Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

TECHNISCHE UNIVERSITÄT WIEN

Dissertation

Safety-Vorgehensmodell zur Konzeption und Entwicklung von sicherheitskritischen Systemen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaft unter der Leitung von

Univ.-Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard

Institut für Konstruktionswissenschaften und Technische Logistik (E307)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Dipl.-Ing. (FH) Hans Tschürtz, MSc, MSc
Matr. Nr. 9027002
Kerschbaumgasse 3/1/502
1100 Wien

Wien, am

Kurzfassung

Unsere Gesamtsysteme werden immer komplexer. Nicht nur, dass sie aus einer wachsenden Anzahl von Teil-Systemen bestehen. Auch der vermehrte Einsatz von Softwarelösungen in technischen Teil-Systemen und die Verkoppelung mit anderen Systemteilen, sowie das Zusammenspiel mit einer meist undefinierten Gesamtsystemumgebung steigern die Komplexität, was im Fehlerfall sehr gefährlich werden kann. Umso wichtiger wird es, nicht nur das Augenmerk auf die einzelnen Teil-Systeme zu legen, sondern die gesamtheitliche Sicht im Auge zu behalten.

Im Automotive-Bereich zum Beispiel spricht man von mehr als 100 Millionen Lines of Code, verteilt auf mindestens 70 Steuergeräte. Hier können systeminhärente Softwarefehler nur mehr schwer vermieden werden. Durch die Verkoppelung der Steuergeräte, sowie durch den Einfluss anderer technischer Systeme und dem Einwirken der Umwelt wird der Sachverhalt noch erschwert. Dadurch wird die Anzahl an inhärenten Systemzuständen im Fahrzeug beträchtlich erhöht. Durch ein Zusammentreffen von Fehlfunktionen einzelner Steuergeräte und der Beeinflussung von außen entstehen unvorhersehbare Systemzustände, die zu schwerwiegenden Unfällen führen können.

Nancy Leveson, Professorin für Aeronautics und Astronautics am Massachusetts Institute of Technology (MIT), internationale Galionsfigur der Safety-Welt, bestätigt diesen Sachverhalt und spricht in diesem Zusammenhang von sogenannten „System Accidents“, einer gefährlichen Art von Unfällen. Aktuell bekannte Safety-Normen gehen auf diese Problemstellung nicht näher ein. Es existieren diesbezüglich auch keine geeigneten Vorgehensmodelle oder Analyse-Methoden, die dieses Problem lösen könnten.

In Bezug auf die genannte Problemstellung werden in dieser Arbeit die wichtigsten Vorgehensweisen, die diversen Ansätze verschiedener Safety-Normen, sowie die dabei anzuwendenden Analyse-Methoden kritisch untersucht und deren Anwendbarkeit auf die Problemstellung analysiert. Insbesondere wird bei den Safety-Normen die Perspektive sowohl der funktionalen Sicherheit (IEC 61508, ISO 26262), als auch die der Systemsicherheit (MIL-STD 882E, DO 178C) eingehend betrachtet. Des Weiteren werden neue Modelle entwickelt und vorhandene erweitert, um die Problemstellung intensiver analysieren zu können.

Darauf basierend wird ein Safety-Vorgehensmodell zur Konzeption und Entwicklung von sicherheitskritischen Systemen vorgeschlagen. Dieses Vorgehensmodell basiert auf Grundlage einer Erweiterten Fehlerkette, um die ganzheitliche Betrachtung von Fehlermechanismen und deren Kausalbeziehungen besser verstehen zu können. Das Vorgehensmodell verfolgt dabei den Ansatz der inhärenten Systemsicherheit. Das bedeutet, dass durch eine systematische Vorgehensweise mit entsprechend entwickelten Analyse-Methoden die Basis für ein *inhärent sicheres Design* (Inherent Safe System Design) gelegt werden soll.

Vor allem das systematisch-methodische Vorgehen selbst, mit seiner Prägung für inhärente Systemsicherheit, sowie die dafür speziell entwickelten Analyse-Methoden, aber auch die Erweiterung traditioneller Methoden, prägen den Neuigkeitswert der Arbeit.

Abstract

Our overall systems become more complex every day. Not only, because they consist of a growing number of sub-systems. Also the increasing application of software solutions in technical sub-systems, the coupling with other system parts, and the interaction with a mostly undefined overall system environment, raise the complexity, which can be very dangerous in the event of a fault. So much the important it gets to not only turn one's attention to the singular sub-systems, but to keep the holistic perspective in view.

In the automotive sector, for instance, we currently talk about more than 100 million lines of code, distributed onto a minimum of 70 control units. Thus, system inherent software faults can hardly be avoided. Due to the coupling of the control units, as well as due to the impact of other technical systems und the influence of the environment, these circumstances are impeded even further. Thereby, the number of inherent system states in a vehicle gets increased considerably. If malfunctions of single control units happen concurrently, together with external influences, unforeseeable system states occur, which can lead to severe accidents.

Nancy Leveson, professor in aeronautics and astronautics at the Massachusetts Institute of Technology (MIT), international figurehead of the safety world, confirms this circumstance and talks in this context about so-called „system accidents“, a dangerous kind of accidents. Presently known safety standards do not address this problem further. In this respect, also no suitable procedure models or analysis methods exist to solve this problem.

With regard to the stated problem, this work will critically examine the most important procedure models, the diverse approaches of different safety standards, as well as the analysis methods to be applied, and also will analyse their applicability onto the said problem. Particularly for the safety standards, both will be examined closely, the functional safety perspective (IEC 61508, ISO 26262), and the system safety perspective (MIL-STD 882E, DO 178C). Furthermore, new models will be developed and existing ones extended, in order to analyse the problem more intensively.

On the basis of these analyses, a procedure model for conception und development of safety-critical systems will be recommended, the so-called „safety approach model“. This safety approach model is founded on the principle of an extended failure model, in order to better understand the holistic inspection of fault mechanisms and their causal relationships. The approach model pursues thereby the idea of inherent system safety. This means, that through a systematic approach, together with correspondingly developed analysis methods, the fundaments for an “inherent safe System Design” should be installed.

Especially this systematic-methodical approach by itself, with its imprint for inherent system safety, as well as the therewith specifically developed analysis methods, but also the extension of traditional methods, form the den news value of this work.

Vorwort

Angeregt von der Baukunst und fasziniert von den Skylines großer Metropolen, mit ihren bis zu 800 Meter hohen Wolkenkratzern, ist mir der Begriff „Inherent Safe Systems“ in den Sinn gekommen. Diese modernen Mega-Bauten scheinen inhärent sicher zu sein, natürlich im Sinne von „Safe“. Es muss doch möglich sein, diesen bewährten Ansatz aus der modernen Bauwelt auch auf softwarebasierte elektronische Systeme umzulegen, so mein Gedanke! Das Ziel sollte also dabei sein, den Betrieb von komplexen sicherheitskritischen elektronischen Systemen stets, und vor allem sicher, aufrecht zu erhalten. Da sich die europäische Industrie fast ausschließlich mit Funktionaler Sicherheit und dem damit zusammenhängenden sicheren Zustand beschäftigt, der in vielen Fällen auch gefährlich sein kann, wird in Zukunft ein Paradigmenwechsel notwendig sein. Der Ansatz sollte sich von der Funktionalen Sicherheit hin zu einer inhärenten Systemsicherheit entwickeln.

Im Gegensatz zu „materiellen“ architektonischen Bauten aus dem Bauwesen ist die Software elektronischer Geräte jedoch immateriell und wird praktisch nicht durch physikalische Gesetze begrenzt. Der Einsatz von Software erhöht einerseits die Flexibilität von technischen Gesamtsystemen, erhöht aber andererseits auch die Anzahl von inhärent gefährlichen Systemzuständen. Hier ist die Sachlage also etwas abstrakter zu betrachten. Daher wird ein wissenschaftlich fundierter Ansatz, nach dem Motto „Inherent System Safety Approach“ benötigt.

Motiviert vom ingenieursmäßigen Vorgehen der Architekten im Bauwesen, die den Schwerpunkt auf Konstruktion, Design und Planung legen, habe ich einen ähnlichen Ansatz gewählt. Der Schwerpunkt soll damit auf einer inhärent sicheren Konzeption liegen, die in ein „Inherent Safe System Design“ mit entsprechender Planung weiterentwickelt wird.

Ich leite an der FH Campus Wien das „Vienna Institut for Safety and Systems Engineering“ und beschäftige mich mit der Safety-Thematik seit mittlerweile 10 Jahren. Im Zuge der durchgeführten F&E-Projekte habe ich viel Erfahrung in diese Arbeit mitnehmen können und möchte auch in Zukunft diesen „Inherent System Safety Approach“ weiterverfolgen, um der Industrie ein Safety-Vorgehensmodell zur Konzeption und Entwicklung von sicherheitskritischen Systemen zur Verfügung stellen zu können.

Ich möchte auch noch erwähnen, dass diese Dissertation noch lange nicht alles ist, was es dazu braucht. Wir werden in Zukunft noch viele wissenschaftliche Ergebnisse benötigen, um diesen Ansatz in der Industrie wirkungsvoll umsetzen zu können. Ich möchte daher auch andere Dissertantinnen und Dissertanten, aber auch meine Studierenden dazu motivieren, diese Thematik weiterzuverfolgen. Die Wissenschaft ist es unserer Gesellschaft und vor allem unseren Kindern schuldig, dass wir unsere Zukunft mit „Inherent Safe Systems“ gestalten können.

Sicherheit hat sehr wohl ihren Preis, das Risiko einen noch höheren!

Wien, 2015 Hans Tschürtz

Danksagung

Ich möchte mich hiermit bei Herrn Univ. Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard recht herzlich für die Übernahme der Betreuung, das entgegengebrachte Vertrauen, seine Anregungen und sein Interesse an der Thematik bedanken.

Herrn Univ. Prof. Dipl.-Ing. Dr. Kurt Matyas danke ich, dass er die Zweitbetreuung übernommen hat und besonders bedanke ich mich für seine Unterstützung und seine Hilfsbereitschaft.

Des Weiteren möchte ich auch Dipl.-Ing. Dr. Andreas Gerstinger für die vielen komplizierten Fragen und Diskussion danken, die mich in der Thematik immer wieder herausgefordert aber auch gestärkt haben.

Inhaltsverzeichnis

1. Einführung 1
1.1 Problemstellung und Motivation 4
1.2 Forschungsfragen 8
1.3 Zielsetzung und Neuigkeitswert? 9
1.4 Gliederung und Aufbau der Arbeit 9

2. Stand der Wissenschaft und Technik 11
2.1 Begriffe und Definitionen 11
2.1.1 Grundbegriffe der technischen Systemtheorie 11
2.1.2 Klassische Fehlerkette 14
2.1.3 Dependability 15
2.1.4 Hazard Triangle 16
2.1.5 Fokus System Safety 17
2.2 Safety Normen 20
2.2.1 IEC 61508 20
2.2.2 ISO 26262 24
2.2.3 MIL-STD 882E 26
2.2.4 DO 178C 30
2.3 Integrativer Safety Prozess (ISaPro®) 33
2.4 Support Prozesse 37
2.4.1 Konfigurationsmanagement 37
2.4.2 Verifikation und Validation 39
2.4.3 Qualitätssicherung 40
2.5 Analyse-Methoden 41
2.5.1 Preliminary Hazard List (PHL) 43
2.5.2 Functional Failure Analysis (FFA) 44
2.5.3 Hazard and Operability Analysis (HAZOP) 45
2.5.4 Fault Tree Analysis (FTA) 46
2.5.5 Failure Modes and Effects Analysis (FMEA) 47
2.6 Safety Case: Goal Structuring Notation (GSN) 48
2.7 Abgrenzung zu vorhandenen Vorgehensmodellen 50
2.7.1 Klassische Vorgehensmodelle 51
2.7.2 Prozessreifegradmodelle 53

2.7.3 Vorgehensmodelle auf wissenschaftlicher Ebene 54

3. Safety-Vorgehensmodell 55
3.1 Problemraum 58
3.1.1 Projekt-Initialisierung 61
3.1.2 Konzeptionierung 72
3.1.3 Preliminary Hazard Identification (PHI) 77
3.1.4 Zusammenfassung des Problemraums 81
3.2 Modellierungsraum 83
3.2.1 Projektplanung 86
3.2.2 Requirements Engineering 93
3.2.3 Functional Hazard Evaluation (FHE) 98
3.2.4 System Design 100
3.2.5 Preliminary System Safety Evaluation (PSSE) 103
3.2.6 Zusammenfassung des Modellierungsraums 108

4. Neue Modelle und Methoden 111
4.1 Umfassende Fehlerkette und Fehlerklassen 111
4.1.1 Erweiterung der Fehlerkette 111
4.1.2 Erweiterte Fehlerkette 113
4.1.3 Fehlerklassen 116
4.2 Shell-Modell 117
4.2.1 Aufbau des Shell-Modells 118
4.2.2 Identifikation der unmittelbaren System-Umgebung 121
4.3 Projekt-Stakeholder-Analyse 123
4.4 Extended Functional Failure Analysis 126
4.5 Exogene Hazard Analysis (exHA) 129
4.6 Hazard-Analyse und Safety-Risiko-Assessment 130
4.6.1 Hazard-Identifikation und -Analyse 131
4.6.2 Failure Analyse und Hazard Beschreibung 133
4.6.3 Safety Risiko Assessment 134
4.7 Safety Case Pattern 141

5. Anwendungsbeispiel: Side-Stand 144
5.1 Side-Stand: Problemraum – Makro-Konzept Entwicklung 145
5.1.1 Side-Stand: Projekt-Initialisierung 146
5.1.2 Side-Stand: Konzeptionierung 151
5.1.3 Side-Stand: Preliminary Hazard Identification (PHI) 157
5.1.4 Side-Stand: Abschluss des Problemraums 165
5.2 Modellierungsraum – Projekthandbuch und Design Spezifikationen 170
5.2.1 Side-Stand: Projektplanung 170
5.2.2 Side-Stand: Requirements Engineering 174
5.2.3 Side-Stand: Functional Hazard Evaluation (FHE) 181
5.2.4 Side-Stand: System Design 188
5.2.5 Side-Stand: Preliminary System Safety Evaluation (PSSE) 198

5.2.6 Side-Stand: Abschluss im Modellierungsraum 210
6. Zusammenfassung und Schlussfolgerungen 212
6.1 Ergebnisse und Diskussion 214
6.2 Nutzen der Erkenntnisse aus der Dissertation 216
Anhang A: Checklisten 217
Anhang B: ISPro® Prozesse 219
Anhang C: Analyseergebnisse 231
Anhang D: SSCS HW-Lösungsvorschlag für HWReq_2 236
Literaturverzeichnis 237
Internetreferenzen 242
Glossar 244
Abkürzungen 249
Tabellenverzeichnis 253
Abbildungsverzeichnis 255
Lebenslauf 258

1. Einführung

In einer Welt des kontinuierlichen, technologischen Wandels und der ständig wachsenden Anforderungen steigt die Komplexität von zu entwickelnden Systemen[1] immer mehr und immer schneller. Der steigende Einsatz von Softwarelösungen erhöht einerseits die Flexibilität für Änderungen und Erweiterungen, andererseits kann das sichere Funktionieren nur sehr bedingt bewiesen werden.

Der Anspruch an die Systemsicherheit wird dabei immer wichtiger, im Speziellen der Nachweis, dass Systeme als sicher eingestuft werden können. Die Rechtsanwälte Eisenberg, Gildeggen, Reuter und Willburger [EGRW_08] begründen das mit: „Trotz der heutigen zivilrechtlichen Herstellerhaftung nimmt die Zahl der sicherheitsrelevanten Produktfehler ständig zu.“ Die Produkthaftung stellt beispielsweise den Anspruch, System-Entwicklungen nach dem aktuellen Stand der Wissenschaft und Technik zu erfüllen [ER_85, Artikel 7, Punkt e]. Normen und Standards werden im Schadensfall gerichtlich dafür herangezogen. Viele Fälle in der Vergangenheit zeigen anhand langer Prozessverhandlungen, dass dieser Nachweis immer wieder sehr schwierig zu erbringen ist. Typische Beispiele dafür liefern die Gletscherbahnkatastrophe von Kaprun [Kap_00] oder das Gaspedal Problem von Toyota [STo_09], aber auch explodierende Handys [SHa_09] in der Consumer Elektronik sowie der Rolltreppenunfall in Beijing [GCh_11]. Im Speziellen bei Personenschäden müssen Unternehmen aufgrund der strafrechtlichen Relevanz bei der Staatsanwaltschaft Rechenschaft ablegen.

In der Geschichte der Entwicklung sicherheitskritischer[2] Systeme gibt es die verschiedensten Ansätze, um Systeme sicher auszulegen und aufzubauen. Diese reichen von einfachen Lösungen im Schaltungsentwurf bis hin zu sehr komplexen Ausführungen von integrierten Schaltkreisen. Derartige Lösungen sind heutzutage nicht mehr wegzudenken und eine wichtige Grundvoraussetzung für sicherheitskritische und sicherheitsrelevante[3] Systeme. Bei sehr komplexen

[1] Der Begriff "System" wird hier als Synonym für technische Anlagen oder Einrichtungen sowie für technische Geräte und Produkte verwendet.

[2] Der Begriff "sicherheitskritisch" wird verwendet, wenn das Fehlverhalten von Betrachtungselementen eines Systems zu einem gefährlichen Zustand führt und unter Kontrolle gehalten werden muss. Hier spricht man von einer direkten Gefährdung.

[3] Der Begriff "sicherheitsrelevant" wird verwendet, wenn ein Fehler im System oder eine Kombination von Fehlern, aber auch ein Zusammenwirken von Ereignissen, zu einer Bedrohung führen. Der Begriff sicherheitsrelevant bewegt sich in einem weiteren Kontext als sicherheitskritisch, da ein Einzelfehler nicht zwingend kritisch sein muss. Das

und umfangreichen Systemen reicht dieser Ansatz jedoch nicht mehr aus. Das Thema "Safety" und die darauf basierenden Safety-Normen schufen in den letzten Jahren eine zusätzliche Perspektive, komplexe Systeme sicherer zu machen. Safety-Normen stellen den Anspruch der Identifikation von Hazards (Gefährdungen), der darauf basierenden Risiko-Einschätzung (Risk Assessment), der Risiko-Reduktion durch entsprechende Maßnahmen und des Rest-Risiko Managements. Je nach Risiko-Klasse stellen Safety-Normen entsprechende Safety-Anforderungen an die zu entwickelnde Sicherheitsfunktion.

Zu den wichtigsten Safety-Normen in dieser Arbeit zählen die generische Norm IEC 61508 [IEC08_10] und die davon abgeleitete ISO 26262 [ISO26_11]. Letztere ist eine sektorspezifische Safety-Norm für die Automotive-Branche für Personenkraftwagen bis 3,5 Tonnen. Die Norm wurde am 15. November 2011 in Kraft gesetzt, das heißt, sie ist noch sehr neu und es gibt wenig Erfahrung bei der Umsetzung – gerade in Bezug darauf liegen noch viele Fragestellungen vor, wie etwa ob der dargestellte System Safety Lifecycle und die geforderten Methoden zur Unfallvermeidung tatsächlich ausreichen. Die ISO 26262 wird vermutlich einen massiven Einfluss auf die Automobil-Industrie haben und ein Umdenken im Entwicklungsprozess verlangen, da strukturelle Vorgehensweisen gefordert sind, die in dieser Art und Weise heute noch nicht durchgeführt werden. Ross [Ros_14, S.1] bringt die Thematik auf den Punkt mit: "Die ISO 26262 verändert zurzeit die Fahrzeugentwicklung in einer Form, wie man sich das vor 10 Jahren, als das Thema Funktionssicherheit in der Automobilindustrie stärker in den Vordergrund getreten ist, nie hätte vorstellen können."

Trotz allem stellt die ISO 26262, genauso wie die IEC 61508, lediglich Anforderungen an die funktionale Sicherheit (Functional Safety[4]) und beschäftigt sich nicht mit der darüber hinausgehenden Systemsicherheit (System Safety[5]). Ziel der funktionalen Sicherheit ist es, unter Berücksichtigung eines festgelegten gefährlichen Vorfalls einen sicheren System-Zustand herbeizuführen und danach aufrechtzuerhalten, argumentiert Reif [Rei_12, S.255]. Sicherheitsfunktionen alleine reichen jedoch morgen nicht mehr aus**,** um die Sicherheit eines Systems in seiner Umgebung auf einem akzeptablen Level zu halten. Dies zeigt auch die ISO 26262 implizit, da sie die Beherrschbarkeit (Controllability) des Fahrzeuges durch den Fahrer sowie die Umgebung des Fahrzeugs stark miteinbezieht.

Darüber hinaus sollte die Sicherheit im System integriert sein. Dieser Ansatz wird in der Systemsicherheit mit "System Safety" bezeichnet. System Safety ist eine Engineering-Disziplin, bei der die Sicherheit in das System "hinein" entwickelt wird. Ericson [Eric2_05, S.2] beschreibt dies ganz klar: "System safety is an engineering discipline for developing safe systems and products,

---

Zusammenwirken von verschiedenen Fehlern und/oder Ereignissen kann hingegen gefährliche Auswirkungen haben. Hier spricht man von einer indirekten Gefährdung.

[4] Funktionale Sicherheit bedeutet die Vermeidung von Fehlfunktionen in technischen Systemen, die die Sicherheit beeinträchtigen. Die funktionale Sicherheit ist Teil der Gesamtsicherheit.

[5] System Safety strebt die Erreichung von ausreichender Sicherheit für ein technisches System in seiner Systemumgebung unter Bezugnahme auf alle notwendigen Engineering und Management Aktivitäten an.

where safety is intentionally designed into the system or product."[6] Die wesentlichen Grundanforderungen für System Safety liefert der amerikanische Standard MIL-STD 882E (Standard Practice for System Safety) [MIL_05] auf der System-Ebene sowie der Standard DO 178C [DO_11] auf der Software-Ebene. Der Anspruch an Entwicklungen von Software für sicherheitskritische Systeme ist dabei besonders hoch.

In vielen Branchen wird darüber hinaus ein genau definierter Prozessreifegrad[7] für die Entwicklung von Systemen gefordert, um systematischen Fehlern in der Software-Entwicklung entgegenzuwirken. Im mitteleuropäischen Raum wird dafür meist die ISO/IEC 15504 [ISO15_06] herangezogen, die auch unter dem Namen SPICE[8] (Software Process Improvement an Capability dEtermination) bekannt ist. Eine spezielle Ausprägung dieser Norm, unter dem Namen Automotive SPICE, ist in der Automotive-Industrie häufig Bestandteil der Anforderungen an die Entwicklungsprozesse. Auch das amerikanische Gegenstück zu SPICE, das Capability Maturity Model Integrated, kurz CMMI [CMMI_11], wird für eine Reifegradbewertung in vielen Industriezweigen herangezogen. CMMI wurde vom Software Engineering Institut (SEI) der Carnegie Mellon University in Pittsburgh entwickelt.

Aber nicht nur auf der technischen Ebene, auch auf der organisatorischen Ebene erhöht sich die Komplexität. Steigender Kostendruck sowie die Möglichkeit, Entwicklung in Niedriglohnländer zu verlagern, veranlassen Unternehmen, Entwicklungen in einem Netzwerk von verschiedenen Entwicklungsstandorten durchzuführen. Projekte mit mehreren, unterschiedlichen Entwicklungs-Teams an verschiedenen Standorten der Welt gehören nach Reif [Rei_12, S.207] in einer globalisierten Weltwirtschaft zum State-of-the-Art. Auch der organisatorische Einfluss diversester Stakeholder führt zu laufenden Änderungen und Erweiterungen über den gesamten Entwicklungszeitraum. All diese Herausforderungen können heutzutage nur mehr mit hochentwickelten Management-Prozessen und -Methoden bewältigt werden. Die Erfüllung dieser Ansprüche wird für viele Unternehmen zur existenziellen Herausforderung.

Die Einhaltung diverser Normen und Standards erfordert aber auch ein dringend nötiges Umdenken in der Vorgehensweise bei System-Entwicklungen. Der Schritt von der funktionalen Sicherheit hin zur System Sicherheit stellt auch einen besonders hohen Anspruch an die Entwicklungsteams in den nächsten Jahren. Dabei ist nicht nur die Engineering Ebene gefordert, noch mehr stellt dieses Problem gewisse Grunderwartungen an die Management-Ebene.

---

[6] "System Safety ist eine Ingenieursdisziplin zur Entwicklung sicherer Systeme und Produkte, bei der Safety explizit in das Design des Systems oder Produkts hineinfließt." (Übersetzung des Verfassers)

[7] Ein Prozessreifegrad formuliert die Anforderungen gemäß eines definierten Qualitätslevels, welcher auf der Prozessebene bewertet wird.

[8] SPICE war Projektname zur Entwicklung eines Prozessreifegradmodelles und stand zunächst für "Software Process Improvement and Capability Evaluation". Später wurde "Evaluation" in "Determination" abgeändert, wobei man das "E" im Wort D**e**termination (Fremdbestimmung) betonte und beibehielt. SPICE war die Initiative für die Entwicklung der ISO/IEC 15504, welche als internationaler Standard zur Bewertung von Prozessen dient. Erst in Verbindung mit einem Prozess-Referenzmodell ist die ISO/IEC 15504 in einem spezifischen Bereich anwendbar. Als Schwerpunkt für die Softwareentwicklung wird die ISO 12207 (Software Lifecycle Process) als Referenzmodell herangezogen.

1.1 Problemstellung und Motivation

In der Automotive-Industrie übernehmen immer mehr rechnergestützte Steuergeräte die Aufgabe von Steuerfunktionen. Die Realisierung in Software bietet dabei sehr viel Flexibilität in der Erfüllung spezieller Funktionalitäten. Dies hat zur Folge, dass die Anzahl der Steuergeräte im Auto und deren Verkoppelung untereinander immer rasanter zunehmen. Holzmann von den Nasa-Laboratories [Hol_07], sprach im Jahr 2007 von 50 Embedded Controllers und mehr als 600.000 Lines of Code in einem Fahrzeug der Oberklasse. Professor DDr. Broy von der TU-München, Experte für Automobil-Software, spricht 2012 von bis zu 100 Millionen Lines of Code, die in 70 bis 100 Mikroprozessoren [IEE_09] in einem Fahrzeug der Premium-Klasse implementiert sind. Die Business Research Firma Frost & Sullivan [IEE_09] prognostiziert für die nähere Zukunft 200 bis 300 Millionen Lines of Code. Eine derartige Komplexitätssteigerung hat zur Folge, dass sich das Fahrzeugverhalten in vielen Fällen immer schwieriger einschätzen lässt. Dadurch kommt es potentiell zu gefährlichen Situationen und Unfällen, welche auch auf Softwarefehler zurückgeführt werden können. Ein typisches Beispiel dafür ist ein im Jahre 2010 vermutetes Bremsproblem bei einem Fahrzeug (Toyota Prius Hybrid), bei dem der begründete Verdacht bestand, dass bei bestimmten Fahrbahnoberflächen die Bremsfunktion deaktiviert wird. Dieses Problem wurde von der NHTSA (National Highway Traffic Safety Administration) untersucht [NHTSA_10] und führte zu einem Rückruf und Software-Update der betroffenen Modelle.

Die Situation lässt sich folgendermaßen beschreiben (siehe Abbildung 1): Die steigende System-Komplexität und die Integration verschiedener Sub-Systeme sowie der nicht mehr zu verhindernde Einsatz von komplexer Software führen zu einer stark *ansteigenden Anzahl von möglichen inhärenten Systemzuständen*[9]. Diese sind in vielen Fällen schwer zu prognostizieren und können potentiell unerwartete Auswirkungen haben. Durch den zusätzlichen Einfluss von Software-, Hardware- oder Design-Fehlern in einzelnen Sub-Systemen bzw. das Zusammenspiel mehrerer Einzelfehler wird die Problematik noch verstärkt. Mangelhaftes System Design auf oberster Ebene und damit einhergehende zu starke Kopplung ("Tight Coupling") führen zu weiteren schwer zu prognostizierenden inhärenten Systemzuständen. Diese Systemzustände wurden bei der Konzeption schlicht nicht bedacht, und somit können solche Zustände gefährlich sein und Unfälle (Accidents) verursachen. Diese Accidents sind somit nicht die Folge einzelner Fehler, sondern ergeben sich durch dieses komplexe Zusammenspiel mehrerer Ursachen, so Holzmann [Hol_07]. Die amerikanische Federal Aviation Authority [FAA3_00] bestätigt diesen Sachverhalt mit „Seldom does a single hazard cause an accident. More often, an accident occurs as the result of a sequence of causes termed initiating and contributory hazards.“[10] Die Herausforderung besteht nun darin, gefährliche System-Zustände zu vermeiden.

[9] Inhärente Systemzustände sind im System innewohnende Zustände die an der Systemgrenze nicht sichtbar sind.

[10] "Selten verursacht ein einzelner Hazard einen Unfall. Öfter ist ein Unfall das Resultat einer Abfolge von Ursachen, die man als auslösende und beitragende Hazards bezeichnen kann." (Übersetzung des Verfassers)

Holzmann [Hol_07] hat sich mit diesem Sachverhalt beschäftigt und dabei noch auf einen zusätzlichen Aspekt hingewiesen, "[…] the probability of any one specific combination of failures will be extremely low, but as experience shows, this is precisely what leads to *major accidents*."[11] Das bedeutet konkret, dass derartige Systemzustände bzw. Ausfälle in ihrer Kombination äußerst selten oder sogar einzigartig vorkommen. Dies erschwert sowohl die zur Untersuchung notwendige Reproduzierbarkeit der Umstände, als auch die Analyse der Ursachen (Causal Factors). Auch General Motors [GMSH_12] bestätigt diesen Sachverhalt mit dem Argument, "System failure can come from the interaction of subsystem deficiencies which individually do not produce an end system failure but may do so in combination"[12].

Leveson [Lev_03], Professorin für Aeronautics und Astronoutics am Massachusetts Institute of Technology, spricht von folgenden und immer wichtiger werdenden Arten von Unfällen: "*system accidents* (arising from disfunctional interactions among components and not just component failures), software-related accidents, complex human decision-making, [...]"[13]. Die steigende Anzahl von Steuergeräten und deren Verkoppelung mit mechanischen Komponenten sowie die Interaktionsmöglichkeiten mit dem Fahrer erhöht die interaktive Komplexität. Die Anzahl der Systemzustände steigt damit ins Unermessliche, wodurch es auch zu Timing-Problemen und Race Conditions in der Software kommt, die sich sehr stark in System-Abstürzen und Hang-ups zeigen.

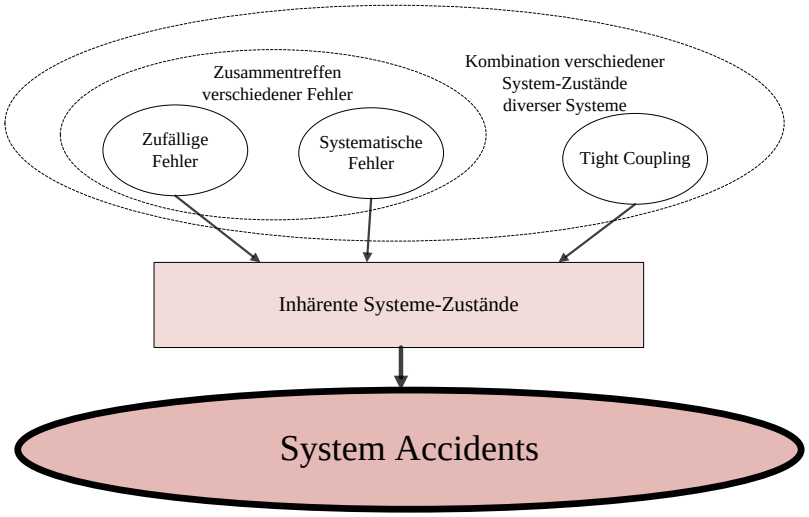


Abbildung 1: System Accidents und deren Ursachen

Leveson [Lev_03] differenziert in diesem Zusammenhang zwischen "Component Accidents" und "System Accidents". Component Accidents sind Unfälle, welche durch einzelne Hardware-Fehler

[11] "…die Wahrscheinlichkeit einer einzelnen spezifischen Kombination von Ausfällen wird äußerst gering sein, aber wie die Erfahrung zeigt, ist dies genau das, was zu schweren Unfällen führt." (Übersetzung des Verfassers)

[12] "Systemausfälle können von der Interaktion von Mängeln in Sub-Systemen kommen, die für sich alleine keine Systemausfälle produzieren, in deren Kombination jedoch schon." (Übersetzung des Verfassers)

[13] "Systemunfälle (die von fehlerhaften Interaktionen unter Komponenten, nicht nur Komponentenfehlern, verursacht werden), softwarebasierte Unfälle, komplexe menschliche Entscheidungen, [..]" (Übersetzung des Verfassers)

verursacht werden. Derartige Fehler können durch Reliability Engineering oder Fault Tolerance Prinzipien, aber auch durch viel Erfahrung, die im Laufe der Zeit gemacht wurde (Fly-Fix-Fly Approach nach Roland und Moriarty [RoMo_90, S.9]), immer weiter reduziert werden. Völlig anders verhält es sich jedoch bei Unfällen, die vorwiegend durch die hohe Anzahl an schwer oder nicht vorhersagbaren Systemzuständen verursacht werden. Durch die zunehmende Implementierung von Lösungen in Software sind solche Unfälle in ihrer Wichtigkeit stark gestiegen. Diesen Wandel bezeichnet Leveson als "*Change in the Nature of Accidents*" und benennt sie "*System Accidents*".

