzt. Daher werden teilweise keine eingehenden Arbeitsprodukte als Grundvoraussetzung festgehalten, sondern lediglich ein Bezug zu den ausgehenden Arbeitsprodukten hergestellt. Eine Modellierung ist für diesen Part daher nicht vorgesehen, da dies in Richtung übergreifender Fragestellungen wie beispielsweise verteilter Entwicklung, Konfigurations- und Änderungsmanagement läuft. Part 9 beschreibt durch welche Analysen die Produkte abgesichert werden müssen und ist daher nur indirekt im Fokus, da die entsprechenden Sicherheitsevaluationen innerhalb der spezifischen Parts detaillierter aufgegriffen werden. Part 10 ist von informativem Charakter [ISO26262, 10-1]. Somit verbleiben die Parts 3-7 und gegebenenfalls Part 2 und Part 9, auf welche eine Anwendung der Prozessmodellierung erfolgen kann.

4.6.2 Hierarchisierung innerhalb der ISO 26262

Schlüsselt man die ISO 26262 tiefergehend auf und berücksichtigt übergreifende Strukturelemente, so wird ersichtlich, dass insgesamt drei Hierarchieebenen von Nöten sind. Das Dokument ISO 26262 selbst repräsentiert die oberste Ebene (n) und wird für sich alleinstehend im Rahmen der Aktivität als Prozess eingestuft. Die mittlere Ebene (n-1) befindet sich auf der Ebene der Clauses und ermöglicht mit der darüberliegenden Ebene ein erstes Verständnis für den gesamtem Entwicklungsprozess. Die unterste Ebene (n-2) repräsentiert bereits die einzelnen Arbeitsschritte und Arbeitsprodukte.

Der elementare Untersuchungsgegenstand für die Prozessbeschreibung bezieht sich somit auf die einzelnen Clauses. Diese repräsentieren die Schritte innerhalb des V-Modells, sowohl im absteigenden als auch aufsteigenden Zweig des „Vs“. Am offensichtlichsten wird dies innerhalb von ISO 26262 für das „Referenz-Phasenmodell für die Softwareentwicklung“ nach [ISO26262, 6-Fig.2] gezeigt. Innerhalb der relevanten Parts stehen die Clauses ausgehend von Clause 5 im Fokus, da die Vorausgehenden jeweils unter anderem einen allgemeinen Überblick, Referenzen und Anforderungen gegenüber anderen Parts der ISO 26262 auflisten.

Ausgehend von Clause 5 des jeweiligen Parts ergibt die Betrachtung der einzelnen Clauses wiederum, dass innerhalb der Sub-Clauses 3 bis 5 die geforderten und hochgradig vernetzten Aktivitäten und Arbeitsprodukte enthalten sind, siehe Abb. 4.21. Diese gilt es auf die *Activities*, *Events* und *DataObjects* zu überführen. Nicht zuletzt sind auch

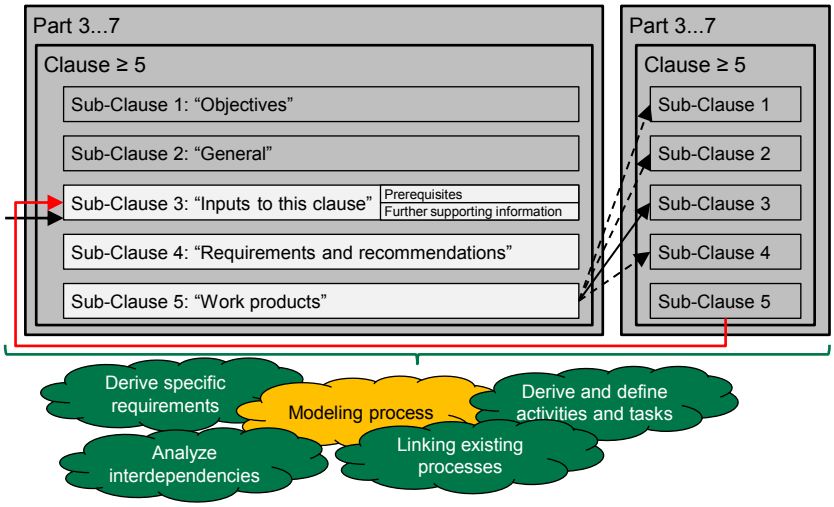


Abb. 4.21: Vereinfachte Dokumentenstruktur der ISO 26262 [Adler2014a]

Anmerkungen zu optionalen Referenzmodellen, weiteren Standards und gegebenenfalls existierenden Arbeitsprodukten vorhanden. Dies kann für die geforderten Prozesse und Arbeitsprodukte von Vorteil sein, um entsprechend direkte Verknüpfungen herzustellen bei gleichzeitiger Aufwandsreduktion unter Einbeziehung vorhandener Dokumente zur Umsetzung eines Sicherheitsnachweises.

4.6.3 Modellierungsalternativen für die Prozessbeschreibung der ISO 26262

Basierend auf dem Metamodell-Vorschlag und der Integration in die EEA-Entwicklungsumgebung werden zwei Modellierungsalternativen für die ISO 26262 als geeignet angesehen. Dies wirkt sich insbesondere auf die grafische Darstellung der Prozessbeschreibungen aus, siehe Abb. 4.22. Alternative 1 nutzt zur Beschreibung die Flusselemente Aktivität und Ereignis, sieht jedoch von der Nutzung der Datenobjekte und Datenspeicher ab. Somit repräsentieren Aktivitäten beispielsweise die einzelnen Clauses der ISO 26262. Die Änderung des Zustandes eines Arbeitsproduktes, beispielsweise eines spezifischen Dokumentes, wird über ein Ereignis dargestellt. Hiermit verknüpft ist die vorausgehende Aktivität, welche durch Initiierung, Änderung oder Finalisierung das Ereignis herbeiführt. Innerhalb der zweiten Alternative wird die Prozessbeschreibung durch die Verwendung von Datenobjekten und Datenspeicher angereichert. Somit repräsentiert die Kombination von Ereignis, als Änderung des Zustandes eines Dokumentes, gemeinsam mit dem Datenobjekt, als Repräsentation für das Dokument, ein Arbeitsprodukt bezogen auf ISO 26262.

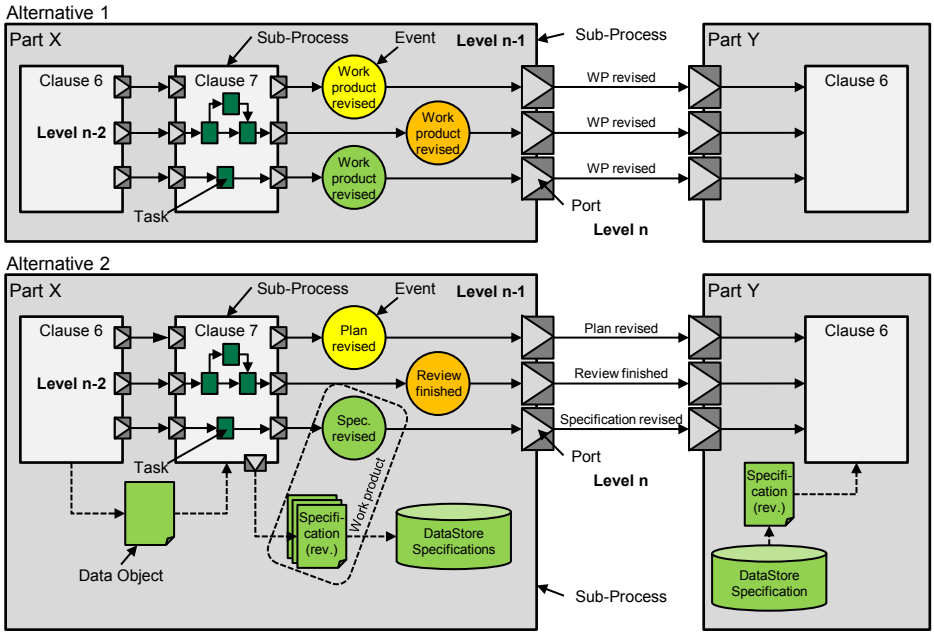


Abb. 4.22: Vorgeschlagene Alternativen für die Prozessmodellierung der ISO 26262

Für beide Modellierungsalternativen wird ein übergeordneter Prozess angelegt, welcher den gesamten Lebenszyklus von funktional sicheren Systemen abdeckt. Innerhalb diesem werden Parts, Clauses sowie gegebenenfalls Sub-Clauses als Aktivitäten instanziiert und als Unterprozesse eingestuft. Innerhalb eines Clauses bzw. Sub-Clauses werden Aktivitäten entsprechend als elementare Arbeitsschritte eingestuft. Die Richtung des Prozessflusses wird durch die Verwendung der entsprechenden Ports festgelegt.

Um den Bezug zum vorgeschlagenen Metamodell herzustellen, wird dies im Folgenden anhand der aufwendigeren zweiten Modellierungsalternative vertieft, da diese sämtliche Aspekte der ersten Modellierungsalternative einschließt. In der zweiten Modellierungsalternative wird die Assoziation von *DataObjects* nach *DataStores* verwendet, um die gespeicherte oder aktualisierte Revision der spezifischen Dokumente zu visualisieren. Jedes Dokument wird repräsentiert durch ein *DataObject*, welches gemäß [ISO19510, 7.3.1] als ein Papierblatt grafisch dargestellt wird. Diese Visualisierung unterstützt den Anwender, um spezifische Dokumente zu identifizieren. Da das *DataObject* nach [ISO19510, 10.4.1] an die Lebenszeit seiner übergeordneten Aktivität gebunden ist, wird die Assoziation zum *DataStore* verwendet, um das Dokument für nachfolgende Prozesse und Sub-Prozesse bereitzustellen.

Hinsichtlich einer besseren Übersichtlichkeit über die verschiedenen Typen von *DataObjects* werden diese nach dem in Kapitel 4.5.5 beschriebenen Prinzip automatisch eingefärbt. Die unterschiedlichen Farben werden für die entsprechenden Attributwerte von *dataObjectType* bereitgestellt. Dies ermöglicht ein gut lesbares Prozessmodell für die ISO 26262 und stellt eine visuelle Gruppierung des gleichen Typs von *DataObjects* bereit.

Jedes *DataObject*, welches ein Dokument oder Arbeitsprodukt im Kontext der ISO 26262 repräsentiert, beinhaltet ein Attribut *state*. Dies ermöglicht die Differenzierung zwischen unterschiedlichen Versionen des gleichen Dokuments. Zum Beispiel wird das Dokument „SafetyPlan“ in zahlreichen Parts von ISO 26262 aufgegriffen, wobei dessen *state* beispielsweise in *refined* oder *final* basierend auf den durchgeführten *Activities* geändert wird.

Die Metamodell-Klasse *Reference*, welche mit einer *Activity* verbunden ist, stellt die Repräsentation für weitere Referenzen bereit. Im Kontext von ISO 26262 sind Referenzen zu beispielsweise einem internen Qualitätsmanagement-System oder anderen Parts der ISO 26262 vorhanden, welche als optional gesehen werden. Diese werden als „further supporting information“ klassifiziert und können durch das Attribut *referenceType* verfeinert beschrieben werden.

Während einer Modellierung der innerhalb der ISO 26262 beschriebenen Prozesse gehen die sukzessive ausgebildeten Querverbindungen zwischen den Parts auf unterschiedlichen Granularitätsebenen hervor. Wenn ein *Event* als Eingang für eine *Activity* in einem anderen Part benötigt wird, müssen neue *Ports* für die Parts als Unterprozesse eingeführt werden. Anschließend können diese *Ports* auf höherer Ebene miteinander verbunden werden.

4.6.4 Prozessmodellierung für ISO 26262

Im Rahmen der exemplarischen Prozessbeschreibung wurde die Darstellungsalternative 1 nach Kapitel 4.6.3 ausgewählt, da die Speicherung von Dokumenten aktuell nicht in der integrierten Entwicklungsumgebung gesehen wird, sondern über Referenzen erfolgt. Somit werden erstellte, überarbeitete oder finalisierte Arbeitsprodukte als Ereignisse angesehen, die einzelnen geforderten Tätigkeiten im Rahmen der ISO 26262 werden auf Aktivitäten abgebildet[62]. Angemerkt sei, dass die spezifischen Attribute der Klasse *DataObject* bedingt durch die Modellierungsalternative 1 auf die Klasse *Event* übertragen wurden.

4.6.4.1 Vorgehensweise der Modellierung

Für die Modellierung der vorgeschriebenen Prozesse, einschließlich aller Aktivitäten und Arbeitsprodukte, wurde innerhalb von PREEvision wie folgt vorgegangen:

[62] Die Erstellung des ISO 26262-Prozessmodells wurde innerhalb der betreuten Arbeit [Schwaer2013] durchgeführt.

Aufbau Typenbibliothek: Vor dem Hintergrund einen direkten sowie eindeutigen Bezug zwischen dem zu erstellenden Prozessmodell und der Namensgebung innerhalb von ISO 26262 zu gewährleisten und damit sowohl den Perspektiven von funktionaler Sicherheit als auch vom Prozess zu genügen, wird eine Typenbibliothek zur Verfügung gestellt. Diese wird aus Sicht der ISO 26262 angelegt und bietet für die Aktivitäten unter anderem die Bibliothekstypen „ISO26262-Part“ sowie „ISO26262-Clause“. Jedes der eindeutigen 95 nach Kapitel 4.6.1 vorkommenden Arbeitsprodukte innerhalb der ISO 26262 erhält wiederum seinen eigenen Bibliothekstyp. Somit ist der Gesichtspunkt einer Wiederverwendung innerhalb der Modellierung gegeben und das mehrmalige Aufgreifen eines Arbeitsproduktes im Zuge der Initiierung, Verfeinerung und Finalisierung kann auf einen bestimmten Typ zurückgeführt werden.

Strukturelle Prozessmodellierung: Für den strukturellen Aufbau des Prozessmodells werden die entsprechenden Bibliothekstypen der Aktivität genutzt, um der identifizierten Hierarchisierung nach Kapitel 4.6.2 zu genügen und die entsprechenden Prozessdiagramme im Zuge eines jeden Unterprozesses anzulegen. Innerhalb der einzelnen Diagramme kann das Prozessmodell unter Verwendung der Bibliothekstypen für die Arbeitsprodukte und dem Anlegen von Ports angereichert und durch die Verbindungen sukzessiv die Querverknüpfungen mit und zwischen den Aktivitäten hergestellt werden. Somit ergeben sich die sequentiellen und parallelen Teilprozesse.

Anreicherung um generische Attribute: Das strukturell erstellte Prozessmodell kann im Folgenden durch die in Kapitel 4.5.2 beschriebene Metrik automatisch um die generischen Attribute nach dem Metamodell-Vorschlag für jedes Artefakte innerhalb des Prozessmodells angereichert werden. Die Anwendung erfolgt hierbei auf dem übergeordnetsten Paket des Prozessmodells.

Eintragung der Attributwerte: Basierend auf den eingebrachten generischen Attributen kann eine Überarbeitung im Zuge einer Verfeinerung des Prozessmodells vorgenommen werden, indem die entsprechenden Attributwerte für die einzelnen Aktivitäten, Arbeitsprodukte sowie Verbindungen aus den entsprechenden Enumerationen ausgewählt werden, vgl. hierzu auch Abb. 4.16. Dies repräsentiert die inhaltlich abgeschlossene Prozessmodellierung für die ISO 26262.

Grafische Aufbereitung des Prozessmodells: Um die Übersichtlichkeit und Lesbarkeit des umfangreichen Prozessmodells zu erleichtern, wird die Metrik zur grafischen Aufbereitung des Prozessmodells nach einem individualisierten Farbschema eingestellt und auf das Prozessmodell angewandt. Somit erfolgt die Einfärbung der Prozess-Flusselemente nach den hinterlegten Attributwerten.

Zuweisung von Ressourcen: Dadurch, dass es sich bei der ISO 26262 um einen Standard handelt und keine direkten Verantwortlichkeiten hervorgehen, wird eine fiktive Organisationsstruktur angelegt und eine exemplarische Zuweisung von Ressourcen auf die Prozesselemente vorgenommen.

Im Folgenden werden Prozessdiagramme aus jeder der drei Hierarchieebenen anhand des modellierten ISO 26262 Prozesses vorgestellt. Kernpunkte, welche durch das Prozessmodell innerhalb des jeweiligen Prozessdiagramms ersichtlich sind, werden herausgearbeitet.

4.6.4.2 Hierarchieebene n

Da sich die ISO 26262, wie in Kapitel 2.6 beschrieben, am V-Modell orientiert, ergibt die Anwendung einer Prozessmodellierung, dass der Prozess durch das Gesamtdokument abgebildet wird. Somit basieren auf oberster Ebene die Unterprozesse auf den einzelnen Parts der ISO 26262. In Abb. 4.23 sind sieben Aktivitäten als Unterprozesse eingestuft ersichtlich. Die Parts 3-7 greifen als Kern die Konzeptphase, die Produktentwicklung auf System-, Hardware- und Softwareebene sowie die Produktion und Betrieb auf. Zusätzlich wurden Part 2 im Kontext des Managements und Part 9 für Analysen mit ihren Bezügen modelliert.

Interessant sind die deutlich ersichtlichen Rückkopplungen der Ergebnisse aus Part 5 für die Produktentwicklung auf Hardwareebene und aus Part 6 für die Produktentwicklung auf Softwareebene in Richtung des Part 4 für die Produktentwicklung auf Systemebene. Betrachtet man das übergeordnete „V“ nach Abb. 2.19 so lässt sich an dieser Stelle der quasi nebenläufige Hardware- und Softwareentwicklungs-Zweig wieder zusammenführen.

Auf dieser Ebene sind keine Arbeitsprodukte ersichtlich, da diese innerhalb der Unterprozesse abgehandelt werden. Jedoch ist über die Schnittstellen zwischen den Aktivitäten in Form der eingehenden und ausgehenden Ports und deren Bezeichnungen deutlich ersichtlich, welche Arbeitsprodukte Part-übergreifend weitergereicht werden. Somit können hierüber die Prozessflüsse in einem ersten Schritt inhaltlich erschlossen werden. Die zwischen den Parts ausgetauschten Arbeitsprodukte können sowohl Voraussetzungen als auch Ausgangsprodukte repräsentieren.

Die Zahlencodes, welche den eigentlichen Bezeichnungen der Ports vorangestellt sind, spiegeln die Referenz auf ISO 26262 wieder und zeigen an, aus welchen entsprechenden Clauses im Zuge der Unterprozesse die darüber fließenden Arbeitsprodukte hervorgehen. Somit kann die Prozessperspektive aus Sicht von funktionaler Sicherheit weiterreichend unterstützt werden. Die Richtungsangabe, abgesehen von dem Konstrukt der eingehenden und ausgehenden Ports, wird durch diese Anreicherung auf sämtlichen Hierarchieebenen detaillierter, indem basierend auf dem jeweiligen Zahlencode durch den Prozessbeteiligten herausgelesen werden kann, ob es sich um ein eingehendes Arbeitsprodukt oder ein im Rahmen des aktuell aufgegriffenen Unterprozesses bearbeitetes Arbeitsprodukt handelt. Die angewendete Namenskonvention ist insbesondere auf den tieferen Hierarchieebenen von deutlichem Mehrwert, da hier die Vernetzung der Aktivitäten und Arbeitsprodukte drastisch zunimmt.

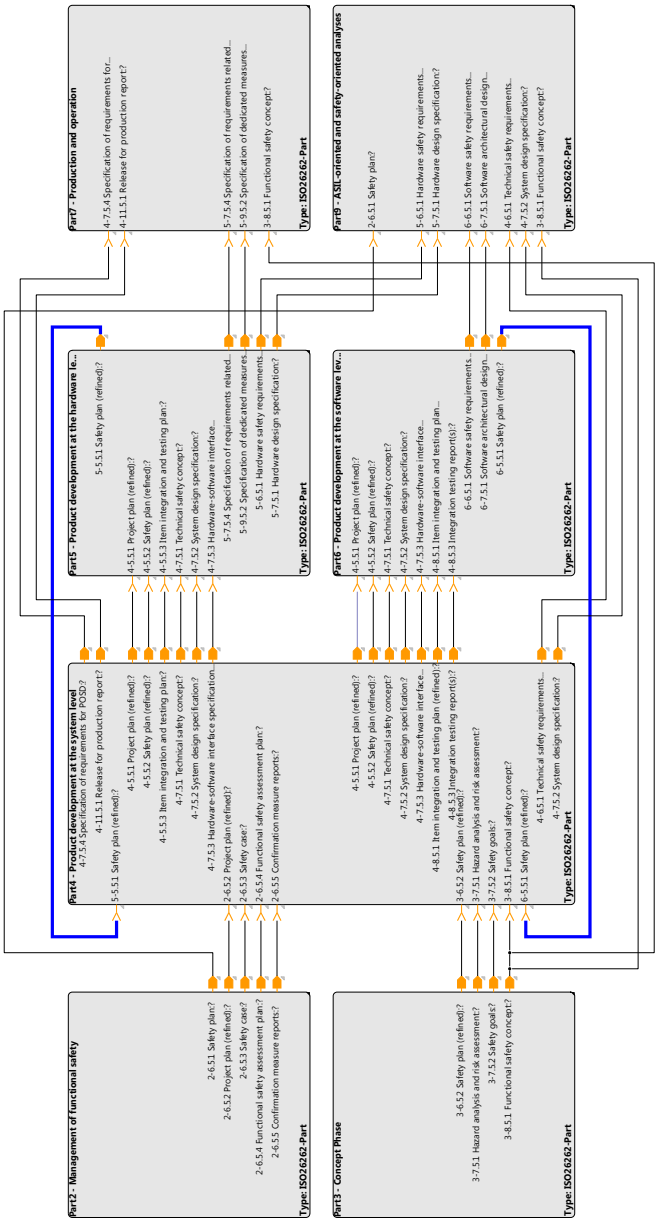


Abb. 4.23: Hierarchieebene n: Auszug des beispielhaften Prozessmodells für ISO 26262 basierend auf [ISO26262]

4.6.4.3 Hierarchieebene n-1

Auf der nächst tieferen Hierarchieebene n-1 können beispielsweise Part 4 und Part 5 betrachtet werden, siehe Abb. 4.24 und 4.25. Aus den Abbildungen wird deutlich ersichtlich, inwiefern die einzelnen Aktivitäten für die Entwicklung funktional sicherer Fahrzeugsysteme einschließlich der Hardware miteinander vernetzt sind.

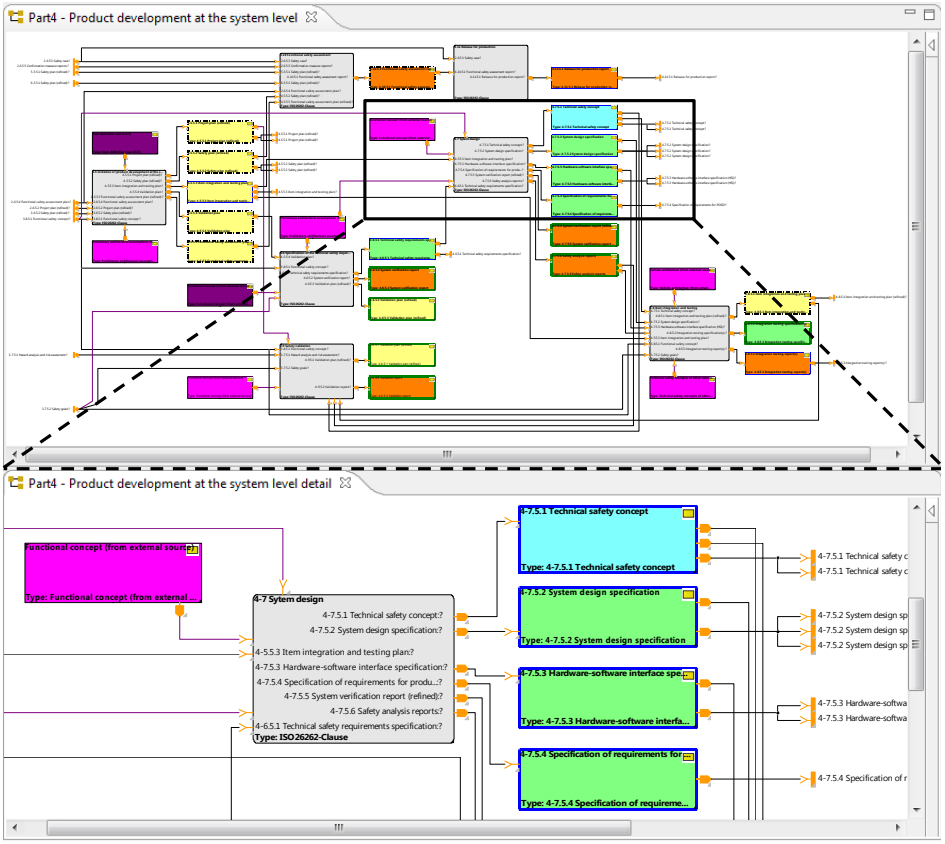


Abb. 4.24: Hierarchieebene n-1: Beispielhafte Prozessmodellierung für Part 4 „*Product development at the system level*“ basierend auf [ISO26262, 4]

Durch die Anwendung der Metrik zur grafischen Aufbereitung der Prozessobjekte sind jedoch bereits die geforderten Pläne, Spezifikationen, Reports sowie Analysen voneinander abgehoben. Hierdurch können die einzelnen Arbeitsprodukte unterschieden werden. Zudem ist der geforderte Status der Ereignisse über die Rahmenart ersichtlich und es

werden somit beispielsweise erarbeitete, überarbeitete oder finalisierte Arbeitsprodukte im Sinne der ISO 26262 gezeigt.

Die optionale Verwendung von weiteren Arbeitsprodukten, im Zuge der ISO 26262 unter „Further supporting information“ gelistet, kann sowohl durch die entsprechende gesonderte Einfärbung der Arbeitsprodukte als auch der Verbindungen erkannt werden. Exemplarisch ist dies bei der Systementwicklung nach [ISO26262, 4-7] für das Einbeziehen eines funktionalen Konzeptes aus einer externen Quelle ersichtlich, siehe Abb. 4.24.

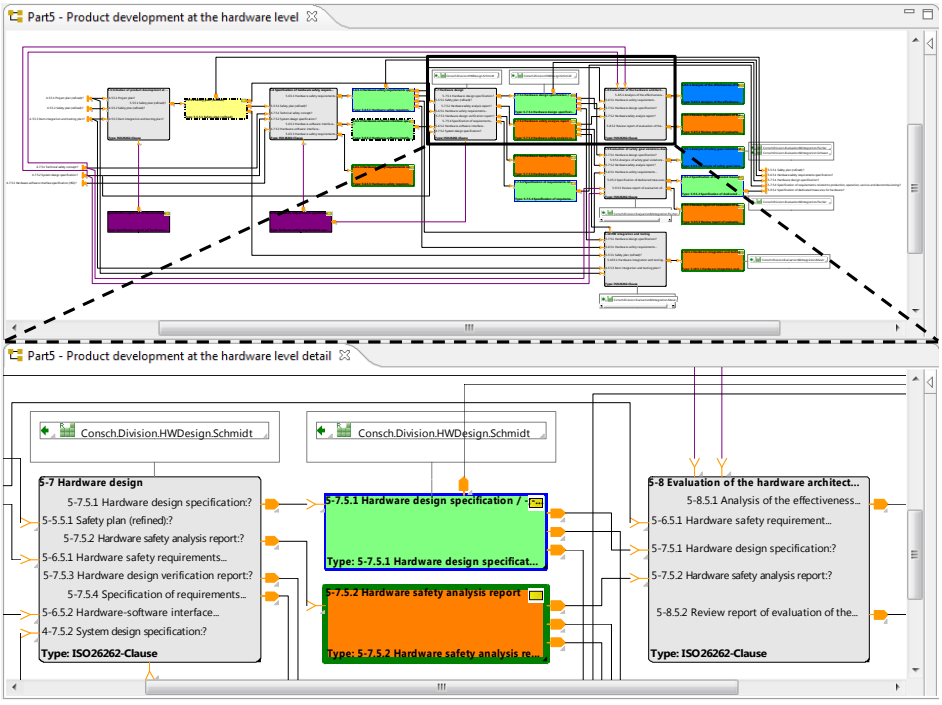


Abb. 4.25: Hierarchieebene n-1: Beispielhafte Prozessmodellierung für Part 5 „*Product development at the hardware level*“ basierend auf [ISO26262, 5] einschließlich exemplarischer Ressourcenzuweisung

Exemplarisch wurde auf dieser Hierarchieebene die Ressource einer Rolle „Consch.Division.HWDesign.Schmidt“ der Aktivität „Hardware design“ zugewiesen, siehe Abb. 4.25. Des Weiteren zeigt das Diagramm weitere Unterprozesse in Form von Clauses. Im Folgenden wird auf Clause 8 als einer dieser eingegangen.

4.6.4.4 Hierarchieebene n-2

Abb. 4.26 zeigt exemplarisch die unterste Hierarchieebene n-2 für Clause 8 innerhalb von Part 5, welcher sich mit den Hardware-Architekturmetriken beschäftigt. Für diese detaillierte Prozessbeschreibung musste teilweise bereits der Fließtext interpretiert werden, um zum einen Aktivitäten im Zuge von elementaren Arbeitsschritten abzuleiten sowie zu definieren und zum anderen den Bezug zu benötigten Daten zur Bearbeitung der Aktivitäten herzustellen. Ferner wurde dies in den übergeordneten Prozess eingebettet, um somit eine durchgängige Verknüpfung der Arbeitsprodukte von der untersten bis zur höchsten Hierarchieebene zu erzielen. Da es sich bei den dargestellten Prozesselementen um Ableitungen sowie Interpretationen aus dem Fließtext von ISO 26262 handelt, wurde ein gesondertes Farbschema angewandt, um somit eine klare Hervorhebung zu den eindeutig beschriebenen Prozessen darzustellen.

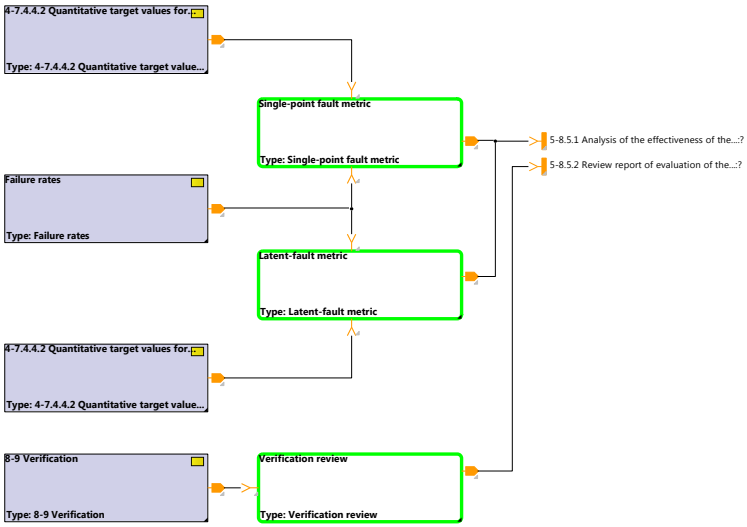


Abb. 4.26: Hierarchieebene n-2: Beispielhafte Prozessmodellierung für Clause 8 „*Evaluation of the hardware architectural metrics*“ basierend auf [ISO26262, 5-8]

4.6.4.5 Hinterlegung von Detailbeschreibungen für Aktivitäten und Arbeitsprodukte

Durch die Prozessbeschreibung in der modellbasierten Entwicklungsumgebung ergibt sich der Vorteil, dass detaillierte textuelle Beschreibungen zu einzelnen Aktivitäten hinterlegt werden können, welche von den Prozessbeteiligten jederzeit eingesehen werden können. Exemplarisch ist die Detailbeschreibung für die Aktivität „Produktentwicklung auf Hardwareebene“ in Abb. 4.27 ersichtlich. Verdeutlicht wird dies dadurch, dass

zusätzlich Informationen aus beispielsweise dem übergreifenden Part 2 für das Sicherheitsmanagement hinsichtlich der jeweiligen Produktentwicklungsphase der Aktivität zugeordnet werden können. Diese Informationen können während einer Bearbeitung der entsprechenden Aktivität eingeholt und individuell erweitert werden.

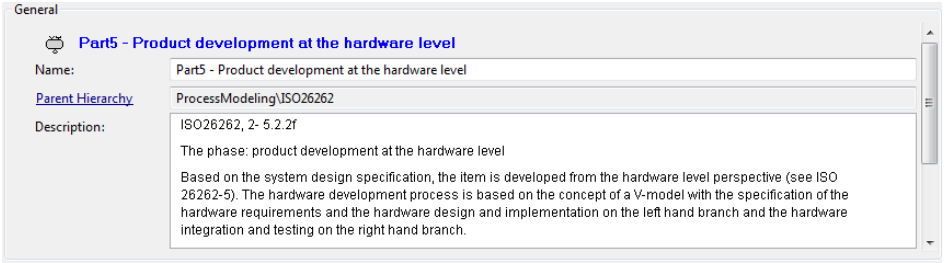


Abb. 4.27: Detailbeschreibung für die Aktivität „Produktentwicklung auf Hardwareebene“

Weiterhin können detaillierte Informationen für die Beschreibungen der einzelnen Arbeitsprodukte innerhalb des Modells eingebracht werden. Dies dient exemplarisch einer Auflistung einzelner Umfänge von Arbeitsprodukten. In Abb. 4.28 sind auszugsweise die aufgeschlüsselten Inhalte für den Sicherheitsplan nach [ISO26262, 2-6.4.3.5] dargestellt.

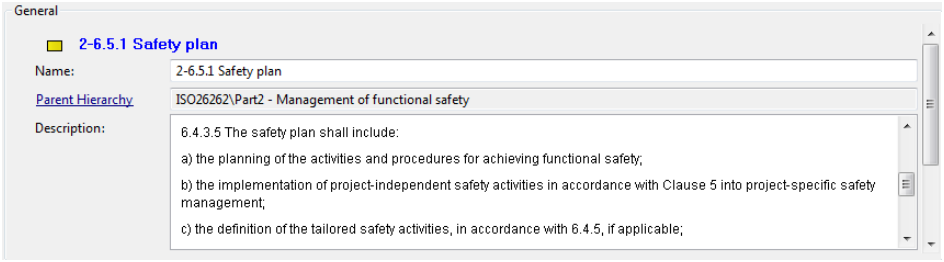


Abb. 4.28: Detailbeschreibung für das Arbeitsprodukt „Sicherheitsplan“

Eine Ausweitung der Beschreibungen auf einzelne Verbindungen innerhalb des Prozessflusses zwischen den Aktivitäten bzw. Arbeitsprodukten sowie das Einbringen von Projekterfahrungen ist gleichermaßen möglich.

4.6.5 Bezug zwischen Prozessmodell für ISO 26262 und dem EEA-Datenmodell

Wird das erstellte Prozessmodell für ISO 26262 dem EEA-Datenmodell von PREEvision gegenübergestellt, so kann ein Bezug zwischen der Prozessbeschreibung und den einzelnen Abstraktionsebenen sowie Datenelementen hergestellt werden. Dies lässt sich unter anderem aus der inhaltlichen Aufteilung der Beschreibungen in ISO 26262 und der Struktur innerhalb der einzelnen Parts begründen. In Abb. 4.29 ist der Bezug abstrahiert dargestellt.

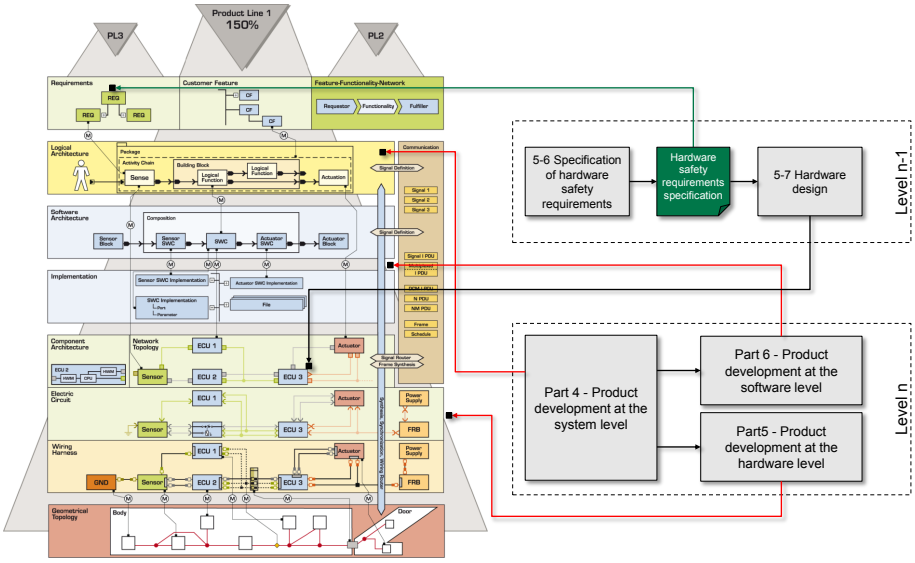


Abb. 4.29: Exemplarischer Bezug zwischen ISO 26262-Prozessbeschreibung und Abstraktionsebenen sowie Modellelementen von PREEvision

Deutlich wird, dass einzelne Parts der ISO 26262 sich gezielt mit den einzelnen Abstraktionsebenen in PREEvision verknüpfen lassen, vgl. zudem [81]. So stehen große Anteile von Part 3 für die Konzeptphase und Part 4 für die Produktentwicklung auf Systemebene in Bezug zur *Logical Architecture*. Weiterhin sind die Verfeinerungen in der Produktentwicklung auf Hardwareebene nach Part 5 und auf Softwareebene nach Part 6 mit den Abstraktionsebenen *Hardware Architecture* und *Software Architecture* zu vereinigen. Die weiteren Parts der ISO 26262 sind orthogonal zu den Abstraktionsebenen anzusehen.

Spezifische Sicherheitsanforderungen werden innerhalb der einzelnen Parts eingeführt und jeweils in den untergeordneten Clauses verfeinert bzw. spezialisiert. Diese lassen sich

übergreifend in der Anforderungsebene bündeln. Hinsichtlich der Nachvollziehbarkeit und Nachverfolgbarkeit kann über das Konstrukt der Mappings ein Bezug hergestellt werden.

Unterprozesse, wie exemplarisch das in Part 5 beschriebene *Hardwaredesign*, lassen sich einzelnen EEA-Datenelementen der jeweiligen Abstraktionsebenen in PREEvision zuordnen. Hierbei können jedoch nur Anteile innerhalb der Prozessbeschreibung aufgegriffen und angebunden werden, welche im Fokus der Architekturbeschreibungssprache stehen.

Die innerhalb der ISO 26262 geforderten Sicherheitsevaluationen können nicht durch Abstraktionsebenen oder darin anzusiedelnde EEA-Datenelemente repräsentiert werden, sondern sind als Implementierungen umzusetzen, welche Informationen aus dem übergreifenden Datenmodell beziehen. Jedoch kann hinsichtlich ihrer Anwendung eine Zuordnung auf die entsprechenden Datenelemente erfolgen.

Die zahlreichen geforderten Arbeitsprodukte können aus den hinterlegten Informationen der einzelnen Ebenen manuell oder rechnergestützt erarbeitet werden. Aus Sicht der Erbringung von Nachweisdokumenten kann dies durch den Bereich der Systemdokumentation innerhalb von PREEvision unterstützt werden.

4.6.6 Unterstützung des Prozessmanagements im Kontext von ISO 26262

Für eine Unterstützung des Prozessmanagements werden das eigentliche Prozessmodell, die hinterlegte Organisationsstruktur und das EEA-Datenmodell aufgegriffen. Im Zuge dessen werden zahlreiche Analyse- und Planungsaspekte für das Prozessmodell unter anderem durch das in Kapitel 4.4 zweckentfremdete Variantenmanagement bereitgestellt, vgl. hierzu auch [Schwaer2013]. Weiterhin können zeitliche Aspekte im Kontext einer Prozessanalyse und -planung berücksichtigt und ein erster Bezug sowohl zum Monitoring von Prozessen als auch einer Prozessausführung hergestellt werden.

Angemerkt sei, dass im Zuge der integrierten Prozessmodellierung und dem aufbauenden Prozessmanagement das Aufsetzen spezifischer Berichtvorlagen innerhalb von PREEvision denkbar ist, um den Prozessbeteiligten entsprechende Dokumentationen bereitzustellen. Diese könnten unter anderem Auflistungen zu Prozessverantwortlichkeiten, aber auch die im Folgenden behandelten Sichtweisen auf den Prozess beinhalten.

4.6.6.1 Strukturierung innerhalb des Variantenmanagements

Um ein Prozessmanagement im Zuge der geforderten Aktivitäten und Arbeitsprodukte sowie der Zuweisung von Ressourcen zu unterstützen, wurde das erstellte Prozessmodell für die innerhalb der ISO 26262 beschriebenen Prozesse verwendet und mit den Variantenmanagement-Konzepten innerhalb von PREEvision verknüpft. Im Kontext der EEA-Entwicklung dienen diese der Definition von Modellvarianten im Zuge unterschiedlicher Produkte und Produktlinien. Hierfür ermöglichen sie zum einen die Zuordnung

der instanziierten Modellobjekte in bestimmte Container und zum anderen eine darauf basierende spezielle grafische Darstellung der Modelle in den angelegten Diagrammen sowie im Modellbaum. In Hinsicht auf eine grafische Visualisierung, bezüglich dem Verblassen von Prozesselementen, welche nicht im aktuellen Fokus sind, kann dies bestens im Rahmen eines ersten Prozessmanagements eingesetzt werden. Basierend auf unterschiedlichen Varianten-Konfigurationen können im Umkehrschluss die spezifischen Prozesselemente im aktuellen Untersuchungskontext hervorgehoben werden.

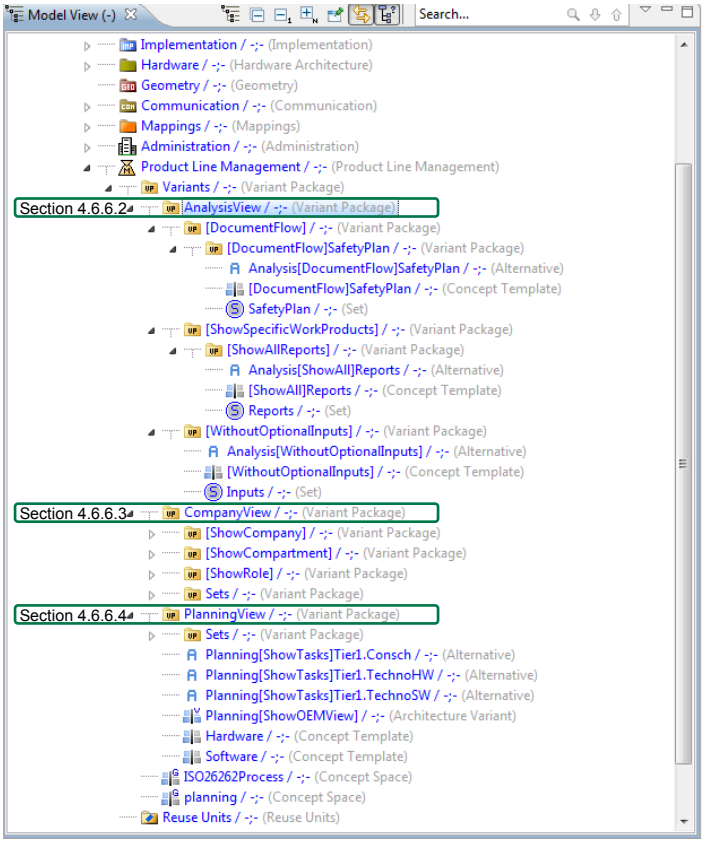


Abb. 4.30: Erstellung der Konzepträume und Strukturierung im Modellbaum

Es werden insgesamt drei Arten von unterschiedlichen Sichtweisen auf den Prozess geboten. Hierbei handelt es sich um die „AnalysisView“, die „CompanyView“, welche eine spezifische Analysesicht unter dem Gesichtspunkt der Ressourcenzuweisung darstellt und daher gesondert gelistet wird, sowie die „PlanningView“, wie in Abb. 4.30 ersichtlich.

Zur Strukturierung wird eine Trennung durch sogenannte Variantenpakete vorgenommen. Innerhalb dieser werden wiederum thematisch abgeschlossene Bereiche durch Unterpakete gebündelt, in welche die einzelnen *Alternatives* eingeordnet werden. Die einzelnen *Sets* werden entweder innerhalb der Unterpakete verankert, wenn diese spezifisch für den jeweiligen Analyse- oder Planungsaspekt sind, anderenfalls werden diese in einem gesonderten Unterpaket aus Gründen der Wiederverwendung bereitgestellt.

Die Sichtweise „AnalysisView“ basiert hierbei ausschließlich auf dem eigentlichen Prozessmodell, die „CompanyView“ und die „PlanningView“ schließen die Organisationsstrukturen zusätzlich mit ein. Da die Prozessplanung nach dem in Kapitel 4.4 beschriebenen Konzept nicht auf dem gleichen Konzeptraum wie die Prozessanalyse aufgespannt werden kann, wird als Ergänzung zum Konzeptraum „ISO26262Process“ ein weiterer Konzeptraum namens „Planning“ zur Verfügung gestellt.

4.6.6.2 Sichtweise „AnalysisView“

Die Sichtweise „AnalysisView“ ermöglicht über eine individuelle Gruppierung von thematisch zusammengehörigen Aktivitäten bzw. Ereignissen sowie deren Verbindungen allgemeine ressourcenunabhängige Analysen und Abfragen auf dem Prozessmodell. Hierbei können gezielt bestimmte Bereiche im Prozessmodell analysiert werden, ohne den Zusammenhang zum Gesamtprozess zu verlieren. Dies kann insbesondere bei der täglichen Arbeit mit dem Prozessmodell von Interesse sein, um beispielsweise die geforderte Erstellung bestimmter Nachweisdokumente innerhalb umfangreicher Prozesse zu identifizieren. So können exemplarisch alle als Bericht eingestuften Arbeitsprodukte durch eine Hervorhebung kenntlich gemacht werden. Dies lässt sich entsprechend sowohl auf weitere Kategorien als auch Prozesselement-Typen anwenden und bietet den Prozessbeteiligten eine aufbereitete Übersicht über die einzelnen verschiedenartigen geforderten Arbeitsprodukte.

Eine weitere Darstellungsmöglichkeit ist, den Dokumentenfluss beispielsweise eines spezifischen Arbeitsproduktes durch den Prozess hervorzuheben. In Abb. 4.31 ist dies anhand des Dokuments für den Sicherheitsplan exemplarisch auf der obersten Hierarchieebene n dargestellt.

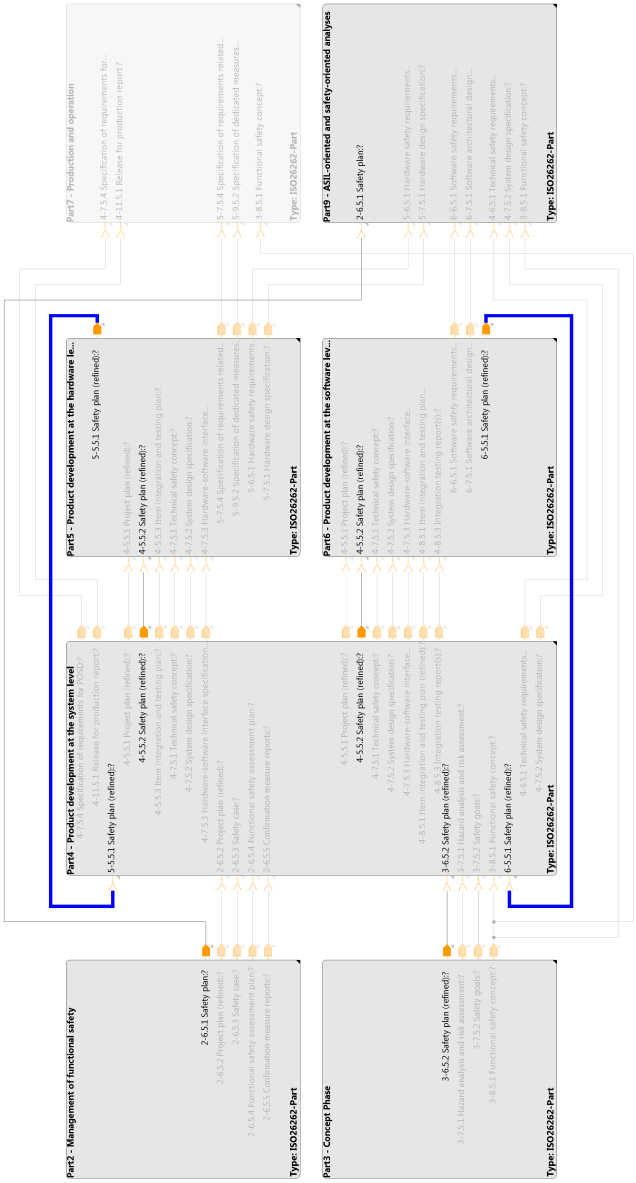


Abb. 4.31: Dokumentenfluss für den Sicherheitsplan auf Hierarchieebene n

Da die Anwendung über die Hierarchieebenen hinweg erfolgt, ist in Abb. 4.32 der Fluss für das gleiche Dokument in der detaillierteren Ebene n-1 für Part 5 ersichtlich. Übergreifend werden sämtliche aktuell nicht-relevanten Prozesselemente verblasst. Hierdurch wird für die Prozessbeteiligten nicht nur der eigentliche Fluss durch den Prozess ersichtlich und nachvollziehbar, sondern es können insbesondere auch die Änderungen des Dokumentenstandes für das Arbeitsprodukt, bedingt durch die dazugehörigen Aktivitäten, herausgestellt werden. Somit wird eine zielgerichtete Erarbeitung der geforderten Arbeitsprodukte unterstützt. Weiterhin ist ersichtlich, an welche nachfolgenden Prozessschritte die Übergabe des Arbeitproduktes zu erfolgen hat und welcher Prozessbeteiligte hierfür die Verantwortlichkeit trägt. Aus Perspektive eines Prozessverantwortlichen kann wiederum betrachtet werden, wer die Verantwortung für ein im Rahmen seiner Aktivität erforderliches und eingehendes Arbeitsprodukt zugewiesen hat.

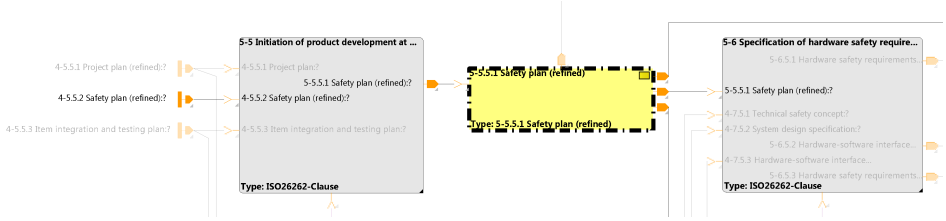


Abb. 4.32: Dokumentenfluss für den Sicherheitsplan auf Hierarchieebene n-1 innerhalb von Part 5

Neben dem Hervorheben von Prozessinformationen können im Umkehrschluss beispielsweise anhand hinterlegter Merkmale in Bezug auf optionale Voraussetzungen alle nicht zwangsweise erforderlichen Aktivitäten und Arbeitsprodukte ausgeblendet werden, um somit einer Übersichtlichkeit hinsichtlich wesentlicher Prozessinformationen zu genügen.

Da neue Fahrzeuge meist auf Vorgängerserien basieren, ist im Kontext der ISO 26262 ein wichtiger Analyseaspekt, an welchen Prozessschritten unter anderem Arbeitsprodukte aus Vorgängerserien eingespeist werden dürfen. Somit können gegebenenfalls Aufwände für einzelne Prozessschritte reduziert werden oder bestenfalls geforderte Aktivitäten entfallen.

Durch Verwendung des Variantenmanagements zur Abbildung von Modellvarianten konnten im Zuge der Prozessanalyse, wie in diesem Kapitel beschrieben, eine Vielzahl an weiteren Perspektiven geboten werden. Diese sind im Rahmen der betreuten Arbeit [Schwaer2013] exemplarisch umgesetzt.

4.6.6.3 Sichtweise „CompanyView“

Verglichen zur vorherigen Sichtweise greift die „CompanyView“ neben dem Prozessmodell zusätzlich Ressourcenzuweisungen auf und stellt Analysen im Zuge der Verantwortlichkeiten gegenüber den Prozessaktivitäten und Arbeitsprodukten bereit. Dies bietet Analysemöglichkeiten hinsichtlich unternehmensabhängiger Verantwortlichkeiten über Zuständigkeiten einzelner Firmen, Fachbereiche und Abteilungen bis herunter auf Mitarbeiter. Neben der Überprüfung auf Abdeckung der Prozesse lassen sich ausgehend von den einzelnen in der Organisationsstruktur hinterlegten Hierarchieebenen gezielt die Prozesselemente im Kontext der jeweiligen Verantwortlichkeiten darstellen.

Exemplarisch wird dies in Abb. 4.33 auf Ebene einer einzelnen Rolle veranschaulicht. Die Verantwortlichkeiten gegenüber den Prozessaktivitäten sind hierbei über ein *Set* gebündelt und werden innerhalb des Prozessmodells hervorgehoben. Somit erhält der Prozessverantwortliche eine eindeutige Darstellung seiner definierten Tätigkeiten innerhalb des Prozessmodells.

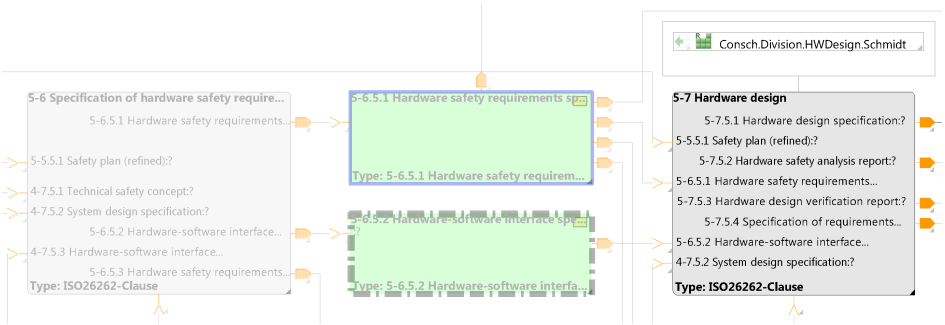


Abb. 4.33: Hervorhebung der Zuständigkeiten im Kontext einzelner Rollen

Durch die *Alternatives* können mehrere *Sets* im Zuge einzelner Rollen zu Abteilungen, Fachbereichen und Unternehmen gebündelt werden. Hierdurch kann eine Art Filter auf die Prozessmodelle gelegt werden, welcher jeweils die Teilprozesse von ausgewählten Prozessverantwortlichkeiten hervorhebt und entsprechend die nicht betroffenen Prozesselemente verblasst.

Diese Art der Darstellung kann aus Auftraggebersicht zudem dafür verwendet werden, sich das Tätigkeitsspektrum eines spezifischen bzw. potentiellen Auftragnehmers darzustellen. Weiterhin kann neben der Analyse abgedeckter Prozesse durch einzelne Auftragnehmer eine Kombination mehrerer Auftragnehmer im Kontext unterschiedlicher Fachgebiete durch eine gemeinsame Hervorhebung der entsprechenden Teilprozesse dargestellt werden. Gleichermaßen kann sich der Auftraggeber eine konträre Aufbereitung des Prozessmodells darstellen lassen, um die Teilprozesse zu identifizieren, welche weiterhin in seiner Verantwortung liegen.

4.6.6.4 Sichtweise „PlanningView“

Um eine nahtlose Umsetzung von Gesamt- bzw. Teilprozessen zu planen und die entsprechende Abdeckung des Prozesses abzusichern, wird ein zweiter Konzeptraum „Planning“ innerhalb des Variantenmanagements aufgespannt. Dieser umfasst die für eine Planung im Fokus stehenden Prozesselemente und ermöglicht eine Gegenüberstellung mehrerer Alternativen im Zuge der Vergabe von Verantwortlichkeiten.

Innerhalb Abb. 4.34 wurde dies exemplarisch aus Sicht des OEMs angewandt. Hinsichtlich der Beauftragung einer externen Firma für die Produktentwicklung der Hardware, unter dem Aspekt der funktionalen Sicherheit nach ISO 26262, kann der OEM zwischen möglichen Alternativen auswählen. Hier beschränkt sich dies auf die zwei dargestellten fiktiven Firmen „Consch“ und „TechnoHW“. Hinter den Alternativen sind die jeweiligen Prozessschritte und damit die Aktivitäten sowie Arbeitsprodukte hinterlegt, welche durch die Firmen umgesetzt werden können. Dies erfolgt über die Zuweisung der *Sets* auf die jeweiligen *Alternatives*, welche sich unter dem Template „Hardware“ befinden. Gleichermaßen kann dies auf weitere thematische Gebiete ausgeweitet werden, exemplarisch dargestellt durch das Template „Software“.