Damit ändert sich die Art der Unfälle. Um potentiell gefährliche Systemzustände prognostizieren zu können, die aus der spezifischen Kombination von Einzelereignissen entstehen, reichen die bekannten Verfahren, wie die Anwendung von Safety-Prinzipien, beispielsweise durch Redundanzen, nicht mehr aus. Ganz im Gegenteil, durch den Einsatz derartiger Prinzipien wird die Komplexität noch erhöht und schafft Potential für zusätzliche Fehler. Ebenso wenig können traditionelle Analyse-Methoden (zum Beispiel FMEA, FTA) diese Fehler entsprechend prognostizieren, oder können ausführliche Testaktivitäten die hohe Anzahl der Systemzustände vollständig abdecken. Derartige Vorgehen, Prinzipien und Methoden wirken sich am besten zur Vermeidung von Component-Accidents aus, aber nicht zur Vermeidung von System Accidents.

Leveson [Lev_03, S.1] begründet System Accidents mit nicht vorhersehbaren Systemzuständen, welche durch Fehler in den Anforderungen (Requirements) oder fehlende Anforderungen, also durch sogenannte "*Flawed Requirements*" verursacht werden. Häufig wird das *System-Umfeld falsch eingeschätzt* oder gar nicht beachtet, (teilweise auch darauf vergessen) – damit werden falsche Annahmen über die betrieblichen Zustände getroffen oder gar nicht berücksichtigt. Dadurch entstehen wiederum Lücken in den Requirements, die sich auf das Design auswirken. So kann eine Hazard-Identifikation sowie das Risk Assessment nicht ordnungsgemäß durchgeführt werden. Die Folge dieser Verkettung sind falsche Maßnahmen bei der Risiko-Reduktion.

Es stellt sich nun die Frage, ob Safety-Normen, wie beispielsweise die ISO 26262, diese Situation ausreichend berücksichtigen und hier Abhilfe schaffen können.

Eine Grundproblematik der ISO 26262 beginnt bereits bei der Item-Definition der ISO 26262 [ISO26_11, Teil 3, Kapitel 5]. Die Norm bezeichnet eine zu betrachtende Einheit, wie zum Beispiel ein Steuergerät, als Item. Ziel der Norm ist, das Item zu spezifizieren, die Abhängigkeiten und Interaktionen mit anderen Items und mit der Item-Umgebung anderer Technologien zu definieren. Dabei wird sehr viel Information benötigt, wie beispielsweise die Zielumgebung oder der Einsatzbereich, die in der Projektinitialisierungsphase meistens nicht bekannt sind. Die Norm versucht dieses Problem anhand von Annahmen zu lösen [ISO26_11, Teil 3, Kapitel 5.4.2], “The boundary of the item, its interfaces, and the assumptions concerning its interaction with other items and elements, shall be defined …"[14].

Safety-Normen liefern kein dafür erforderliches *systematisches Vorgehen*, welches das System-Umfeld richtig identifiziert und eine entsprechende Rückverfolgbarkeit aufweist. Dadurch ergeben

[14] "Die Grenze der Betrachtungseinheit, ihre Schnittstellen und die Annahmen bzgl. der Interaktionen mit anderen Betrachtungseinheiten und Elementen sollen definiert werden" (Übersetzung des Verfassers)

sich häufig mangelhafte und/oder fehlerhafte Spezifikationen, auch "Flawed Requirements" genannt. Wegen der hohen System-Komplexität der elektrischen/elektronischen Systeme in einem Fahrzeug wird diese Problematik durch nicht prognostizierbare Systemzustände noch verstärkt oder kann sich sogar vervielfachen. Auch die ISO 26262 spricht von einem "trend of increasing complexity, software content and mechatronic implementation"[15] und den damit einhergehenden "increasing risks from systematic failures and random hardware failures"[16] [ISO26_11, Teil 1, Introduction].

Im Zusammenhang mit Risiken aus systematischen oder zufälligen Ausfällen taucht im Raffinerie-Bereich (speziell im Off-Shore-Bereich bei Öl- und Gas-Plattformen) der Begriff des inhärent sicheren Designs auf. Kletz und Amyotte beschreiben in [KlAm_10, S.19] „The essence of the inherently safer approach to plant design is the avoidance of hazards rather than their control by added protective equipment.“[17] So ein inhärent sicherer Ansatz verspricht im Zusammenhang mit dem gesamtheitlichen Denken eine noch weitergehende Verminderung des System-Gesamtrisikos, da Gefahrenquellen von vornherein *vermieden* werden sollen. Kletz und Amyotte schränken jedoch diesen Begriff auf „[…] ways of eliminating large inventories of hazardous materials and hazardous equipment and operations […]”[18] ein. Zum anderen taucht der Begriff der inhärent sicheren Konstruktion in der ISO 12100 [ISO12_11] auf. Darunter versteht die ISO 12100 eine „Schutzmaßnahme, die entweder Gefährdungen beseitigt oder die mit den Gefährdungen verbundenen Risiken vermindert, indem ohne Anwendung von trennenden oder nichttrennenden Schutzeinrichtungen die Konstruktions- oder Betriebseigenschaften der Maschine verändert werden“. Das heißt, auch hier geht es darum, *Vermeidung vor Kontrolle* zu setzen. Diese Ideen über inhärent sichere Ansätze beschränken sich aber nur auf die mechanischen Einflussfaktoren (z.B. Auswirkungen durch gefährliche Materialien) bzw. die mechanische Konstruktion. Speziell im Hinblick auf moderne Elektronik und die damit zwangsläufig verbundene Software, ist der Begriff der inhärenten Sicherheit noch gänzlich unbelegt.

Doch im Sinne von System Safety ist das noch lange nicht alles. Bereits 1986 hat Jerome Lederer festgestellt, dass System Safety über die technischen Aspekte weit hinausgeht und auch folgende Betrachtungsweisen inkludiert (zitiert nach Leveson [Lev_11, S. 29]):

> "[…] attitudes and motivation of designers and production people, employee/management rapport, the relation of industrial associations among themselves and with government, human factors in supervision and quality control, documentation on the interfaces of industrial and public safety with design and operations, the interest and attitudes of top management, the effects of the legal system on accident investigations and exchange of information, the certification of

[15] "Trend von zunehmender Komplexität, zunehmender Software und mechatronischen Lösungen" (Übersetzung des Verfassers)

[16] "zunehmende Risiken systematischer Ausfälle und zufälliger Hardware Ausfälle" (Übersetzung des Verfassers)

[17] „Die Essenz eines inhärent sichereren Ansatzes für Raffinerie-Design, ist die Vermeidung von Gefährdungen, anstatt diese durch hinzugefügte Sicherheitseinrichtungen zu kontrollieren.“ (Übersetzung des Verfassers)

[18] „Wege zur Eliminierung von großen Beständen von gefährlichen Materialien, gefährlichem Equipment und Betrieb.“ (Übersetzung des Verfassers)

> critical workers, political considerations, resources, public sentiment and many other non-technical but vital influences on the attainment of an acceptable level of risk control. These non-technical aspects of system safety cannot be ignored."[19]

Leveson schreibt dazu im Jahre 2011: "Too often, however, these non-technical aspects are ignored"[20] [Lev_11, S. 29], und spricht damit dieselbe Problematik an. Dieser Bezug verdeutlicht, dass die genannten Aspekte nichts von ihrer Aktualität eingebüßt haben und bei allen Analyseprozessen mitzubedenken sind. Ansatzweise geht auch der MIL-STD 882E [MIL_05] auf diese Problematik ein, er betont bei der Definition von System Safety das Zusammenspiel von Safety-Management und Safety-Engineering.

Um den ursprünglichen Ansatz der inhärenten Sicherheit mit dem der System Safety zu verbinden, bedarf es also einer ganzheitlichen Kombination von Management- und Engineering-Aspekten über alle Disziplinen hinweg. Dieser vielversprechende Ansatz wird in dieser Arbeit mit „Inherent System Safety Approach" bezeichnet.

1.2 Forschungsfragen

Die zentrale Forschungsfrage der vorliegenden Arbeit ist, ob es ein Vorgehensmodell geben kann, sodass die zuvor adressierten Probleme, speziell im Hinblick auf System Accidents, effektiv lösbar werden. Dazu ergeben sich folgende untergeordnete Fragestellungen:

1. Wie muss dieses Vorgehensmodell aufgebaut sein, um ein „inhärent sicheres Design" zu unterstützen?
2. Inwieweit kann so ein inhärent sicheres Design „System Accidents" verhindern?
3. Welche Methoden muss das Vorgehensmodell zur Verfügung stellen, um „Flawed Requirements" zu verhindern bzw. aufzudecken?
4. Welche Aspekte muss das Vorgehensmodell beinhalten, um im Sinne eines „inhärent sicheren Ansatzes" die Basis für Planung und Realisierung eines entsprechend inhärent sicheren Designs zu legen?
5. Welche Anforderungen muss das Vorgehensmodell stellen, damit die Grundlage für einen Sicherheitsnachweis entsprechend der „inhärenten System Safety" gegeben ist?

---

[19] "[…] Einstellungen und Motivation der Mitarbeiter in Design und Produktion, Mitarbeiter/Management Verhältnis, das Verhältnis von Berufsverbänden untereinander und mit der Regierung, menschliche Einflüsse in der Überwachung und Qualitätslenkung, Dokumentation der Schnittstellen zwischen industriellen und öffentlichen Sicherheitseinrichtungen mit Design und Betrieb, das Interesse und die Einstellung vom Top Management, die Auswirkungen des Rechtssystems auf Unfallanalysen und Informationsaustausch, die Zertifizierung von Schlüsselarbeitskräften, politische Überlegungen, Ressourcen, die öffentliche Meinung und viele andere nicht-technische aber wesentliche Einflüsse auf die Erreichung eines akzeptierbaren Niveaus an Risikosteuerung. Diese nicht-technischen Aspekte von System Safety dürfen nicht ignoriert werden." (Übersetzung des Verfassers)

[20] "zu oft jedoch werden diese nicht-technischen Aspekte ignoriert" (Übersetzung des Verfassers)

1.3 Zielsetzung und Neuigkeitswert?

Ziel der Arbeit ist die Entwicklung eines Vorgehensmodells zur Konzeption und Entwicklung von sicherheitskritischen technischen Systemen. Mit dem erarbeiteten Vorgehen sollen „System Accidents“ vermieden werden. Das Vorgehensmodell soll über die funktionale Sicherheit hinaus die wichtigsten Ansätze der Systemsicherheit erfüllen und ein *inhärent sicheres System Design* (Inherent Safe System Design) mit entsprechender *Planung der Realisierung* bewirken. Des Weiteren soll das Vorgehensmodell die Struktur für den Sicherheitsnachweis (Safety Case) vorgeben, um die Nachweisführung hinsichtlich der inhärenten Systemsicherheit zu ermöglichen. Dieser „*Inherent System Safety Approach*“ soll als systematisch-methodische Vorgehensweise primär „Flawed Requirements“ verhindern und Hazards vermeiden helfen, wodurch die schon genannten System Accidents reduziert werden.

Das Fundament für das Vorgehensmodell liefert der von Tschürtz [HTS_09] entwickelte „Integrative Safety Prozess“, kurz als ISaPro® bezeichnet (siehe dazu Kapitel 2.3). Der ISaPro® stellt ein generisches Rahmenwerk einer Prozesslandschaft zur Verfügung, mit den dafür erforderlichen Prozessen der Disziplinen Projektmanagement, Safety und Engineering. Der ISaPro® dient der Strukturierung komplexer Projekte im sicherheitskritischen Bereich.

Das Vorgehensmodell konzentriert sich dabei auf die ersten beiden Phasen des ISaPro®, das sind die Vor-Projektphase und die Planungphase. Das Vorgehensmodell bietet die Möglichkeit, disziplinenübergreifende Thematiken, in einem innovativen Modell abzubilden. Damit wird das Vorgehen im Projektverlauf systematisiert und synchronisiert, sowie ein ganzheitlicher Systemansatz und ein ganzheitlicher Überblick über das gesamte Projekt geschaffen. Problematische Schnittstellen können auf diese Weise in sehr frühen Projektphasen identifiziert und entsprechende Gegenmaßnahmen gesetzt werden.

Ein derartiges Vorgehensmodell, welches das Ziel eines „*Inherent System Safety Approach*“ verfolgt, mit einer detaillierten Ausprägung auf ein *systematisch-methodisches Vorgehen* mit *speziell entwickelten und erweiterten Analyse-Methoden* prägt den Neuigkeitswert dieser Arbeit. Die Evaluierung wird anhand von einem durchgeführten Anwendungsbeispiel gezeigt. Dieses Vorgehensmodell wird in dieser Arbeit mit „*Safety-Vorgehensmodell*“ bezeichnet.

1.4 Gliederung und Aufbau der Arbeit

An dieses Einführungskapitel schließt das zweite Kapitel mit dem aktuellen Stand der Wissenschaft und Technik an. Darin werden die wichtigsten Definitionen erläutert und die für diese Arbeit relevanten Safety-Normen beschrieben. Darüber hinaus wird das, für das Safety-Vorgehensmodell, strukturgebende Rahmenwerk „Integrativer Safety Prozess“, kurz ISaPro®, dargestellt. Die dazugehörigen Support-Prozesse und deren Bezug zu Safety werden anschließend beschrieben. Die klassischen Analyse-Methoden und die Goal Structuring Notation sind nachfolgend dargestellt. Mit der Abgrenzung zu anderen Vorgehensmodellen, aus dem industriellen und dem wissenschafltichen Bereich, wird dieses Kapitel abgeschlossen.

Im Kapitel drei wird das mustergültige neue Safety-Vorgehensmodell herausgearbeitet. Dabei werden der systematisch-methodische Ablauf in jeder Phase sowie die Wechselwirkungen der einzelnen Prozessgebiete genau dargestellt. Der Fokus liegt dabei in der Vor-Projektphase und in der Projektplanungsphase. Auf die Projekt-Realisierungsphase wird dabei nicht näher eingegangen, da diese Phasen nicht im Fokus dieser Arbeit liegen.

Die für das Safety-Vorgehensmodell spezifischen Modelle und Methoden werden im Kapitel 4 beschrieben. Die „Erweiterten Fehlerkette“ untermauert das Safety-Vorgehensmodell theoretisch, um Fehlermechanismen und deren Kausalbeziehungen besser verstehen zu können. Um den Ansatz von „Inherent System Safety“ sichtbar zu machen, ist es wichtig das zu entwickelnde System in seinem Gesamtkontext erfassen zu können. Dafür wurde eine graphisch basierte Darstellungsform, das Shell-Modell, entwickelt. Erweiterte und neu entwickelte Analyse-Methoden, die das neue Vorgehen benötigt, schließen an das Shell-Modell an. Das Safety Cases Pattern, eine Vorlage für den Sicherheitsnachweis, basierend auf der Goal Structuring Notation, schließt dieses Kapitel ab.

Ein Anwendungsbeispiel im Kapitel 5 konkretisiert und evaluiert das Vorgehensmodell. Es werden dabei nur die wesentlichen Prozessschritte näher betrachtet, damit der Ablauf überschaubar bleibt. Anhand dieses Beispiels soll auch veranschaulicht werden, wie wichtig ein systematisch-methodisches Vorgehen ist, um der Komplexität der Safety-Thematik entsprechen zu können. Die Notwendigkeit von speziellen Safety-Analyse Methoden und deren systematischer Einsatz zur Vermeidung von System Accidents wird hier deutlich sichtbar. Dieses Kapitel 5 ist wie Kapitel drei nach der Struktur des Vorgehensmodells aufgebaut. Um ein ständiges Nachlesen im Theorieteil (Kapitel 3) zu vermeiden, sind alle abzuarbeiteten Prozessschritte in jedem Prozess, in Kapitel 5 nochmals kurz beschrieben.

Im abschließenden Kapitel 6 werden die wesentlichen Erkenntnisse aus dieser Arbeit zusammengefasst und der Ansatz des „Inherent System Safety Approach“ eingehend diskutiert. Eine kurze Darstellung über den Nutzen des Safety-Vorgehensmodells schließt die Dissertation ab.

Es sei an dieser Stelle noch erwähnt, dass die behandelten Fachbegriffe - soweit sinnvoll - auch ins Deutsche übersetzt wurden. Allgemein werden die Fachausdrücke jedoch in der fachlich-gewohnten Umgangssprache verwendet. Zur Förderung eines einheitlichen Verständnisses werden komplexere Begriffe in den einzelnen Kapiteln in der Fußzeile kurz erläutert beziehungsweise im Glossar (siehe Anhang) dargestellt.

2. Stand der Wissenschaft und Technik

Im Einführungskapitel wurden bereits einige Begriffe aus der Safety-Welt erwähnt sowie auf aktuelle Safety-Normen verwiesen. Dieses Kapitel erläutert diese und weitere wichtigste Begriffe und Definitionen. Darüber hinaus werden, die für diese Arbeit, relevanten Safety-Normen näher beschrieben und deren Zugang zur inhärenten Systemsicherheit kurz diskutiert. Damit sind die Grundelemente für das Rahmenwerk des ISaPro® geschaffen, welches nachfolgend vorgestellt ist. Der ISaPro® und die dazugehörigen Support-Prozesse, welche darauffolgend dargestellt werden, sind das strukturgebendes Rahmenwerk für das Safety-Vorgehensmodell. Dieses wird durch die klassischen Analyse-Methoden untermauert. Die Safety Cases Methode von Kelly [Kel_98] dokumentiert eine mögliche Darstellungsform für den Sicherheitsnachweis von technischen Systemen. Die Abgrenzung zu anderen Vorgehensmodellen schließt dieses Kapitel ab.

2.1 Begriffe und Definitionen

Zu den wichtigsten Begriffen im Safety-Bereich gehören auch die Grundbegriffe der technischen Systemtheorie, da sie für die Identifikation von Gefährdungen (Hazards) extrem wichtig sind. Um Safety-Analysen erfolgreich durchführen zu können, müssen die Systemkonzepte und Systemgrenzen klar definiert sein. Anschließend an die Systemtheorie wird schrittweise in die Safety-Thematik eingeführt, beginnend mit der klassischen Fehlerkette von Laprie, über den Dependability-Tree und dem Hazard-Triangle bis hin zur funktionalen- und Systemsicherheit.

2.1.1 Grundbegriffe der technischen Systemtheorie

Safety-Normen fordern grundsätzlich eine klare Systemabgrenzung, da Hazards nur an der betrachteten Systemgrenze auftreten und auch nur dort identifiziert werden können. Die IEC 61508 fordert ein definiertes und dokumentiertes Konzept [IEC08_10, Teil 1, S.25-26] des zu kontrollierenden Systems und lässt dabei viel Freiraum bei der Definition. Die ISO 26262 verlangt die klare Definition der zu betrachtenden Einheit, im ISO 26262 Kontext als „Item“ bezeichnet, welches beispielsweise ein Steuergerät eines Autos darstellt.

Diese Arbeit beschäftigt sich mit technischen Systemen, die immer in ein größeres Ganzes eingebettet sind, und direkt oder indirekt Schnittstellen mit Menschen aufweisen. Die nachfolgende Definition von Schildt [Sch_98] entspricht dem üblichen Verständnis technischer Systeme:

> „Ein System besteht aus einer Menge von Elementen und einer Menge von Relationen, die zwischen den Elementen besteht. Jedes System ist charakterisiert durch seine Elemente, die Relationen zwischen ihnen und die Systemgrenze gegenüber der Umgebung."

Eine solche Systemdefinition spiegelt jedoch laut Ropohl [Rop_09] nur *einen* Aspekt von Systemen wieder. Es handelt sich hier um eine *strukturale* Sichtweise auf ein System, nämlich dass ein System aus einer Menge interagierender Elemente besteht. Ropohl [Rop_09, S.76] fügt neben diesem strukturalen Systemkonzept noch zwei weitere Sichtweisen auf ein System hinzu: Das *funktionale* Systemkonzept und das *hierarchische* Systemkonzept.

Beim funktionalen Systemkonzept wird ein System als eine Black-Box angesehen, welche Inputs hat, diese Inputs verarbeitet, einen *internen Zustand* besitzt und Outputs generiert. Diese Sichtweise entspricht der typischen Sichtweise auf ein System von außen, also als Benutzer, der sich Dienstleitungen oder Services erwartet, für den/die die interne Struktur aber nicht relevant ist.

Schließlich gibt es noch das hierarchische Systemkonzept, bei dem die Betonung darauf liegt, dass man Systeme auf vielen Ebenen betrachten kann, also *dass ein System immer Teil eines größeren Ganzen ist* (dem Supersystem) und dass Systeme selbst auf niedrigerer Ebene wieder Systeme (Subsysteme) enthalten.

Ropohl [Rop_09, S. 77] betont die Bedeutung dieser drei Systemkonzepte, die sich gegenseitig ergänzen: „Diese drei Systemkonzepte schließen einander keineswegs aus, sondern können leicht miteinander verbunden werden. Man beginnt etwa mit der Funktion eines Untersuchungsgegenstandes, fragt dann nach dem inneren Aufbau, aus dem die Funktion zu erklären ist, und zieht schließlich den größeren Zusammenhang in Betracht, in dem der Untersuchungsgegenstand angesiedelt ist." Diese drei Sichtweisen, in Abbildung 2 zusammengefasst, sind auch für die vorliegende Arbeit von zentraler Bedeutung. Das in Kapitel 3 entwickelte und vorgeschlagene Safety-Vorgehensmodell wird Methoden beinhalten, um all diese Aspekte des betrachteten Systems abzudecken.

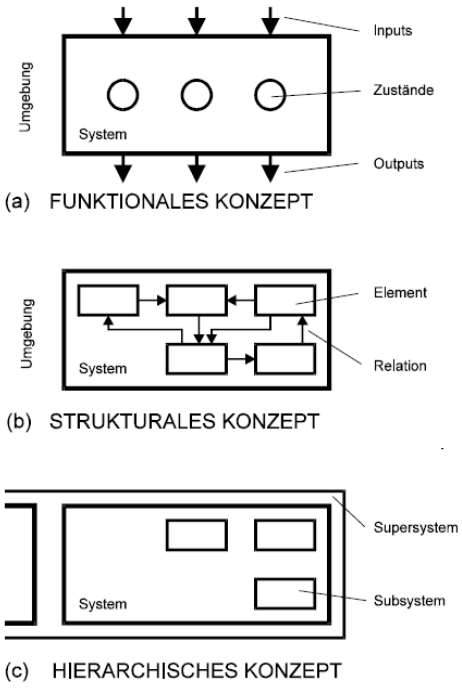


Abbildung 2: Konzepte der Systemtheorie ([Rop_09, S. 76]

Bei der technischen Systementwicklung ist nun eine klare Abgrenzung des zu betrachtenden Systems von seiner Umgebung wichtig. Die Schnittstellen vom betrachteten System zu seiner Umgebung werden als externe Schnittstellen bezeichnet. Diese klare Abgrenzung sowie die Identifikation der externen Schnittstellen sind nicht immer eindeutig und gestalten sich meist komplex.

Um ein System und seine Schnittstellen zur Umgebung klar zu definieren, wird in der ISO 26262, wie oben kurz erwähnt, der Begriff „Item“ verwendet. Die Norm verlangt eine klare Definition des Items, um dem Gesamthersteller in der Praxis die Auswahl potentiellen Lieferanten erleichtern zu können. Die ISO 26262 beschreibt nicht, wie die System-Definition bzw. System-Abgrenzung zu erfolgen hat, sie spricht in [ISO26_11, Teil 3, Kapitel 5] lediglich von zwei wesentlichen Zielen, die bei der Item-Definition erreicht werden müssen. Einerseits soll das Item definiert und beschrieben sein, andererseits soll ein angemessenes Verständnis dafür aufgebaut werden, um die entsprechenden Aktivitäten im System Safety Lifecycle planen zu können. Um eine angemessene Item-Definition durchführen zu können, müssen die funktionalen Anforderungen und die Sicherheitsziele (Safety Goals) vom Kunden zur Verfügung gestellt werden sowie der genaue Einsatzbereich definiert sein.

Schließlich spielt im Zuge dieser Arbeit noch die Komplexität von Systemen eine wesentliche Rolle. Komplexität ist jedoch ein schwer zu begreifendes Konzept, da die Erfassung von Komplexität letztlich subjektiv ist. Was für einen Betrachter hoch komplex ist, kann für einen anderen Betrachter als wenig komplex gelten. Es hat sich jedoch als Faustregel herausgestellt, dass für die menschlichen Betrachter eine Komplexitätsgrenze bei einer Anzahl von 7±2 Komponenten liegt. Fünf (7-2) oder weniger Komponenten werden als trivial angesehen, neun (7+2) oder mehr als komplex, so beschreibt es Schäuffele [ScZu_13, S.115]. Diese Sicht alleine ist jedoch für Systemkomplexität

nicht ausreichend. Deshalb gibt es auch Versuche, Komplexität zu quantifizieren, die darauf beruhen, bestimmte Einflussfaktoren zu betrachten, die die empfundene Komplexität beeinflussen: Im Falle von Systemen lässt sich die Komplexität beispielsweise anhand (1) der Anzahl der System-Elemente, (2) der Anzahl aller möglichen Beziehungen zwischen den Elementen, (3) der Verschiedenartigkeit der Beziehungen, (4) der Entwicklung dieser drei Faktoren im Zeitverlauf, (5) der Anzahl der System-Zustände beschreiben. Wenn in weiterer Folge also von der Komplexität eines Systems gesprochen wird, sind diese Kriterien als grundlegend zu erachten.

2.1.2 Klassische Fehlerkette

Avizienis und Laprie [AvLa_04] definieren eine Fehlerkette, die beschreibt, wie es zu Ausfällen in Systemen kommt. Diese Fehlerkette sowie auch die zugehörige Terminologie ist mittlerweile sinngemäß in die maßgeblichen Standards ([IEC08_10] und [ISO26_11]) übernommen worden.

Die Fehlerkette beginnt mit einem „Fault“ (Fehler), welcher zu einem unerwünschten und fehlerhaften Systemzustand („Error“ bzw. „Fehlzustand“) führen kann. Dieser fehlerhafte Systemzustand kann nun in weiterer Folge zu einem „Failure“ (Ausfall) des Systems führen. Der Ausfall ist das Ereignis, welches an der definierten Systemgrenze wahrnehmbar ist – Fehler und Fehlerzustände sind nur systemintern und haben nach außen keine erkennbare Wirkung. Es sei noch bemerkt, dass der Begriff Ausfall hier für jegliche Abweichung von der gewünschten Funktionalität (bzw. von den Requirements) steht, die nach außen sichtbar ist. Der Begriff Ausfall steht also nicht unbedingt für einen *Totalausfall* des Systems, sondern gleichermaßen für eine Fehlfunktion („Malfunction“).

Die Wirkmechanismen der ursprünglichen Fehlerkette aus [AvLa_04] sind folgende: Ein Fehler wird durch etwas verursacht. Dieser Fehler kann dann im System aktiviert werden (wodurch er einen Fehlzustand im System hervorruft) und dieser Fehlzustand kann sich so an die Systemgrenze propagieren, dass er zu einem Ausfall führt. Diese Fehlerkette und deren Wirkmechanismen sind in Abbildung 3 dargestellt.

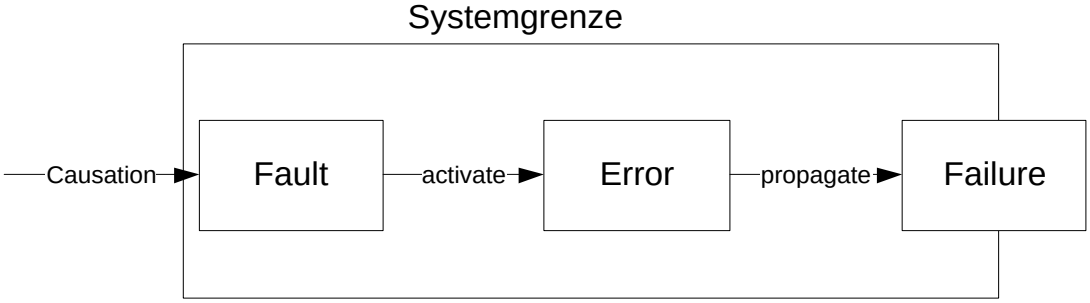


Abbildung 3: Fehler-Kette nach Laprie [AvLa_04]

Wie erwähnt ist erst der Ausfall an der Systemgrenze sichtbar. Daraus folgt, dass ein- und dasselbe, je nach Definition und Lage der Systemgrenze, entweder ein Fehler (Fault) oder ein Ausfall (Failure) sein kann. Ein Ausfall in einem Subsystem kann sich beispielsweise auf Systemebene lediglich als ein Fehler oder Fehlzustand äußern. Ein Beispiel soll diesen Vorgang zusätzlich illustrieren: Ein

Software-Entwickler verursacht einen Programmierfehler in einer Sortierroutine für das Navigationsgerät in einem Fahrzeug. Dieser hat zur Folge, dass die vom Benutzer gespeicherten Destinationen nicht korrekt sortiert werden. Die Software hat somit von Beginn an einen Fehler (fault). Nun kann es aber sein, dass der Benutzer dieses Feature in der Software nie nutzt, da er keine Destinationen abspeichert. Somit wird dieser Fehler niemals ausgelöst und führt auch zu keinem Error (Fehlzustand). Aber im Fall einer Nutzung wird die Liste vorerst intern falsch sortiert, was ein Fehlzustand ist. Es kann nun sein, dass sich der Benutzer die Liste niemals anzeigen lässt, sodass dieser Fehlzustand zu keinem Ausfall bzw. keiner Fehlfunktion führt. Lässt er sich die Liste jedoch anzeigen, so ist diese falsch sortiert, was einem Ausfall bzw. einer Fehlfunktion – einer Nichterfüllung der gewünschten Funktionalität – gleichkommt.

Ein Fehler *kann*, aber *muss nicht*, zu einem Fehlzustand führen, d.h. es kann sein dass der Fehler gar nicht aktiviert wird. Genauso muss ein Fehlzustand nicht unbedingt zu einem Ausfall/einer Fehlfunktion führen. Ob ein Fehlzustand, also zu einem Ausfall propagiert wird, oder nicht, hängt davon ab, *wie* ein System benutzt wird.

2.1.3 Dependability

„Dependability“, wie in Avizienis und Laprie 2004 definiert, ist ein übergeordnetes Konzept der Verlässlichkeit von Systemen, welches unter anderem Safety beinhaltet und den Begriff in einen universalen Kontext stellt. Avizienis und Laprie [AvLa_04] haben eine Basis geschaffen, die die Terminologie und die Zusammenhänge klar und vereinfacht darstellt. Dependability (Verlässlichkeit) wird anhand ihrer Attribute, Threats (Gefahren) und Means (Mittel) in einer Baumstruktur, dem „Dependability Tree“ (Abbildung 4), abgebildet.

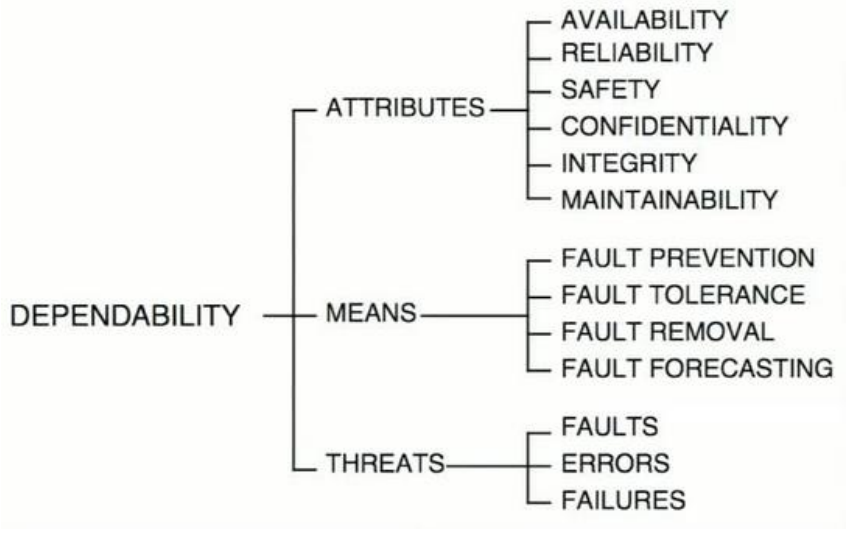


Abbildung 4: Dependability Tree [AvLa_04]

Safety – der Schwerpunkt dieser Arbeit – ist im Dependability Tree als eines der Attribute zu finden. Diese *Attribute* beschreiben die Eigenschaften von Dependability. Sie tragen alle zur Erreichung von Dependability bei, wobei deren Priorisierung vom jeweiligen System abhängt. In einem System

kann beispielsweise Reliability wichtiger als Availability sein, in einem anderen umgekehrt. Die *Means*, also die Mittel und Wege um Dependability zu erreichen, stellen die Realisierungsmöglichkeiten dar. Das in dieser Arbeit beschriebene Ableiten geeigneter Prozesse um Fehler (Faults) zu verhindern, kann primär als Fault Prevention gesehen werden, wobei geeignete Tests – die auch in Prozessen vorgesehen sind – auf Fault Removal abzielen. Die *Threats* lassen sich als die Faktoren bestimmen, die der Erreichung von Dependability abträglich sind: Faults, Errors und Failures, wie im Kapitel 2.1.2 beschrieben.