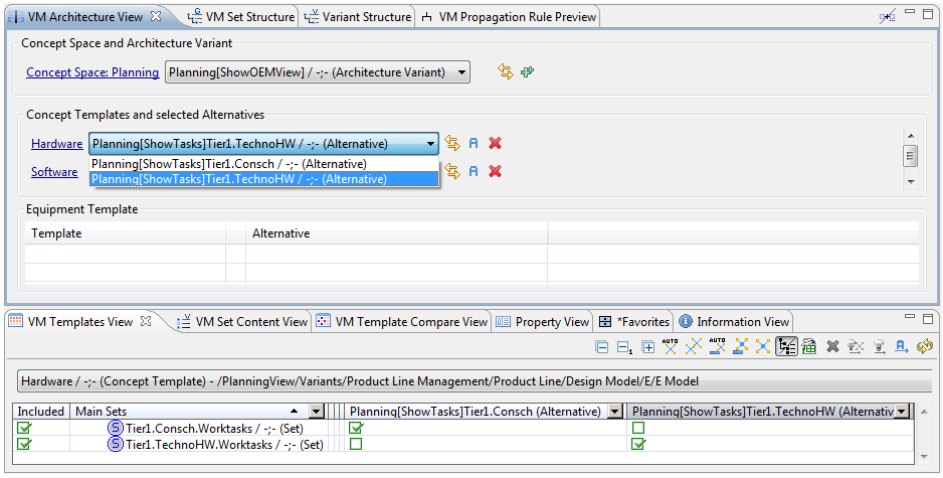


Abb. 4.34: Auswahl zwischen unterschiedlichen Ressourcen für die Durchführung von Teilprozessen basierend auf Alternativen

Die Prozessplanung zeichnet sich dadurch aus, dass sich je nach ausgewählter Alternative über die Themengebiete hinweg die Darstellung der abgedeckten Teilprozesse aus der Kombination der ausgewählten Alternativen ergibt. Gleichzeitig kann ein Vergleich der Lieferumfänge im Zuge des Prozesses für verschiedene Zulieferfirmen vorgenommen werden.

Dieses Konzept einschließlich der Umsetzung ist insbesondere für verteilte Entwicklungen bedeutend, da hierdurch eindeutige Schnittstellen definiert und kommuniziert werden können. Die Abdeckung der Prozesse im Kontext der Ressourcenzuweisung wird direkt über die grafische Darstellung des Prozessmodells ersichtlich.

Da das Konzept für das Prozessmanagement mit dem innerhalb von PREEvision zur Verfügung gestellten Variantenmanagement prototypisch verschmolzen werden konnte, kann die Prozessanalyse und -planung nicht nur hinsichtlich der Prozesse erfolgen, sondern auch auf die Elemente aus dem EEA-Datenmodell ausgeweitet werden, was im ursprünglichen Fokus des Variantenmanagements stand. Dies ermöglicht zusätzliche Sichtweisen auf den Prozess in Kombination mit dem Datenmodell. Exemplarisch kann über den Bezug zwischen Teilprozessen und Datenelementen eine Hervorhebung im Zuge der relevanten umzusetzenden Teilprozesse gegeben werden. Nicht zuletzt kann über die Verknüpfungen ggf. die Relevanz in Bezug auf die Produktentwicklung für eine mögliche Priorisierung der Teilprozesse erschlossen werden.

4.6.6.5 Analyse und Planung zeitlicher Aspekte

Sowohl die Berechnung von zeitlichen Aspekten in Hinsicht auf die frühesten und spätesten Start- sowie Endzeitpunkte als auch die Durchführung einer CPM konnten als prototypisches PlugIn für PREEvision bereitgestellt werden, siehe Abb. 4.35. Durch die Festlegung eines spezifischen Start- und Endereignisses sowie des Start- oder Endtermins für den zu betrachtenden Prozess können über eine Vorwärts-/Rückwärtsterminierung die entsprechenden Zeitpunkte für die Aktivitäten und die Ereignisse berechnet werden. Zusätzlich wird die CPM angewandt, um somit den zeitkritischen Pfad durch den Prozess aufzuzeigen.

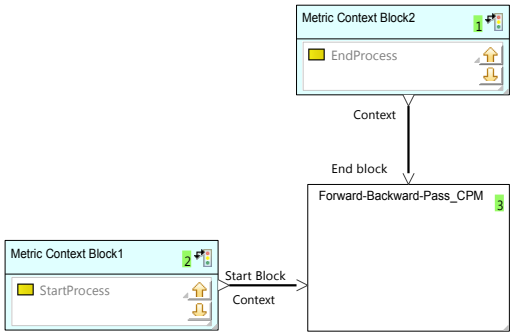


Abb. 4.35: Metrik zur Berechnung von zeitlichen Aspekten

Abb. 4.36 zeigt einen Screenshot mit einer fiktiven Prozessbeschreibung einschließlich einer Hierarchie in Form von Unterprozessen. Für die Aktivität „Activity2-5“ sind in der

Eigenschaftsansicht die berechneten frühesten und spätesten Start- sowie Endzeitpunkte ersichtlich. Zudem ist für diese Aktivität durch die Anwendung der CPM das Attribut *criticalPath* auf *true* gesetzt worden. Basierend darauf konnte der kritische Pfad durch den Prozess für die untere Hierarchieebene rot eingefärbt werden. Dass diese Aktivität im kritischen Pfad enthalten ist, lässt sich unter anderem an den Werten für die jeweiligen frühesten und spätestens Start- und Endzeitpunkte erkennen, welche in diesem Fall jeweils übereinstimmen.

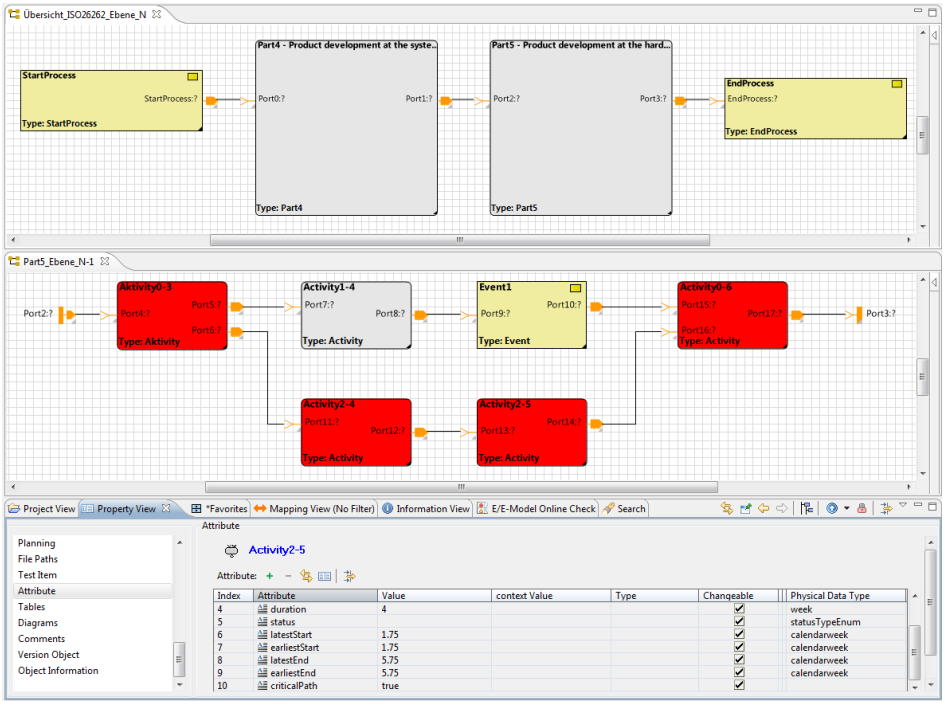


Abb. 4.36: Screenshot: Berechnung zeitlicher Aspekte und Hervorhebung des kritischen Pfades

4.6.6.6 Monitoring der Prozesse

Weitergehend kann durch die Verwendung der Konstrukte des Variantenmanagements nach den vorangegangenen Beschreibungen eine erste Anwendung für ein Monitoring der Prozesse erfolgen. Hierzu wird auf das eingeführte Attribut *status* der jeweiligen Aktivitäten sowie Ereignisse zurückgegriffen und die abgearbeiteten Prozess-Flusselemente einem speziellen *Set* zugeordnet. Eine grafische Aufbereitung des aktuellen Prozessfortschritts erfolgt analog durch Hervorhebung über das Prozessmanagement.

Dies ist prinzipiell auch über die Anwendung einer automatischen Einfärbung der Prozesselemente nach Kapitel 4.5.5 möglich, jedoch würde hierdurch die vorgeschlagene grafische Aufbereitung einzelner Arbeitsprodukte verloren gehen. Eine hierarchieübergreifende Anwendung auf die Aktivitäten kann jedoch durchgeführt werden.

4.6.6.7 Automatisierte Ausführung von Teilprozessen

Durch eine Verknüpfung des Prozessmodells mit einem Framework zur Ausführung von Operationen können ausgewählte Teilprozesse im Zuge der Aktivitäten automatisiert und somit bestimmte Ereignisse in Form der geforderten Arbeitsprodukte herbeigeführt werden. Hierzu ist die in Kapitel 2.5.3.5 dargestellte BPEL als BPMN-Engine denkbar, aber auch die Verwendung vorliegender Rahmenwerke. In modellbasierten Entwicklungsumgebungen nach Kapitel 2.3 ist dies durch Werkzeuglösungen wie zum Beispiel innerhalb von PREEvision durch das Metrik-Framework in einem ersten Schritt realisierbar. Hierzu gilt es, sowohl die automatisiert ausführbaren Aktivitäten als auch generierbaren Arbeitsprodukte zu identifizieren und eine Zuordnung zu einer Maschine zu beschließen, vgl. [51, S.76]. Die auszuführenden Schritte auf der Maschine sind hierbei deutlich detaillierter zu definieren, als die für menschliche Ausführung grobgranular beschriebenen Prozesse, vgl. [51, S.76].

In Bezug auf die Integration einer Prozessmodellierung in eine modellbasierte Entwicklungsumgebung wird hierdurch ein Mehrwert geschaffen, indem einzuhaltende Prozesse für Personen eingebracht aber auch gleichzeitig hinter bestimmten Prozessschritten Automatismen hinterlegt werden können. Dies repräsentiert eine hybride Prozessausführung, allerdings gilt es hierzu weitere Methodiken zu erarbeiten und eine Harmonisierung zwischen den Prozessmodellen und den ausführbaren Einheiten herzustellen.

Betrachtet man speziell das Metrik-Framework von PREEvision, so ist insbesondere die automatisierte Generierung von Reports hinsichtlich bestimmter Dokumentationspflichten durch beispielsweise normative Vorgaben von Interesse. Eine Verknüpfung zwischen Reportgenerierung und dem Metrik-Framework als ausführende Einheit liegt innerhalb des Werkzeugs vor. Durch den Ansatz von Verweisen, welche den Bezug von Prozesselementen zu entsprechenden Metriken herstellen, kann dies in erster Art und Weise für zu erbringende Nachweisdokumente bereitgestellt werden. Dies gilt ebenfalls für Aktivitäten, deren Wirkungsbereiche nicht die Grenzen der integrierten Entwicklungsumgebung überschreiten.

Vorausgreifend wird auf Kapitel 5 verwiesen, in dem detailliert auf die Umsetzung von einzelnen Aktivitäten zur automatisierten Ausführung im Kontext der Hardware-Sicherheitsevaluationen eingegangen wird. Weiterhin wird die Automatisierung für zu erstellende Arbeitsprodukte im Zuge von zugehörigen Berichten dargestellt.

4.7 Ergebnisse und Diskussion

Die Erstellung von Prozessmodellen, vgl. Abb. 4.37, in einer integrierten Entwicklungsumgebung ermöglicht, dass sich alle beteiligten Parteien sowie einzelne Rollen in umfangreichen fachspezifischen Gesamtprozessen neben den firmenspezifischen Prozessen wiederfinden. Zudem wird den Mitarbeitern ein Gefühl der Sicherheit vermittelt, da keine eigenen Mitschriften hinsichtlich des Prozesses notwendig sind, sondern auf eine allzeit verfügbare Wissensdatenbasis zurückgegriffen werden kann, welche nicht nur die eigentlichen Prozessbeschreibungen beinhaltet, sondern auch mit Zusatzinformationen und Best-Practice angereichert werden kann. Des Weiteren können bestimmte Teilprozesse zur gezielten Betrachtung hervorgehoben werden. Dies unterstützt eine Einordnung innerhalb der Zulieferpyramide und stellt wohldefinierte Schnittstellen zur Verfügung.

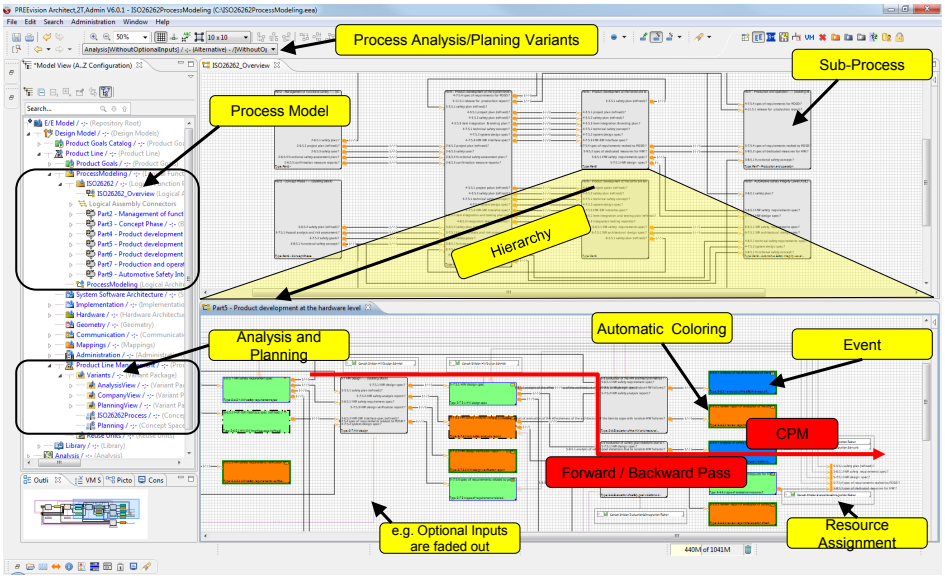


Abb. 4.37: Ergebnisse der Integration einer Prozessmodellierung und eines darauf aufsetzenden Managements in einer EEA-Entwicklungsumgebung

Angelehnt an dem von BPMN geforderten primären Ziel nach einer einfach verständlichen Notation [ISO19510, 1.1] konnte durch die Integration der gleiche Grundsatz in einer domänenspezifischen Entwicklungsumgebung weiter verfolgt und umgesetzt werden. In diesem Zuge musste der grafischen Darstellung der Prozesselemente in zweierlei Hinsicht Genüge getan werden. So konnte der Wiedererkennungswert in Richtung BPMN fast vollständig gewahrt werden, bei gleichzeitig bekannter Darstellung

in der modellbasierten Entwicklungsumgebung. Somit ist eine Lesbarkeit sowohl von Prozessanalysten als auch von technischen Entwicklern gewährleistet.

Verglichen mit anderen Prozessmodellierungsumgebungen bietet die prototypische Integration in eine Entwicklungsumgebung für EEAs einen deutlichen Vorteil, da keine Werkzeugwechsel oder -brüche vorkommen. Zudem stellt die Verknüpfung zwischen EEA-Datenmodell und Prozessmodell einen gravierenden Nutzen gegenüber einer alleinstehenden Prozessbeschreibung dar. Auf dieser soliden Basis können auch aufkommende Herausforderungen, wie eine prozessgetriebene Entwicklung beispielsweise im Kontext von AUTOSAR, aufgesetzt werden.

Insbesondere bei anzuwendenden Standards unterstützt die vorgeschlagene Methodik für die Modellierung und Analyse der Prozesse in einer modellbasierten Entwicklungsumgebung dahingehend, dass der Umgang mit dem Standard erleichtert wird, eine single-source Datenbank aufgebaut wird, relevante Absprachen identifiziert werden können und die beteiligten Rollen sich im Prozess wiederfinden. Daneben können durch die Zuweisung von Ressourcen auch Verantwortlichkeiten abgebildet werden. Weiterhin können die innerhalb des Standards beschriebenen Prozesse mit bekannten Vorgehensweisen verglichen und Abhängigkeiten eingebracht werden. Nicht zuletzt können Inkonsistenzen und Verbesserungspotentiale innerhalb von Teilprozessen oder des gesamtheitlichen Prozesses aufgedeckt werden.

Die Anwendung der beschriebenen Methodik auf ISO 26262 ermöglichte die dargestellte Prozessbeschreibung. Dies dient einem kommunizierbaren Prozessmodell und durch die interaktive Navigation innerhalb des Prozessmodells wird sowohl die Vermittlung als auch das Aneignen von Wissen hinsichtlich funktionaler Sicherheit erleichtert. Im Rahmen der Modellierung und einer nachfolgenden Untersuchung konnten gezielt Aspekte im Kontext der Prozessbeschreibung innerhalb des Dokuments ISO 26262 identifiziert werden. So stellt ISO 26262 einen äußerst strukturierten und in sich konsistenten Standard dar, welcher verglichen mit anderen Standards sehr feingranular aufgebaut ist und an das über Jahre bewährte V-Modell anknüpft. Einer möglichen bevorstehenden Ausweitung des Anwendungsbereiches für die ISO 26262 auf weitere Kraftfahrzeuge gemäß [64, S.35] ist im Kontext der hervorragenden Strukturierung des Standards eine mehr als gute Basis geboten.

Kleinere Unstimmigkeiten, nicht inhaltlicher Natur, konnten innerhalb der unterschiedlichen Hierarchieebenen aufgezeigt werden, welche allerdings bei dem umfangreichen Dokument von der Anzahl her erstaunlich wenig sind: So sind geringfügige Inkonsistenzen durch fehlerhafte Namensgebungen vorhanden, wie beispielsweise der in [ISO26262, 4-10.3.1] aufgegriffene „audit report if available in accordance with ISO 26262-2:2011, 6.5.4;“, welcher auf „6.5.4 Functional safety assessment plan, resulting from 6.4.9“ innerhalb von [ISO26262, 2-6.5.4] verweist. Weiterhin sind unvollständige oder fehlerhafte Referenzen vorhanden, wie beispielsweise das innerhalb von [ISO26262, 5-10.4.2] aufgegriffene Arbeitsprodukt „item integration and testing plan given in ISO 26262-4:2011, 5.5.5“,

welches jedoch unter [ISO26262, 4-5.5.3] gelistet ist. Zudem tauchen beispielsweise innerhalb von Part 8 ungenaue Prozessvoraussetzungen im Zuge der „Prerequisites“ nach ISO 26262 auf, welche allerdings durch die einheitlich gestaltete Dokumentenstruktur bedingt sind. Die Identifikation von fehlerhaften Zusatzbeschreibungen der Arbeitsprodukte im Zuge der Initiierung, Verfeinerung und Finalisierung sind als äußerst wichtig einzustufen. So ist beispielsweise der Zusatz „refined“ beim eingehenden Arbeitsprodukt „item integration and testing plan (refined) in accordance with ISO 26262-4:2011, 5.5.3“, aufgegriffen im Zuge der Voraussetzungen nach [ISO26262, 5-5.3.1] und [ISO26262, 5-10.3.1], nicht korrekt, da dieser erst im Rahmen von Part 4 [ISO26262, 4-5.5.3] initiiert wurde.

ISO 26262 greift in Part 2 das Management von funktionaler Sicherheit auf. Innerhalb diesem sind im Rahmen der geforderten Planung von Sicherheitsaktivitäten nach [ISO26262, 2-6.4.3.6] das Ziel, die Abhängigkeiten zu anderen Aktivitäten oder Informationen, benötigte und verantwortliche Ressourcen für die Durchführung der Aktivitäten, zeitlicher Startpunkt und Dauer sowie die Identifikation der entsprechenden Arbeitsprodukte von Bedeutung. Durch die umgesetzte Prozessbeschreibung für die ISO 26262 kann somit den Anforderungen der geforderten Planung entsprochen werden. Zudem unterstützt es den in [ISO26262, 2-6.4.3.5] genannten „Safety Plan“ beispielsweise hinsichtlich projektspezifischem Sicherheitsmanagement, zugeschnittenen Sicherheitsaktivitäten und dem DIA. Das vorgestellte Prozessmodell bildet nicht nur eine solide Basis um die Sicherheitskultur in einem Unternehmen zu leben, sondern erhöht auch das Potential für Verbesserungen im Bereich der Prozessumsetzung.

Das erstellte Prozessmodell von ISO 26262 kann einen direkten Einfluss auf den Fähigkeitsgrad sowie Reifegrad im Zuge von CMMI nach Kapitel 2.2.4 für CMMI-ACQ, CMMI-DEV als auch CMMI-SVC aufweisen. Insbesondere durch den Aspekt von Part 8 Clause 5 für die Schnittstellen bei verteilten Entwicklungen ist dies im Fokus von CMMI-ACQ und CMMI-DEV. Weiterhin kann durch die integrierte Prozessmodellierung und das darauf aufsetzende Prozessmanagement im Rahmen von CMMI eine Bewertung hinsichtlich Prozessverbesserung unterstützt werden, unter anderem durch die Ressourcenzuweisung und Verknüpfungen in Richtung des EEA-Datenmodells. Gleichermaßen können die unterstützenden Prozesse nach Part 8 im Rahmen von Clause 7 Konfigurationsmanagement, Clause 8 Änderungsmanagement, Clause 9 Verifikation und Clause 10 Dokumentation durch SPICE und CMMI nach [161, S.189] unterstützt werden. Durch die Integration der Prozessbeschreibungen in die EEA-Entwicklungsumgebung wird dies innerhalb einer Werkzeugumgebung zur Verfügung gestellt, da die Prozesse und die aufgelisteten Aspekte miteinander harmonisiert werden können. Durch die vorliegende Prozessbeschreibung der ISO 26262 kann diese auch weiteren automobilen Entwicklungsprozessen und -vorgehensweisen gegenübergestellt werden, welche sich grundlegend unterscheiden können. Erwähnt sei hierzu auch der Beitrag [120], in welchem Automotive SPICE, IEC 61508 und ISO WD 26262 miteinander in Bezug gebracht wurden.

Hinsichtlich der Anwendung auf funktionale Sicherheitsstandards aus anderen Domänen konnte anhand einer ersten Untersuchung der IEC 61508 gezeigt werden, dass sich die zur Verfügung gestellte Methodik aufgrund ihrer generischen Anwendbarkeit und Erweiterbarkeit auch für diesen übergreifenden Standard eignet. Dagegen erweist sich im Vergleich zur ISO 26262 und IEC 61508 eine Prozessbeschreibung für DIN 13849 [DIN13849] aufgrund der geringeren Strukturierung als aufwendiger. Hier gilt es den Fließtext intensiver zu interpretieren, um eine entsprechende Prozessbeschreibung ableiten zu können.

Funktionale Sicherheit erfordert übergreifende Systemkenntnisse sowie ein detailliertes Verständnis im Bereich der Hardware und Software. Gleichzeitig müssen die normativen Anforderungen bei den Beteiligten vertieft sein, um durch eine strukturierte Vorgehensweise den entsprechenden Forderungen nachzukommen. Hierbei unterstützen die Prozessmodellierung und das dazugehörige Management in der EEA-Entwicklungsumgebung, indem eine integrierte Sichtweise auf die unterschiedlichen Perspektiven geboten wird. Die innerhalb von Part 5 beschriebenen Aktivitäten für die Produktentwicklung auf Hardwareebene stehen im Fokus sowohl des Automobilherstellers als auch der Zulieferfirmen für beispielsweise Steuergeräte. Durch die Prozessmodellierung konnte dies bis auf die einzelnen Arbeitsschritte und Arbeitsprodukte herausgearbeitet werden. Durch die feingranulare Darstellung wird ersichtlich, dass insbesondere bestimmte Teilprozesse automatisiert innerhalb von modellbasierten Entwicklungsumgebungen unterstützt werden können.

Betrachtet man die Prozessmodellierung und das -management unter dem Gesichtspunkt, dass im Rahmen des ISO-Standardisierungsprozesses nach Kapitel 2.4 die ISO 26262 in die „Review stage“ übergeht und gleichzeitig auf deutscher Ebene ein eigener Arbeitsausschuss mit aktuell insgesamt fünf Arbeitskreisen gebildet wurde, so lässt sich daraus schließen, dass die ISO 26262 nicht nur fachlich sowie inhaltlich geprüft wird, sondern auch durch gezielte Erweiterungen ausgeweitet wird. Funktionale Sicherheit gewinnt damit weiter an Bedeutung. Die in diesem Kapitel vorgestellte Methodik sowie deren Anwendung auf ISO 26262 bietet eine grundlegende Unterstützung hinsichtlich kommender Herausforderungen, ggf. herbeigeführt durch eine Revision des Standards.

5 Modellbasierte Hardware-Sicherheitsevaluationen

Im vorherigen Kapitel wurde eine Methodik zur Beschreibung von in Standards geforderten Prozessen dargestellt. Konzepte für die Prozessmodellierung, Analyse und Planung innerhalb einer integrierten Entwicklungsumgebung für Elektrik-/Elektronik-Architekturen konnten hierfür bereitgestellt werden. Das in diesem Rahmen erarbeitete Prozessmodell für ISO 26262 ermöglicht einen zielgerichteten Überblick über die zu erfüllenden Aktivitäten und Arbeitsprodukte. Im Kontext der Integration in eine EEA-Entwicklungsumgebung konnte weiterhin eine erste prototypische Verknüpfung mit Architekturelementen hergestellt werden.

Wie in den Auszügen des Prozessmodells in Kapitel 4.6 ersichtlich, sind die Prozesse der ISO 26262 stark miteinander vernetzt. Dies gilt sowohl gesamtheitlich über den abgebildeten Standard hinweg als auch innerhalb der einzelnen Parts. Der Part 5 „*Product development at the hardware level*“ der ISO 26262 beschreibt die Produktentwicklung auf Hardwareebene. Nach [ISO26262, 5-7.2] wird von einem zweistufigen Entwicklungsprozess für das *Hardwaredesign* ausgegangen. Zu unterscheiden sind das *Hardware-Architekturdesign* und das *detaillierte Hardwaredesign*. Während dieser unterschiedlichen Entwurfsstadien müssen nach [ISO26262, 9-8.2] sich ergänzende qualitative und quantitative Sicherheitsanalysen auf angemessener Abstraktionsebene durchgeführt werden. Im Zuge der Absicherung gegenüber zufälligen Ausfällen der Hardware werden die nach Part 5 vorgeschriebenen Hardware-Sicherheitsevaluationen gefordert.

Im Sinne von verteilten Entwicklungen müssen insbesondere Schnittstellen zwischen den Auftraggebern und Auftragnehmern unterstützt werden, um hierdurch einer Entwicklungsschnittstellen-Vereinbarung (DIA) zu genügen. Zudem werden die Evaluationsergebnisse als Argument für den geforderten Sicherheitsnachweis benötigt. Hierfür bieten sich insbesondere modellbasierte Entwicklungsumgebungen zur Beschreibung von Fahrzeugsystemen einschließlich der *Hardwaredesigns* an.

Für eine Unterstützung im Bereich der Hardware-Sicherheitsevaluationen sind bisher lediglich Umsetzungen hervorgegangen, welche eine Verknüpfung von Werkzeugketten erfordern oder im Kontext einzelner Evaluationen stehen, vgl. Kapitel 3.2. Um zum einen sowohl Werkzeugwechsel als auch -brüche zu vermeiden und zum anderen den Automatisierungsgrad zu erhöhen, wird eine integrierte Lösung vorgestellt, welche sämtliche geforderten Evaluationen nach ISO 26262 bereitstellt. Somit können Entwickler und Sicherheitsverantwortliche der unterschiedlichen Firmen bei ihren täglichen Aufgaben im Kontext funktionaler Sicherheit entlastet werden.

Hierzu wurden insbesondere Part 5 sowie Part 10 aus ISO 26262 analysiert, um einen Metamodell-Vorschlag für die Fehlerbeschreibung von *Hardware-Elementen* auf unterschiedlichen Abstraktionsebenen und eine modellbasierte Vorgehensweise für die Durchführung der Sicherheitsevaluationen zur Absicherung von *Hardwaredesigns* bereitzustellen. Das erarbeitete Metamodell erweitert Konstrukte vorhandener Architekturbeschreibungssprachen zur strukturellen Beschreibung von Hardware um Fehlerinformationen. Durch die Integration der Konzepte in eine modellbasierte Werkzeugumgebung für großformatige EEAs kann eine datenbankgestützte Entwicklung erfolgen. Für die Hinterlegung von Ausfallraten und Ausfallarten wird hierzu ein Bibliothekskonzept eingebunden, welches sowohl *Hardware-Komponenten* als auch *Hardware-Bauteile* mit ihren dazugehörigen Fehlerbeschreibungen berücksichtigt. Neben der Modellierung von *Hardwaredesigns* durch Verwendung angelegter Typenbibliotheken können die Ausfallarten der sicherheitsbezogenen *Hardware-Elemente* im Kontext der zugehörigen Sicherheitsziele klassifiziert werden. Um ein Abbild des aktuellen Entwicklungsstandes zu erhalten, werden die Struktur-, Sicherheits- und Fehlerinformationen aus dem Modell akquiriert und die geforderten Sicherheitsevaluationen durchgeführt. Dies umfasst die Hardware-Architekturmetriken, die Ausfallratenklassen-Methode und die Probabilistische Metrik für zufällige Ausfälle der Hardware, welche durch eine Fehlerbaumanalyse (FTA) unterstützt wird. Detaillierte Ergebnisse der Evaluationen einschließlich der Überprüfung auf Erfüllung der festgelegten Zielwerte sowie zusätzliche *dedizierte Maßnahmen* werden für den Sicherheitsnachweis in automatisiert generierten Berichten dokumentiert.

Teile dieser Arbeit konnten bereits innerhalb der Veröffentlichungen [Adler2012b], [Adler2013a] und [Adler2013b] sowie Deliverables im Projekt SAFE und Projektberichten von FTA/EE als eigener Beitrag publiziert werden.

5.1 Analyse der geforderten Prozesse und Arbeitsprodukte nach ISO 26262

Basierend auf dem Dokument ISO 26262 und einer auf Kapitel 4 basierenden Prozessmodellierung konnten nicht nur die Anforderungen, sondern auch die Prozesse mit den Aktivitäten sowie den zu erbringenden Arbeitsprodukten für die Entwicklung funktional sicherer Hardware abgeleitet werden, um diese durch einen modellbasierten Ansatz zu unterstützen.

Die Analyse erfolgte insbesondere innerhalb Part 5, da dieser die Produktentwicklung auf Hardwareebene aufgreift, siehe Abb. 5.1. Aus dem beschriebenen Prozessmodell sind bei der Analyse folgende Aspekte ersichtlich: Die *„Initiation of product development at the hardware level"* in Clause 5 beinhaltet die Erfassung und Planung der Sicherheitsaktivitäten während sämtlicher Phasen der Hardwareentwicklung und bringt diese in den Sicherheitsplan ein.

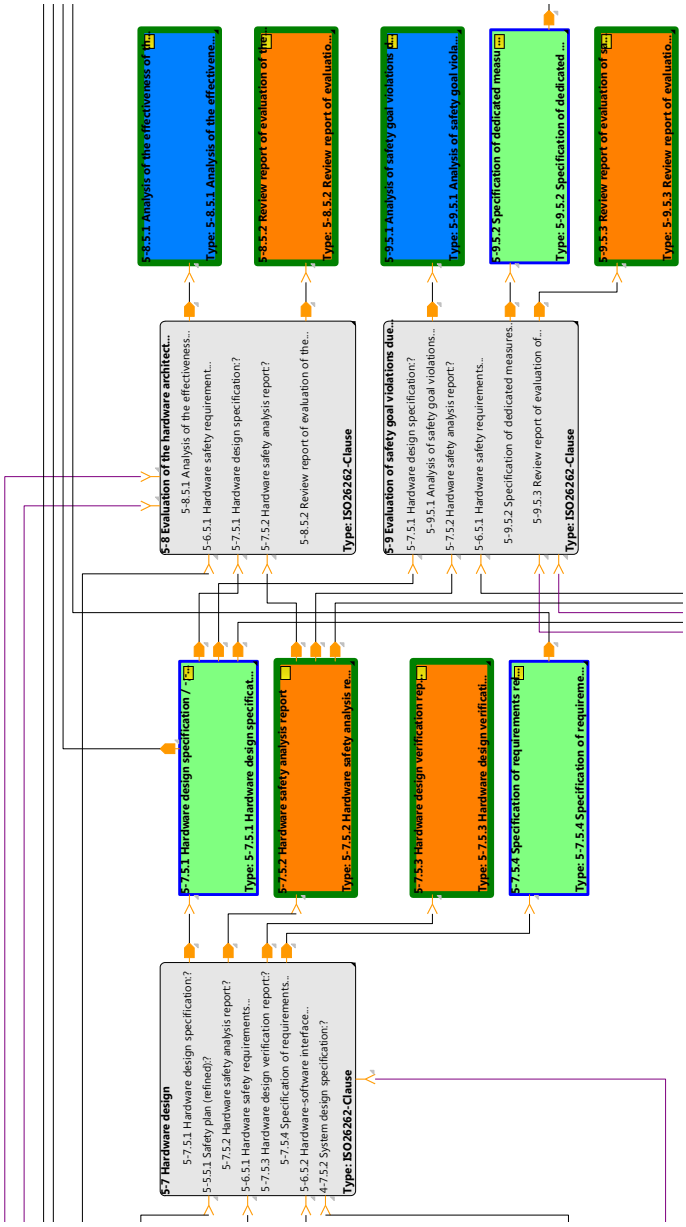


Abb. 5.1: Auszug aus dem Prozessmodell: Hardwaredesign, geforderte Sicherheitsevaluationen und Arbeitsprodukte basierend auf [ISO26262, 5]

Die „*Specification of hardware safety requirements*“ in Clause 6 greift vor allem die Spezifikation von Hardware-Sicherheitsanforderungen auf, welche aus dem technischen Sicherheitskonzept und der Systemdesign-Spezifikation abgeleitet und gegen diese wiederum verifiziert werden. Nicht zuletzt wird die Schnittstelle zur Software aufgegriffen indem das HSI initiiert wird. In Clause 7 „*Hardware design*“, Clause 8 „*Evaluation of the hardware architectural metrics*“ und Clause 9 „*Evaluation of safety goal violations due to random hardware failures*“ wird umfassend auf das iterative *Hardwaredesign* und die zur Absicherung von Hardware durchzuführenden Sicherheitsevaluationen eingegangen. Die in Clause 10 „*Hardware integration and testing*“ beschriebenen Aktivitäten dienen der Überprüfung auf Konformität gegen die anfangs aufgestellten Hardware-Sicherheitsanforderungen. Weiterhin werden im Zuge der Hardwareintegration anhand einer Reihe von vorgeschlagenen Methoden bestimmte Testfälle für Testaktivitäten abgeleitet, welche unter anderem der Kontrolle auf Vollständigkeit und Korrektheit der eingebrachten Sicherheitsmechanismen dienen.

Die Planung der Sicherheitsaktivitäten, wie in Clause 5 gefordert, wird durch den vorgestellten Ansatz in Kapitel 4 unterstützt. Für den Umgang mit Sicherheitsanforderungen sowie deren Verfeinerung in Bezug auf sowohl funktionale/technische Anforderungen als auch Hardware-/Software-Anforderungen existieren bereits zahlreiche bewährte Ansätze und Werkzeuge für das Anforderungsmanagement, auch im Kontext von funktionaler Sicherheit. Somit sind die Aktivitäten von Clause 5 und Clause 6 nicht im Hauptfokus dieses Kapitels. Auch hinsichtlich der Ableitung von Testfällen, welche die Implementierungen von Sicherheitsmechanismen verifizieren, wird auf die innerhalb von Clause 10 verzeichneten Methoden verwiesen. Der Fokus muss daher auf die in Clause 7, 8 und 9 beschriebenen Aktivitäten und deren geforderten Arbeitsprodukte gelegt werden, da diese das Fundament für den iterativen Entwicklungsprozess funktional sicherer Hardware darstellen.

5.1.1 Aktivität Clause 7: Hardwaredesign

In Clause 7 werden spezifische Anforderungen für die Entwicklung des *Hardwaredesigns* aufgegriffen. Die zwei verfolgten Ziele sind einerseits die Hardware konform mit der Systemdesign-Spezifikation sowie den Hardware-Sicherheitsanforderungen zu designen und anderseits anschließend gegen diese zu verifizieren.

Dieser Clause spezifiziert dabei zwei unterschiedliche *Hardwaredesigns*, abgeleitet von unterschiedlichen Entwicklungsphasen: Das *Hardware-Architekturdesign* und das *detaillierte Hardwaredesign*, welche in [ISO26262, 5-7.4.1] und [ISO26262, 5-7.4.2] genauer beschrieben werden. Das *Hardware-Architekturdesign* stellt demnach eine abstrakte Sichtweise auf das *Hardwaredesign* dar, indem gezielt durch Prinzipien der Modularität, angepasstes Granularitätslevel und Einfachheit eine hohe Komplexität vermieden wird [ISO26262, 5-7.4.1.6]. Ein Bezug zu den Hardware-Sicherheitsanforderungen soll bis auf unterste Ebene der *Hardware-Komponenten* gepflegt werden und durch Vererbung das

ASIL hinzugefügt werden, um eine vollständige Nachverfolgbarkeit zu gewährleisten [ISO26262, 5-7.4.1.1 ff]. Das *Hardware-Architekturdesign* ist somit ein erstes Design in einer frühen Entwicklungsphase. Das *detaillierte Hardwaredesign* stellt die Implementierung der *Hardware-Komponenten* in späteren Entwicklungsphasen dar und befindet sich auf der Ebene von elektronischen Schaltplänen. Hier wird der Fokus auf eine hohe Detailtiefe sowie Vollständigkeit gelegt. Grundsätze für ein robustes *Hardwaredesign*, beispielsweise die konservative Spezifikation von *Hardware-Bauteilen*, sollten während der Entwicklung beachtet werden [ISO26262, 5-7.4.2.4].

Des Weiteren müssen durch Sicherheitsanalysen die Ursachen und Auswirkungen von Fehlern betrachtet werden. Somit steht Clause 7 in direktem Bezug zu Clause 8 und 9. In Clause 8 wird die „*Evaluation of the hardware architectural metrics*“ und in Clause 9 die „*Evaluation of safety goal violations due to random hardware failures*“ aufgegriffen. Nicht zuletzt gilt es das *Hardwaredesign* zu verifizieren und die Aspekte für Produktion, Betrieb, Kundendienst und Dekommissionierung zu berücksichtigen.

Als Arbeitsprodukte sind die Hardware-Designspezifikation, der Hardware-Sicherheitsanalyse-Bericht, der Hardware-Designverifikations-Bericht und die Spezifikation von Anforderungen bezogen auf Produktion, Betrieb, Kundendienst und Dekommissionierung zu nennen.

5.1.2 Aktivität Clause 8 und 9: Hardware-Sicherheitsevaluationen

Die in Clause 8 und 9 verfolgten Ziele sind, das *Hardwaredesign* unter dem Item robust zu designen und zugleich abzusichern, dass das Restrisiko einer Sicherheitsziel-Verletzung, bedingt durch zufällige Ausfälle der Hardware, ausreichend niedrig ist. Als Eingangsdokumente für die Aktivität beider Sicherheitsevaluationen werden die Spezifikation der Hardware-Sicherheitsanforderungen, die Hardware-Designspezifikation und der Hardware-Sicherheitsanalyse-Bericht benötigt. Als unterstützende Informationen können das technische Sicherheitskonzept und die Systemdesign-Spezifikation hinzugezogen werden.

Mit Fokus auf die unterschiedlichen Sicherheitsanalysen und -evaluationen während der Produktentwicklung von Hardware, beschreibt ISO 26262 Part 5 „*Product development at the hardware level*“ quantitative Evaluationen, welche für die *Hardwaredesigns* durchgeführt werden müssen, um Robustheit abzusichern, einen Nachweis zu ermöglichen sowie den Anforderungen bzgl. Fehlerbehandlung zu genügen. Die geforderten Evaluationen sind in Abb. 5.2 ersichtlich.

Clause 8 „*Evaluation of the hardware architectural metrics*“ beschreibt die Evaluation von *Hardwaredesigns* durch Anwendung der Hardware-Architekturmetriken. Diese Metriken stellen eine quantitative Evaluation gegenüber zufälligen Hardwarefehlern dar. Hierbei stehen objektive und vergleichbare Ergebnisse für das *Hardwaredesign* im Fokus. Die Evaluation ist beschränkt auf die Betrachtung von sicherheitsbezogenen *Hardware-*

Elementen, welche signifikant zu einer Verletzung des zugehörigen Sicherheitsziels beitragen. Daher werden die Hardware-Architekturmetriken in die Single-Point Fault Metric (SPFM) und die Latent-Fault Metric (LFM) unterteilt, welche beide angewandt werden müssen. Die Ergebnisse der Evaluationen werden anschließend gegenüber Zielwerten, abgeleitet von der ASIL-Klassifizierung der zugeordneten Sicherheitsziele, verifiziert.

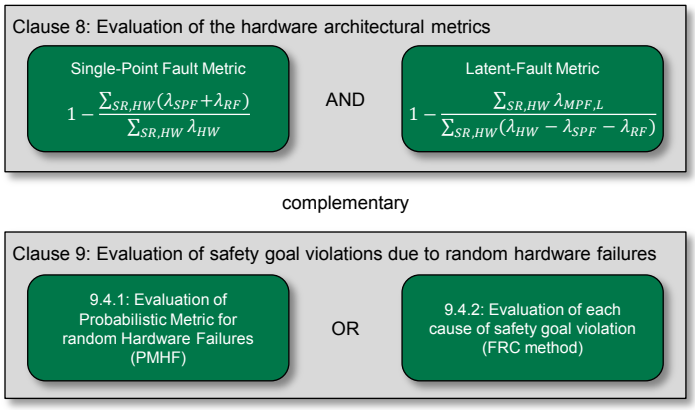


Abb. 5.2: Übersicht der geforderten Hardware-Sicherheitsevaluationen nach ISO 26262

Da es sich bei der *„Evaluation of the hardware architectural metrics“* um relative Verhältnisse handelt, gilt es komplementär das Restrisiko für die Verletzungen der Sicherheitsziele nach Clause 9 *„Evaluation of safety goal violations due to random hardware failures“* zu evaluieren. ISO 26262 schlägt zwei alternative Methoden vor: *„Evaluation of Probabilistic Metric for random Hardware Failures (PMHF)“* oder die *„Evaluation of each cause of safety goal violation“* durch Verwendung von sogenannten Ausfallratenklassen (FRCs). PMHF repräsentiert einen globalen Ansatz, welcher die maximale Wahrscheinlichkeit für eine Verletzung des Sicherheitsziels beschreibt. Die FRC-Methode berücksichtigt eine individuelle Evaluation einer jeden Sicherheitsziel-Verletzung.

Wie in Abb. 5.2 ersichtlich, sind beide sich ergänzende Evaluationen nach Clause 8 und 9 gefordert. Die Arbeitsprodukte der beiden Clauses sind einerseits die entsprechenden Analysen zum Nachweis der Anfälligkeit der Architektur gegenüber zufälligen Ausfällen der Hardware und Verletzungen von Sicherheitszielen sowie andererseits geforderte Berichtsdokumente in Form von Review Reports. Als ein spezifisches Nachweisdokument geht das zusätzliche Arbeitsprodukt *„Specification of dedicated measures for hardware“* aus Clause 9 hervor.

5.2 Konzept für modellbasierte Hardware-Sicherheitsevaluationen

Für die modellbasierte Entwicklung von funktional sicherer Hardware ist die Bereitstellung einer formalen Beschreibung des *Hardwaredesigns* einschließlich Fehlerinformationen für die Ausführung von Hardware-Sicherheitsevaluationen von maßgeblicher Bedeutung.

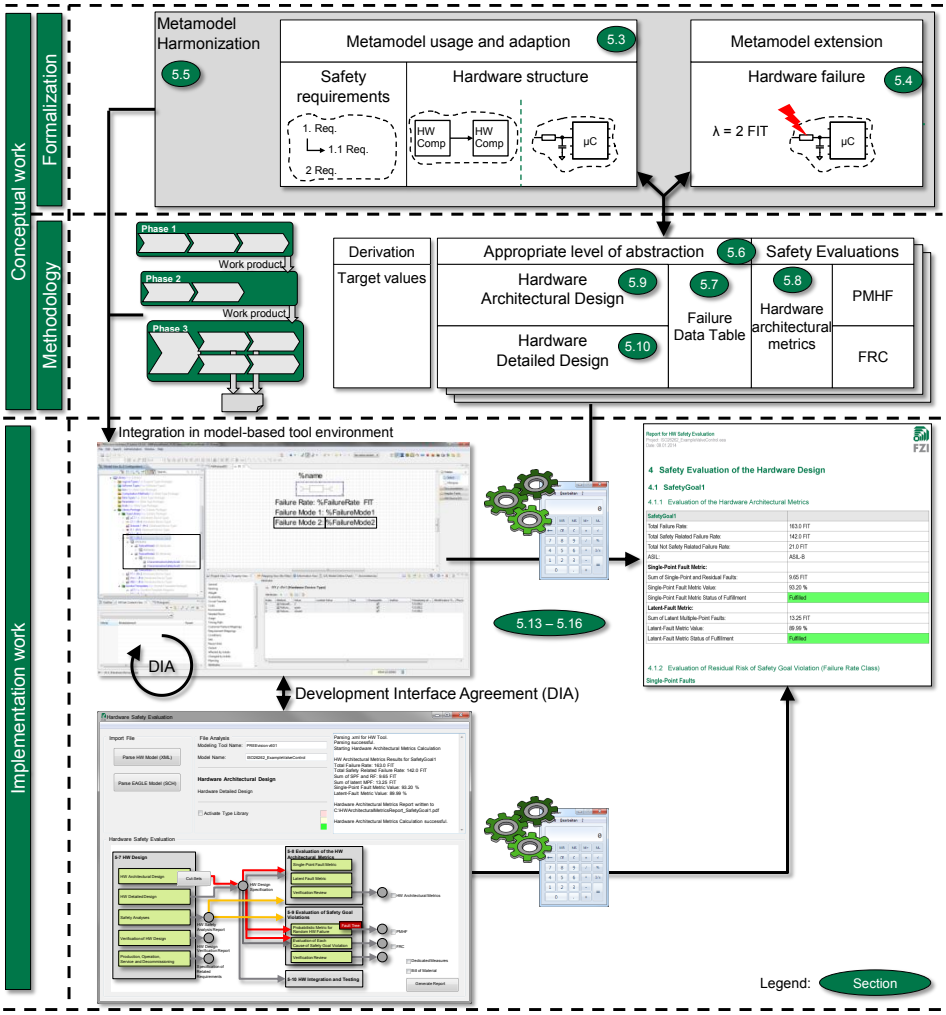


Abb. 5.3: Konzept für modellbasierte Entwicklung funktional sicherer Hardware

In Abb. 5.3 ist die Übersicht des erarbeiteten Konzeptes ersichtlich. Hierfür konnten unter anderem aus der ISO 26262 abgeleitete Anforderungen, welche im Rahmen des SAFE-Projektes als eigener Beitrag unter [D322] und [D21c] verankert wurden, einbezogen werden. Um die Hardware-Sicherheitsevaluationen rechnergestützt in modellbasierten Entwicklungsumgebungen durchzuführen, ist grundlegend eine konzeptionelle Erarbeitung in Form einer Formalisierung und einer Vorgehensweise sowie eine darauf aufbauende Umsetzung in einer Werkzeugumgebung für EEAs erforderlich.

Um sowohl bereits während der frühen Konzeptphase als auch in der fortgeschrittenen Entwicklung den *Hardwaredesign*-Prozess zu unterstützen, ist eine formalisierte Strukturbeschreibung für das *Hardwaredesign* auf den unterschiedlichen vorgestellten Abstraktionsebenen eine der Grundvoraussetzungen. Im Kontext von funktionaler Sicherheit erfordert dies zudem die Beschreibung von Sicherheitsanforderungen, wie beispielsweise der Sicherheitsziele (SGs), funktionaler (FSRs) und technischer (TSRs) Sicherheitsanforderungen, sowie deren Verknüpfung mit der strukturellen Hardwarebeschreibung. Da die Zielwerte für die Evaluationen sich aus den Sicherheitsanforderungen und deren jeweiligem ASIL ableiten, muss eine übergreifende Nachverfolgbarkeit im Datenmodell gewährleistet werden. Weiterhin gilt es zu untersuchen, inwiefern die Sicherheitsanforderungen mit den Hardware-Sicherheitsevaluationen und den Zielwerten konzeptionell zu harmonisieren sind. Das Fehler-Metamodell repräsentiert den ausschlaggebenden Metamodell-Beitrag im Kontext von funktionaler Sicherheit, welcher als Erweiterung angelehnt an bestehende Architekturbeschreibungssprachen erarbeitet werden muss, um darauf aufbauend die Hardware-Sicherheitsevaluationen zu unterstützen. Hier gilt es die erforderlichen Metamodell-Konstrukte unter Einbeziehung der in Kapitel 2.7 dargestellten Fehlerinformationen für Hardware zu erarbeiten und an die strukturellen Konstrukte anzubinden.

Aufbauend auf der Möglichkeit einer konsistenten modellbasierten Beschreibung gilt es, die Hardware-Sicherheitsevaluationen mit ihren zugehörigen Zielwerten konzeptionell zu untersuchen. Hierbei ist es bedeutend, die zugrundeliegenden Methoden zu erfassen und zu analysieren. Weiterhin sind die geforderten Hardware-Sicherheitsevaluationen auf ihre Anwendbarkeit hinsichtlich den unterschiedlichen Abstraktionsebenen zu betrachten. Darauf aufbauend sind angepasste Vorgehensweisen für die Durchführung der Sicherheitsevaluationen spezifisch für die jeweiligen Abstraktionsebenen abzuleiten. Die Zielwerte, gegen welche die Evaluationsergebnisse geprüft werden, sind spezifisch für die Evaluation und basierend auf den verknüpften Sicherheitsanforderungen individuell abzuleiten. Da die Zielwerte von unterschiedlichen Parametern abhängen, muss hierzu erarbeitet werden, inwiefern sich die Zielwerte oder Zielwerttabellen durch Begründungen verändern oder dynamisch erweitern lassen.

Um eine modellbasierte Unterstützung zu bieten und diese für einen produktiven Einsatz geeignet zu machen, ist eine Integration der Metamodell-Vorschläge in eine Architekturbeschreibungssprache und Einbindung in bestehende Werkzeugprozesse erforderlich. Die Integration zieht eine Erweiterung und gegebenenfalls Anpassung vorhandener

Konstrukte in der Entwicklungsumgebung nach sich. Zudem sind angepasste Vorgehensweisen zur Ausführung der Hardware-Sicherheitsevaluationen zu hinterlegen. Für diese sind die Berechnungsvorschriften als Implementierungen umzusetzen und die Zielwerte einzubringen, damit die Metriken auf dem Datenmodell ausgeführt werden können. Eine wichtige Anforderung an die ausführbaren Hardware-Sicherheitsevaluationen ist, dass diese in iterative und kollaborative Entwicklungen eingebunden werden können, da diese zu unterschiedlichen Entwicklungsphasen durchgeführt werden müssen. Dies ist aus den Gesichtspunkten von kurzen Entwicklungszyklen, Absicherung hinsichtlich funktionaler Sicherheit sowie Optimierung des *Hardwaredesigns* relevant. Gleichzeitig muss die Integration eine benutzerfreundliche Bedienung ermöglichen, um sie hierdurch in die gängigen Entwicklungsabläufe einzubinden. Weiterhin sollen Berichte und Nachweisdokumente unter anderem aus den Evaluationsergebnissen über bereitgestellte Dokumentenvorlagen automatisiert generiert werden können, um geforderten Arbeitsprodukten zu genügen.

5.3 Strukturelle Hardwarebeschreibung und Bezug zu Sicherheitsanforderungen

5.3.1 Strukturelle Beschreibung von Hardwaredesigns

Benötigt werden nach Kapitel 2.7.1.2 zwei unterschiedliche Abstraktionsebenen für die Beschreibung des *Hardware-Architekturdesigns* und des *detaillierten Hardwaredesigns*. Dieser Anforderung einer strukturellen Beschreibungsmöglichkeit von *Hardwaredesigns* genügen ADLs mit ihren zur Verfügung gestellten Abstraktionsebenen und Formalisierungen sowie den Konstrukten zur Beschreibung von Hardware, vgl. hierzu die Klassen *HardwareComponentType* und *HardwareComponentPrototype* aus dem Bereich der strukturellen Hardware-Modellierung von EAST-ADL. Somit soll für die Strukturbeschreibung auf bereits vorhandene Konstrukte zurückgegriffen werden. Es gilt, diese auf Vollständigkeit im Kontext der geforderten Hardware-Sicherheitsevaluationen zu untersuchen. Ein Vorschlag für eine Erweiterung im Sinne der Fehlerinformationen und eine Ankopplung an die Strukturbeschreibungen sind zu erarbeiten. Angemerkt sei, dass die strukturelle Beschreibung auf Ebene des *detaillierten Hardwaredesigns* nicht nur für rein elektrische/elektronische Elemente, sondern auch für elektromechanische *Hardware-Bauteile* berücksichtigt werden muss. Unter dem Gesichtspunkt der Sicherheitsevaluationen werden jedoch nur die elektrischen/elektronischen Aspekte nach [ISO26262, 5-8.2] berücksichtigt.

5.3.2 Verknüpfung mit Sicherheitsanforderungen

Da die Hardware-Sicherheitsevaluationen auf unterschiedlichen Abstraktionsebenen und damit zu verschiedenen Designphasen angewandt werden, sind zunächst Sicherheits-

anforderungen als übergreifende Klasse im Fokus. Hierbei müssen die Sicherheitsziele sowie die davon abgeleiteten funktionalen und technischen Sicherheitsanforderungen berücksichtigt werden. Für die Sicherheitsevaluationen werden im Rahmen dieser Arbeit, angelehnt an [ISO26262, 5-AnnexE], die Sicherheitsziele verwendet. Zusätzlich ist nach ISO 26262 eine Klassifizierung des ASIL vorzunehmen, um die Zielwerte abzuleiten, gegen welche im Rahmen der Hardware-Sicherheitsevaluationen geprüft werden muss.

Um den Bezug zum *Hardwaredesign* herzustellen, sind die Sicherheitsziele mit den relevanten *Hardware-Elementen* je nach betrachteter Abstraktion auf Ebene der *Hardware-Komponenten* oder *Hardware-Bauteile* zu verknüpfen. Dies wird über Konstrukte zur ebenenübergreifenden Verknüpfung von Anforderungen in den ADLs bereitgestellt. Somit können je nach Sicherheitsziel unterschiedliche Teilbereiche des *Hardwaredesigns* aufgespannt werden, welche anschließend durch die Hardware-Sicherheitsevaluationen analysiert werden. Eine Überlagerung der durch die Sicherheitsziele abgegrenzten Teilbereiche wird hierdurch ebenfalls ermöglicht. Dies wird beispielsweise insbesondere in Bereichen benötigt, in denen sich auf Schaltplanebene komplexere Bauteile wie Microcontroller befinden, da diese meist im Fokus mehrerer Sicherheitsziele stehen.

Angemerkt sei, dass somit die Sicherheitsziele mit den einzelnen *Hardware-Bauteilen* verknüpft werden, vgl. [ISO26262, 5-AnnexE]. Dies ist unter dem Gesichtspunkt der Granularität auf Schaltplanebene diskutierbar, da nach [ISO26262, 5-6.4] die Hardware-Sicherheitsanforderungen aus den auf Hardware zugewiesenen technischen Sicherheitsanforderungen abgeleitet werden sollen. Die technischen Sicherheitsanforderungen wurden aus den funktionalen Sicherheitsanforderungen und diese wiederum aus den Sicherheitszielen abgeleitet. Da jedoch innerhalb des Anforderungsmanagements und in Richtung der Hardware-Abstraktionsebenen eine Nachverfolgbarkeit gegeben ist, kann auf diese im Sinne der geforderten Verifikation zurückgegriffen werden.

5.3.3 Aspekt der Sicherheitsbezogenheit

Als Verfeinerung der Zugehörigkeit eines *Hardware-Elements* zu einem oder mehreren Sicherheitszielen wirft ISO 26262 eine zusätzliche Einstufung der Sicherheitsbezogenheit auf, siehe [ISO26262, 5-7.4.3.2]. Alle danach als sicherheitsbezogen eingestuften *Hardware-Elemente* sind bei den Hardware-Sicherheitsevaluationen zu berücksichtigen. Dies hat als Auswirkung, dass alle Ausfallarten von nicht-sicherheitsbezogenen *Hardware-Elementen* wiederum als SFs klassifiziert werden, vgl. [ISO26262, 5-AnnexB]. Die Einstufung nicht-sicherheitsbezogen kann eingebracht werden, wenn nach [ISO26262, 5-Fig.B2] der Ausfall eines *Hardware-Elements* einen nicht signifikanten Einfluss auf die Verletzung des Sicherheitsziels zur Folge hat. Da die Einstufung keinen direkten Aspekt des *Hardware-Elements* widerspiegelt, sondern im Kontext zu einem Sicherheitsziel gesehen werden muss, wird vorgeschlagen, diese Erweiterung als Eigenschaft einer Verknüpfung zwischen dem Sicherheitsziel und dem entsprechenden strukturellen *Hardware-Element* zu hinterlegen.

5.4 Metamodell zur Hardware-Fehlerbeschreibung

Um Fehlerinformationen für *Hardware-Elemente*, im Folgenden übergreifend für *Hardware-Komponenten* und *Hardware-Bauteile* verwendet, bereitzustellen, ist eine Erweiterung der Beschreibung um Konstrukte zur Hardware-Fehlerbeschreibung für die unterschiedlichen Abstraktionsebenen erforderlich. In diesem Abschnitt wird daher ein Auszug aus dem zugehörigen Metamodell-Vorschlag für „Hardware Failure“ präsentiert, da dies die grundlegende Erweiterung zu bestehenden Ansätzen wie EAST-ADL darstellt. Dieser ist als UML-Klassendiagramm zur Verfügung gestellt.

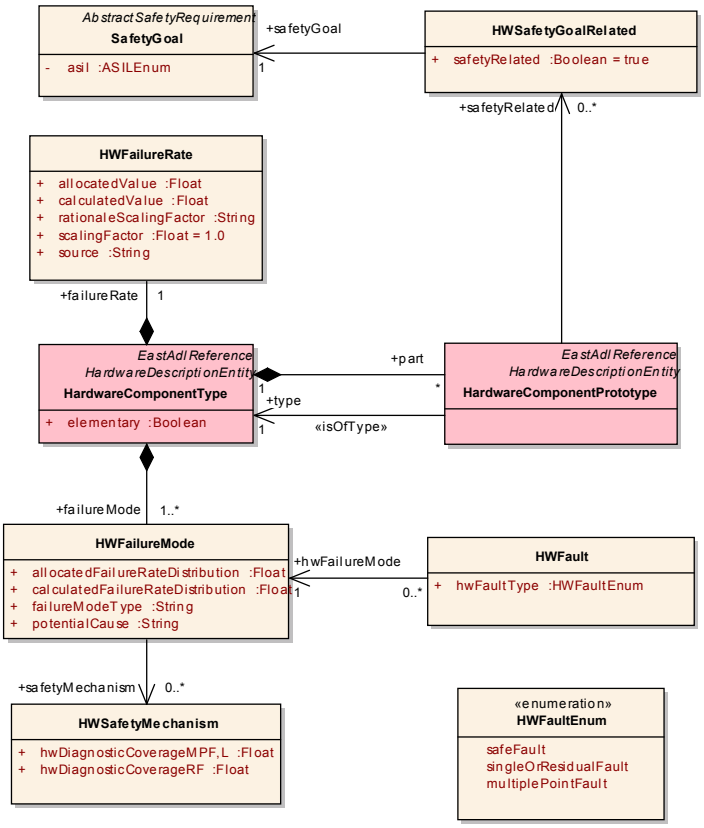


Abb. 5.4: Klassendiagramm „Hardware Failure“ - vorgeschlagenes Metamodell nach [Adler2013a]

Ausschlaggebend ist, dass der Metamodell-Vorschlag sowohl an unterschiedliche existierende Metamodelle angekoppelt als auch adaptiert werden kann. Hierbei werden die Klassen *HardwareComponentType* und *HardwareComponentPrototype* exemplarisch aus

EAST-ADL genutzt, um den Bezug zur strukturellen Hardwarebeschreibung innerhalb von ADLs zu verdeutlichen. Das Konzept für die Erweiterung im Hinblick auf Fehlerinformationen sowie die Klassen und deren Beziehungen zueinander sind in Abb. 5.4 dargestellt und werden im Folgenden vorgestellt. Diese konnten unter anderem im Artikel [Adler2013a] und dem SAFE Projekt-Deliverable [D322] verankert werden. In Letzterem konnte ein umfangreicher Metamodell-Vorschlag für die Erweiterung von EAST-ADL hinsichtlich funktionaler Sicherheit im Bereich Hardware erarbeitet werden.

Die Klasse *HardwareComponentPrototype* ist nach [21, 7.2.6] ein Teil des *Hardware-ComponentType* und repräsentiert das Vorkommen eines *Hardware-Elements* mit Bezug zum entsprechenden definierten Typ. Der *HardwareComponentPrototype* kann wiederum selbst einen Typ repräsentieren. Somit können hierarchische Strukturen bereitgestellt werden und durch die Typisierung ist eine einmalige Definition im Sinne einer Wiederverwendung gegeben. Durch das zusätzlich eingebrachte Attribut *elementary* vom Datentyp *Boolean* wird ermöglicht, eine Information beispielsweise hinsichtlich eines elementaren *Hardware-Bauteils* nach ISO 26262 zu hinterlegen. Hingewiesen wird, dass Vorsicht bei den Bezeichnungen geboten ist, da die *HardwareComponent* aus EAST-ADL in diesem Fall mit dem *Hardware-Element* nach ISO 26262 gleichzusetzen ist.

5.4.1 Klasse HWFailureRate

Die Klasse *HWFailureRate* wird der Klasse *HardwareComponentType* zugeordnet und soll die Annotation einer entsprechenden Ausfallrate für *Hardware-Elemente* ermöglichen. Der Komposition wird die Multiplizität 1 zugewiesen aufgrund der Tatsache, dass jedes *Hardware-Element* eine Ausfallrate besitzt. Die Klasse stellt die Attribute *calculatedValue* und *allocatedValue* vom Typ der Gleitkommazahlen zur Verfügung, um zwischen einer kalkulierten Ausfallrate, welche bereits bestimmt oder von einem Standard bzw. einer Industriereferenz entnommen wurde, und einer zugewiesenen Ausfallrate, welche als eine anfängliche Annahme definiert ist, zu unterscheiden. Letzteres kann exemplarisch insbesondere für komplexe *Hardware-Elemente* ohne anfangs bekannte Datenbank-Ausfallrate genutzt werden. Die Datenquelle aus der die Ausfallrate entnommen wird, ist unter dem Attribut *source* zu hinterlegen. Durch den Datentyp *String* dient es einer rein textuellen Dokumentation. Ein Skalierungsfaktor wird nach ISO 26262 unter anderem dafür genutzt, um Ausfallraten aus unterschiedlichen Quellen zu kombinieren ohne eine Verfälschung in der Quantifizierung zu bewirken. Datentyp des Attributs *scalingFactor* ist *Float* mit einem Vorgabewert von 1.0. Somit wird initial die gleiche Quelle für alle Ausfallraten angenommen. Soll ein Skalierungsfaktor nach [ISO26262, 5-9.4.2.7] angewandt werden, so ist zusätzlich eine textuelle Argumentation über *rationaleScaling-Factor* anzugeben. Eine Anwendung hierfür wird als informativer Teil für PMHF in [ISO26262, 5-AnnexF] beschrieben.

5.4.2 Klasse HWFailureMode

Die Klasse *HWFailureMode* wird ebenfalls der Klasse *HardwareComponentType* angehängt und dient der Beschreibung von einer bis mehreren Ausfallarten eines *Hardware-Elements*. Daher wird der Komposition die Multiplizität 1 bis unendlich zugeordnet. Das Attribut *failureModeType* ermöglicht die textuelle Beschreibung der Ausfallart, zum Beispiel „Kurzschluss“ für einen Widerstand auf Ebene des *detaillierten Hardwaredesigns*. Über die Attribute *allocatedFailureRateDistribution* und *calculatedFailureRateDistribution* vom Datentyp der Gleitkommazahlen wird eine prozentuale Verteilung ausgehend von der Gesamtausfallrate, welche in der Klasse *HWFailureRate* festgelegt wurde, auf die einzelnen spezifischen Ausfallarten des *Hardware-Elements* vorgenommen. Die zwei Attribute stellen das gleiche Konzept zur Unterscheidung bereit, wie bereits für die Klasse *HWFailureRate* vorgestellt. Das Attribut *potentialCause* ermöglicht optional über eine textuelle Beschreibung eine nicht-funktionale Ursache für die Ausfallart eines *Hardware-Elements* zu hinterlegen. Der Eintritt der spezifischen Ausfallart kann beispielsweise durch Temperatur oder Vibration bedingt sein, vgl. [ISO26262, 5-7.4.1.7, 5-7.4.2.2].

5.4.3 Klasse HWSafetyMechanism

Die Klasse *HWSafetyMechanism* ist mit der Klasse *HWFailureMode* assoziiert und repräsentiert abstrakt einen Sicherheitsmechanismus, welcher die Verletzung eines spezifischen Sicherheitszieles durch eine Ausfallart zu einem bestimmten Anteil abdeckt. Die Einführung eines Sicherheitsmechanismus führt beispielsweise von einem Einzelfehler (SPF) zu einem Restfehler (RF). Gleichwohl kann dies auch bei Mehrfachfehlern zu einem latenten MPF führen. Die Attribute *diagnosticCoverageRF* und *diagnosticCoverageMPF,L* beschreiben den Diagnoseabdeckungsgrad $K_{DC,RF}\,(SM)$ und $K_{DC,MPF,L}\,(SM)$ in Prozent. Die Assoziation erhält die Multiplizität 0 bis unendlich, da die Existenz eines Sicherheitsmechanismus nicht zwingend erforderlich ist. Angemerkt sei, dass nach [ISO26262, 5-AnnexE] die Verknüpfung von einzelnen Sicherheitsmechanismen auf Ebene der *Hardware-Bauteile* erfolgt, allerdings nach [ISO26262, 10-AnnexA] eine Zuordnung auf Ebene einzelner Ausfallarten besteht. Durch die feine Granularität auf Ebene der Ausfallarten können modellierungstechnisch beide Ansätze abgedeckt werden.

5.4.4 Klasse HWFault

Die Klasse *HWFault* wird der Klasse *HWFailureMode* assoziiert und stellt die Klassifikation einer Ausfallart im Kontext eines jeden spezifischen Sicherheitsziels zur Verfügung, vgl. [ISO26262, 5-FigB.1]. Der zugewiesene Datentyp des Attributs *hwFaultType* ist die Enumeration *HWFaultEnum*. Diese enthält drei Literale *safeFault*, *singleOrResidualFault* und *multiplePointFault*, welche die Differenzierung zwischen verschiedenen Klassifikationen der zugehörigen Ausfallart ermöglicht. Eine detaillierte Spezifikation

der Klassifikation, exemplarisch ob es sich um einen Einzel- oder Restfehler handelt, ist nicht notwendig, da dies durch das Vorhandensein eines Sicherheitsmechanismus automatisch erschlossen werden kann. Eine einzelne Ausfallart eines *Hardware-Elements* kann eine unterschiedliche Klassifikation für jedes spezifische Sicherheitsziel besitzen. Daher ist die Einstufung in Abhängigkeit des zu betrachtenden Sicherheitsziels vorzunehmen und die Assoziation erhält die Multiplizität von 0 bis unendlich.

5.4.5 Klasse SafetyGoal

Die Klasse *SafetyGoal* wird aus einer Superklasse für abstrakte Sicherheitsanforderungen abgeleitet und repräsentiert eine Anforderung im Kontext der funktionalen Sicherheit auf höchster Ebene (Top-Level). Unter dieser Superklasse sind auch die funktionalen und technischen Sicherheitsanforderungen sowie die Hardware-Sicherheitsanforderungen anzusiedeln, welche im Zuge der Hardwareentwicklung berücksichtigt werden müssen. Über das zugewiesene Attribut *asil* mit seinen Enumerationsliteralen kann eine Einstufung in das ASIL eingebracht werden, vgl. [21, 19.2.1], welches für die Ableitung der Zielwerte für die Hardware-Sicherheitsevaluationen benötigt wird.

5.4.6 Klasse HWSafetyGoalRelated

Um zu beschreiben, dass ein *HardwareComponentPrototype* im Sinne eines *Hardware-Elements* nach ISO 26262 für ein spezifisches Sicherheitsziel sicherheitsbezogen ist, wird die Klasse *HWSafetyGoalRelated* bereitgestellt. Diese Klasse hat ein einziges Attribut *safetyRelated* über welches die Sicherheitsbezogenheit als boolescher Wert eingebracht wird. Zudem muss die Multiplizität von 0 bis unendlich für die Assoziation zwischen *HardwareComponentPrototype* und *HWSafetyGoalRelated* eingebracht werden. Dies ist erforderlich, da einzelne *Hardware-Elemente* im Kontext verschiedener Sicherheitsziele betrachtet werden können.

5.5 Unterstützung der Entwicklungsschnittstellen-Vereinbarung

Nach [ISO26262, 1-1.24] beschreibt ISO 26262 eine Entwicklungsschnittstellen-Vereinbarung (DIA), welche zwischen dem Auftraggeber und Zulieferer spezifiziert werden muss und setzt damit hauptsächlich den Fokus auf verteilte Entwicklungen. Diese Übereinkunft nimmt einen großen Einfluss auf die Art und Weise wie Informationen untereinander ausgetauscht werden, unabhängig von der Häufigkeit. Das DIA selbst stellt ein Arbeitsprodukt nach [ISO26262, 8-5.5.2] dar, die Planung sollte im Rahmen des Sicherheitsplans nach [ISO26262, 2-6.4.3.5f] verankert werden.

Das DIA enthält nach [ISO26262, 8-5.4.3.1] unter anderem die Prozesse und Aktivitäten, welche auf beiden Seiten abgearbeitet werden, den Austausch von Informationen und Arbeitsprodukten, die Verantwortlichkeiten für die einzelnen Aktivitäten und die Kommunikation von Zielwerten im Zuge der Sicherheitsevaluationen für Hardware. Nicht zuletzt werden die Werkzeugschnittstellen zum Informationsaustausch aufgegriffen, um eine Kompatibilität in verteilten Entwicklungen zu gewährleisten.

Durch die vorgestellte integrierte Prozessmodellierung mit Zuweisung von Verantwortlichkeiten auf die einzelnen Aktivitäten und Arbeitsprodukte können von den adressierten Inhalten bereits zahlreiche erfüllt werden. Betrachtet man speziell die Hardwareentwicklung, so kann durch das Konzept einer integrierten modellbasierten Entwicklung funktional sicherer Hardware das DIA weitergehend unterstützt werden. Dies betrifft sowohl die formale Beschreibung der *Hardwaredesigns* angereichert mit Fehlerinformationen als auch die zugehörigen Sicherheitsanforderungen sowie die auszuführenden Sicherheitsevaluationen einschließlich der festgelegten Zielwerte.

Bedingt durch die Verwendung von in Unternehmen über Jahre etablierter Werkzeuge soll zusätzlich eine Möglichkeit geboten werden, das DIA unabhängig von den jeweiligen Werkzeugen zu unterstützen, um nicht nur integrierte Ansätze hervorzuheben. Weiterhin soll der sukzessiven Einführung einer übergreifenden modellbasierten Entwicklung nichts im Wege stehen, vgl. [92]. Hierzu kann eine Schnittstelle in verteilten Entwicklungen in einem ersten Schritt durch die Beschreibung einer XSD unterstützt werden, um einen Austausch im Sinne des DIA zu ermöglichen. Durch das dazugehörige XML-Austauschformat wird somit eine Serialisierung der Modellinformationen bereitgestellt. Da sowohl Auftraggeber als auch Zulieferer gleichermaßen berücksichtigt werden müssen, wird dieses generisch gehalten, um die Struktur- und Fehlerinformationen hinsichtlich beider Abstraktionsebenen des *Hardwaredesigns* zu hinterlegen. Im Fokus des Austauschformates stehen im Rahmen des DIA weiterhin auch Zielwertinformationen, welche in die eine oder andere Richtung bereitgestellt werden. Zusätzliche Informationen zum Design, wie beispielsweise das verwendete Tool oder spezifische Einstellungen, welche vorgenommen wurden, sind enthalten. Bestimmte Elemente und Attribute, wie beispielsweise die Verwendung des Bibliotheksansatzes, sind hierbei optional nutzbar.

Abb. 5.5 gibt einen Überblick über die Gesamtstruktur der XSD auf oberster Ebene, welche über das Wurzelelement *Model* aufgespannt wird. Auf der ersten Hierarchieebene existieren sechs untergeordnete Elemente: *CommonInformation*, *Settings*, *SafetyRequirements*, *SafetyMechanisms*, *HWElements* und *LibraryHWElementTypes*. Bezüglich der Elemente der XSD in allen Hierarchieebenen ist standardmäßig die Multiplizität 1 vorgegeben. Falls die Attribute *minOccurs* und *maxOccurs* somit für ein Element nicht explizit überschrieben werden, besitzen beide den Wert 1. Die einzelnen Elemente dieser obersten Ebene werden in den folgenden Abschnitten vorgestellt.

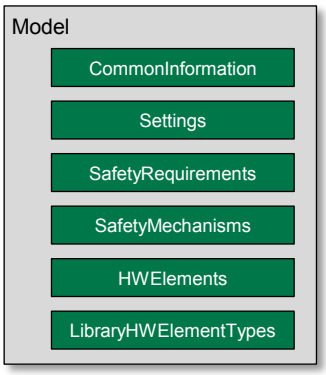


Abb. 5.5: Überblick über den Aufbau der XSD zur Unterstützung des DIA

CommonInformation: Unter dem ersten Element sind sämtliche allgemeine Informationen zum Modell aufgehangen. Das Element *ModelName* enthält die Bezeichnung für das gekapselte Modell. Anschließend kann über das Element *Description* eine individuelle Beschreibung hinzugefügt werden, welche optional gehalten ist. Das Elemente *ModelingTool* fasst mit seinen untergeordneten Elementen *Name* und *Version* Informationen zum genutzten Werkzeug zusammen. Für alle hier dargestellten Elemente ist der Datentyp *String* gewählt, da diese textuelle Informationen bereitstellen. Auszug 5.1 zeigt zusammenfassend die Struktur des Elements *CommonInformation*.

```
1 <xs:element name="CommonInformation">
2   <xs:element name="ModelName" type="xs:string">
3   <xs:element name="Description" type="xs:string" minOccurs="0">
4   <xs:element name="ModelingTool">
5     <xs:element name="Name" type="xs:string">
6     <xs:element name="Version" type="xs:string">
```

Quellcode 5.1: Struktur von *CommonInformation*

Settings: Unter dem Element *Settings* sind sämtliche Parameter enthalten, die für die Durchführung der Sicherheitsevaluationen relevant sind. Diese sind beispielsweise die Abstraktionsebene des *Hardwaredesigns*, die Verwendung der Typenbibliothek als Basis für die Fehlerinformationen oder auch sämtliche Zielwerte für die Hardware-Sicherheitsevaluationen. Die Zielwerte sind initial bereits mit den vorgeschlagenen Werten aus ISO 26262 befüllt, können aber vom Benutzer frei überschrieben werden. Die vollständige Struktur des Elements *Settings* ist in Auszug 5.2 dargestellt.

```
<xs:element name="Settings">
  <xs:element name="AbstractionLevel" type="AbstractionLevel">
  <xs:element name="ActivateTypeLibrary" type="xs:boolean" minOccurs="0">
  <xs:element name="ActivateFaultTypeDetermination" type="xs:boolean" minOccurs="0">
  <xs:element name="ExecuteArchitecturalMetrics" type="xs:boolean" minOccurs="0">
  <xs:element name="ExecuteFRC" type="xs:boolean" minOccurs="0">
  <xs:element name="ExecutePMHF" type="xs:boolean" minOccurs="0">
  <xs:element name="Lifetime" type="xs:float">
  <xs:element name="TargetValues">
    <xs:element name="ActivateDefaultISOTargetValues" type="xs:boolean">
    <xs:element name="SPFMetric" minOccurs="0">
      <xs:element name="ASILB" type="Percentage">
      <xs:element name="ASILC" type="Percentage">
      <xs:element name="ASILD" type="Percentage">
      <xs:element name="Rationale" type="xs:string">
    <xs:element name="LFMetric" minOccurs="0">
      <xs:element name="ASILB" type="Percentage">
      <xs:element name="ASILC" type="Percentage">
      <xs:element name="ASILD" type="Percentage">
      <xs:element name="Rationale" type="xs:string">
    <xs:element name="PMHF" minOccurs="0">
      <xs:element name="ASILB" type="FIT">
      <xs:element name="ASILC" type="FIT">
      <xs:element name="ASILD" type="FIT">
      <xs:element name="Rationale" type="xs:string">
    <xs:element name="FRCMethod" minOccurs="0">
      <xs:element name="numberFRCClasses" type="xs:integer" minOccurs="0">
      <xs:element name="FRCValue" type="FIT" minOccurs="0">
      <xs:element name="Divider" type="xs:float" minOccurs="0">
      <xs:element name="Rationale" type="xs:string">
```

Quellcode 5.2: Struktur von *Settings*

Innerhalb des Elements *Settings* wird neben den primitiven Datentypen *String* sowie *Boolean* auch auf eigens definierte Datentypen zurückgegriffen. Die entsprechenden Typendefinitionen befinden sich ebenfalls innerhalb der XSD. Der Datentyp *Abstraction-Level* stellt eine Enumeration mit den Literalen *architectural* und *detailed* zur Beschreibung der Abstraktionsebene nach [ISO26262, 5-7.2] bereit. Hierbei ist lediglich einer dieser beiden Literale als Wert zulässig. Im Kontext der Zielwerte wird der definierte Datentyp *Percentage* genutzt. Dieser beschreibt einen *float*-Wert mit der einschließenden unteren und oberen Schranke 0 bzw. 100. Als Einheit ist das Prozentzeichen „%“ hinterlegt. Der Datentyp *FIT* beschreibt den Wert einer Ausfallrate. In diesem Zusammenhang ist die exkludierte Schranke von 0 definiert. Als Einheit ist *„FIT“* angegeben.

SafetyRequirements: Das Element *SafetyRequirements* stellt einen Container für eine beliebige Anzahl von Elementen der Art *SafetyRequirement* dar und enthält damit alle im Modell hinterlegten Sicherheitsanforderungen. Die Charakteristika in Form eines Namens, einer eindeutigen ID, einer textuellen Beschreibung sowie eines bestimmten ASIL sind als untergeordnete Elemente vorhanden. Zusätzlich existiert unter der Sicherheitsanforderung eine Liste von *RelatedHardwareElements*, für welche die Sicherheitsbezogenheit im Kontext der spezifischen Sicherheitsanforderung über das Attribut *SafetyRelated* angegeben wird. Das entsprechende *Hardware-Element* ist eindeutig über die *ID* zuweisbar. Die Struktur des Elements *SafetyRequirements* ist in Auszug 5.3 gezeigt.

```
<xs:element name="SafetyRequirements">
  <xs:element name="SafetyRequirement" minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Name" type="xs:string">
    <xs:element name="ID" type="xs:string">
    <xs:element name="Description" type="xs:string" minOccurs="0">
    <xs:element name="ASIL" type="ASIL" minOccurs="0">
    <xs:element name="RelatedHardwareElements" minOccurs="0">
      <xs:element name="RelatedHardwareElement" minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Name" type="xs:string">
        <xs:element name="ID" type="xs:string">
        <xs:element name="SafetyRelated" type="xs:boolean">
```

Quellcode 5.3: Struktur von *SafetyRequirements*

Für das Element *ASIL* wurde ebenfalls ein neu definierter Datentyp *ASIL* eingeführt. Dieser stellt eine Enumeration mit den entsprechenden Einstufungen nach [ISO26262, 3-7.4.4.1, 9-5] als Literale bereit.

SafetyMechanisms: Ein Container für alle vorhandenen Sicherheitsmechanismen wird durch das Element *SafetyMechanisms* repräsentiert. Die spezifischen Attribute wie Name, ID sowie entsprechende Diagnoseabdeckungsgrade in Bezug auf RFs bzw. latente MPFs sind untergeordnet eingeschlossen. Die Struktur ist in Auszug 5.4 abgebildet.

```
<xs:element name="SafetyMechanisms">
  <xs:element name="SafetyMechanism" minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Name" type="xs:string" minOccurs="1">
    <xs:element name="ID" type="xs:string" minOccurs="1">
    <xs:element name="DiagnosticCoverage" minOccurs="1"><xs:element name="Name"
        type="xs:string">
      <xs:element name="ResidualFault" type="Percentage">
      <xs:element name="LatentFault" type="Percentage">
```

Quellcode 5.4: Struktur von *SafetyMechanisms*

HWElements: Das *Hardwaredesign* ist durch die einzelnen *Hardware-Elemente* in den *HWElements* abgebildet. Hier sind auch die Fehlerinformationen sowie mögliche Zugehörigkeiten zu Hardwaretypen aus der Bibliothek enthalten. Die Ausfallarten der *Hardware-Elemente* sind als untergeordnete Elemente eingeschlossen. Für die Ausfallarten kann eine Abdeckung durch Sicherheitsmechanismen über eine referenzierte ID eingebracht werden. Zudem sind Klassifikationen im Kontext von referenzierten Sicherheitsanforderungen angegliedert. Weiterhin ist unter den Ausfallarten die Möglichkeit gegeben, Ausgangsabweichungen zur textuellen Beschreibung von Fehlerpropagationen abzubilden, welche für eine darauf aufbauende FTA genutzt werden können. Die Struktur von *HWElements* ist in Auszug 5.5 dargestellt.

```
<xs:element name="HWElements">
  <xs:element name="HWElement" minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Name" type="xs:string">
    <xs:element name="ID" type="xs:string">
    <xs:element name="HWElementTypeRefID" type="xs:string" minOccurs="0">
    <xs:element name="FailureRate" type="FIT" minOccurs="0">
    <xs:element name="Value" type="xs:string" minOccurs="0">
    <xs:element name="FailureModes">
```

```
9         <xs:element name="FailureMode" maxOccurs="unbounded">
10          <xs:element name="Name" type="xs:string">
11          <xs:element name="ID" type="xs:string">
12          <xs:element name="Description" type="xs:string" minOccurs="0">
13          <xs:element name="FailureRateDistribution" type="Percentage">
14          <xs:element name="SafetyMechanismRefID" type="xs:string" minOccurs="0>
15          <xs:element name="Classifications" minOccurs="0">
16            <xs:element name="Classification" minOccurs="0" maxOccurs="unbounded">
17              <xs:element name="SafetyRequirement">
18                <xs:element name="Name" type="xs:string">
19                <xs:element name="RefID" type="xs:string">
20              <xs:element name="SinglePointFault" type="xs:boolean">
21              <xs:element name="MultiplePointFault" type="xs:boolean">
22          <xs:element name="OutputDeviations" minOccurs="0">
23            <xs:element name="OutputDeviation" minOccurs="0" maxOccurs="unbounded">
24              <xs:element name="Name" type="xs:string">
25              <xs:element name="ID" type="xs:string">
26              <xs:element name="LogicalExpression" type="LogicalExpression">
```

Quellcode 5.5: Struktur von *HWElements*

Für die Ausgangsabweichungen wurde ein Datentyp *LogicalExpression* definiert. Dieser spezifiziert die verwendbaren Symbole und beschränkt diese auf Klammern bzw. die beiden logischen Operatoren AND und OR. Die entsprechenden Ausfallarten innerhalb des *Hardware-Elements* sowie Ausgangsabweichungen von weiteren *Hardware-Elementen* werden hierbei über den Namen referenziert.

LibraryHWElementTypes: Als letztes Element auf der obersten Ebene der XSD ist die Bibliothek zu nennen. Diese kann optional eingesetzt werden, um für bestimmte Typen von *Hardware-Elementen* die Fehlerinformationen wie beispielsweise Ausfallraten und Ausfallarten für alle vorhandenen Instanzen zu kapseln. Zusätzlich sind Attribute für referenzierte Dokumente oder Kosten als nicht-technische Aspekte eingebracht. Die Struktur ist in Auszug 5.6 ersichtlich.

```
1  <xs:element name="LibraryHWElementTypes" minOccurs="0">
2    <xs:element name="HWElementType" minOccurs="0" maxOccurs="unbounded">
3      <xs:element name="Name" type="xs:string">
4      <xs:element name="ID" type="xs:string">
5      <xs:element name="Description" type="xs:string" minOccurs="0">
6      <xs:element name="Category" type="xs:string" minOccurs="0">
7      <xs:element name="FailureRate" type="FIT" minOccurs="0">
8      <xs:element name="ReferenceDocument" type="xs:string" minOccurs="0">
9      <xs:element name="ScalingFactor" type="xs:float" minOccurs="0">
10     <xs:element name="Rationale" type="xs:string" minOccurs="0">
11     <xs:element name="Costs" type="xs:float" minOccurs="0">
12     <xs:element name="FailureModes" minOccurs="0">
13       <xs:element name="FailureMode" maxOccurs="unbounded">
14         <xs:element name="Name" type="xs:string">
15         <xs:element name="ID" type="xs:string">
16         <xs:element name="Description" type="xs:string" minOccurs="0">
17         <xs:element name="FailureRateDistribution" type="Percentage">
```

Quellcode 5.6: Struktur von *LibraryHWElementTypes*

5.6 Anwendbarkeit der Hardware-Sicherheitsevaluationen auf unterschiedlichen Abstraktionsebenen

Die von ISO 26262 geforderte Anwendung der Hardware-Sicherheitsevaluationen, beschrieben in Kapitel 2.7, erfordert eine Untersuchung in Bezug zur *Hardwaredesign*-Phase. Im Folgenden werden daher die Evaluationen auf ihre Anwendbarkeit im Kontext der unterschiedlichen Abstraktionsebenen analysiert, siehe Tab. 5.1.

	Evaluation of the hardware architectural metrics			Evaluation of safety goal violations due to random hardware failures		
	single-point fault metric	latent-fault metric	complementary	Evaluation of Probabilistic Metric for random Hardware Failures (PMHF)	OR	Evaluation of each cause of safety goal violation (FRC method)
Hardware Architectural Designs	X	X		X		
Hardware Detailed Designs	X	X				X

Tab. 5.1: Überblick über Anwendbarkeit der geforderten Sicherheitsevaluationen auf unterschiedlichen Abstraktionsebenen

5.6.1 Anwendung auf Hardware-Architekturdesigns

Die Anwendung der „*Evaluation of the hardware architectural metrics*“ ist für das *Hardware-Architekturdesign* geeignet, da durch die Bildung eines relativen Verhältnisses von Ausfallraten die Robustheit des Gesamtdesigns, zunächst als ein vorläufiges Design anzusehen, beurteilt und in einer iterativen Vorgehensweise verfeinert sowie gegen die abgeleiteten Sicherheitsanforderungen überprüft werden kann. Auf diese Weise kann die Einführung von zusätzlichen Sicherheitsmechanismen sowie die Bestimmung von möglichen Schwachstellen im *Hardware-Architekturdesign* unterstützt werden. Auf der Ebene des *Hardware-Architekturdesigns* können die Hardware-Architekturmetriken weiterhin zur Definition von Zielwerten für darunterliegende Abstraktionsebenen und verteilte Entwicklungen genutzt werden [ISO26262, 5-8.2, 8-5.4.3.1f], welche dahingehend im Rahmen des DIA zu kommunizieren sind.

Die Anwendung der FRC-Methode nach Clause 9 ist prinzipiell denkbar, jedoch ist fraglich, welcher Nutzen aus den Ergebnissen gezogen werden könnte. Geht man davon aus, dass die Ausfallarten einer *Hardware-Komponente* durch Grundereignisse im zugehörigen Fehlerbaum repräsentiert werden könnten, so ist diese individuelle Evaluation nicht wirklich belastbar. Auch die innerhalb der ISO 26262 erwähnten *dedizierten Maßnahmen*,

welche sehr technisch detailliert aufgezeigt sind, sprechen nicht für die Anwendung auf einer solchen Abstraktionsebene. Angemerkt sei jedoch, dass sich eine Umsetzung der Methode auf dieser Abstraktionsebene durch die angereicherten Informationen realisieren ließe.

Alternativ ist die PMHF nach Clause 9 als globaler probabilistischer Ansatz anwendbar. Im Vergleich zu den Hardware-Architekturmetriken sowie der FRC-Methode gibt es für die Evaluation einer PMHF keine explizit vorgeschriebene Metrik. Nach [ISO26262, 5-9.2] kann jedoch eine quantitative FTA genutzt werden, um die maximale Verletzung eines Sicherheitsziels zu analysieren. Eine Berechnung der PMHF kann weiterhin nach [ISO26262, 10-8.3.3] durchgeführt werden, jedoch kommen in der Industrie teilweise unterschiedliche angewandte Berechnungsvorschriften zum Einsatz. Häufig ist nicht die in ISO 26262 beschriebene Berechnungsvorschrift, sondern eine vereinfachte Worst-Case Betrachtung im Fokus, auch im Sinne einer ersten Approximation. Gelegentlich fällt hier der Begriff einer „vereinfachten PMHF", vgl. auch [97, S.46], deren Methode insbesondere auf einer Summenbildung der Ausfallraten von SPFs und RFs aller sicherheitsbezogenen *Hardware-Elemente* basiert, $M_{PMHF} \approx \sum_{SR,HW}(\lambda_{SPF} + \lambda_{RF})$, da diese aufgrund ihrer Distanz von n=1 den größten Einfluss auf die Verletzung des Sicherheitsziels haben.

Der Fokus zur Unterstützung der PMHF liegt im Rahmen des vorgestellten Konzeptes auf einer qualitativen und quantitativen FTA, da diese aufgrund ihrer Top-Down Methodik insbesondere in frühen Entwicklungsphasen Anwendung findet. Basierend auf einer quantitativen FTA kann der probabilistische Wert für den Eintritt des Hauptereignisses, welches die Verletzung eines Sicherheitsziels darstellt, bestimmt werden. Dieser kann anschließend gegenüber den vorgeschriebenen Zielwerten nach ISO 26262 im Zuge der PMHF überprüft werden. Weiterhin ist durch die Bestimmung von Minimalschnitten des Fehlerbaums eine automatische Klassifikation von Ausfallarten im Kontext des betrachteten Sicherheitsziels realisierbar, vgl. [ISO26262, 10-AnnexB.4], welche für die Anwendung der Hardware-Architekturmetriken bereitgestellt werden können.

5.6.2 Anwendung auf detaillierte Hardwaredesigns

Auf Ebene von Schaltplänen ist die „*Evaluation of the hardware architectural metrics*" in Form der SPFM und LFM anzuwenden. Die berechneten Werte der Metriken $Metric_{SPF}$ und $Metric_{MPF,L}$ sind in Prozent angegeben und ermöglichen eine iterative Evaluation sowie eine Verifikation des finalen *Hardwaredesigns* gegen die Zielwerte. Hieraus lässt sich der aktuelle Status des *detaillierten Hardwaredesigns* durch die Evaluationen hinsichtlich aller relevanten Sicherheitsziele bestimmen. Zudem ist ersichtlich, inwiefern bereits vorhandene Sicherheitsmechanismen verbessert oder weitere Sicherheitsmechanismen eingeführt werden müssen, um die Zielwerte zu erfüllen. Die Anwendung der Hardware-Architekturmetriken ermöglicht somit eine globale Evaluation der Robustheit des *detaillierten Hardwaredesigns*.

Die FRC-Methode ermöglicht durch Einstufung in Ausfallratenklassen die individuelle Untersuchung der Verletzung einzelner *Hardware-Bauteile* im Kontext des jeweiligen Sicherheitsziels. Dies ist insbesondere hilfreich, um bereits frühzeitig Hotspots im Schaltplan aufzuzeigen und mit entsprechenden Maßnahmen entgegenwirken zu können. Einzelne Ausreißer in Bezug auf die Ausfallrate, welche im Rahmen der Evaluation der Hardware-Architekturmetriken durch das Bilden eines Verhältnisses nicht identifiziert wurden, gehen hieraus eindeutig hervor. Insbesondere bezogen auf die Granularität des *detaillierten Hardwaredesigns* ist eine solche Bewertung einzelner *Hardware-Bauteile* sinnvoll anwendbar. Dies ist bedingt durch die Verwendung statistisch erhobener Ausfallraten aus anerkannten Industriesammlungen. Im Vergleich zum globalen Ansatz einer PMHF ist ein weiterer Vorteil, dass jedes einzelne *Hardware-Bauteil* die entsprechenden Grenzwerte erfüllen muss und dadurch nach [94] die FRC-Methode als robustere Analyse angesehen wird.

Eine Anwendung der PMHF im Rahmen einer qualitativen und quantitativen FTA ist auf dieser Abstraktionsebene nicht angedacht. Vor dem Hintergrund der Systemsicherheit und meist angewandten konservativen Vorgehensweisen würde dies bei der iterativen Absicherung eines komplexen Gesamtsystems unter zeitlichen Aspekten auf solch einer feingranularen Ebene nicht zielführend unterstützen. Möglich wäre hierbei die Evaluation einer „vereinfachten PMHF“ nach den vorgestellten Berechnungsvorschriften.

5.6.3 Abgeleitete Vorgehensweise für die unterschiedlichen Abstraktionsebenen

Basierend auf dem vorgestellten Konzept nach Kapitel 5.2 sowie der Anwendbarkeit der Hardware-Sicherheitsevaluationen auf den unterschiedlichen Abstraktionsebenen konnte eine gesamtheitliche Vorgehensweise für die modellbasierte Entwicklung von funktional sicheren *Hardwaredesigns* erarbeitet werden, siehe Abb. 5.6.

Hierbei sind zwei aufeinander abgestimmte spezifische Vorgehensweisen für die beiden Abstraktionsebenen beschrieben. Zuerst ist das *Hardware-Architekturdesign* strukturell zu modellieren sowie entsprechende Fehlerinformationen und Ausgangsabweichungen zu annotieren. Anschließend können eine entsprechende Fehlerbaumanalyse sowie nachgelagert die Hardware-Sicherheitsevaluationen ausgeführt werden. Innerhalb der Ebene kann iterativ gearbeitet werden, um anschließend für das *detaillierte Hardwaredesign* entsprechende Rahmenbedingungen und Zielwerte vorzugeben. Der Austausch kann exemplarisch basierend auf dem vorgestellten Konzept für ein DIA erfolgen. Das *detaillierte Hardwaredesign* wird zunächst strukturell modelliert und entsprechende Fehlerinformationen annotiert. Da auf Grundlage des vorgestellten Konzeptes für das *detaillierte Hardwaredesign* keine FTA durchgeführt wird, ist die Klassifikation der Ausfallarten für spezifische *Hardware-Bauteile* durch den Entwickler einzubringen, um anschließend die Hardware-Sicherheitsevaluationen anwenden zu können. Evaluationsergebnisse

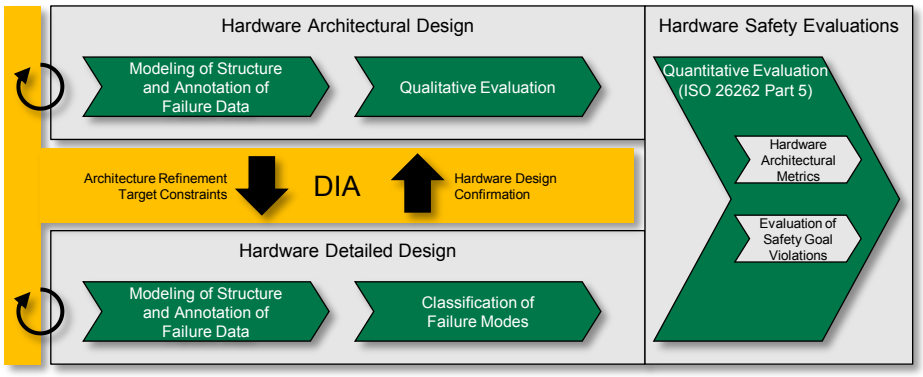


Abb. 5.6: Gesamtheitliche Vorgehensweise für die modellbasierte Entwicklung funktional sicherer Hardwaredesigns

vom *detaillierten Hardwaredesign* können anschließend zur finalen Verifikation bzw. zur Bestätigung der festgelegten Zielwerte genutzt werden.

5.6.4 Anforderungen und Empfehlungen basierend auf ASIL-Klassifizierung

Im Rahmen des Part 4 werden die Anforderungen für das Systemdesign in Clause 7 gelegt. Hierbei sind im Zuge der Untersuchung von Hardware und Software insbesondere Maßnahmen für die Vermeidung von systematischen Fehlern vorzunehmen [ISO26262, 4-7.4.3]. In Bezug auf Hardware sind die Maßnahmen zur Handhabung von zufälligen Ausfällen der Hardware während des Betriebs zu nennen [ISO26262, 4-7.4.4]. Hier werden bereits erste Festlegungen für die zu betrachtenden ASILs in Bezug zu den Hardware-Sicherheitsevaluationen nach Part 5 vorgenommen.

Die in ISO 26262 aufgelisteten Anforderungen sowie Empfehlungen sind grundlegend auf ASIL-A bis ASIL-D anzuwenden [ISO26262, 5-4.3], sofern nichts anderes vermerkt ist. Im Falle einer vorausgehenden ASIL Dekomposition ist dieser anzuwenden. Im Standard eingeklammert dargestellte ASILs, wobei die Klammerung nicht im Sinne der ASIL Dekomposition vorliegt, unterliegen einer Empfehlung und keiner Anforderung.

ASIL-A wird in den Bereichen des modularen *Hardwaredesigns* [ISO26262, 5-Tab.1], der „Hardware design safety analysis“ [ISO26262, 5-Tab.2] und der „Hardware design verification“ [ISO26262, 5-Tab.3] mit berücksichtigt. Zudem wird ASIL-A wieder im Bereich des „Hardware integration and testing“ [ISO26262, 5-10] durch die Tabellen [ISO26262, 5-Tab.10], [ISO26262, 5-Tab.11] und [ISO26262, 5-Tab.12] aufgegriffen.

Die in Part 5 beschriebenen Evaluationen sind nach einer Analyse für ASIL (B), C und D des Sicherheitsziels anzuwenden. Vergleiche hierzu ausgehend von „Measures

for control of random hardware failures during operation“ [ISO26262, 4-7.4.4] auch [ISO26262, 5-6.4.3] für die Evaluationen nach Clause 8 und [ISO26262, 5-6.4.4] für die Evaluationen nach Clause 9. Bezogen auf das *Hardware-Architekturdesign* muss die höchste ASIL-Klassifizierung der implementierten Hardware-Sicherheitsanforderungen vererbt werden [ISO26262, 5-7.4.1].

5.7 Fehlerdatentabelle als zentrales Element

Werden die anzuwendenden Sicherheitsevaluationen unabhängig von der Abstraktionsebene betrachtet, so wird ersichtlich, dass hierfür Informationen aus der Typenbibliothek und dem Datenmodell akquiriert werden müssen, siehe Abb. 5.7. Diese basieren auf einem gemeinsamen Anteil, welcher unabhängig von der Abstraktionsebene und der durchzuführenden Evaluation betrachtet und als Ausgangsbasis für die einzelnen Berechnungen genutzt werden kann. Die dargestellte Strukturierung und damit verbundene Modularisierung dient einer Hervorhebung der Zusammenhänge und Gleichanteile der Evaluationen und erleichtert die Umsetzung der vorgestellten Vorgehensweisen.

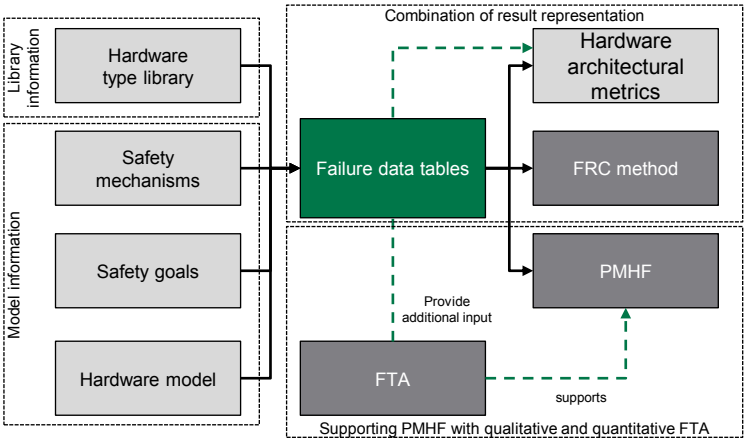


Abb. 5.7: Fehlerdatentabelle als zentrales Element der Hardware-Sicherheitsevaluationen

Somit wird das Konzept von Fehlerdatentabellen, angelehnt an FMEDA-Formblätter, als zentrales Element für ein gemeinsames Fundament vorgeschlagen, welche alle relevanten Informationen aus dem Modell im Kontext der zu betrachtenden Sicherheitsziele strukturiert. Zugleich können bereits vorbereitende Schritte für die Evaluationen, wie beispielsweise Konsistenzprüfungen auf den befüllten Fehlerdatentabellen, durchgeführt werden. Diese sind notwendig, um insbesondere zu frühen Designphasen die in den Bibliotheken hinterlegten, möglicherweise unvollständigen Angaben vor den Berechnungen im Rahmen der Sicherheitsevaluationen abzusichern.

Ausgehend von einer befüllten Fehlerdatentabelle im Kontext eines jeweiligen Sicherheitsziels können die entsprechenden Evaluationen darauf aufsetzend durchgeführt werden. Eine Anwendung der Hardware-Architekturmetriken kann unabhängig von der Abstraktionsebene basierend auf der Fehlerdatentabelle erfolgen. Die FTA als Unterstützung der PMHF sowie die FRC-Methode nehmen hierbei eine Sonderrolle ein, da diese Evaluationen lediglich auf einer spezifischen Abstraktionsebene durchgeführt werden. Weiterhin können durch die qualitative Analyse des Fehlerbaumes zusätzliche Informationen hinsichtlich der Klassifikation der Ausfallarten in die Fehlerdatentabelle eingespeist werden.

5.7.1 Aufbau der Fehlerdatentabelle

In Abb. 5.8 ist vereinfacht der Bezug zu den Fehlerinformationen aus dem Datenmodell ersichtlich, mit denen die Fehlerdatentabelle initial befüllt werden kann. Durch Aufgreifen von verknüpften Sicherheitsmechanismen und den Berechnungen der einzelnen spezifischen Ausfallraten für die Ausfallarten der *Hardware-Elemente* wird die Ausgangsbasis für die Hardware-Sicherheitsevaluationen gelegt.

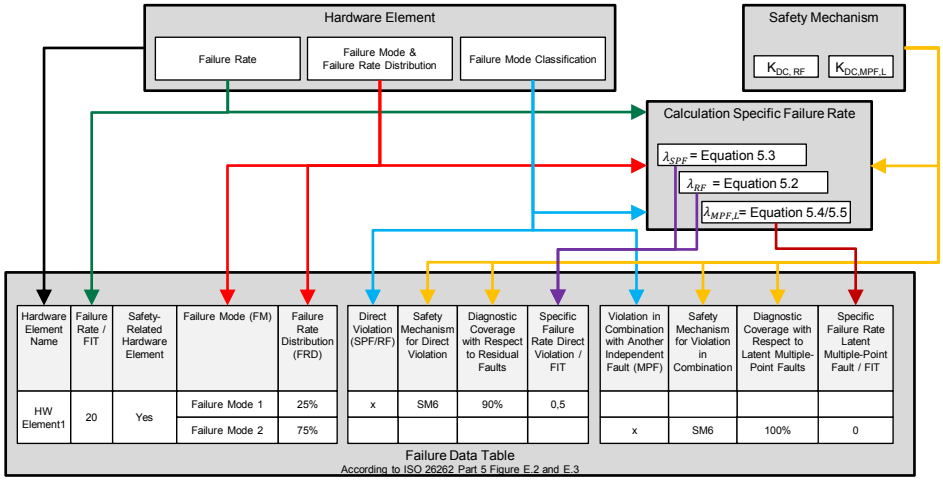


Abb. 5.8: Aufbau und Befüllung der Fehlerdatentabelle

Im Rahmen der Fehlerdatentabelle werden die einzelnen Zeilen zur Auflistung aller dem Sicherheitsziel zugehörigen *Hardware-Elemente* verwendet. Hinsichtlich der Spalten lässt sich der grundsätzliche Aufbau dreigeteilt in folgende Bereiche zusammenfassen. Der linke Tabellenbereich spiegelt die zu den eingetragenen *Hardware-Elementen* zugehörigen Fehlerinformationen einschließlich der Sicherheitsbezogenheit wider. Im Rahmen des mittleren Tabellenbereichs werden Informationen zu einer potentiellen direkten

Verletzung des Sicherheitsziels für die einzelnen Ausfallarten festgehalten. Hierbei sind SPFs und RFs gemeinsam betrachtet. Der rechte Tabellenbereich greift eine potentielle Verletzung in Kombination mit anderen Fehlern im Kontext von latenten MPFs auf. Die jeweiligen Berechnungsvorschriften in Bezug auf die spezifischen Ausfallraten werden gesondert im folgenden Kapitel behandelt.

Durch die Verwendung der Fehlerdatentabelle als zentrales Element können insbesondere zu jedem Zeitpunkt der Designphasen sowohl die zu einem Sicherheitsziel zugehörigen *Hardware-Elemente* und deren sicherheitsbezogene Einstufung erfasst werden als auch gezielt händisch Änderungen im Design basierend auf den dargestellten Informationen vorgenommen werden. Dies ermöglicht eine iterative Verbesserung des Designs, sei es hinsichtlich der geforderten Zielwerte oder aus Robustheitsgründen.

Ein weiterer Vorteil hinsichtlich der Fehlerdatentabelle ist, dass Evaluationsergebnisse zurückgespiegelt werden können, um nicht nur eine einfache Nachvollziehbarkeit zu erreichen, sondern auch eine zügige Optimierung des Designs durch beispielsweise Einführung weiterer Sicherheitsmechanismen zu bewirken. Dies kann insbesondere für eine kombinierte Darstellung mit den Ergebnissen der Hardware-Architekturmetriken und der FRC-Methode angewandt werden, vgl. Abb. 5.7.

5.7.2 Befüllung der Fehlerdatentabelle

Basierend auf den Verknüpfungen zwischen dem Sicherheitsziel und den *Hardware-Elementen*, werden diese einzeln gelistet und die jeweilige Einstufung der Sicherheitsbezogenheit eingebracht. Die für die *Hardware-Elemente* hinterlegten Fehlerinformationen, wie Ausfallrate, Ausfallarten und Ausfallratenverteilung, werden über die Typisierung des *Hardware-Elements* aus den entsprechenden Bibliotheken übernommen. Des Weiteren werden bestehende Verknüpfungen zwischen den Sicherheitsmechanismen und den *Hardware-Elementen* untersucht. Falls vorhanden, können diese in der Fehlerdatentabelle vermerkt und die entsprechenden Werte für die Diagnoseabdeckungsgrade entnommen und eingetragen werden.

Vor dem Hintergrund, dass zu frühen Entwurfsphasen nicht nur die Designs an sich, sondern gegebenenfalls die in den Bibliotheken hinterlegten Informationen ungenau oder unvollständig sind, werden gezielte Überprüfungen auf Konsistenz sowie Plausibilität bei Verwendung der einzelnen Bibliothekstypen durchgeführt. Die Anwendung erfolgt zeitlich gesehen vor den Sicherheitsevaluationen und erfordert die Überprüfung von Informationen sowohl aus der Bibliothek als auch des Modells. So werden hinsichtlich der Bibliothek Konsistenzprüfungen durchgeführt, welche beispielsweise die Ausfallratenverteilung absichert, da diese spezifisch für eine Ausfallart einen prozentualen Wert zwischen 0 und 100 besitzen muss. Weiterhin muss die Summe der Ausfallratenverteilungen über alle Ausfallarten eines *Hardware-Element*-Typs gleich 100 Prozent sein. Als Überprüfung auf Ebene der *Hardware-Elemente* gilt es zu untersuchen, ob eine Klassifikation

für jede relevante Ausfallart eingetragen wurde, um die Berechnung der entsprechenden spezifischen Ausfallrate zu ermöglichen, welche im Folgenden genauer beschrieben wird.

5.7.3 Berechnung von spezifischen Ausfallraten

Vor der Durchführung der Hardware-Sicherheitsevaluationen können vorbereitende Schritte unternommen werden, welche zudem bereits beim Designen der Hardware unterstützen. Dies betrifft die Berechnung der spezifischen Ausfallraten aller Ausfallarten von sicherheitsbezogenen *Hardware-Elementen* basierend auf den eingeholten Bibliotheks- und Modellinformationen. Die spezifische Ausfallrate hängt dabei von der Gesamtausfallrate des *Hardware-Elements*, der Ausfallart (FM), der Ausfallratenverteilung (FRD) und vor allem von der Klassifikation der Ausfallart für das spezifische Sicherheitsziel sowie den zugewiesenen Sicherheitsmechanismen ab. Diese Informationen sind bereits in der Fehlerdatentabelle enthalten, worauf basierend die spezifischen Ausfallraten wie folgt berechnet werden können.

Ist die Ausfallart eines sicherheitsbezogenen *Hardware-Elements* als sicherer Fehler (SF) klassifiziert, kann die spezifische Ausfallrate durch das Produkt aus der Gesamtausfallrate des *Hardware-Elements* und der Ausfallratenverteilung der Ausfallart berechnet werden, siehe Gleichung 5.1. Hinsichtlich von SFs wird eine Unterscheidung in Bezug von nicht-sicherheitsbezogenen und sicherheitsbezogenen *Hardware-Elementen* vorgenommen. Hierbei werden die Ausfallarten von nicht-sicherheitsbezogenen *Hardware-Elementen* im Rahmen der Evaluation nicht betrachtet, vgl. [ISO26262, 5-B.2].

$$\lambda_{SF} = \lambda_{HW} \cdot FRD_{FM} \tag{5.1}$$

Wenn eine Ausfallart als Restfehler (RF) klassifiziert ist, wird die spezifische Ausfallrate durch das Produkt der Gesamtausfallrate des *Hardware-Elements* multipliziert mit der Ausfallratenverteilung und dem Anteil des Fehlers, welcher nicht durch den Sicherheitsmechanismus abgedeckt wird, angegeben, siehe Gleichung 5.2. Hier wird der Diagnoseabdeckungsgrad des Sicherheitsmechanismus in Bezug auf Restfehler verwendet, welcher in Prozent ausgedrückt wird.

$$\lambda_{RF} = \lambda_{HW} \cdot FRD_{FM} \cdot \left(1 - \frac{K_{DC,RF}}{100}\right) \tag{5.2}$$

Die Berechnung der Ausfallrate für einen Einzelfehler (SPF) nach Gleichung 5.3 ist ein Sonderfall der Gleichung 5.2, wenn kein Sicherheitsmechanismus vorhanden ist und damit der Diagnoseabdeckungsgrad 0 Prozent ist.

$$\lambda_{SPF} = \lambda_{HW} \cdot FRD_{FM} \tag{5.3}$$

Wenn eine Ausfallart ausschließlich als Mehrfachfehler (MPF) klassifiziert ist, kann nur der latente Mehrfachfehler das Sicherheitsziel verletzen. Dies repräsentiert den Anteil des Fehlers, welcher nicht durch einen Sicherheitsmechanismus abgedeckt ist. Hier wird der Diagnoseabdeckungsgrad bezüglich latenter Mehrfachfehler des Sicherheitsmechanismus betrachtet. Die Berechnung der spezifischen Ausfallrate ergibt sich aus Gleichung 5.4.

$$\lambda_{MPF,L} = \lambda_{HW} \cdot FRD_{FM} \cdot \left(1 - \frac{K_{DC,MPF,L}}{100}\right) \tag{5.4}$$

Wird die Ausfallart als Mehrfachfehler klassifiziert, während diese zusätzlich das Sicherheitsziel auch als Einzel- oder Restfehler verletzt, wird die Ausfallrate der direkten Verletzung (Einzel- oder Restfehler) auf gleiche Weise wie in Gleichung 5.2 oder 5.3 berechnet. Diese spezifische Ausfallrate der direkten Verletzung kann nicht mehr zur spezifischen Ausfallrate des latenten Mehrfachfehlers beitragen, vgl. [ISO26262, 5-Fig.E.2.note6]. Die Berechnung der Ausfallrate für latente Mehrfachfehler wird in diesem Fall in Gleichung 5.5 aufgezeigt.

$$\lambda_{MPF,L_{SPF,RF}} = [(\lambda_{HW} \cdot FRD_{FM}) - \lambda_{SPF,RF}] \cdot \left(1 - \frac{K_{DC,MPF,L}}{100}\right) \tag{5.5}$$

Die spezifischen Ausfallraten werden für alle Ausfallarten von sicherheitsbezogenen *Hardware-Elementen* eines zu bewertenden Sicherheitsziels berechnet. Alle *Hardware-Elemente*, ihre Gesamtausfallrate, Ausfallarten, Sicherheitsbezogenheit, spezifische Ausfallraten und Sicherheitsmechanismen mit Diagnoseabdeckungsgrad geben einen Überblick über den Beitrag jedes einzelnen *Hardware-Elements* in Bezug auf die Verletzung eines Sicherheitsziels. Hinsichtlich dem in Kapitel 5.7.1 aufgegriffenen Aufbau der Fehlerdatentabelle sind die Gleichungen 5.2 und 5.3 für RFs und SPFs dem mittleren Tabellenbereich und die Gleichungen 5.4 und 5.5 für latente MPFs dem rechten Tabellenbereich zuzuordnen.