2.1.4 Hazard Triangle

Ein detailliertes Modell für die Beschreibung eines Hazards sowie für dessen Übergang zu einem Accident beschreibt Ericson [Eric2_05, S.17]. Ohne auf die Details dieses Modells einzugehen, soll hier ein kurzer Abriss davon wiedergegeben werden.

Ein Hazard ist gekennzeichnet durch drei Eigenschaften:

- Das „Hazard Element“, also die Entität, die den Hazard hervorruft. Dies kann typischerweise ein beliebiger Energieträger sein, der sich potentiell unkontrolliert entladen kann. Ein Hazard-Element ist beispielsweise eine offene Hochspannungsleitung, die eine Gefährdung für Menschen darstellt.
- Die „Initiating Mechanisms“, also die Ereignisse, die (möglicherweise in einer bestimmten Reihenfolge) eintreten müssen, um den Hazard zu einem Unfall zu transformieren. Ein Initiating Mechanism ist beispielsweise eine Berührung der offenen Hochspannungsleitung vom Menschen.
- Die „Targets/Threats“, also die Entitäten, die durch den Hazard bzw. den daraus folgenden Unfall potentiell gefährdet sind sowie die Bedrohung, die für diese Entitäten zu befürchten sind. Targets sind also in der Regel Lebewesen und Umwelt, Threats sind beispielsweise Verletzung oder Tod.

Das Analysieren von Hazards mithilfe dieser Eigenschaften ermöglicht, die Hazards besser zu verstehen sowie auch deren Risiko besser einschätzen zu können. Aus diesem Grund wird in dem in dieser Arbeit vorgestellten Safety-Vorgehensmodell bei der Hazard-Analyse großer Wert auf die Bestimmung dieser Eigenschaften gelegt. So ist das „Hazard Element“ die potentiell gefährliche Ressource, die vielleicht schon beim Entwurf eines Systems durch weniger gefährliche Alternativen ersetzt werden kann. In diesem Fall spricht diese Arbeit von „Hazard Avoidance“ Die „Targets/Threats“ bestimmen die Schwere des Hazards bzw. des daraus folgenden Unfalls, wobei die „Initiating Mechanisms“ dessen Wahrscheinlichkeit bestimmen. Diese Eigenschaften, in einem Dreieck dargestellt (siehe Abbildung 5), ergeben nach Clifton Ericson das sogenannte „Hazard Triangle“.

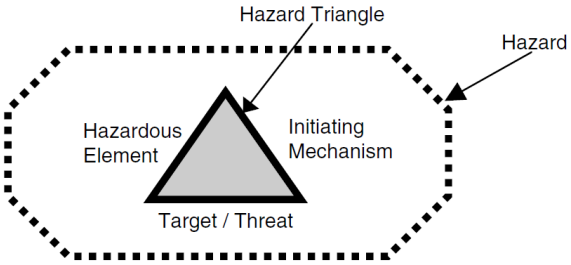


Abbildung 5: Hazard Triangle [Eric2_05, S. 17]

2.1.5 Fokus System Safety

Mit Sicherheit im Sinne von Safety ist in dieser Arbeit grundsätzlich immer die Vermeidung von Schaden an Menschen und Umwelt gemeint, wie es auch der MIL-STD 882E [MIL_05] definiert. Der englische Begriff „Security“ wird zwar ebenso mit „Sicherheit“ übersetzt, bedeutet aber etwas anderes. Security beschäftigt sich mit dem Schutz von Infrastruktur und Systemen vor unerlaubten Zugriff von außen, darauf soll in der vorliegenden Arbeit nicht eingegangen werden. Nachdem die begriffliche Unterscheidung zwischen Safety und Security auf Deutsch nicht gegeben ist, wird in weiterer Folge der englische Begriff „Safety“ verwendet.

Es gibt unzählige Definitionen von Safety, die jedoch inhaltlich großteils überlappend sind. Teilweise wird in die Definitionen, die immer auf die Vermeidung von Schaden an Menschen und Umwelt abzielen, auch die Vermeidung von finanziellem Schaden inkludiert. Eine schlüssige Definition beschreibt beispielsweise das British Railway Board [BRB_93]: “The avoidance of death, injury or poor health to customers, employees, contractors and the general public; also avoidance of damage to property and the environment […]”[21]. Die IEC 61508 definiert Safety als “freedom from unacceptable risk of harm”[22], die ISO 26262 definiert Safety sehr ähnlich, als “absence of unreasonable risk”[23].

Safety ist stark verknüpft mit dem Begriff des Hazards (siehe auch Kapitel 2.1.4), auf Deutsch der Gefährdung. Mangelnde Sicherheit beruht immer auf einem oder mehreren Hazards. Das sind Zustände oder Ereignisse[24], die sich an der Systemgrenze des zu betrachtenden Systems manifestieren, also von außerhalb des Systems wahrnehmbar sind. Um ausreichende Sicherheit zu erreichen, müssen Hazards erkannt und eliminiert werden.

[21] "Die Vermeidung von Tod, Verletzung oder Gesundheitsgefährdung von KundInnen, Angestellten, VertragspartnerInnen und der Öffentlichkeit, sowie die Vermeidung von Schaden an Eigentum und der Umwelt." (Übersetzung des Verfassers)

[22] "Freiheit von unvertretbaren Risiken" (Übersetzung in der Norm)

[23] "Abwesenheit von nicht vertretbarem Risiko" (Übersetzung des Verfassers)

[24] Manchmal werden Hazards ausschließlich als Zustände oder ausschließlich als Ereignisse definiert. Hier wird jedoch bewusst beides in die Definition übernommen. Ein gefährlicher Zustand wäre beispielsweise "Öl auf der Fahrbahn", ein gefährliches Ereignis ein Blitzeinschlag. Beides muss nicht zu einem Unfall führen, stellt aber einen Hazard dar.

Der MIL-Standard[25] definiert Hazard ganz klar und eindeutig als potentielle Gefahrenquelle mit dem Potential Schaden am System, an der Umwelt, an Personen oder Einrichtungen und Besitz zu verursachen. Die ISO 26262[26] definiert einen Hazard als potentielle Quelle für einen Schaden, welcher durch ein Fehlverhalten (Malfunctioning) eines „Items“ (i.e. des betrachteten Systems) verursacht wurde. Ein Fehlverhalten des Systems, also eine Malfunction, muss aber nicht unbedingt zu einem Hazard führen. Es gibt auch Fehlverhalten oder sogar Ausfälle, die weder einen Hazard darstellen, noch zu einem Unfall führen können.

Ein Hazard birgt somit das Potential Schaden anzurichten, also einen Unfall (Accident) auszulösen. Der Unfall selbst ist dann das nicht geplante bzw. nicht gewollte Ereignis oder die Serie von Ereignissen, welche Personen, Umwelt, anderen Systemen oder dem System selbst Schaden zufügen. Daraus ist ersichtlich, dass der Unfall letztlich das zentrale Ereignis ist, das es zu vermeiden gilt. Um dieses Ziel zu erreichen, muss man jedoch die Kausalkette der Unfallursachen identifizieren und diese Ursachen, also Hazards, Fehlfunktionen, Ausfälle etc. eliminieren bzw. reduzieren.

Der Begriff Safety steht oft nicht alleine, sondern wird mit anderen Begriffen kombiniert. Sehr häufig anzutreffen sind „System Safety“ und „Functional Safety“. Während der Begriff „Safety“ alleine „Sicherheit“ im Allgemeinen bedeutet, und noch keine Einschränkung vornimmt, worauf man sich bezieht, so wird mit *System Safety und Functional Safety* klargemacht, dass es sich um *technische Systeme* handelt. Damit werden Aspekte, wie beispielsweise Arbeitssicherheit etc., aus dem Bedeutungsumfang entfernt.

Functional Safety bezieht sich in der IEC 6108 [IEC08_10] auf die Sicherheitsfunktionen eines elektrisch/elektronisch/programmierbar elektronischen sicherheitsbezogenen Systems, in dieser Arbeit wird der englische Kurz-Begriff „Safety-related System“ (SRS) verwendet. Ein SRS hat die Aufgabe mit seinen definierten Sicherheitsfunktionen das Risiko eines zu kontrollierenden übergeordneten Systems (Equipment under Control, EUC) auf einem definierten Sicherheitslevel zu halten. Tritt im EUC ein gefährlicher Fehler auf, also ein Hazard, so versetzt die Sicherheitsfunktion des SRSs das EUC in den sicheren Zustand (Safe State). Inhaltlich ähnliche Definitionen finden sich in davon abgeleiteten Normen. In der ISO 26262 [ISO26_11] ist Functional Safety als “absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems[27]” definiert. Functional Safety bezieht sich somit auch hier auf die Funktion eines „E/E Systems“, und zwar

[25] Hazard nach [MIL_05, S.8]: "any real or potential condition that can cause injury, illness, or death to personnel; damage to or loss of a system, equipment or property; or damage to the environment." ("jede reale oder potentielle Situation, die Verletzungen, Krankheit oder den Tod von Menschen, Schaden an oder Verlust von einem System, Geräten oder Eigentum, oder Schaden an der Umwelt verursachen kann", Übersetzung des Verfassers)

[26] Hazard nach [ISO26_11, Teil 1, Punkt 1.57]: "potential source of harm (1.56) caused by malfunctioning behaviour (1.73) of the item (1.69)." ("potentielle Quelle von Schaden, verursacht durch Fehlfunktion einer Betrachtungseinheit.", Übersetzung des Verfassers)

[27] "Abwesenheit von unvertretbarem Risiko durch Gefährdungen, die durch Fehlfunktionen von E/E Systemen ausgelöst werden." Ein E/E system nach der [ISO26_11, Teil 1, Punkt 1.31]: "system (1.29) that consists of electrical and/or electronic elements (1.32), including programmable electronic elements." ("System, welches elektrische und/oder elektronische Elemente beinhaltet, inklusive programmierbarer elektronischer Elemente", Übersetzung des Verfassers)

darauf, dass kein Hazard durch Fehlfunktion auftreten darf. Diese Definition ist für den Zweck dieser Arbeit jedoch etwas zu eng, da gerade der gesamtheitliche Aspekt, der für das Erreichen der Sicherheitsziele notwendig ist, im Begriff „Functional Safety“ untergeht.

Mit dem „gesamtheitlichen Aspekt“ ist das System innerhalb und inklusive seiner Umgebung gemeint. Wie in den folgenden Kapiteln gezeigt wird, muss für die Entwicklung eines sicherheitskritischen Systems der gesamte Lifecycle angepasst werden, müssen Management und Prozesse dementsprechend vorbereitet sein. Deshalb wird in dieser Arbeit besonderes Augenmerk auf die detaillierte Analyse der System-Umgebung sowie die Ermittlung der Einflüsse von außen auf das System gelegt. Aus diesem Grund wird in der Folge auch von System Safety gesprochen, da System Safety über Functional Safety hinausgeht. Eine im Kontext dieser Arbeit sinnvolle Definition für “System Safety” findet sich beispielsweise im MIL-STD 882E [MIL_05]: “The application of engineering and management principles, criteria, and techniques to achieve acceptable risk within the constraints of operational effectiveness and suitability, time, and cost throughout all phases of the system life-cycle.”[28]

Die Zusammenhänge dieser Sicherheitsbegriffe lassen sich nochmals gut anhand von Abbildung 6 illustrieren. Safety als Gesamtsicherheit ist der Überbegriff über Systemsicherheit, welche wiederum funktionale Sicherheit als Teilaspekt beinhaltet, aber darüber hinausgeht.

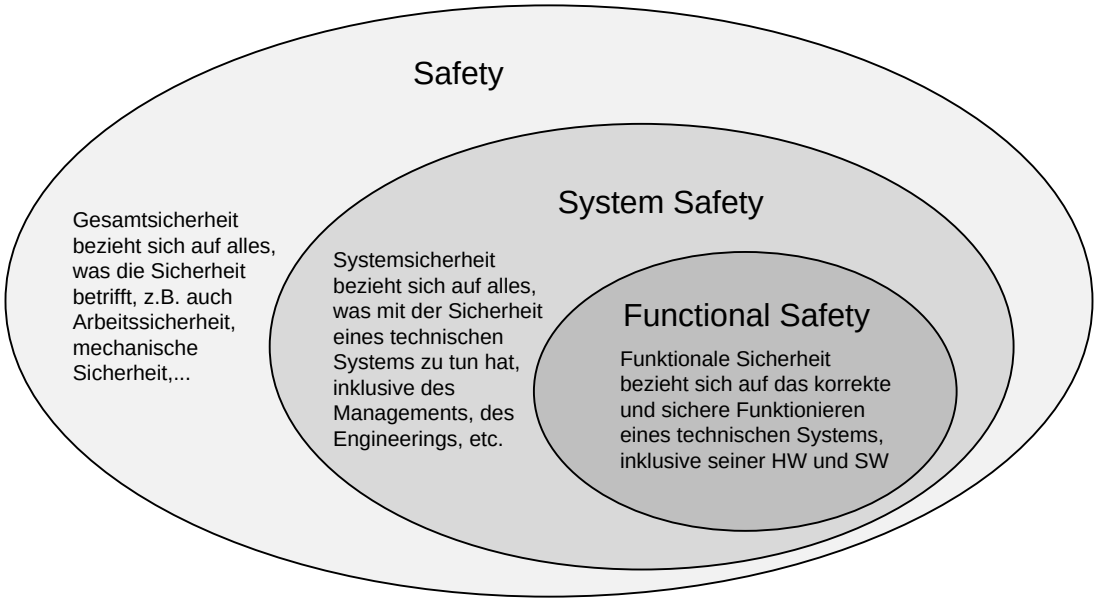


Abbildung 6: Safety vs. System Safety vs. Functional Safety

Da die Arbeit starken Bezug auf die ISO 26262 [ISO26_11] und die IEC 61508 [IEC08_10] nimmt soll jedoch auch der funktionale Aspekt nicht untergehen, weshalb folgende Ausrichtung von „System Safety“ als am besten passende angenommen werden kann: Primäres Ziel von System

[28] "Die Anwendung von Ingenieurs- und Managementprinzipien, Kriterien und Techniken, um akzeptierbares Risiko innerhalb der Grenzen betrieblicher Effektivität, Zeit und Kosten über alle Phasen des Lebenszyklus zu erreichen." (Übersetzung des Verfassers)

Safety ist einerseits das korrekte und sichere Funktionieren eines Systems in seinem Umfeld und andererseits der Nachweis ausgereifter Prozesse zur Entwicklung solcher Systeme im sicherheitskritischen Bereich. Das Risiko von Gefahren muss dabei präventiv auf einem akzeptierbaren Level gehalten werden, dass weder Menschen und anderen Lebewesen, noch die Umwelt oder das System selbst zu Schaden kommen.

2.2 Safety Normen

In den nachfolgend untergeordneten Kapiteln werden die für diese Arbeit wichtigsten Normen und Standards beschrieben. Mit einer kurzen Einführung wird der geschichtliche Hintergrund der Normen beschrieben, um die Philosophie der jeweiligen Herangehensweisen besser verstehen zu können. Danach werden der Aufbau und anschließend der Safety Lifecycle beschrieben. In den ersten beiden Unterkapiteln werden die Functional Safety Normen IEC 61508 und ISO 26262 beschrieben gefolgt von den System Safety Standards MIL-STD 882E und DO 178C.

2.2.1 IEC 61508

Die IEC 61508 [IEC08_10] hat ihren Ursprung im Seveso-Unglück, so Kletz [Kle_01]. Bei diesem bisher größten Chemieunfall Europas wurden große Mengen an Dioxin TCDD (Servogift) freigesetzt und damit große Umweltschäden angerichtet. Als Folge wurden verschärfte Gesetzte und Verordnungen zum Schutz von Menschen, anderen Lebewesen, der Umwelt und Sachschäden eingeführt, darunter auch die IEC 61508. Im Jahr 1995 wurde eine erste Draft-Version unter den Namen IEC 1508 von der International Electronical Commission (IEC) veröffentlicht. In den darauffolgenden Jahren wurde der Standard immer wieder erweitert und in den Jahren 1998 und 2000 als Safety-Norm publiziert. Die letzte Revision wurde 2010 durchgeführt. Der Fokus der IEC 61508 liegt in der Identifikation von Risken und deren Limitierung auf ein akzeptables Niveau durch sogenannte Sicherheitsfunktionen.

Die IEC 61508 wird als generische Safety-Norm bezeichnet, welche auf alle Branchen anwendbar sein soll. Sie soll dort angewendet werden wo keine sektorspezifische Safety-Norm vorhanden ist. Sie wird oft als Mutter-Normen für den Safety-Bereich bezeichnet und ist so ausgerichtet, dass davon sektorspezifische Safety-Normen abgeleitet werden können. Beispiele sind dafür die ISO 26262 oder die CENELEC-Normen für den Railway-Bereich (EN 50126, EN 50127, EN 50128, EN 50129), aber auch die Maschinennormen wie die EN 13849 und IEC 62061.

Die IEC 61508 hat ihren Ursprung, wie schon aufgrund ihres geschichtlichen Hintergrunds ersichtlich ist, im Anlagenbau und ist daher für andere Bereiche oft schwer interpretierbar. Sie konzentriert sich primär auf die Anwendung zur Entwicklung, Inbetriebnahme, Nutzung und Außerbetriebnahme von technischen Systemen im industriellen Bereich. Die Norm geht dabei von

einem übergeordneten technischen System aus, welches als Equipment under Control[29] (EUC) bezeichnet wird. Häufig werden das/die dafür erforderliche(n) Steur- und Kontrollsystem(e) in der Norm als Equipment Control System (ECS) bezeichnet und als eigene Systeme dargestellt. Das ECS kann jedoch auch im EUC integriert sein. Die erforderlichen Sicherheitsfunktionen, zum Schutz für Mensch und Umwelt, werden jedoch von einem separaten System, dem Safety-related System (SRS), zur Verfügung gestellt. Die nachfolgende Abbildung 7 zeigt die Zusammenhänge.

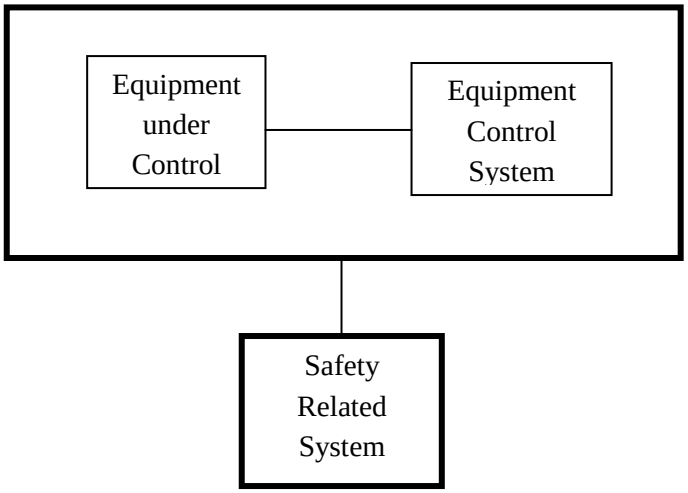


Abbildung 7: Zusammenhang EUC, ECU und SRS nach der IEC 61508

Die IEC 61508 wird als generische Grundnorm zur funktionalen Sicherheit für elektrische, elektronische und elektronisch programmierbare sicherheitsbezogene Systeme (E/E/EP-Systeme) bezeichnet. Diese E/E/EP-Systeme werden in dieser Arbeit mit Safety-related Systems, kurz SRS, benannt. Das heißt, der Hauptfokus der Norm liegt also auf dem SRS, welches die dafür erforderlichen Sicherheitsfunktionen zur Verfügung stellt. Die grundsätzliche Vorgehensweise besteht in der Identifikation von Gefährdungen im EUC-System und der Risikominderung durch das separate SRS. Dabei sollen gefährliche Ausfälle des EUC-Systems bzw. des ECS durch die Sicherheitsfunktion des SRS beherrscht werden. Die Norm definiert dabei die Anforderungen an das SRS mit dem Ziel das definierte Sicherheitsziel zu erreichen. Damit können Gefährdungen vom zu kontrollierenden System, also vom EUC, welches keiner anderen Safety-Norm unterliegt, auf einen akzeptierbaren Level reduziert werden.

Im Teil 1 der Norm ist der Safety Lifecycle beschrieben, der ein wesentlicher Bestandteil der Norm ist. Er wird auch als generischer Safety Lifecycle bezeichnet. Er besteht aus 16 Phasen und beschreibt die Anforderungen an das SRS von der Konzeptionierung über die Entwicklung bis hin zur Wartung und Entsorgung. Die Phasen 1 bis 5 beziehen sich auf die Definition des EUC Systems, sowie auf die Identifikation der Gefährdungen, die vom EUC ausgehen und den davon abgeleiteten Anforderungen an das SRS. Die Phasen 6 bis 8 beschreiben die Anforderungen an die

[29] Ein Equipment under Control würde man mit dem „Gerät oder System welches sicherheitstechnisch beherrscht werden soll“ beschreiben [Ros_14].

Planungsaktivitäten über den gesamten System-Lifecycle. Die Phase 9 und 10 konzentrieren sich auf die Entwicklung des SRS.

Die Phasen 12 bis 16 behandeln die Installation und Inbetriebnahme des Systems sowie die Validierung, Maintenance, Reparatur und Entsorgung des Systems. Die nachfolgende Abbildung 8 zeigt das komplette Lifecycle-Modell der IEC 61508.

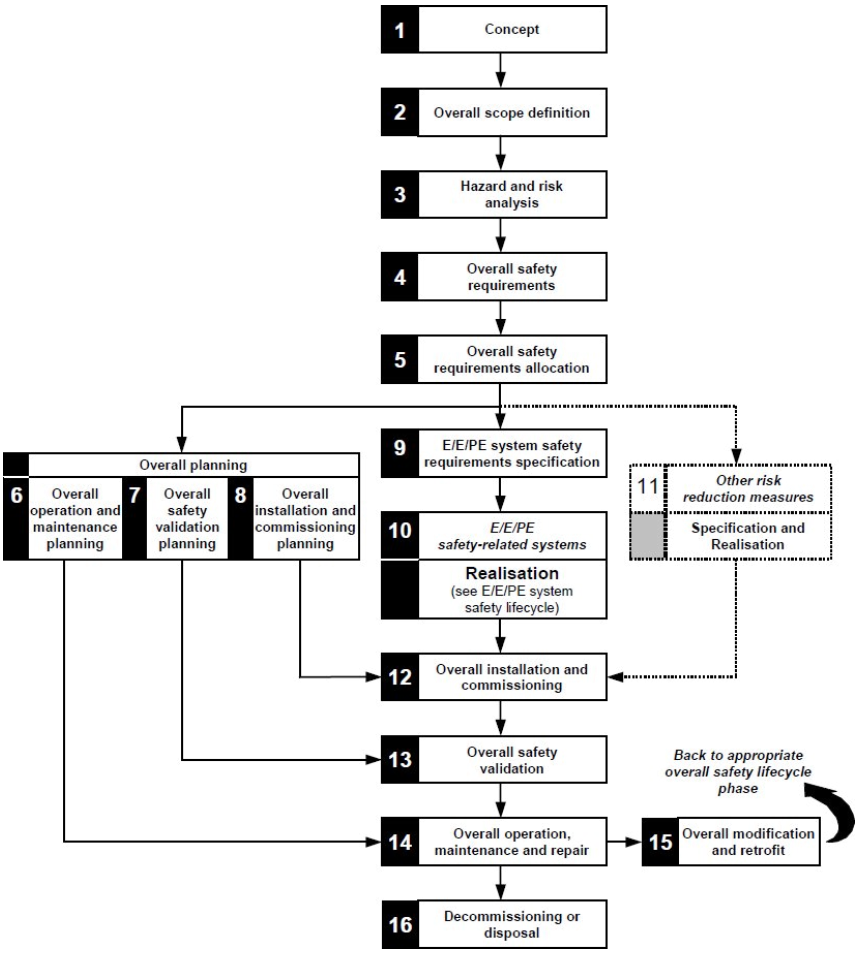


Abbildung 8: Safety Lifecycle der IEC 61508 [IEC08_10, Teil 1, S.18]

Der Safety Lifecycle dient als technisches Rahmenwerk für die standardkonforme Entwicklung des SRS. Die Phase der Konzeptionierung fordert ein definiertes und dokumentiertes Konzept des zu kontrollierenden Systems, also des EUC-Systems und, wenn nicht integriert, des ECS. Dabei sollen mögliche Gefährdungen identifiziert und diese hinsichtlich Wechselwirkungen mit anderen Systemen oder Anlagen analysiert werden. Eine wichtige Voraussetzung ist dabei die genaue Kenntnis des EUC Systems. Die darauffolgende Phase 2 fordert die Definition des Anwendungsbereichs. In Phase 3 sollen die Gefährdungen sowie die gefahrbringenden Ereignisse des EUC und des ECS identifiziert werden. Darauf basierend wird der Risiko-Level abgeleitet. Phase 4 fordert die Spezifikation, der Sicherheitsfunktionen die zur Sicherstellung der erforderlichen funktionalen Sicherheit notwendig sind. In Phase 5 werden die Sicherheitsfunktionen dem SRS zugeordnet, wobei jeder Sicherheitsfunktion die erforderliche Sicherheitsanforderungsstufe (Safety Integrity Level,

SIL) zugeordnet wird. Die IEC 61508 gibt vier Safety Integrity Levels[30] (SIL 1 bis SIL 4) vor, dabei entspricht SIL 1 dem niedrigsten und SIL 4 dem höchsten Level. SIL 4 sagt aus, dass ein sehr hohes Risiko besteht, sodass die Sicherheitsfunktion zur Risikoreduzierung umso zuverlässiger ausgeführt werden muss. Jeder SIL repräsentiert dabei einen in der Norm definierten Wertebereich.

Ziel der Phase 6 ist die Entwicklung eines Konzepts für den Betrieb und die Wartung des SRSs, sodass die funktionale Sicherheit während des Betriebs und der Instandhaltung aufrechterhalten wird. Phase 7 fordert ein Konzept zur Gesamt-Sicherheitsvalidierung des SRSs. Phase 8 umfasst die Entwicklung eines Konzepts für die kontrollierte Installation und Abnahme des SRSs. In Phase 9 wird das SRS gemäß der Spezifikation der Sicherheitsfunktion aus Phase 4 und nach den Anforderungen des Safety Integrity Levels aus Phase 5 realisiert. Auf die Phasen 10 bis 16 wird nicht näher eingegangen, da sie für diese Arbeit nicht relevant sind.

Im Teil 2 der Norm werden die Grundanforderungen an das SRS sowie die Ziele, welche durch die funktionale Sicherheit erreicht werden müssen beschrieben, der Fokus liegt hierbei auf der System- und Hardware-Ebene. Der Teil 3 der Norm beschreibt die Anforderungen an die Software-Entwicklung des SRS. Im Teil 4 werden die in der Norm verwendeten Begriffe und Definitionen beschrieben. Da die IEC 61508 keine definitive Risikomatrix vorgibt, werden im informativen Teil 5 Informationen über grundlegende Konzepte zur Ermittlung des Risikos beschrieben. Des Weiteren wird der Zusammenhang zwischen Risiko und Safety Integrity Level dargestellt sowie die erforderlichen Methoden, die zur Erfüllung der Anforderungsstufe vorgegeben werden. Im Teil 6 werden Informationen und Richtlinien für die Anwendung der Anforderungen an die SILs dargestellt. Dabei werden Beispiele zur Berechnung von Wahrscheinlichkeiten von Hardwareausfällen und Berechnungen für den Diagnosegrad dargestellt sowie Methoden zur Quantifizierung der Auswirkungen von hardwarebezogenen Ausfällen aufgezeigt. Der Teil 7 gibt Anwendungshinweise über Verfahren und Maßnahmen zur Beherrschung von zufälligen Hardwareausfällen und zur Vermeidung von systematischen Ausfällen. Ein Überblick über Verfahren und Maßnahmen zur Erreichung der SILs der Software und ein probabilistischer Ansatz zur Bestimmung des SILs von vorentwickelter Software schließen die IEC 61508.

Zusammenfassend kann gesagt werden, dass sich die IEC 61508 primär auf das SRS und dessen Sicherheitsfunktionen konzentriert, welche das EUC-System in gefährlichen Situationen in den sicheren Zustand überführen. Das heißt, die geforderte Sicherheit wird in ein zusätzlich sicherheitsrelevantes E/E/EP-System, also in das SRS, ausgelagert. Da die Philosophie der Norm beschränkt sich auf das SRS, welches eine Überwachungsfunktion ausführt und auf eine Gefährdung reaktiv mit der Einleitung des sicheren Zustands einwirkt. Da die Norm einen reaktiven Charakter hat, ist sie für die Entwicklung von inhärent sicheren Systemen, im Sinne des „Inherent System Safety Approach“ nicht ausreichend geeignet.

[30] Der Sicherheitsintegritätslevel stellt ein Maß für die Wahrscheinlichkeit dar, dass eine Sicherheitsfunktion innerhalb der erwarteten Einsatzumgebung in einem bestimmten Zeitraums fehlerfrei ausgeführt wird.

2.2.2 ISO 26262

Die ISO 26262 [ISO26_11] „Road vehicles – Functional Safety“ ist eine sektorspezifische Safety-Norm zur funktionalen Sicherheit von elektrischen, elektronischen oder programmierbar elektronischen Systemen (E/E-Systeme) in Personenkraftwagen. Die Norm wurde von der IEC 61508 abgeleitet und auf die Automobil-Industrie angepasst. Die Anwendung der ISO 26262 bezieht sich dabei auf sicherheitsbezogene Systeme in „*Serien-Personenkraftwagen*“, die in der Norm als „road vehicles“ bezeichnet werden. Dabei handelt es sich um Fahrzeuge mit 4 Rädern bis zu einem maximalen Gewicht von 3,5 Tonnen.

Mit der Norm soll sichergestellt werden, dass die E/E-Systeme die definierten Funktionen nach den Anforderungen der ISO 26262 mit angemessener Sicherheit erfüllen. Die Frage, ob die berechtigten Sicherheitserwartungen an das Fahrzeug diese Funktionen voraussetzen, vermag die ISO 26262 nicht zu beantworten. Das heißt, ob der Fahrzeughersteller beispielsweise ein ABS-System einbauen muss, gibt die ISO 26262 nicht vor. Der Fokus der Norm liegt also in der Funktionalen Sicherheit von E/E-Systemen. Neben der Betrachtung der Entwicklung des E/E-Systems einschließlich der Vernetzung mit anderen Systemen, werden auch der Einsatzbereich, die Produktion, die Wartung und Instandhaltung bis zur Entsorgung in den Lebenszyklus mit einbezogen

In der nachfolgenden Abbildung ist der Safety Lifecycle der ISO 26262 dargestellt, der in drei Lifecycle-Phasen unterteilt ist. Das sind die Konzeptphase, die Phase der Serienentwicklung und in die Phase der Produktion und Betrieb.