Auf Ebene der *Hardware-Architekturdesigns* ist es aufgrund des Entwicklungsstadiums teilweise sinnvoll, statt einer Gesamtausfallrate für ein *Hardware-Element* λ_{HW} und einer spezifischen Ausfallratenverteilung für jede Ausfallart FRD_{FM}, direkt Ausfallraten für die einzelnen Ausfallarten anzugeben. Hierbei sind allerdings noch keine Auswirkungen von Sicherheitsmechanismen miteinbezogen, die spezifische Ausfallrate muss gesondert bestimmt werden. Da die Summe der Ausfallratenverteilungen über alle Ausfallarten eines *Hardware-Elements* 100 % ergeben muss, ist eine Umrechnung in beide Richtungen jederzeit möglich, siehe auch Gleichungen 5.6 bzw. 5.7.

$$\lambda_{FM} = \lambda_{HW} \cdot FRD_{FM} \tag{5.6}$$

$$\lambda_{HW} = \sum_{FM,HW} \lambda_{FM} \tag{5.7}$$

5.8 Evaluation der Hardware-Architekturmetriken

Der Clause 8 in Part 5 fordert die Evaluation der Hardware-Architekturmetriken, um die Gesamtrobustheit des *Hardwaredesigns* gegenüber spezifischen Ausfällen der Hardware zu verifizieren. Diese sind auf beiden Abstraktionsebenen nach [ISO26262, 5-8.2] angeraten und können iterativ während den einzelnen Designphasen angewandt werden. Aus diesem Grund werden die Hardware-Architekturmetriken vor den einzelnen Vorgehensweisen, welche spezifisch für die Abstraktionsebenen ausgeprägt sind, allgemein vorgestellt.

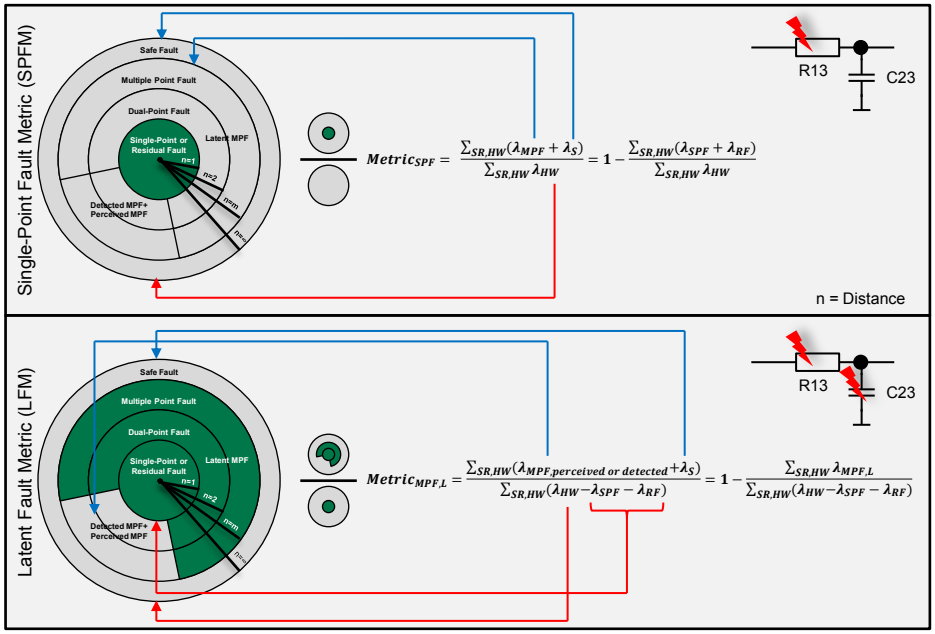


Abb. 5.9: Hardware-Architekturmetriken nach ISO 26262

Die Hardware-Architekturmetriken setzen sich aus der SPFM und der LFM zusammen, wie in Abb. 5.9 ersichtlich. Diese sind angelehnt an die SFF aus der IEC 61508 und stellen ein relatives Verhältnis von gefährlichen Ausfällen zur Gesamtheit der Ausfälle dar. Da ISO 26262 im Vergleich eine detailliertere Klassifikation der Ausfallarten bereitstellt, sind zwei verschiedene Metriken vorgeschlagen. Die SPFM untersucht die Auswirkungen von SPFs und RFs auf das System, wohingegen die LFM die Auswirkungen von latenten MPFs betrachtet. Die Metriken müssen für jedes einzelne Sicherheitsziel durchgeführt werden und erfordern hierzu die Fehlerinformationen aus der Hardware-Bibliothek, die Einstufung der Sicherheitsbezogenheit jedes einzelnen *Hardware-Elements*, die Verknüp-

fung von vorhandenen Sicherheitsmechanismen sowie die Klassifikationen der Ausfallarten und daraus berechneten spezifischen Ausfallraten.

5.8.1 Berechnungsvorschriften

Die SPFM betrachtet sowohl die Einzelfehler (SPFs) als auch die Restfehler (RFs) im Kontext der Gesamtausfallrate über alle als sicherheitsbezogen eingestuften *Hardware-Elemente*, um eine Aussage über die Robustheit des *Hardwaredesigns* in Bezug auf direkte Verletzungen des Sicherheitsziels zu geben, siehe Gleichung 5.8c. Hierbei werden analog zur Fehlerdatentabelle die SPFs und RFs gemeinsam betrachtet, da beide zu einer direkten Verletzung des Sicherheitsziels führen. Das Ergebnis der Metrik wird in Prozent angegeben. Ein hoher prozentualer Wert der Metrik impliziert, dass der Anteil an Fehlern, welche das Sicherheitsziel direkt verletzen, niedrig ist, vgl. [ISO26262, 5-AnnexC.2]. Dies kann durch ein robustes Design an sich oder durch eine entsprechende Abdeckung von möglichen Verletzungen des Sicherheitsziels mit Hilfe eingebrachter Sicherheitsmechanismen erreicht werden.

$$Metric_{SPF} = \frac{\sum_{SR,HW} (\lambda_{MPF} + \lambda_S)}{\sum_{SR,HW} \lambda_{HW}} \tag{5.8a}$$

$$= 1 - \frac{\sum_{SR,HW} (\lambda_{HW} - \lambda_{MPF} - \lambda_S)}{\sum_{SR,HW} \lambda_{HW}} \tag{5.8b}$$

$$= 1 - \frac{\sum_{SR,HW} (\lambda_{SPF} + \lambda_{RF})}{\sum_{SR,HW} \lambda_{HW}} \tag{5.8c}$$

Die LFM ist das Pendant hinsichtlich der Robustheit gegenüber latenter Mehrfachfehler. Für die LFM werden die spezifischen Ausfallraten aller latenten MPFs in den Fokus gerückt. Da direkte Verletzungen durch SPFs und RFs bereits in der SPFM berücksichtigt wurden, werden diese von den Gesamtausfallraten der *Hardware-Elemente* abgezogen und nicht für die Berechnung der LFM berücksichtigt, siehe Gleichung 5.9c.

$$Metric_{MPF,L} = \frac{\sum_{SR,HW} (\lambda_{MPF,DP} + \lambda_S)}{\sum_{SR,HW} (\lambda_{HW} - \lambda_{SPF} - \lambda_{RF})} \tag{5.9a}$$

$$= 1 - \frac{\sum_{SR,HW} (\lambda_{HW} - \lambda_{MPF,DP} - \lambda_S)}{\sum_{SR,HW} (\lambda_{HW} - \lambda_{SPF} - \lambda_{RF})} \tag{5.9b}$$

$$= 1 - \frac{\sum_{SR,HW} \lambda_{MPF,L}}{\sum_{SR,HW} (\lambda_{HW} - \lambda_{SPF} - \lambda_{RF})} \tag{5.9c}$$

Auch hier repräsentiert ein hoher prozentualer Wert der Metrik einen niedrigen Anteil an latenten Mehrfachfehlern im Design. Durch Sicherheitsmechanismen abgedeckte Fehler oder Fehler, welche durch den Fahrer bei einer möglichen Verletzung des Sicherheitsziels wahrgenommen werden, sind abgesichert. Hierdurch können höhere Werte der LFM

erreicht werden, vgl. [ISO26262, 5-AnnexC.3]. Angemerkt sei, dass wenn ein Mehrfachfehler sowohl durch einen Sicherheitsmechanismus abgedeckt als auch vom Fahrer innerhalb eines vorgeschriebenen Zeitrahmens wahrgenommen wird, die Klassifikation entweder als abgedeckter oder wahrgenommener Mehrfachfehler erfolgen muss. Eine gleichzeitige Einstufung in beide Klassifikationen ist nicht gestattet, da ansonsten das Ergebnis der LFM verfälscht wird [ISO26262, 10-8.1.5.note].

Durch die Anwendung der Hardware-Architekturmetriken kann das *Hardwaredesign* evaluiert und zugleich mit vorherigen Versionen im Hinblick auf funktionale Sicherheit verglichen werden. Hierbei steht nicht nur das Metrikergebnis im Vordergrund, sondern auch die Identifikation, inwiefern getroffene Sicherheitsmaßnahmen in Form von Sicherheitsmechanismen Auswirkungen auf die im Design vorliegenden Wirkketten haben. Gleichzeitig können durch die Einführung von Sicherheitsmechanismen indirekt systematische Fehler abgesichert werden. Die Hardware-Architekturmetriken liefern objektive Ergebnisse für den Vergleich unterschiedlicher *Hardwaredesigns* mit gleicher Funktionalität. Dies wird unter anderem auch indirekt durch die vorliegende Beschreibung des *Hardwaredesigns* und die Fehlerdatentabellen unterstützt.

5.8.2 Ableitung der Zielwerte und Verifikation

Die berechneten Werte der Hardware-Architekturmetriken können im nächsten Schritt gegen Zielwerte verifiziert werden, welche von der ASIL-Klassifizierung der zugehörigen Sicherheitsziele abhängen. Um gegen Zielwerte prüfen zu können, müssen diese für die unterschiedlichen ASILs zunächst definiert und festgelegt werden. Die quantitativen Referenzzielwerte können aus unterschiedlichen Quellen stammen. Sollten bereits von vergleichbaren und zudem vertrauenswürdigen *Hardwaredesigns* Zielwerte aus der Anwendung der Hardware-Architekturmetriken vorliegen, so können diese herangezogen werden. Alternativ kann auf Vorgabewerte der ISO 26262 zurückgegriffen werden, siehe Tab. 5.2. Diese sind aufgeschlüsselt nach dem ASIL und stehen im Kontext von Sicherheit für ein ausreichend hohes Maß an Integrität. Hinsichtlich der Anwendung der SPFM ist die Überprüfung eines Sicherheitszieles mit Einstufung ASIL-B als optional zu betrachten, bei der LFM wird zudem ASIL-C als optional angesehen, vgl. [ISO26262, 5-8.4.7, 5-8.4.8]. Die Erfüllung der Hardware-Architekturmetriken lässt sich aus den zugehörigen Zielwerten für das ASIL des Sicherheitsziels und dem Ergebnis der jeweiligen SPFM und LFM bestätigen.

		ASIL B	ASIL C	ASIL D
Single-point fault metric	Default	≥90 %	≥97 %	≥99 %
	Derivation	similar well-trusted design principles		
Latent-fault metric	Default	≥60 %	≥80 %	≥90 %
	Derivation	similar well-trusted design principles		

Tab. 5.2: Referenzzielwerte für die Hardware-Architekturmetriken nach ISO 26262

5.9 Untersuchung des Hardware-Architekturdesigns

Im Folgenden wird die iterative Vorgehensweise für die Modellierung sowie anschließende Evaluation des *Hardware-Architekturdesigns* im Kontext der Entwicklung funktional sicherer Hardware detailliert vorgestellt. Hierzu müssen die Systemdesign-Spezifikation und das technische Sicherheitskonzept nach [ISO26262, 4] als Voraussetzung für ein initiales *Hardwaredesign* bereitgestellt werden, siehe Abb. 5.10. Im Rahmen des Hardware-Entwicklungsprozesses muss das *Hardware-Architekturdesign* iterativ in Bezug auf die Sicherheitsanforderungen analysiert und überarbeitet werden, bevor das *detaillierte Hardwaredesign* auf Ebene von elektronischen Schaltplänen initiiert werden kann. Die Beschreibung der Vorgehensweise für das *detaillierte Hardwaredesign*, welche auszugsweise in Abb. 5.10 dargestellt ist, wird anschließend in Kapitel 5.10 detailliert beschrieben und ist daher an dieser Stelle nicht im Fokus.

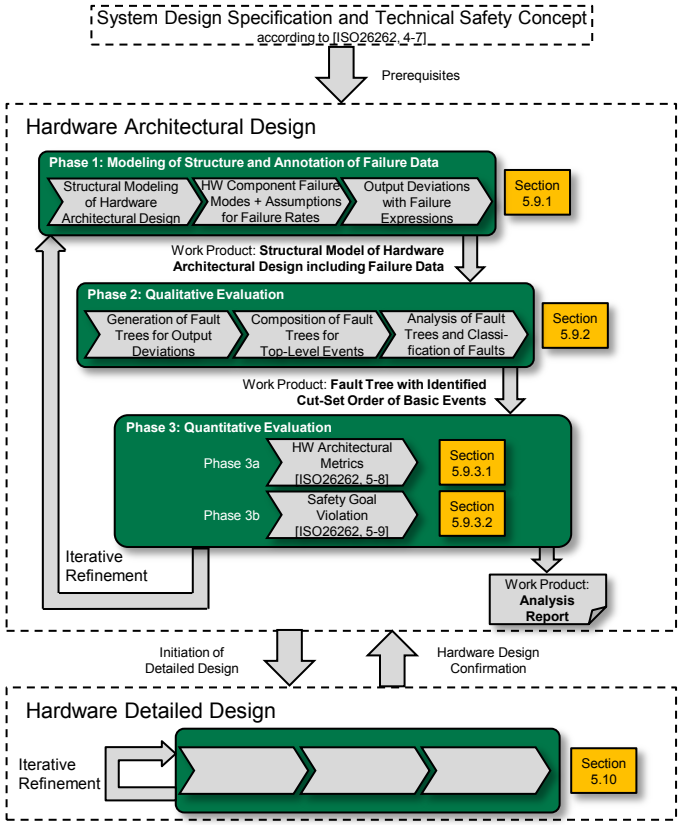


Abb. 5.10: Vorgehensweise für die Evaluation von Hardware-Architekturdesigns

Um die Sicherheitsevaluationen auf Ebene des *Hardware-Architekturdesigns* durchzuführen, wird der Fokus auf einen deduktiven Ansatz mittels einer Fehlerbaumanalyse gelegt, als eine der qualitativen und quantitativen Methoden, welche in ISO 26262 [ISO26262, 9-8] empfohlen werden. Vorgeschlagen wird eine dreiphasige Vorgehensweise für die Sicherheitsevaluation von *Hardware-Architekturdesigns*, welche iterativ angewendet werden kann. Somit wird unter anderem eine Budgetierung von zulässigen Fehleranteilen sowie die Angabe von Zielwerten im Zuge des Systemdesigns unterstützt. Diese sind auch im Rahmen eines DIA erwähnt und können für die anschließende Initiierung des *detaillierten Hardwaredesigns* genutzt werden. Die einzelnen Schritte für das *Hardware-Architekturdesign* werden in den folgenden Abschnitten beschrieben.

5.9.1 Modellierung der Struktur und Annotation von Fehlerdaten

Zuerst wird der Fokus auf die strukturelle Beschreibung des *Hardware-Architekturdesigns* einschließlich der Annotation von Fehlerinformationen gelegt. Diese strukturelle Modellierung basiert auf den Anforderungen des technischen Sicherheitskonzepts. Die korrespondierenden *Hardware-Komponenten* und ihre Verbindungen müssen definiert und in einem Modell erfasst werden, welches ein *Hardwaredesign* auf einer höheren Abstraktionsebene repräsentiert. Existierende Architekturbeschreibungssprachen, wie in Kapitel 2.3 vorgestellt, ermöglichen diese strukturelle Beschreibung. Es wird zu einer Differenzierung der Darstellung vom *detaillierten Hardwaredesign* auf der Ebene von elektronischen Schaltplänen geraten.

Um die geforderten Hardware-Sicherheitsevaluationen durchzuführen, müssen die Fehlerinformationen der *Hardware-Komponenten*, angebunden an die strukturelle Modellierung, bereitgestellt werden. Hierzu wurde in Kapitel 5.4 ein Metamodell zur Fehlerbeschreibung vorgeschlagen. Die Ausfallraten für das *Hardware-Architekturdesign* stellen in dieser frühen Designphase meist eine Annahme dar und können beispielhaft aus vorangegangenen Designs, firmenspezifischen Datenbanken oder Expertenwissen abgeleitet werden. Die beschriebenen Ausfallarten der *Hardware-Komponenten* repräsentieren Grundereignisse der zugehörigen Fehlerbäume. Hierfür ist es teilweise sinnvoll, statt einer Ausfallratenverteilung für die Ausfallarten sowie einer Gesamtausfallrate für die *Hardware-Komponenten*, direkt Ausfallraten der einzelnen Ausfallarten anzugeben, vgl. Kapitel 5.7.3.

Für eine modellbasierte FTA müssen zusätzlich die Ausgangsabweichungen (output deviations) der *Hardware-Komponenten* spezifiziert werden. Diese repräsentieren einen Teilfehlerbaum im Kontext einer *Hardware-Komponente* und beschreiben die Propagation der über die Grundereignisse abgeleiteten Ausfallarten. Die Ausgangsabweichungen werden an die Ausgänge einer jeweiligen *Hardware-Komponente* annotiert und sind damit dieser zugeordnet. Somit wird die lokale modellbasierte Beschreibung von Teilfehlerbäumen, auch im Kontext verteilter Entwicklung, unterstützt. Die zugeordneten Fehlerterme stellen logische Kombinationen von Ausfallarten der jeweiligen

Hardware-Komponenten sowie Ausgangsabweichungen von vorausgehenden verbundenen *Hardware-Komponenten* dar.

Jeder Ausgangsabweichung ist exakt ein logischer Fehlerterm zugeordnet, die einzelnen *Hardware-Komponenten* können mehrere Ausgangsabweichungen besitzen. Die Beschreibung der logischen Fehlerterme kann durch die Verwendung eines grafisch notierten Modellierungsansatzes durchgeführt werden, wie im Beitrag [Adler2012b] dargestellt, oder vergleichbar mit dem Ansatz HiP-HOPS [68], welcher eine textuelle Beschreibung für die logischen Fehlerterme ermöglicht. Zusätzlich können Ausgangsabweichungen als Ende des funktionalen Pfades innerhalb des Systems markiert werden, um damit im Kontext dieses Ansatzes die Verletzung eines Sicherheitsziels darzustellen. Diese Ausgangsabweichungen stellen die Hauptereignisse im Fehlerbaum dar.

Modellierung der Fehlerpropagation durch grafische Annotation: Ein grafischer Ansatz für die Beschreibung der Ausgangsabweichungen ist in [Adler2012b] vorgestellt. Im Rahmen dieser Arbeit wurde die FTA auf Systemebene bereitgestellt, daher wurde eine Modellierung sowie die grafisch notierte Beschreibung auf einer logischen Abstraktionsebene vorgenommen, um das Fehlerverhalten zunächst abstrahiert von Hardware und Software beschreiben zu können. Ein dahinterstehendes Hierarchisierungskonzept für große Architekturmodelle war hierfür zwingend erforderlich. Eine vereinfachte Darstellung des Konzeptes ist in Abb. 5.11 dargestellt.

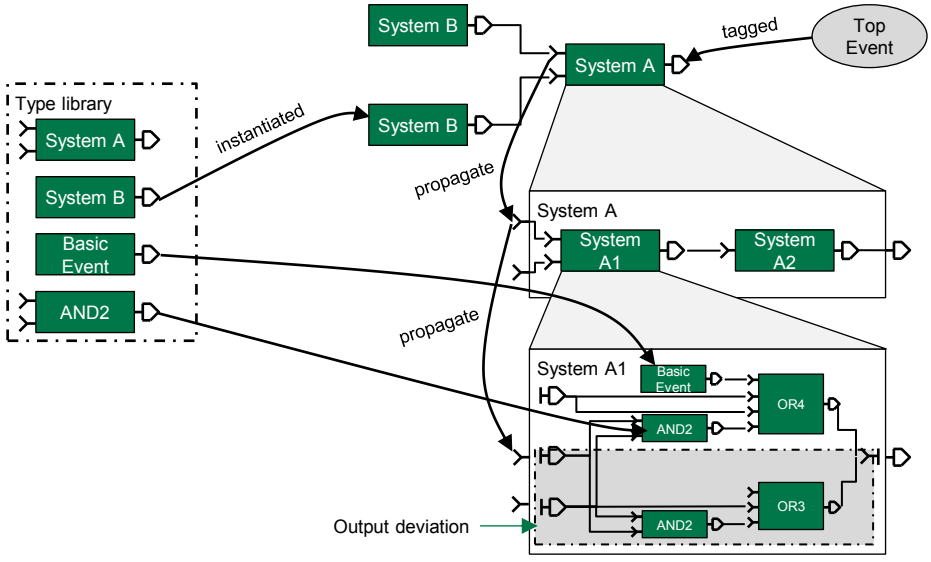


Abb. 5.11: Grafisch notierte Fehlermodellierung von Fehlerpropagationen

Auf Grundlage einer Typenbibliothek für Systemelemente, Grundereignisse und logische Gatter kann an die Ausgänge der Systeme eine entsprechende Ausgangsabweichung annotiert werden. Die Beschreibung der Ausgangsabweichung erfolgt grafisch, basierend auf den meistverwendeten logischen Gattern *UND* und *ODER*. Im Rahmen der Entwicklung funktional sicherer Hardware kann das vorgestellte Konzept auf die Abstraktionsebene für *Hardwaredesigns* übertragen werden.

Modellierung der Fehlerpropagation durch textuelle Annotation: Alternativ zur grafischen Modellierung der logischen Fehlerterme kann die Beschreibung in rein textueller Form erfolgen, angelehnt an den Ansatz von HiP-HOPS. Bei diesem enthält der logische Fehlerterm eine textuelle Verknüpfung der internen Ausfallarten der Komponenten sowie vorausgehenden Ausgangsabweichungen. Im Rahmen der Propagation erfolgt das Referenzieren über Namen. Daher ist bei der Umsetzung auf eine eindeutige Namensgebung insbesondere für die *Hardware-Komponenten* zu achten, innerhalb der Komponenten müssen die einzelnen Ausfallarten eindeutig benannt sein.

5.9.2 Qualitative Sicherheitsevaluation

Die zweite Phase beschreibt die qualitative Evaluation des *Hardware-Architekturdesigns*. Diese berücksichtigt das Verhalten des gesamten *Hardwaredesigns* im Falle des Eintritts einzelner Ausfallarten. In diesem Kontext ist der Fehlerbaum für die Verletzung eines spezifischen Sicherheitsziels zu betrachten. Die annotierten Ausgangsabweichungen mit ihren logischen Fehlertermen ermöglichen ein automatisches Zusammensetzen des Gesamtfehlerbaums. Das Hauptereignis des Fehlerbaums wird durch eine markierte Ausgangsabweichung repräsentiert. Die Analyse des Fehlerbaums kann wiederum die Klassifikation von Ausfallarten im Kontext des Sicherheitsziels für die anschließenden quantitativen Hardware-Sicherheitsevaluationen unterstützen.

Als Vorbereitung für die qualitative Fehlerbaumanalyse müssen aus den logischen Fehlertermen der annotierten Ausgangsabweichungen einzelne Teilfehlerbäume generiert werden. Für die Generierung ist die Art der Annotation (grafisch oder textuell) gleichzusetzen, womit die nächsten Schritte übergreifend beschrieben werden.

In einem ersten Schritt wird jede der Ausgangsabweichungen einer *Hardware-Komponente* in einen einzelnen Teilfehlerbaum transferiert. Die entsprechenden Teilfehlerbäume der exemplarischen Ausgangsabweichungen *OutputDeviation-Amplifying* sowie *OutputDeviation-ComputationControlling* sind in Abb. 5.12 ersichtlich. Die generierten Teilfehlerbäume enthalten als Grundereignisse entweder interne Ausfallarten der *Hardware-Komponente* oder den Bezug zu Ausgangsabweichungen von vorausgehenden *Hardware-Komponenten*.

Nach der Generierung aller relevanten Teilfehlerbäume für die *Hardware-Komponenten*, kann der Gesamtfehlerbaum, beginnend von einer markierten Ausgangsabweichung als Hauptereignis aus den lokalen Teilfehlerbäumen zusammengesetzt werden. Die

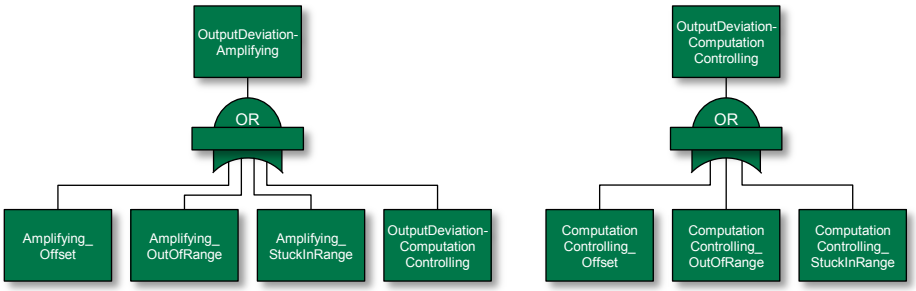


Abb. 5.12: Exemplarische Darstellung einzelner Teilfehlerbäume der Ausgangsabweichungen [D333b]

weiteren Ausgangsabweichungen, welche in Beziehung zum Hauptereignis stehen, enthalten die logische Verbindung unter anderem von Ausgangsabweichungen der vorausgehenden *Hardware-Komponenten*, was ein Zusammensetzen des Gesamtfehlerbaums ermöglicht. Hinsichtlich des dargestellten Beispiels wurde die Ausgangsabweichung *OutputDeviation-Amplifying* als Hauptereignis markiert und enthält den Bezug zur Ausgangsabweichung *OutputDeviation-ComputationControlling*. Diese beiden Teilfehlerbäume können nun zu einem Gesamtfehlerbaum zusammengesetzt werden, wie in Abb. 5.13 dargestellt.

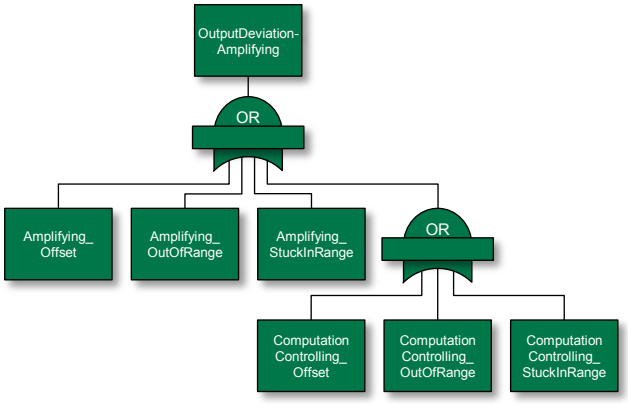


Abb. 5.13: Exemplarische Darstellung des zusammengesetzten Gesamtfehlerbaums für die markierte Top-Level Ausgangsabweichung „OutputDeviation-Amplifying“ [D333b]

In dem dargestellten Gesamtfehlerbaum sind nun als Grundereignisse lediglich die Ausfallarten der unterschiedlichen *Hardware-Komponenten* vorhanden. Die Ausgangs-

abweichungen zwischen den Komponenten werden dabei durch das Zusammensetzen des Gesamtfehlerbaums eliminiert.

Alternativ bietet der Ansatz die Möglichkeit nach klassischem Vorgehen einen Fehlerbaum für das Gesamtsystem aufzubauen und die entsprechenden Ausfallarten, repräsentiert durch die Grundereignisse, den einzelnen *Hardware-Komponenten* zuzuordnen. Hierfür würde die Annotation von Ausgangsabweichungen, die Generierung der Teilfehlerbäume sowie das Zusammensetzen des Gesamtfehlerbaums entfallen.

In Bezug auf das *Hardware-Architekturdesign* und ISO 26262 ist die Klassifikation der Ausfallarten erforderlich, um nachfolgend die geforderten Hardware-Sicherheitsevaluationen durchzuführen. Auf Basis des zusammengesetzten Fehlerbaums, unter Berücksichtigung der logischen Gatter zwischen den Grundereignissen und dem Hauptereignis, kann der Beitrag eines jeden Grundereignisses, welches eine Ausfallart der *Hardware-Komponente* repräsentiert, in Bezug auf das Hauptereignis analysiert werden. Dies wird durch die Nutzung einer Minimalschnitt-Analyse im Kontext der verschiedenen zugehörigen Hauptereignisse durchgeführt.

Jede Ausfallart kann unterschiedliche Klassifikationen im Kontext der verschiedenen Hauptereignisse besitzen. ISO 26262 beschreibt die Klassifikation der Ausfallarten gemäß [ISO26262, 5-7.4.3.2] und [ISO26262, 10-Fig.B.4], siehe hierzu auch Kapitel 2.7.4.4. Die Klassifikationen der Ausfallarten und deren automatische Ableitung aus einer Minimalschnitt-Analyse sind in Abb. 5.14 gezeigt, basierend auf der Auswirkung auf das Hauptereignis.

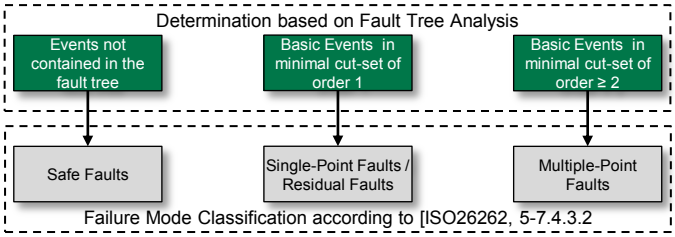


Abb. 5.14: Ableitung Klassifikation der Ausfallarten basierend auf einer qualitativen FTA [Adler2013b]

Ein Grundereignis, das nicht im Fehlerbaum des spezifischen Hauptereignisses enthalten ist, hat keine Auswirkung auf das Hauptereignis und wird als SF klassifiziert. Ein direkter Einfluss eines Grundereignisses auf ein spezifisches Hauptereignis kann identifiziert werden, wenn lediglich ODER-Gatter in dem korrespondierenden Zweig vorhanden sind. In diesem Kontext wird ein Minimalschnitt der Ordnung 1 ermittelt. Das Grundereignis führt direkt zum Hauptereignis und wird als SPF oder RF klassifiziert. Basierend auf der Existenz eines Sicherheitsmechanismus kann die Klassifikation weitergehend in SPF bzw. RF unterschieden werden.

Wenn das Grundereignis mit dem Hauptereignis über mindestens ein logisches UND-Gatter in dem Zweig angegliedert ist, so ist der Minimalschnitt größer als oder gleich der Ordnung 2. Das spezifische Grundereignis führt nur in Kombination mit anderen Grundereignissen zum Hauptereignis und wird als MPF klassifiziert. ISO 26262 erwähnt die Einschränkung auf DPF nach [ISO26262, 5-7.4.3.2.note1]. Die abgeleiteten Klassifikationen der Ausfallarten werden in das Modell annotiert, um die gesamten Fehlerinformationen für die Hardware-Sicherheitsevaluationen im Kontext von *Hardware-Architekturdesigns* bereitzustellen.

Die Sicherheitsbezogenheit wird auf Ebene der *Hardware-Elemente* und nicht auf Ebene der Ausfallarten bestimmt, vgl. [ISO26262, 5-AnnexE]. Im Kontext des Vorgehens für *Hardware-Architekturdesigns* und Verwendung einer qualitativen Analyse könnte hierzu der Ansatz nach Abb. 5.15 verwendet werden.

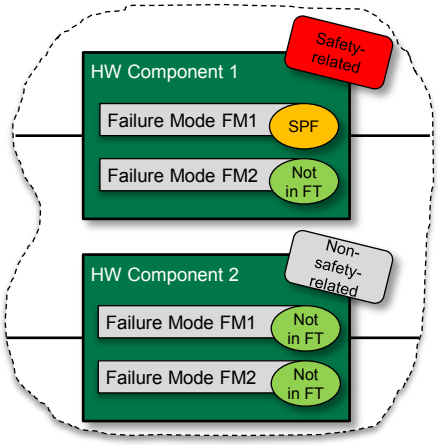


Abb. 5.15: Bestimmung der Sicherheitsbezogenheit von Hardware-Komponenten

Wenn die betrachtete *Hardware-Komponente* mindestens ein zugehöriges Grundereignis im entsprechenden Fehlerbaum enthält, welches zu einer Verletzung des Sicherheitszieles führen kann, wird diese als sicherheitsbezogen eingestuft. Beispielsweise ist Ausfallart FM1 der *Hardware-Komponente* 1 basierend auf der FTA als SPF klassifiziert, somit ist diese *Hardware-Komponente* sicherheitsbezogen. FM2 ist in diesem Fall ein sicherer Fehler (SF), welcher zur sicherheitsbezogenen Gesamtausfallrate beiträgt. Wenn die *Hardware-Komponente* kein Grundereignis besitzt, welches im Fehlerbaum enthalten ist, wird diese als nicht-sicherheitsbezogen eingestuft. Daher tragen die ebenfalls als SF klassifizierten Ausfallarten FM1 und FM2 der *Hardware-Komponente* 2 nicht zur sicherheitsbezogenen Gesamtausfallrate bei. Alternativ könnten für die Bestimmung der sicherheitsbezogenen Gesamtausfallrate lediglich die im Fehlerbaum enthaltenen Grundereignisse herangezogen werden, vgl. [97, S.52].

5.9.3 Quantitative Sicherheitsevaluation

5.9.3.1 Hardware-Architekturmetriken

ISO 26262 fordert nach [ISO26262, 5-8] die Evaluation der Hardware-Architekturmetriken in Bezug auf die Robustheit des *Hardwaredesigns*. Die Methodik für diese quantitative Evaluation bzgl. des *Hardware-Architekturdesigns* wurde bereits in Kapitel 5.8 dargestellt. Hierbei werden die relevanten Daten der *Hardware-Komponenten* in die Fehlerdatentabelle nach Kapitel 5.7 eingetragen. Die Gesamtausfallrate der *Hardware-Komponente* wird in diesem Kontext als Summe der angenommenen konstanten Ausfallraten λ_{FM} aller Ausfallarten bestimmt, falls der Ansatz nach Kapitel 5.7.3 verwendet wurde. Die Ausfallratenverteilung wird mit Hilfe der Ausfallrate der spezifischen Ausfallart dividiert durch die Gesamtausfallrate der *Hardware-Komponenten* bestimmt und in Prozent ausgedrückt.

Die restlichen Spalten der Fehlerdatentabelle sind unterteilt in Informationen hinsichtlich direkter Verletzung (SPF oder RF) und Verletzung in Kombination mit anderen Ausfällen (MPF). Die Klassifikation der Ausfallarten wird im Rahmen des *Hardware-Architekturdesigns* von einer Minimalschnitt-Analyse automatisiert abgeleitet. Zusätzlich wird für RFs und MPFs der vorliegende Sicherheitsmechanismus hinterlegt.

Die Ergebnisse der Hardware-Architekturmetriken auf Ebene der *Hardware-Komponenten* dienen als initiale Sicherheitsevaluation des vorläufigen *Hardwaredesigns* gegen die Zielwerte, die von der ASIL-Klassifizierung des Sicherheitsziels abgeleitet werden. Basierend hierauf können iterative Sicherheits-Designentscheidungen sowie Zielwert-Vorgaben unter Beachtung von weiteren Entwicklungsphasen erfasst und definiert werden.

5.9.3.2 Evaluation der Sicherheitsziel-Verletzungen nach PMHF

Die quantitativen Evaluationen des *Hardwaredesigns* nach [ISO26262, 5] erfordern, zusätzlich zur Anwendung der Hardware-Architekturmetriken, die Evaluation des Restrisikos von Sicherheitsziel-Verletzungen, wie in Clause 9 beschrieben. Im Kontext des *Hardware-Architekturdesigns* wurde basierend auf der FTA die Unterstützung einer PMHF vorgeschlagen, siehe auch Kapitel 5.6. PMHF beschreibt einen globalen probabilistischen Wert für die Verletzung eines Sicherheitsziels. In Anbetracht dieses Ansatzes kann die Wahrscheinlichkeit für den Eintritt des Hauptereignisses, als Verletzung des Sicherheitsziels, bestimmt werden, indem eine quantitative Analyse auf den zusammengesetzten Gesamtfehlerbaum angewendet wird.

Daher müssen die spezifischen Ausfallraten der Ausfallarten einschließlich ihrer Abdeckung durch die Sicherheitsmechanismen in probabilistische Werte $Q_{BasicEvent}$ transformiert und zu den entsprechenden Grundereignissen im Fehlerbaum annotiert werden. Die Lebensdauer eines Fahrzeugsystems $t_{SystemLifetime}$ ist für die Umrechnung zu berücksichtigen. Im Rahmen dieser Arbeit ist in Anlehnung an [ISO26262, 10-AnnexB.4] eine näherungsweise Umrechnung basierend auf einer linearisierten Exponentialverteilung

vorgesehen, siehe Gleichung 5.10. Diese Linearisierung kann für den sehr kleinen Wertebereich von $\lambda \cdot t_{SystemLifetime}$ vorgenommen werden.

$$Q \approx \lambda \cdot t_{SystemLifetime} \tag{5.10}$$

Unter der System-Lebensdauer $t_{SystemLifetime}$ versteht man die gesamte Betriebsdauer eines Fahrzeugsystems. Für die Lebensdauer wird im Bereich Automotive nach ISO 26262 exemplarisch mit 5000 h [ISO26262, 10-AnnexB.4] gerechnet. Dies entspricht umgerechnet ca. 208 Tagen und damit etwa 7 Monaten oder 0,57 Jahren. Bei einer durchschnittlichen Fahrgeschwindigkeit von 60 km/h erreicht man den auch häufig in Spezifikationen festgehaltenen Wert von 300.000 km.

Basierend auf den probabilistischen Werten der Grundereignisse kann mit Hilfe der Wahrscheinlichkeitsrechnung durch die Verwendung der logischen Verbindungen von Grundereignissen mit dem Hauptereignis ein quantitativer Wert $Q_{TopLevelEvent}$ bestimmt werden. Der berechnete globale probabilistische Wert kann im Rahmen einer PMHF genutzt werden. Diese Methodik unterstützt somit eine Anwendung der PMHF auf Hardware-Architektur-Ebene.

5.10 Untersuchung des detaillierten Hardwaredesigns

Im Rahmen der Entwicklung funktional sicherer Hardware kann im Anschluss an die iterative Verfeinerung und Evaluation des *Hardware-Architekturdesigns* das *detaillierte Hardwaredesign* initiiert werden. Hierfür ist eine dreiphasige Vorgehensweise nach Abb. 5.16 vorgeschlagen, welche sich von der Entwicklung des *Hardware-Architekturdesigns* abgrenzt, da eine detailliertere Abstraktionsebene betrachtet wird. Insbesondere die Klassifikation der Ausfallarten sowie die Durchführung spezifischer Hardware-Sicherheitsevaluationen sind hervorzuheben.

Um das *detaillierte Hardwaredesign* einschließlich aller Fehlerinformationen zu modellieren und eine Evaluation hinsichtlich funktionaler Sicherheit durchführen zu können, muss die dargestellte Vorgehensweise angewendet werden. In der ersten Phase besteht das Hauptarbeitsprodukt im Rahmen des *detaillierten Hardwaredesigns* in der Entwicklung eines Schaltplans, in dem alle grundlegenden Fehlerinformationen hinterlegt sind. Hierbei basieren die statistisch erhobenen Ausfallraten der *Hardware-Bauteile* beispielsweise auf Standardwerken, siehe Kapitel 2.7. Die zweite Phase beschreibt die Klassifikation einer jeden Ausfallart für jedes zugeordnete Sicherheitsziel. In der letzten Phase werden die für das *detaillierte Hardwaredesign* relevanten Hardware-Sicherheitsevaluationen inklusive der Verifikation gegen die Zielwerte durchgeführt. Für das *detaillierte Hardwaredesign* sind hier die Hardware-Architekturmetriken und die FRC-Methode vorgeschlagen, siehe Kapitel 5.6. Beide Evaluationen können automatisiert durchgeführt werden, indem die im Datenmodell hinterlegten Fehlerinformationen verwendet werden.

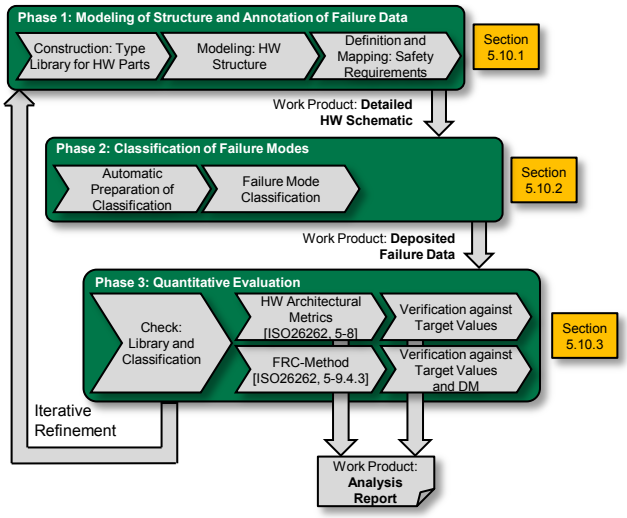


Abb. 5.16: Vorgehensweise für die Evaluation von detaillierten Hardwaredesigns

Auf Ebene des *detaillierten Hardwaredesigns* bietet es sich zudem an, die Evaluation der Hardware-Architekturmetriken und die FRC-Methode miteinander zu kombinieren. Insbesondere im Hinblick auf eine gemeinsame Ergebnisdarstellung im Rahmen der Fehlerdatentabelle kann somit die Robustheit des *Hardwaredesigns* und die Auswirkungen von eingeführten Sicherheitsmechanismen auf einen Blick betrachtet und überprüft werden.

5.10.1 Modellierung der Struktur und Annotation von Fehlerinformationen

Im Rahmen des *detaillierten Hardwaredesigns* wird eine Typenbibliothek für gleichartige *Hardware-Bauteile* angelegt, welche um Fehlerinformationen angereichert ist. Diese kann beispielhaft einen spezifischen surface-mounted device (SMD)-Widerstand kapseln, da alle Verwendungen dieses Widerstandstyps im Schaltplan die gleiche Fehlercharakteristik besitzen. Somit bietet die Hinterlegung der entsprechenden Fehlerinformationen in Bezug auf Ausfallraten und Ausfallarten in der Typenbibliothek einen großen Mehrwert für die Modellierung. Optionale Skalierungsfaktoren erlauben hier eine Anpassung von Ausfallraten aus verschiedenen anerkannten Industriesammlungen.

Nachdem die Typenbibliothek angelegt ist und alle für den Schaltplan genutzten Typen von *Hardware-Bauteilen* bereitgestellt sind, wird im nächsten Schritt das *detaillierte Hardwaredesign* in Form eines Schaltplans grafisch aufgebaut. Zusätzlich muss der Designer Sicherheitsziele mit ihrem ASIL sowie Sicherheitsmechanismen mit entsprechenden Diagnoseabdeckungsgraden anlegen, falls diese nicht bereits vorhanden

sind. Für die Analyse und Rückverfolgbarkeit werden diese mit den *Hardware-Bauteilen* des Schaltplans verknüpft. Am Ende der ersten Phase liegt als Arbeitsprodukt ein Schaltplan mit allen grundlegenden Fehlerinformationen vor.

5.10.2 Klassifikation der Ausfallarten

Durch die Verknüpfung von Sicherheitsanforderungen mit den modellierten *Hardware-Bauteilen* im vorhergehenden Schritt, ist die Zugehörigkeit der *Hardware-Bauteile* zu spezifischen Sicherheitszielen bereits vorliegend. Hierzu kann nun die Einstufung der Sicherheitsbezogenheit erfolgen. Anschließend kann eine Klassifikation der Ausfallarten im Sinne von SFs, SPFs oder RFs und latenten MPFs unter den jeweiligen Sicherheitszielen vorgenommen werden.

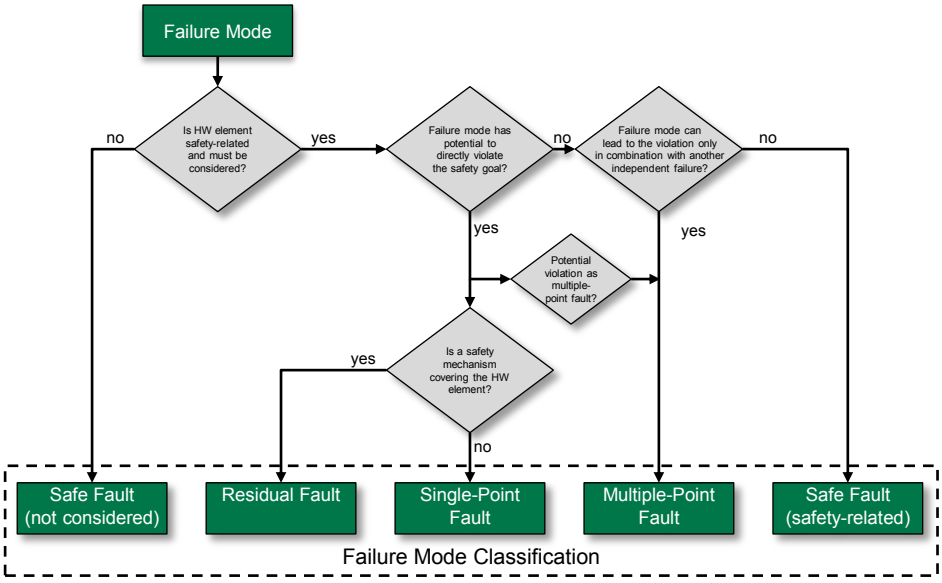


Abb. 5.17: Vereinfachtes Flussdiagramm für die Bestimmung der Klassifikation der Ausfallarten [D333b]

Hierbei ist zu erwähnen, dass die entsprechenden Klassifikationen immer im Kontext einer Ausfallart eines instanziierten *Hardware-Bauteils* vorzunehmen sind, sodass die entsprechenden Ausfallarten in der Bibliothek lediglich als Vorlage dienen, aber ein spezifisches Modellelement für die Ausfallarten eines jeden *Hardware-Bauteils* vorliegen muss. Weiterhin kann ein *Hardware-Bauteil* selbst im Kontext von mehreren Sicherheitszielen stehen, wobei die Ausfallarten unterschiedliche Klassifikationen aufweisen können. Im Rahmen des *Hardware-Architekturdesigns* wurde die Klassifikation über eine FTA auto-

matisiert vorgenommen. Dies ist auf Ebene des *detaillierten Hardwaredesigns* im Rahmen der vorgestellten Methodik nicht vorgesehen. Somit müssen die Klassifikationen vom Entwickler händisch für die Ausfallarten der *Hardware-Bauteile* eingetragen werden. Ein vereinfachtes Flussdiagramm in Anlehnung an [ISO26262, 10-Fig.9] als Unterstützung der Klassifikation ist in Abb. 5.17 dargestellt.

Arbeitsprodukt der zweiten Phase ist die Einstufung der Sicherheitsbezogenheit sowie die abgeschlossene Klassifikation der Ausfallarten für alle *Hardware-Bauteile* im Kontext der relevanten Sicherheitsziele. Zusammen mit dem strukturellen Modell und den grundlegenden Fehlerinformationen aus der ersten Phase kann nun die Durchführung der Hardware-Sicherheitsevaluationen initiiert werden.

5.10.3 Hardware-Sicherheitsevaluationen

Die Durchführung der Hardware-Sicherheitsevaluationen basiert analog zum *Hardware-Architekturdesign* auf einer Fehlerdatentabelle. Diese wird auf Grundlage der Fehlerinformationen des Modells sowie der strukturellen Modellierung und den eingebrachten Klassifikationen der Ausfallarten befüllt. Die spezifischen Ausfallraten der Ausfallarten sind automatisiert berechnet und stehen für die anschließenden Hardware-Sicherheitsevaluationen zur Verfügung. Nach Kapitel 5.6 werden für das *detaillierte Hardwaredesign* die Hardware-Architekturmetriken sowie die FRC-Methode für die Evaluation der Verletzung der Sicherheitsziele durchgeführt. Vor dieser werden die einzelnen Fehlerinformationen der *Hardware-Bauteil*-Typen unter den entsprechenden Sicherheitszielen auf Konsistenz und zusätzlich die Klassifikation der Ausfallarten der einzelnen instanziierten *Hardware-Bauteile* auf eine vorhandene Einstufung überprüft, siehe Kapitel 5.7.

5.10.4 Evaluation der Hardware-Architekturmetriken

Werden die Hardware-Architekturmetriken auf Ebene der *detaillierten Hardwaredesigns* angewandt, so sind in der Bibliothek die *Hardware-Bauteile* sowie deren entsprechende Ausfallraten und Ausfallarten mit zugehöriger Ausfallratenverteilung zu hinterlegen. Dies entspricht der klassischen Vorgehensweise bei einer FMEDA basierend auf dem modellierten Schaltplan, da sowohl die Auswirkung der Ausfallarten auf das Verhalten des Systems als auch die Abdeckung von Sicherheitsmechanismen und quantitative Ausfallraten miteinbezogen werden. Die Evaluation der Hardware-Architekturmetriken erfolgt anschließend nach Kapitel 5.8. Für die Anwendung der Hardware-Architekturmetriken auf *detaillierte Hardwaredesigns* können weiterhin vom vorherigen *Hardware-Architekturdesign* Zielwerte oder Rahmenbedingungen vorgegeben sein, insbesondere im Kontext einer verteilten Entwicklung. Die Resultate der Hardware-Architekturmetriken können anschließend gegenüber den Zielwerten geprüft werden, im

Rahmen des *detaillierten Hardwaredesigns* entspricht dies auch einer Verifikation des finalen *Hardwaredesigns*. Der Status der Erfüllung und alle berechneten Werte können in spezifischen Berichten dokumentiert werden. Diese sind im Rahmen des zu erbringenden Sicherheitsnachweises relevant.

5.10.5 Evaluation der Sicherheitsziel-Verletzung nach FRC-Methode

Ergänzend zu der *„Evaluation of the hardware architectural metrics“* wird im Folgenden der Umgang mit der zweiten beschriebenen Methode von *„Evaluation of safety goal violations due to random hardware failures“* vorgestellt, welche den Namen *„Evaluation of each cause of safety goal violation“* trägt und teilweise namentlich vereinfachend auch als FRC-Methode bezeichnet wird. Die Anwendung der FRC-Methode ist der Evaluation einer PMHF im Kontext des *detaillierten Hardwaredesigns* vorzuziehen, siehe Kapitel 5.6.

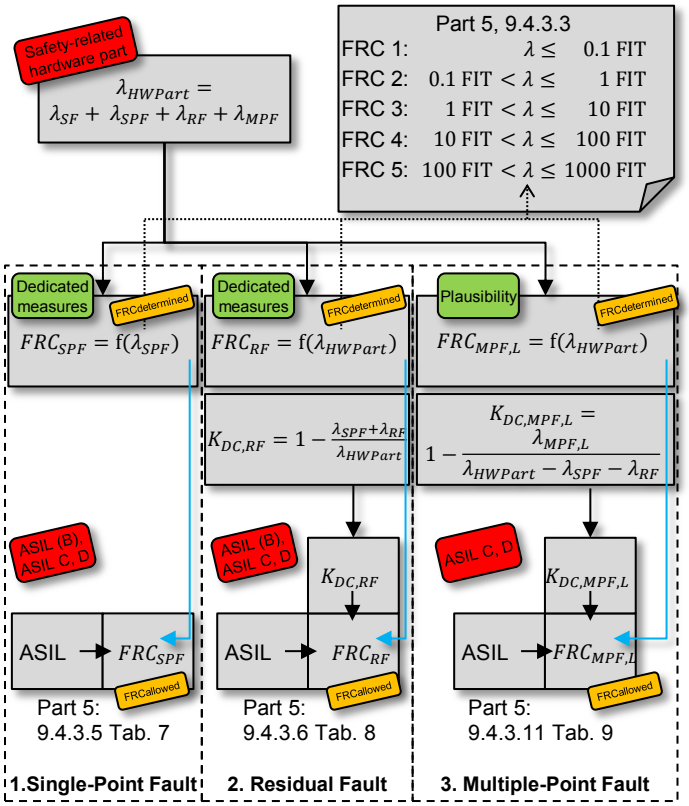


Abb. 5.18: Überblick über FRC-Evaluationsprozeduren für SPFs, RFs und latente MPFs

Um das Restrisiko einer Sicherheitsziel-Verletzung zu evaluieren und die Hardware-Architekturmetriken zu ergänzen, führt die zweite Methode nach [ISO26262, 5-9.4.3] eine individuelle Untersuchung einer jeden Sicherheitsziel-Verletzung mit definierten Ausfallratenklassen (FRCs) durch. Alle Ausfallarten, welche als Einzelfehler (SPF), Restfehler (RF) oder Mehrfachfehler (MPF) klassifiziert wurden, können zur Verletzung des Sicherheitsziels beitragen und müssen daher untersucht werden. Hierfür können die Klassifikationen sowie entsprechende spezifische Ausfallraten direkt aus der Fehlerdatentabelle entnommen werden. Part 5 suggeriert eine Einschränkung der Mehrfachfehler auf Zweifachfehler (DPF), welche die Distanz n=2 repräsentieren [ISO26262, 5-7.4.3.2.note1]. Somit können MPFs mit einer größeren Distanz als SFs betrachtet werden. ISO 26262 empfiehlt die Anwendung dieser Methode auf Ebene des *detaillierten Hardwaredesigns*, vgl. [ISO26262, 5-9.4.3.2]. Die Untersuchung erfolgt somit bei der FRC-Methode nicht auf Ebene des Items sondern auf Schaltplanebene für jedes einzelne *Hardware-Bauteil*.

Die FRC-Methode gliedert sich grundlegend dreiteilig. Hierbei handelt es sich um gesonderte Evaluierungsprozeduren für SPFs, RFs und latente MPFs, in denen eine finale Überprüfung durch festzulegende Zielwerttabellen nach [ISO26262, 5-Tab.7] für SPFs, [ISO26262, 5-Tab.8] für RFs und [ISO26262, 5-Tab.9] für latente MPFs durchgeführt wird. Abb. 5.18 gibt einen Überblick über die Evaluationsprozeduren, welche in den folgenden Abschnitten im Detail erklärt werden. Die Evaluation im Rahmen der FRC-Methode ist für ASIL-D und ASIL-C erforderlich und kann hinsichtlich SPFs und RFs für ASIL-B zusätzlich angewendet werden.

5.10.5.1 Einordnung in Ausfallratenklassen

Als Grundvoraussetzungen für die FRC-Methode werden sogenannte Ausfallratenklassen (FRCs) definiert, die eine Einordnung einer zu evaluierenden Ausfallrate erlauben. Von der ISO 26262 ist eine Einordnung nach [ISO26262, 5-9.4.3.3] vorgeschlagen. Die Einordnung in die entsprechende FRC unterscheidet sich für die Betrachtung nach SPFs, RFs sowie MPFs dadurch, dass wie in Abb. 5.18 ersichtlich, die Einstufung von unterschiedlichen Ausfallraten abhängt. Hierbei ist wichtig, dass für die Einordnung von SPFs nach [ISO26262, 5-9.4.3.5.note] die SFs mit einbezogen werden können, währenddessen für die Bewertung der RFs bzw. MPFs nach [ISO26262, 5-9.4.3.6.note1] und [ISO26262, 5-9.4.3.11.note1] keine Auswirkungen von Sicherheitsmaßnahmen mit einbezogen werden. Für alle Einzelfehler eines *Hardware-Bauteils* muss daher die Summe der spezifischen Ausfallraten aller dazugehörigen Ausfallarten gebildet werden. Dieser Summenwert wird in eine Ausfallratenklasse eingeordnet. Für die Restfehler und latenten Mehrfachfehler eines *Hardware-Bauteils* muss die Gesamtausfallrate des *Hardware-Bauteils* gemäß den Ausfallratenklassen eingeordnet werden.

Für die Einordnung in eine FRC kann nach [ISO26262, 5-9.4.3.3] eine Tabelle erstellt werden, welche die entsprechenden Wertebereiche, in denen sich die Ausfallrate bewegen darf, durch eine obere und untere Grenze für die Einordnung beschränkt. Hierbei

kann der Zielwert einer PMHF für ASIL-D aus [ISO26262, 5-Tab.6] für die Einordnung der niedrigsten Ausfallratenklasse FRC 1 einfließen [ISO26262, 5-9.4.3.3.note2], um die beiden alternativ anwendbaren Methoden nach Clause 9 zu harmonisieren. Dieser wird hierfür durch einen festzulegenden Teiler, standardmäßig von ISO 26262 auf 100 festgelegt, geteilt. Ausgehend von diesem berechneten Wert für die FRC 1 können die weiteren Ausfallratenklassen durch den Teiler von 10 aufgespannt werden. Verallgemeinert ist dies über einen Teiler von $10^{(i-1)}$ für die weiteren Ausfallratenklassen FRC i möglich [ISO26262, 5-9.4.3.3d]. Eine niedrigere Ausfallratenklasse bedeutet somit immer eine kleinere Ausfallrate des *Hardware-Bauteils* und damit höhere Zuverlässigkeit. Die ISO 26262 beschreibt die Nutzung von fünf Ausfallratenklassen, diese können aber beliebig erweitert werden, basierend auf den im Rahmen des *Hardwaredesigns* vorhandenen Ausfallraten, siehe auch Sonderfall in [ISO26262, 5-9.4.3.3.note4].

Der Teiler, welcher für die Einstufung der FRC 1 genutzt wird, kann nach [ISO26262, 5-9.4.3.4] durch eine niedrigeren Wert ersetzt werden, basierend auf einer entsprechenden Begründung. Somit müssen sowohl dieser Teiler als auch die Anzahl der Ausfallratenklassen im Rahmen der Evaluation vom Nutzer parametrierbar sein. Die Auswirkungen eines niedrigeren Teilers sind, dass sich die obere Grenze aller Ausfallratenklassen einer höheren Ausfallrate zuordnen lässt, wodurch bei gleichbleibender Ausfallrate eine Einordnung in eine niedrigere Ausfallratenklasse zutreffen kann. Durch Verwendung des standardmäßigen Teilers werden folgende Bereiche der Ausfallratenklassen abgesteckt, siehe Tab. 5.3:

Failure rate class ranking					
FRC 1:			λ	≤	0,1 FIT
FRC 2:	0,1 FIT	<	λ	≤	1 FIT
FRC 3:	1 FIT	<	λ	≤	10 FIT
FRC 4:	10 FIT	<	λ	≤	100 FIT
FRC 5:	100 FIT	<	λ	≤	1000 FIT
FRC …:	…	<	λ	≤	…
FRC max.:	…	<	λ	≤	…

Based on max. defined FRC

Based on PMHF target value for ASIL D (default: 10^{-8} h^{-1}) and divider (default:100)

Tab. 5.3: Abgeleitete FRC-Wertebereiche basierend auf den vorgeschlagenen Eingangswerten nach [ISO26262, 5-9.4.3.3]

5.10.5.2 Diagnoseabdeckungsgrad im Kontext der FRC

Der im Folgenden beschriebene Diagnoseabdeckungsgrad (Diagnostic Coverage) unterscheidet sich von dem Diagnoseabdeckungsgrad im Zuge der Sicherheitsmechanismen. Daher wird zusätzlich eine Kennung mit (HW) in den zugehörigen Gleichungen nachgestellt.

Die Gleichungen zur Berechnung der $K_{DC,RF}\,(HW)$ und $K_{DC,MPF,L}\,(HW)$ sind nicht explizit in der ISO 26262 erwähnt, können allerdings aus [ISO26262, 5-9.4.3.6.note4] für $K_{DC,RF}\,(HW)$ und [ISO26262, 5-9.4.3.11.note4] für $K_{DC,MPF,L}\,(HW)$ abgeleitet werden.

Diese verweisen auf die Gleichungen [ISO26262, 5-C5] für die SPFM und [ISO26262, 5-C6] für die LFM. Die Unterscheidung liegt darin, dass bei den Hardware-Architekturmetriken die Summenbildung im Rahmen des Fahrzeugsystems (Items) erfolgt, für die Diagnoseabdeckungsgrade im Rahmen der FRC-Methode ist dies nur bezogen auf ein einzelnes *Hardware-Bauteil*. Somit ergibt sich für den Diagnoseabdeckungsgrad in Bezug auf RFs die Gleichung 5.11.

$$K_{DC,RF}\left(HW\right) = \left(1 - \frac{\lambda_{SPF} + \lambda_{RF}}{\lambda_{HWpart}}\right) \cdot 100 \text{ in [\%]} \tag{5.11}$$

Die Summe im Nenner repräsentiert die Gesamtausfallrate eines einzelnen *Hardware-Bauteils*. Im Vergleich zur SPFM fehlen die Summen über die sicherheitsbezogenen *Hardware-Bauteile*, da hier nur das einzelne *Hardware-Bauteil* betrachtet wird. Der Multiplikator 100 dient zur Angabe in Prozent. Dadurch dass λ_{SPF} für die Berechnung mit einbezogen wird, wird der Wert der $K_{DC,RF}\left(HW\right)$ verringert. Die Auswirkungen sind, dass hierdurch möglicherweise eine niedrigere zu erreichende FRC zutrifft und somit höhere Ansprüche an das zugehörige *Hardware-Bauteil* im Sinne einer niedrigeren Ausfallrate gestellt werden.

$$K_{DC,MPF,L}\left(HW\right) = \left(1 - \frac{\lambda_{MPF,L}}{\lambda_{HWpart} - \lambda_{SPF} - \lambda_{RF}}\right) \cdot 100 \text{ in [\%]} \tag{5.12}$$

Die Berechnung des Diagnoseabdeckungsgrads in Bezug auf latente MPFs ist in Gleichung 5.12 dargestellt. Vergleicht man Gleichung 5.12 mit der Berechnungsvorschrift der LFM, so fehlen ebenfalls die Summen über die sicherheitsbezogenen *Hardware-Bauteile*, da hier nur das einzelne *Hardware-Bauteil* betrachtet wird. Auch hierbei erfolgt die Multiplikation mit 100 für eine Angabe in Prozent. Im Rahmen der $K_{DC,MPF,L}\left(HW\right)$ werden die SPFs und RFs nicht berücksichtigt und von der Ausfallrate des *Hardware-Bauteils* abgezogen.

Da die FRC-Methode auf Ebene der *Hardware-Bauteile* durchgeführt wird, müssen alle SPFs, RFs und latenten MPFs jeweils für die einzelnen *Hardware-Bauteile* im Zuge der Berechnung der Diagnoseabdeckungsgrade aufsummiert werden. Somit kann die Ausfallrate λ_{SPF} im Rahmen der Fehlerdatentabelle für eine Ausfallart stehen, während diese im Zuge der FRC die Summe aller entsprechend klassifizierten Ausfallarten innerhalb eines *Hardware-Bauteils* darstellt. Dies war bereits für die Einordnung der FRC im Zuge der Evaluationsprozedur für SPF notwendig. Daher ist eine getrennte Aufschlüsselung in der Fehlerdatentabelle unter dem Gesichtspunkt der Wiederverwendung sinnvoll.

5.10.5.3 Überprüfung gegenüber Zielvorgaben

Für die Einordnung in FRCs wurde eine gemeinsame Tabelle für SPFs, RFs und latente MPFs genutzt. Die Bezüge und Abhängigkeiten zu den unterschiedlichen ASIL-Klassifizierungen werden durch die entsprechenden Zeilen nach [ISO26262, 5-Tab.7/8/9] spezifisch hergestellt. Vorausgreifend sei erwähnt, dass im Zuge der Überprüfung auf Erfüllung hinsichtlich SPFs, RFs und latenter MPFs spezifische Sonderfälle betrachtet werden müssen. Dies betrifft *dedizierte Maßnahmen*, die generische Erweiterung der Zielwerttabellen sowie die Prüfung auf Plausibilität.

Je weiter oben in der Zielwerttabelle, und bei den Zielwerttabellen für RFs und MPFs durch den zusätzlichen Parameter des Diagnoseabdeckungsgrades je weiter rechts, umso niedriger ist die zu erreichende Ausfallratenklasse, da der ASIL höhere Anforderungen stellt, bei gleichzeitig niedrig vorliegender $K_{DC,RF}\,(HW)$ oder $K_{DC,MPF,L}\,(HW)$. Hierbei sind gegebenenfalls zusätzlich *dedizierte Maßnahmen* erforderlich, die zusätzlich zur zulässigen Ausfallratenklasse in den Zielwertvorgaben vermerkt sind.

Erfüllung im Kontext SPF:

Ob die zulässige FRC durch die basierend auf der Ausfallrate eingeordnete FRC erfüllt wird, kann im Kontext der SPFs durch eine einfache Gegenüberstellung nach Tab. 5.4 überprüft werden. Die in den Zielwerttabellen niedergeschriebenen FRCs entsprechen den zulässigen FRCs, welche mindestens einzuhalten sind. Mindestens in diesem Kontext heißt, dass niedrigere Ausfallratenklassen zulässig aber nicht erforderlich sind, siehe [ISO26262, 5-9.4.3.6.note2] bzw. [ISO26262, 5-9.4.3.11.note2].

ASIL of the safety goal	Failure rate class
D	Failure rate class 1 + dedicated measures[a]
C	Failure rate class 2 + dedicated measures[a] or Failure rate class 1
(B)	Failure rate class 2 or Failure rate class 1

[a] The note in requirement 9.4.2.4 gives examples of dedicated measures.

Tab. 5.4: FRC-Zielwerte im Kontext von SPFs [ISO26262, 5-Tab.7]

Für ASIL-D und ASIL-C sind im Rahmen der Evaluation der SPFs zusätzlich *dedizierte Maßnahmen* zu ergreifen. Diese zusätzliche Maßnahme ist spezifiziert, da SPFs ohne Absicherung durch einen Sicherheitsmechanismus direkt zu einer Verletzung des Sicherheitsziels führen und somit die gravierendsten Fehler darstellen.

Erfüllung im Kontext RF:

Für die Bestimmung der zu erreichenden FRC wird im Rahmen der Evaluation der RFs sowohl der Diagnoseabdeckungsgrad $K_{DC,RF}\,(HW)$ als auch der entsprechende ASIL benötigt. Hierbei werden für die Überprüfung gegenüber den Zielwerten die Spalten für

unterschiedliche Diagnoseabdeckungsgrade und die Zeilen für die ASIL-Klassifizierung verwendet, die einzelnen Zellen spiegeln die zulässige Ausfallratenklasse für die spezifischen Parameter wider. Die in diesem Kontext aus der Gesamtausfallrate des *Hardware-Bauteils* bestimmte FRC kann mit der zulässigen FRC aus der Zielwerttabelle verglichen werden. Eine mögliche Zielwerttabelle für RFs zeigt Tab. 5.5.

ASIL of the safety goal	Diagnostic coverage with respect to residual faults				
	≥ 99,99 %	≥ 99,9 %	≥ 99 %	≥ 90 %	< 90 %
D	FRC 5	FRC 4	FRC 3	FRC 2	FRC 1 + dedicated measures[a]
C	FRC 6	FRC 5	FRC 4	FRC 3	FRC 2 + dedicated measures[a]
(B)	(FRC 6)	FRC 5	FRC 4	FRC 3	FRC 2

[a] The note in requirement 9.4.2.4 gives examples of dedicated measures.

Tab. 5.5: FRC-Zielwerte im Kontext von RFs (maximale FRC von i=6) [ISO26262, 5-Tab.8]

Die hier dargestellte Tabelle entspricht der Anwendung der maximalen FRC 6 für ASIL-C und FRC 5 für ASIL-D. In der ISO 26262 ist die Zielwerttabelle für maximal FRC 5 abgebildet. Die Zielwerttabelle für RFs ist somit dynamisch nach links erweiterbar, falls höhere Diagnoseabdeckungsgrade vorhanden sind oder höhere Ausfallratenklassen benötigt werden.

Soll nun eine weitere Spalte hinzugefügt werden, so sind die Gleichungen nach [ISO26262, 5-9.4.3.7] mit $[100 - 10^{(3-i)}]\%$ für ASIL-D und $[100 - 10^{(4-i)}]\%$ für ASIL-C anzuwenden. Somit ist die zulässige Ausfallratenklasse bei identischen Diagnoseabdeckungsgraden für ASIL-C entsprechend um 1 höher. Für ASIL-B ist nach [ISO26262, 5-9.4.3.7] keine Berechnungsvorschrift vorgesehen, daher wird im Zuge dessen vorgeschlagen, den Wert von ASIL-C zu übernehmen.

Klar ersichtlich wird der aus den Formeln für ASIL-D und ASIL-C vorhandene Unterschied, der in der Potenz vorliegt und dafür sorgt, dass sich das farblich hervorgehobene diagonale Muster in der Zielwerttabelle ergibt. Auch im Rahmen der RFs können *dedizierte Maßnahmen* für bestimmte Zielwerte erforderlich sein.

Erfüllung im Kontext MPF:

Bei latenten Mehrfachfehlern kann die zulässige Ausfallratenklasse basierend auf dem Diagnoseabdeckungsgrad $K_{DC,MPF,L}\,(HW)$ und dem zugehörigen ASIL aus der Zielwerttabelle abgeleitet werden. Ähnlich wie bei der Erweiterung der Tabelle für RFs nach [ISO26262, 5-Tab.8] ist auch eine Erweiterung der Zielwerttabelle für MPFs denkbar, was jedoch nicht explizit in ISO 26262 erwähnt wird. Vergleicht man die Zielwerttabellen für RFs und MPFs so ist ersichtlich, dass wenn ein Grenzwert für $K_{DC,RF}\,(HW)$ dem Grenzwert von $K_{DC,MPF,L}\,(HW)$ gleicht, die dazugehörige FRC bei gleichem ASIL einen Unterschied der Ausfallratenklasse von 1 aufweist. Dies ist darauf zurückzuführen, dass RF durch den direkten Einfluss auf die Verletzung des Sicherheitsziels höheren

Anforderungen genügen müssen. Eine mögliche Zielwerttabelle für MPFs ist in Tab. 5.6 abgebildet. Da die Evaluation für latente MPFs lediglich für ASIL-D und ASIL-C angewendet wird, enthält die zugehörige Zielwerttabelle zwei Zeilen.

ASIL of the safety goal	Diagnostic coverage with respect to latent faults			
	≥ 99,9 %	≥ 99 %	≥ 90 %	< 90 %
D	FRC 5	FRC 4	FRC 3	FRC 2
C	FRC 6	FRC 5	FRC 4	FRC 3

Tab. 5.6: FRC-Zielwerte im Kontext von MPFs [ISO26262, 5-Tab.9]

Umgang mit *dedizierten Maßnahmen*:

Die zuvor vorgestellten Zielwerttabellen für SPFs nach Tab. 5.4 und RFs nach Tab. 5.5 enthalten für bestimmte FRC die Zusätze für *dedizierte Maßnahmen*. Bei den *dedizierten Maßnahmen*, im Englischen Dedicated Measures, handelt es sich nach [ISO26262, 1-1.20] um eine Maßnahme, um die Ausfallrate abzusichern. Als *dedizierte Maßnahmen* können nach [ISO26262, 5-9.4.2.4] beispielsweise Burn-In-Tests, spezielle Test zur Untersuchung des Auftretens von bestimmten Ausfallarten oder physikalische Trennung auf einem PCB herangezogen werden. In Bezug auf die Entwicklung von Mikrocontrollern ist dies unter [ISO26262, 10-Tab.A.8] beispielhaft in der Designphase „Safety-related special characteristics during chip production“ gelistet.

Dedizierte Maßnahmen können in diesem Ansatz aus zweierlei Hinsicht berücksichtigt werden: Einerseits aus der Sicht einer zusätzlichen Überprüfung, was dazu veranlasst die *dedizierten Maßnahmen* als Design-Elemente zu betrachten und zu modellieren oder die notwendigen Informationen zu annotieren. Anderseits können die *dedizierten Maßnahmen* aus Sicht einer Auflage gesehen werden, welche im Falle einer entsprechenden Einordnung explizit als Zusatz gefordert sind.

Bezogen auf die Tabelle 7 [ISO26262, 5-Tab.7] gibt es für ASIL-C zwei Möglichkeiten dem Zielwert gerecht zu werden: „Failure Rate Class 2 + dedicated measures“ oder „Failure Rate Class 1“. Da sich die Ausfallraten der einzelnen SPFs eines *Hardware-Bauteils* deutlich unterscheiden können, besteht die Möglichkeit, anstelle der Einführung von *dedizierten Maßnahmen*, die FRC-Einordnung zu erniedrigen indem einzelne Ausfallraten-Anteile der SPFs verringert werden. Dies ist insbesondere interessant, wenn es bei den SPFs einen deutlichen Ausreißer gibt.

Umgang mit Plausibilität für MPF:

Die Plausibilität ist nur im Fokus der Evaluationsprozedur für MPFs [ISO26262, 5-9.4.3.1]. Hierbei wird betrachtet, ob ein bestimmter MPF überhaupt plausibel ist, da nur plausible MPFs im Rahmen der FRC-Methode evaluiert werden müssen. Problematisch ist die Feststellung der Plausibilität insbesondere beim *detaillierten Hardwaredesign*, da der Bezug zwischen den Ausfallarten von unterschiedlichen *Hardware-Bauteilen*, welche zum MPF führen, teilweise nicht direkt offensichtlich ist.

Um dennoch eine Unterstützung zu bieten, kann über eine abgeleitete Vorgehensweise, wie nachfolgend beschrieben, der Untersuchungsraum eingeschränkt werden. Eine Plausibilität liegt demnach vor, wenn nach [ISO26262, 5-9.4.3.8a] für ASIL-D die $K_{DC,MPF,L}\,(HW)$ weniger als 90 % beträgt und nach [ISO26262, 5-9.4.3.9a] für ASIL-C die $K_{DC,MPF,L}\,(HW)$ weniger als 80 % beträgt. Die $K_{DC,MPF,L}\,(HW)$ kann für alle *Hardware-Bauteile* bestimmt werden. Ergibt sich hieraus, dass der Wert für ein *Hardware-Bauteil* über der genannten Grenze liegt, kann allerdings noch keine finale Aussage darüber getroffen werden, ob dieser MPF auch tatsächlich nicht-plausibel ist, aufgrund dass dieser in Kombination mit einem anderen betroffenen *Hardware-Bauteil* eingestuft wird. Liegt der $K_{DC,MPF,L}\,(HW)$ jedoch für sämtliche *Hardware-Bauteile* des *detaillierten Hardwaredesigns* über dem Grenzwert, so kann abgeleitet werden, dass in allen möglichen Kombinationen ein MPF nicht-plausibel wäre. Hierbei ist auch eine Überprüfung nach der Zielwerttabelle gemäß [ISO26262, 5-Tab.9] nicht mehr erforderlich. Ein weiterer Ansatz wäre, die Evaluation in jedem Falle durchzuführen. Werden für die Untersuchung der MPFs des *detaillierten Hardwaredesigns* die Zielwerte nach [ISO26262, 5-Tab.9] erreicht, so ist die Prüfung auf Plausibilität im Rahmen der Sicherheitsevaluation nicht mehr relevant.

5.10.6 Kombination der Ergebnisdarstellung der Hardware-Architekturmetriken und FRC-Methode

Die in Kapitel 5.7 beispielhaft dargestellte Fehlerdatentabelle wird im Kontext des *detaillierten Hardwaredesigns* als Grundlage zur Durchführung der „*Evaluation of the hardware architectural metrics*“ und „*Evaluation of each cause of safety goal violation*“ verwendet. Für diese Fehlerdatentabelle wird eine Erweiterung der Tabellenspalten vorgeschlagen, um somit die Ergebnisse der Hardware-Architekturmetriken sowie der FRC-Methode auf Schaltplanebene zu kombinieren und aufzubereiten, siehe Tab. 5.7.

Da die FRC-Methode auf Ebene der *Hardware-Bauteile* durchgeführt wird, sind die entsprechenden Zeilen übergeordnet für alle Ausfallarten eines *Hardware-Bauteils* zusammengefasst. Weiterhin ist durch die individuelle Evaluationsprozedur von SPFs, RFs und MPFs eine Aufteilung und Anbindung der Informationen an die thematisch zugehörigen Spalten der Fehlerdatentabelle vorgesehen. Da die SPFs und RFs im Rahmen der Fehlerdatentabelle gemeinsam betrachtet werden, sind die entsprechenden Informationen der FRC-Methode im mittleren Tabellenbereich abgebildet. Im Kontext der Evaluationsprozedur für latente MPF werden die Informationen im rechten Tabellenbereich angebunden. Hierbei sind sämtliche zur Evaluation benötigte Informationen wie die Summen der spezifischen Ausfallraten im Kontext des *Hardware-Bauteils*, die eingeordnete Ausfallratenklasse, die Diagnoseabdeckungsgrade sowie die Erfüllung der Zielwerte enthalten.

Die Evaluation der Hardware-Architekturmetriken erfolgt für das jeweilige Sicherheitsziel, somit sind die zugehörigen Spalten übergreifend ans Ende der Fehlerdatentabelle angehängt. Hier sind sämtliche relevanten Gesamtausfallraten, unter anderem die nichtsicherheitsbezogene und sicherheitsbezogene Gesamtausfallrate, dargestellt. Weiterhin sind die Summen der spezifischen Ausfallraten für SPFs bzw. RFs sowie latente MPFs über alle *Hardware-Bauteile* enthalten. Die Ergebnisse der SPFM und der LFM inklusive der Überprüfung gegenüber den Zielwerten bilden die weiteren Spalten. Die Erfüllung der Zielwerte ist sowohl für die FRC-Methode als auch für die Evaluation der Hardware-Architekturmetriken grafisch hervorgehoben, um eine Aufbereitung der Evaluationsergebnisse zu ermöglichen.

Tab. 5.7: Erweiterte Fehlerdatentabelle zur Integration der Ergebnisse der Hardware-Architekturmetriken sowie der FRC-Methode

5.11 Beispiel für eine Ventilregelung nach ISO 26262 Part 5

Innerhalb von Part 5 wird in Annex E [ISO26262, 5-AnnexE] auf ein abstrakt dargestelltes, möglicherweise fiktives Beispiel eines *Hardwaredesigns* für eine Ventilregelung eingegangen, welches im Folgenden für die Verifikation der vorgestellten Konzepte, Methodiken und Implementierungen genutzt wird. Ein Bezug zu einem realen System wird in ISO 26262 nicht hergestellt.

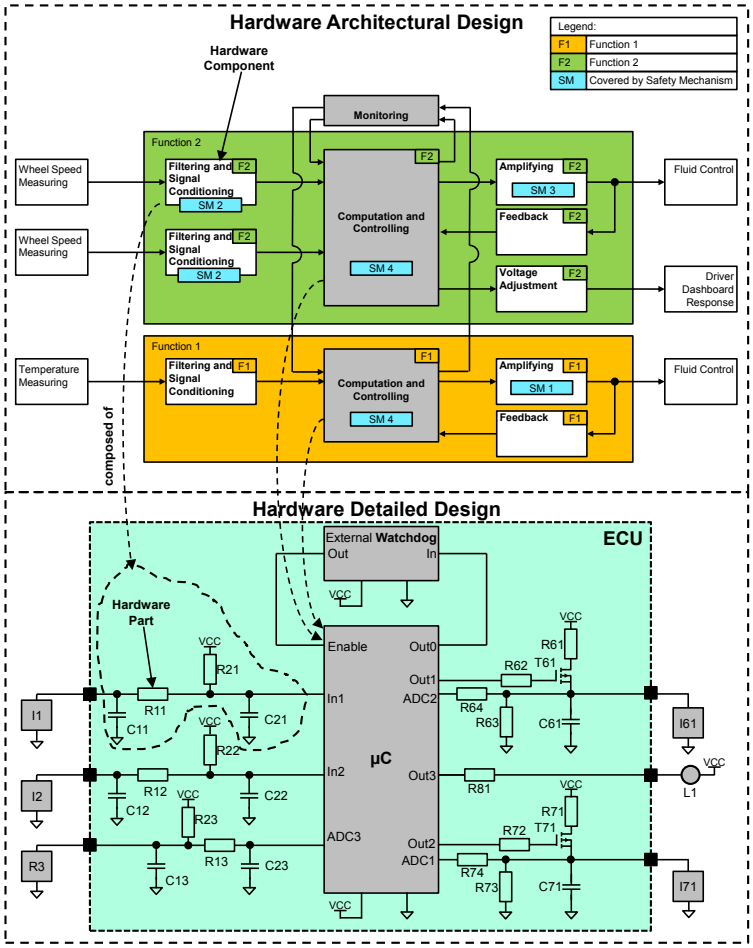


Abb. 5.19: Initiale Unterscheidung zwischen einem nachgebildeten Hardware-Architekturdesign und dem detaillierten Hardwaredesign basierend auf dem Schaltplan-Beispiel für eine Ventilregelung nach Part 5 Annex E [Adler2013b]

Da dieses Beispiel im Rahmen der ISO 26262 nur auf Ebene des *detaillierten Hardwaredesigns* vorliegt, wurde das *Hardware-Architekturdesign* nachgebildet, um eine entsprechende Unterscheidung zu bieten. Abb. 5.19 zeigt beide *Hardwaredesign*-Ebenen.

In [ISO26262, 5-AnnexD.1] wird in Bezug zur Evaluation von Diagnoseabdeckungsgraden ein beispielhaftes *Hardwaredesign* dargestellt, welches auf einer höheren Abstraktionsebene anzusiedeln ist. Dieses greift sowohl allgemeine und elektrische Elemente als auch allgemeine und spezifische Halbleiterelemente auf. Letztere werden nochmals in Prozessoreinheiten und Kommunikation verfeinert. Der strukturelle Aufbau ähnelt dem *detaillierten Hardwaredesign* aus Annex E, lässt sich allerdings nicht ganz deckungsgleich miteinander verknüpfen.

5.11.1 Hardware-Architekturdesign

Grundlegend zeigt das *Hardware-Architekturdesign* die Bereiche der Sensorik, Verarbeitung und Aktuatorik und bildet damit eine Wirkkette ab. Hier finden sich die entsprechenden *Hardware-Komponenten* wieder, welche im Rahmen des *detaillierten Hardwaredesigns* durch die *Hardware-Bauteile* implementiert werden. Auf dieser Abstraktionsebene ist bereits ersichtlich, dass zwei Funktionen im Fokus des Fahrzeugsystems stehen.

Funktion 1 liest einen Temperaturwert ein und verarbeitet diesen, um bei einer Temperatur größer als 90° C ein Ventil auf der Aktuatorikseite zu öffnen und damit einen Flüssigkeitsdurchfluss zu regeln. Über eine Rückkopplung wird zudem die Ausgangsstufe für die Ventilregelung überwacht. Funktion 2 hat als Eingangsdaten die Werte zweier Messungen für die Raddrehzahl. Durch eine Mittelwertbildung wird die Raddrehzahl berechnet. Ergibt sich für die daraus bestimmte Fahrzeuggeschwindigkeit ein Wert größer als 90 km/h, so wird veranlasst, dass ein anderes Ventil geöffnet wird. Wie bei der ersten Funktion wird auch hier über eine Rückkopplung die Ausgangsstufe überwacht. Zudem verfügt Funktion 2 über die Möglichkeit, bei Bedarf eine optische Rückmeldung an den Fahrer zu geben. Übergreifend wird die Verarbeitung beider Funktionen durch eine zusätzliche Monitoring-Komponente überwacht.

5.11.2 Detailliertes Hardwaredesign

Das *detaillierte Hardwaredesign* bildet die Realisierung der Funktionen auf Schaltplanebene ab. Hinsichtlich der Sensorik handelt es sich bei I1 und I2 um open-collector Raddrehzahl-Sensoren, R3 ist ein Temperatursensor. Auf Seiten der Aktuatorik repräsentieren I61 sowie I71 die elektrische Ansteuerung für die Ventile. Die LED L1, welche sich im Armaturenbrett befindet, dient einer optischen Signalisierung für den Fahrer. Ein Aufbau der Spannungsversorgung sowie unterschiedliche Massen gehen aus dem Schaltplan nicht hervor.

Die zwei vorgestellten Funktionen werden auf einer gemeinsamen ECU umgesetzt. Auf das dazugehörige *detaillierte Hardwaredesign* wird im Folgenden genauer eingegangen.

Hierbei ist im verarbeitenden Teil insbesondere der uC, welcher keine interne Redundanz aufweist [ISO26262, 5-AnnexE], als komplexer Baustein im Vergleich zu der Zusatzverschaltung durch beispielsweise Widerstände anzusehen. Ein außerhalb des uC in einem eigenen Package befindlicher externer Watchdog dient einer Überwachung. Hierfür wird das Lebenssignal des uC eingelesen. Sollte dieses eine bestimmte Zeit ausbleiben, so wird der Ausgang des Watchdog auf Null gesetzt und damit das System auf einen sicheren Betriebszustand, im Englischen safe state, geschaltet. Bedingt durch interne Selbsttests und den externen Watchdog wird somit der Sicherheitsmechanismus SM4 realisiert.

Für die Funktion 1 wird der Temperatursensor R3 eingelesen, bei dem es sich um einen NTC handelt, da mit steigender Temperatur der Widerstandswert abnimmt. Dieser liefert sein Signal an den analogen Eingang ADC3 des uC. Das Eingangssignal wird nicht überwacht. Mit dem Ausgang Out2 des uC kann über den Transistor T71 der Controller I71 für das Ventil geregelt werden. Eine Überwachung im Sinne des Sicherheitsmechanismus SM1 ist durch die Rückkopplung auf den analogen Eingang ADC1 gegeben. Funktion 2 greift auf die Eingangswerte der zwei Raddrehzahlsensoren I1 und I2 zurück, welche über den gleichen Schaltungsaufbau jeweils mit einem Eingangspin des uC verbunden sind. Die Pulse der Sensoren werden eingelesen und über eine Mittelwertbildung im uC wird die Fahrzeuggeschwindigkeit bestimmt. Der Sicherheitsmechanismus SM2 vergleicht die Eingangswerte der 2 Sensoren, erkennt Ausfälle und stellt ggf. den sicheren Betriebszustand her. Sollte dies der Fall sein, wird zur Signalisierung an den Fahrer gleichzeitig die LED L1 eingeschaltet. Wie im Pfad von Funktion 1 liegt auch hier eine Rückkopplung vor, welche den Sicherheitsmechanismus SM3 für die Ausgangsstufe abbildet.

Angemerkt sei, dass etliche *Hardware-Bauteile* innerhalb des Schaltplanes zur Absicherung von ESD und elektrischem Schutz dienen [ISO26262, 5-Fig.E.2.note2/3/4]. Hierbei muss die Klassifikation der entsprechenden Ausfallarten individuell von Fall zu Fall unter Berücksichtigung der Häufigkeit sowie der Auswirkung des Verlustes der Absicherung vorgenommen werden [ISO26262, 5-Fig.E.2.note7].

5.11.3 Anforderungen an funktionale Sicherheit

Für den Beispiel-Schaltplan ergeben sich zwei funktionale Pfade von der Sensorik hin zur Aktuatorik. Beide Pfade ziehen sich durch den gemeinsamen uC. Beiden Funktionen F1 und F2 wird jeweils ein Sicherheitsziel unter dem Fahrzeugsystem assoziiert. Die Ventilregelung muss somit den zwei folgenden Sicherheitszielen nach ISO 26262 genügen: Das der Funktion 1 assoziierte Sicherheitsziel SG1 lautet „valve 2 shall not be closed for longer than x ms when the temperature is higher than 100° C“ und erhält die Einstufung ASIL-B. Der sichere Zustand wird repräsentiert durch ein geöffnetes Ventil 2. Dies ist der Fall, wenn I71 nicht bestromt wird. Das der Funktion 2 assoziierte Sicherheitsziel SG2 lautet „valve 1 shall not be closed for longer than y ms when the speed is higher than 100 km/h“ und wird als ASIL-C eingestuft. Der sichere Zustand wird erreicht, wenn das Ventil 1 geöffnet ist, bedingt dadurch dass I61 stromlos ist.

5.11.4 Fehlerdatentabellen für die Sicherheitsziele

In den Tabellen 5.8 und 5.9 sind die Fehlerdatentabellen, angelehnt an die Darstellung in ISO 26262, für die Sicherheitsziele SG1 und SG2 abgebildet[63]. Ersichtlich ist, dass für das Beispiel keine Ausfälle des Leitungssatzes berücksichtigt wurden. In der Fehlerdatentabelle werden neben den relevanten Fehlerinformationen auch die Klassifikation der Ausfallarten und sowohl die Summe über die spezifischen Ausfallraten der SPFs und RFs als auch über die latenten MPFs dargestellt. Exemplarisch sind für die beiden Sicherheitsziele zusätzlich die berechneten Metrikwerte für die SPFM und die LFM angegeben.

Component Name	Failure rate/FIT	Safety-related component to be considered in the calculations?	Failure Mode	Failure rate distribution	Failure mode that has the potential to violate the safety goal in absence of safety mechanisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage wrt. violation of safety goal	Residual or Single-Point Fault failure rate/FIT	Failure mode that may lead to the violation of safety goal in combination with an independent failure of another component?	Detection mean? Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage wrt. Latent failures	Latent Multiple-Point Fault failure rate/FIT	SF	SPF	RF	MPF
R3	3	YES	open	30 %	X	none	0 %	0,90						X		
			short	10 %									X			
			drift 0,5	30 %									X			
			drift 2	30 %	X		0 %	0,90						X		
R13	2	YES	open	90 %	X	none	0 %	1,80						X		
			short	10 %	X		0 %	0,20						X		
R23	2	YES	open	90 %		none							X			
			short	10 %	X		0 %	0,20						X		
C13	2	YES	open	20 %	X	none	0 %	0,40						X		
			short	80 %									X			
C23	2	NO	open	20 %									X			
			short	80 %									X			
WD	20	YES	Out. Stuck at 1	50 %					X	none	0 %	10,00				X
			Out. Stuck at 0	50 %									X			
T71	5	YES	open	50 %		SM1				SM1			X			
			short	50 %	X		90 %	0,25	X		80 %	0,45			X	X
R71	2	YES	open	90 %						none			X			
			short	10 %					X		0 %	0,20				X
R72	2	YES	open	90 %						none			X			
			short	10 %					X		0 %	0,20				X
R73	2	NO	open	90 %									X			
			short	10 %									X			
R74	2	YES	open	90 %					X	none	0 %	1,80				X
			short	10 %					X		0 %	0,20				X
I71	5	NO	open	70 %									X			
			short	30 %									X			
C71	2	YES	open	20 %					X	none	0 %	0,40				X
			short	80 %									X			
R81	2	NO	open	90 %									X			
			short	10 %									X			
L1	10	NO	open	90 %									X			
			short	10 %									X			
uC	100	YES	All1	50 %	X	SM4	90 %	5,00	X	SM4	100 %	0,00			X	X
			All2	50 %									X			
							Summe =	9,65			Summe =	13,25				

Total failure rate	163	Single-Point Fault Metric =	93,20 %
Total Safety Related	142		
Total not Safety Related	21	Latent Fault Metric =	89,99 %

Tab. 5.8: Fehlerdatentabelle nach [ISO26262, 5-Fig.E.2] für Sicherheitsziel SG 1 mit Ergebnissen der *„Evaluation of the hardware architectural metrics“*

[63] Folgende Anpassungen im Vergleich zur ISO 26262 wurden vorgenommen: Die Ausfallart *closed* wurde übergreifend zu *short* umbenannt. In Bezug auf Tabelle 5.8: *Hardware-Bauteil* 171 wurde zu I71 umbenannt und bei der Ausfallart *short* die Ausfallratenverteilung auf 30 % korrigiert. Die Ausfallart *open* von C71 hat den Eintrag 0 % bei der *Failure mode coverage with respect to latent failures* erhalten. In Bezug auf Tabelle 5.9: Die Ausfallratenverteilung von *Hardware-Bauteil* I61 wurde auf 30 % für die Ausfallart *short* korrigiert. Bei C61 wurde der fehlende Wert von 20 % für die Ausfallratenverteilung der Ausfallart *open* eingetragen.

Component Name	Failure rate/FIT	Safety-related component to be considered in the calculations?	Failure Mode	Failure rate distribution	Failure mode that has the potential to violate the safety goal in absence of safety mechanisms?	Safety mechanisms(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage wrt. violation of safety goal	Residual or Single-Point Fault failure rate/FIT	Failure mode that may lead to the violation of safety goal in combination with an independent failure of another component?	Detection means? Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage wrt. Latent failures	Latent Multiple-Point Fault failure rate/FIT	SF	SPF	RF	MPF
R11	2	YES	open	90 %	X	SM2	99 %	0,02	X	SM2	100 %	0,00			X	X
			short	10 %	X		99 %	0,00	X		100 %	0,00			X	X
R12	2	YES	open	90 %	X	SM2	99 %	0,02	X	SM2	100 %	0,00			X	X
			short	10 %	X		99 %	0,00	X		100 %	0,00			X	X
R21	2	YES	open	90 %	X	SM2	99 %	0,02	X	SM2	100 %	0,00			X	X
			short	10 %	X		99 %	0,00	X		100 %	0,00			X	X
R22	2	YES	open	90 %	X	SM2	99 %	0,02	X	SM2	100 %	0,00			X	X
			short	10 %	X		99 %	0,00	X		100 %	0,00			X	X
C11	2	YES	open	20 %	X	SM2	99 %	0,00	X	SM2	100 %	0,00			X	X
			short	80 %	X		99 %	0,02	X		100 %	0,00			X	X
C12	2	YES	open	20 %	X	SM2	99 %	0,00	X	SM2	100 %	0,00			X	X
			short	80 %	X		99 %	0,02	X		100 %	0,00			X	X
C21	2	YES	open	20 %		SM2				SM2			X			
			short	80 %	X		99 %	0,02	X		100 %	0,00			X	X
C22	2	YES	open	20 %		SM2				SM2			X			
			short	80 %	X		99 %	0,02	X		100 %	0,00			X	X
I1	4	YES	open	70 %	X	SM2	99 %	0,03	X	SM2	100 %	0,00			X	X
			short	20 %	X		99 %	0,01	X		100 %	0,00			X	X
			drift 0,5	5 %	X		99 %	0,00	X		100 %	0,00			X	X
			drift 2	5 %									X			
I2	4	YES	open	70 %	X	SM2	99 %	0,03	X	SM2	100 %	0,00			X	X
			short	20 %	X		99 %	0,01	X		100 %	0,00			X	X
			drift 0,5	5 %	X		99 %	0,00	X		100 %	0,00			X	X
			drift 2	5 %									X			
WD	20	YES	open	50 %					X	none	0 %	10,00				X
			short	50 %									X			
T61	5	YES	open	50 %		SM3				SM3			X			
			short	50 %	X		90 %	0,25	X		100 %	0,00			X	X
R61	2	YES	open	90 %						none			X			
			short	10 %					X		0 %	0,20				X
R62	2	YES	open	90 %						none			X			
			short	10 %					X		0 %	0,20				X
R63	2	NO	open	90 %									X			
			short	10 %									X			
R64	2	YES	open	90 %					X	none	0 %	1,80				X
			short	10 %					X		0 %	0,20				X
I61	5	NO	open	70 %									X			
			short	30 %									X			
C61	2	YES	open	20 %					X	none	0 %	0,40				X
			short	80 %									X			
R81	2	NO	open	90 %									X			
			short	10 %									X			
L1	10	NO	open	90 %									X			
			short	10 %									X			
uC	100	YES	All1	50 %	X	SM4	90 %	5,00	X	SM4	100 %	0,00			X	X
			All2	50 %									X			
							Summe =	5,48			Summe =	12,80				

Total failure rate	176	Single-Point Fault Metric =	96,51 %
Total Safety Related	157		
Total not Safety Related	19	Latent Fault Metric =	91,55 %

Tab. 5.9: Fehlerdatentabelle nach [ISO26262, 5-Fig.E.3] für Sicherheitsziel SG 2 mit Ergebnissen der „*Evaluation of the hardware architectural metrics*“

Als Sonderheiten seien die Ausfallarten All1 und All2 mit einer Ausfallratenverteilung von 50 % für den uC genannt. Hier schlägt ISO 26262 als konservativen Ansatz vor, die Hälfte der Ausfälle als sicheren Fehler (SF) einzustufen, falls für komplexe *Hardware-Bauteile* keine detaillierten Fehlerinformationen vorliegen. Somit wird All2 als sicherer Fehler eingestuft, während All1 zur Verletzung des Sicherheitsziels beiträgt.

Im Gegensatz zu den Widerständen wie beispielsweise R11, welche sich in einem digitalen Signalpfad befinden, erhält der Widerstand R3 zusätzlich Ausfallarten für Drift. Dies ist nicht nur durch die angewandte elektrische Leistung, sondern auch durch die Umgebungstemperatur bedingt.

Exemplarisch ist bei T71 ersichtlich, dass bei der Ausfallart Kurzschluss der Anteil, welcher zur direkten Verletzung des Sicherheitsziels führt, nicht mehr im Rahmen der latenten MPFs berücksichtigt wird, siehe hierzu auch die Berechnungsvorschrift nach Gleichung 5.5.

Innerhalb der beiden Fehlerdatentabellen ist ersichtlich, dass sich insbesondere die *Hardware-Bauteile* WD, L1 und uC durch hohe Werte der Ausfallrate auszeichnen und damit gegebenenfalls einen großen Einfluss auf die Ergebnisse der „*Evaluation of the hardware architectural metrics*“ nehmen.

Bei der Funktionsbeschreibung wurden die Sensoren und Aktuatoren als Eingangs- bzw. Ausgangsseite betrachtet. Im Zuge der Betrachtung der Sicherheitsziele sind sie diesen entsprechend zugeordnet und werden in den Fehlerdatentabellen gelistet. Folglich geht die Betrachtung der Hardware-Architekturmetriken über die Systemgrenzen der ECU hinaus. Jedoch sei angemerkt, dass für die Berechnung der Hardware-Architekturmetriken lediglich die sicherheitsbezogenen Sensoren berücksichtigt werden, jedoch nicht die nicht-sicherheitsbezogenen Aktuatoren.

Da die Verarbeitung beider Funktionen dem komplexen *Hardware-Bauteil* uC zugeordnet wird, ist dieser gemeinsam mit dem Watchdog als Gleichanteil unter beiden Fehlerdatentabellen gelistet. Die Signalisierung an den Fahrer über die LED L1 mit Vorwiderstand R81 wird beiden Sicherheitszielen zugeordnet, jedoch als nicht-sicherheitsbezogen eingestuft.

Im Kontext des Sicherheitsziels SG1 und dem Sicherheitsmechanismus SM1, welcher Ausfälle von Transistor T71 abdeckt, wird der sichere Zustand herbeigeführt, jedoch keine Rückmeldung an den Fahrer gegeben. Dieser erkennt den Ausfall lediglich an einer Degradation der Funktionalität. Daher wird in Bezug auf den Diagnoseabdeckungsgrad $K_{DC,RF}$ ein Wert von 90 % und in Bezug auf $K_{DC,MPF,L}$ ein Wert von lediglich 80 % zugewiesen. Für Transistor T61 lässt sich im Kontext von SG2 der Wert $K_{DC,MPF,L}$ des Sicherheitsmechanismus SM3 sogar auf 100 % einstufen.

Da der Sicherheitsmechanismus SM2 die Werte der Sensoren I1 und I2 miteinander vergleicht, sind sämtliche *Hardware-Bauteile* in diesen Pfaden durch den Sicherheitsmechanismus abgedeckt. Bedingt durch das Erkennen und Herbeiführen des sicheren Zustandes wird $K_{DC,RF}$ auf 99 % festgelegt und durch das zusätzliche Anschalten der LED als Signalisierung an den Fahrer wird $K_{DC,MPF,L}$ ein Wert von 100 % zugewiesen, da es sich vollständig um wahrgenommene Ausfälle handelt. Bedingt durch interne Selbsttests und die Überwachung durch den Watchdog kann für den Sicherheitsmechanismus SM4 der Wert $K_{DC,RF}$ auf 90 % und $K_{DC,MPF,L}$ sogar auf 100 % festgelegt werden.

Nach der Zielwerttabelle 5.2 ergibt sich für das Sicherheitsziel SG1 mit zugewiesenem ASIL-B für die SPFM ein geforderter Zielwert größer gleich 90 % und für die LFM größer gleich 60 %. Diese Werte werden deutlich erreicht. Für das zweite Sicherheitsziel SG2 mit assoziiertem ASIL-C sind die Zielwerte größer gleich 97 % für die SPFM und größer gleich 80 % für die LFM zu erreichen. Dies wird wie in Tabelle 5.9 ersichtlich nur für die LFM erreicht. Mit einem Wert von 96,5 % für die SPFM ist der entsprechende Zielwert nicht erreicht. Somit sind nicht alle Sicherheitsziele des Fahrzeugsystems abgesichert.

5.12 Umsetzung des detaillierten Hardwaredesigns in EDA-Tools

Um zunächst ein altbewährtes Bild auf die Schaltpläne zu bieten und auch eine mögliche Unterstützung für verteilte Entwicklungen im Rahmen eines DIA aufzugreifen, wurde der in ISO 26262 dargestellte Schaltplan für das *detaillierte Hardwaredesign* in den Electronic Design Automation (EDA) Werkzeugumgebungen EAGLE PCB Design Software und Altium Designer® nachgebildet.

5.12.1 EAGLE-Projekt

Die Umsetzung des Schaltplanes erfolgte in EAGLE PCB Design Software Version 6.4. Dies umfasste das Anlegen von Bibliotheken, die Erstellung des Schaltplanes basierend auf einer Bauteil-Bibliothek und eine optionale prototypische Anreicherung des Modells mit Fehlerinformationen im Kontext der Sicherheitsziele. Angemerkt sei, dass im Rahmen des vorgestellten Konzeptes die Anreicherung mit Fehlerinformationen auch unabhängig von der Schaltplanerstellung in einem anderem Werkzeug vorgenommen werden könnte. Ein Design des PCB war in diesem Zuge nicht erforderlich.

5.12.1.1 Angelegte Bibliotheken

Für die Umsetzung des Schaltplanes in EAGLE wurden insgesamt drei Bibliotheken angelegt, welche thematisch die *Hardware-Bauteile* mit Anreicherung von Fehlerinformationen, die Sicherheitsziele und die Sicherheitsmechanismen aufgreifen. Die erste Bibliotheksdatei beschreibt die für das Schaltplan-Beispiel der ISO 26262 relevanten *Hardware-Bauteil*-Typen. Die zwei weiteren Bibliotheksdateien stellen zweckentfremdete Elemente für die Definition von Sicherheitszielen sowie Sicherheitsmechanismen zur Verfügung, welche in den Schaltplan eingebracht werden können. Somit sind die entsprechenden Fehlerinformationen aus den Bibliotheksdateien auch in der Schaltplandatei von EAGLE vorhanden.

Die Bibliothek „ISO26262example_EAGLE_Parts.lbr“ dient der Erstellung einer gewöhnlichen Bauteil-Bibliothek. Zusätzlich wurden jedem *Hardware-Bauteil* ein Attribut für die Ausfallrate in FIT mit einer Fließkommazahl und entsprechende Ausfallarten mit den jeweiligen Ausfallratenverteilungen in Prozent hinterlegt, wie in Abb. 5.20 ersichtlich.

Technologies	Attributes		
	FAILURERATE	FM-OPEN	FM-SHORT
R-EU	2	90	10

Abb. 5.20: Exemplarische Darstellung in EAGLE: Attribute der Hardware-Bauteil-Typen innerhalb der Bibliothek

In der Bibliothek „ISO26262example_EAGLE_SafetyGoals.lbr“ wird jedes spezifische Sicherheitsziel (SG) erstellt. Unter dem Attributnamen „ASIL“ muss für den Wert der entsprechende ASIL eingetragen werden, siehe Abb. 5.21. Der unter dem Attributnamen „DESCRIPTION“ hinterlegte Wert dient einer textuellen Beschreibung des Sicherheitsziels als *String*.

Technologies	Attributes	
	ASIL	DESCRIPTION
SG1	B	"Valve 2 shall not be closed for longer than x ms when the temperature is higher than 100 °C"

Abb. 5.21: Exemplarische Darstellung in EAGLE: Attribute der Sicherheitsziele innerhalb der Bibliothek

In der Bibliothek „ISO26262example_EAGLE_SafetyMechanism.lbr“ wurden in gleicher Art und Weise wie bei der Bibliothek „ISO26262example_EAGLE_SafetyGoals.lbr“ die vorhandenen Sicherheitsmechanismen (SMs) angelegt. Diese erhalten zwei Attribute mit den Attributnamen „KDCSPF“ und „KDCMPFL“, um die prozentualen Werte der entsprechenden Diagnoseabdeckungsgrade zu hinterlegen, siehe Abb. 5.22.

Technologies	Attributes	
	KDCMPFL	KDCSPF
SM1	80	90

Abb. 5.22: Exemplarische Darstellung in EAGLE: Attribute der Sicherheitsmechanismen innerhalb der Bibliothek

5.12.1.2 Erstellter Schaltplan

Unter Verwendung der angelegten Bibliotheken konnte der Schaltplan nach dem Beispiel von ISO 26262 nachgebildet werden. Wie in Abb. 5.23 ersichtlich, befinden sich zwei instanziierte SGs und vier SMs auf dem Schaltplan. Diese werden benötigt, um im Rahmen der Schaltplandatei sämtliche Fehlerinformationen abzubilden.

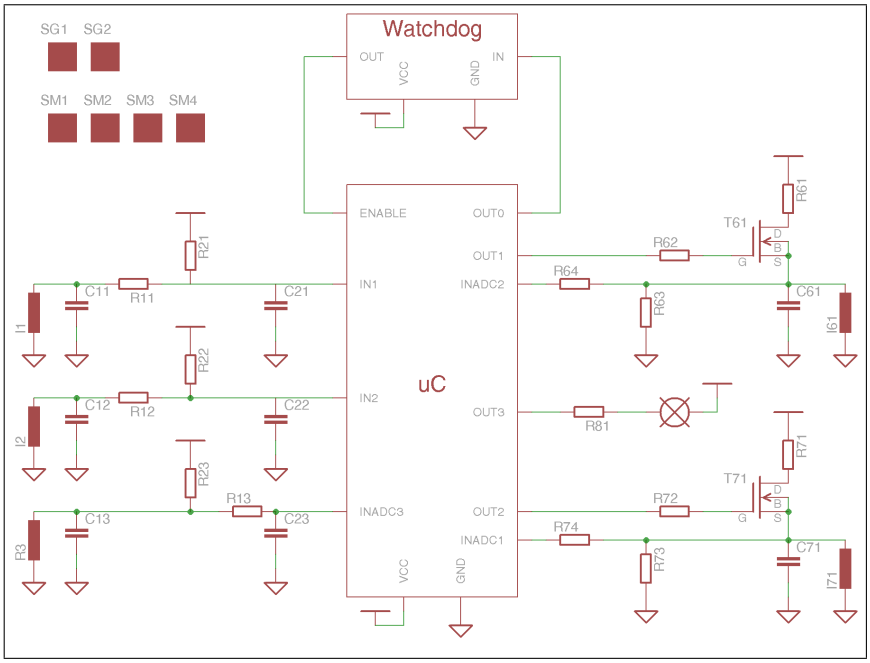


Abb. 5.23: Nach [ISO26262, 5-Fig.E.1] erstellter Schaltplan in EDA-Umgebung EAGLE PCB Design Software

5.12.1.3 Anreicherung mit Fehlerinformationen im Kontext der Sicherheitsziele

Nach der Erstellung des Schaltplanes in EAGLE PCB Design Software können die Bauteile in Bezug zu den Sicherheitszielen und Sicherheitsmechanismen gebracht werden. Da innerhalb von EAGLE PCB Design Software keine Verknüpfungskonstrukte oder strukturierende Kontexte wie in Architekturbeschreibungssprachen zur Verfügung stehen, wird dies über zusätzliche Attribute auf den instanziierten *Hardware-Bauteilen* durchgeführt, siehe Abb. 5.24.

Hierbei sind die Attribute, welche aus der Bibliothek übernommen wurden, durch das Buch-Symbol kenntlich gemacht. Es handelt sich um die Ausfallrate und die Ausfallarten

mit ihren Ausfallratenverteilungen, welche auf den Instanzen nicht verändert werden können. In Abb. 5.24 sind exemplarisch die Attribute des *Hardware-Bauteils* R11 gelistet.

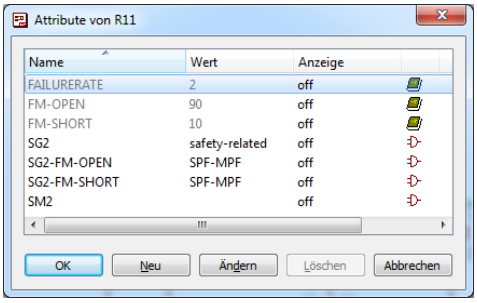


Abb. 5.24: Exemplarische Darstellung: Attribute des instanziierten Hardware-Bauteils R11

Des Weiteren wird durch ein Attribut mit dem Namen des Sicherheitsziels der Bezug zu diesem hergestellt. Der Wert des Attributs legt fest, ob das Bauteil im Kontext des jeweiligen Sicherheitsziels als sicherheitsbezogen eingestuft wird. Für jede innerhalb der Bibliothek angelegte Ausfallart muss wiederum im Kontext eines einzelnen *Hardware-Bauteils* eine entsprechende Klassifikation hinterlegt werden. Dies erfolgt innerhalb des Attributnamens über das Voranstellen des Namens des Sicherheitsziels zum Namen der Ausfallart. Ist ein Sicherheitsmechanismus zur Abdeckung vorhanden, so muss dieser über einen einfachen Eintrag als Attributname hinzugefügt werden. Hierbei ist eine Zuweisung auf Ebene des *Hardware-Bauteils* vorgesehen.

5.12.1.4 Verwendung der Schaltplandatei

Das erstellte EAGLE-Projekt besteht aus drei Bibliotheksdateien im Dateiformat *.lbr und einer Schaltplandatei im Dateiformat *.sch, welche zusätzlich die definierten Devices aus den Bibliotheken enthält. Bei der Schaltplandatei (*.sch) handelt es sich um eine XML, welche bedingt durch das nicht-proprietäre Dateiformat für eine Nutzung in anderen Toolumgebungen eingelesen werden kann. Diese beinhaltet die Bibliotheken der *Hardware-Bauteile*, welche sich im Schaltplan befinden, und alle instanziierten *Hardware-Bauteile* mit den entsprechenden Fehlerinformationen.

5.12.2 Altium-Projekt

Im Tool Altium Designer® wurde zur Abbildung des Schaltplanes aus ISO 26262 eine Bauteil-Bibliothek angelegt, welche alle relevanten *Hardware-Bauteile* des Beispiels beinhaltet. Der erstellte Schaltplan enthält keine Fehlerinformationen und spiegelt somit die strukturelle Beschreibung des *detaillierten Hardwaredesigns* wider, wie in Abb. 5.25 ersichtlich.