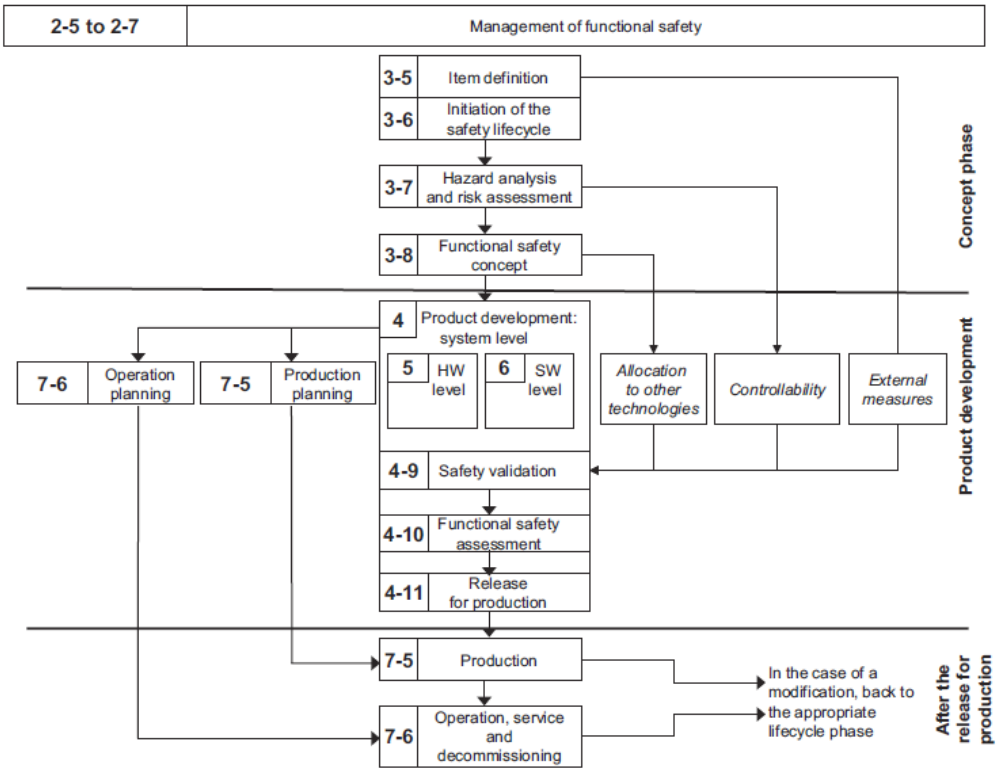


Abbildung 9: Safety Lifecycle der ISO 26262 [ISO26_11, Teil 2, S.4]

Die Anforderungen der einzelnen Phasen sind in den zehn Teilen der Norm beschrieben und sind in der Kapitelstruktur kongruent zu den einzelnen Phasen im Safety Lifecycle. Im Vergleich zum Lifecycle-Modell der IEC 61508 wird schon auf den ersten Blick ersichtlich, dass es sich bei der ISO 26262 um die Entwicklung von Serienprodukten handelt. Die Installation und Gesamtvalidierung, wie sie in der Grundnorm beschrieben sind finden im Automotive-Bereich keine Anwendung. Die ISO 26262 konzentriert sich in diesem Teil-Bereich auf die Produktfreigabe und Produktion.

In den Phasen 2-5 bis 2-7 „Management of functional safety" beschreibt die Norm die Anforderungen an die Planung, Koordination und Dokumentation aller sicherheitsbezogenen Aktivitäten über den gesamten Safety Lifecycle. In der IEC 61508 ist dieser Aspekt weniger ausgeprägt, sie fordert diesbezüglich keine konkreten Arbeitsprodukte.

Die Phase 3-5 fordert die Abgrenzung der Betrachtungseinheit, die in der Norm als Item bezeichnet wird. Der Begriff „Item" fungiert hier als Platzhalter für das zu entwickelnde System. Damit wird ermöglicht, dass die Norm unabhängig von der Position des Anwenders in der Lieferkette ist. Die Norm fordert in dieser Phase die Definition der Funktionalitäten, der Item-Interfaces sowie der Item-Umgebung und deren Interaktionen mit dem Item. Dabei sollen auch die gesetzlichen Auflagen (legal conditions) berücksichtigt werden. Die genaue Definition des Items bestimmt dabei auch den Umfang und den Inhalt des Sicherheitsnachweises. Basierend darauf sollen erste Gefährdungspotentiale identifiziert werden. Zu berücksichtigen sind auch Elemente anderer Technologien, wie beispielsweise mechanische oder hydraulische Elemente. Eine eingeschränkte Abgrenzung auf E/E-Systeme würde lediglich Teil-Resultate liefern und wäre für den Endkunden nicht zufriedenstellend.

Phase 3-6 fordert die Initiierung des Safety Lifecycles, der je nach Item-Konstellation angepasst werden soll. Hier unterscheidet die Norm folgende Kriterien: Neu-Entwicklung oder Modifikation eines vorhandenen Items. Basierend auf diesen Kriterien kann der Lifecycle mit entsprechender Begründung modifiziert werden.

Die beiden vorangegangenen Phasen liefern die Basis für die Phase 3-7, die sich der Hazard-Analyse und dem Risk Assessment widmet. Hier soll, ausgehend von den identifizierten Hazards, das Risiko bestimmt werden und mit einem gesellschaftlich akzeptablen Rest-Risiko, welches jedem technischen Element innewohnt, verglichen werden. Die Gesamtheit der risikoreduzierenden Maßnahmen wird mit dem Sicherheitsintegritätsmaß (Automotiv Safety Integrity Level, ASIL) bestimmt. Der ASIL bezeichnet einen von insgesamt vier Levels. Die Levels sind von ASIL A bis ASIL D definiert, wobei ASIL A den niedrigsten und ASIL D den höchsten Level darstellen. Der jeweilige ASIL stellt entsprechende Anforderungen an das Item zur Vermeidung von nicht tolerierbaren Restrisiken. Das genaue Vorgehen zur Risiko-Bestimmung ist in Kapitel 4.6 „Hazard-Analyse und Safety-Risiko-Assessment" beschrieben.

Die Phase 3-8 fordert das funktionale Sicherheitskonzept (Functional Safety Concept, FSC) indem festgelegt werden soll, durch welche Funktionen die Sicherheit des Gesamtfahrzeuges erreicht werden. Das FSC ist ein Set von funktionalen Sicherheits-Anforderungen (Functional Safety Requirements), welche von einem übergeordneten Sicherheitsziel (Safety Goal) abgeleitet werden. Die Safety Requirements können, außer Hardware und Software, auch andere Technologien (z.B.

Mechanik, Hydraulik) betreffen. Der Ausfall eines Motors kann beispielsweise ein Hardware- oder ein Software-Fehler sein, aber auch ein mechanischer Defekt. Darüber hinaus fordert die Norm die Verifikation des FSCs und weist damit nach, dass das jeweilige Safety-Goal erfüllt ist.

Die Anforderungen auf System-Ebene werden in Phase 4 beschrieben. In dieser Phase werden die technischen Sicherheitsanforderungen von den funktionalen Sicherheitsanforderungen abgeleitet. Dabei sind die Anforderungen an die *Sicherheitsmechanismen* zu definieren. Das ist beispielsweise die *Entdeckung und Beherrschung von Fehlern* im System, oder wie der *sichere Zustand* erreicht wird. Basierend auf den Ergebnissen dieser Teilschritte wird das technische Sicherheitskonzept spezifiziert. Der Systementwurf muss aber auch die Anforderungen zur Unterstützung in der Fertigung sowie im Betrieb, bei der Wartung und bei den Außerbetriebnahmen berücksichtigen. Die Norm fordert darüber hinaus auch die eindeutige Definition der Schnittstellen zur Hard- und Software. Die Anforderungen an die Hardware-Entwicklung (5) werden im Teil 5 der Norm beschrieben, die Anforderungen an die Software-Entwicklung (6) im Teil 6. Die Norm fordert hier eine genaue Abstimmung zwischen Hard- und Software, die in einer HW-SW-Interface Spezifikation zu definieren sind. Die Safety-Validierung (4-9) wird basierend auf den Safety-Goals auf Fahrzeugebene durchgeführt. Im Functional Safety Assessment (4-10), das entsprechend geplant werden muss, wird die funktionale Sicherheit bewertet. Der Abschluss der Serienentwicklung ist die Produktfreigabe (4-11). Die Phasen 7-5 und 7-6 werden nicht näher beschrieben, da sie für die Arbeit nicht relevant sind.

Zusammenfassend kann auch hier gesagt werden, dass die ISO 26262 eine sektorspezifische Safety-Norm für Funktionale Sicherheit ist. Sie konzentriert dabei auf das korrekte Funktionieren der Betrachtungseinheit, also des Items. Das heißt, sie fordert Sicherheitsmechanismen, welche Fehler identifizieren und beherrschen, im Sinne von Hazard Controlling. Wird ein Fehler als gefährlich identifiziert, tritt also ein Hazard auf, wird das Gesamtsystem, in diesem Fall das Fahrzeug, meist in den sicheren Zustand übergeführt. Dieser gesicherte Zustand legt in den meisten Fällen die Funktionsfähigkeit des Fahrzeugs still oder schränkt dessen Funktionalität massiv ein.

Da auch die Philosophie der ISO 26262 einen reaktiven Charakter hat, ist sie für die Entwicklung von inhärent sicheren Systemen, im Sinne des „Inherent System Safety Approach“ nur eingeschränkt geeignet.

2.2.3 MIL-STD 882E

Der MIL-STD 882E beruht auf dem Konzept von „System Safety“, welches sich 1960 aus dem US-Raumfahrtprogramm heraus entwickelt hat. 1962 wurde das Konzept das erste Mal von der U.S. Air Force beim „Minuteman Lenkwaffen Programm“ angewendet. Mit kleinen Änderungen wurde das damalige Dokument „System Safety Engineering for Development of the Air Force Ballistic Missiles“ in den MIL-STD 882E übertragen und zum „Mandatory Requirement“ des amerikanischen Verteidigungs-Ministeriums (Department of Defense, DoD) erklärt, und zwar für alle Produkte und Systeme, die beschafft werden. Integral wird dabei der System Safety Approach angewendet, welcher die Unfalls-Risiken, hier als „Mishap Risk“ bezeichnet, über den gesamten System-Lebenszyklus managt [vgl. MIL_05].

Der Hauptzweck des MIL-STD 882E ist, private Personen sowie die Öffentlichkeit vor Unfalls-Risiken zu schützen. Hierzu zählen Unfalltod, Verletzung und Berufskrankheiten. Dabei wird das Ziel verfolgt, ein akzeptierbares Unfalls-Risiko zu erreichen. Ein systematisches Vorgehen, welches aus der Hazard-Analyse, dem Risiko-Assessment und Risiko-Management besteht, wird dabei angestrebt.

Der MIL-STD 882E unterteilt System Safety in zwei Disziplinen, System Safety Management und System Safety Engineering. Für beide Disziplinen definiert der Standard Kriterien und Methoden, anhand derer das Unfalls-Risiko auf einem akzeptablen Level zu bringen ist. Dabei ist die Betrachtung der öffentlichen Rahmenbedingungen und der betrieblichen Effektivität von wesentlicher Bedeutung. Unter Safety Management versteht der MIL-STD 882E die Planung der dafür erforderlichen Aufgaben und Aktivitäten über den gesamten Lifecycle. Safety Engineering ist diejenige Disziplin, welche die geplanten Aktivitäten und Aufgaben durchführt.

System Safety spricht nicht von Sicherheitsfunktionen, die durch ein externes sicherheitsrelevantes System abgedeckt werden, sondern vom sogenannten *„design-in-safety approach"*. Dabei soll die Sicherheit in das System hineinentwickelt werden. System Safety betrachtet alle Aspekte und Charakteristiken eines Systems und verliert dabei nie den *Blick auf das Ganze*. Dabei soll keine einzelne Komponente isoliert vom Gesamtsystem *betrachtet* werden. System Safety verfolgt einen ganzheitlichen Ansatz und betrachtet dabei das Subjekt als „integrated sum-of-the-parts combination".

Der MIL-STD 882E beschreibt einen System Safety Prozess, welcher aus 8 Elementen besteht, die hier als Prozessschritte bezeichnet werden. Die nachfolgende Abbildung 10 zeigt den Ablauf der geforderten Schritte.

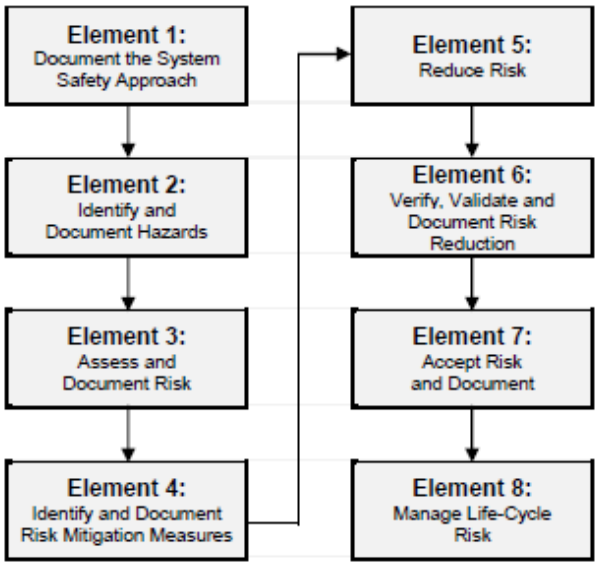


Abbildung 10: System Safety Process des MIL-STD 882E [MIL_05, S.9]

Prozessschritt 1 fordert eine Beschreibung, wie das Risiko-Management in den Engineering-Prozessen integriert ist und eine Dokumentation über die vorgeschriebenen und hergeleiteten

Requirements. Die systematische Identifikation und Dokumentation der Hazards wird im Prozessschritt 2 gefordert, die im darauffolgenden Prozessschritt 3 einem Risiko-Assessment unterzogen wird. Hier liefert der MIL-STD 882E Vorschläge in Bezug auf Severity-Kategorien, Probability-Levels und eine Risiko-Matrix, welche in dieser Arbeit in Kapitel 4.6 „Hazard-Analyse und Safety-Risiko-Assessment" genau beschrieben sind. Prozessschritt 4 fordert die Identifikation und Dokumentation der Risiko-Limitierungsmaßnahmen, wobei der MIL-STD 882E hier eine „order of precedence" vorgibt. In der ersten Rangordnungsstufe wird ein „Safe Design" gefordert, um Risiken soweit als möglich auf Design-Ebene zu vermeiden. Wenn das nicht möglich oder das Rest-Risiko noch immer als zu hoch eingestuft ist, werden Sicherheits-Funktionen gefordert. In der darauffolgenden dritten Priorisierungsstufe werden Funktionen vorgeschlagen, die Personen bei einer Gefährdung rechtzeitig warnen. In der letzten Priorisierungsstufe werden Verfahrensanweisungen und Schulungen empfohlen. In diesem Fall kann es beispielsweise auch eine persönliche Schutzausrüstung für Personen, die sich im Gefahrenbereich befinden, sein. Für Risiken der Severity-Stufe „katastrophal" oder „kritisch" sind schriftliche Warnhinweise jedoch zu vermeiden und Maßnahmen der höheren Rangordnungsstufen anzuwenden.

Die Risiko-reduzierenden Maßnahmen sind dann im Prozessschritt 5 zu treffen. Die dafür erforderlichen Kosten in Bezug auf die Effektivität sollen dabei berücksichtigt werden. Die Darstellung der Machbarkeit ist in der Planung zu beschreiben. Die Verifikation und Validation der definierten Risiko-reduzierenden Maßnahmen wird im Schritt 6 gefordert. Im Schritt 7 ist die Akzeptanz des Rest-Risikos von allen relevanten Parteien einzuholen. Der Schritt 8 schreibt ein Hazard-Tracking System vor, welches über den gesamten Lifecycle des Systems geht.

Der MIL-STD 882E gibt keinen Safety Lifecycle vor, sondern stellt die Safety-Anforderungen anhand von Task Sektions dar. Nachfolgend werden die Tasks Sektions 100 bis 400 kurz beschrieben. Die Beschreibung der Tasks ist etwas abstrahiert dargestellt, damit sie auch unabhängig von militärischen Anwendungen benützt werden können. Tasks die sich auf das DoD beschränken werden nicht angeführt.

<u>Task Section 100 – Management</u>

- Task 101: Hazard Identification and mitigation effort using the system safety methodology
- Task 102: System safety program plan
- Task 103: Hazard management plan
- Task 104: Support of government review/audit
- Task 105: Integrated product team/working group support
- Task 106: Hazard tracking system
- Task 107: Hazard management progress report
- Task 108: Hazardous material management plan

Task 101 fordert die Definition der erforderlichen Safety-Aktivitäten und -Ressourcen, die zur Hazard Identification and Mitigation notwendig sind. Darüber hinaus sind die dafür nötigen Rollen

und Verantwortlichkeiten sowie die erforderliche Kommunikationsstruktur festzulegen. Task 102 fordert die Entwicklung eines Safety-Plans, in dem alle erforderlichen Safety-Aktivitäten festgelegt sind. Als Minimum sollte der „scope of effort“ für das zu entwickelnde System und der dafür erforderliche Projekt-Lifecycle dienen. Im Task 103 wird ein definiertes Vorgehen zur Identifikation, Klassifikation und Reduktion von Hazards gefordert. Der Task 104 fordert entsprechende Reviews und Sicherheits-Audits, die an die Projektsituation angepasst sind. Task 105 ist nicht relevant für diese Arbeit. Der Task 106 fordert ein „Closed-Loop Hazard Tracking System“, das sicherstellt, dass Hazards im Projektverlauf und darüber hinaus nachverfolgt werden können. Laufende Hazard Reports werden im Task 107 gefordert. Der Task 108 ist für diese Arbeit nicht relevant.

System Safety fordert die Anwendung von Safety-Analyse-Methoden zur Identifikation von Hazards und die Reduktion der daraus folgenden Risiken auf einen akzeptablen Level über den gesamten System-Lifecycle. In der Task Section 200 sind die Anforderungen, in 10 Tasks, an der durchzuführenden Analysen beschrieben.

Task Section 200 – Analysis

- Task 201: Preliminary Hazard List
- Task 202: Preliminary Hazard Analysis
- Task 203: System Requirements Hazard Analysis
- Task 204: Sub-System Hazard Analysis
- Task 205: System Hazard Analysis
- Task 206: Operating and Support Hazard Analysis
- Task 207: Health Hazard Analysis
- Task 208: Functional Hazard Analysis
- Task 209: System-of-Systems Hazard Analysis
- Task 210: Environmental Hazard Analysis

In der Task Section 300 werden die Evaluierungsmaßnahmen, wie sie vom DoD gefordert werden, im Detail beschrieben. Da sie auf diese Arbeit keinen Einfluss haben, werden sie nicht näher beschrieben.

Task Section 300 – Evaluation

- Task 301: Safety Assessment Report
- Task 302: Hazard Management Assessment Report
- Task 303: Test and Evaluation Participation

- Task 304: Review of Engineering Change Proposal, Change Notices, Deficiency Reports, Mishaps, and Request for Deviations/Waiver

In der Task Section 400 werden die Verifikationsmaßnahmen beschrieben. Da auch diese sehr DoD-spezifisch sind, wird auch hier nicht mehr näher darauf eingegangen.

Task Section 400 – Verification

- Task 401: Safety Verification
- Task 402: Explosives Hazard Classification Data
- Task 403: Explosive Ordnance Disposal Data

Zusammenfassend kann gesagt werden, dass der MIL-STD 882E eine andere Philosophie als die IEC 61508 und die ISO 26262 verfolgt. Hier werden keine Sicherheitsfunktionen gefordert, sondern ein „Design-in-Approach". Das heißt, die Sicherheit muss in das System hineinentwickelt werden. Dabei soll der Blick auf das Ganze nicht verloren gehen. Im Gegensatz zur IEC 61508 und zur ISO 26262, in denen nur eine „Hazard and Risk-Analysis" notwendig ist, fordert der MIL-STD 882E mehrere Safety Analyse-Methoden über den gesamten System-Lifecycle. Der MIL-STD 882E besagt auch, dass keine einzelne Komponente isoliert betrachtet werden darf, sondern immer als „integrated sum-of-the-parts combination" betrachtet werden muss. Im Gegensatz dazu, stellt die ISO 26262 den Anspruch der Item-Definition, d.h. hier wird das Item vom Gesamtsystem Fahrzeug eher isoliert betrachtet. Der MIL-STD 882E spricht von Hazard-Prevention und der Risiko-Reduktion und kommt damit dem Ansatz des „Inherent System Safety Approach" schon wesentlich näher.

2.2.4 DO 178C

Durch den rapiden Anstieg von Software-Anwendungen im Airborne Bereich wurden in den 80er Jahren Richtlinien zur Entwicklung von Avionik-Software[31] im sicherheitskritischen Bereich der Luftfahrt eingeführt. Diese Richtlinien wurden von der Radio Technical Commission for Aeronautics (RTCA) durch den Safety-Standard DO178C standardisiert. Die RTCA ist eine non-profit Organisation und dient als föderative Gutachterkommission (Federal Advisory Committee) mit Sitz in den USA. Die Empfehlungen der RTCA werden teilweise auch durch die US-amerikanische Bundesluftfahrtbehörde (Federal Aviation Administration, FAA) sowie von privaten Organisationen übernommen. Auch die europäische gemeinnützige Organisation EUROCAE (European Organization for Civil Aviation Equipment), welche sich mit der Standardisierung von Elektronik in der Luftfahrt beschäftigt hat die DO178C komplett übernommen, welche in Europa unter den Namen ED-12B bekannt ist. Die FAA sowie die Europäische Agentur für Flugsicherheit

[31] Avionik bezeichet die Gesamtheit der elektrischen und elektronischen Gerätschaften an Board eines Flugzeugsystems.

(EASA, European Aviation Safety Agency) fordern den Standard für Software-Entwicklungen ebenso.

Ein wichtiger Aspekt wird in einem eignen Kapitel in der Norm beschrieben, das ist der Informationsfluss zwischen System-Lifecycle und Software-Lifecycle. Hier werden Anforderungen in beide Richtungen gestellt. Das sind beispielsweise die von den System Requirements abgeleiteten Software-Requirements, aber auch die System Safety Ziele und der Software-Integritätslevel, sowie die vorgegebenen Rahmenbedingen für die Software-Entwicklung. Es ist aber auch ein Informationsfluss vom Software-Lifecycle zum System-Lifecycle gefordert. Hier werden beispielsweise „Derived Requirements“, wenn erforderlich, an den System-Lifecycle zurückgemeldet. Unter Derived Requirements werden zusätzliche Anforderungen auf System-Ebene verstanden.

Die DO178C [DO_11] beschreibt keinen vollständigen Lifecycle, sondern stellt dafür Prozesse zur Verfügung. Diese sind in drei Gruppen geteilt, in den Planungs-Prozess, in die Entwicklungs-Prozesse und in die Integralen Prozesse.

Die Software-Entwicklungs Prozesse bestehen aus:

- Software Requirements Prozess
- Software Design Prozess
- Software Coding Prozess
- Software Integrations Prozess

Die Integralen Prozesse bestehen aus:

- Software Verification Prozess
- Software Configuration Management Prozess
- Software Quality Assurance Prozess
- Certification Liaison Prozess

Der Lifecycle wird für jede Komponente nach definierten Kriterien neu entworfen. Kriterien dafür sind beispielsweise die Komplexität und Größe der zu entwickelnden Software, oder der Anteil des Software Re-Use Teils. Die nachfolgende Abbildung 11 zeigt Prozess-Kombinationsmöglichkeiten für ein Software-Projekt.

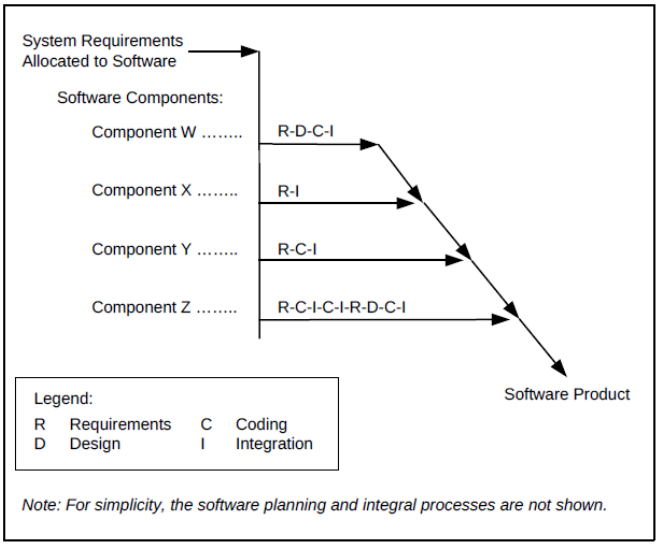


Abbildung 11: Prozess-Sequenzen für Software-Projekte nach der DO 178C [DO_11, S.22]

Den Anforderungen an den Planungsprozess zur Durchführung der Software-Entwicklung wird ein eigenes Kapitel gewidmet. Zweck des Planungsprozesses ist es, die erforderlichen Mittel für die Software-Entwicklung zu definieren und zu planen. Das Ziel ist dabei, die definierten System-Anforderungen zu erfüllen und damit wird ein entsprechendes *Vertrauen in die Flugtauglichkeit* geliefert. Die Aktivitäten beziehen sich dabei auf die Definition des Lifecycles für das zu entwickelnde Software-System und der dazu benötigten Methoden, Tools und Ressourcen. Der Planungsprozess hat die Integralen Prozesse dabei so zu koordinieren, dass die Konsistenz mit den Strategien des Softwareplans sichergestellt wird.

Die Entwicklungsprozesse und die „Integral Processes“ der DO178C, welche parallel zum Software-Lifecycle durchgeführt werden, sind von der Norm vorgegeben. Ein wichtiger Prozess ist dabei der „Certification Liaison Prozess“, der das Vorgehen bei der Zertifizierung regelt. Darüber hinaus sind die Charakteristiken und Inhalte der zu erstellenden Arbeitsprodukte, die während der Durchführung des Software-Lifecycles erzeugt werden, vorgegeben.

Zusammenfassend kann gesagt werden, dass die DO178C viel Flexibilität bei der Gestaltung des Lifecycles zeigt. Dieser wird an die jeweilige Projektsituation angepasst und durch die integralen Prozesse Verification, Quality Assurance, Configuration Management und Certification Liaison unterstützt. Darauf basierend wird die Planung der dafür erforderlichen Aktivitäten abgeleitet. Als besonders wichtig wird die Schnittstelle von den System Requirements zu den Software-Requirements gesehen, um Lücken an dieser, doch sehr wichtigen Schnittstelle, zu verhindern. Damit bestätigt die Norm, wie wichtig ein strukturgebendes Rahmenwerk ist, das aus Management-, Engineering und Support-Prozessen besteht und wie man Requirement Flaws an den Schnittstellen entgegenwirkt. Durch den Prozessorientierten Ansatz wird demonstriert, wie man systematische Fehler verhindert und wie man damit Fehlfunktionen entgegenwirken kann. Das sind wichtige Ansätze für ein Safety-Vorgehensmodell.

2.3 Integrativer Safety Prozess (ISaPro®)

Der Integrative Safety Prozess (ISaPro®) von Tschürtz [HTS_09] ist ein Rahmenwerk für Prozesslandschaften. Es ist der Struktur nach einer Prozesslandschaft mit Management-, Kern- und unterstützenden Prozessen aufgebaut und durch Safety-Prozesse ergänzt, siehe Abbildung 12.

Der ISaPro® wurde in Folge innerhalb eines 5-jährigen Forschungsprojektes[32] [COIN_14] auf unterschiedliche, projektspezifische Gegebenheiten angewendet, verfeinert und validiert. Die Ergebnisse wurden auf mehreren Safety-Konferenzen anhand von Papers publiziert und vorgetragen. Darüber hinaus wurde er in verschiedenen Unternehmen institutionalisiert.

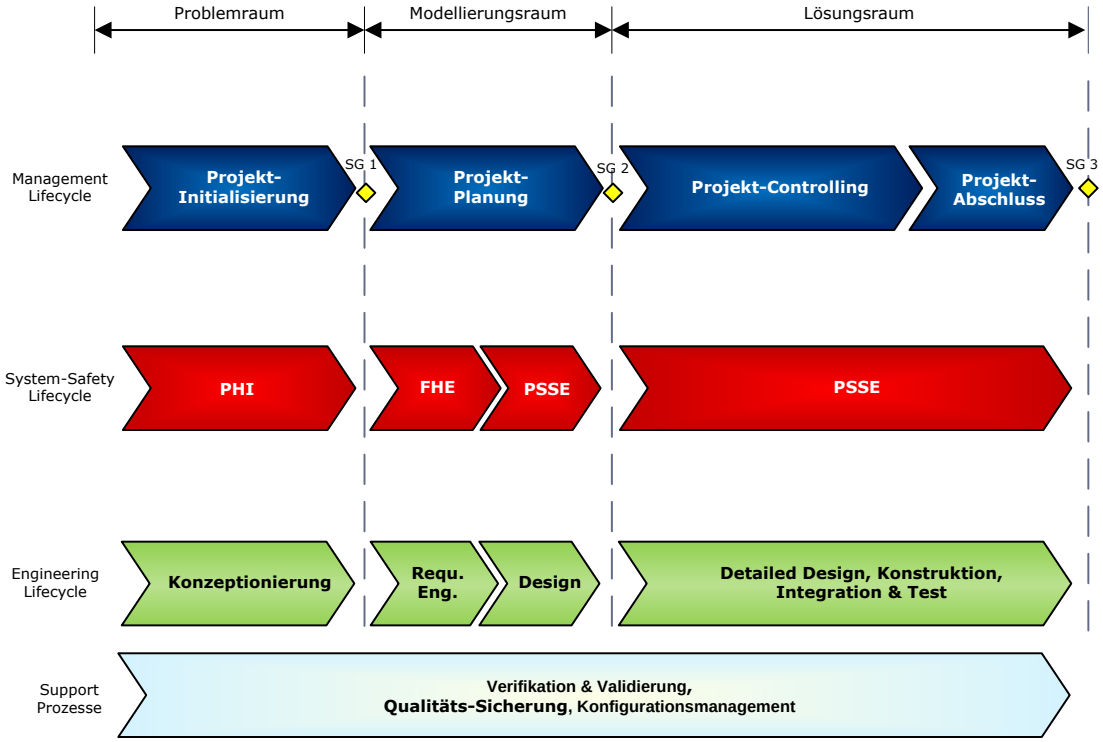


Abbildung 12: Prozess-Modell des Integrativen Safety Prozesses (ISaPro®)

Dieses ISaPro® Rahmenwerk stellt die grundlegenden Prozesse samt Basis-Praktiken für Entwicklungs-Projekte zur Verfügung, er beschreibt jedoch kein systematisches Vorgehen für die jeweiligen Disziplinen. Das Rahmenwerk legt nur fest „was zu tun“ ist; das in dieser Arbeit entwickelte Safety-Vorgehensmodell hingegen beschreibt „wie dabei systematisch-methodisch vorzugehen ist“, um die für ein sicherheitsrelevantes Entwicklungs-Projekt notwendigen Prozesse und deren Abfolge zu definieren. Damit können vor allem die Wechselwirkungen zwischen den einzelnen Prozesskatgorien nahezu lückenlos abgedeckt werden. Durch die strukturierte Prozess-

[32] Gefördert von der österreichischen Forschungsförderungsgesellschaft FFG, Programmlinie COIN

Abfolge von einander gegenseitig prüfenden Prozessen, sollen Flawed Requirements rechtzeitig entdeckt und behoben bzw. vermieden werden. Damit wird die Zielerreichung erst ermöglicht, nämlich, inhärent sichere Systeme zu entwickeln.

Die einzelnen Prozess-Phasen, sind in Abbildung 12 durch die vertikalen unterbrochenen Linien getrennt dargestellt, Tschürtz [HTS_09] spricht hier von Prozess-Räumen. Das Rahmenwerk unterscheidet Problem-, Modellierungs- und Lösungs-Raum. Die zeitliche Dimension ist durch sogenannte Stage Gates (SG), nach Cooper [Coop_02], getrennt. Sie dienen zur rigorosen Überprüfung des Projektverlaufs zwischen den einzelnen Räumen des ISaPro® (bzw. Stages nach Cooper). Eine Stage wird abgeschlossen, wenn geplante Leistungskriterien erfüllt wurden. Es wird beispielsweise überprüft, ob die geplanten Aktivitäten durchgeführt wurden und ob sich die relevanten Arbeitsprodukte im geforderten Status befinden und unter Konfigurationsmanagement stehen. Die Betrachtung liegt zudem nicht nur intern, sondern es wird auch das Projektumfeld mitbewertet. Anhand von klar definierten Go-/No-go-Kriterien wird in den Stage-Gates Meetings entschieden, ob die nächste Stage durchgeführt wird oder ob das Projekt gestoppt oder sogar ganz abgebrochen wird. Der Vorteil der Stage Gates ist, dass jeder geplannte Abschnitt disziplinenübergreifend vom Management bewertet wird.

Der erste Raum des ISaPro®, *der Problemraum, entspricht der Vor-Projektphase*. Der initiale Trigger hierfür ist eine erste Kundenanfrage, eine Ausschreibung oder eine Problemstellungen die eine technische Lösung benötigt. Der Problemraum beinhaltet die Projekt-Initialisierung, die Konzeptionierung und die Preliminary Hazard Identification (PHI). In der Projekt-Initialisierung wird das Projekt-Vorhaben beschrieben, die Projekt-Ziele definiert und eine Grob-Planung durchgeführt. Auf Basis eines technischen Konzepts aus der Konzeptionierung und den darauf basierenden Safety-Analysen, wie sie in der PHI vorgegeben sind, können die Projekt-Leistungen abgeschätzt und die Projekt-Kosten errechnet werden.

Der Modellierungsraum beinhaltet die Projektplanung, das Requirements-Engineering sowie die Entwicklung des System Designs, basierend auf den geforderten Requirements. Der Prozess der Functional Hazard Evaluation (FHE) analysiert die geforderten Funktionalitäten auf mögliche Hazards. Die identifizierten Hazards werden einem Safety-Risiko-Assessment unterzogen und davon werden die Sicherheitsanforderungsstufe sowie die Safety Requirements abgeleitet. Die Safety Requirements fließen in die nun schon vorhandene Requirements-Spezifikation ein. Sie müssen jedoch als Safety Requirements gekennzeichnet werden. Die identifizierte Sicherheitsanforderungs-stufe wird anhand von Safety-Reports an den Projektmanagement Prozess berichtet und fließt damit in die Detailplanung ein.

Die Preliminary System Safety Evaluation (PSSE) ist eine der wichtigsten Prozesse im System Safety Lifecycle. Die PSSE beantwortet die Frage, ob durch die im System Design erarbeiteten Lösungen ein akzeptabler Risiko-Level erreicht wurde. Es werden alle im Design spezifizierten Sub-Systeme und Komponenten sowie deren Schnittstellen analysiert. Diese Analysen sollen auch aufzeigen, ob mögliche Funktionsfehler oder Design-Lösungen zusätzliche Hazards in das System injizieren. Auf Basis dieser Untersuchungen werden möglicherweise zusätzliche Safety Requirements identifiziert, die rückwirkend in die Requirements-Spezifikation des Prozesses „Requirements Engineering“ einfließen müssen. Diese werden im Design realisiert und wiederum

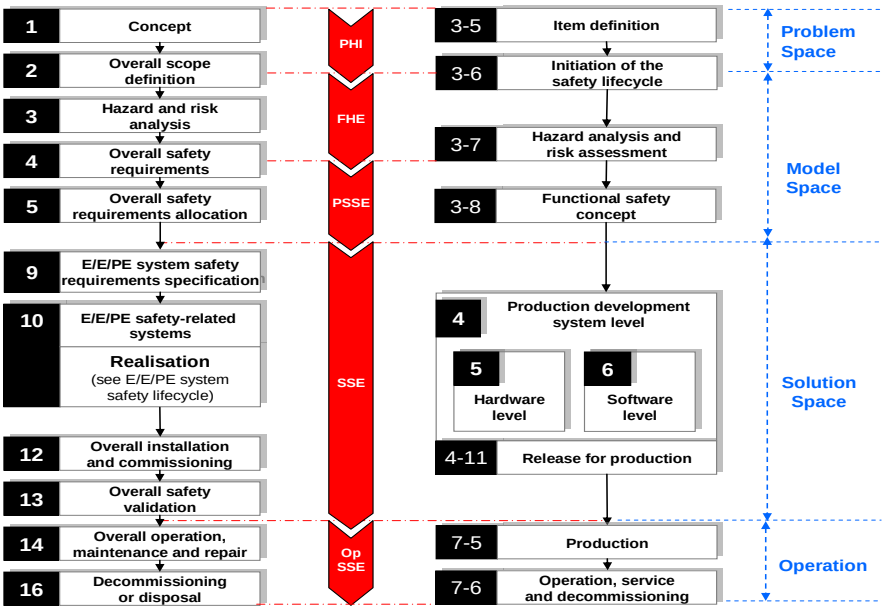
einer Safety-Analyse unterzogen. Die Ergebnisse, sowohl auf Design als auch auf Safety-Ebene, werden dann wieder anhand eines Reports, zur Verfeinerung der Projektplanung, an den Projektmanagement Prozess übermittelt. Basierend auf dieser Feinplanung werden die Kosten nochmals abgeschätzt und mit den ursprünglichen Kosten, aus der Vor-Projektphase, verglichen. Wenn hier größere Abweichungen sichtbar werden, können die Projektkosten gegebenenfalls noch einmal nachverhandelt werden. Ist dies nicht mehr möglich, so bestehen noch zwei Optionen, nämlich, das Projekt mit noch relativ geringen bereits aufgelaufenen Kosten zu stoppen, oder es mit entsprechendem Risiko weiterzuführen.

Im Lösungsraum kommen die geplanten Ressourcen zum vollen Einsatz. Die Umsetzung des Designs geht in die Realisierung. Das Projekt-Controlling im Projektmanagement-Lifecycle führt in regelmäßigen Abständen die Überwachung und Steuerung des Projekts durch. Der Prozess der System Safety Evaluation (SSE) überwacht, ob die Safety Requirements korrekt umgesetzt wurden. Dabei werden die Safety Goals und Safety-Anforderungen verifiziert, was auch als Beweis für den Safety Case dient. Der Safety Case wird als „lebendes“ Dokument geführt und durch die Aktivitäten und deren Ergebnisse in jeder Phase erweitert.

Die Support-Prozesse finden je nach Projektsituation Anwendung und sind so früh als möglich einzusetzen. Die hier typischen Prozesse sind das Konfigurationsmanagement, welches auch das Change- und Defekt-Management beinhaltet sowie die Qualitätssicherung und die Verifikation und Validation.

In der horizontalen Leserichtung des ISaPro® (siehe Abbildung 12) befinden sich die Lifecycles der einzelnen Disziplinen, nämlich der Projektmanagement-Lifecycle, der Safety Lifecycle, der Engineering Lifecycle und der Sammel-Lifecycle für die Support-Prozesse.

Wie in Tschürtz [HTS_09] dargelegt, wurden die zentralen Anforderungen der wichtigsten Normen in den Prozessen dieser Lifecycles als Basis-Praktiken definiert. Hierzu zählen beispielsweise CMMI [CMMI_11] oder die ISO/IEC 15504 [ISO15_06], die ISO/IEC 15288 [ISO88_08], die ISO/IEC 12207 [ISO07_07] sowie die IEC 61508 [IEC08_10] oder die ISO 26262 [ISO26_11]. Dadurch wird verständlich, wieso sich der eigentlich generische System Safety Lifecycle des ISaPro® mit den Lifecycles von IEC 61508 und ISO 26262 in Einklang bringen lässt, wie Abbildung 13 zeigt:



PHI … Preliminary Hazard Identification
PSSE … Preliminary System Safety Evaluation
OpSSE …Operational SSE
FHE … Functional Hazard Evaluation
SSE … System Safety Evaluation

Abbildung 13: System Safety Lifecycles basierend auf der IEC 61508 und der ISO 26262

Auf der linken Seite von Abbildung 13 ist der Safety Lifecycle der IEC 61508 dargestellt, auf der rechten Seite der Safety Lifecycle der ISO 26262 und in der Mitte der generische System Safety Lifecycle, wie er im ISaPro® dargestellt ist. Der ISaPro® System Safety Lifecycle besteht dabei aus mehreren Prozessen, die über den gesamten Lebenszyklus des zu entwickelnden Systems entsprechende Safety-Aktivitäten fordern.

Der System Safety Lifecycle des ISaPro® hat hier eine spezielle Bedeutung, da er sich von den gängigen Safety Lifecycle-Modellen in den Safety-Normen wesentlich unterscheidet. Im ISaPro® System Safety Lifecycle werden ausschließlich die Safety-Aktivitäten definiert, wie beispielsweise die Planung der Safety-Aktivitäten, die Durchführung der Safety-Analysen und das Führen des Safety Cases. Damit können sie dem verantwortlichen Safety-Ingenieur bzw. der verantwortlichen Safety-Ingenieurin zugeteilt werden. In den Safety Lifecycle-Modellen der Safety-Normen ISO 26262 und IEC 61508 wird zwischen Safety-, Engineering- und Management-Aktivitäten nicht unterschieden. Das führt in der Praxis zum Problem, dass im Prozess nicht unterschieden werden kann, welche Aktivität von welcher Disziplin bzw. Projektrolle durchgeführt werden soll. Der ISaPro® bietet den Vorteil, dass alle Aktivitäten den jeweiligen Disziplinen und Projekt-Rollen genau zugeteilt sind und damit schon die Aufgaben der einzelnen Projektteammitglieder klar strukturiert sind.

Mit dem Rahmenwerk des ISaPro® lassen sich so Gesamtzusammenhänge aller involvierten Disziplinen über den gesamten Entwicklungs-Lifecycle erkennen. Abläufe werden transparenter, Schwachstellen können entdeckt und die geforderte Konformität zu den geforderten Normen und Standards kann nachgewiesen werden.

2.4 Support Prozesse

Zu den drei wichtigsten Support-Prozessen im Safety-relventen Bereich zählen das Konfigurationsmanagement, die Verifikation und Validation sowie die Qualitätssicherung. Sie stehen in starker Wechselbeziehung mit Management-, Safety- und Engineering-Prozessen und unterstützen einen geregelten Ablauf. Sie stellen sicher, dass das System, dessen Sub-Systeme und Artefakte eindeutig identifizierbar sind und Zusammenhänge klar erkennbar werden. Sie bestätigen, dass das System und dessen Sub-Systeme die vorgeschriebenen Anforderungen erfüllen und stellen sicher, dass vordefinierte Prozesse und Pläne eingehalten werden.

2.4.1 Konfigurationsmanagement

Konfigurationsmanagement stellt für Organisationen eine wichtige Basis dar, um die gesetzlichen Vorgaben im Produkt-Garantiefall zu erfüllen. Es umfasst die Methoden, Werkzeuge und Hilfsmittel, welche die Entwicklung und Pflege eines Produktes als eine Folge von kontrollierten Änderungen (Revisionen) und Ergänzungen (Varianten) an gesicherten Prozessergebnissen unterstützt. Bezogen auf die Projektdurchführung regelt das Konfigurationsmanagement die im Projekt benötigten bzw. definierten Arbeitsergebnisse und Lieferobjekte und ordnet diese vergangenheits-, gegenwarts- und zukunftsbezogen zu. Die Änderungen im Projekt werden ebenso gesteuert wie die Änderungen an den Arbeitsprodukten.

Unter Konfigurationsmanagement wird das Identifizieren, Bestimmen, Verwalten und Lenken von definierten, unter Konfigurationsmanagement stehenden Arbeitsprodukten[33] verstanden. Die Aufgabe des Konfigurationsmanagements ist die Integrität[34] der Arbeitsprodukte, die im Projekt entstehen, zu bewahren. Dabei wird die Definition, Kennzeichnung, Versionierung, Verwaltung, Aufbewahrung und Wiederherstellung definierter Arbeitsprodukte[35] sichergestellt und laufend überprüft. Regelmäßige Konfigurations-Audits und Status-Reports stellen die Funktionsfähigkeit des Konfigurationsmanagement-Systems sicher. Bei großen und komplexen Projekten ist ein Konfigurationsmanagement-Plan zu erstellen, welcher den gesamten Ablauf des Konfigurations-managements und dessen Aktivitäten im Projektverlauf sicherstellt. Bei kleinen Projekten kann der Konfigurationsmanagement-Plan im Projektmanagementplan integriert sein.

Ziel des Konfigurationsmanagements ist es, die aktuellen Arbeitsprodukte in übersichtlicher Form den relevanten Stakeholdern zur Verfügung zu stellen. Dies geschieht, indem festgelegt wird, wie

---

[33] Arbeitsprodukte, die unter Konfigurationsmanagement stehen, können beispielsweise Pläne, Dokumente, Spezifikationen sowie Hardware-Komponenten und Software Code-Teile sein.

[34] Integrität bedeutet nach IEEE-Std. 610.12 [IE610_90]: „The degree to which a system or component prevents unauthorized access to, or modification of computer programs or data.“ Das heißt, es wird darunter das Verhindern von unerlaubtem Zugriff und unerlaubter Änderung von Computer Programmen und/oder Daten verstanden.

[35] Unter einem Arbeitsprodukt (work product) werden Artefakte verstanden die im Prozessablauf entstehen, das sind beispielsweise Zwischenprodukte des zu entwickelnden Systems oder Sub-Systems, das können aber auch Dokumente, Spezifikationen, Pläne und Zeichnungen sowie Hardware- und Software-Komponenten sein.

die Arbeitsprodukte identifiziert, kontrolliert, nachverfolgt, dokumentiert und freigegeben werden. Der Einsatz von Konfigurationsmanagement in der System-Entwicklung ermöglicht das Wiederherstellen früherer Versionen von Arbeitsprodukten und stellt so deren Verfügbarkeit sicher. Der Fokus liegt dabei auf der Transparenz der durchgeführten Entwicklungsschritte und Änderungen, sodass diese jederzeit effizient nachvollzogen werden können. Darüber hinaus unterstützt das Konfigurationsmanagement die Variantenbildung (z.B. verschiedene Produkt-varianten eines Systems) und gewährleistet, dass die Entwicklungsarbeiten auf Basis freigegebener Versionen durchgeführt werden.

Arbeiten mehrere Personen an einem Arbeitsprodukt oder werden mehrere Versionen eines Endprodukts ausgeliefert, so ist auch ein Verzweigungsmanagement (Branch-Management) notwendig. Darin wird festgelegt, in welchen Fällen Verzweigungen erlaubt sind, wie Kopien der Arbeitsprodukte benannt und versioniert werden, wie die Zusammenführung (Merging) erfolgt und welche Verifikationsschritte erforderlich sind.

Die Planung von Baselines und deren Durchführung sollen definierte Entwicklungsstände sichern, wobei die darin dargestellten Konfigurationselemente zu definieren sind und deren Status zu dokumentieren ist. Auf diese Weise werden die gesicherten Arbeitsprodukte vor Änderungen geschützt und können jederzeit reproduziert werden.

Eine Baseline ist ein Satz von definierten Arbeitsprodukten/Konfigurationselementen, der formal geprüft und genehmigt wurde, als Basis für weitere Arbeiten dient und der nur durch ein dokumentiertes Änderungsverfahren verändert werden kann. Eine Baseline stellt die Zuweisung einer eindeutigen Bezeichnung an eine Konfigurationseinheit oder eine Gruppe von Konfigurationseinheiten und zugehörigen Einheiten dar. Im Verlauf der Entstehung eines Produkts können mehrere Baselines verwendet werden, um seine Entwicklung und Prüfung zu lenken. Zumindest vor jeder Auslieferung sollte eine Baseline gezogen werden (vgl. [CMMI_11]).

In der Regel wird eine Baseline immer dann erstellt, wenn das Arbeitsprodukt bzw. die Arbeitsprodukte gewisse Kriterien erfüllen und dadurch als Basis für weitere Aktivitäten herangezogen werden können. Dabei wird besonderes Augenmerk auf die Vollständigkeit und Konsistenz der Baselines gelegt.

Das Änderungsmanagement (Change Management) ist Teil des Konfigurationsmanagements und dient wie zuvor erwähnt der Lenkung der Arbeitsprodukte, wenn Änderungen im Projekt durchgeführt werden sollen/müssen. Änderungsanträge werden im Change Control Board (CCB) bewertet und genehmigt, zurückgewiesen oder „on hold“ gesetzt. Der Grad der Konfigurationssteuerung ist von den Projekt-Zielen, -Risiken und -Ressourcen abhängig und kann je nach dem verwendeten Phasenmodell des Projekts, der Art des zu entwickelnden Systems und spezifischen Projekt-Erfordernissen variieren. Die Spannweite reicht von einer informellen Steuerung, die lediglich Änderungen während der Entwicklung der Arbeitsprodukte aufzeichnet bis hin zur formalen Konfigurationsmanagement-Steuerung anhand von Baselines (siehe unten), die sich nur im Rahmen eines formalen Konfigurationsmanagement-Ablaufs ändern lassen.

Der Konfigurationsmanagement-Prozess steuert die administrativen und technischen Abläufe über den gesamten Lifecycle. Er stellt sicher, dass die identifizierten Komponenten im System und die

dafür benötigten Dokumente über den gesamten Lebenszyklus des Systems sichergestellt werden. Des Weiteren regelt er das Vorgehen bei Modifikationen (Änderungsmanagement) und Freigaben und stellt die Abhängigkeiten und Rückverfolgbarkeit (Traceability) sicher.

Safety Normen, wie die IEC 61508 [IEC08_10, Teil 1, Kapitel 6.2.10] oder die ISO 26262 [ISO26_10, Teil 8, Kapitel 7], aber auch die DO178C [DO_11, Kapitel 7] fordern das Konfigurationsmanagement. Vordergründig dabei ist, das Safety Requirements unter Konfigurationsmanagement gestellt werden müssen. Damit kann die Traceability vom Hazard über die Safety Requirements bis hin zur Implementierung und den dafür erforderlichen Testaktivitäten sichergestellt werden. Änderungen und deren Auswirkungen können damit sichtbar und jederzeit nachverfolgbar gemacht werden. Mit einem guten Konfigurationsmanagement-Prozess können auch Schnittstellen lückenlos abgedeckt werden. Diesen Aspekt betont die DO178C im Besonderen, da damit die Schnittstelle von den System Requirements zu den Software-Requirements sichergestellt werden sollen (siehe dazu Kapitel 2.2.4 „DO 178C“). Dieser Gesichtspunkt ist auch für das Safety-Vorgehensmodell besonders wichtig, da damit an allen Schnittstellen Requirement Flaws verhindert werden können.

2.4.2 Verifikation und Validation

Die ISO/IEC 15288 [ISO88_08] sowie CMMI [CMMI_11] und die ISO/IEC 15504 [ISO15_06] definieren Verifikation und Validation in Projekten sinngemäß wie folgt:

- Der Zweck der Verifikation ist, sicherzustellen, dass die ausgewählten Arbeitsergebnisse mit den jeweils festgelegten Anforderungen übereinstimmen.
- Das Ziel der Validierung ist, zu zeigen, dass ein System oder ein Systembestandteil in der vorhergesehenen Umgebung seinen beabsichtigten Verwendungszweck erfüllt.

Verifikation- und Validations-Aktivitäten sollen über den gesamten definierten Lebenszyklus geplant und durchgeführt werden, von der Requirements Phase über die Design Phase bis hin zur Testphase. Über den Projekt-Lifecycle hinaus sollen die Prozesse auch in der Betriebs- und Maintenance-Phase sowie bei Modifikationen Anwendung finden.

Verifikations- und Validations-Aktivitäten müssen in einer definierten Umgebung mit den dafür einzusetzenden Methoden und Tools in einem Lifecycle-Modell geplant und durchgeführt werden, um eine effektive und effiziente Abarbeitung zu sichern. Dafür ist ein klar definierter Lebenszyklus, wie ihn das Rahmenwerk des ISaPro® (siehe Kapitel 2.3) vorschlägt, mit festgelegten Phasen und Prozessen sowie mit definierten Inputs und Outputs erforderlich. Dies liefert eine wichtige Basis für ein einheitliches Verständnis und die dafür erforderliche Planung.

Der Verifikations-Prozess liefert damit objektive Nachweise, dass die vorgesehenen Outputs jeder definierten Phase im Entwicklungs-Lifecycle alle spezifizierten Anforderungen erfüllen. Der Validations-Prozess zeigt, dass das System in seiner bereitgestellten Form dem beabsichtigten Verwendungszweck entspricht. Eine Priorisierung dieser Aktivitäten ist im safety-relevanten Umfeld notwendig, da sicherheitskritische Funktionen die höchste Priorität haben sollen.

Die Verifikation und Validation sind Kernelemente von Safety-Normen ([ISO26_11, Teil 8, Kapitel 9 „Verification“], [IEC08_10], [DO_11, Kapitel 6 „Software Verification Process“]). Sie werden über den gesamten Lifecycle gefordert. Damit werden die Zwischenprodukte, also die während des Projektverlaufs entstehenden Arbeitsprodukte, laufend evaluiert und können damit auf einen hohen Sicherheitsstandard gebracht werden. Sichere Arbeitsprodukte leisten einen wesentlichen Beitrag für die Qualität und das Sicherheitsniveau des Gesamtsystems. Eine wesentliche Rolle spielt dabei die Safety-Validation, sie basiert auf den definierten Sicherheitszielen (Safety-Goals) und den Sicherheitsanforderungen (Safety Requirements). Bei der Validation wird immer auf der definierten Gesamtsystemebene evaluiert. Bei der Sicherheitsvalidierung soll der Beweis geführt werden, dass die Sicherheitsziele (Safety Goals) korrekt und komplett erreicht wurden. Es soll aber auch bewiesen werden, dass die Sicherheitsziele für das System ausreichend ausgelegt sind, dem Stand der Technik entsprechen und erreicht wurden.

2.4.3 Qualitätssicherung

Zweck der Qualitätssicherung in Projekten ist, dass durch eine unabhängige Person oder Instanz, also Personen außerhalb des Projekts, zu hinterlegen ist, dass die Arbeitsprodukte und Prozesse vordefinierte Vorschriften und Pläne erfüllen. Dabei werden definierte Prozessabläufe und Arbeitsprodukte, die für das Projekt festgelegt wurden, von solchen Personen oder Instanzen anhand von Qualitäts-Audits überwacht.

Qualitäts-Audits müssen im Projektverlauf geplant werden. In diesen Audits wird überprüft, ob das Projekt prozesskonform durchgeführt wird und ob die Arbeitsprodukte in der vorgegebenen Qualität vorhanden sind. Bei Abweichungen oder Nichtkonformität müssen entsprechende Maßnahmen festgelegt werden, bei Nichteinhaltung wird in die nächste Projekt-Instanz eskaliert. Mit diesem Vorgehen soll eine einheitliche und systematisierte Arbeitsweise sichergestellt werden.

Die Qualitätssicherung ist mit der Qualitätskontrolle nicht zu verwechseln. Die Qualitätskontrolle konzentriert sich dabei auf die Qualität der System-Inhalte, die Qualitätssicherung auf die Qualität der Prozesse.

Die ISO 26262 [ISO26_11, Teil 2, Kapitel 6.4.7] fordert im Zusammenhang mit den Confirmation Measures sogenannte Safety Audits in denen die Einhaltung der definierten Prozesse überprüft wird. Die DO 178C [DO_11, Kapitel 8] fordert in einem eigenen Kapitel „Quality Assurance“, dass mit dem Prozess der Qualitätssicherung die geforderten Prozess-Ziele sichergestellt werden sollen. Mit dem Prozess der Qualitätssicherung können die Forderungen der beiden Safety-Normen sichergestellt werden.

Durch reife Prozesse und der Sicherung zur Einhaltung, sollen bei der Projekt-Durchführung systematische Fehler vermieden werden. Dieser Prozess liefert damit einen wesentlichen Beitrag zur einheitlichen Arbeitsweise und vermeidet damit Prozessfehler.

2.5 Analyse-Methoden

Um passende Safety Analyse-Methoden auswählen zu können, ist es wesentlich, deren Eigenschaften genau zu kennen. Ericson [Eric2_05] führt eine Aufstellung, Klassifikation und Beschreibung zahlreicher Safety Analyse-Methoden. Die Methoden sind nach mehreren Kriterien klassifiziert. Sie unterscheiden sich beispielsweise darin, welchen Scope (Umfang) sie abdecken, wann im Lebenszyklus sie angewendet werden sollten, ob es sich um qualitative oder quantitative Methoden handelt und ob ihr Vorgehen induktiv oder deduktiv ist.

Zu letzterer Klassifikation hat Redmill [Red_99] zusätzlich zur Unterscheidung induktiv vs. deduktiv noch die Klasse der explorativen Methoden hinzugefügt, die eine Kombination darstellen. Induktive Analyse-Methoden sind sogenannte „What-if“ Analysen und gehen vom Spezifischen ins Generelle, d.h. von der Ursache zu möglichen Konsequenzen. Dabei wird das System in einzelne Komponenten unterteilt und dann analysiert, was passiert, wenn beispielsweise eine dieser Komponenten ausfällt, nach dem Motto „What happens if something goes wrong?“. Deduktive Analysen werden als „How-can“ Analysen bezeichnet und analysieren vom Generellen ins Spezifische. Dabei wird untersucht, wie etwas passieren kann und was dessen kausale Faktoren sind. Das heißt in diesem Fall, ausgehend von den Konsequenzen werden die Ursachen analysiert. Bei explorativen Methoden wird in beide Richtungen analysiert, d.h. vom identifizierten Fehler vorwärts zur möglichen Konsequenz und rückwärts zu den möglichen Ursachen.

Die Anwendung von Methoden aller Gattungsfamilien ist vorteilhaft, da nur eine Analyse-Methode das Ergebnis einschränken würde und dadurch Hazards übersehen werden könnten. Die angemessene Kombination von Methoden fordern auch Safety-Normen, beschreibt Löw [LPP_10, S.143] und wenn alternativ nur eine verwendet wird, dann muss dies begründet werden. Es ist auch von besonderer Wichtigkeit, die richtige Methode zum richtigen Zeitpunkt einzusetzen. Induktive Methoden können bei komplexen Systemen aufgrund ihrer großen Anzahl von Elementen sehr aufwendig werden. Auch können Fehlerkombinationen nicht analysiert werden, was bei deduktiven Methoden möglich ist. Bei der Anwendung von deduktiven Methoden sind jedoch große Mengen an Daten sowie eine detaillierte Dokumentation des zu untersuchenden Systems erforderlich, sie sind damit auch sehr zeitaufwendig. So hat jede Methode ihre Stärken und Schwächen, und die Wahl der optimalen Kombination von sich ergänzenden Methoden (der richtige „Methoden-Mix“) ist sowohl aus Gründen der Effektivität sowie auch aus Gründen der Effizienz wesentlich.

Häufig verwendete Methoden sind beispielsweise die Fault Tree Analysis (FTA) [FTA_81], die Failure Mode and Effect Analysis (FMEA) [IEC12_06] sowie die Hazard and Operability Study (HAZOP) [Def58_00]. Die FMEA ist eine induktive Methode, die auf mehreren Ebenen der Systementwicklung eingesetzt werden kann. Sie analysiert die kritischen Fehler und deren Konsequenzen. Die FTA, eine deduktive Methode, geht stark ins Detail und identifiziert „Cause Consequence Failures und deren „Root Causes“. Sie ist dementsprechend aufwendig und betrachtet einen sehr eingeschränkten Bereich. Die HAZOP ist eine explorative Methode, weil sie in beide Richtungen analysiert. Mit der HAZOP können einzelne Elemente sowie auch die Beziehungen zweier Elemente untersucht werden, sie zeigt dabei systematische Abweichungen auf.

Von großer Bedeutung bei der Auswahl und Anwendung von Analyse-Methoden ist außerdem die Frage, ob es sich um eine quantitative[36] oder um eine qualitative[37] Methode handelt. Beide Zugänge haben zweifelsohne ihre Vor- und Nachteile und die Auswahl der „richtigen" Herangehensweise ist sicherlich zu einem gewissen Teil „more of an art than a science"[38] [Eric2_05, S. 51]. Eine quantitative Methode hat einen höheren Anspruch auf Objektivität. Andererseits sind die Ergebnisse nur so gut wie die Inputdaten, die man zur Verfügung hat. Gerade dort, wo man mit hypothetischen Ereignissen (zum Beispiel Hazards) rechnet, deren Auftrittswahrscheinlichkeit von vielen weitgehend nur geschätzten Parametern abhängt, kann eine quantitative Methode auch fälschlicherweise eine nicht vorhandene Genauigkeit suggerieren. In solchen Fällen ist dann eine qualitative Methode, die obendrein in der Regel weniger aufwendig in der Anwendung ist, potentiell besser geeignet. Auch aufgrund der tendenziell größeren Zeitintensität bei der Anwendung von quantitativen Methoden und der daraus entstehenden höheren Kosten empfiehlt Ericson [Eric2_05] die Anwendung von qualitativen Methoden. Quantitative Methoden empfiehlt er lediglich bei der Untersuchung von sogenannten „High-Consequence Hazards".

> "System safety prefers the qualitative risk characterization method because for a large system with many hazards it can become cost prohibitive to quantitatively analyse and predict the risk of each and every hazard. In addition, low-risk hazards do not require the refinement provided by quantitative analysis. It may be necessary to conduct a quantitative analysis only on a select few high-consequence hazards. Experience over the years has proven that qualitative methods are very effective and in most cases provide decision-making capability comparable to quantitative analysis."[39] [Eric2_05, S. 52]

Braband [Bra_05, S. 97] verstärkt die Aussage: „…] beim Durchführen einer quantitativen Risikoanalyse erkennt man, dass mit der Anzahl der modellierten Parameter nicht nur der Aufwand steigt, sondern in der Regel die Aussagegenauigkeit des Ergebnisses sinkt, da die für die quantitative Analyse notwendigen Daten nicht in ausreichendem Umfang zur Verfügung stehen."

Um diese „high-consequence hazards" zu identifizieren ist also eine Priorisierung der Hazards notwendig. Ericson [Eric2_05, S. 52] empfiehlt folglich: "Qualitative risk characterization provides a very practical and effective approach when cost and time are concerns and/or when the availability

[36] "Qualitative analysis or evaluation based on qualitative values. Mathematical calculations are generally not involved; however, qualitative indices may be combined. A qualitative result is produced, which is considered subjective and/or fuzzy." [Eric2_05, S. 477]

[37] "Quantitative analysis or evaluation based on numerical values and/or mathematical calculations. A quantitative result is produced, which is considered objective and concrete." [Eric2_05, S. 477]

[38] "mehr Kunst als Wissenschaft" (Übersetzung des Verfassers)

[39] "System Safety bevorzugt die qualitative Risikobeschreibungsmethode, da es für ein großes System mit vielen Hazards unerschwinglich (teuer) werden kann, jeden einzelnen Hazard quantitativ zu analysieren und das Risiko quantitativ vorherzusagen. Außerdem benötigen Hazards mit geringem Risiko nicht die Exaktheit, die eine quantitative Analyse bietet. Es kann notwendig sein, eine quantitative Analyse nur für wenige ausgewählte Hazards mit großem Risiko durchzuführen. Die Erfahrung über die Jahre hat gezeigt, dass qualitative Methoden sehr effektiv sind und in den meisten Fällen eine gleichwertige Basis für die Entscheidungsfindung bieten wie eine quantitative Analyse." (Übersetzung des Verfassers)

of supporting data is low. The key to developing a qualitative risk characterization approach is by carefully defining severity and mishap probability categories."[40]

Aus wissenschaftlicher Sicht werden quantitative Methoden in der der Regel bevorzugt, da sie einen höheren Anspruch an Objektivität erfüllen [Eric2_05]. Der Aufwand für quantitative Analysen muss jedoch immer mit dem Nutzen verglichen werden. Diese Arbeit stellt den Anspruch eine Vorgehensweise zu entwickeln, welche die Effizienz und Effektivität in den Vordergrund stellt. Der strategische Ansatz zielt darauf ab, dass in sehr frühen Phasen Hazards identifiziert werden können, mit einfach anzuwendenden qualitativen Methoden.

Im Vordergrund steht ebenso die richtige Methode zum gegebenen Zeitpunkt anzuwenden. Dies soll durch das Safety-Vorgehensmodell in dieser Arbeit sichergestellt werden. Das heißt, wann, wo und wie die ausgewählten Methoden zum Einsatz kommen. Wie schon Ericson [Eic2_05] schreibt "One particular hazard analysis type does not necessarily identify all the hazards within a system; it may take more than one type, […]"[41], reicht die Anwendung einer einzelnen Methode oft nicht aus. Durch einen gut überlegten Methoden-Mix kann ein hoher Abdeckungsgrad von identifizierten Hazards erreicht werden. Ziel ist, das gesamte Spektrum des Systems und des System-Umfeldes abzudecken. Für die konkrete Durchführung stehen verschiedene Methoden zur Verfügung, die im Folgenden kurz skizziert werden.

2.5.1 Preliminary Hazard List (PHL)

Die „Preliminary Hazard List" (PHL), wie von Ericson [Eric2_05] beschrieben, ist die einfachste Hazard Analyse-Methode, die auch schon sehr früh im Lifecycle angewendet werden kann. Das Ergebnis ist eine Liste von tatsächlichen oder vermuteten Hazards für ein geplantes System. In ihrer einfachsten Form kann diese Methode mittels Brainstorming in einem Team durchgeführt werden. Ericson [Eric2_05] schlägt eine etwas formalere Form vor, die auf dem Design des Systems (soweit bekannt), auf Hazard-Checklisten und auf „Lessons Learned" aufsetzt.

Die Vorgehensweise ist dann wie folgt ([Eric2_05, Table 4.1]): Es wird ein Team an Experten gebildet und die PHL geplant. Dann werden alle zum gegebenen Zeitpunkt vorhandenen Informationen zum System identifiziert. Im nächsten Schritt werden taxativ alle Hardware-Komponenten und die Systemfunktionen anhand von vorgegebenen Checklisten analysiert, um potentielle Hazards zu identifizieren. Schlussendlich werden Maßnahmen zur Linderung der identifizierten Hazards empfohlen und die Analyse dokumentiert.

---

[40] "Qualitative Risikoanalyse ist ein sehr praktischer und effektiver Zugang, wenn Kosten und Zeit wesentlich sind und/oder wenn wenig unterstützende Daten verfügbar sind. Der Schlüssel, um einen qualitativen Risikoanalyse-Ansatz zu entwickeln, ist die klare Definition von Schweregrad- und Unfallwahrscheinlichkeitskategorien." (Übersetzung des Verfassers)

[41] "Eine bestimmte Hazard-Analyse-Methode ermittelt nicht notwendigerweise alle Hazards im System; es könnte mehr als eine Methode benötigen […]" (Übersetzung des Verfassers)

2.5.2 Functional Failure Analysis (FFA)

Die Functional Failure Analysis, von Mc Dermid [McD_02] entwickelt, ist eine Analyse-Methode, mit der Fehlfunktionen identifiziert werden können. Die zu analysierenden Funktionen werden dabei mit sogenannten „Guidewords" kombiniert (jede Funktion mit allen Guidewords), um mögliche Abweichungen zu identifizieren. Guidewords sind allgemeine wiederverwendbare Schlüsselbegriffe, die dazu dienen, im Rahmen der Analyse auch nicht sofort offensichtliche Abweichungen zu identifizieren. Sie stellen hypothetische, also potentiell mögliche Fehler-Modi zu Funktionen dar. Die Anwendung von Guidewords gibt der Methode eine gewisse Systematik, da man damit im Rahmen der Analyse gezwungen ist, alle Kombinationen zu bedenken.