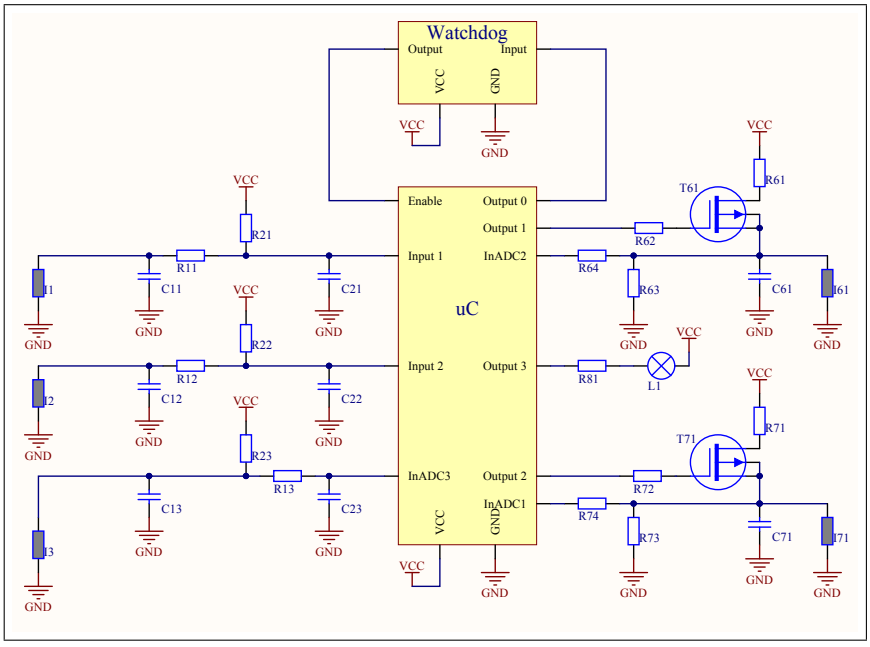


Abb. 5.25: Nach [ISO26262, 5-Fig.E.1] erstellter Schaltplan in EDA-Umgebung Altium Designer®

5.13 Übersicht der prototypischen Implementierungen und umgesetzten Features

Abb. 5.26 stellt eine Übersicht sämtlicher prototypischer Implementierungen sowie deren Schnittstellen und Austauschformate für die modellbasierte Unterstützung der Entwicklung funktional sicherer Hardware dar. Die Implementierungen basieren auf den vorgestellten Metamodell-Vorschlägen, spezifischen Vorgehensweisen und Anwendungen der Hardware-Sicherheitsevaluationen.

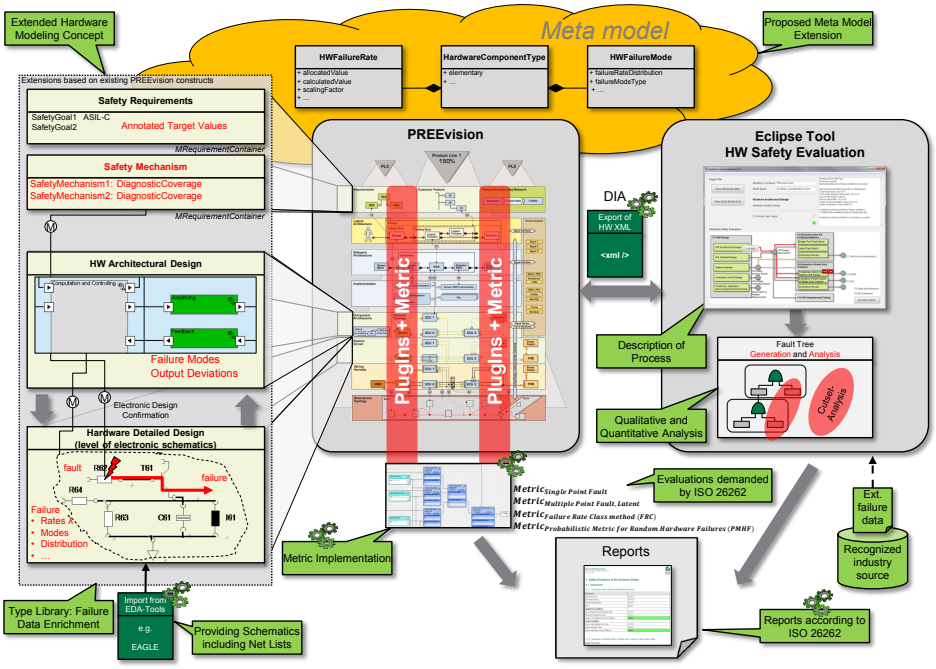


Abb. 5.26: Übersicht der prototypischen Implementierungen für die Entwicklung funktional sicherer Hardware

Für die wissenschaftlich prototypische Umsetzung einer durchgängigen strukturellen Hardware-Modellierung auf beiden Abstraktionsebenen eignet sich die modellbasierte EEA-Entwicklungsumgebung PREEvision, da diese die relevanten Konstrukte in spezifischen grafischen Diagrammeditoren zur Verfügung stellt und Erweiterungen über beispielsweise generische Attribute ermöglicht. Zudem ist es ein in der Industrie etabliertes Werkzeug, unter anderem bedingt durch die nahtlose Nachverfolgbarkeit von Anforderungen und eine grafische Aufbereitung der Modelle. Basierend auf dem Metrik-Framework, welches Berechnungen auf dem Datenmodell ermöglicht, konnte eine

prototypische Umsetzung der Hardware-Sicherheitsevaluationen erfolgen. In anderen Entwicklungsumgebungen wären die Implementierungen prinzipiell jedoch auch möglich. Für die dargestellten Implementierungen wurde PREEvision Version 6.0 SP1 genutzt.

Um die Hardware-Sicherheitsevaluationen auch im Rahmen eines DIA unabhängig von der genutzten Modellierungsumgebung ausführen zu können, wurde eine wissenschaftlich prototypische Implementierung in einem Eclipse-Projekt umgesetzt. Dieses unterstützt im Zuge des *Hardware-Architekturdesigns* zusätzlich eine qualitative und quantitative Fehlerbaumanalyse, die grafische Darstellung von Fehlerbäumen sowie eine automatische Klassifikation der Ausfallarten. Hierbei wurde für den Austausch der strukturellen Modelle, Fehlerinformationen sowie Zielwerte im Rahmen des DIA die in Kapitel 5.5 vorgestellte XSD verwendet. Die Ergebnisse der Hardware-Sicherheitsevaluationen sowie zusätzliche Informationen können sowohl aus PREEvision als auch aus dem Eclipse-Tool in Form von Reports zu Dokumentationszwecken exportiert werden.

Detaillierte Hardwaredesigns können in etablierten EDA/Electrical CAD (ECAD) Werkzeugen wie Altium Designer® oder CadSoft EAGLE PCB Design Software strukturell erstellt werden, siehe auch Kapitel 5.12. Die Schaltplandateien inklusive der Netzlisten im Rahmen von EAGLE sind sowohl in PREEvision als auch in das Eclipse-Projekt importierbar. Eine Erweiterung um Schnittstellen beispielsweise für Electronic Design Interchange Format (EDIF) und IP-XACT ist möglich. Zusätzlich können Fehlerinformationen aus anerkannten Industriesammlungen für die prototypischen Implementierungen berücksichtigt werden.

Im Folgenden wird auf die Modellierungsunterstützung in PREEvision einschließlich dem Aufbau der Demomodelle für das Beispiel der Ventilregelung nach Kapitel 5.11 eingegangen. Weiterhin werden die prototypischen Implementierungen für die modellbasierte Ausführung der Hardware-Sicherheitsevaluationen innerhalb von PREEvision sowie des wissenschaftlichen Eclipse-Projektes detailliert vorgestellt.

5.14 Prototypische Implementierung in PREEvision

Die erarbeiteten Konzepte konnten durch eine prototypische Implementierung in der modellbasierten Entwicklungsumgebung PREEvision angewendet und verifiziert werden[64]. Ein zusätzlicher Gesichtspunkt war die Harmonisierung mit bereits etablierten Entwicklungsprozessen. Hinsichtlich des Metamodell-Vorschlages aus Kapitel 5.4 mussten bei der prototypischen Umsetzung in PREEvision Kompromisse eingegangen werden.

Die innerhalb von PREEvision umgesetzten Konzepte und Implementierungen werden im Kontext des Beispiels der Ventilregelung nach Kapitel 5.11 dargestellt. In Abb. 5.27 sind das erarbeitete *Hardware-Architekturdesign* sowie das *detaillierte Hardwaredesign* ge-

[64] Erster Prototyp wurde innerhalb der betreuten Arbeit [Otten2012] umgesetzt.

meinsam abgebildet[65]. Exemplarisch sind auch die Verknüpfungen der Sicherheitsmechanismen SM1 und SM4 auf Ebene des *detaillierten Hardwaredesigns* eingeblendet.

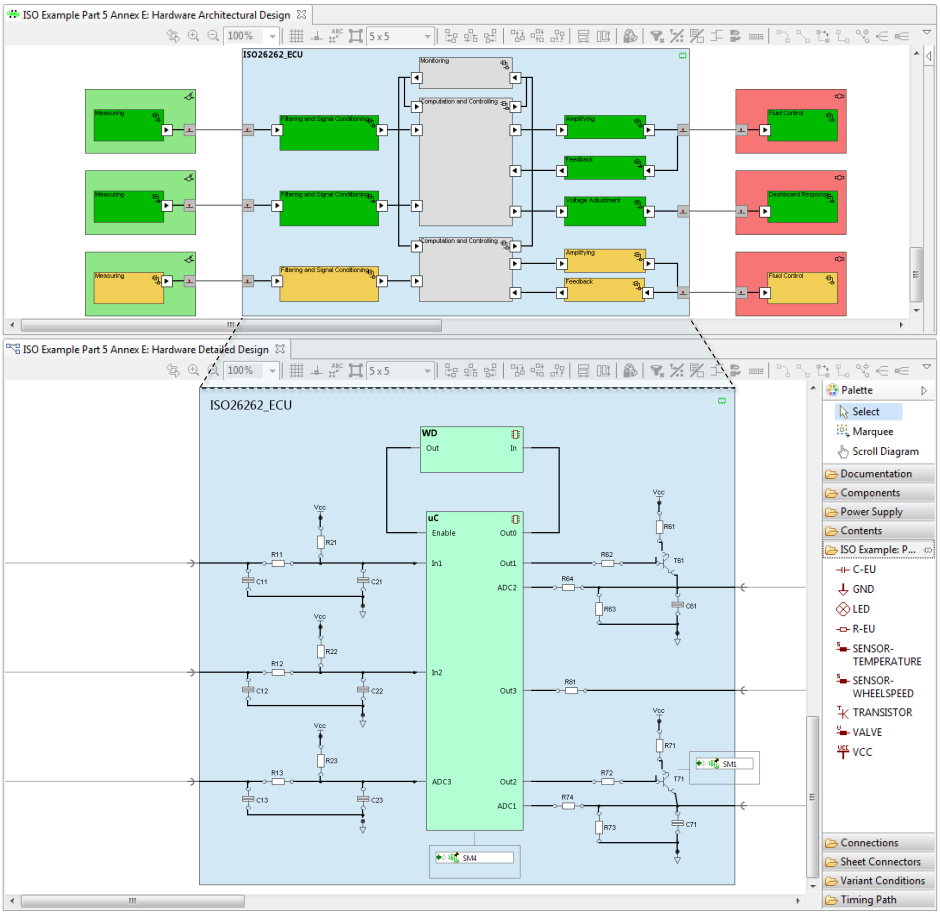


Abb. 5.27: Erarbeitetes Modell für das Hardware-Architekturdesign und detaillierte Hardwaredesign

5.14.1 Modellierung des Hardware-Architekturdesigns

Das *Hardware-Architekturdesign* besteht vorrangig aus den *Hardware-Komponenten*, für welche im Rahmen der prototypischen Implementierung die PREEvision Metamodell-

[65] Anmerkung zur grafischen Darstellung: Aufgrund der in PREEvision v6.0.1 verfügbaren Mittel war die grafische Repräsentation von Eingangspins rechtsseitig und Ausgangspins linksseitig geöffnet. Die Bausteine I1, I2, R3, I61, I71 sind für das *detaillierte Hardwaredesign* gesondert dargestellt.

Klasse *HardwareModule* verwendet wurde. Zusätzlich ist der mittlere Teil der Schaltung in einer übergeordneten *ECU* sowie die Sensorik bzw. Aktuatorik in *Sensor* und *Actuator* abgebildet, um eine klare Trennung herbeizuführen. Spezifizierte Ports, welche zusätzlich noch Richtungsinformationen beinhalten, sowie Verbindungen komplettieren das *Hardware-Architekturdesign*. Aufgrund der vorhandenen Gegebenheiten wurde im Rahmen des *Hardware-Architekturdesigns* auf die Nutzung einer Typenbibliothek verzichtet und somit die Fehlerinformationen direkt für die *Hardware-Komponenten* eingebracht. Zur Beschreibung der Ausgangsabweichungen sind zusätzliche Attribute eingebracht, welche für die qualitative Fehlerbaumanalyse benötigt werden.

5.14.2 Modellierung des detaillierten Hardwaredesigns

Um den Vorgehensweisen für die Erstellung von Schaltplänen zu genügen, wurde eine Typenbibliothek *LibraryPackage* für die benötigten *Hardware-Bauteile*, in PREEvision durch die Klasse *HWDevice* repräsentiert, angelegt. Diese umfasst die entsprechenden *Hardware-Bauteil*-Typen in Form der Klasse *HWDeviceType* sowie eine zugehörige grafische Repräsentation in den Diagrammen.

Für die grafischen Repräsentationen wurde basierend auf einem *SymbolTemplatePackage* jeweils ein *TemplateDiagram* genutzt, um einerseits die geometrischen Formen bzw. Schaltsymbole der einzelnen *Hardware-Bauteile* abzubilden und andererseits über *InternalSchematicConnectorIn* und *InternalSchematicConnectorOut* die entsprechenden Ein- bzw. Ausgangspins bereitzustellen. Eine Zuordnung der grafischen Repräsentation zu den Bibliothekstypen der *Hardware-Bauteile* ist durch PREEvision bereitgestellt.

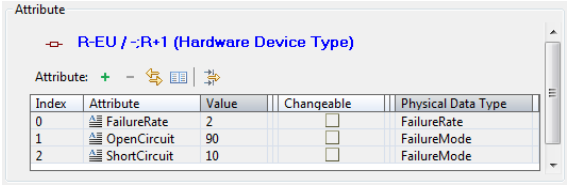


Abb. 5.28: Beispielhafte Darstellung der Attribute für den *HWDeviceType* R-EU zur Hinterlegung von Fehlerinformationen

Angelehnt an dem vorgestellten Metamodell zur Fehlerbeschreibung, siehe Kapitel 5.4, wurden zusätzliche Attribute für die Hardware-Bibliotheksbauteile eingeführt. Die Attribute für einen exemplarisch abgebildeten Bibliothekstyp R-EU eines *Hardware-Bauteils*, hier ein Widerstand, sind in Abb. 5.28 dargestellt. Da sowohl die Ausfallrate als auch die Ausfallarten spezifisch für den jeweiligen Bibliothekstyp sind, dürfen diese nicht veränderbar sein. Für sämtliche eingeführte Attribute wurde in der Bibliothek ein spezifischer Datentyp mit dazugehöriger Einheit hinterlegt.

Die grafische Repräsentation ist beispielhaft für einen Widerstand in Abb. 5.29 dargestellt. Neben dem grafischen Schaltsymbol und dem Ein- bzw. Ausgangspin sind auch zusätzliche Beschriftungen eingebracht. Hier werden neben dem Namen auch der Widerstandswert, die Ausfallrate sowie die spezifischen Ausfallarten angezeigt. Der Operator „%“ kennzeichnet eine dynamische Wertzuweisung aus dem dazugehörigen Attribut.

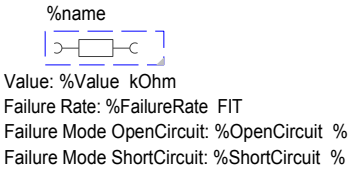


Abb. 5.29: Beispielhafte Darstellung eines *TemplateDiagram* für den *HWDeviceType* R-EU einschließlich Fehlerinformationen

5.14.3 Modellierung der Sicherheitsziele und Sicherheitsmechanismen

Für die Modellierung der Sicherheitsziele, Sicherheitsmechanismen sowie *dedizierten Maßnahmen* wurde die Anforderungsebene in PREEvision verwendet und diese durch einzelne Anforderungspakete zur Unterscheidung strukturiert. Diese sind unabhängig von der Abstraktionsebene des *Hardwaredesigns* eingebracht. Abb. 5.30 zeigt die erstellte hierarchische Struktur in der Anforderungsebene mit entsprechenden Erweiterungen in Form von generischen Attributen.

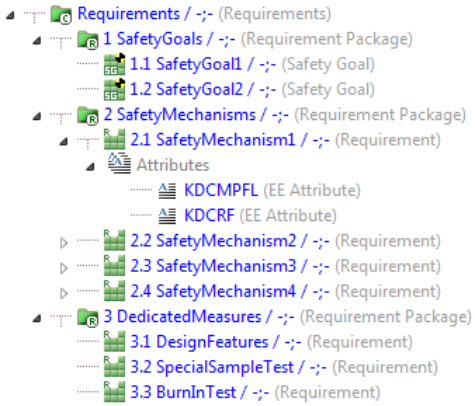


Abb. 5.30: Modellierte Sicherheitsziele und Sicherheitsmechanismen

Im Zuge der ISO 26262 [ISO26262, 5-AnnexE] wird von zugehörigen Sicherheitszielen gesprochen, welche entsprechend in der Modellierung umgesetzt wurden. PREEvision

ermöglicht bereits das Hinterlegen des dazugehörigen ASIL und eine textuelle Beschreibung des Sicherheitsziels. Eine Verfeinerung in Hardware-Sicherheitsanforderungen ist insbesondere im Rahmen des *detaillierten Hardwaredesigns* denkbar, da es sich hier um eine sehr detaillierte Abstraktionsebene handelt.

Für die Modellierung der Sicherheitsmechanismen wurde die Anforderungsebene zweckentfremdet, da keine spezifischen Konstrukte in PREEvision existent waren. Somit sind die Sicherheitsmechanismen in der Modellierung allesamt als Anforderungen repräsentiert. Dies ermöglicht sowohl ein Mapping der Sicherheitsmechanismen in die Hardwareebenen als auch die Erweiterung durch generische Attribute zum Hinterlegen der Diagnoseabdeckungsgrade für RFs und latente MPFs.

Die nach ISO 26262 beschriebenen Beispiele für *dedizierte Maßnahmen* [ISO26262, 5-9.4.2.4] wurden in einem ersten Zug ebenfalls über Konstrukte aus der Anforderungsebene abgebildet. Somit können diese sowohl mit Bibliothekstypen als auch instanziierten *Hardware-Elementen* verknüpft werden.

Der Bezug von Sicherheitszielen bzw. Sicherheitsmechanismen zu den einzelnen *Hardware-Komponenten* oder *Hardware-Bauteilen* wird innerhalb von PREEvision über Mappings hergestellt. Durch die Verknüpfung eines Sicherheitsziels mit einem *Hardware-Element* wird die Zugehörigkeit des *Hardware-Elements* zum entsprechenden funktionalen Pfad spezifiziert. Beispielhaft ist in Abb. 5.31 das Mapping eines Sicherheitsziels und eines Sicherheitsmechanismus auf verschiedene *Hardware-Elemente* dargestellt.

Abb. 5.31: Mapping von Sicherheitszielen und Sicherheitsmechanismen auf Hardware-Elemente

5.14.4 Einstufung der Sicherheitsbezogenheit und Klassifikation der Ausfallarten

Basierend auf der im Rahmen eines Mappings spezifizierten Zugehörigkeit eines *Hardware-Elements* zu einem Sicherheitsziel kann eine erste modellbasierte Unterstützung in Bezug auf die Sicherheitsbezogenheit der *Hardware-Elemente* sowie die Klassifikation der Ausfallarten erfolgen. Da die Sicherheitsbezogenheit sowie die Klassifikation der Ausfallarten im Gegensatz zu den in der Bibliothek hinterlegten Fehlerinformationen auf Instanzebene vorgenommen werden muss, unterstützt hierbei die in Abb. 5.32 abgebildete Metrik vorbereitend die modellbasierte Umsetzung.

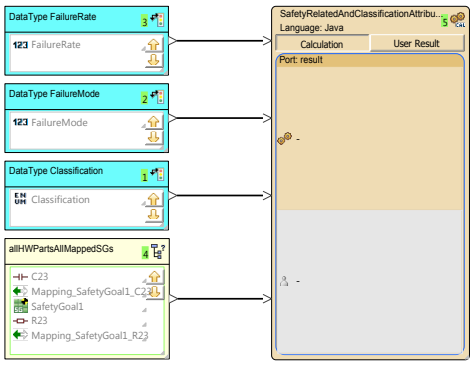


Abb. 5.32: Metrik zur Erstellung der Konstrukte zur Einstufung der Sicherheitsbezogenheit und zur Klassifikation von Ausfallarten

Die Metrik kann jeweils im Anschluss an neu angelegte Mappings der Sicherheitsziele auf *Hardware-Elemente* ausgeführt werden, um die benötigten Modellkonstrukte zu erstellen. Hierzu wird für jedes gemappte *Hardware-Element* ein boolesches Attribut zur Einstufung der Sicherheitsbezogenheit im Kontext jedes Sicherheitsziels angelegt. Falls für ein bestimmtes Mapping eines Sicherheitsziels bereits ein entsprechendes Attribut vorhanden ist, wird dieses bei erneuter Ausführung der Metrik nicht überschrieben. Somit können jederzeit Designänderungen, beispielsweise das Hinzufügen und Verknüpfen weiterer Sicherheitsziele, vorgenommen werden.

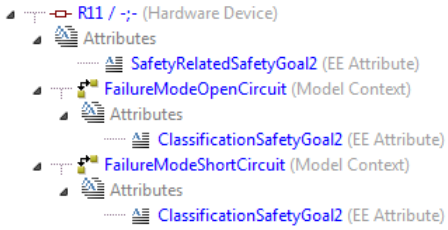


Abb. 5.33: Modellansicht eines Hardware-Elements im Kontext der Sicherheitsbezogenheit sowie der Klassifikation der Ausfallarten

Die Klassifikation der Ausfallarten muss auf Ebene der spezifischen *Hardware-Elemente* und zusätzlich im Kontext eines jeweiligen Sicherheitsziels erfolgen. Hierzu wird ein Modellkontext für jede Ausfallart des *Hardware-Elements* angelegt. Anschließend wird innerhalb des Modellkontextes für jedes zugewiesene Sicherheitsziel ein Attribut für die Klassifikation erstellt. Hierbei wird der hinterlegte Enumerations-Datentyp für die Klassifikation den Attributen zugewiesen, um die entsprechenden Auswahlmöglichkeiten für

SF, SPF oder RF und MPF vorzugeben. Die sich aus den Erweiterungen ergebende Struktur für das exemplarische *Hardware-Element* R11 ist in Abb. 5.33 aufgezeigt.

5.14.5 Fehlerdatentabelle und Hardware-Sicherheitsevaluationen

Die Hardware-Sicherheitsevaluationen wurden im Rahmen von PREEvision basierend auf dem Metrik-Framework umgesetzt. Ein Screenshot der Gesamtstruktur der Metrik für die modellbasierte Durchführung der Hardware-Sicherheitsevaluationen einschließlich der Fehlerdatentabelle und eingebrachter Modellabfragen ist in Abb. 5.34 dargestellt. Die Ergebnisse der Hardware-Sicherheitsevaluationen sowie Zwischenergebnisse, wie beispielsweise die Fehlerdatentabelle, sind für die entsprechenden Dokumentationen in Report-Ergebnisblöcken bereitgestellt. Für die Ausführung der Metrik können ein oder mehrere zu untersuchende Sicherheitsziele im Fokus stehen. Im Folgenden wird auf die einzelnen umrandeten Bereiche der Metrik detaillierter eingegangen.

5.14.5.1 Erstellung und Befüllung der Fehlerdatentabelle

Basierend auf der vorgestellten Modellierung des *Hardwaredesigns* auf verschiedenen Abstraktionsebenen, der Erweiterung mit Fehlerinformationen sowie den Verknüpfungen mit Sicherheitszielen und Sicherheitsmechanismen wird die entsprechende Fehlerdatentabelle für jedes Sicherheitsziel erstellt. Diese werden basierend auf implementierten Modellabfragen sowie Berechnungsblöcken im Rahmen der Metrik automatisiert befüllt, einschließlich der Berechnung der spezifischen Ausfallraten. Die Fehlerdatentabellen bilden die Grundlage für die modellbasierte Durchführung der Hardware-Sicherheitsevaluationen.

Die Fehlerinformationen können über den Typ eines *Hardware-Elements* aus der Bibliothek akquiriert werden. Die Einstufung der Sicherheitsbezogenheit sowie die Klassifikation der Ausfallarten werden über die zugehörigen Konstrukte der *Hardware-Elemente* vorgenommen. Im Rahmen der Befüllung der Fehlerdatentabelle sind zusätzliche Überprüfungen implementiert. In Bezug auf die Fehlerinformationen muss jeder Typ in der Bibliothek genau ein Attribut für die Ausfallrate besitzen. Der Wert des Attributs muss größer als 0 sein. Bezüglich der Ausfallarten muss mindestens ein Attribut vorhanden sein. Der Wertebereich der entsprechenden Ausfallratenverteilung ist hier von einschließlich 0 bis 100, da die Angabe in Prozent erfolgt. Zusätzlich muss die Summe aller Ausfallratenverteilungen für einen Typ genau 100 Prozent ergeben. Falls in der Bibliothek an dieser Stelle Inkonsistenzen gefunden werden, wird eine entsprechende Fehlermeldung ausgegeben und die anschließende Ausführung der Hardware-Sicherheitsevaluationen nicht zugelassen.

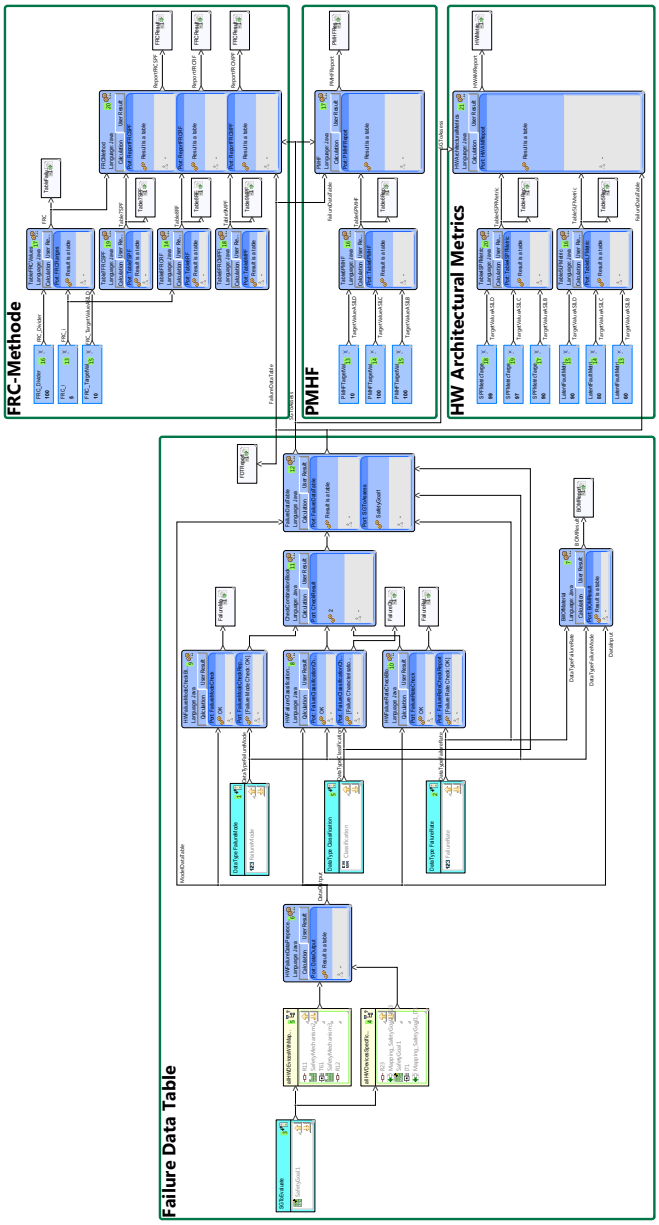


Abb. 5.34: Metrik zur Erstellung der Fehlerdatentabelle und Durchführung der Hardware-Sicherheitsevaluationen

Im Bezug auf die Klassifikationen der Ausfallarten wird überprüft, ob für jede Ausfallart eine zugehörige Klassifikation ausgewählt wurde. Falls dies nicht der Fall ist, wird eine Warnung an den Entwickler gegeben, die anschließenden Hardware-Sicherheitsevaluationen können dennoch ausgeführt werden. Dies ermöglicht eine erste vorläufige Evaluation des *Hardwaredesigns*, auch wenn noch nicht alle Ausfallarten klassifiziert wurden.

5.14.5.2 Anwendung der „*Evaluation of the hardware architectural metrics*“

In Anschluss an die Befüllung der Fehlerdatentabelle und nach erfolgter Überprüfung auf Konsistenz kann die Ausführung der Hardware-Architekturmetriken initiiert werden. Die Zielwerte müssen der Metrik übergeben werden und sind aufgrund von [ISO26262, 5-8.4.5] und [ISO26262, 5-8.4.6] in Form von Parameterblöcken individuell festlegbar. Unter dem Gesichtspunkt der Nachvollziehbarkeit und Nachweispflicht ist eine Verknüpfung und Einbringung in den entsprechenden Bericht notwendig.

Um die Berechnungen der Hardware-Architekturmetriken basierend auf der Fehlerdatentabelle durchzuführen, werden für die SPFM und die LFM die Gleichungen 5.8c und 5.9c verwendet. Hierbei ist für beide Metriken die Summenbildung über sämtliche Ausfallraten der einem Sicherheitsziel zugehörigen *Hardware-Elemente* erforderlich. Es werden jedoch nur die *Hardware-Elemente* betrachtet, welche als sicherheitsbezogen eingestuft sind. Des Weiteren gilt es, die Summen über die spezifischen Ausfallraten hinsichtlich einer direkten Verletzung des Sicherheitsziels oder einer Verletzung in Kombination mit einem weiteren Ausfall zu berechnen. Somit ist dies sowohl für die SPFs und RFs erforderlich, welche gemeinsam über $\sum_{SR,HW} \lambda_{SPF,RF}$ aufsummiert werden, als auch für die latenten MPFs, welche sich über $\sum_{SR,HW} \lambda_{MPF,L}$ ergeben.

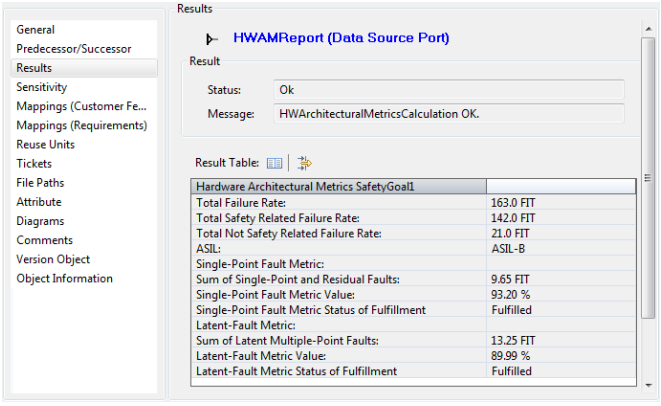


Abb. 5.35: Anwendung der Hardware-Architekturmetriken

Für die Ergebnisausgabe sind insbesondere die Ergebnisse der SPFM und der LFM von Bedeutung. Gleichwohl muss gegen die aktuell eingepflegten Zielwerte verifiziert werden. Die Ergebnisse sowie hilfreiche zusätzliche Informationen bzw. Zwischenergebnisse können, wie in Abb. 5.35 gezeigt, ausgegeben werden.

5.14.5.3 Anwendung der „*Evaluation of Probabilistic Metric for random Hardware Failures (PMHF)*“

Die Evaluation einer PMHF wurde im Rahmen dieses Ansatzes für das *Hardware-Architekturdesign* basierend auf einer qualitativen und quantitativen FTA unterstützt, vgl. Kapitel 5.9 und dahingehend im Kontext des Eclipse-Projekts prototypisch umgesetzt. Im Rahmen der Implementierung in PREEvision wurde die Umsetzung einer „vereinfachten PMHF“ nach Kapitel 5.6.1 als erste Approximation vorgenommen. Hierzu sind die Zielwerte nach [ISO26262, 5-9.4.2.1] in Form von Parameterblöcken bereitgestellt.

5.14.5.4 Anwendung der „*Evaluation of each cause of safety goal violation*“

Für die modellbasierte Evaluation im Rahmen der FRC-Methode sind die folgenden Parameter zu übergeben: Der Zielwert und der entsprechende Teiler für die Ausfallratenklasse FRC 1 sowie die Gesamtanzahl der benötigten Ausfallratenklassen. Hieraus sind die beschriebenen Zielwerttabellen nach [ISO26262, 5-9.4.3.5/6/11] teilweise dynamisch aufgebaut. Die Evaluation erfolgt für SPFs, RFs und MPFs in einem Berechnungsblock. Die Untersuchungen basieren auf den nach ISO 26262 standardmäßig vorgeschlagenen Ausfallratenklassen [ISO26262, 5-9.4.3.3].

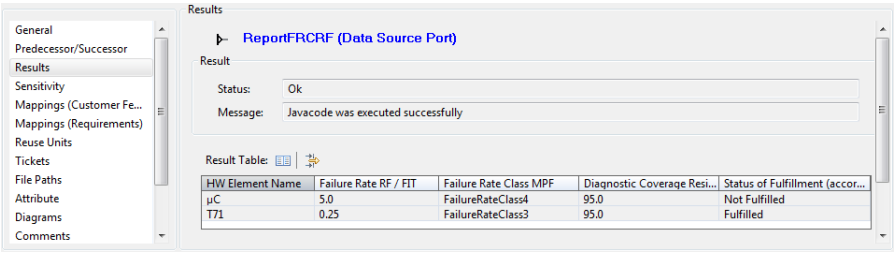
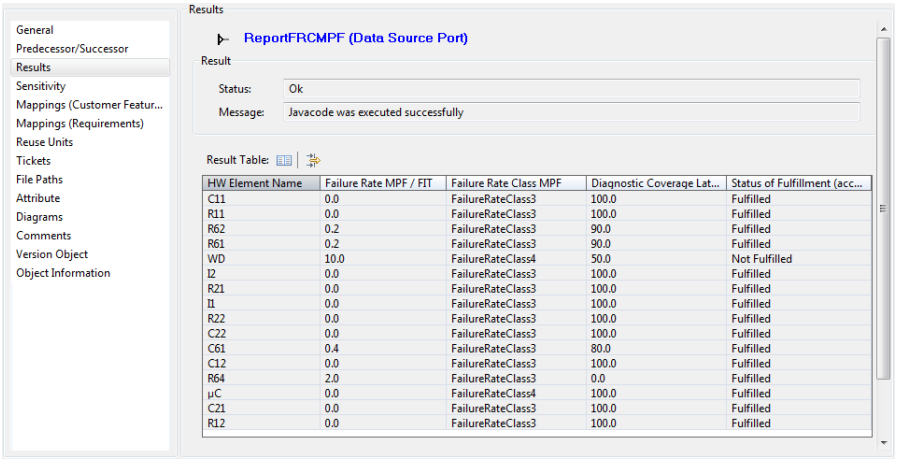


Abb. 5.36: Anwendung der FRC-Methode für RFs

Exemplarisch ergibt die Anwendung der „*Evaluation of each cause of safety goal violation*“ für die zwei *Hardware-Bauteile* T71 und uC, welche RFs unter dem Sicherheitsziel SG1 mit zugewiesenem ASIL-B sind, bezogen auf die Zielwerte für den uC ein „Not Fulfilled“ und für T71 ein „Fulfilled“, siehe Abb. 5.36.



HW Element Name	Failure Rate MPF / FIT	Failure Rate Class MPF	Diagnostic Coverage Lat...	Status of Fulfillment (acc...
C11	0.0	FailureRateClass3	100.0	Fulfilled
R11	0.0	FailureRateClass3	100.0	Fulfilled
R62	0.2	FailureRateClass3	90.0	Fulfilled
R61	0.2	FailureRateClass3	90.0	Fulfilled
WD	10.0	FailureRateClass4	50.0	Not Fulfilled
I2	0.0	FailureRateClass3	100.0	Fulfilled
R21	0.0	FailureRateClass3	100.0	Fulfilled
I1	0.0	FailureRateClass3	100.0	Fulfilled
R22	0.0	FailureRateClass3	100.0	Fulfilled
C22	0.0	FailureRateClass3	100.0	Fulfilled
C61	0.4	FailureRateClass3	80.0	Fulfilled
C12	0.0	FailureRateClass3	100.0	Fulfilled
R64	2.0	FailureRateClass3	0.0	Fulfilled
µC	0.0	FailureRateClass4	100.0	Fulfilled
C21	0.0	FailureRateClass3	100.0	Fulfilled
R12	0.0	FailureRateClass3	100.0	Fulfilled

Abb. 5.37: Anwendung der FRC-Methode für latente MPFs

Hinsichtlich der Anwendung der FRC-Methode auf latente MPFs im Kontext des Sicherheitsziels SG2 ist in Abb. 5.37 ersichtlich, dass der Watchdog WD den Zielwert im Gegensatz zu den anderen *Hardware-Bauteilen* nicht erreicht.

5.14.6 Import- und Exportschnittstellen

Um den Übergang von bewährten EDA-Tools in die Entwicklungsumgebung von PREEvision zu erleichtern, wurde exemplarisch eine Import-Schnittstelle für Schaltplandateien von EAGLE umgesetzt. Zudem wurde eine Export-Schnittstelle im Rahmen eines DIA für die in Kapitel 5.5 definierte XSD implementiert. Hierüber generierte Dateien im XML-Format können im Rahmen des Eclipse-Projekts eingelesen werden, um eine werkzeugunabhängige Durchführung der Sicherheitsevaluationen aufzuzeigen.

5.14.6.1 Import von Schaltplandateien aus EAGLE

Um die Vorzüge einer integrierten domänenspezifischen Entwicklungsumgebung nutzen zu können, wird ein Import von Schaltplandateien durch ein prototypisches PREEvision PlugIn ermöglicht. Da die Schaltplandatei im XML-Format vorliegt ist ein Einlesen dieser Informationen möglich. Eine Abbildung aus Sicht von EAGLE nach PREEvision erfolgt nach folgendem Schema:

Einholen der Bibliotheksinformationen zu den Typen von *Hardware-Bauteilen* aus der EAGLE-Bibliothek: Über die Auszeichner (Tags) <drawing, schematic, libraries, library, symbols> werden für jedes symbol die Attribute name und pin mit direction auf *InternalSchematicConnectorInOut/-In/-Out* abgebildet.

Einholen der im Schaltplan instanziierten *Hardware-Bauteile*: Über die Auszeichner <drawing, schematic, parts> werden für jedes part die Attribute name und deviceset eingeholt und auf *HWDevice* abgebildet sowie der Typ über *HWDeviceType* einbracht.

Für die Abbildung der Netze sind die relevanten Informationen in der XML-Struktur unter den Bezeichnern <drawing, schematic, sheets, sheet, nets> zu finden. Für jedes net wird das Attribut name eingeholt und darunter das entsprechende Netz über die Bezeichner <segment, pinref> mit deren Attributen part und pin aufgespannt. Die Abbildung erfolgt auf *InternalSchematicHWConnection*.

Die Herausforderung bei diesem Ansatz liegt in der Synchronisation der *Hardware-Bauteil*-Bibliotheken. Hier gilt es je nach Entwicklungsvorgehen die Entscheidung zu treffen, ob die Bibliothek in EAGLE oder PREEvision gepflegt wird. Entsprechend kann der Umgang mit den Bibliotheksinformationen umgesetzt werden, da eine Anreicherung mit Hardware-Fehlerinformationen nicht nur in PREEvision sondern auch innerhalb von EAGLE erfolgen könnte, vgl. Kapitel 5.12.

5.14.6.2 Export von XML-Dateien im Rahmen eines DIA

Das Konzept zum Export von Modellinformationen nach der festgelegten XML soll vor dem Hintergrund einer zügigen Verifikation im Kontext eines DIA über eine Metrik zur Verfügung gestellt werden. Hierfür konnten ein bereits vorhandenes Konzept und die dazugehörige Implementierung aus [Adler2012b] wiederverwendet und entsprechend adaptiert werden. Dieses basiert auf der Verwendung der Apache Velocity Engine, welche innerhalb von PREEvision zur Verfügung gestellt wird.

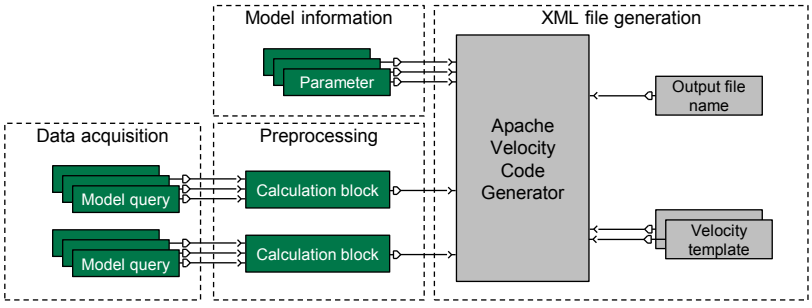


Abb. 5.38: Metrikkonzept für die Generierung einer XML, vgl. [Adler2012b, Fig.4]

Das Metrikkonzept ist in Abb. 5.38 dargestellt. Erforderlich sind allgemeine Informationen aus dem Modell, welche eine Rückverfolgbarkeit der XML erlauben. Zudem gilt es die relevanten Modellinformationen zu akquirieren und aufzubereiten und damit die vorgesehenen Platzhalter in der über die Velocity Templates abgebildeten XML-Struktur zu befüllen.

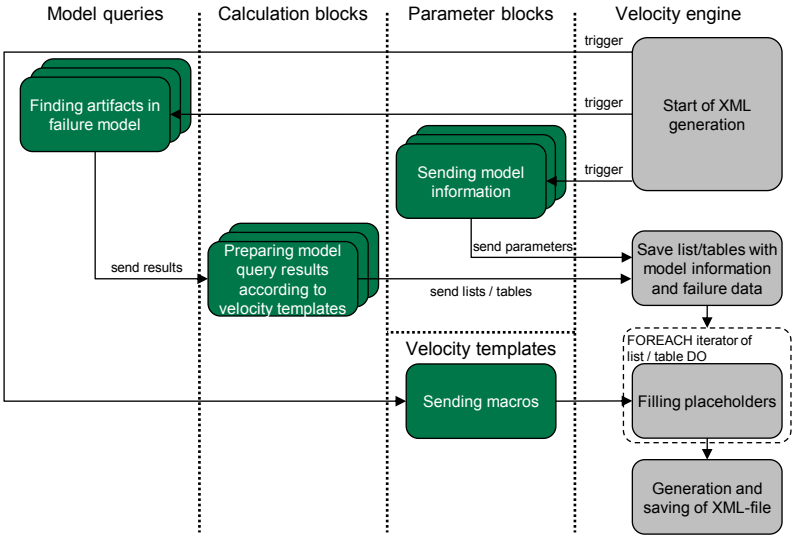


Abb. 5.39: Sequenz für die Generierung einer XML, vgl. [Adler2012b, Fig.5]

Die Abfolge zur Generierung ist in Abb. 5.39 ersichtlich. Im Rahmen der Hardware-Sicherheitsevaluationen und der vorgestellten Methodik sind die relevanten Inhalte der XML bereits über die in Kapitel 5.5 definierte XSD vorgegeben. Somit sind diese Informationen nun aus dem Modell zu entnehmen. Die Umsetzung erfolgte in einem Metrikdiagramm von PREEvision, wie in Abb. 5.40 dargestellt.

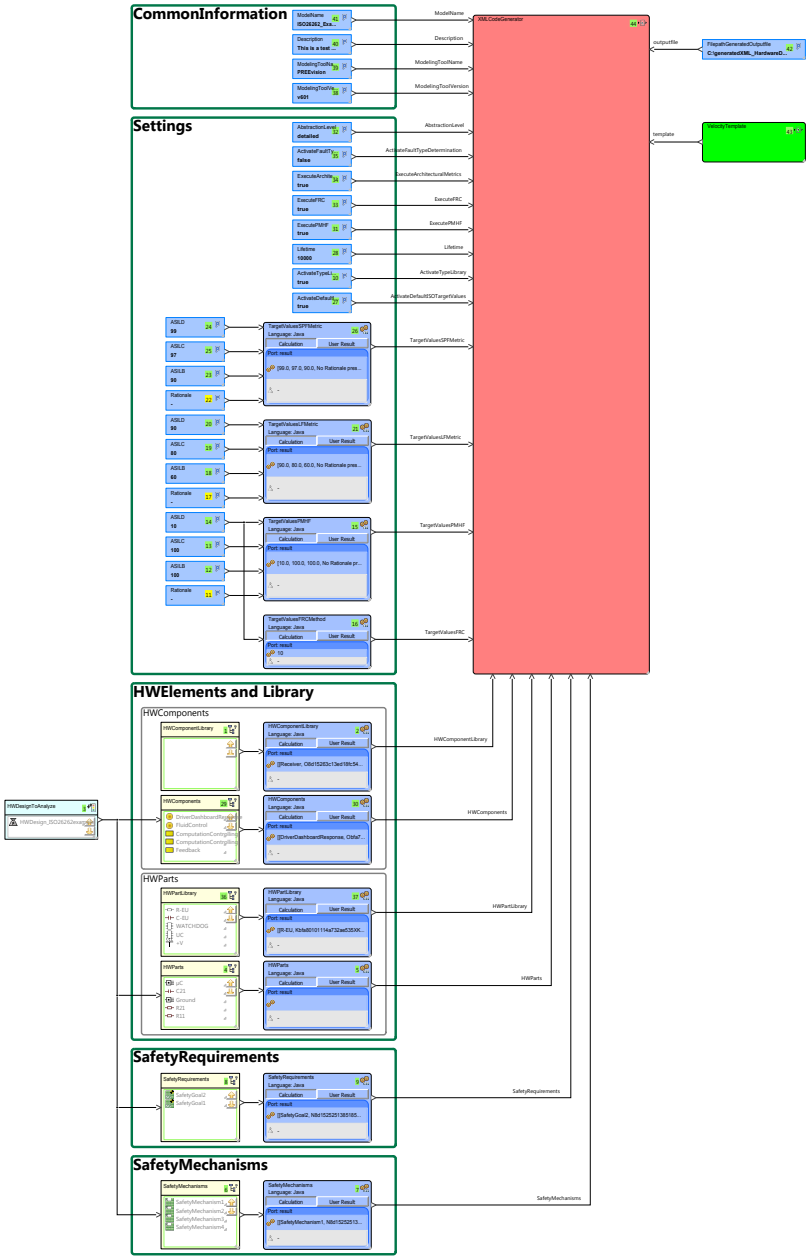


Abb. 5.40: Metrik zur XML-Generierung

Der rote Metrikblock stellt den Apache Code Generator für die Erzeugung der XML-Datei dar. Über den grünen Metrikblock wird die Grundstruktur der zugehörigen XSD aus Kapitel 5.5 im Rahmen eines Velocity Templates abgebildet. Der Speicherort und Dateiname für die XML wird über einen Parameterblock festgelegt. Sämtliche weitere Berechnungs- bzw. Parameterblöcke dienen dem Einfügen der entsprechenden Modellinformationen. Die umrandeten Bereiche beschreiben die Bereitstellung der entsprechenden Modellinformationen, unterteilt nach der Grundstruktur der XSD. Im Bereich der *Hardware-Elemente* ergibt sich durch die spezifizierbare Abstraktionsebene (architectural oder detailed) eine weitere Unterteilung. Hierbei werden die im Modell vorliegenden Informationen für beide Abstraktionsebenen bereitgestellt, die Metrik überführt automatisiert basierend auf der gewählten Abstraktionsebene die zugehörigen Informationen in die XML. Ein exemplarischer Auszug der generierten XML für die Abstraktionsebene des *detaillierten Hardwaredesigns* ist in Abb. 5.41 dargestellt.

generatedXML_HardwareDetailedDesign.xml

```
<Model>
    <CommonInformation>
        <ModelName>ISO26262_ExampleValveControl</ModelName>
        <Description>This is an example design.</Description>
        <ModelingTool>
            <Name>PREEvision</Name>
            <Version>v601</Version>
        </ModelingTool>
    </CommonInformation>
    <Settings>
        <AbstractionLevel>detailed</AbstractionLevel>
        <ActivateTypeLibrary>no</ActivateTypeLibrary>
        <ActivateFaultTypeDetermination>false</ActivateFaultTypeDetermination>
        <ExecuteArchitecturalMetrics>true</ExecuteArchitecturalMetrics>
        <ExecuteFRC>true</ExecuteFRC>
        <ExecutePMHF>true</ExecutePMHF>
        <Lifetime>10000.0</Lifetime>
        <TargetValues>
            <ActivateDefaultISOTargetValues>true</ActivateDefaultISOTargetValues>
            <SPFMetric>
            <LFMetric>
            <FRCMethod>
            <PMHF>
        </TargetValues>
    </Settings>
    <SafetyRequirements>
    <SafetyMechanisms>
    <HWElements>
        <HWElement>
            <Name>µC</Name>
            <ID>W4168cc0b0f77b60f26097d3c4d476e08XN8d152525138510498d545b700</ID>
            <HWElementTypeRefID>M8d15252513847fd61a1bbeXM8d1525251384fd61a1bbd00</HWElementTypeRefID>
            <FailureRate>100</FailureRate>
            <Value>No Value present</Value>
            <FailureModes>
                <FailureMode>
                    <Name>All1</Name>
                    <ID>M8d15252513847fd61a1bbexM8d1525251384c33fab7bca00</ID>
                    <Description></Description>
                    <FailureRateDistribution>50</FailureRateDistribution>
                    <SafetyMechanismRefID>N8d152525138562273fc35f1XN8d152525138562273fc35f000</SafetyMechanismRefID>
                    <Classifications>
                        <Classification>
                            <SafetyRequirement>
                                <Name>SafetyGoal1</Name>
                                <RefID>N8d15252513851859b481d7fXN8d15252513851859b481d7e00</RefID>
                            </SafetyRequirement>
```

Abb. 5.41: Auszug einer generierten XML-Datei für ein detailliertes Hardwaredesign

5.15 Prototypische Implementierung als Eclipse-Projekt

Wie im Rahmen der Übersicht der prototypischen Implementierungen vorgestellt, wurde neben der Implementierung in PREEvision auch ein wissenschaftliches Eclipse-Tool für modellbasierte Hardware-Sicherheitsevaluationen umgesetzt. Dies war aus zwei Gesichtspunkten relevant: Einerseits konnten die Konzepte für das DIA insbesondere im Kontext verteilter Entwicklungen nach Kapitel 5.5 verifiziert werden. Andererseits konnte hiermit das Konzept einer Fehlerbaumanalyse für das *Hardware-Architekturdesign* inklusive der grafischen Darstellung des Fehlerbaums umgesetzt werden. Das entwickelte Eclipse-Tool stellt somit eine werkzeugunabhängige Lösung zur Evaluation der funktionalen Sicherheit von *Hardwaredesigns* dar. Der wissenschaftliche Prototyp wurde als Implementierung in Eclipse Modeling Tools Version Juno Service Release 2 umgesetzt. In Abb. 5.42 ist ein Screenshot der grafischen Benutzeroberfläche ersichtlich, welche unter anderem durch Statusausgaben eine intuitive Bedienung ermöglicht und anhand eines vereinfachten Auszugs der Clauses aus Part 5 der ISO 26262 den Bezug zu den geforderten Sicherheitsevaluationen aufzeigt.

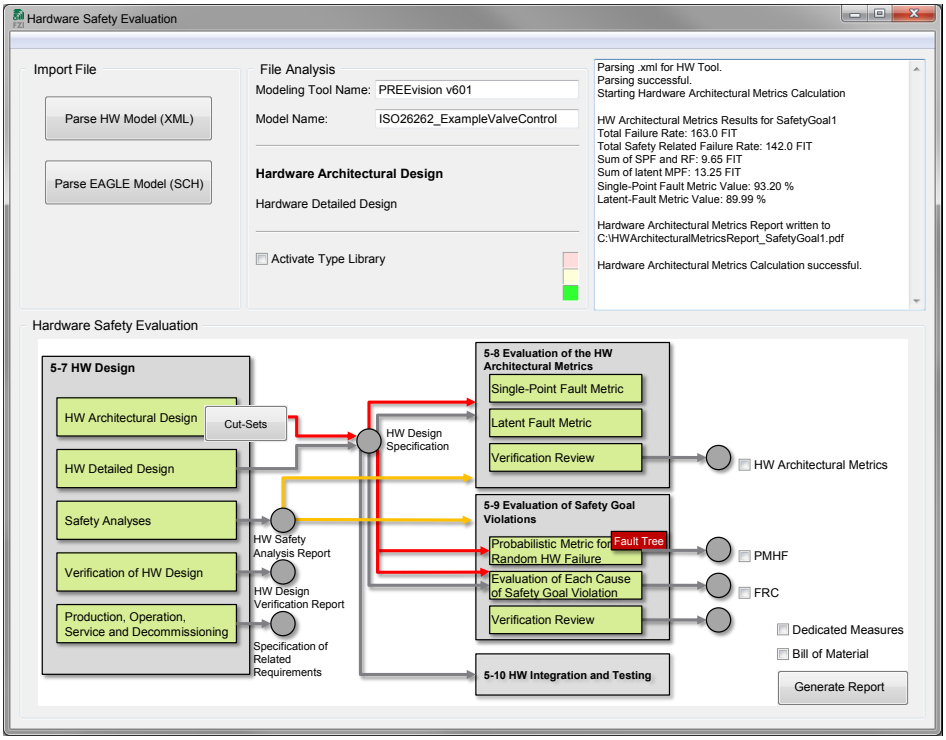


Abb. 5.42: Prototypische Implementierung für Hardware-Sicherheitsevaluationen

Im Folgenden wird ein Auszug der Funktionalitäten beschrieben:

Importschnittstellen: Die Importschnittstellen ermöglichen das Einlesen von Dateien im Rahmen der eigens definierten XSD sowie der Schaltplandatei aus EAGLE im XML-Format. Die eingelesene Datei wird zudem analysiert, um unter anderem sowohl Modellinformationen als auch Informationen zur Abstraktionsebene darzubieten. Durch die Ampel-Anzeige wird der Status auf Konformität der eingelesenen Datei abgebildet.

Hardware-Sicherheitsevaluationen: Parallel zu den Implementierungen in PREEvision ermöglicht das Eclipse-Tool die Durchführung sämtlicher nach ISO 26262 Part 5 geforderten Sicherheitsevaluationen. Dies umfasst die Hardware-Architekturmetriken, die FRC-Methode sowie eine qualitative und quantitative FTA zur Unterstützung der PMHF. Hierbei können die Sicherheitsevaluationen unabhängig vom genutzten Modellierungswerkzeug basierend auf der definierten Schnittstelle ausgeführt werden.

Fehlerbaumanalyse: Im Zuge der *Hardware-Architekturdesigns* hinterlegte Ausgangsabweichungen können zu Gesamtfehlerbäumen zusammengesetzt werden, auf denen eine qualitative und quantitative FTA ausgeführt wird. Diese werden grafisch visualisiert mit Hilfe der Open Source Lösung Graph Visualization Software (Graphviz)[66]. Um basierend auf der qualitativen FTA einschließlich einer Bestimmung von Minimalschnitten die Hardware-Architekturmetriken durchzuführen, wird eine automatische Klassifikation der Ausfallarten für *Hardware-Architekturdesigns* zur Verfügung gestellt.

Reportgenerierung: Im Kontext der Reportgenerierung kann aus einzelnen Reportmodulen für die Hardware-Sicherheitsevaluationen sowie zusätzlichen Informationen, wie BOM und *dedizierte Maßnahmen*, ausgewählt werden. Durch die Analyse der eingelesenen Datei und der dadurch erschlossenen Abstraktionsebene werden bereits entsprechende Reportmodule vorausgewählt. Im Rahmen der Reportgenerierung konnte als Framework die Apache Velocity Engine eingebracht werden, um entsprechende Nachweisdokumente im PDF-Dateiformat zu generieren. Die Berichte werden für sämtliche Sicherheitsziele basierend auf der Auswahl der einzelnen Reportmodule individuell erstellt und beinhalten auch die Fehlerdatentabellen. Je nach Auswahl der Hardware-Sicherheitsevaluationen können zudem Berichte für die qualitative und quantitative FTA mit den generierten Fehlerbaum-Darstellungen erstellt werden.