Fehler-Modi werden durch drei hypothetische Abweichungen hervorgerufen: (1) Omission: Funktion steht nicht zur Verfügung, wenn sie benötigt wird; (2) Commission: Funktion wird ausgeführt, wenn sie nicht gefordert wird und (3) Incorrect: Funktion wird fehlerhaft ausgeführt bzw. es liegt eine Funktionsstörung vor.

Diese obigen drei Guidewords sind anwendungsunabhängig, da man sich für jede Funktion die folgenden drei Fragen stellen kann:

- (Omission) Was passiert, wenn die Funktion nicht ausgeführt wird, obwohl sie ausgeführt werden sollte? Dies ist also der typische „*Ausfall*" einer Funktion, also der vermutlich intuitivste Fehlerfall.
- (Commission) Was passiert, wenn die Funktion ausgeführt wird, obwohl sie nicht ausgeführt werden sollte? Dies beinhaltet eine *fälschliche Aktivierung* bzw. eine Aktivierung der Funktion zu einer falschen Zeit (zu früh bzw. zu spät).
- (Incorrect) Was passiert, wenn die Funktion zwar zur richtigen Zeit ausgeführt wird, sie allerdings fehlerhaft ausgeführt wird? Dies ist vermutlich der umfassendste Fehlerfall, da sich dahinter unzählige Fehlermodi verstecken können (zum Beispiel falsche Daten werden retourniert, die Funktion wird zu schnell/langsam ausgeführt, die Funktion hat Seiteneffekte…). Hier ist somit die Expertise der an der FFA teilnehmenden ExpertInnen besonders gefragt, um potentielle Risiken weder über- noch unterzubewerten.

Im Rahmen der FFA wird nun jede Funktion mit jedem dieser drei Guidewords kombiniert, um mögliche konkrete Abweichungen zu identifizieren und deren Sicherheitskritikalität zu bewerten. Zu diesem Zweck müssen die relevanten Experten, die diese Identifikation und Bewertung vornehmen, alle Funktionen dahingehend analysieren. In der Regel wird dies nur für Funktionen auf höherer Ebene angewendet, da zu dem Zeitpunkt, in dem eine FFA in der Regel durchgeführt wird, oft nur Top-Level Funktionen bekannt sind und sich mit jeder weiteren Detaillierungsebene der Aufwand multipliziert. Das Ergebnis ist eine Liste an Fehlfunktionen und eine Klassifikation bzgl. deren Sicherheitskritikalität.

2.5.3 Hazard and Operability Analysis (HAZOP)

Die „Hazard and Operability Analysis“, kurz HAZOP, ist eine weit verbreitete Methode, die als Erweiterung der FFA gesehen werden kann. Eine HAZOP ist nicht auf funktionale Untersuchungen beschränkt, sondern es können damit auch Prozesse, Designs oder Software untersucht werden. Der englische Standard DEF-STAN 00-58 [Def58_00] beschreibt den HAZOP-Prozess im Detail. Da die HAZOP als Methode im Safety-Vorgehensmodell angewendet wird, folgt hier eine kurze Beschreibung.

Das Wesen der HAZOP ist, die Untersuchung von möglichen Abweichungen der „Design Intention“, also des gewünschten Designs oder Verhaltens, um mögliche Hazards zu identifizieren. Die Analyse geht sehr strukturiert vor, der [Def58_00] beinhaltet einen Flow Chart, der alle Schritte der Analyse im Detail erklärt. Kurz zusammengefasst sind folgende Schritte nötig:

1. Klärung des Scopes, Klärung der vorhandenen Designinformation.

2. Für jede Komponente alle gewünschten Attribute (Verhalten) auflisten.

3. Für jedes gewünschte Verhalten alle Guidewords anwenden.

4. Glaubwürdige Abweichungen dokumentieren.

Glaubwürdige Abweichungen können nun gefährlich sein oder nicht. Wenn sie gefährlich sind, dann handelt es sich um Hazards, die entsprechend behandelt werden müssen. Das Wesen der Analyse ist das Anwenden der Guidewords (Schlüsselbegriffe) auf das gewünschte Verhalten. Die Liste der Guidewords kann je nach Anwendungsfall definiert werden, eine generische Liste wird aber in [Def58_00] vorgeschlagen (siehe Tabelle 1).

Guideword	Generic Meaning
No	This is the complete negation of the design intention. No part of the intention is achieved and nothing else happens.
More	This is a quantitative increase.
Less	This is a quantitative decrease.
As well as	All the design intention is achieved together with additions.
Part of	Only some of the design intention is achieved.
Reverse	The logical opposite of the intention is achieved.
Other than	Complete substitution, where no part of the original intention is achieved but something quite different happens.
Early	Something happens earlier than expected relative to clock time.
Late	Something happens later than expected relative to clock time.
Before	Something happens before it is expected, relating to order or sequence.
After	Something happens after it is expected, relating to order or sequence.

Tabelle 1: Liste von HAZOP-Guidewords aus [Def58_00]

Eine HAZOP wird idealerweise in einem multidisziplinären Team durchgeführt. Sie sollte von einen erfahrenen Moderator geführt werden, da die kognitive Belastung der Teilnehmer sehr hoch ist, und die Gefahr besteht, bei einzelnen Aspekten zu tief ins Detail zu gehen. Solche Abschweifungen sollten vom Moderator erkannt und entsprechend gegengesteuert werden.

Eine vollständig durchgeführte HAZOP kann sich über mehrere Sessions ziehen, die innerhalb von Tagen oder Wochen stattfinden. Das Hauptergebnis sind die gefundenen Hazards, deren Auftreten nun mit entsprechenden Maßnahmen verhindert werden kann. Ein wesentliches sekundäres Ergebnis, ist jedoch das während der Diskussionen entstandene gemeinsame Verständnis über das System, von dem alle Diskussionsteilnehmer profitieren können, und das zur Linderung von Kommunikationsproblemen führt und gerade für das Verständnis von Safety wichtig ist.

2.5.4 Fault Tree Analysis (FTA)

Die Fault Tree Analysis (FTA), im deutschsprachigen Raum als Fehlerbaumanalyse [DIN25_81] bezeichnet, ist ein seit den 1960er Jahren im safety-relevanten Bereich eine sehr bekannte und beliebte Methode. Eines der ersten Dokumente, in dem die FTA vollständig beschrieben wurde, ist das Fault Tree Handbook der „Office of Nuclear Regulatory Research“ [FTA_81], eine aktuellere Beschreibung findet sich in [Eric2_05].

Die FTA ist eine sogenannte Top-Down Analyse, in der deduktiv vorgegangen wird. Das heißt, man geht von einem (unerwünschten) Ereignis aus und analysiert Schritt für Schritt die potentiellen Ursachen. Meist liegt diesem unerwünschten Ereignis kein tatsächlicher Vorgang zugrunde, sondern ein theoretisch denkbarer und vorhersehbarer Hazard. Ziel der Analyse ist es nun, für diesen Hazard die Letztursachen („root causes“, in Fault Trees oft „basic events“ genannt) und deren Kombinationen zu finden, die zum Hazard führen. So kann ermittelt werden, ob diese Ursachen vermeidbar sind bzw. wie hoch deren Wahrscheinlichkeiten sind.

Der Aufbau eines Fault Trees (siehe Abbildung 14 für einen sehr einfachen, beispielhaften Fehlerbaum) lässt sich wie folgt vornehmen: Man beginnt mit dem unerwünschten Ereignis, also dem Hazard. Dieses Ereignis wird ganz oben dargestellt, es bildet die Wurzel des Baumes. Dann werden die unmittelbaren, notwendigen und hinreichenden Ereignisse dargestellt. Diese werden mit logischen Gattern (typischerweise UND und ODER, die FTA kennt aber auch zahlreiche speziellere Gattertypen) verknüpft, um darzustellen, ob alle Ereignisse (UND) oder nur eines (ODER) eintreten muss, um zum Hazard zu führen. Dies wird solange durchgeführt, bis man zu den „basic events“ gelangt (wobei man natürlich eine Grenze setzen muss, bis zu welcher Ebene die Analyse durchführt werden soll, z. B. Bauteilebene). Fault Trees können durchaus sehr groß werden und aus mehreren hundert Events und Gattern bestehen.

Die Darstellung in einem Fault Tree gibt wertvolle qualitative Einblicke in die Wirkmechanismen, die zu einem Hazard bzw. einem unerwünschten Ereignis führen. Zusätzlich bietet sich die Möglichkeit, die Analyse auch quantitativ durchzuführen, also Wahrscheinlichkeiten für das unerwünschte Ereignis zu berechnen. Dazu müssen allerdings die Wahrscheinlichkeiten der „basic events“ bekannt sein. Wenn es sich hierbei beispielsweise um Bauteilausfälle handelt, so sind diese Wahrscheinlichkeiten in der Regel bekannt.

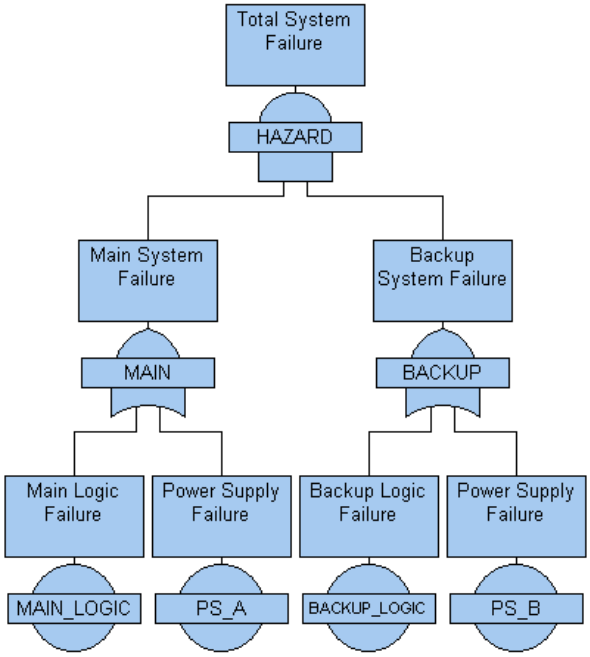


Abbildung 14: Beispiel einer Fault Tree Analyse

Die FTA ist also ein wertvolles Werkzeug in der Safety- und Risikoanalyse, da wie erwähnt Risiken damit sowohl qualitativ als auch quantitativ bewertet werden können.

2.5.5 Failure Modes and Effects Analysis (FMEA)

Die Failure Modes and Effects Analysis (FMEA) [IEC12_06] ist eine weit verbreitete Analyse-Methode, die in verschiedenen Industriezweigen angewendet wird. Seit vielen Jahren wird sie in der Automotive-Industrie eingesetzt [VDA_12], forciert durch den Ford Pinto Skandal[42] [PiMa_77]. Sie verfolgt den Grundgedanken Fehler präventiv, also bevor sie noch passieren, zu vermeiden. Die FMEA ist eine Methode, mit deren Hilfe man systematisch alle Elemente eines Systems auf potentielle Fehlfunktionen untersucht, deren Auswirkungen ermitteln und dokumentiert. Die Familie an FMEA-Methoden, die sich mit Fehlermodis und deren Auswirkungen beschäftigen ist sehr aktuell in Wedrich [Wer_12] zusammengefasst.

Die FMEA ist eine Bottom-Up Analyse, in der induktiv vorgegangen wird. Das heißt, die FMEA geht von einer hierarchischen Zerlegung des Systems aus und beginnt mit der Analyse der Elemente auf der untersten Ebene. Dabei werden alle Ausfallsarten dieser definierten Elemente identifiziert und dokumentiert. Nach der Fehler-Identifikation werden für jeden Fehler die Fehler-Auswirkungen und Fehler-Ursachen analysiert. Die nachfolgende Tabelle 2 zeigt eine Beispielszeile einer FMEA. Das zu analysierende Element ist in diesem Fall eine Kühlwasserpumpe eines chemischen Reaktors, deren Ausfall mit der FMEA analysiert werden soll.

[42] Ford Pinto Skandal: Durch einen gefährlichen Mangel in der Konstruktion zerbrach bei Auffahrunfällen der Tank. Dadurch gerieten die Fahrzeuge in Brand, welcher mehrere hundert Todesopfer zur Folge hatte.

#	Element	Failure	Unmittelbarer Effekt	Effect auf Gesamtsystem-Ebene	Mögliche Ursache
01	Kühlwasser-pumpe	Pumpe steckt	Kühlwasserversorgung ist unterbrochen	Chemischer Reaktor wird überhitzt	Mechanischer Defekt Elektrischer Defekt Stromausfall

Tabelle 2: Beispiel einer tabellarischen FMEA

Als Failure wird die stecken gebliebene Pumpe angenommen, deren unmittelbarer Effekt die unterbrochene Kühlwasserversorgung ist. Auf Gesamtsystem-Ebene wirkt sich das in einem überhitzten Reaktor aus. Als mögliche Ursachen werden ein mechanischer- und/oder ein elektrischer Defekt und/oder ein Stromausfall prognostiziert. Ziel ist es nun, auf Basis der identifizierten Ursachen, Gegenmaßnahmen zu definieren. Das Ergebnis ist eine vollständige Bewertung aller möglichen Fehlermodis.

Im Gegensatz zur FTA handelt es sich bei der FMEA um eine induktive Methode, die von Einzelausfällen auf die Auswirkungen im Gesamtsystem schließt. Ein wesentliches Merkmal ist, dass nur die Auswirkungen von Einzelausfällen untersucht werden. Was die Effekte von mehreren simultan auftretenden Ausfällen sind, bleibt bei dieser Analyse verborgen. Im Falle der FMEA handelt es sich um eine qualitative Analyse.

Es sei an dieser Stelle noch die FMECA erwähnt, die eine Erweiterung der FMEA ist. Die Grundstruktur ist dabei die gleiche, sie enthält nur zusätzliche Mittel zur Klassifizierung der Schwere und der Ausfallsarten, um die Einstufung auf Dringlichkeit von Abhilfemaßnahmen zu ermöglichen. Dies geschieht durch Kombination des Maßes für die Schwere mit der (erwarteten) Eintrittshäufigkeit, um so eine „Kritizität“ genannte Metrik zu erzeugen.

Die FMEA ist eine meist eingesetzten Methoden im Qualitätsmanagement und wird auch von vielen Safety-Normen empfohlen.

2.6 Safety Case: Goal Structuring Notation (GSN)

Ein Safety Case (Sicherheitsnachweis) ist eine schlüssige Argumentation, die die ausreichende Sicherheit eines gegebenen Systems in einer definierten gegebenen Umgebung über seine Lebensdauer belegt[43]. Viele Safety Standards (unter anderen auch die ISO 26262) fordern einen solchen Safety Case. Der Safety Case ist das zentrale Dokument für ein sicherheitskritisches System, auf dessen Grundlage das System zugelassen wird oder nicht. Wenn der Safety Case nicht schlüssig

[43] Eine häufig zitierte und treffende Definition für "Safety Case" findet sich auf http://www.adelard.com: "A document body of evidence that provides a demonstrable and valid argument that a system is adequately safe for a given application and environment over its lifetime."

genug ist, also der zulassenden Stelle die Sicherheit eines Systems nicht unzweifelhaft beweist, so darf ein System nicht in Betrieb genommen werden. Aus diesem Grund kann der Safety Case als genauso zentral wie das System selbst angesehen werden.

Der Aufbau eines Safety Case ist in der Regel so, dass bestimmte Behauptungen (Claims) aufgestellt, die durch Argumente (Arguments) begründet werden, die wiederum durch Beweisstücke (Evidences) belegt sind. Diese Argumentation kann rein beschreibend formuliert werden. Der Nachteil dabei ist allerdings, dass die Übersicht leicht verloren geht, da solche Safety Cases Gefahr laufen, schwer lesbar zu sein und somit die Gültigkeit des Arguments nicht leicht überprüfbar ist (siehe Kelly und Weaver [KeWe_04], Seite 2 für Beispiele). Aus diesem Grund schlägt Kelly [Kel_98] eine Notation vor, die die Lesbarkeit und Überprüfbarkeit von Safety Cases massiv verbessern soll. Die „Goal Structuring Notation“ (GSN) stellt explizit dar, was die Behauptungen, Argumente und Beweisstücke in einem Safety Case sind. Die Notation der GSN besteht aus zahlreichen Symbolen, deren Erklärung den Rahmen der Arbeit sprengen würde. Aus diesem Grund werden in Abbildung 15 nur die wesentlichsten Symbole dargestellt. Das „Goal“ stellt die zu beweisende Behauptung auf, welche in einem bestimmten „Context“ stehen kann. Die Korrektheit von „Goal“ wird mittels einer „Strategy“ argumentiert, wobei das Goal in der Regel in mehrere Sub-Goals aufgeteilt wird. Die Korrektheit der untersten Sub-Goals wird mit „Solutions“ bewiesen. Die GSN ist in einem „Community Standard“ ([Ori_11]) vollständig definiert, Spriggs [Spr_12] beinhaltet eine gut lesbare Einführung zur GSN.

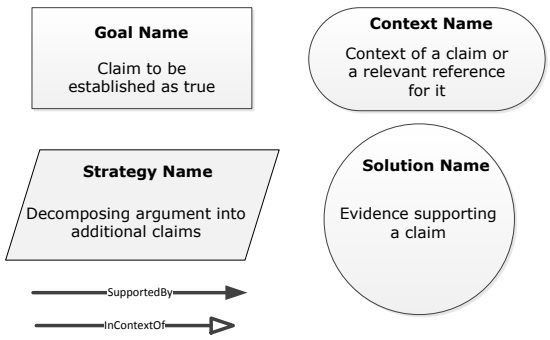


Abbildung 15: GSN Symbole und deren Bedeutung, Symbolik nach [Ori_11]

Zur Illustration beinhaltet Abbildung 16 eine vereinfachte Form eines Safety Arguments in GSN Notation: Es wird behauptet, dass ein bestimmtes System in einer bestimmten Umgebung sicher ist. Dies wird dadurch argumentiert, dass ausreichendes Hazard Management betrieben wurde. Sodann wird dargelegt, dass Hazards identifiziert und auf ein akzeptables Niveau reduziert wurden. Dies wird mittels zweier Dokumente belegt: mit der Hazard Analyse sowie dem Hazard Log.

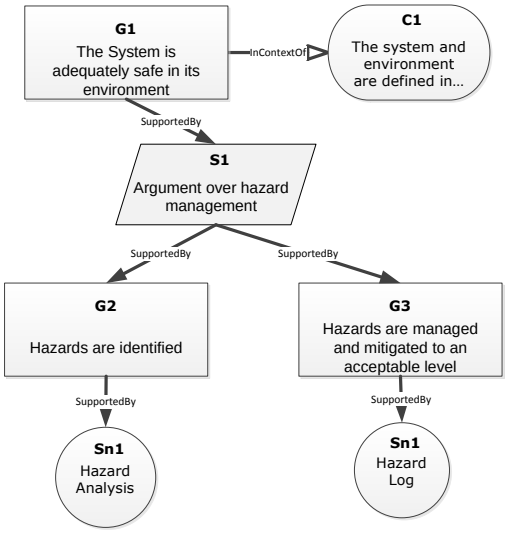


Abbildung 16: Einfache GSN Struktur

Ein Safety Case, der erst gegen Ende des Lifecycles, also kurz vor der Inbetriebnahme bzw. Serienfertigung eines Systems festgemacht wird, stellt ein potentielles Risiko dar. Sollte beispielsweise zu spät erkannt werden, dass für ein schon gewähltes Design ein schlüssiger Safety Case nicht erstellt werden kann, so muss ein Re-Design stattfinden. Aus diesem Grund schlägt Kelly [Kel_98, S. 68] vor, den Safety Case phasenweise zu erstellen, und schon sehr früh im Lifecycle damit zu beginnen. Diese Empfehlung wird vom Safety-Vorgehensmodell umgesetzt: Die Erstellung des Safety Case wird als eine kontinuierliche Aktivität im Laufe des gesamten Lifecycles angesehen. Im Safety-Vorgehensmodell wird eine generische Safety Case Struktur in GSN vorgeschlagen, die im Laufe des Lifecycle sukzessive ausgebaut werden kann.

2.7 Abgrenzung zu vorhandenen Vorgehensmodellen

Der Begriff Vorgehensmodell wird häufig verwendet, eine exakte Definition jedoch oft vermieden. Aus diesem Grund soll der Begriff an dieser Stelle für diese Arbeit definiert werden:

Ein Vorgehensmodell ist eine modellhafte Vorgabe, welche beschreibt, wie zur Lösung eines bestimmten komplexen Problemtyps vorgegangen werden soll. Es beinhaltet unter anderem Phasen, Prozesse, Methoden, Rollen, Techniken und Hilfsmittel, die dabei angewendet werden sollen. Da ein Vorgehensmodell auf eine ganze Klasse von Problemstellungen anwendbar sein soll, ist es immer auf einem höheren Abstraktionsniveau angesiedelt und muss für einen konkreten Fall präzisiert und adaptiert werden.

Es gibt nun zahlreiche Prozesse, Standards und Vorgaben, die dem, was hier unter Vorgehensmodell verstanden wird, nahe kommen. Dieses Kapitel wird auf die wichtigsten davon eingehen und darlegen, welche wesentlichen Eigenschaften das Safety-Vorgehensmodell hat, die in den anderen nicht zu finden sind.

2.7.1 Klassische Vorgehensmodelle

Vom Trend zu immer komplexeren Systemen, den ständig wachsenden und immer schneller ändernden Anforderungen bis hin zu einem unzureichenden Verständnis der Beteiligten für das Zusammenwirken der einzelnen Entwicklungsaufgaben sind unterschiedliche Vorgehensmodelle entwickelt worden (vgl. [LiRo_05]). Speziell im Bereich der Software-Entwicklung ist diese Problemstellung häufig aufgetreten. Aus dieser Sachlage heraus wurde eine Reihe an ingenieursmäßigen Vorgehen entwickelt. Fröhlich (zitiert nach Grechenig [GrBe_09, S. 372]) behauptet 2004, dass die große Anzahl von zum Teil „konkurrierenden" Modellen darauf schließen lässt, dass bisher keine einheitliche Systematik in der Software-Entwicklung durchgesetzt hat.

Mittlerweile hat sich eine Reihe an Vorgehensmodellen durchgesetzt, die nicht nur auf Software-Entwicklungen anwendbar sind, sondern auch auf Systemebene benützt werden. Vorgehensmodelle basieren, wie schon einleitend erwähnt, auf der grundlegenden Idee von Phasen oder Prozessen. Prozesse sind in unterschiedlichster Form in jedem Vorgehensmodell enthalten und zählen als kleinster gemeinsamer Nenner in Vorgehensmodellen. Royce entwickelte 1970 das Wasserfallmodell das eine Abfolge von Prozessen aneinander reiht, wie Anforderung, Analyse, Entwurf, Realisierung, Test und Betrieb. Davon leitet Boehm 1979 [Boe_79, S.711-719], in seiner Arbeit über Verifikation und Validation, das V-Modell ab. Er schaffte damit die Grundlage für heute aktuelle Vorgehensmodelle, wie beispielsweise das V-Modell 97 [Drö_00] oder das V-Modell XT [HöHö_08]. Das V-Modell wird auch im Safety-Bereich von der IEC 61508 [IEC08_10, Teil 3, S.17] vorgeschlagen. Das Vorgehen in der gesamten ISO 26262 [ISO26_10] ist nach dem V-Modell ausgerichtet.

Mit dem V-Modell wird eine zeitliche und inhaltliche Strukturierung von Entwicklungstätigkeiten angestrebt. Es zerlegt das Vorgehen in aufeinander abgestimmte Phasen, für jede Phase werden Tätigkeiten in Form von Prozessen und Ergebnisse definiert. Beim V-Modell nach Boehm (siehe Abbildung 17) wird auf der x-Achse der zeitliche Entwicklungsfortschritt dargestellt und auf der y-Achse der Detaillierungsgrad des Fortschritts. Die eher konstruktiven Maßnahmen, also die Spezifizierungsphasen sind im linken Ast des V dargestellt und münden in der Realisierungsphase. Im rechten Ast des V sind die Integrations- und Testphasen dargestellt. Dadurch, dass den Spezifikationsphasen die Testphasen gegenübergestellt sind, macht das Modell deutlich, dass Fehler auf der jeweiligen Abstraktionsstufe gefunden werden, jedoch zu eher relativ späten Zeitpunkten (vgl. [Mas_07]).

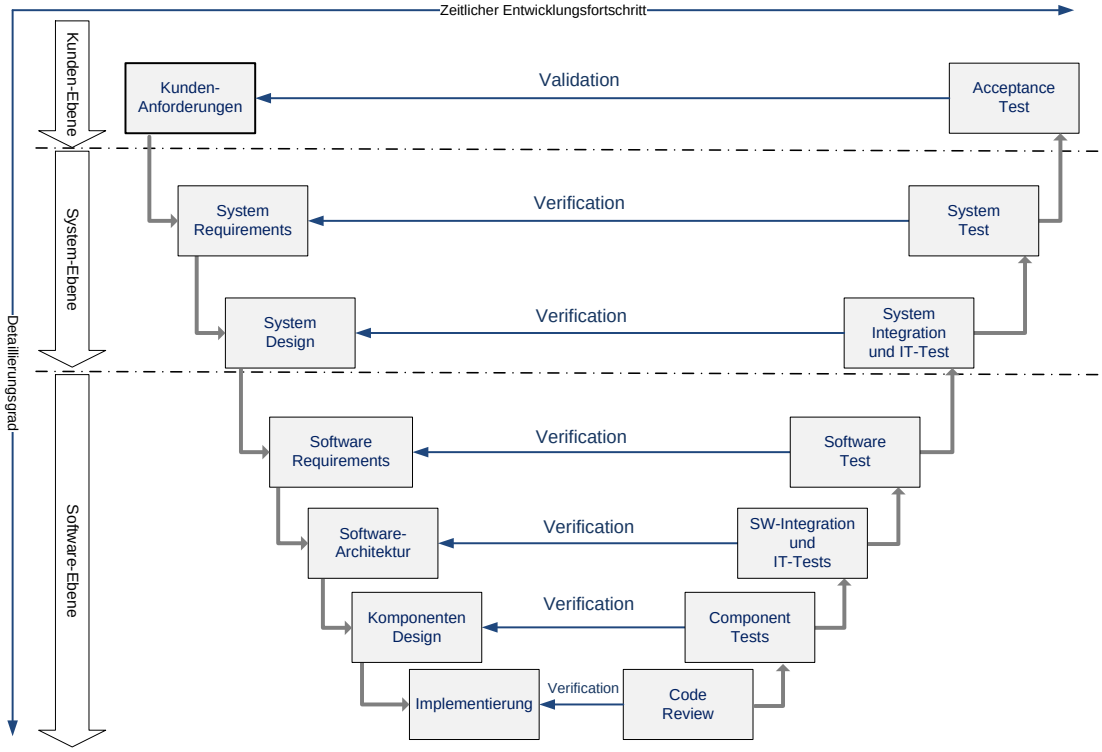


Abbildung 17: Klassisches V-Modell nach Boehm

Das V-Modell startet auf der linken Achse mit den Anforderungen auf Kunden-Ebene und verläuft dann phasenweise, über die System-Ebene bis zur Software-Ebene, nach unten. Auf System-Ebene werden die System Requirements von den Kunden-Anforderungen abgeleitet und darauf basierend das System Design entwickelt. Auf der Software-Ebene wird anhand der Software-Requirements die Software-Architektur entwickelt und danach die einzelnen Komponenten der Architektur spezifiziert. Die linke Achse endet mit der Implementierung. Die zweite Achse spiegelt die Verifikation gegenüber den Spezifikationen anhand der zugeteilten Test-Aktivitäten auf Software- und System-Ebene. Die rechte Achse endet mit dem Acceptance-Test, in dem die Validation der Kunden-Anforderungen durchführt wird.

Im V-Modell sind jedoch nur die Engineering-Phasen dargestellt und wie bereits oben erwähnt werden Fehler erst in den Testphasen aufgedeckt. Auch werden keine Projektmanagement und Safety-Aktivitäten durch das V-Modell abgedeckt. Das V-Modell liefert jedoch eine wichtige Basis für die Herleitung der Engineering-Prozesse im Engineering Lifecycle des ISaPro®, siehe dazu Kapitel 2.3. Das genaue Vorgehen dazu wird im nachfolgenden Kapitel 3 „Safety-Vorgehensmodell“ beschrieben. Hier wird auch die Hardware-Ebene dargestellt, die sich in den klassischen Modellen nicht findet.

Weitere Vorgehensmodelle im Entwicklungsbereich sind das Spiralmodell, das Prototyping-Modell, das Objektorientierte Lebenszyklusmodell und das Extreme-Programming-Prozessmodell aber auch agile Methoden werden dazu gezählt. Diese Modelle sind der Vollständigkeit halber hier erwähnt, da sie jedoch keinen Bezug auf diese Arbeit haben, wird nicht weiter darauf eingegangen.

2.7.2 Prozessreifegradmodelle

Das seit Beginn der 1990er Jahre bekannte CMMI (Capability Maturity Model Integration) [CMMI_11] Modell ist eine Sammlung an "best practices", die es ermöglichen sollen, reife Prozesse[44] in Organisationen zu etablieren. Es beinhaltet auch eine Methode, wie man die "Reife" einer Organisation in Bezug auf ihre Prozesse bewerten kann – dies war eine der ersten Anforderungen an das Modell, um Organisationen in eine von 5 Stufen ("Maturity Levels") einzuordnen. Ursprünglich war das Modell auf Softwareentwicklung fokussiert, das aktuelle Modell lässt sich jedoch auf Entwicklungsprozesse im Allgemeinen anwenden. Außerdem wurden auch Modelle für Beschaffung und Serviceerbringung definiert. Es handelt sich dabei jedoch nicht um ein Vorgehensmodell im oben beschrieben Sinn, sondern eher um eine umfangreiche, aber relativ lose zusammenhängende Sammlung an Prozessen. CMMI beinhaltet auch keinen Fokus auf sicherheitskritische Systeme.

Das "International Council on Systems Engineering" (INCOSE) publiziert ein "Systems Engineering Handbook" [INC_04], in dem die für die Systementwicklung relevanten "key process activities" beschrieben werden. Es handelt sich dabei um eine sehr ausführliche Beschreibung zahlreicher Prozesse, inklusive ihrer Schnittstellen zu anderen Prozessen. Die beschriebenen Prozesse fokussieren zwar auf den technischen Aspekt, beinhalten aber sehr wohl auch Projektprozesse, wie Planung und Risikomanagement. Prozesse für die Entwicklung sicherheitskritischer Systeme sind jedoch nicht enthalten.

Ähnlich verhält es sich mit Standards zur System- bzw. Softwareentwicklung, wie ISO/IEC 15504 [ISO15_06], ISO 15288 [ISO88_08] und ISO 12207 [ISO07_07]. Hierbei handelt es sich wiederum um eine Sammlung von Prozessen, die im Detail beschrieben sind, deren Zusammenhänge aber nicht in beispielsweise einem Lebenszyklusmodell dargelegt sind. So steht in der ISO 12207 [ISO07_07, S.12] beispielsweise "This International Standard does not require the use of any particular life cycle model."[45] Der Grund, warum der Standard keine Abfolgen von Prozessen definiert, wird mit einem Mangel an Konsens argumentiert: "For lack of consensus on or use of a universal time-dependent sequence, the user of this standard may select and order the processes, activities, and tasks as appropriate and effective."[46] Der Standard definiert also wiederum kein Vorgehensmodell im oben erwähnten Sinn. Außerdem handelt es sich wiederum um einen Standard, der nicht explizit auf sicherheitsrelevante Entwicklungsvorhaben eingeht.

---

[44] Ein Prozess ist ein Satz zusammenhängender Tätigkeiten, die Eingaben in Ausgaben umwandeln, um einen vorgegebenen Zweck zu erreichen [CMMI_11].

[45] "Dieser internationale Standard fordert nicht die Verwendung eines bestimmten Lebenszyklus-Modelles." (Übersetzung des Verfassers)

[46] "Durch den Mangel an einem Konsens über die Verwendung einer universellen zeitabhängigen Reihenfolge, darf der Benutzer des Standards die Reihenfolge der Prozesse, Aktivitäten und Aufgaben so auswählen, wie sie sinnvoll und effektiv sind." (Übersetzung des Verfassers)

2.7.3 Vorgehensmodelle auf wissenschaftlicher Ebene

Auf wissenschaftlicher Ebene gibt es auch nur einige Ansätze, integrierte Vorgehensmodelle zu definieren. Im Folgenden wird kurz auf die Vorgehensmodelle von Catherine Menon [Men_10], Brahim Hamid [Ham_12] und Yae Han Yoon [Yoo_08] eingegangen.

Ein Anlass, solche Vorgehensmodelle zu entwickeln ergeben sich, wenn ein Standard nicht spezifisch genug ist und keine klare Vorgehensweise vorgibt. So verhält es sich mit der letzten Version (Issue 4) des englischen Safety Management Standard aus dem Militärbereich (DS 00-56, [DS0_07]). Dieser Standard gibt vor *was* erforderlich ist, aber nicht *wie* es zu tun ist. Aus diesem Grund wird in Menon [Men_10] ein solches konkretes Vorgehen vorgeschlagen, welches das "wie" definiert. Es wird ein "Standard of Best Practice" (SoBP) beschrieben, wie man sicherstellen kann, dass die Software zum oben genannten Standard "compliant" ist.

In diesem Standard of Best Practice wird insofern ein ähnlicher Ansatz wie in der hier vorliegenden Arbeit gewählt, dass die Management Prozesse und die technischen Prozesse sowie deren Interaktionen dargestellt und beschrieben werden. In sogenannten Swimlane Diagrammen werden die Aktivitäten dieser Prozesse im Detail beschrieben. Der Fokus liegt jedoch ausschließlich auf der Einhaltung des Standards DEF STAN 00-56 [Def56_07]. Die vorliegende Arbeit schlägt eine abstrahierte Vorgehensweise mit Hilfe des "System Safety" Ansatzes vor, sodass das Safety-Vorgehensmodell auf mehrere Safety-Standards anwendbar ist.

Im Vorgehensmodell von Hamid [Ham_12] wird ein Versuch, den Safety-Prozess in den Software-Entwicklungsprozess zu integrieren, präsentiert. Der integrierte Prozess wurde dann anhand einer Case Study aus dem Eisenbahnbereich validiert. Das Prozess-Metamodell ist anwendbar auf Embedded Systeme und legt einen großen Fokus auf die Modellierung des Systems. Die organisatorischen Aspekte, wie z. B. das Projektmanagement werden jedoch nicht berücksichtigt. Das in dieser Arbeit vorgestellte Safety-Vorgehensmodell integriert zusätzlich noch diese organisatorischen Aspekte.

Yae Han Yoon [Yoo_08] geht einen ähnlichen Weg, der hier vorgestellte Prozess ist jedoch nicht nur für kleinere Embedded Systeme geeignet, sondern für "complex large-scale systems with safety-critical requirements." ("komplexe große Systeme mit sicherheitskritischen Anforderungen"). Es wird ein integrierter Prozess entwickelt, der einen standardisierten Systems Engineering-Prozess mit Hazard-Analyse-Techniken nach Ericson [Eric2_05] kombiniert. Interessant ist hier, dass explizit – wie in der vorliegenden Arbeit – der "System Safety" Ansatz gewählt wird. Die Integration umfasst aber wieder nur die technischen Prozesse und die Safety-Prozesse. Die organisatorischen und die Projektmanagement-Aspekte werden wieder nicht miteinbezogen.

Keines der oben genannten Vorgehensmodelle enthält jedoch Ansätze, wie man System Accidents verhindern kann. Beginnend bei der Umfeldbetrachtung des Systems, um Flawed-Requirements zu verhindern, über erweiterte Analyse-Methoden bis hin zur speziellen Analysen mit denen man interne Systemzustände prognostizieren kann. Und keines der erwähnten Vorgehensmodelle verfolgt den „Inherent System Safety Approach“.

3. Safety-Vorgehensmodell

Beim Safety-Vorgehensmodell steht der systematisch-methodische Ablauf im Mittelpunkt, der bei der Konzeption und Entwicklung von sicherheitskritischen Systemen erforderlich ist. Bei Projektvorhaben die sich mit der Entwicklung von sicherheitskritischen Systemen auseinander setzen, sind die Konzeptionierung und die Planung die beiden wichtigsten Phasen. Damit wird der Grundstein für die Sicherheit solcher Systeme gelegt. Bereits in der Vor-Projektphase müssen wichtige Entscheidungen getroffen werden, wie das zu entwickelnde System zu konzipieren ist und welche Aufwände dafür notwendig sind. Werden hier Fehlentscheidungen getroffen, so wirkt sich das sofort auf falsch veranschlagte Projektkosten aus, die man im Projektverlauf meist nicht mehr nachverhandeln kann. Zu geringe Projektkosten gehen meist auf das Konto der Qualität und Sicherheit.

Im beauftragten Projekt wird in der ersten Projektphase die Entwicklung eines inhärent sicheren System Design sowie die Planung zur Realisierung des Projekts gefordert, bevor das Projekt in die Realisierungsphase geht. Hier wird nicht nur über Erfolg oder Misserfolg des Projekts entschieden, sondern auch über ein „Safe System“ oder ein „Un-Safe System“. Solche „Unsafe Systems“ können für die Gesellschaft und Umwelt verheerende Folgen haben, wie schon im Einführungkapitel aufgezeigt wurde. Um das geforderte inhärent sichere System Design klar abgrenzen zu können und die Verwechslung mit einem inhärenten System Design auf Basis von Security zu vermeiden, wird in diesem Dokument an geeineter Stelle der Begriff „Inherent Safe System Design“ verwendet. Wird der Begriff System Design oder kurz SyD verwendet, ist immer ein Inherent Safe System Design damit gemeint.

Da der Vor-Projektphase und der Planungsphase in der Vergangenheit noch wenig wissenschaftliche Aufmerksamkeit geschenkt wurde und diese ausschlaggebend für die Entwicklung eines Inherent Safe System Designs ist, widmet sich diese Arbeit ausschließlich diesen beiden Phasen. Die Vor-Projektphase wird als „Problemraum“ bezeichnet, die Planungsphase als „Modellierungsraum“. Dies begründet sich darin, da die beiden Phasen mehrere Disziplinen abdecken und nach der Struktur des ISaPro® aufgebaut sind (siehe dazu Kapitel 2.3 „Integrativer Safety Prozess (ISaPro®)“). Die Projekt-Realisierungsphase wird hier als „Lösungsraum“ bezeichnet. Die Inhalte des Lösungsraums werden nur in der Projektplanung dargestellt.

Um nun ein Inherent Safe System Design entwickeln zu können, ist ein Umdenken im Vorgehen bestehender Safety-Normen notwendig. Das Safety-Vorgehensmodell fordert hier eine ganzheitliche Struktur, um Lücken im Arbeitsablauf zu erkennen und vermeiden zu können. Mit reifen Prozessen,

wie sie der ISaPro® zur Verfügung stellt, wird nicht nur der Arbeitsablauf systematisiert, es wird auch die Arbeitsweise in der Projektorganisation vereinheitlicht und systematischen Fehlern entgegengewirkt. Damit wird bereits ein wesentlicher Beitrag zur Vermeidung von „Process Flaws" geleistet.

Um jedoch ein Inherent Safe System Design entwickeln zu können, sind darüber hinaus noch weitere Aspekte zu erfüllen. Und hier unterscheidet sich das Safety-Vorgehensmodells von anderen Vorgehensmodellen und bestehenden Safety-Normen im Besonderen. Denn etwas schwieriger ist die Sachlage auf der System-Ebene. Es sollen nach Möglichkeit alle Hazards identifiziert werden, sowohl endogene als auch exogene Hazards. Hier ist eine durchdachte Systematik unabdinglich, die einerseits die System-Umgebung sowie das System-Umfeld ausreichend identifiziert und andererseits darauf basierend alle exogenen Einflüsse analysiert. Die dabei identifizierten exogenen Hazards sollen nun, im Sinne der *„Hazard Avoidance"*, vermieden werden, um den Grundgedanken des Inherent System Safety Approach zu folgen.

Diejenigen Hazards die nicht vermieden werden können, müssen einem Safety-Risiko-Assessment unterzogen werden. Wird das Risiko als nicht-zulässig eingeschätzt, so sind nun Design-Lösungen gefordert, welche das Auftreten von Hazards in gefährlichen Situationen *verhindern*. Dieses Vorgehen wird in dieser Arbeit mit *„Hazard Prevention"* bezeichnet. Für Hazards die nicht vermieden oder verhindert werden können, wird erst jetzt die Philosophie der Funktionalen Sicherheit verfolgt. Dabei wird eine Sicherheitsfunktion (Safety Function) spezifiziert, die das System in einen definierten sicheren Zustand (Safe State) führt. Hier wird der Begriff des *„Hazard Controlling"* verwendet.

Grundsätzlich ist an dieser Stelle noch zu erwähnen, dass das „Hazard- and Risk Assessment" der Safety-Normen keine Hazards vermeidet, sondern hilft das Risiko der definierten Hazards einzuschätzen. Bei zu hohem Risiko müssen vom Anwender notwendige Gegenmaßnahmen eingeleitet werden, um dieses Risiko auf einen akzeptablen Risiko-Level minimieren zu.können

Betrachtet man nun, bezogen auf den Inherent System Safety Approach, die IEC 61508 bzw. die modernere ISO 26262 etwas näher, so müssen an dieser Stelle noch weitere wesentliche Teilaspekte diskutiert werden, die für das Safety-Vorgehensmodell wichtig sind.

Betrachtet man die Safety Lifecycle-Modelle der beiden Normen, so wird der Anspruch an ein ganzheitliches Vorgehen nur sehr bedingt erfüllt. In den Normen werden zwar die wesentlichen Anforderungen beschrieben – was zu tun ist um ein sicheres System zu entwickeln. Sie beschreiben jedoch nicht, wie dabei vorzugehen ist.

Die ISO 26262 startet in ihrem Lifecycle Modell mit der Concept Phase [ISO26_11, Teil 3] und spricht hier von der Item Definition, der Initialisierung des System Safety Lifecycles, der Hazard Analysis und dem Risk Assessment. Die Norm beschreibt jedoch nicht, wie das Item-Umfeld systematisch identifiziert wird und wie darauf basierend Safety-Analysen durchgeführt werden, um exogene Hazards zu vermeiden. Sie spricht von technischen Safety Requirements, aber nicht, wie im Zusammenhang mit dem System-Umfeld die funktionalen-, nicht-funktionalen und Safety Requirements hergeleitet werden. Sie beschreibt auch kaum den Umgang mit nicht-Safety-relevanten Requirements, und wie diese mit den Safety Requirements in Beziehung stehen. Das sind

jedoch wichtige Aspekte, um das Problem der „Flawed Requirements“ und damit den unvorhergesehenen und nicht zulässigen inhärenten Systemzuständen entgegenzuwirken, wie in der Problemstellung in Kapitel 1.1 beschrieben ist.

Des Weiteren fordern die Normen in ihren Safety Lifecycles nur eine einzige Hazard Analysis und ein Risk Assessment. Dabei stellt sich die Frage, ob eine Hazard Analysis und ein Risk Assessment über den gesamten Lebenszyklus ausreichen. Auch Storey [Sto_96] bemängelt schon 1996 das Lifecycle-Modell der IEC 1508, Vorgänger der IEC 61508, mit „For simplicity, the model shows hazard and risk analysis as a single phase near the beginning of the project. […] we know that hazard and risk analysis is a continuing activity that goes on throughout the development process.“[47] Es werden in den Normen zwar weitere Analyse-Methoden vorgeschlagen, jedoch nicht wann und wo genau diese anzuwenden sind. Es wird auch nicht beschrieben, welche Analyse-Methoden auf Design-Ebene wie anzuwenden sind, um unzulässige System-Zustände soweit als möglich zu vermeiden, womit die gefährlichen „System Accidents“ verhindert werden können.

Im Teil 6 der ISO 26262 [ISO26_11] wird beschrieben, dass beim Designen systematische Fehler zu vermeiden sind. Es ist jedoch nicht beschrieben, wie in diesem Zusammenhang andere Disziplinen zu berücksichtigen sind, wie beispielsweise die Wechselwirkungen von den Engineering-Aktivitäten zu den Safety-Aktivitäten bzw. zu den Management-Aktivitäten dabei vorzugehen ist. Dadurch können Gesamtzusammenhänge nicht erkannt werden und so entstehen potentiell Verluste an den organisatorischen sowie an den technischen Schnittstellen, die wiederum in „Flawed Requirements“ münden. Darüber hinaus stellt sich in der Praxis noch die Frage nach der Aufgabenteilung; wessen Aufgabe ist beispielsweise die Konzeptentwicklung oder wer plant die Safety und die Nicht-Safety-Aktivitäten?

Last but not least geht die ISO 26262 nicht darauf ein, wie nicht-safety-relevante Aktivitäten richtig geplant werden, um systematische Fehler in der Realisierungsphase zu vermeiden. Einen wesentlichen Beitrag zur Vermeidung von systematischen Fehlern im Software-Bereich, liefert der in der Automotive-Industrie geforderte Standard Automotive SPICE (ASPICE), eine domänenspezifische Variante des internationalen Standards ISO 15504. Dieser Standard wird jedoch in der ISO 26262 nicht berücksichtigt. Ein ganzheitlicher Ansatz ist jedoch notwendig, damit die organisatorische Komplexität über den gesamten Entwicklungs-Lifecycle strukturiert plan- und steuerbar wird, um hier Lücken im Vorgehen (Process Flaws) zu vermeiden und der technischen Komplexität entsprechen zu können.

Das Safety-Vorgehensmodell widmet sich den angesprochenen Problemstellungen und liefert damit die Grundvoraussetzungen zur *Vermeidung von „System Accidents“*. Das Konzept des systematisch-methodischen Vorgehens, welches in den nachfolgenden Kapiteln dargestellt ist, bildet das Fundament zur *Vermeidung von „Flawed Requirements“* und von nicht-prognostizierbaren, potentiell gefährlichen System-Zuständen. Ziel des Safety-Vorgehensmodells ist, die Entwicklung des Inherent Safe System Designs. Darüber hinaus soll darauf basierend eine *vollständige Planung*

[47] Da Modell zeigt der einfachheithalber die Hazard- und Risiko-Analyse als eine einmalige Phase, am Beginn eines Projekts […] es ist jedoch bekannt, dass die Hazard- und Risiko-Analyse eine kontinuierliche Aktivität, über den gesamten Entwickluns-Lifecycle darstellt. (Übersetzung des Verfassers)

zur Realisierung sichergestellt werden. Der Sicherheitsnachweis soll dabei aufgesetzt und systematisch weiterentwickelt werden. Das Safety-Vorgehensmodell verfolgt dabei den *„Inherent System Safety Approach“*.

Das Safety-Vorgehensmodell stellt den Anspruch des systematisch-methodischen, disziplinenübergreifenden Vorgehens bei der Entwicklung von technischen Systemen, wie sie Safety-Normen so nicht darstellen. In den nachfolgenden Kapiteln 3.1 „Problemraum“ und 3.2 „Modellierungsraum“ wird bei der Beschreibung des Vorgehens immer wieder auf die Basis-Prozesse des ISaPro® und deren Basis-Aktivitäten verwiesen. Die Prozesse für den Problem- und Modellierungsraum sind im Anhang B „ISaPro® Prozesse“ detailliert dargestellt.

Die Verlinkung zu den Prozessschritten ist mit einer Abkürzung des Prozess-Namens und der jeweiligen Basis-Aktivität dargestellt. Das folgende Beispiel mit [PjP_BA1] soll zeigen, wie es zu lesen ist: „PjP“ steht für den Projektplanungs-Prozess und „BA1“ für die Basis-Aktivität 1. Die Basis-Aktivitäten in den ISaPro®-Prozessen sind dann noch auf die jeweilige Norm bzw. Standard verlinkt. Damit kann man von den Durchführungsaktivitäten im Vorgehen über die ISaPro®-Prozesse bis zu den Safety-Normen und anderen verwendeten Standards zurückfinden.

3.1 Problemraum

Der Problemraum stellt die Vor-Projektphase dar. Der Zweck des Problemraums ist die Erstellung eines Makro-Konzepts, um darauf basierend die Projektaufwände so genau wie möglich einschätzen und planerisch darstellen zu können. Das Makro-Konzept ist eine erste gesamtheitliche Darstellung des Projektvorhabens, bestehend aus dem organisatorischen, dem technischen und dem Safety-Konzept, siehe dazu Abbildung 18. Auf der organisatorischen bzw. projektmanagement Ebene wird vorerst das Projekt initialisiert (Prozess Projekt-Initialisierung), d.h. das Projektvorhaben wird strukturiert und umfassend abgebildet. Um den Umfang des Projekts auf der technischen Ebene besser verstehen und bewerten zu können wird ein technisches Konzept (Prozess Konzeptionierung) entwickelt.

Die IEC 61508 fordert beispielsweise im Kapitel „Concept“ [IEC08_10, Teil 1, Kapitel 7.2] sowie die ISO 26262 im Kapitel „Item definition“ [ISO26_11, Teil 3, Kapitel 5] ein definiertes und dokumentiertes technisches Konzept des zu entwickelnden Systems. Dabei soll das System-Umfeld mitgedacht werden, damit die systemexternen Einflüsse auf das zu entwickelnde System mitbetrachtet werden können. Das *technische Konzept* ist ein erster technischer Entwurf des Systems, basierend auf den Kunden-Anforderungen sowie den identifizierten Rahmenbedingungen aus dem System-Umfeld. Damit sind die wichtigsten Voraussetzungen zur Durchführung der ersten Safety-Analyse in der PHI (Prozess Preliminary Hazard-Identifikation) erfüllt.

In der PHI werden danach, basierend auf dem technischen Konzept, die Top-Level Hazards (TLHa) identifiziert und eine Safety-Risiko-Bewertung durchgeführt. Gefordert wird dies von der IEC 61508 im Kapitel „Hazard and Risk Analysis“ [IEC08_10, Teil 1, Kapitel 7.4.2] sowie von der ISO 26262 im Kapitel „Hazard Analysis and Risk Assessment“ [ISO26_11, Teil 2, Kapitel 5.2.2, Punkt c].

Darauf basierend wird die *Sicherheitsanforderungsstufe* bestimmt sowie die Safety Requirements davon abgeleitet. Die Ergebnisse dieser Analysen liefern die Basisinformationen für das zu erstellende Safety-Konzept, welches von der ISO 26262 im Kapitel „Functional Safety Concept“ [ISO 26_11, Teil 2, Kapitel 5.2.2, Punkt d] gefordert wird. Dies ist nötig, da zum Beispiel eine nicht bekannte Sicherheitsanforderungsstufe beträchtliche Auswirkungen auf die Projektaufwände haben kann: Die Freigabe eines safety-relevanten technischen Systems auf einer höheren Sicherheitsanforderungsstufe erfordert beispielsweise größere Unabhängigkeit bei den Rollen im Projekt und mehr Überprüfungen als eine niedrigere Sicherheitsanforderungsstufe (siehe dazu auch Kapitel „Confirmation Measures: Types, independency and authority“ in der ISO 26262 [ISO26_11, Teil 2, Kapitel 6.4.7]), aber auch rigorosere Anforderungen an die Entwicklungs-Aktivitäten. Mit den Ergebnissen des organisatorischen, des technischen und des Safety-Konzepts können die Aufwände für das Projektvorhaben entsprechend abgeschätzt werden.

Abbildung 18 zeigt den Problemraum und dessen drei Prozesse: Projekt-Initialisierung, Preliminary Hazard Identification (PHI) und Konzeptionierung. Die Projekt-Initialisierung ist Teil des Projektmanagement-Lifecycles und hat als Ergebnis das organisatorische Konzept. Die PHI ist Teil des System Safety Lifecycle und liefert als Output das Safety-Konzept. Das technische Konzept ist das Resultat des Konzeptionierung-Prozesses, der Teil des Engineering Lifecycles ist. Alle drei Arbeitsprodukte werden zum Makro-Konzept zusammengefasst. Das Makro-Konzept gibt einen ganzheitlichen Überblick über das gesamte Projektvorhaben.

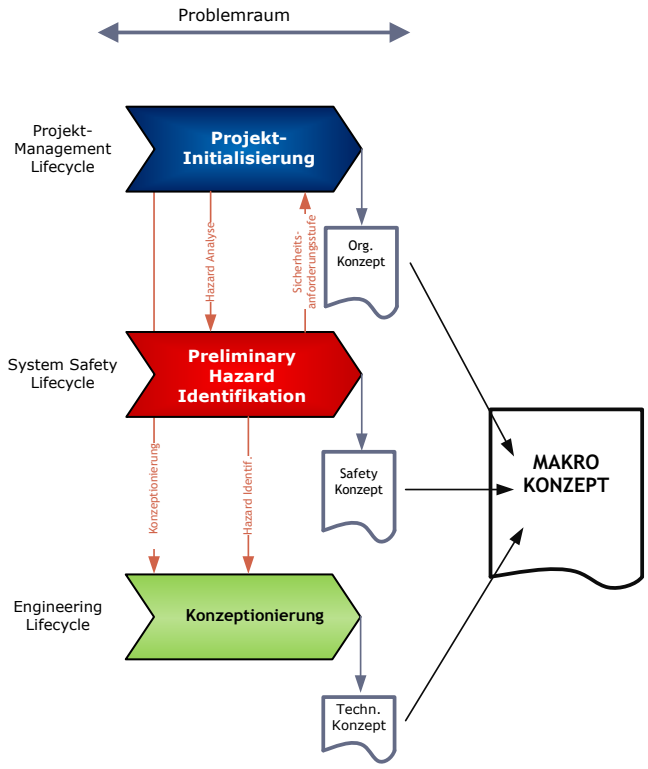


Abbildung 18: Prozesse im Problemraum

Des Weiteren sind in Abbildung 18 die Wechselwirkungen der Prozesse grob dargestellt. Das ist im ersten Schritt die Anweisung zur Erstellung eines technischen Konzepts. Im zweiten Schritt ist es die Anweisung zur Durchführung einer Hazard-Identifikation und der Safety-Risiko-Bewertung, basierend auf dem technischen Konzept. Rückgemeldet wird die Sicherheitsanforderungsstufe an das Projektmanagement.

Diese Vorgehensweise bietet den Vorteil, dass bereits in der Vor-Projektphase Hazards vermieden werden können. Falls das nicht möglich ist, muss Sicherheitsanforderungsstufe identifiziert, die notwendigen Anforderungen der jeweiligen Safety-Norm im technischen Konzept berücksichtigt und im organisatorischen Konzept geplant werden. Damit können die Projektkosten in der Vor-Projektphase sehr genau eingeschätzt werden.

Der zeitliche Ablauf des Problemraums wird in Abbildung 19 dargestellt. Auf der Zeitskala werden die Prozesse der Projekt-Initialisierung, die Preliminary Hazard-Identifikation (PHI) und die Konzeptionierung parallel positioniert. Die nicht unterbrochenen waagrechten Pfeile symbolisieren die für den aktuellen Schritt verantwortlichen Disziplinen, das heißt, für die Projekt-Initialisierung ist das Projektmanagement verantwortlich, für die Konzeptionierung das Engineering und für die PHI das System Safety Engineering. Die unterbrochenen, waagrechten Pfeile symbolisieren, dass die Mitarbeit der anderen Disziplinen gefordert ist und die punktierten senkrechten Pfeile zeigen die Schnittstellen/Wechselwirkungen zu anderen Disziplinen.

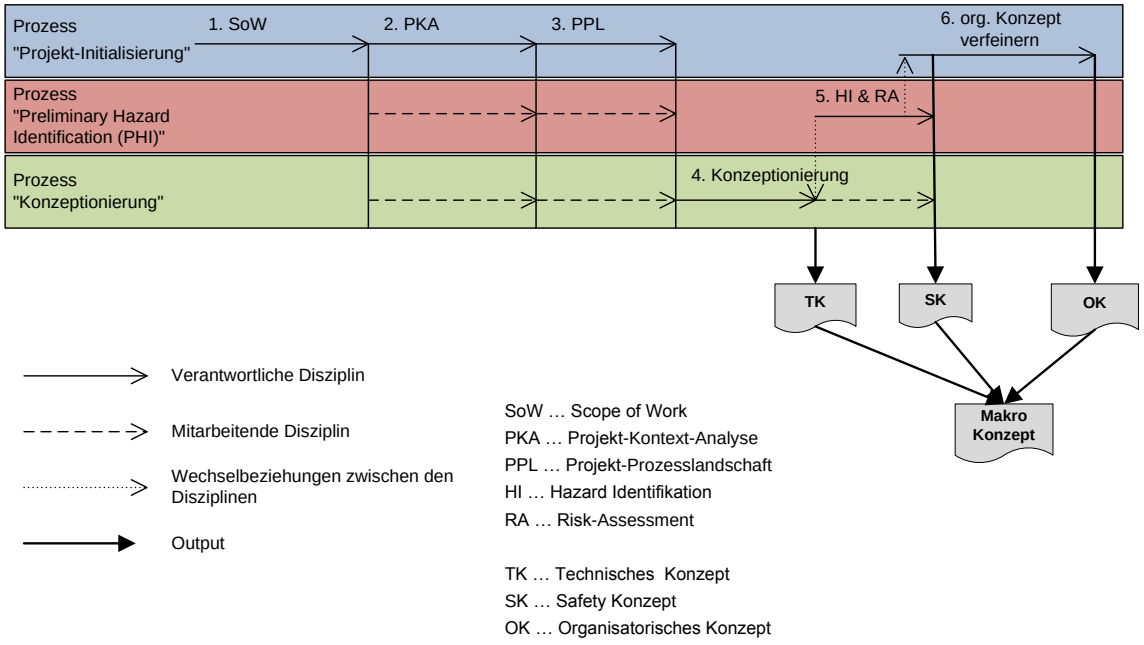


Abbildung 19: Zeitlich-systematischer Ablauf des Problemraumes

Beginnend beim Prozess der Projekt-Initialisierung auf Projektmanagement-Level wird im ersten Schritt das „Scope of Work“ (SoW) erarbeitet, in dem der Projektumfang auf Top-Level Ebene dargestellt wird. Sollten die Anforderungen an das zu entwickelnde System noch nicht eindeutig veranschaulicht werden können, müssen entsprechende Annahmen getroffen werden, wie es beispielsweise auch in der ISO 26262 [ISO 26_11, Teil 3] gefordert wird.

Im zweiten Schritt wird die „Projekt-Kontext-Analyse“ (PKA) durchgeführt, in der die beeinflussenden Faktoren auf das Projektvorhaben identifiziert werden. Die Informationen aus der PKA liefern die Basis für den Schritt drei, in dem die „Projekt-Prozesslandschaft“ (PPL) entwickelt wird. Alle drei Teilergebnisse sind Bestandteil des organisatorischen Konzepts und liefern bis zu diesem Zeitpunkt einen ersten „Draft“. Im Schritt 4 wird darauf basierend das technische Konzept erstellt. Das technische Konzept definiert das zu entwickelnde System, grenzt es von seiner Umgebung ab und betrachtet alle Aspekte der System-Umbebung und des System-Umfeldes. Das technische Konzept bildet außerdem die Basis für die Safety Engineering Aktivitäten im Schritt 5, welche im Prozess der PHI definiert sind. Die Durchführung der „Hazard-Identifikation“ (HI) und des „Risiko-Assessments“ (RA) wird in Abbildung 19 mit HI&RA bezeichnet. Je mehr Detailinformationen im technischen Konzept erarbeitet werden, desto ausführlicher kann die Hazard-Identifikation durchgeführt werden. Damit lässt sich im Safety-Risiko-Assessment das Risiko genauer bewerten und die Sicherheitsanforderungsstufe exakter bestimmen. Die Sicherheitsanforderungsstufe kann auf das Safety-Konzept, das technische Konzept und auf die Planung wesentlichen Einfluss haben. Nach einer erforderlichen Überarbeitung des technischen Konzepts kann im Schritt 6 das organisatorische Konzept vervollständigt werden. Das organisatorische-, das technische- und das Safety-Konzept werden danach zusammengefasst und im Makro-Konzept dargestellt.

Die nachfolgenden Unterkapitel beschreiben das systematisch-methodische Vorgehen im Detail. Dabei werden durch das systematische Vorgehen auch die Wechselwirkungen zwischen den Prozessen Projektmanagement, Engineering und Safety stark betont. Hier handelt es sich also keinesfalls um separat abgeschlossene Prozesse, wie das die Abbildung 18 möglicherweise vermuten lässt, sondern ganz im Gegenteil, das Safety-Vorgehensmodell betont gerade diese „*Verzahnungen*“ der einzelnen Prozesse.

3.1.1 Projekt-Initialisierung

Ziel der Projekt-Initialisierung ist die *ganzheitliche Betrachtung* des Projektvorhabens auf organisatorischer Ebene; aus geschäftlicher Sicht handelt es sich um die Ermittlung der Projektkosten und Projektdauer. Diese ganzheitliche Betrachtung ist notwendig, um alle erforderlichen Aktivitäten (Safety und nicht-Safety) über den gesamten Lebenszyklus des Systems verstehen und planen zu können. Das Ergebnis aus dieser Phase ist das organisatorische Konzept. Das organisatorische Konzept ist ein wesentlicher Bestandteil des Makro-Konzepts.

Mit Hilfe des Makro-Konzepts kann die organisatorische und technische Machbarkeit eingeschätzt werden. Die Einschätzung der Machbarkeit fordert die ISO 26262 im Kapitel „Competence Management“ [ISO26_11, Teil 2 Kapitel 5.4.3]. Das heißt, die erforderliche technologische Herausforderung sowie die benötigten Kompetenzen und Qualifikationen der Projekt-Organisation können mit dem Safety-Vorgehensmodell schon in der Vor-Projektphase gut eingeschätzt werden.

Die Projekt-Initialisierung ist in sechs grundlegende Basis-Aktivitäten untergliedert. Die ersten drei Basis-Aktivitäten liefern mit der Definition des Scope of Work (SoW) [PjI_BA1], der Projekt-Kontext-Analyse (PKA) [PjI_BA2] und der Erstellung der Projekt-Prozesslandschaft (PPL)

[PjI_BA3] das Fundament für das organisatorische Konzept. Diese Basis-Aktivitäten sind Voraussetzung für die Beauftragung zur Konzeptionierung und für die Beauftragung der PHI [PjI_BA4]. Die Ergebnisse aus der Konzeptionierung und der PHI fließen in das organisatorische Konzept (bestehend aus SoW, PKA und PPL) [PjI_BA5] ein. Darauf basierend kann das Makro-Konzept erstellt [PjI_BA6] und freigegeben werden. Die nachfolgende Tabelle 3 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im Prozess der Projekt-Initialisierungs gefordert sind.

PjI_BA1	Scope of Work (SoW) definieren
PjI_BA2	Projekt-Kontext-Analyse (PKA) durchführen
PjI_BA3	Erstansatz der Projekt-Prozesslandschaft (PPL) erstellen
PjI_BA4	Konzeptionierung und PHI beauftragen/veranlassen
PjI_BA5	Organisatorisches Konzept erstellen
PjI_BA6	Makro-Konzept erstellen und freigeben

Tabelle 3: Basis-Aktivitäten der Projekt-Initialisierung

Scope of Work (SoW) definieren [PjI_BA1]

Die erste Basis-Aktivität der Projekt-Initialisierung [PjI_BA1] besteht aus 7 Teilschritten:

1. Projektbeschreibung erstellen [PjI_BA1.1]
2. Projekt-Zweck definieren [Pj_BA1.2]
3. Operational Requirements spezifizieren [PjI_BA1.3]
4. Definierte(s) Lieferobjekte definieren [PjI_BA1.4]
5. Projekt-Ziele definieren [PjI_BA1.5]
6. Projekt-Nicht-Ziele definieren [PjI_BA1.6]
7. Hauptaufgaben im Projekt festlegen [PjI_BA1.7]

Im ersten Prozessschritt, wie in Abbildung 20 dargestellt, wird der Projektumfang anhand des „Scope of Work (SoW)“ dargestellt. Die Erarbeitung des SoW beginnt mit der Projektbeschreibung [PjI_BA1.1]. In der Projektbeschreibung werden der Zweck [PjI_BA1.2] des zu entwickelnden Systems sowie die betrieblichen Anforderungen (Operational Requirements) [PjI_BA1.3] spezifiziert. Die genaue Definition des am Ende abzuliefernden Systems, also das Lieferobjekt [PjI_BA1.4], konkretisiert das Projekt-Ergebnis. Mit einer eindeutigen Definition der Projekt-Beschreibung wird auch der Nutzen des Systems hervorgehoben, welcher ein wichtiges Kriterium für die Validierung der Projektergebnisse darstellt. Die Projekt-Ziele [PjI_BA1.5] werden von der Projektbeschreibung abgeleitet. Die Projekt-Nicht-Ziele [PjI_BA1.6] dienen der genauen Abgrenzung zu dem, was realisiert wird und was nicht realisiert wird. Nach Patzak [Pat_04, S. 90] grenzen Nicht-Ziele die Graubereiche eines Projekts aus und schützen vor Nachforderungen des

Projektauftraggebers. Um die Projekt-Ziele erfüllen zu können, müssen entsprechende Leistungen, in dieser Arbeit nach Patzak [Pat_04, S. 93] als Hauptaufgaben [PjI_BA1.7] bezeichnet, festgelegt werden. Der Projektstart- und Projektend-Termin werden am Ende der Projekt-Initialisierung fixiert, nachdem die Ergebnisse der Konzeptionierung und der PHI ausgeführt und die Abschätzung der erforderlichen Leistungen durchgeführt wurden. Die nachfolgende Abbildung 20 zeigt die Vorlage für ein SoW.