[66] http://www.graphviz.org/ (Zugriff am 15.10.2014)

Ein exemplarischer visualisierter Fehlerbaum, generiert aus dem Eclipse-Tool, ist in Abb. 5.43 dargestellt. Hierbei ist neben der Zuordnung der Ausfallarten zu entsprechenden *Hardware-Komponenten* über eine farbliche Hervorhebung auch die quantitative Analyse des Fehlerbaums vollzogen. Die Wahrscheinlichkeit für den Eintritt des Hauptereignisses ist somit bereits annotiert.

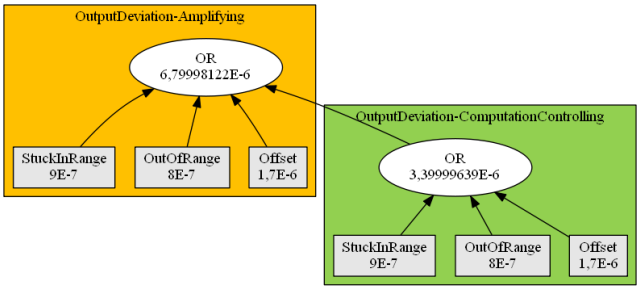


Abb. 5.43: Exemplarischer Fehlerbaum generiert mit Eclipse-Tool unter Nutzung von Graphviz

5.16 Berichte als Nachweisdokumente

Sowohl aus der Implementierung in PREEvision als auch aus dem wissenschaftlichen Eclipse-Tool lassen sich sämtliche Modellinformationen sowie Ergebnisse der Hardware-Sicherheitsevaluationen in automatisch generierten Reports dokumentieren. Diese sind im Portable Document Format (PDF) plattformunabhängig nutzbar. In den Abb. 5.44 bis 5.49 ist ein Auszug des generierten Reports aus PREEvision ersichtlich. Dieser beinhaltet unter anderem allgemeine Informationen zur durchgeführten Hardware-Sicherheitsevaluation, wie Version des Modells und Ausführungsdatum der Evaluation. Gleichzeitig wird das Diagramm des *Hardwaredesigns* als Abbildung eingebracht. Neben den Sicherheitszielen, die im Fokus der Evaluation standen, kann auch eine BOM verankert werden. Für jedes zu untersuchende Sicherheitsziel werden die zugehörigen Fehlerdatentabellen und nicht zuletzt die Ergebnisse der spezifischen Sicherheitsevaluationen in den Report eingebracht. Die Inhalte der jeweiligen Reports sind vorkonfiguriert, können aber beliebig zusammengestellt werden. Dies kann unter anderem für eine Argumentation im Sinne des Sicherheitsnachweises genutzt werden.

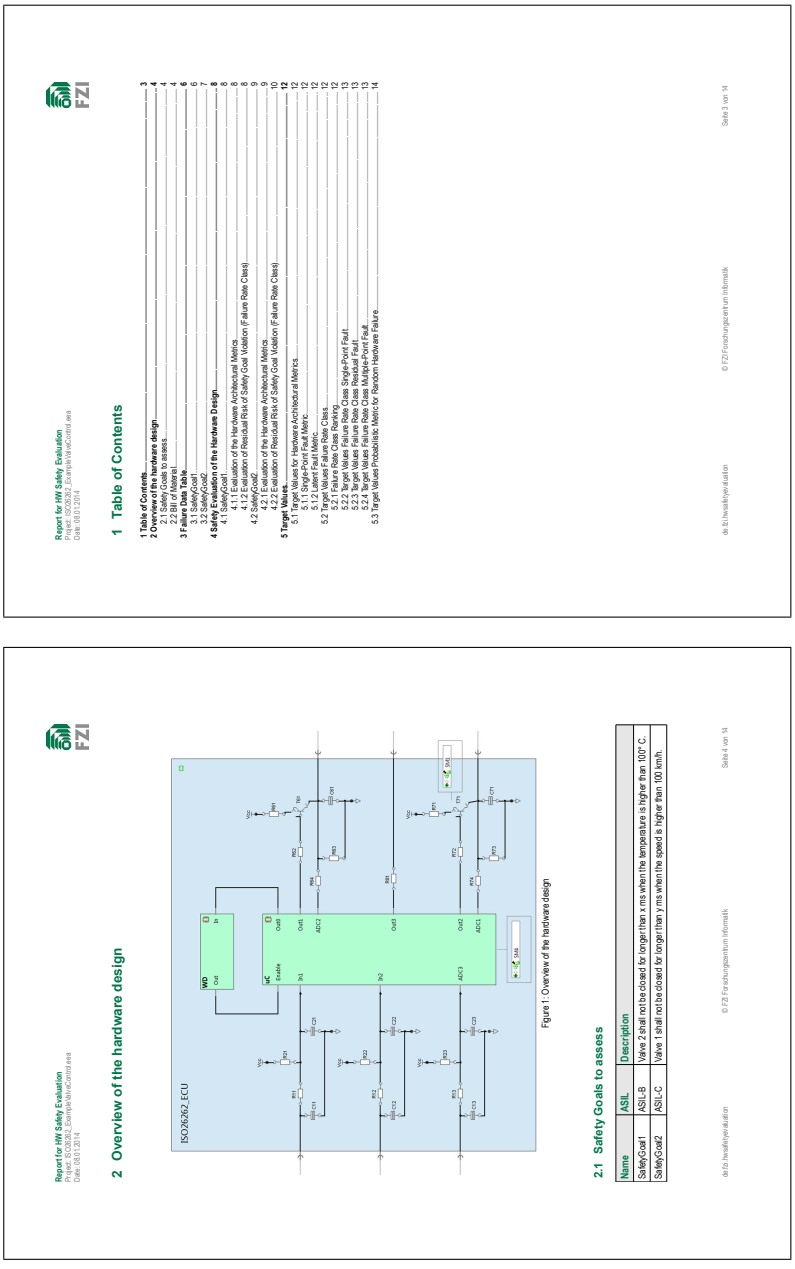
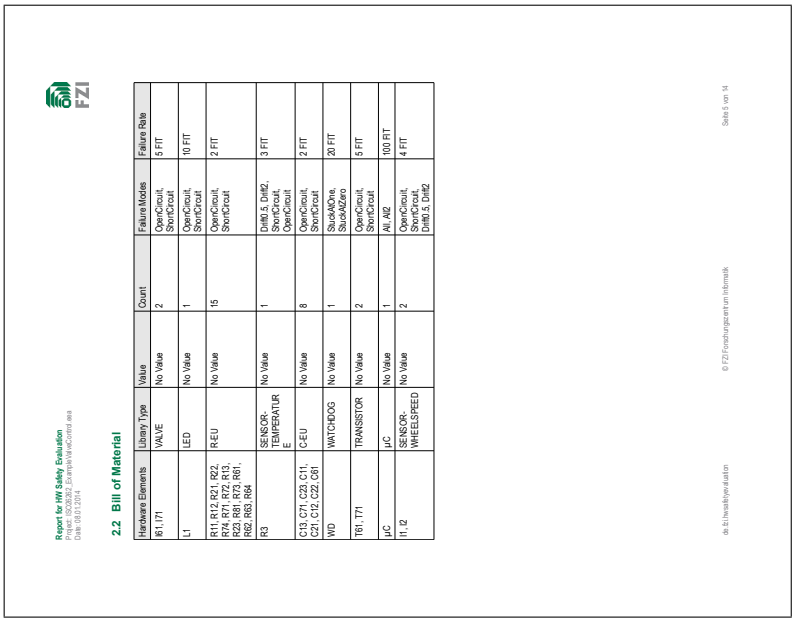


Abb. 5.44: Auszug des generierten Berichts Teil 1/6



Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControlArea
Date: 08.01.2014

FZI

2.2 Bill of Material

Hardware Elements	Library Type	Value	Count	Failure Modes	Failure Rate
M1, I171	VALVE	No Value	2	OpenCircuit, ShortCircuit	5 FIT
L1	LED	No Value	1	OpenCircuit, ShortCircuit	10 FIT
R11, R12, R21, R22, R74, R71, R72, R13, R23, R81, R73, R61, R62, R63, R64	R-EU	No Value	15	OpenCircuit, ShortCircuit	2 FIT
R3	SENSOR-TEMPERATURE	No Value	1	DriftO.5, DriftO2, ShortCircuit, OpenCircuit	3 FIT
C13, C71, C23, C11, C21, C12, C22, C81	C-EU	No Value	8	OpenCircuit, ShortCircuit	2 FIT
WD	WATCHDOG	No Value	1	StuckAOne, StuckAZero	20 FIT
T61, T71	TRANSISTOR	No Value	2	OpenCircuit, ShortCircuit	5 FIT
µC	µC	No Value	1	All, Al2	100 FIT
I1, I2	SENSOR-WHEELSPEED	No Value	2	OpenCircuit, ShortCircuit, DriftO.5, DriftO2	4 FIT

Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControlArea
Date: 08.01.2014

FZI

3 Failure Data Table

3.1 SafetyGoal1

Hardware Element Name	Failure Rate/ FIT	Safety-Related Violation Element	Failure Mode	Failure Rate Distribution	Direct Violation (SPFM?)	Safety Mechanism to Detect Violation	Diagnostic Coverage with Respect to Residual Faults	Specific Failure Rate with Respect to Violation / FIT	Violation in Combination with Another Failure Mode in Part	Safety Mechanism in Combination	Diagnostic Coverage with Respect to Latent Multi-Point Faults	Specific Failure Rate Latent Multi-Point Fault / FIT
C13	2.0	YES	OpenCircuit	20.0 %	X	none	0.0%	0.4				
C13	2.0	YES	ShortCircuit	80.0 %								
C23	2.0	NO	OpenCircuit	20.0 %								
C23	2.0	NO	ShortCircuit	80.0 %								
C71	2.0	YES	OpenCircuit	20.0 %					X	none	0.0 %	0.4
C71	2.0	YES	ShortCircuit	80.0 %								
R13	2.0	YES	OpenCircuit	90.0 %	X	none	0.0%	1.8				
R13	2.0	YES	ShortCircuit	10.0 %	X	none	0.0%	0.2				
R11	2.0	YES	OpenCircuit	90.0 %								
R11	2.0	YES	ShortCircuit	10.0 %					X	none	0.0 %	0.2
R73	2.0	NO	OpenCircuit	90.0 %								
R73	2.0	NO	ShortCircuit	10.0 %								
R63	2.0	YES	OpenCircuit	90.0 %								
R82	2.0	YES	ShortCircuit	10.0 %	X	none	0.0%	0.2				
R74	2.0	YES	OpenCircuit	90.0 %					X	none	0.0 %	1.8
R74	2.0	YES	ShortCircuit	10.0 %					X	none	0.0 %	0.2
R81	2.0	NO	OpenCircuit	90.0 %								
R81	2.0	NO	ShortCircuit	10.0 %								
µC	100.0	YES	All	50.0 %	X	SMI	90.0 %	0.0	X	SMI	100.0 %	0.0
µC	100.0	YES	Al2	50.0 %								
R3	3.0	YES	DriftO.5	30.0 %								
R3	3.0	YES	DriftO2	30.0 %	X	none	0.0%	0.9				
R3	3.0	YES	ShortCircuit	10.0 %								
R3	3.0	YES	OpenCircuit	30.0 %	X	none	0.0%	0.9				
L1	10.0	NO	OpenCircuit	90.0 %								
L1	10.0	NO	ShortCircuit	10.0 %								
T71	5.0	YES	OpenCircuit	50.0 %								
T71	5.0	YES	ShortCircuit	50.0 %	X	SMI	90.0 %	0.25	X	SMI	80.0 %	0.45
I71	5.0	NO	OpenCircuit	10.0 %								
I71	5.0	NO	ShortCircuit	30.0 %								
R72	2.0	YES	OpenCircuit	90.0 %								
R72	2.0	YES	ShortCircuit	10.0 %					X	none	0.0 %	0.2
WD	20.0	YES	StuckAOne	50.0 %					X	none	0.0 %	10.0
WD	20.0	YES	StuckAZero	50.0 %								

Abb. 5.45: Auszug des generierten Berichts Teil 2/6

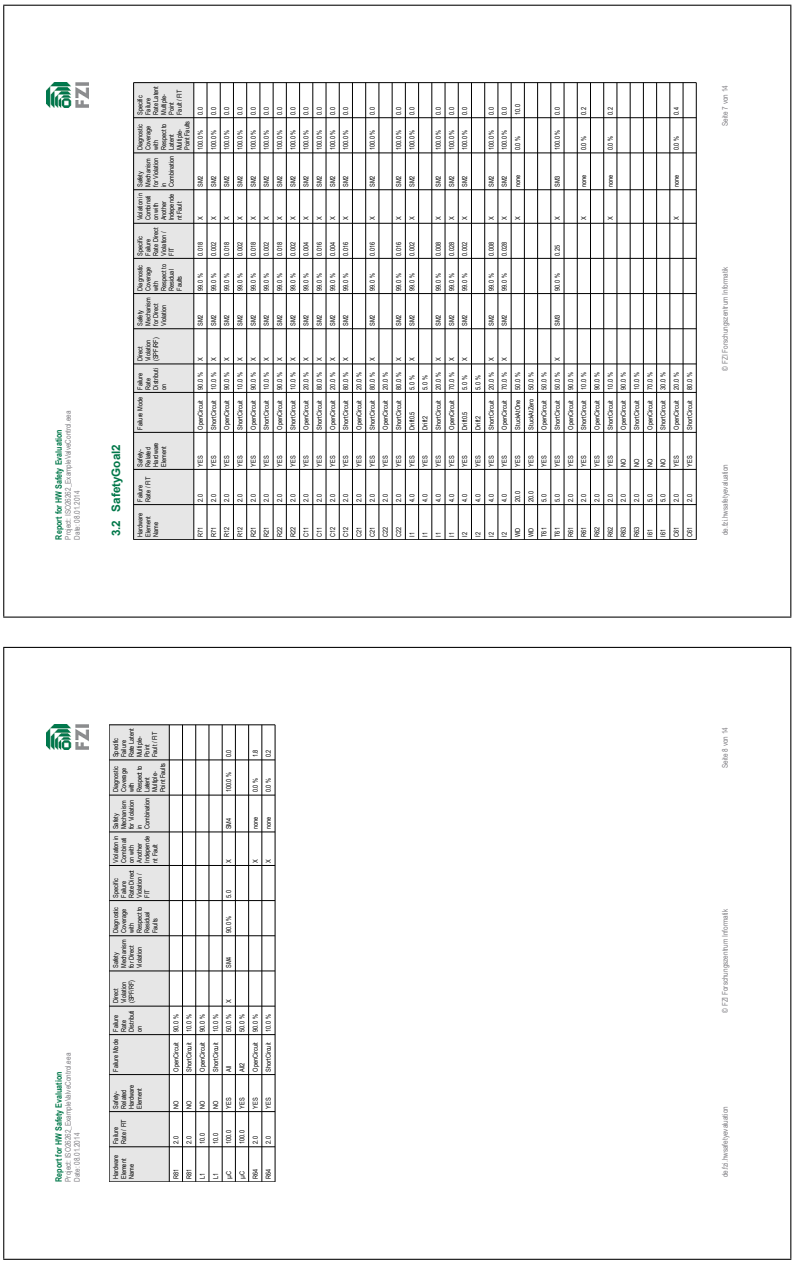


Abb. 5.46: Auszug des generierten Berichts Teil 3/6

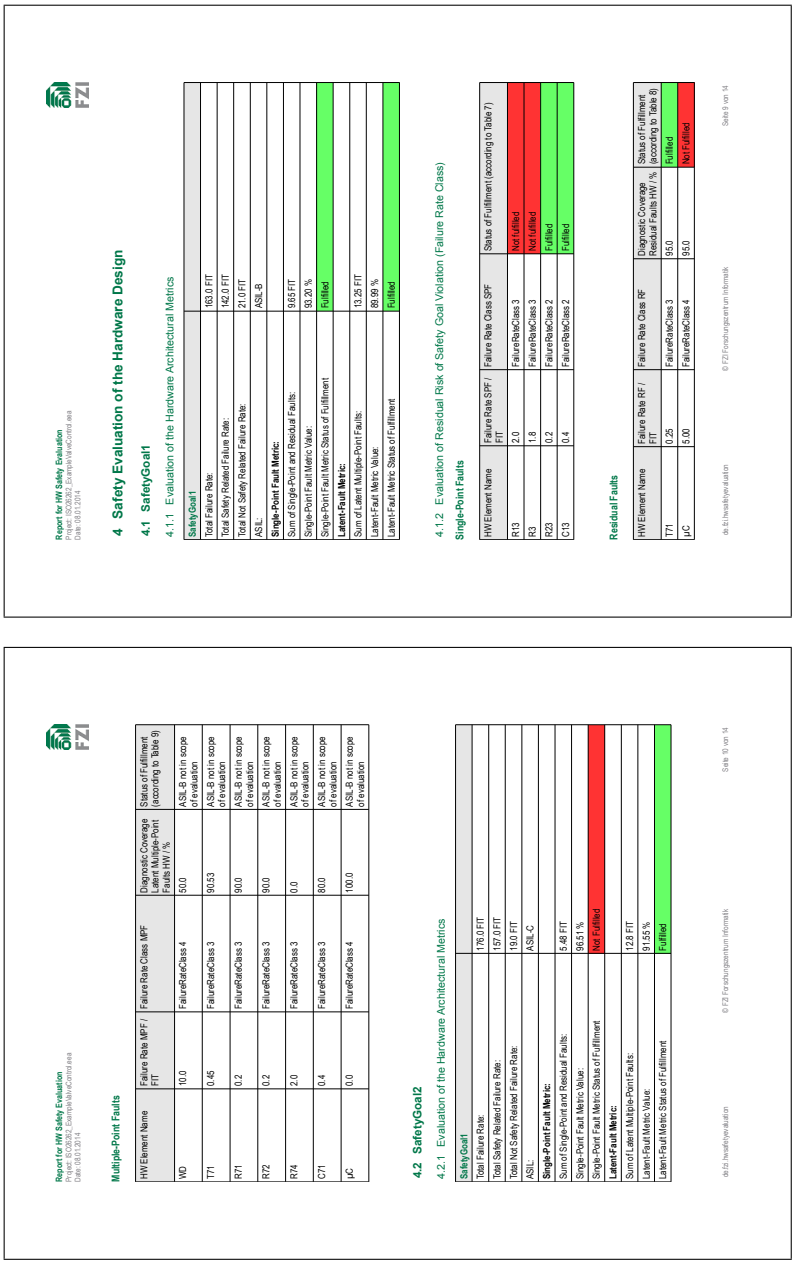


Abb. 5.47: Auszug des generierten Berichts Teil 4/6

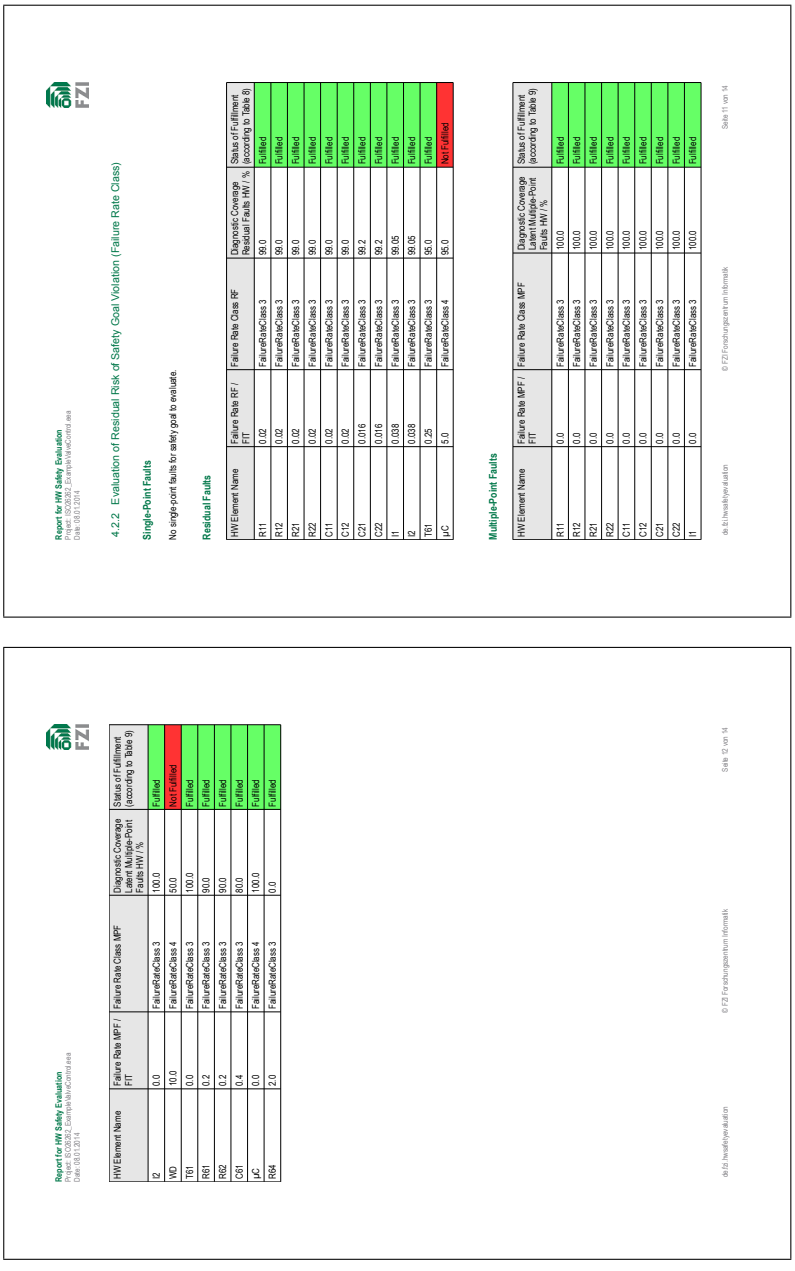


Abb. 5.48: Auszug des generierten Berichts Teil 5/6

Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControlArea
Date: 08.01.2014

FZI

5 Target Values

5.1 Target Values for Hardware Architectural Metrics

5.1.1 Single-Point Fault Metric

	ASIL-B	ASIL-C	ASIL-D
Single-point fault metric	>=90.0 %	>=97.0 %	>=99.0 %

5.1.2 Latent Fault Metric

	ASIL-B	ASIL-C	ASIL-D
Latent fault metric	>=60.0 %	>=80.0 %	>=90.0 %

5.2 Target Values Failure Rate Class

5.2.1 Failure Rate Class Ranking

Failure Rate Classes	Lower Bound	Upper Bound
FailureRateClass1		0.1
FailureRateClass2	0.1	1.0
FailureRateClass3	1.0	10.0
FailureRateClass4	10.0	100.0
FailureRateClass5	100.0	1000.0

5.2.2 Target Values Failure Rate Class Single-Point Fault

ASIL of Safety Goal	Failure Rate Class
ASIL-D	FailureRateClass1 + Dedicated Measures
ASIL-C	FailureRateClass2 + Dedicated Measures or FailureRateClass1
ASIL-B	FailureRateClass2 or FailureRateClass1

de.fzi.hwsafetyevaluation © FZI Forschungszentrum Informatik Seite 13 von 14

5.2.3 Target Values Failure Rate Class Residual Fault

ASIL of Safety Goal	Diagnostic Coverage >=99.99%	Diagnostic Coverage >=99.9%	Diagnostic Coverage >=99%	Diagnostic Coverage >=90%	Diagnostic Coverage <90%
ASIL-D	FailureRateClass5	FailureRateClass4	FailureRateClass3	FailureRateClass2	FailureRateClass1 + Dedicated Measures
ASIL-C	FailureRateClass6	FailureRateClass5	FailureRateClass4	FailureRateClass3	FailureRateClass2 + Dedicated Measures
ASIL-B	FailureRateClass6	FailureRateClass5	FailureRateClass4	FailureRateClass3	FailureRateClass2

5.2.4 Target Values Failure Rate Class Multiple-Point Fault

ASIL of Safety Goal	Diagnostic Coverage >=99%	Diagnostic Coverage >=90%	Diagnostic Coverage <90%
ASIL-D	FailureRateClass4	FailureRateClass3	FailureRateClass2
ASIL-C	FailureRateClass5	FailureRateClass4	FailureRateClass3

5.3 Target Values Probabilistic Metric for Random Hardware Failure

ASIL of Safety Goal	Target Values for Random Hardware Failures
ASIL-D	<10.0 FIT
ASIL-C	<100.0 FIT
ASIL-B	<1000 FIT

de.fzi.hwsafetyevaluation © FZI Forschungszentrum Informatik Seite 14 von 14

Abb. 5.49: Auszug des generierten Berichts Teil 6/6

5.17 Ergebnisse und Diskussion

Die modellbasierte Unterstützung für die Produktentwicklung auf Hardwareebene nach Part 5 der ISO 26262 stand im Mittelpunkt dieses Kapitels. Hierfür konnten aufbauend auf der erarbeiteten Prozessbeschreibung für den Standard die vorhandenen Lücken im Rahmen der Entwicklung funktional sicherer Hardware identifiziert werden. Insbesondere rückten das *Hardwaredesign* auf unterschiedlichen Abstraktionsebenen sowie die geforderten Sicherheitsevaluationen einschließlich ihrer Arbeitsprodukte in den Vordergrund. Um dies modellbasiert zu unterstützen, wurden drei große Teilbereiche identifiziert. Die Formalisierung von Fehlerinformationen, welche mit den strukturellen Beschreibungen von Hardware harmonisiert werden musste, die Erarbeitung einer Vorgehensweise für die einzelnen Sicherheitsevaluationen angepasst an die unterschiedlichen Abstraktionsebenen und nicht zuletzt die Integration der Formalisierungen zur Bereitstellung der entsprechenden Modellkonstrukte sowie die Umsetzung der Evaluationen in einer modellbasierten Entwicklungsumgebung.

Als wesentlichster Mehrwert zu anderen wissenschaftlichen Arbeiten und Werkzeugumgebungen im Umfeld der EEA-Entwicklung erfolgte eine durchgängige und integrierte Umsetzung der Methodiken für beide von der ISO 26262 vorgestellten Abstraktionsebenen. Somit werden Werkzeugwechsel und -brüche gezielt vermieden. Von zusätzlicher Bedeutung ist hierbei, dass nicht nur die Anwendung einzelner Analysemethodiken im Vordergrund stand, sondern sämtliche von der ISO 26262 geforderten Hardware-Sicherheitsevaluationen in einer integrierten Umgebung bereitgestellt werden. Für die Unterstützung des zu erbringenden Sicherheitsnachweises können sämtliche Ergebnisse der Evaluationen sowie zusätzliche Informationen in automatisiert generierten Berichten als Nachweisdokumente festgehalten werden.

Unter dem Gesichtspunkt von verteilten Entwicklungen konnte zudem eine erste Unterstützung hinsichtlich eines DIA zur Verfügung gestellt werden. Diese umfasst in einer serialisierten Form sämtliche relevante Informationen hinsichtlich der strukturellen Beschreibung und Anreicherung um Fehlerinformationen der einzelnen *Hardware-Elemente* sowie die Hinterlegung von Zielwerten. Die Verwendung des Austauschformates konnte durch die prototypische Implementierung als Eclipse-Projekt im Kontext einer werkzeugunabhängigen Umsetzung gezeigt werden. Weiterhin kann hierdurch auch die im Rahmen des Beitrags [92] angesprochene sukzessive Einführung einer modellbasierten Systementwicklung unterstützt werden.

Das im Rahmen der modellbasierten Entwicklung von *Hardwaredesigns* vorgestellte Metamodell zur Fehlerbeschreibung konnte gemeinsam mit einer Vorgehensweise für die Hardware-Sicherheitsevaluationen auf unterschiedlichen Abstraktionsebenen integriert und umgesetzt werden. Die Beiträge zur Hardware-Fehlerbeschreibung sind auch im Rahmen des Projektes SAFE in das Gesamt-Metamodell [D35b] eingebracht worden, eine unabhängige Metamodell-Implementierung ist in [121] beschrieben.

Vergleichbar zu den bisher häufig eingesetzten Tabellenkalkulationsprogrammen für die Durchführung der Hardware-Sicherheitsevaluationen bietet die hier dargestellte Umsetzung einen weiteren Mehrwert, dadurch dass ein in der Industrie verwendetes Frontend inklusive der dahinterliegenden umfassenden Datenbank genutzt wurde. Dies ermöglicht eine Unterstützung insbesondere in kollaborativen Entwicklungen und kann somit einen großen Mehrwert für die Serienentwicklung bieten. Zudem sind durch die verwendete Werkzeugumgebung keine langen Toolketten erforderlich. Somit ist die Aktualität der Modelldaten auch im Kontext einer begleitenden Nachweisführung gewahrt und die Grenzen des Untersuchungsgegenstandes können verlagert werden. Waren bisher nur einzelne spezifische Teilbereiche des *Hardwaredesigns* hinsichtlich funktionaler Sicherheit von Interesse für die Anwendung der Sicherheitsevaluationen, so kann dies durch den beschriebenen Ansatz auf das gesamte *Hardwaredesign* ohne zeitlichen Mehraufwand ausgedehnt werden.

Innerhalb individuell wählbarer Iterationsschleifen auf beiden Abstraktionsebenen können die Designs erstellt und die Sicherheitsevaluationen durchgeführt werden. Somit wird eine Unterstützung bei der Entwicklung, Evaluation und Optimierung funktional sicherer Hardware in der täglichen Arbeit geboten. Gleichzeitig können neue Designs durch die Wiederverwendung, beispielsweise der Bibliothek oder auch bereits evaluierter (Teil-)Designs aus Vorgängerserien, mit einer höheren Effizienz erstellt werden.

Die aktuelle Implementierung ermöglicht eine gleichzeitige Evaluation und Prüfung gegen die Zielwerte für mehrere Sicherheitsziele. Hierdurch kann nicht nur ein schneller Fortschritt gewährleistet werden, sondern auch jederzeit ein globaler Überblick über den aktuellen Entwicklungsstand des Fahrzeugsystems (Items) gewonnen werden, welcher gegebenenfalls zur Identifikation von Schwachstellen genutzt werden kann.

Die Diagnoseabdeckungsgrade $K_{DC,RF}(SM)$ und $K_{DC,MPF,L}(SM)$ werden für jeden Sicherheitsmechanismus durch den Benutzer eingetragen. Einen Nachweis sowie eine Argumentation der gewählten Werte muss aktuell durch den Hardwareentwickler erbracht werden. Eine Unterstützung und Beispiele zur Ableitung spezifischer Diagnoseabdeckungsgrade finden sich im informativen Annex D der ISO 26262 [ISO26262, 5-Tab.D.1]. Die aktuelle Implementierung ermöglicht als Verfeinerung die Zuordnung eines Sicherheitsmechanismus auf Ebene der Ausfallarten. Hierbei ist diskutierbar, ob unterschiedliche Sicherheitsmechanismen für die Abdeckung von Restfehlern und latenten Mehrfachfehlern eingebracht werden können.

ISO 26262 zeigt im informativen Teil [ISO26262, 10-AnnexB] im Rahmen der Untersuchung von Ausfällen beispielhaft die zwei in der Praxis häufig eingesetzten Methoden der FMEA und FTA. ISO 26262 betrachtet diese als sich ergänzende Methoden, welche auf unterschiedlichen Detaillierungsebenen angewandt werden können [ISO26262, 5-Tab.2.note]. Darauf aufbauend wird der Vorschlag einer Kombination dieser beiden Methoden unterbreitet, mit der Begründung einer Ausgewogenheit zwischen den beiden Vorgehensweisen. Dies kann nach [D322b, S.61] und [Adler2013b] auch als Design-

bestätigung der Hardware verwendet werden. In diesem Kontext existieren jedoch insbesondere firmenspezifische Vorgehensweisen, vgl. hierzu [97]. Der im Rahmen dieser Arbeit beschriebene Ansatz stellt sämtliche Hilfsmittel bereit, um den Forderungen der ISO 26262 hinsichtlich Hardware-Sicherheitsevaluationen zu genügen und lässt Freiheitsgrade für firmenspezifische Anpassungen.

In ISO 26262 wird von der „Confidence in the use of software tools“ im Rahmen der Entwicklung gesprochen [ISO26262, 8-11]. Die Einstufung in sogenannte „tool confidence level“ (TCL), welche sich aus den Klassen „tool impact“ (TI) und „tool error detection“ (TD) ergibt, ist für die in Kapitel 5.14 vorgestellten Prototypen zu diskutieren. Die Einstufung des TI erfolgt abhängig davon, ob die Möglichkeit besteht, dass eine Fehlfunktion des Softwaretools Fehler in Bezug auf ein sicherheitsbezogenes Fahrzeugsystem einbringt oder nicht erkennt [ISO26262, 8-11.4.5.2a]. Die Klassifizierung der TD beschreibt das Vertrauen in Maßnahmen, welche eine Fehlfunktion des Tools verhindern oder erkennen, ob diese vorliegt [ISO26262, 8-11.4.5.2b].

Da die Namensgebung wiederkehrend auf unterschiedlichen Abstraktionsebenen ist, siehe hierzu [ISO26262, 10-Fig.5], und eine Beziehung zwischen diesen hergestellt werden kann, ist immer zu berücksichtigen, in welchem Kontext bestimmte Begrifflichkeiten genutzt werden. Beispielsweise wird in ISO 26262 sowohl von der „Failure mode classification“ [ISO26262, 5-AnnexB] als auch von der „Fault classification“ [ISO26262, 5-AnnexC.1] gesprochen. Dies lässt sich durch die in [ISO26262, 10-4.3] beschriebene Beziehung zwischen Faults, Errors und Failures auf unterschiedlichen Ebenen ableiten.

Die innerhalb dieses Kapitels vorgestellten Methodiken sind bedingt durch die ISO 26262 im Automotive-Bereich verankert und angewandt. Hinsichtlich anderer Domänen sind in Bezug auf funktionale Sicherheit Adaptionen vorstellbar. Hierzu müssen zunächst die entsprechenden Sicherheitsevaluationen und Vorgehensweisen domänenspezifisch untersucht werden. Weiterhin sind Rahmenbedingungen für die Modellierung abzuleiten.

Betrachtet man die Entwicklung von Hardware, so reichen die eingesetzten Hardwarebausteine von FPGAs bis hin zu hochkomplexen ASICs. Bisher gibt es hierzu keine übergreifende modellbasierte Vorgehensweise hinsichtlich der Absicherung von zufälligen Ausfällen der Hardware. Die im Rahmen dieser Arbeit vorgestellten Methodiken beziehen sich auf den Bereich von der Systemebene bis hinunter zur Ebene von elektronischen Schaltplänen. Eine Anwendung der Modellierung und Analyse auf Ebene von komplexen *Hardware-Bauteilen* in modellbasierten Entwicklungsumgebungen ist rein prinzipiell denkbar und mit den dargestellten Methodiken sowie Implementierungen durchführbar, erfordert allerdings weiterreichende Adaptionen und Anpassungen für die spezifischen Gegebenheiten. Zudem sind etablierte und speziell für die Entwicklung von komplexen *Hardware-Bauteilen* angepasste Tools verfügbar. Diese sind unter den Gesichtspunkten der unterschiedlichen Technologien zwingend erforderlich und nicht wegzudenken. Somit wäre eine Schnittstelle zum Import der notwendigen Informationen für eine modell-

basierte Ausführung der Hardware-Sicherheitsevaluationen erforderlich. Hierfür könnte die im Rahmen dieser Arbeit erarbeitete XSD als Basis dienen.

Durch die modellbasierte Unterstützung der Entwicklung funktional sicherer *Hardwaredesigns* können im Zuge der implementierten Sicherheitsevaluationen geforderte Aktivitäten im Rahmen einer automatisierten Prozessausführung bereitgestellt werden. Somit kann wiederum der Bogen zu den im vorherigen Kapitel 4 beschriebenen Prozessen im Rahmen der ISO 26262 gespannt werden. Weiterhin werden die geforderten Arbeitsprodukte durch die automatisierte Erstellung von Nachweisdokumenten zur Verfügung gestellt.

6 Zusammenfassung und Ausblick

Die Veröffentlichung neuer Standards, deren Inhalte zum Zeitpunkt der Publikation als Stand der Wissenschaft und Technik angesehen werden und daher die Grundlage für die Entwicklung bilden, sorgen für Herausforderungen bei der Integration in bestehende Unternehmens- und Entwicklungsprozesse. Von einer frühzeitigen Einführung bis zum laufenden Betrieb kann ein enormer Personenaufwand für die Umsetzung der aus den Anforderungen abgeleiteten Prozesse und Vorgehensweisen anfallen. Die Umsetzungszeit schlägt sich daher für Industrieunternehmen aus wirtschaftlicher Sicht in hohen Kosten nieder, erzielt aber gleichzeitig steigende Entwicklungsqualität und eine Verbesserung der Methoden und Entwicklungswerkzeuge auf Seiten der Original Equipment Manufacturers (OEMs) sowie Zulieferer.

Bezogen auf die Automobildomäne im Bereich der Elektrik-/Elektronik-Architektur (EEA)-Entwicklung ist der Standard ISO 26262 „*Road vehicles - Functional safety*“ normativ und wird in der Industrie angewandt. Zudem ist eine Ausweitung des Anwendungsbereichs auf weitere Fahrzeugklassen in Vorbereitung. Jedoch wirft der Standard immer noch Diskussionspotentiale und Unklarheiten bei Prozessbeteiligten auf, sowohl bei Experten für funktionale Sicherheit als auch insbesondere bei Involvierten aus anderen Fachbereichen. Weiterhin steigt die Komplexität von automobilen Elektrik-/Elektronik-Systemen und OEMs sowie Zulieferer müssen dem Kostendruck und den gestiegenen Ansprüchen von Kunden bezüglich Qualität, Sicherheit und Komfort gerecht werden. Zugleich muss eine Unterstützung durch Entwurfswerkzeuge hinsichtlich funktionaler Sicherheit verbessert werden, um eine übergreifende und durchgängige Absicherung zu gewährleisten und die produktive Entwicklungsarbeit zu erleichtern.

6.1 Zusammenfassung

Um Prozesse einschließlich ihrer geforderten Aktivitäten und Arbeitsprodukte datenbankbasiert zu hinterlegen, sind modellbasierte Entwicklungsumgebungen für großformatige EEAs, in welchen domänenspezifisch und kollaborativ über den gesamten Produktlebenszyklus gearbeitet wird, prädestiniert. Daher konnte innerhalb dieser Arbeit ein Konzept für eine integrierte Prozessmodellierung und ein darauf aufsetzendes Prozessmanagement erarbeitet werden. Die Methodik der Prozessmodellierung basiert auf der Business Process Model and Notation (BPMN), welche durch die Standardisierung unter OMG und ISO/IEC an zunehmender Bedeutung gewonnen und

sich auch in fachfremden Bereichen durch ihre einfach verständliche grafische Notation etabliert hat.

Durch die Formalisierung in Form eines Metamodell-Vorschlags für EEA-Entwicklungsumgebungen wurde im Rahmen von Prozessbeschreibungen die Modellierung von Prozessen sowie von Organisationsstrukturen mit Zugehörigkeiten einzelner Rollen und Verantwortlichkeiten gegenüber Aktivitäten und Arbeitsprodukten berücksichtigt. Eine Verknüpfung der modellierten Prozesse mit den Elementen aus dem EEA-Datenmodell ermöglicht Transparenz und eindeutige Nachverfolgbarkeit. Basierend auf einem Konzept zum Prozessmanagement und einer prototypischen Integration in eine modellbasierte EEA-Entwicklungsumgebung wird aufbauend sowohl die Analyse von Prozessmodellen als auch eine unternehmensinterne und -übergreifende Prozessplanung unterstützt. Weiterhin ist neben der Berücksichtigung von zeitlichen Aspekten und der Anwendung der Methode des kritischen Pfades auch eine erste Möglichkeit des Monitorings für die Prozesse gegeben.

Als Anwendungsfall wurde der automobile Standard ISO 26262 aufgegriffen, welcher in seiner Gesamtheit hinsichtlich der geforderten Aktivitäten und Arbeitsprodukte abgebildet wurde. Angewandt auf Standards bietet die Modellierung der Prozesse den Vorteil einer möglicherweise frühzeitigen Identifikation von Inkonsistenzen in den Prozessketten oder deren Schnittstellen. Durch das erarbeitete Prozessmodell für ISO 26262 wird eine einfache Einarbeitung in die geforderten Prozesse über Themenfelder, wie beispielsweise Hardware und Software, hinweg für alle Prozessbeteiligten ermöglicht. Dies konnte verwendet werden, um den umfangreichsten Part 5 „*Product development at the hardware level*“ der ISO 26262 detaillierter zu betrachten.

Im Kontext der Produktentwicklung auf Hardwareebene sind nach diesem Part spezifische Aktivitäten hinsichtlich des *Hardwaredesigns* und der Durchführung von vorgeschriebenen Hardware-Sicherheitsevaluationen auf angemessener Abstraktionsebene erforderlich. Diese fokussieren sich auf die Absicherung gegenüber zufälligen Ausfällen der Hardware. Hierbei handelt es sich um die „*Evaluation of the hardware architectural metrics*“ zur Untersuchung der Robustheit des *Hardwaredesigns* sowie die ergänzenden Evaluationen „*Evaluation of Probabilistic Metric for random Hardware Failures (PMHF)*“ und „*Evaluation of each cause of safety goal violation*“ zur Untersuchung der Verletzung von Sicherheitszielen.

Durch eine Formalisierung im Zuge der Bereitstellung eines Metamodell-Vorschlags für Hardware-Fehlerinformationen konnte eine Erweiterung für etablierte Architekturbeschreibungssprachen zur Verfügung gestellt werden. Nach einer Untersuchung der Anwendbarkeit der Sicherheitsevaluationen bezogen auf unterschiedliche Designphasen konnte eine übergreifende Methodik erarbeitet werden, welche sich durch eine nahtlose und iterative Entwicklung funktional sicherer *Hardwaredesigns* auszeichnet. Die ausgearbeitete Vorgehensweise berücksichtigt neben der strukturellen Modellierung, basierend auf einem erweiterten Bibliothekskonzept zur Anreicherung um Fehlerinformationen,

den Bezug zu den relevanten Sicherheitszielen im Kontext des Fahrzeugsystems. Aus diesen können die zu erfüllenden Zielwerte, gegen welche im Zuge der Hardware-Sicherheitsevaluationen geprüft werden muss, abgeleitet werden.

Die Metamodell-Vorschläge konnten in ein Entwicklungswerkzeug für EEAs einfließen, um sowohl eine integrierte Sicht auf die *Hardwaredesigns* als auch die Anreicherung um Fehlerinformationen in einem etablierten Frontend zu bieten. Durch eingebrachte Implementierungen innerhalb der Werkzeugumgebung werden neben der Bereitstellung einer modellbasierten Durchführung sämtlicher geforderter Sicherheitsevaluationen sowie einer zusätzlich unterstützenden Fehlerbaumanalyse zudem auch Dokumentenvorlagen zur Verfügung gestellt. Diese dienen einer automatisierten Erstellung der von ISO 26262 aufgelisteten Arbeitsprodukte in Form von Nachweisdokumenten und unterstützen damit maßgebend den Sicherheitsnachweis für das sich in Entwicklung befindende Fahrzeugsystem. Bezogen auf die innerhalb der Entwicklungsumgebung bereitgestellte Prozessbeschreibung für ISO 26262, den Zusammenhang zwischen den geforderten Aktivitäten sowie Arbeitsprodukten und den eingebrachten Implementierungen konnte somit ein erster Schritt hinsichtlich einer automatisierten Ausführung von Teilprozessen im Zuge der Unterstützung von funktionalen Sicherheitsprozessen aufgezeigt werden.

Um weiterhin eine sukzessive Einführung der modellbasierten Entwicklung insbesondere im Kontext von verteilten Entwicklungen zu unterstützen, konnte neben dem Prozessmodell für ISO 26262 eine weitere Unterstützung für die Entwicklung von *Hardwaredesigns* im Zuge der Entwicklungsschnittstellen-Vereinbarung nach dem DIA bereitgestellt werden. Hierfür wurde eine Serialisierung aller relevanten Designinformationen im Kontext funktionaler Sicherheit bereitgestellt, welche unter anderem auch die Vergabe von Zielwerten im Austausch zwischen Auftraggeber und Auftragnehmer berücksichtigt. Dies konnte anhand einer weiteren prototypischen Implementierung im Rahmen eines Eclipse-Projektes verifiziert werden, welches zudem dieselben Funktionalitäten zur Ausführung der Sicherheitsevaluationen und Generierung von Nachweisdokumenten zur Verfügung stellt.

Innerhalb dieser Arbeit konnte ein Beitrag zur modellbasierten Entwicklung funktional sicherer Hardware geleistet werden. Dieser stellt eine integrierte Prozessmodellierung und ein darauf aufsetzendes Prozessmanagement unter Berücksichtigung von Organisationsstrukturen sowie deren Verantwortlichkeiten und Ressourcenzuweisungen zur Verfügung. Neben Formalisierungen zur Erweiterung von Architekturbeschreibungssprachen konnte ein Rahmenwerk für die iterative Durchführung von Design, Evaluation und Optimierung im Kontext unterschiedlicher Hardware-Entwicklungsphasen und sämtlicher geforderter Sicherheitsevaluationen nach ISO 26262 zur Verfügung gestellt werden. Die Verknüpfungen mit dem Prozessmodell für die ISO 26262 runden die Bearbeitung der geforderten Aktivitäten ab. Somit ist auch hinsichtlich einer möglichen Erweiterung des Anwendungsbereichs für den Standard ein mehr als solider Grundstein gelegt. Gleichermaßen kann durch die integrierte Prozessmodellierung und das Prozess-

management eine Harmonisierung der funktionalen Sicherheitsprozesse mit firmenspezifischen Entwicklungsprozessen erfolgen.

6.2 Ausblick

Durch die Publikation von ISO 26262 wurde ein domänenspezifischer Standard für funktionale Sicherheit als Stand der Technik für die Automobilindustrie bereitgestellt. Die Prozessmodellierung, -analyse und -planung konnte beispielhaft auf ISO 26262 angewandt werden. Eine Anwendung auf Standards hinsichtlich funktionaler Sicherheit aus anderen Domänen, wie beispielsweise Luftfahrt oder Medizintechnik, könnte in einem nächsten Schritt angegangen werden, um somit unter anderem inhaltliche Gemeinsamkeiten aufzuzeigen und bewährte Methodiken gegenseitig in die Prozesse einfließen zu lassen.

Im Rahmen von Bestrebungen einer zunehmenden Vernetzung hinsichtlich Car2Car bis hin zu Car2X rückt das Themenfeld Security immer mehr in den Fokus der Automobilindustrie. Aufbauend auf dem Prozessmodell für ISO 26262 könnte eine wissenschaftliche Untersuchung hierzu angegangen werden. Ein zusätzlicher Betrachtungspunkt ist die mögliche Abhängigkeit sowie gegenseitige Beeinflussung von einzelnen Aktivitäten in Bezug auf Safety und Security. Hieraus könnte unter anderem ein erster kombinierter „Safety-Security-Prozess“ hervorgehen, welcher die beiden Themenfelder aus Prozesssicht miteinander verknüpft bzw. koppelt. Dies würde eine engere Harmonisierung der Prozessmodelle mit dem EEA-Datenmodell erfordern, um eine weiterreichende Unterstützung bei den Entwicklungstätigkeiten zu bieten.

Die Produktentwicklung auf Hardwareebene ist einer der umfassendsten Teilbereiche der ISO 26262. Für die Entwicklung funktional sicherer Systeme wird zusätzlich das Pendant der Softwareseite nach Part 6 benötigt, um hierdurch die Produkte auf Systemebene zu vervollständigen. Hierzu gilt es eine modellbasierte Vorgehensweise für die Softwareentwicklung zu erarbeiten und eine Schnittstelle zur Hardwareentwicklung herzustellen. Eine Verknüpfung kann insbesondere durch das innerhalb der ISO 26262 beschriebene Hardware-Software Interface (HSI) bereitgestellt werden. Darauf aufbauend sind die Iterationen in der Produktentwicklung auf Hardware- und Softwareebene in Einklang zu bringen. Dies betrifft insbesondere Änderungen mit Auswirkungen auf Systemebene unter dem Gesichtspunkt der funktionalen Sicherheit. Somit können gewonnene Erkenntnisse und Ergebnisse aus den jeweiligen Fachbereichen im Rahmen eines funktional sicheren Gesamtsystems zusammengeführt werden.

In der Entwicklung von Fahrzeugsystemen ist unter anderem die Betrachtung von Produktvarianten von Bedeutung. Dies spannt eine orthogonale Ebene zu den betrachteten Architekturebenen auf. Im Kontext funktionaler Sicherheit ist bedeutend, an welchen Stellen in Hinsicht auf Struktur- und Fehlerbeschreibung für Hardware die Variationspunkte gesetzt werden können. Hierbei gilt es unter anderem zu unter-

suchen, inwiefern der Umgang mit verschiedenen Varianten im Rahmen der Hardware-Sicherheitsevaluationen erfolgen kann.

Die erwähnte mögliche Ausweitung des Anwendungsbereichs der ISO 26262 auf insbesondere weitere Fahrzeugklassen und Halbleiterbauelemente kann vorbereitend angegangen werden. Diese würde eine Untersuchung der bereits erarbeiteten Methodiken und Vorgehensweisen basierend auf der aktuellen Edition der ISO 26262 unter den jeweiligen Aspekten erfordern.

Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

Abkürzung	Beschreibung	Seite
ADL	Architecture Description Language	19
AADL	Architecture Analysis and Design Language	19
ARIS	Architecture of Integrated Information Systems	43
Artop	AUTOSAR Tool Platform	26
AUTOSAR	AUTomotive Open System ARchitecture	21
BOM	Bill of materials	101
BPM	Business Process Management	38
BPEL	Business Process Execution Language	51
BPMI	Business Process Management Initiative	44
BPMN	Business Process Model and Notation	40
BPMN-DI	BPMN Diagram Interchange	49
BPMS	Business Process Management System	39
BPR	Business Process Reengineering	38
CAD	Computer-Aided Design	32
CAN	Controller Area Network	11
CMMI	Capability Maturity Model Integration	16
CPM	Critical Path Method	126
EAST-ADL	Electronics Architecture and Software Technology - Architecture Description Language	21
EATOP	EAST-ADL Tool Platform	26
ECAD	Electrical CAD	230
EDA	Electronic Design Automation	224
EDIF	Electronic Design Interchange Format	230
EE	Elektrik-/Elektronik	11
EEA	Elektrik-/Elektronik-Architektur	12
EEA-ADL	Electric Electronic Architecture - Analysis Design Language	30
EPF	Eclipse Process Framework	81
EMF	Eclipse Modeling Framework	27
EPK	Ereignisgesteuerte Prozesskette	40
eEPK	erweiterte EPK	44
EPRD	Electronic Parts Reliability Data	77
FEV	Fully Electric Vehicle	25
FMD	Failure Mode / Mechanism Distribution	77
FMECA	Failure Mode, Effects and Criticality Analysis	68
FMEDA	Failure Mode, Effects and Diagnostic Analysis	69
GSN	Goal Structuring Notation	64
HARA	Hazard Analysis and Risk Assessment	34
HDL	Hardware Description Language	99
HiP-HOPS	Hierarchically Performed Hazard Origin and Propagation Studies	97
HTML	Hypertext Markup Language	84
IP	Intellectual Property	33
JWT	Java Workflow Tooling	82
LIN	Local Interconnect Network	11
MARTE	Modeling and Analysis of Real-Time and Embedded Systems	24
MBSE	Model-Based Systems Engineering	93
MOF	Meta Object Facility	45
MOST	Media Oriented Systems Transport	11

MTBF	Mean Time Between Failures	72
MTTF	Mean Time To Failure	21
MTTFF	Mean Time To First Failure	72
MTTR	Mean Time To Repair	72
NPRD	Nonelectronic Parts Reliability Data	77
OEM	Original Equipment Manufacturer	12
OSATE	Open Source AADL Tool Environment	20
PCB	printed circuit board	76
PDF	Portable Document Format	247
PNML	Petri Net Markup Language	43
RIAC	Reliability Information Analysis Center	77
RM	Requirements Management	93
RTP	Reference Technology Platform	25
SFF	Safe Failure Fraction	69
SMD	surface-mounted device	205
SoC	System-on-Chip	98
SOP	Start Of Production	13
SPEM	Software & Systems Process Engineering Meta-Model	81
SPICE	Software Process Improvement and Capability Determination	16
SysML	System Modeling Language	19
uC	Microcontroller	63
WS-BPEL	Web Services Business Process Execution Language	45
XMI	XML Metadata Interchange	49
XSD	XML Schema Definition	49

ISO 26262 spezifisches Abkürzungsverzeichnis

Abkürzung	Beschreibung	Seite
ACC	Adaptive Cruise Control	54
ASIC	Application-Specific Integrated Circuit	63
ASIL	Automotive SIL	58
DC	Diagnostic Coverage	75
DIA	Development Interface Agreement	61
DPF	Dual-Point-Fault	75
ECU	Electronic Control Unit	11
ETA	Event Tree Analysis	65
FRC	Failure Rate Class	96
FSC	Functional Safety Concept	58
FSR	Functional Safety Requirement	58
FIT	Failures In Time	71
FMEA	Failure Mode and Effects Analysis	21
FPGA	Field Programmable Gate Array	63
FTA	Fault Tree Analysis	34
HAZOP	HAZard and OPerability analysis	65
HSI	Hardware-Software Interface	58
HW	Hardware	92
LFM	Latent-Fault Metric	170
MPF	Multiple-Point-Fault	69
PMHF	Probabilistic Metric for random Hardware Failures	96

Abkürzungsverzeichnis im Kontext von Standards

Projektbezeichnungen

Projekte mit eigenem Beitrag im Kontext dieser Arbeit:

Abkürzung	Beschreibung	Seite
FTA/EE	„Fehlermodellierung und -analyse für E/E-Architekturmodelle“	
SAFE	„Safe Automotive soFtware architEcture“	26

Projekte ohne eigenen Beitrag:

Abkürzung	Beschreibung	Seite
ADAMS	„Action for the Dissemination and Adoption of the MARTE and related Standards for component based middleware“	24
ATESST	„Advancing Traffic Efficiency and Safety through Software Technology“	23
ATESST2	„Advancing Traffic Efficiency and Safety through Software Technology 2“	24
CESAR	„Cost-efficient methods and processes for safety relevant embedded systems“	24
EASIS	„Electronic Architecture and System Engineering for Integrated Safety Systems“	27
EAST-EEA	„Electronics Architecture and Software Technology - Embedded Electronic Architecture“	21
EDONA	„Environnements de Développement Ouverts aux Normes de l’Automobile“	25
JasPar	„Japan automotive software Platform and architecture“	27
MAENAD	„Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles“	25
MBAT	„Combined Model-based Analysis and Testing of Embedded Systems“	26
TIMMO	„Timing Model“	25
TIMMO-2-USE	„Timing Model - TOols, algorithms, languages, methodology, USE cases“	26

Liste der Formelzeichen

$f(t)$	Dichtefunktion
$F(t)$	Ausfallwahrscheinlichkeit
$K_{DC,RF}\ (SM)$	Diagnoseabdeckungsgrad eines SM in Bezug auf Restfehler
$K_{DC,MPF,L}\ (SM)$	Diagnoseabdeckungsgrad eines SM in Bezug auf latente Mehrfachfehler
$K_{DC,RF}\ (HW)$	Diagnoseabdeckungsgrad im Kontext der FRC bezogen auf Restfehler
$K_{DC,MPF,L}\ (HW)$	Diagnoseabdeckungsgrad im Kontext der FRC bezogen auf latente Mehrfachfehler
$\lambda(t)$	Ausfallrate
λ_{HW}	Gesamtausfallrate eines *Hardware-Elements*
λ_{SF}	Ausfallrate der sicheren Fehler eines *Hardware-Elements*
λ_{SPF}	Ausfallrate der Einzelfehler eines *Hardware-Elements*
λ_{RF}	Ausfallrate der Restfehler eines *Hardware-Elements*
λ_{MPF_L}	Ausfallrate der latenten Mehrfachfehler eines *Hardware-Elements*
$\lambda_{SPF/RF}\ (FM)$	Ausfallrate eines Einzelfehlers und Restfehlers einer spezifischen Ausfallart
$\lambda_{MPF_L}\ (FM)$	Ausfallrate eines latenten Mehrfachfehlers einer spezifischen Ausfallart
$R(t)$	Zuverlässigkeit

Quellennachweise

Veröffentlichungen

[1] M. Bernard, C. Buckl, V. Döricht, M. Fehling, L. Fiege, H. von Grolman, N. Ivandic, C. Janello, C. Klein, K.-J. Kuhn, C. Patzlaff, B. C. Riedl, B. Schätz, and C. Stanek, *Mehr Software (im) Wagen: Informations- und Kommunikationstechnik (IKT) als Motor der Elektromobilität der Zukunft*. van Acken Druck GmbH, 2011.

[2] Kraftfahrt-Bundesamt (KBA) (Ed.), *Jahresbericht 2012*. Flensburg: Druckzentrum KBA, 2012.

[3] Verband der Automobilindustrie e.V. (VDA) (Ed.), *Jahresbericht 2013*. Meckenheim: DCM Druck Center Meckenheim.

[4] Verband der Automobilindustrie e.V. (VDA) (Ed.), *Jahresbericht 2014*. Meckenheim: DCM Druck Center Meckenheim.

[5] R. Belschner, J. Freess, and M. Mroßko, "Gesamtheitlicher Entwicklungsansatz für Entwurf, Dokumentation und Bewertung von E/E-Architekturen," in *12. Internationaler Kongress Elektronik im Kraftfahrzeug*, ser. VDI-Berichte Nr. 1907. Düsseldorf: VDI, 2005, pp. 511–521.

[6] M. Broy, G. Reichart, and L. Rothhardt, "Architekturen softwarebasierter Funktionen im Fahrzeug: von den Anforderungen zur Umsetzung," *Informatik-Spektrum*, vol. 34, no. 1, pp. 42–59, Feb. 2011.

[7] G. Reichart and M. Haneberg, "Key Drivers for a Future System Architecture in Vehicles," in *Convergence International Congress & Exposition On Transportation Electronics*, Oct. 2004.

[8] W. W. Royce, "Managing the development of large software systems," in *Proceedings of the 9th International Conference on Software Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1987, pp. 328–338, reprinted from Proceedings, IEEE WESCON, August 1970, pages 1-9. Copyright 1970 by The Institute of Electrical and Electronics Engineers, Inc. Originally published by TRW.

[9] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[10] B. Boehm and P. Bose, "A Collaborative Spiral Software Process Model Based on Theory W," in *Proceedings of the Third International Conference on the Software Process: Applying the Software Process*. IEEE, Oct. 1994, pp. 59–68.

[11] B. W. Boehm, "Guidelines for Verifying and Validating Software Requirements and Design Specifications," in *Proceedings of the European Conference on Applied Information Technology of the International Federation for Information Processing (Euro IFIP)*. Amsterdam: North-Holland Publishing Company, Sept. 1979, pp. 711–719.

[12] B. W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," *IEEE Software*, vol. 1, no. 1, pp. 75–88, Jan. 1984.

[13] D. E. Barber, "An Overview of the Systems Engineering Capability Model EIA/IS 731," in *Proceedings of the 17th AIAA/IEEE/SAE Digital Avionics Systems Conference (DASC)*, vol. 1, 1998, pp. B34 1–7.

[14] AADL community, "AADL tools," Sep. 2014, [Zugriff am 15.10.2014]. [Online]. Available: https://wiki.sei.cmu.edu/aadl/index.php/AADL_tools

[15] P. H. Feiler, D. P. Gluch, and J. J. Hudak, *CMU/SEI-2006-TN-011: The Architecture Analysis & Design Language (AADL): An Introduction*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Feb. 2006.

[16] P. H. Feiler, B. A. Lewis, and S. Vestal, "The SAE Architecture Analysis & Design Language (AADL): A Standard for Engineering Performance Critical Systems," in *Proceedings of the IEEE Conference on Computer Aided Control Systems Design*. IEEE, Oct. 2006, pp. 1206–1211.

[17] P. Feiler and A. Rugina, *CMU/SEI-2007-TN-043: Dependability Modeling with the Architecture Analysis & Design Language (AADL)*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Jul. 2007.

[18] R. Behjati, T. Yue, S. Nejati, L. Briand, and B. Selic, "Extending SysML with AADL Concepts for Comprehensive System Architecture Modeling," in *Proceedings of the 7th European Conference Modelling Foundations and Applications (ECMFA)*, ser. Lecture Notes in Computer Science, vol. 6698. Berlin, Heidelberg: Springer, Jun. 2011, pp. 236–252.

[19] B. Viaud and P. Labrèche, "Citrus: Model-Based Avionics Development with Zest!" in *SAE AeroTech Congress & Exhibition*, Sep. 2013.

[20] *ITEA 2 Magazine no. 1*. Eindhoven: ITEA Office, Jun. 2008.

[21] *EAST-ADL Domain Model Specification Version V2.1.12*. EAST-ADL Association, 2013.

[22] EAST-ADL Association, "About EAST-ADL," [Zugriff am 15.10.2014]. [Online]. Available: http://www.east-adl.info/Specification.html

[23] *EAST-ADL Introduction: Methodology*. MAENAD project, 2013.

[24] *Deliverable D4.1.1 (public): EAST-ADL Domain Model Specification - Version number 2.1 RC3 (Release Candidate)*. ATESST2 project, 2010.

[25] M. Hobelsberger, S. Kuntz, and J. Mottok, "Architekturmodellierung: Vergleich von EAST ADL und SAE AADL," in *HANSER automotive 7-8*, 2007, pp. 43–47.

[26] Y. Dajsuren, M. van den Brand, A. Serebrenik, and R. Huisman, "Automotive ADLs: A Study on Enforcing Consistency Through Multiple Architectural Levels," in *Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures (QoSA)*. New York, NY: ACM, 2012, pp. 71–80.

[27] A. Johnsen and K. Lundqvist, "Developing Dependable Software-Intensive Systems: AADL vs. EAST-ADL," in *Proceedings of the 16th Ada-Europe International Conference on Reliable Software Technologies*, ser. Lecture Notes in Computer Science, vol. 6652. Berlin, Heidelberg: Springer, 2011, pp. 103–117.

[28] *EAST-ADL Introduction: EAST-ADL Overview*. MAENAD project, 2013.

[29] *EAST-ADL Introduction: Support for ISO26262*. MAENAD project, 2012.

[30] *ITEA 2 Magazine no. 5*. Eindhoven: ITEA Office, Dec. 2009.

[31] *ITEA 2 Magazine no. 14*. Eindhoven: ITEA Office, Dez. 2012.

[32] S. Voget, "The SAFE Technology Platform - An Open Source Tool Platform for Safety Modeling and Analysis," in *SafeTRANS News 2/2013*, Jul. 2013, pp. 14–15.

[33] *ITEA 2 Magazine no. 15*. Eindhoven: ITEA Office, Apr. 2013.

[34] *ITEA 2 Magazine no. 17*. Eindhoven: ITEA Office, Jan. 2014.

[35] D. Umanovskis and S. Voget, *Deliverable D5.3.1 v3.0 (final, public): EATOP: An EAST-ADL Tool Platform for Eclipse*. MAENAD project, 2014.

[36] Y. Ma, "EATOP architecture," Apr. 2014, [Zugriff am 15.10.2014]. [Online]. Available: http://wiki.eclipse.org/EATOP/architecture

[37] M. Kühl, C. Reichmann, I. Protel, and K. D. Müller-Glaser, "From Object-Oriented Modeling to Code Generation for Rapid Prototyping of Embedded Electronic Systems," in *Proceedings of the*

13th IEEE International Workshop on Rapid System Prototyping (RSP). IEEE, 2002, pp. 108–114.

[38] C. Reichmann, M. Kühl, P. Graf, and K. D. Müller-Glaser, "GeneralStore - A CASE-Tool Integration Platform Enabling Model Level Coupling of Heterogeneous Designs for Embedded Electronic Systems," in *Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS)*. IEEE, May 2004, pp. 225–232.

[39] K. D. Müller-Glaser, C. Reichmann, P. Graf, M. Kühl, and K. Ritter, "Heterogeneous Modeling for Automotive Electronic Control Units using a CASE-Tool Integration Platform," in *IEEE International Symposium on Computer Aided Control Systems Design*. IEEE, Sep. 2004, pp. 83–88.

[40] J. Matheis, D. Gebauer, M. Kühl, C. Reichmann, and K. D. Müller-Glaser, "Vorstellung einer Methodik zur E/E-Architektur-Modellierung und -Bewertung in der frühen Konzeptphase," in *Moderne Elektronik im Kraftfahrzeug*, ser. Haus der Technik, vol. 67. Expert Verlag, 2006, pp. 66–80.

[41] J. Matheis, D. Gebauer, C. Reichmann, and K. D. Müller-Glaser, "Ganzheitliche abstraktionsebenenübergreifende Beschreibung konsistenter Elektrik/Elektronik-Architekturen," in *Systems Engineering Infrastructure Conference (SEISCONF)*, 2008.

[42] D. Gebauer, J. Matheis, C. Reichmann, and K. D. Müller-Glaser, "Ebenenübergreifende, variantengerechte Beschreibung von Elektrik/Elektronik-Architekturen," in *Diagnose in mechatronischen Fahrzeugsystemen*. Expert Verlag, 2008, pp. 142–153.

[43] D. Gebauer, J. Matheis, M. Kühl, and K. D. Müller-Glaser, "Integrierter, graphisch notierter Ansatz zur Bewertung von Elektrik/Elektronik-Architekturen im Fahrzeug," in *Moderne Elektronik im Kraftfahrzeug IV*, ser. Haus der Technik, vol. 105. Expert Verlag, 2009, pp. 49–61.

[44] R. Zhang and A. Krishnan, "Using Delta Model for Collaborative Work of Industrial Large-Scaled E/E Architecture Models," in *Model Driven Engineering Languages and Systems (MODELS)*, ser. Lecture Notes in Computer Science, vol. 6981. Berlin, Heidelberg: Springer, 2011, pp. 714–728.

[45] S. Keul, E. Metzker, and D. Lederer, "Durchgängige Realisierung von Steuergerätesoftware nach ISO 26262," in *ATZelektronik*. Springer Automotive Media, Oct. 2013, vol. 8, no. 5, pp. 330–335.

[46] Deutsches Institut für Normung e. V., "Normungsorganisationen," [Zugriff am 15.10.2014]. [Online]. Available: http://www.din.de

[47] International Organization for Standardization (Ed.), *ISO Focus+*, Jul.-Aug. 2011, vol. 2, no. 7.

[48] Verband der Automobilindustrie e. V. (VDA) (Ed.), *Auto & Normung: NA-Automobil Jahresbericht 2014*.

[49] A. Gillhuber, "Interview Prof. Dr. Thomas Klindt: Normenkonformität bedeutet nicht Rechtskonformität!" Jul. 2013, [Zugriff am 15.10.2014]. [Online]. Available: http://www.elektroniknet.de/automation/sonstiges/artikel/99162/

[50] M. Hammer, "Reengineering Work: Don't Automate, Obliterate," *Harvard Business Review*, pp. 104–112, Jul.-Aug. 1990.

[51] B. Curtis, M. I. Kellner, and J. Over, "Process modeling," *Communications of the ACM*, vol. 35, no. 9, pp. 75–90, Sep. 1992.

[52] B.-J. Hommes and V. van Reijswoud, "Assessing the Quality of Business Process Modelling Techniques," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. IEEE, Jan. 2000.

[53] E. D. Falkenberg, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F. J. M. Van Assche, A. A. Verrijn-Stuart, and K. Voss (Eds.), *The FRISCO Report: A framework of information system concepts*, 1998.

[54] G. Keller, M. Nüttgens, and A.-W. Scheer, "Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)"," Jan. 1992.

[55] S. A. White, "Process Modeling Notations and Workflow Patterns," in *BPTrends*, Mar. 2004.

[56] A. Schmietendorf, "Assessment of Business Process Modeling Tools under Consideration of Business Process Management Activities," in *Software Process and Product Measurement*, ser. Lecture Notes in Computer Science, vol. 5338. Berlin, Heidelberg: Springer, 2008, pp. 141–154.

[57] BPMN Model Interchange Working Group (BPMN MIWG), "BPMN Tools tested for Model Interchange," [Zugriff am 12.10.2014]. [Online]. Available: http://bpmn-miwg.github.io/bpmn-miwg-tools/

[58] Verband der Automobilindustrie e. V. (VDA) (Ed.), *Auto & Normung: Normenausschuss Kraftfahrzeuge Jahresbericht 2006*.

[59] Verband der Automobilindustrie e. V. (VDA) (Ed.), *Auto & Normung: NA-Automobil Jahresbericht 2011*.

[60] DIN Deutsches Institut für Normung e.V. (Ed.), *Normenausschuss Automobiltechnik: Normung - für die Mobilität der Zukunft*, 2009.

[61] Europäisches Parlament, Rat der Europäischen Union (Ed.), *Amtsblatt der Europäischen Union: Richtlinie 2007/46/EG des Europäischen Parlaments und des Rates vom 5. September 2007 zur Schaffung eines Rahmens für die Genehmigung von Kraftfahrzeugen und Kraftfahrzeuganhängern sowie von Systemen, Bauteilen und selbstständigen technischen Einheiten für diese Fahrzeuge*, Oct. 2007.

[62] *Bundesgesetzblatt Teil I Nr. 18*. Bundesanzeiger Verlag, Mai 2012.

[63] Economic Commission for Europe (Ed.), *ECE/TRANS/WP.29/78/Rev.3: Consolidated Resolution on the Construction of Vehicles (R.E.3)*, Jan. 2014.

[64] Verband der Automobilindustrie e. V. (VDA) (Ed.), *Auto & Normung: NA-Automobil Jahresbericht 2012*.

[65] T. P. Kelly, "Managing Complex Safety Cases," in *Current Issues in Safety-Critical Systems*. London: Springer, 2003, pp. 99–115.

[66] T. Kelly, "A Systematic Approach to Safety Case Management," in *SAE 2004 World Congress & Exhibition*, Mar. 2004.

[67] T. Kelly and R. Weaver, "The Goal Structuring Notation - A Safety Argument Notation," in *Proceedings of the Dependable Systems and Networks 2004, Workshop on Assurance Cases*, 2004.

[68] Y. Papadopoulos, M. Walker, D. Parker, E. Rüde, R. Hamann, A. Uhlig, U. Grätz, and R. Lien, "Engineering failure analysis and design optimisation with HiP-HOPS," *Engineering Failure Analysis*, vol. 18, no. 2, pp. 590 – 608, 2011.

[69] Verband der Automobilindustrie e.V. (VDA) (Ed.), *Sicherung der Qualität in der Prozesslandschaft*.

[70] J. C. Grebe and W. M. Goble, "FMEDA - Accurate Product Failure Metrics," Feb. 2007.

[71] R. E. Collett and P. W. Bachant, "Integration of BIT Effectiveness with FMECA," in *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS)*, 1984, pp. 300–305.

[72] Moore Products Co., "FMEDA Analysis of CDM (Critical Discrete Module) - QUADLOG," 1994.

[73] A. Joshi, M. P. E. Heimdahl, S. P. Miller, and M. W. Whalen, *NASA/CR-2006-213953: Model-Based Safety Analysis*, Feb. 2006.

[74] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Jan.-Mar. 2004.

[75] R. Bendraou, J. Jézéquel, M.-P. Gervais, and X. Blanc, "A Comparison of Six UML-Based Languages for Software Process Modeling," *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 662–675, Sep.-Oct. 2010.

[76] J.-P. Blanquart, E. Armengaud, P. Baufreton, Q. Bourrouilh, G. Griessnig, M. Krammer, O. Laurent, J. Machrouh, T. Peikenkamp, C. Schindler, and T. Wien, "Towards Cross-Domains Model-

Based Safety Process, Methods and Tools for Critical Embedded Systems: The CESAR Approach," in *Proceedings of the 30th International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, ser. Lecture Notes in Computer Science, vol. 6894. Berlin, Heidelberg: Springer, 2011, pp. 57–70.

[77] J.-P. Blanquart, E. Armengaud, P. Baufreton, Q. Bourrouilh, J. Machrouh, A. Mitschke, M. Oertel, T. Peikenkamp, and T. Wien, "A multi-domain platform of safety process methods and tools for critical embedded systems," in *Proceedings of the 6th European Congress on Embedded Real Time Software and Systems (ERTS^2)*, 2012.

[78] The Eclipse Foundation, "Eclipse Process Framework Project (EPF)," [Zugriff am 09.09.2014]. [Online]. Available: http://www.eclipse.org/epf

[79] A.-W. Scheer and M. Nüttgens, "ARIS Architecture and Reference Models for Business Process Management," in *Business Process Management: Models, Techniques, and Empirical Studies*, ser. Lecture Notes in Computer Science, vol. 1806. Berlin, Heidelberg: Springer, 2000, pp. 366–379.

[80] J. Becker, M. Rosemann, and C. Uthmann, "Guidelines of Business Process Modeling," in *Business Process Management: Models, Techniques, and Empirical Studies*, ser. Lecture Notes in Computer Science, vol. 1806. Berlin, Heidelberg: Springer, 2000, pp. 30–49.

[81] R. Librino, H. Lönn, F. Stappert, F. Tagliabo, S. Torchiaro, and S. Voget, *Deliverable D2.2.1 v3.0 (final, public): Design methodology (Methodology description for embedded systems development with EAST-ADL)*. MAENAD project, 2013.

[82] The MAENAD Consortium, "MAENAD Methodology Model," Nov. 2013, [Zugriff am 15.10.2014]. [Online]. Available: http://www.maenad.eu/public/Methodology/EAST-ADL_MultiAspect_Final.htm

[83] J. Recker, M. Indulska, M. Rosemann, and P. Green, "How Good is BPMN Really? Insights from Theory and Practice," in *Proceedings of the 14th European Conference on Information Systems (ECIS)*, 2006.

[84] J. Recker, "BPMN Modeling - Who, Where, How and Why," in *BPTrends*, May 2008.

[85] PricewaterhouseCoopers AG Wirtschaftsprüfungsgesellschaft (Ed.), *Zukunftsthema Geschäftsprozessmanagement*, Feb. 2011.

[86] N. Knaub, *Horus Methode: Incorporating the Power of Knowledge from Business Communities*. Horus software GmbH, Nov. 2011.

[87] SparxSystems Software GmbH, "MDG Technology für die Business Process Modeling Notation (BPMN)," [Zugriff am 15.10.2014]. [Online]. Available: http://www.sparxsystems.de/ressourcen/bpmn/

[88] BOC Information Technologies Consulting AG, "Geschäftsprozessmanagement mit ADONIS und ADONIS Prozessportal," [Zugriff am 15.10.2014]. [Online]. Available: http://www.boc-group.com/de/produkte/adonis/

[89] BOC Information Technologies Consulting AG (Ed.), *Das Geschäftsprozessmanagement-Werkzeug ADONIS: Stärken und Einsatzszenarien*.

[90] BOC Information Technologies Consulting AG (Ed.), *Erfolg lässt sich planen: Geschäftsprozessmanagement mit ADONIS und dem ADONIS Prozessportal*.

[91] E. Andrianarison and H. Apperly, *ISO 26262 with Atego Process Director*. Atego, Jan. 2014.

[92] T. Lovric, M. Schneider-Scheyer, and S. Sarkic, "SysML as Backbone for Engineering and Safety - Practical Experience with TRW Braking ECU," in *SAE 2013 World Congress & Exhibition*, Apr. 2014.

[93] K. Svancara, J. Priddy, T. Lovric, J. D. Miller, M. Kudanowski, and W. J. Forbes, "Advantages of the Alternative Method for Random Hardware Failures Quantitative Evaluation - a Practical Survey for EPS," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*,

vol. 6, no. 2, pp. 377–388, Aug. 2013.

[94] K. Svancara, W. J. Forbes, J. Priddy, M. Kudanowski, T. Lovric, and J. D. Miller, "Experience with the second method for EPS hardware analysis: „Evaluation of each cause of safety goal violation due to random hardware failures"," in *VDA Automotive SYS Conference on Quality and Functional Safety Management for Automotive software-based Systems*, May 2012.

[95] M. Walker, Y. Papadopoulos, D. Parker, H. Lönn, M. Törngren, D. Chen, R. Johansson, and A. Sandberg, "Semi-Automatic FMEA supporting complex systems with combinations and sequences of failures," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 2, no. 1, pp. 791–802, Oct. 2009.

[96] Y. Papadopoulos, M. Walker, M.-O. Reiser, M. Weber, D. Chen, M. Törngren, D. Servat, A. Abele, F. Stappert, H. Lonn, L. Berntsson, R. Johansson, F. Tagliabo, S. Torchiaro, and A. Sandberg, "Automatic Allocation of Safety Integrity Levels," in *Proceedings of the 1st Workshop on Critical Automotive applications: Robustness & Safety (CARS)*. New York, NY: ACM, 2010, pp. 7–10.

[97] F. Meurville and P. Cuenot, *Deliverable D3.3.1.b (public): Methodology and Tool specification for analysis of qualitative and quantitative cut-sets issued from error failure propagation analyses*. SAFE project, 2013.

[98] M. Bellotti and R. Mariani, "How future automotive functional safety requirements will impact microprocessors design," *Microelectronics Reliability*, vol. 50, no. 9-11, pp. 1320 – 1326, Sep.-Nov. 2010.

[99] M. Kvas, S. Valach, and P. Fiedler, "Reliability and Safety Issues of FPGA Based Designs," in *Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDeS)*, May 2012, pp. 201–206.

[100] R. Mariani, T. Kuschel, and H. Shigehara, "A flexible microcontroller architecture for fail-safe and fail-operational systems," in *Proceedings of the 2nd HiPEAC Workshop on Design for Reliability (DFR)*, Jan. 2010.

[101] R. Mariani and G. Boschi, "A systematic approach for Failure Modes and Effects Analysis of System-On-Chips," in *Proceedings of the 13th IEEE International On-Line Testing Symposium (IOLTS)*. IEEE, Jul. 2007, pp. 187–188.

[102] R. Mariani and P. Fuhrmann, "Comparing fail-safe microcontroller architectures in light of IEC 61508," in *Proceedings of the 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*. IEEE, Sep. 2007, pp. 123–131.

[103] S. Yoneki, B. Collerais, H. Eki, and R. Mariani, "Fail-operational EPS by distributed architecture," in *Proceedings of the 5th International Munich Chassis Symposium*. Wiesbaden: Springer Fachmedien, 2014, pp. 421–441.

[104] Yogitech Spa, "Company Profile," [Zugriff am 15.10.2014]. [Online]. Available: http://www.yogitech.com/en/company-profile-0

[105] Y.-C. Chang, L.-R. Huang, H.-C. Liu, C.-J. Yang, and C.-T. Chiu, "Assessing Automotive Functional Safety Microprocessor with ISO 26262 Hardware Requirements," in *Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, Apr. 2014.

[106] B. Machmur, A. Hayek, and J. Börcsök, "Practical Application of the Reliability Model for HDL in Safety Related Systems," in *Recent Researches in Circuits, Communications and Signal Processing*. WSEAS Press, Jan. 2013, pp. 31–36.

[107] A. Hayek and J. Börcsök, "Safety-Related ASIC-Design in Terms of the Standard IEC 61508," in *The Third International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO)*. IARIA, Apr. 2013, pp. 16–21.

[108] A. Hayek, M. Schreiber, B. Machmur, and J. Börcsök, "Design and Implementation of on-chip Safety Controller in terms of the Standard IEC 61508," in *Recent Advances in Circuits, Systems and Automatic Control*. WSEAS Press, Dec. 2013, pp. 159–165.

[109] P. Conmy and I. Bate, "Component-Based Safety Analysis of FPGAs," *IEEE Transactions on*

Industrial Informatics, vol. 6, no. 2, pp. 195–205, May 2010.

[110] S.-H. Jeon, J.-H. Cho, Y. Jung, S. Park, and T.-M. Han, "Automotive Hardware Development According to ISO 26262," in *Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT)*. IEEE, Feb. 2011, pp. 588–592.

[111] P. Sinha, "Architectural design and reliability analysis of a fail-operational brake-by-wire system from ISO 26262 perspectives," *Reliability Engineering and System Safety*, vol. 96, no. 10, pp. 1349 – 1359, Oct. 2011.

[112] P. Sinha and V. Agrawal, "Evaluation of Electric-Vehicle Architecture Alternatives," in *Proceedings of the 7th IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, Sep. 2011.

[113] F. Leitner-Fischer and S. Leue, *Technical Report: The QuantUM Approach in the Context of the ISO Standard 26262 for Automotive Systems*. Chair for Software Engineering, University of Konstanz, 2011.

[114] F. Leitner-Fischer and S. Leue, "QuantUM: Quantitative Safety Analysis of UML Models," in *Proceedings of the Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL)*, Apr. 2011.

[115] A. Kramer, "Automotive and medical: can we learn from each other?" *Journal of Software: Evolution and Process*, vol. 25, no. 4, pp. 373–379, Apr. 2013.

[116] E. Holz, *Consistent Application of Safety Analysis Methods using SysML Architecture Models*. ikv++ technologies ag, Nov. 2012.

[117] M. Schulze, J. Mauersberger, and D. Beuche, "Functional Safety and Variability: Can It Be Brought Together?" in *Proceedings of the 17th International Software Product Line Conference (SPLC)*. New York, NY: ACM, 2013, pp. 236–243.

[118] Nobel Media AB, "The Nobel Committee for Physiology or Medicine," 2014, [Zugriff am 22.10.2014]. [Online]. Available: http://www.nobelprize.org/nobel_prizes/medicine/prize_awarder/committee.html

[119] Nobel Media AB, "The Nobel Prize in Physiology or Medicine 2014," 2014, [Zugriff am 22.10.2014]. [Online]. Available: http://www.nobelprize.org/nobel_prizes/medicine/laureates/2014/

[120] R. Messnarz, H.-L. Ross, S. Habel, F. König, A. Koundoussi, J. Unterrreitmayer, and D. Ekert, "Integrated Automotive SPICE and Safety Assessments," *Software Process: Improvement and Practice*, vol. 14, no. 5, pp. 279–288, Sep.-Oct. 2009.

[121] S. Voget and Y. Ma, *Deliverable D4.1.1b (public): SAFE Tool Platform User Guide*. SAFE project, 2013.

Bücher und Dissertationen

[122] J. Matheis, "Abstraktionsebenenübergreifende Darstellung von Elektrik/Elektronik-Architekturen in Kraftfahrzeugen zur Ableitung von Sicherheitszielen nach ISO 26262," PhD dissertation, Karlsruher Institut für Technologie (KIT), 2010.

[123] H.-J. Gevatter and U. Grünhaupt (Eds.), *Handbuch der Mess- und Automatisierungstechnik im Automobil: Fahrzeugelektronik, Fahrzeugmechatronik*, 2nd ed. Berlin, Heidelberg: Springer, 2006.

[124] J. Broy, "Modellbasierte Entwicklung und Optimierung flexibler zeitgesteuerter Architekturen im Fahrzeugserienbereich," PhD dissertation, Karlsruher Institut für Technologie (KIT), 2010.

[125] H. Wallentowitz and K. Reif (Eds.), *Handbuch Kraftfahrzeugelektronik: Grundlagen, Komponenten, Systeme, Anwendungen*. Wiesbaden: Vieweg, 2006.

[126] X. Chen, "Requirements and concepts for future automotive electronic architectures from the view of integrated safety," PhD dissertation, Universität Karlsruhe (TH), 2008.

[127] H.-H. Braess and U. Seiffert (Eds.), *Vieweg Handbuch Kraftfahrzeugtechnik*, 6th ed. Wiesbaden: Vieweg+Teubner, 2011.

[128] T. Streichert and M. Traub, *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*. Berlin, Heidelberg: Springer, 2012.

[129] J. Schäuffele and T. Zurawka, *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen*, 5th ed. Wiesbaden: Springer Vieweg, 2013.

[130] R. Wenger, "Elektronischer Vergabeprozess bei direkten Gütern," PhD dissertation, Universität St.Gallen, 2006.

[131] H. Wildemann, *Advanced Purchasing - Leitfaden zur Einbindung der Beschaffungsmärkte in den Produktentstehungsprozess*. München: TCW-Verlag, 2005.

[132] H.-L. Ross, *Funktionale Sicherheit im Automobil: ISO 26262, Systemengineering auf Basis eines Sicherheitslebenszyklus und bewährter Managementsysteme*. München, Wien: Hanser, 2014.

[133] R. Höhn and S. Höppner (Eds.), *Das V-Modell XT: Anwendungen, Werkzeuge, Standards*. Berlin, Heidelberg: Springer, 2008.

[134] C. Haubelt and J. Teich, *Digitale Hardware/Software-Systeme: Spezifikation und Verifikation*. Berlin, Heidelberg: Springer, 2010.

[135] K. Reif (Ed.), *Automobilelektronik: Eine Einführung für Ingenieure*, 3rd ed. Wiesbaden: Vieweg+Teubner, 2009.

[136] P. H. Feiler and D. P. Gluch, *Model-based engineering with AADL : an introduction to the SAE Architecture Analysis & Design Language*. Upper Saddle River, NJ: Addison-Wesley, 2013.

[137] *PREEvision User Manual Version 6.0*. Stuttgart: Vector Informatik GmbH, 2013.

[138] J. vom Brocke and M. Rosemann (Eds.), *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Berlin, Heidelberg: Springer, 2010.

[139] F. W. Taylor, *The Principles of Scientific Management*. New York, London: Harper & Brothers Publishers, 1911.

[140] S. Koch, *Einführung in das Management von Geschäftsprozessen: Six Sigma, Kaizen und TQM*. Berlin, Heidelberg: Springer, 2011.

[141] D. W. Embley and B. Thalheim (Eds.), *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. Berlin, Heidelberg: Springer, 2011.

[142] J. Becker, W. Probandt, and O. Vering, *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement*. Berlin, Heidelberg: Springer, 2012.

[143] A. Gadatsch, *Grundkurs Geschäftsprozess-Management: Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker*, 7th ed. Wiesbaden: Vieweg+Teubner, 2013.

[144] C. A. Petri, "Kommunikation mit Automaten," PhD dissertation, Technischen Hochschule Darmstadt, 1962.

[145] W. Reisig, *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Wiesbaden: Vieweg+Teubner, 2010.

[146] A.-W. Scheer, *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen*, 4th ed. Berlin, Heidelberg: Springer, 2001.

[147] J. Börcsök, *Funktionale Sicherheit: Grundzüge sicherheitstechnischer Systeme*, 3rd ed. Berlin, Offenbach: VDE, 2011.

[148] T. P. Kelly, "Arguing Safety - A Systematic Approach to Managing Safety Cases," PhD dissertation, University of York, 1998.

[149] M. Werdich (Ed.), *FMEA - Einführung und Moderation: Durch systematische Entwicklung zur übersichtlichen Risikominimierung (inkl. Methoden im Umfeld)*, 2nd ed. Wiesbaden: Vieweg+Teubner, 2012.

[150] M. Hillenbrand, "Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen," PhD dissertation, Karlsruher Institut für Technologie (KIT), 2012.

[151] J. Becker, M. Kugeler, and M. Rosemann (Eds.), *Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung*, 7th ed. Berlin, Heidelberg: Springer Gabler, 2012.

[152] J. Freund and B. Rücker, *Praxishandbuch BPMN 2.0*, 3rd ed. München, Wien: Hanser, 2012.

[153] B. Honke, "Situational Method Engineering for the Enactment of Method-Centric Domain-Specific Languages," PhD dissertation, Universität Augsburg, 2013.

[154] H. Seidlmeier, *Prozessmodellierung mit ARIS®: Eine beispielorientierte Einführung für Studium und Praxis*, 3rd ed. Wiesbaden: Vieweg+Teubner, 2010.

[155] F. Schönthaler, G. Vossen, A. Oberweis, and T. Karle, *Geschäftsprozesse für Business Communities: Modellierungssprachen, Methoden, Werkzeuge*. München: Oldenbourg, 2011.

[156] A.-W. Scheer, M. Boczanski, M. Muth, W.-G. Schmitz, and U. Segelbacher, *Prozessorientiertes Product Lifecycle Management*. Berlin, Heidelberg: Springer, 2006.

[157] *Enterprise Architect User Guide Version 9.3*. Sparx Systems Pty Ltd, 2012.

[158] Y. Papadopoulos, "Safety-Directed System Monitoring Using Safety Cases," PhD dissertation, University of York, 2000.

[159] *medini ™analyze: functional safety and reliability analysis for ISO 26262 (product version 2.5 - May 2014)*. ikv++ technologies ag, 2014.

[160] K. Pohl, H. Hönninger, R. Achatz, and M. Broy (Eds.), *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Berlin, Heidelberg: Springer, 2012.

[161] P. Löw, R. Pabst, and E. Petry, *Funktionale Sicherheit in der Praxis: Anwendung von DIN EN 61508 und ISO/DIS 26262 bei der Entwicklung von Serienprodukten*. Heidelberg: dpunkt, 2010.

Standards und Normen

[AUTOSAR] *AUTOSAR Project Objectives V3.0.0 R4.0 Rev 3*. AUTOSAR Consortium, Dec. 2014.

[BPEL4People] *WS-BPEL Extension for People (BPEL4People) Specification, Version 1.1*. Organization for the Advancement of Structured Information Standards (OASIS), Aug. 2010.

[BPMN] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group (OMG), Jan. 2011.

[BPMNex] *BPMN 2.0 by Example, Version 1.0*. Object Management Group (OMG), Jun. 2010.

[CMMIacq] *CMU/SEI-2010-TR-032: CMMI for Acquisition, Version 1.3*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Nov. 2010.

[CMMIdev] *CMU/SEI-2010-TR-033: CMMI for Development, Version 1.3*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Nov. 2010.

[CMMIsvc] *CMU/SEI-2010-TR-034: CMMI for Services, Version 1.3*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Nov. 2010.

[DIN13849] *DIN EN ISO 13849: Sicherheit von Maschinen - Sicherheitsbezogene Teile von Steuerungen*. Berlin: DIN Deutsches Institut für Normung e.V., Dec. 2008 - Feb. 2012.

[DIN25424-1] *DIN 25424: Fehlerbaumanalyse - Methode und Bildzeichen*. Berlin: DIN Deutsches Institut für Normung e.V., Sep. 1981.

[DIN25424-2] *DIN 25424: Fehlerbaumanalyse - Handrechenverfahren zur Auswertung eines Fehlerbaumes*. Berlin: DIN Deutsches Institut für Normung e.V., Apr. 1990.

[DIN61025] *DIN EN 61025: Fehlzustandsbaumanalyse (IEC 61025:2006)*. Berlin: DIN Deutsches Institut für Normung e.V., Aug. 2007.

[DINfb158] *DIN-Fachbericht 158: Modell zum prozessorientierten Vorgehen in der öffentlichen Verwaltung*. Berlin: DIN Deutsches Institut für Normung e.V., Sep. 2009.

[DINsp91281] *DIN SPEC 91281: Einführung von prozessorientiertem Wissensmanagement in kleinen und mittleren Unternehmen*. Berlin: DIN Deutsches Institut für Normung e.V., Apr. 2012.

[EIA731] *EIA-731.1: Systems Engineering Capability Model*. Arlington, Virginia: Electronic Industries Alliance, Aug. 2002.

[FIDES] *FIDES guide 2009 Edition A (English): Reliability Methodology for Electronic Systems*. FIDES Group, Sep. 2010.

[IEC61508] *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*, 2nd ed. International Electrotechnical Commission, Apr. 2010.

[IECTR62380] *IEC/TR 62380: Reliability data handbook - Universal model for reliability prediction of electronics components, PCBs and equipment*, 1st ed. Geneva: International Electrotechnical Commission, Aug. 2004.

[ISO9001] *DIN EN ISO 9001: Qualitätsmanagementsysteme - Anforderungen*. Berlin: DIN Deutsches Institut für Normung e.V., Dec. 2008.

[ISO15504] *ISO/IEC 15504: Information technology - Process assessment*. Geneva: International Organization for Standardization, Jan. 2004 - Jun. 2013.

[ISO19510] *ISO/IEC 19510: Information technology - Object Management Group Business Process Model and Notation*, 1st ed. Geneva: International Organization for

Standardization, Jul. 2013.

[ISO26262] *ISO 26262: Road vehicles - Functional safety*, 1st ed. Geneva: International Organization for Standardization, Nov. 2011 - Aug. 2012.

[ISOdir-1] *ISO/IEC Directives, Part 1: Consolidated ISO Supplement - Procedures specific to ISO*, 5th ed. Geneva: International Organization for Standardization, International Electrotechnical Commission, 2014.

[ISOdir-2] *ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards*, 6th ed. Geneva: International Organization for Standardization, International Electrotechnical Commission, 2011.

[MIL217F] *MIL-HDBK-217F, notice 2: Military Handbook - Reliability Prediction of Electronic Equipment*. Washington D.C.: Departement of Defense, Dec. 1991.

[MIL338B] *MIL-HDBK-338B: Military Handbook - Electronic Reliability Design Handbook*. Washington D.C.: Departement of Defense, Oct. 1998.

[MIL1629A] *MIL-STD-1629A: Procedures for performing a Failure Mode, Effects and Criticality Analysis*. Washington D.C.: Departement of Defense, Nov. 1980.

[NUREG0492] *NUREG-0492: Fault Tree Handbook*. Washington D.C.: United States Nuclear Regulatory Commission, Jan. 1981.

[SECMM95] *SECMM-95-01 | CMU/SEI-95-MM-003: A Systems Engineering Capability Maturity Model, Version 1.1*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, Nov. 1995.

[SPEM] *Software & Systems Process Engineering Meta-Model Specification, Version 2.0*. Object Management Group (OMG), Apr. 2008.

[SPICEpamdt] *Automotive SPICE® Prozessassessment, Version 2.3*, 1st ed. Oberursel: Verband der Automobilindustrie e.V. (VDA), Jun. 2007.

[SPICEpam] *Automotive SPICE® Process Assessment Model, Version 2.5*. The SPICE User Group, May 2010.

[SPICEprm] *Automotive SPICE® Process Reference Model, Version 4.5*. The SPICE User Group, May 2010.

[UMLinf] *OMG Unified Modeling Language™ (OMG UML), Infrastructure, Version 2.4.1*. Object Management Group (OMG), Aug. 2011.

[UMLsup] *OMG Unified Modeling Language™ (OMG UML), Superstructure, Version 2.4.1*. Object Management Group (OMG), Aug. 2011.

[VModell97] *Entwicklungsstandard für IT-Systeme des Bundes (V-Modell 97), Allgemeiner Umdruck Nr. 250-252*. Ottobrunn: IABG Industrieanlagen-Betriebsgesellschaft mbH, Jun 1997.

[VModellXT] *V-Modell® XT, Version 1.4*. Ottobrunn: IABG Industrieanlagen-Betriebsgesellschaft mbH, Jun. 2012.

[WSBPEL] *Web Services Business Process Execution Language, Version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS), Jan. 2007.

[WS-HT] *Web Services - Human Task (WS-HumanTask) Specification Version 1.1*. Organization for the Advancement of Structured Information Standards (OASIS), Aug. 2010.

Eigene Konferenzbeiträge

[Adler2010] N. Adler, "Mit guten Planungswerkzeugen die Entwicklungskosten im Automobilbau senken," in *FZI Jahresbericht 09|10*. Karlsruhe: FZI Forschungszentrum Informatik, 2010, pp. 56–57.

[Adler2011a] N. Adler, D. Gebauer, C. Reichmann, and K. D. Müller-Glaser, "Modellbasierte Erfassung von Optimierungsaktivitäten als Grundlage zur Systemoptimierung von Elektrik-/Elektronik-Architekturen," in *14. Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV)*. OFFIS-Institut für Informatik, 2011, pp. 163–172.

[Adler2011b] N. Adler, P. Graf, and K. D. Müller-Glaser, "Model-based Consistency Checks of Electric and Electronic Architectures against Requirements," in *Proceedings of the Fourth International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2011), held as part of the 2011 International Conference on Model Driven Engineering Languages and Systems (MoDELS'11)*, vol. 795, 2011, pp. 71–83.

[Adler2012a] N. Adler, P. Graf, and K. D. Müller-Glaser, "Model-Based Consistency Checks of Electric and Electronic Architectures against Requirements," in *Models in Software Engineering: Workshops and Symposia at MODELS 2011, Wellington, New Zealand, October 2011, Reports and Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 7167. Berlin, Heidelberg: Springer, 2012, pp. 262–275.

[Adler2012b] N. Adler, M. Hillenbrand, K. D. Müller-Glaser, E. Metzker, and C. Reichmann, "Graphically notated fault modeling and safety analysis in the context of electric and electronic architecture development and functional safety," in *23rd IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, Oct. 2012, pp. 36–42.

[Adler2013a] N. Adler, S. Otten, P. Cuenot, and K. D. Müller-Glaser, "Performing Safety Evaluation on Detailed Hardware Level according to ISO 26262," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 6, no. 1, pp. 102–113, May 2013.

[Adler2013b] N. Adler, S. Otten, M. Mohrhard, and K. D. Müller-Glaser, "Rapid Safety Evaluation of Hardware Architectural Designs Compliant with ISO 26262," in *24rd IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, 2013, pp. 66–72.

[Adler2014a] N. Adler, S. Otten, M. Schwär, and K. D. Müller-Glaser, "Managing Functional Safety Processes for Automotive E/E Architectures in Integrated Model-Based Development Environments," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 7, no. 1, pp. 103–114, May 2014.

[Adler2014b] N. Adler, E. Metzker, and A. Rudolph, "Design und Analyse funktional sicherer Hardware in einem EBS," in *ATZelektronik*. Springer Automotive Media, Dec. 2014, vol. 9, no. 6, pp. 24–29.

[Adler2014c] N. Adler, E. Metzker, and A. Rudolph, "Design and analysis of functionally safe hardware in an EBS," in *ATZelektronik worldwide*. Springer Automotive Media, Dec. 2014, vol. 9, no. 6, pp. 10–15.

[Cuenot2014] P. Cuenot, C. Ainhauser, N. Adler, S. Otten, and F. Meurville, "Applying Model Based Techniques for Early Safety Evaluation of an Automotive Architecture in Compliance with the ISO 26262 Standard," in *Proceedings of the 7th European Congress on Embedded Real Time Software and Systems (ERTS^2)*,

2014.

[Heinz2009] M. Heinz, N. Adler, K. D. Müller-Glaser, and R. Puls, "Untersuchungen zur Eignung von FlexRay für die Bühnentechnik," in *7. GI/GMM/ITG-Workshop Multi-Nature-Systems: Entwicklung von Systemen mit elektronischen und nichtelektronischen Komponenten*, Feb. 2009.

[Hillenbrand2010a] M. Hillenbrand, M. Heinz, N. Adler, J. Matheis, and K. Müller-Glaser, "Failure mode and effect analysis based on electric and electronic architectures of vehicles to support the safety lifecycle ISO/DIS 26262," in *21st IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, Jun. 2010.

[Hillenbrand2010b] M. Hillenbrand, M. Heinz, K. Müller-Glaser, N. Adler, J. Matheis, and C. Reichmann, "An Approach for Rapidly Adapting the Demands of ISO/-DIS 26262 to Electric/Electronic Architecture Modeling," in *21st IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, Jun. 2010.

[Hillenbrand2010c] M. Hillenbrand, M. Heinz, N. Adler, K. D. Müller-Glaser, J. Matheis, and C. Reichmann, "ISO/DIS 26262 in the Context of Electric and Electronic Architecture Modeling," in *Proceedings of the 1st International Symposium on Architecting Critical Systems (ISARCS)*, ser. Lecture Notes in Computer Science, vol. 6150. Berlin, Heidelberg: Springer, 2010, pp. 179–192.

[Hillenbrand2011] M. Hillenbrand, M. Heinz, K. D. Müller-Glaser, and N. Adler, "A metric-based safety workflow for electric/electronic architectures of vehicles," in *Proceedings of the 2nd International ACM SIGSOFT Symposium on Architecting Critical Systems (ISARCS)*. New York: ACM, 2011, pp. 105–114.

[Koellner2012] C. Köllner, N. Adler, and K. D. Müller-Glaser, "System#: High-level Synthesis of Physical Simulations for FPGA-based Real-Time Execution," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, Aug. 2012, pp. 731–734.

[Schaaff2012] K. Schaaff, R. Degen, N. Adler, and M. T. Adam, "Measuring Affect Using a Standard Mouse Device," in *Biomedizinische Technik / Biomedical Engineering*, vol. 57, Sept. 2012, pp. 761–764.

Deliverables aus Projekt SAFE mit eigenem Beitrag

[D121] Project Management Committee (PMC) members, *Deliverable D1.2.1 (private): Preliminary exploitation plan and license models*. SAFE project, 2012.

[D122] Project Management Committee (PMC) members, *Deliverable D1.2.2 (private): Final exploitation plan and license models*. SAFE project, 2013.

[D21b] SAFE project partners, *Deliverable D2.1b (public): Needs description to apply ISO26262 with architecture and component modeling*. SAFE project, 2012.

[D21c] SAFE project partners, *Deliverable D2.1c (public): Needs description to apply ISO26262 with architecture and component modeling*. SAFE project, 2014.

[D231b] SAFE project partners, *Deliverable D2.3.1b (public): Project Glossary*. SAFE project, 2014.

[D322] P. Cuenot, N. Adler, and S. Otten, *Deliverable D3.2.2 (public): Proposal for extension of Meta model for hardware modeling*. SAFE project, 2013.

[D322b] P. Cuenot, N. Adler, and S. Otten, *Deliverable D3.2.2 (public): Proposal for extension of Meta model for hardware modeling*. SAFE project, 2013.

[D333a] N. Adler, S. Otten, M. Hillenbrand, A. Baumgart, T. Peikenkamp, and M. Khalil, *Deliverable D3.3.3a (protected): Initial specification for comparison of architecture*. SAFE project, 2012.

[D333b] N. Adler, S. Otten, E. Metzker, T. Peikenkamp, M. Oertel, A. Baumgart, and M. Khalil, *Deliverable D3.3.3b (public): Final specification for comparison of architecture*. SAFE project, 2013.

[D34a] M. Schulze, M. Oertel, T. Peikenkamp, N. Adler, and M. Hillenbrand, *Deliverable D3.4a (protected): Guideline for description of variant management techniques*. SAFE project, 2013.

[D34b] M. Schulze, M. Oertel, T. Peikenkamp, N. Adler, and M. Hillenbrand, *Deliverable D3.4b (public): Guideline for description of variant management techniques*. SAFE project, 2014.

[D35a] Vector, BMW, Conti-F, Conti-G, fortiss, FZI, Dassault Systèmes, LaBRi, OFFIS, PS, TTTech, VEEM, *Deliverable D3.5a (protected): Initial proposal for meta model definition*. SAFE project, 2012.

[D35b] Vector, BMW, Conti-F, Conti-G, Dassault Systèmes, fortiss, FZI, Itemis, LaBRi, OFFIS, PS, TTTech, VEEM, *Deliverable D3.5b (public): Intermediary release of System, SW, HW reference meta-model definition*. SAFE project, 2013.

[D35c] Vector, BMW, Conti-F, Conti-G, Dassault Systèmes, fortiss, FZI, Itemis, LaBRi, OFFIS, PS, TTTech, VEEM, *Deliverable D3.5c (public): Final release of System, SW, HW reference meta-model definition*. SAFE project, 2013.

[D35d] Vector, BMW, Conti-F, Conti-G, Dassault Systèmes, fortiss, FZI, Itemis, LaBRi, OFFIS, PS, TTTech, VEEM, *Deliverable D3.5d (public): Final release of System, SW, HW reference meta-model definition*. SAFE project, 2014.

[D426a] N. Adler, S. Otten, and E. Metzker, *Deliverable D4.2.6a (private): First version of plug-in for safety and multi criteria architecture modeling and benchmarking*. SAFE project, 2013.

[D426b] N. Adler, S. Otten, and E. Metzker, *Deliverable D4.2.6b (public): Final version of plug-in for safety and multi criteria architecture modeling and benchmarking*. SAFE project, 2014.

[D43b] E. Metzker, N. Adler, and S. Otten, *Deliverable D4.3b (protected): PREEvision Extensions*. SAFE project, 2013.

[D6a] K. Niewiadomski, P. Cuenot, T. Peikenkamp, T. Wenzel, M. Khalil, A. Rudolph, J. Lucas, J. Kemmerich, S. Voget, H.-L. Ross, A. Eckel, E. Biendl, N. Adler, and S. Otten, *Deliverable D6a (public): Initial methodology and application rules*. SAFE project, 2014.

[D6b] P. Cuenot, T. Peikenkamp, T. Wenzel, M. Khalil, A. Rudolph, J. Lucas, J. Kemmerich, S. Voget, H.-L. Ross, A. Eckel, E. Biendl, N. Adler, S. Otten, and S. Buch, *Deliverable D6b (public): Final methodology and application rules*. SAFE project, 2014.

[D71a] SAFE project partners, *Deliverable D7.1a (protected): Initial dissemination and standardization plans*. SAFE project, 2012.

[D71b] SAFE project partners, *Deliverable D7.1b (private): Final dissemination and standardization plans*. SAFE project, 2013.

[D72] E. Metzker, N. Adler, S. Otten, S. Voget, M. Khalil, and M. Schulze, *Deliverable D7.2 (public): Training Material*. SAFE project, May 2014.

[DE35a] M. Khalil, N. Adler, A. Baumgart, S. Otten, and E. Metzker, *Deliverable D3.5a (protected): Initial specification for a multidimensional design space exploration method and architecture benchmarking metrics*. SAFE-E project, 2013.

[DE44a] M. Khalil, N. Adler, S. Otten, and E. Metzker, *Deliverable D4.4a (protected): First version of plug-in for safety and multi criteria architecture modeling and benchmarking*. SAFE-E project, 2013.

Betreute studentische Abschlussarbeiten

[Bellwald2011] M. Bellwald, *Diplomarbeit ID-1462: Ausführungsstrategien für die Synthese von Elektrik-/Elektronik-Architekturen*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2011.

[Degen2011] R. Degen, *Diplomarbeit ID-1500: Entwicklung einer drucksensitiven Maus für die Durchführung physio-ökonomischer Studien*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2011.

[Diehl2014] J. Diehl, *Diplomarbeit ID-1910: Modell-basierte Beschreibung und Analyse von komplexen Hardware-Komponenten im Kontext funktionaler Sicherheit*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), laufende Arbeit mit voraussichtl. Veröffentlichung im Dez. 2014.

[Glock2012] T. Glock, *Diplomarbeit ID-1603: Entwicklungsunterstützung für modulare Elektrik-/Elektronik-Architekturen im Kontext von E-Fahrzeugen*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2012.

[Kramer2012] J. Kramer, *Diplomarbeit ID-1552: Erstellung eines Systemarchitektur-Modells für den Antriebsstrangdes Mercedes-Benz SLS AMG E-CELL unter Berücksichtigung funktionaler Sicherheit*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2012.

[Lin2010] Y. Lin, *Diplomarbeit ID-1386: Performance-Analyse für Automobile Architekturmodelle*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2010.

[Otten2012] S. Otten, *Masterarbeit ID-1641: Hardware Fehlermodelle für Elektrik-/Elektronik-Architekturen unter Berücksichtigung funktionaler Sicherheit*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2012.

[Schmitt2012] T. Schmitt, *Diplomarbeit ID-1634: Multikriterielle Bewertung und Benchmarking von Elektrik-/Elektronik-Architekturen im Kontext funktionaler Sicherheit*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2012.

[Schwaer2010] M. Schwär, *Bachelorarbeit ID-1451: Sensitivitätsanalyse als Grundlage zur Entscheidungsfindung bei der multikriteriellen Optimierung von Elektrik-/Elektronik-Architekuren*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2010.

[Schwaer2013] M. Schwär, *Masterarbeit ID-1736: Prozessmodell für die Entwicklung von sicherheits-relevanten automobilen Systemen*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2013.

[Weischer2014] T. Weischer, *Bachelorarbeit ID-1820: Verwendung von integrierten Prozessbeschreibungen in modell-basierten Elektrik/Elektronik-Architektur Entwicklungsumgebungen*. KIT, Institut für Technik der Informationsverarbeitung (ITIV), 2014.

Projekte im Kontext dieser Arbeit mit eigenem Beitrag

[SAFE] ITEA 2 Projekt, EUREKA cluster program Σ! 3674, *Safe Automotive soFtware architEcture (SAFE)*. Bundesministerium für Bildung und Forschung (BMBF), Juli 2011 - Dezember 2014.

[EEExplore] KMU-innovativ - Verbundprojekt, *EEExplore: Methoden und Werkzeuge zur E/E-Architekturexploration und -Optimierung*. Bundesministerium für Bildung und Forschung (BMBF), November 2008 - Januar 2011.

[MSM] ZIM-Kooperationsprojekt, *„Mechatronic Systems Modeler - MSM" zur interdisziplinären Modellierung technischer Systeme*. Bundesministerium für Wirtschaft und Energie (BMWi), Januar 2011 - Oktober 2011.

[FTAEE] ZIM-Kooperationsprojekt, *Fehlermodellierung und -analyse für E/E-Architekturmodelle (FTA/EE)*. Bundesministerium für Wirtschaft und Energie (BMWi), November 2010 - Oktober 2011.

STEINBUCH SERIES ON ADVANCES IN INFORMATION TECHNOLOGY

Institut für Technik der Informationsverarbeitung
Karlsruher Institut für Technologie (KIT) | ISSN 2191-4737

Band 1 OLIVER SANDER
Skalierbare adaptive System-on-Chip-Architekturen für Inter-Car und Intra-Car Kommunikationsgateways. 2011
ISBN 978-3-86644-601-4

Band 2 BENJAMIN GLAS
Trusted Computing für adaptive Automobilsteuergeräte im Umfeld der Inter-Fahrzeug-Kommunikation. 2011
ISBN 978-3-86644-602-1

Band 3 RALF NÖRENBERG
Effizienter Regressionstest von E/E-Systemen nach ISO 26262. 2012
ISBN 978-3-86644-135-4

Band 4 MARTIN HILLENBRAND
Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen. 2012
ISBN 978-3-86644-803-2

Band 5 MATTHIAS HEINZ
Modellbasierte Entwicklung und Konfiguration des zeitgesteuerten FlexRay Bussystems. 2013
ISBN 978-3-86644-816-2

Band 6 Nicht erschienen

Band 7 CHRISTOPH SCHMUTZLER
Hardwaregestützte Energieoptimierung von Elektrik/Elektronik-Architekturen durch adaptive Abschaltung von verteilten, eingebetteten Systemen. 2012
ISBN 978-3-86644-875-9

Band 8 MARTIN JAENSCH
Modulorientiertes Produktlinien Engineering für den modellbasierten Elektrik/Elektronik-Architekturentwurf. 2012
ISBN 978-3-86644-863-6

Band 9 MAHTAB NIKNAHAD
Using Fine Grain Approaches for highly reliable Design of FPGA-based Systems in Space. 2013
ISBN 978-3-7315-0038-4

Band 10 NICO ADLER
Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262. 2015
ISBN 978-3-7315-0442-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

STEINBUCH SERIES ON ADVANCES
IN INFORMATION TECHNOLOGY

Karlsruher Institut für Technologie (KIT)
Institut für Technik der Informationsverarbeitung

Die Absicherung von funktionaler Sicherheit ist im Kontext der zunehmenden Elektrifizierung von Fahrzeugen ein herausforderndes Unterfangen. Mit der Publikation von ISO 26262 „Road vehicles - Functional safety“ wurde ein domänenspezifischer Standard zur Umsetzung von funktionaler Sicherheit über den gesamten Produktlebenszyklus in der Automobilindustrie etabliert.

Diese Arbeit behandelt eine integrierte Prozessmodellierung und ein Prozessmanagement innerhalb von modellbasierten Architekturbeschreibungssprachen, um den Umgang mit umfangreichen Standards zu erleichtern. Die Anwendung der erarbeiteten Methodik auf ISO 26262 ermöglicht im Konkreten eine detaillierte Betrachtung der Aktivitäten für die Produktentwicklung auf Hardwareebene. Für die modellbasierte Entwicklung von funktional sicherer Hardware wurden ein Konzept und eine Vorgehensweise erarbeitet, welche sich durch die Beschreibung von Hardwaredesigns, Anreicherung um Fehlerinformationen sowie Ausführung der geforderten Sicherheitsevaluationen auszeichnet. Die dargestellte Implementierung ermöglicht iteratives Design, Analyse und Optimierung der Hardware.

ISSN 2191-4737
ISBN 978-3-7315-0442-9