	Scope of Work	
Projektbeschreibung: *Kurze prägnante Beschreibung des Projektvorhabens* Projektzweck: *Begründung des Projekts* Betriebliche Anforderungen / Operationale Requirements: Lieferobjekt:		
Projektstarttermin:		Projektendtermin:
Projektziele:		Nicht-Projektziele:
Hauptaufgaben:		Projekt-Aufwände: Projekt-Kosten:
Projekt-Risiken:		

Abbildung 20: Scope of Work – Template

Die Projekt-Aufwände und Projekt-Kosten sowie die Projekt-Risiken werden als State-of-the-Art betrachtet, daher wird an dieser Stelle nicht näher darauf eingegangen.

Projekt-Kontext-Analyse (PKA) durchführen [PjI_BA2]

Im darauffolgenden Prozessschritt wird eine Projekt-Kontext-Analyse (PKA) durchgeführt, bei der alle Einflüsse auf das Projektvorhaben betrachtet werden. Dies empfiehlt auch die ISO 26262 [ISO26_11, Teil 3 Kap. 5.3.2] im weitesten Sinne: “any information that already exists concerning

the item, e.g. a product idea, a project sketch, relevant patents, the results of pre-trials, the documentation from predecessor items, relevant information on other independent items."[48]

Bei Projekten im safety-relevanten Bereich ist die Projekt-Kontext-Analyse besonders wichtig, da die äußeren Einflüsse auf das Projekt entsprechend berücksichtigt werden müssen. Dies erfordert ein strukturiertes Vorgehen in 7 Teilschritten:

1. Projekt-Inputs identifizieren [PjI_BA2.1]
2. Nach-Projektphase analysieren [Pj_BA2.2]
3. Andere Projekte im Unternehmen analysieren [PjI_BA2.3]
4. Safety-Management im Unternehmen und Safety-Kultur darstellen [PjI_BA2.4]
5. Qualitätsmanagementsystem darstellen [PjI_BA2.5]
6. Identifikation der wichtigsten Stakeholder [PjI_BA2.6]
7. Zuständige Behörden, Regulatoren und Zertifizierungseinrichtungen identifizieren [PjI_BA2.7]

Im ersten Teilschritt werden die Projekt-Inputs [PjI_BA2.1] identifiziert. Das sind beispielsweise Hazard- und/oder Unfall-Checklisten. Sind diese nicht vorhanden, so kann auf vorhandene Hazard-Checklisten, wie sie beispielsweise auch Ericson [Eric2_05. S.483ff] anführt, zugegriffen werden. Weitere Inputs können auch Re-Use Teile oder Standardarchitekturen sein, welche bereits einem Safety Assessments unterzogen wurden und die Anforderungen einer definierten Sicherheitsanforderungsstufe erfüllen. Es können aber auch einzukaufende System-Elemente sein, die sich bereits am Markt bewährt haben, in der Fachsprache als „Proven in Use" bezeichnet. Erfahrungen und/oder Erfahrungswerte und Metriken aus Vorgänger-Projekten sollen dabei ebenfalls berücksichtigt werden. Diese Projekt-Inputs sind wichtig, da sich mit bewährten Re-Use Teilen und Erfahrungswerten die Fehlermöglichkeiten einschränken lassen, die wiederum die Anzahl der nicht prognostizierbaren Systemzustände und damit die System Accidents vermindern oder sogar verhindern können.

Die sogenannte Nach-Projektphase [PjI_BA2.2] muss ebenso in diesem Schritt mitbedacht werden. Das können beispielsweise Follow-up Projekte sein, welche die Ergebnisse dieses Projekts weiterverwenden müssen. In die Betrachtung der Nach-Projektphase fallen auch die erforderlichen Maintenance-Aktivitäten im operativen Bereich sowie die Entsorgung des Systems, welche bei safety-relevanten Systemen besonders wichtig sind. Das Safety-Vorgehensmodell empfiehlt, auch andere Projekte im Unternehmen zu analysieren [PjI_BA2.3], welche gewollten oder ungewollten Einfluss auf das Projekt ausüben können. Diese können synergetischer Natur sein oder können Konflikte aufweisen, wie beispielsweise die Aufteilung von vorhandenen Ressourcen, was immer wieder zu Problemen führt. Darüber hinaus kann sich der Einfluss der Unternehmenskultur [PjI_BA2.4] beträchtlich auf das Projekt auswirken. Die ISO 26262 fordert beispielsweise im Kapitel „Safety Culture" [ISO26_11, Teil 2, Kapitel 5.4.2] und im Kapitel „Quality Management

[48] Jegliche Information die bezüglich der Betrachtungseinheit existieren, z.B. Produktidee, Projektskizze, relevante Patente, Resultate aus Vorversuchen, Dokumentation bezüglich Vorgänger Betrachtungseinheiten, relevante Informationen von anderen unabhängigen Betrachtungseinheiten. (Übersetzung des Verfassers)

during the safety lifecycle“ [ISO26_11, Teil 2, Kapitel 5.4.4.1] den Nachweis, dass Management-Systeme [PjI_BA2.5] im Unternehmen eingeführt sind, die sich nachweislich bewährt haben.

Auch eine erste Stakeholderanalyse [PjI_BA2.6] ist an dieser Stelle von Bedeutung. Dabei sind nach Möglichkeit alle wichtigen Stakeholder, die auf das Projekt Einfluss nehmen könnten, zu identifizieren, was aus Zeit- und Kostengründen in dieser Phase nicht immer so einfach möglich ist. Besonders wichtig ist die Identifikation der Behörden und Regulatoren derjenigen Länder, in denen das System ausgeliefert wird. Damit können die behördlichen Rahmenbedingungen sowie die einzuhaltenden Gesetze, Normen und Standards identifiziert und berücksichtigt werden.

Weitere wichtige Stakeholder sind die Zertifizierungseinrichtungen [PjI_BA2.7], die nach den geforderten Safety-Normen die Zertifizierung durchführen. Das sind zentrale Inputs für die Identifikation der Zertifizierungsaufwände, die von verschiedenen Zertifizierern unterschiedlich gesehen werden können. Die Einflussbereiche der verschiedenen Stakeholder auf das Projektvorhaben sind entsprechend zu berücksichtigen und im organisatorischen Konzept zu dokumentieren sowie entsprechende Management-Maßnahmen zu setzen.

Erstansatz der Projekt-Prozesslandschaft (PPL) erstellen [PjI_BA3]

Basierend auf den bisher ermittelten Informationen wird nun das strategische Vorgehen – in Form eines konkreten Modells für das Projekt definiert. Dieses projektspezifische Modell wird an die Projektsituation angepasst und als Projekt-Prozesslandschaft (PPL) dargestellt. Die PPL gibt einen Gesamtüberblick über das durchzuführende Projekt und stellt alle dafür benötigten Prozesse in diesem Modell dar. Diese Darstellung hilft, auf einem hohen Abstraktionslevel, im Sinne einer *ganzheitlichen Betrachtung*, einen Überblick über das gesamte Projektvorhaben zu bekommen. Damit werden Gesamtzusammenhänge erkennbar, Abläufe transparenter und Schwachstellen sichtbar. Des Weiteren können kritische Phasen identifiziert werden. Das empfohlene Vorgehen zum Ableiten der PPL wird anhand eines Beispiels im Folgenden gezeigt. Es sei bemerkt, dass für diese Ableitung selbstverständlich Erfahrung nötig ist, beziehungsweise dass diese Ableitung mehr erfordert als ein bloßes „Abarbeiten“ der beschriebenen Schritte. Die Basis-Aktivität [PjI_BA3] ist in 5 Teilschritte unterteilt:

1. Prozesse von den Hauptaufgaben des SoW ableiten und in einem V-Modell darstellen [PjI_BA3.1]
2. Engineering Lifecycle vom V-Modell ableiten [Pj_BA3.2]
3. System Safety Lifecycle darstellen und Safety-Prozesse den Engineering-Prozessen zuordnen [PjI_BA3.3]
4. Support-Prozesse in die PPL einbinden [PjI_BA3.4]
5. Projektmanagement-Lifecycle einbinden [PjI_BA3.5]

Der erste Teilschritt beschäftigt sich mit der Identifikation der benötigten Engineering-Prozesse. Die Basis dafür liefern die Hauptaufgaben aus dem Scope of Work. Dabei wird für jede Hauptaufgabe ein entsprechender Prozess aus dem ISaPro® ausgewählt und in einem V-Modell dargestellt [PjI_BA3.1]. Stehen für bestimmte Hauptaufgaben keine Prozesse zur Verfügung, so sind dafür fiktive Prozesse einzusetzen. Auch die organisatorischen Rahmenbedingungen können einen Einfluss auf die Darstellung der Engineering-Prozesse haben – soll beispielsweise eine im

Unternehmen vorgegebene Standardarchitektur verwendet werden, so wird der Prozess zur Erstellung der Systemarchitektur nicht benötigt. Außerdem werden je nach Komplexität und Art der System-Ziele die einzelnen Engineering-Prozesse, wie beispielsweise ein oder mehrere Komponenten-Designs, benötigt oder nicht. Die auf diese Weise ausgewählten Engineering-Prozesse werden nun in einem V-Modell dargestellt. Im V-Modell selbst wird auch ersichtlich, in welcher Ausprägung Verifikations- und Validationsmaßnahmen nötig sind. Abbildung 21 zeigt ein fiktives Beispiel, welches für eine System-Entwicklung mit Hardware- und Software-Anteil hergeleitet wurde.

Im diesem Beispiel werden als erstes die Kunden-Requirements dargestellt, die der Auftraggeber bzw. der Kunde zur Verfügung stellt. Von den Kunden-Requirements werden die System Requirements abgeleitet. Basierend auf den System Requirements wird das System Design entwickelt, dann spaltet sich das V-Modell in den Hardware- und Software-Pfad auf. Im Software-Pfad werden vom System Design die Software-Requirements abgeleitet und auf Basis dieses Vorgehens die Software-Architektur entwickelt. Kann aufgrund der Komplexität der Software-Architektur die Implementierung nicht direkt durchgeführt werden, dann muss beziehungsweise müssen, wie in diesem Beispiel dargestellt, ein oder mehrere Software-Komponenten Designs entworfen werden. Der letzte Prozess des V-Modells im Software-Pfad ist die Implementierung.

Der Hardware-Pfad ist ähnlich aufgebaut. Die Hardware-Requirements werden ebenfalls vom System Design abgeleitet und darauf basierend die Hardware-Architektur entwickelt. Die Hardware-Architektur ist in diesem Fall die Basis für die Layout-Entwicklung und die Realisierung der Hardware. Für die Layout-Entwicklung, Hardware Realisierung und Hardware-Test stehen keine generischen Prozesse zur Verfügung und können von Organisation zu Organisation auch unterschiedlich ausgeprägt sein. Daher werden hier fiktive Prozesse eingesetzt. Das sind sogenannte Platzhalter, damit die dafür erforderlichen Aktivitäten nicht verloren gehen.

Der rechte Teil des V-Modells zeigt die Test- und Integrations-Prozesse. Die Komponenten-Tests verifizieren die Komponenten-Designs. In diesem Beispiel sind die Komponenten-Tests mit Hardware geplant. Hierfür wird das erste Hardware-Ergebnis, meist als Lab0-Muster bezeichnet, dafür herangezogen. Danach werden die einzelnen Software-Komponenten schrittweise integriert. Dafür sollen nächste, verbesserte Hardware Lab1-Muster herangezogen werden. Nach erfolgreicher Integration aller Software-Komponenten wird der Software-Test unter Einbeziehung des Lab2-Musters durchgeführt. Darauf folgen die Integration von Hardware und Software nach den Anforderungen des Prozesses für die System-Integration und ein Integrationstest. Mit dem System-Test werden die System-Anforderungen verifiziert. Die Engineering-Aktivitäten werden mit dem Acceptance-Test, der mit dem Kunden und/oder den Behörden durchgeführt wird, abgeschlossen. Dieser letzte Prozess erfüllt die System-Validierung.

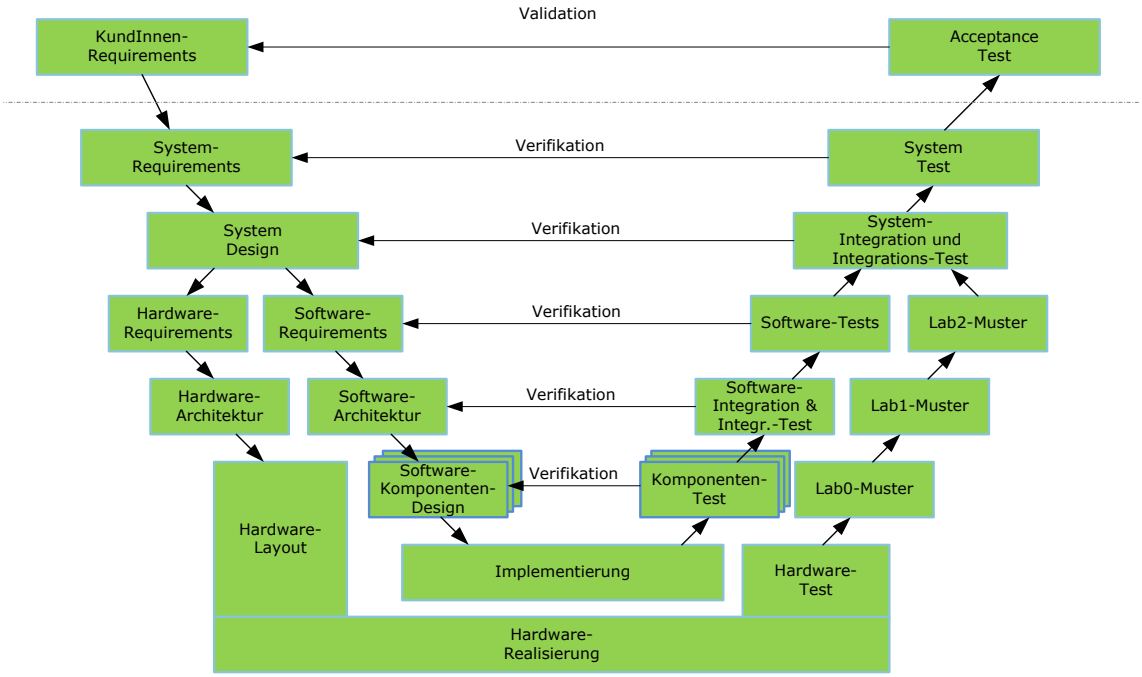


Abbildung 21: V-Modell, angepasst auf eine Projekt-Situation (Beispiel)

Das V-Modell stellt die erforderlichen Engineering-Prozesse dar, jedoch nicht die zusätzlich notwendigen Safety-, Support- und Management-Prozesse. Integriert man diese Prozesse in das V-Modell, würde dabei die Übersichtlichkeit verloren gehen. Daher wird nun das V-Modell „aufgeklappt“ und als sequentieller Engineering Lifecycle dargestellt [PjM0_BA3.2], siehe dazu Abbildung 22. Dies ist der initiale Schritt für die Darstellung in der Projekt-Prozesslandschaft.

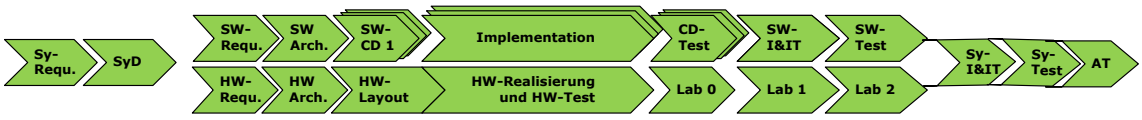


Abbildung 22: Engineering Lifecycle, abgeleitet vom V-Modell (Beispiel)

Im nächsten Schritt wird der System Safety Lifecycle an die Engineering-Prozesse angepasst [PjM0_BA3.3]. Die dafür erforderlichen Safety-Prozesse sind von den jeweiligen Engineering-Prozessen abhängig. Die nachfolgende Tabelle 4 zeigt die Zuordnung bzw. die gegenseitigen Abhängigkeiten.

Engineering-Prozesse	**Safety Prozesse**
Problemraum	
▪ Konzeptionierung	▪ Preliminary Hazard Identification (PHI)
Modellierungsraum	

▪ System Requirements	▪ Functional Hazard Evaluation (FHE)
▪ System Design	▪ Preliminary System Safety Evaluation (PSSE)
Lösungsraum	
▪ Software Requirements ▪ Hardware Requirements	▪ Software Safety Requirements Evaluation (SSRE) ▪ Hardware Safety Requirements Evaluation (HSRE)
▪ Software Architecture ▪ Hardware Architektur	▪ Software Safety Architecture Evaluation (SSAE) ▪ Hardware Safety Architecture Evaluation (HSAE)
▪ Software Komponenten Design	▪ Software Safety Component Evaluation (SSCE)
▪ Software Implementierung ▪ Software Integration ▪ Software Test	▪ Software Safety Code and Test Evaluation (SSCTE)
▪ Hardware Realisierung	▪ Hardware Safety Component and Test Evaluation (HSCTE)
▪ Software Component Test ▪ Software Integration und Integrations-Test ▪ Software Test ▪ System Integration ▪ System Test ▪ Acceptance Test	▪ System Safety Evaluation (SSE)

Tabelle 4: Zuordnungstabelle Engineering-Prozesse zu den Safety-Prozessen

Jedem vorhandenen Engineering-Prozess wird in der PPL der entsprechenden Safety Prozess zugeordnet. Der Grund dafür ist, dass alle ISaPro®-Prozesse aufeinander abgestimmt und somit die Wechselwirkungen zwischen den jeweiligen Prozessen bereits berücksichtigt sind.

An dieser Stelle sei nochmals erwähnt, dass der Problemraum, der an dieser Stelle beschrieben wird, nicht Teil der eigentlichen Projekt-Durchführung ist, sondern die Vor-Projektphase demonstriert. Daher wird der Problemraum auch nicht in der PPL dargestellt.

Die Zuordnung der Prozesse im Problemraum soll aber trotzdem hier beschrieben werden, da er für das Gesamtverständnis notwendig und auch ein wesentlicher Teil des Safety-Vorgehensmodells ist. Im *Problemraum* ist die Zuordnung stets gleich, der Konzeptionierung wird die Preliminary Hazard Identification (PHI) zugeordnet. Dies ist dadurch begründet, dass in der Projekt-Initialisierung die Sicherheitsanforderungsstufe zur Abschätzung der Aufwände und Kosten und somit für die Angebotserstellung benötigt wird. Die Sicherheitsanforderungsstufe muss jedoch in der PHI – basierend auf dem technischen Konzept – identifiziert werden.

Im *Modellierungsraum* wird dem Requirements Engineering auf System-Ebene der Functional Hazard Evaluation Prozess (FHE) zugeordnet. Im FHE-Prozess werden mögliche Abweichungen von den Requirements, die eine gefährliche Situation auslösen können, identifiziert. Dem Design Prozess wird die Preliminary System Safety Evaluation (PSSE) zugeordnet. Das Ziel der PSSE ist,

zu analysieren, ob das Design alle Safety Requirements umsetzt und ob das Design selbst keine zusätzlichen Hazards verursacht.

Etwas komplizierter ist die Zuordnung im *Lösungsraum*. Dieser ist von den Rahmenbedingungen des Projekts und der notwendigen Prozesskonstellation der Engineering-Prozesse abhängig. Handelt es sich um ein Entwicklungsprojekt, in welchem Software und Hardware zu realisieren sind, wie auch in Abbildung 22 angeführt, so müssen Software- und Hardware-Requirements erstellt werden. Dem Software Requirements Prozess wird der Software Safety Requirements Evaluation (SSRE) Prozess und dem Hardware Requirements Prozess der Hardware Safety Requirements Evaluation (HSRE) Prozess zugeordnet.

Die SSRE überprüft, ob die Sicherheitsanforderungsstufe richtig durchgereicht wurde und ob die Safety Requirements aus dem Modellierungsraum an die Software Requirements entsprechend durchpropagiert wurden. Die funktionalen Software Requirements werden zudem auf mögliche Hazards analysiert, zusätzlich überprüft man ob die Safety-Integritäts-Requirements der Safety-Normen eingehalten wurden. Das sind beispielsweise die Entdeckung und Beherrschung von Fehlern im System und mit interagierenden Systemen.

Ähnlich verhält es sich beim HSRE Prozess, auch hier wird überprüft, ob die Sicherheitsanforderungsstufe exakt übernommen wurde und die Hardware Safety Requirements aus dem Modellierungsraums richtig durchpropagiert und als Requirements für die Hardware-Sicherheitsmechanismen richtig spezifiziert wurden. Auch auf Basis der Hardware-Funktionen wird nochmals eine Safety-Analyse durchgeführt. Darüber hinaus ist es notwendig, die von der Safety-Norm geforderten Requirements, wie beispielsweise Hardware-Prinzipien und Hardware-Metriken, richtig zu spezifizieren, die wiederum stark von der Sicherheitsanforderungsstufe abhängig sind.

Der Software Architektur wird dem Software Safety Architecture Evaluation (SSAE) Prozess zugewiesen, welcher im Prinzip dieselbe Aufgabe erfüllt, wie die PSSE auf System Design Ebene. Der Hardware Architektur wird dem Hardware Safety Architecture Evaluation (HSAE) Prozess zugeordnet. Auch hier sind die geforderten Aktivitäten ähnlich der PSSE.

Dem Software Component-Designs wird die Software Safety Component Evaluation (SSCE) zugeordnet. Im Rahmen der SSCE wird das Component Design dahingehend analysiert, dass die bislang definierten Safety Requirements erfüllt werden und das jeweilige Component Design keine zusätzlichen Hazards enthält. Der Software Safety Code and Test Evaluation (SSCTE) Prozess ist der Implementierung sowie den Test-Aktivitäten auf Software-Ebene zugeordnet. Der SSCTE Prozess evaluiert die Umsetzung auf Code-Ebene und verifiziert die Software-Safety-Anforderungen. Dem Hardware Component Design und der Hardware-Realisierung wird der Hardware Safety Test Evaluation (HSTE) Prozess zugeordnet, der die Realisierung der Hardware Safety Requirements überprüft und die Hardware auf mögliche Hazards analysiert.

Den Test-Prozessen auf Component-Ebene sowie dem System Integrations-Prozess und den finalen Test-Prozessen auf System-Ebene wird die System Safety Evaluation (SSE) zugewiesen. Die SSE überprüft somit letztlich, ob die tatsächlich implementierten System-Elemente bzw. das System die Safety Requirements erfüllt. Die nachfolgende Abbildung zeigt nun die aktualisierte PPL mit den Engineering- und den zugeordneten Safety-Prozessen.

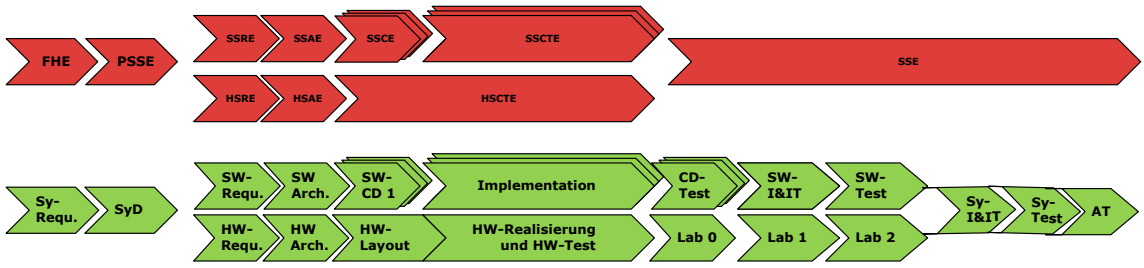


Abbildung 23: Projekt-Prozesslandschaft mit Engineering- und Safety-Prozessen (Beispiel)

Im nächsten Schritt werden die erforderlichen Support-Prozesse (Quality Assurance, Konfigurationsmanagement, Verifikation und Validierung) hinzugefügt [PjM0_BA3.4]. Die Support-Prozesse laufen über den gesamten Entwicklungs-Lifecycle.

Im letzten Teilschritt wird der Projektmanagement-Lifecycle hinzugefügt [PjM0_BA3.5]. Die vollständige PPL ist in Abbildung 24 dargestellt. Dabei ist darauf zu achten, die Planungsphase an die Rahmenbedingungen des Projekts anzupassen. Das heißt, je ungenauer die Anforderungen an das Projekt sind, desto mehr Zeit und Aufwand sollte für die Planungsphase veranschlagt werden.

Konzeptionierung und PHI beauftragen/veranlassen [PjI_BA4]

Das SoW und die PPL sowie die Ergebnisse aus der PKA liefern die Basis für die Beauftragung zur Konzeptionierung und zur Durchführung der darauf basierenden Safety-Analysen, wie sie der PHI-Prozess fordert. In der Konzeptionierung wird das System abgegrenzt, die System-Umgebung definiert sowie die Top-Level Requirements identifiziert und ein erster System-Entwurf entwickelt. Davon ausgehend werden in der PHI die ersten Safety-Analysen durchgeführt, die Sicherheitsanforderungsstufe festgelegt und die Safety Requirements abgeleitet.

Organisatorisches Konzept erstellen [PjI_BA5]

Basierend auf den Safety Requirements aus der PHI kann nun das organisatorische Konzept vervollständigt werden und ist somit ein wichtiger Input für die Projekt-Initialisierung. Die Erstellung des organisatorischen Konzepts ist in 6 Basis-Aktivitäten unterteilt:

1. Scope of Work (SoW) verfeinern [PjI_BA5.1]
2. Projekt Prozesslandschaft (PPL) verfeinern/nacharbeiten [Pj_BA5.2]
3. Machbarkeit überprüfen (technisch und organisatorisch) [PjI_BA5.3]
4. Projekt-Risiko einschätzen [PjI_BA5.4]
5. Leistungen abschätzen [PjI_BA5.5]
6. Projekt-Kosten errechnen [PjI_BA5.6]

Die in der PHI identifizierte Sicherheitsanforderungsstufe wird in die Ziele-Definition des SoW [PjM_BA5.1] aufgenommen. Wenn dadurch zusätzliche Hauptaufgaben erforderlich sind, müssen diese ergänzt bzw. vorhandene verfeinert werden. Auch die PPL ist dann entsprechend zu überarbeiten bzw. zu verfeinern [PjM_BA5.2]. In die PPL werden danach noch die Stage Gates (SG2 und SG3), die sogenannten strategischen Projekt-Eckpunkte, die das Projekt in einzelne Stages unterteilen, integriert (siehe Kapitel 2.3 „Integrativer Safety Prozess (ISaPro®)“). Danach werden die

erforderlichen Assessments und Audits festgelegt. In diesem Beispiel sind drei Safety Assessments eingeplant. Das erste vor der Implementierung bzw. der Realisierung, das zweite nach den Komponenten-Tests und das dritte nach der System-Integration. Dabei soll auch überprüft werden, ob das System inhärent sicher ist. Die Safety Assessments werden je nach Sicherheitsanforderungsstufe von unabhängigen Personen oder Organisationen durchgeführt.

Wenn die Entwicklung nach den Anforderungen der ISO/IEC 15504 nach ASPICE Level 2 erfolgen soll, was in der Automobil-Branche der Fall ist, ist der Quality Assurance Prozess unabdingbar. Höhrmann [Höh_09, S.65] begründet folgendermaßen: „Mit der Etablierung dieses Prozesses in einem Projekt wird bereits eine wichtige Voraussetzung geschaffen, die Anforderungen von Automotive SPICE mindestens bis zu einem Level 2 zu erfüllen."

Im Zuge der Erfüllung des SPICE Level 2 werden auch mehrere Quality Audits gefordert. Eine unabhängige Instanz muss also zu definierten Zeitpunkten überwachen, ob die Prozessdurchführung geplant ist, die Verantwortlichkeiten definiert und die Arbeitsprodukte (Zwischenprodukte, Dokumente, Spezifikationen, etc.) qualitätsgesichert sind. Des Weiteren fordert die ISO 15504 [ISO15_08] bei der Erfüllung von ASPICE Level 2, dass alle Arbeitsprodukte unter Konfigurationsmanagement stehen müssen. Aus diesem Grund sind zwei Konfigurations-Audits (CM-Audit) geplant sowie fünf Baselines. Die Baselines werden hier auch als Nachweise (Evidenzen) für den Safety Case herangezogen. Die nachfolgende Abbildung 24 zeigt die PPL für das dargestellte theoretische Beispiel.

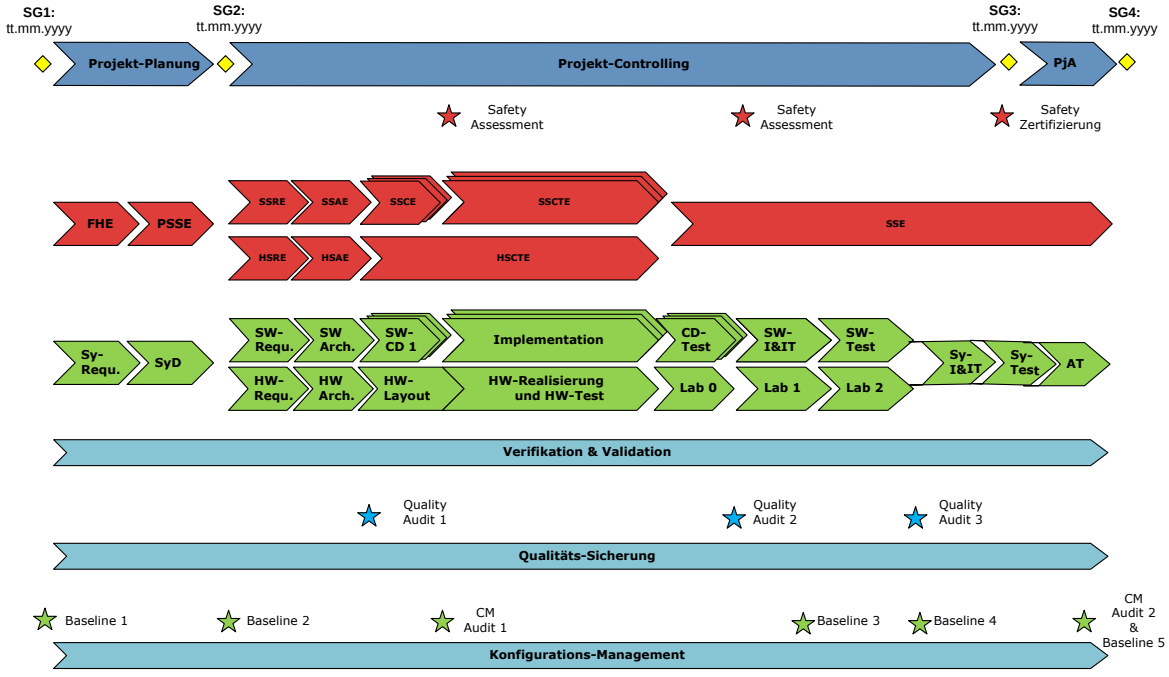


Abbildung 24: Projekt-Prozesslandschaft komplett (Beispiel)

Die umfassende Projekt-Prozesslandschaft gibt den strategischen Überblick über das gesamte Entwicklungsprojekt und liefert damit die Basis für das organisatorische Konzept.

Basierend auf den drei Konzepten kann die technische und organisatorische Machbarkeit [PjI_BA5.3] überprüft werden. Die technische Machbarkeit wird auf Basis des technischen Konzepts und der dafür erforderlichen Technologien überprüft. Die organisatorische Machbarkeit anhand der erforderlichen Ressourcen und der dafür geforderten Kompetenzen. Dies ist auch ein wichtiger Input für die Einschätzung der Projekt-Risiken [PjI_BA5.4].

Im nächsten Teilschritt können nun, auf Basis der PPL und der darin zugeordneten Prozesse, die Leistungen über den gesamten Projektverlauf abgeschätzt [PjI_BA5.5] werden. Die Basis-Aktivitäten der vorgegebenen Prozesse liefern die Grundlage für die zu erbringenden Leistungen. Mit dem Vorgehen, dabei auch auf definierte Prozesse zurückzugreifen, wird auch die Basis für ein strukturiertes Abschätzungsverfahren gelegt. Damit können Abschätzungsdaten auch in weiteren Projekten wiederverwendet werden, die für schnellere und genauere Abschätzungen sorgen.

Anhand der Ergebnisse der Abschätzungen erfolgt nun die Planung der Eckdaten in der PPL. Dabei werden der Projektstarttermin und der Projektendtermin sowie die Durchlaufzeit der einzelnen Projektphasen festgelegt. Darauf stützend können die Plandaten für die Stage Gates, Audits, Assessments und Baselines fixiert werden. Damit sind die wichtigsten Voraussetzungen zur Überprüfung der Durchführbarkeit und für die Berechnung der Projektkosten hergestellt [PjI_BA5.6].

Makro-Konzept erstellen und freigeben [PjI_BA6]

Die letzte Basis-Aktivität fordert die Erstellung des Makro-Konzepts sowie die Durchführung eines Reviews, mit allen Verantwortlichen Personen und Experten. Nach einem positiv abgeschlossenen Review wird das Makro-Konzept freigegeben. Das freigegebene Makro-Konzept schließt den Problemraum ab und liefert somit die Basis für die Beauftragung des Projekts.

3.1.2 Konzeptionierung

Der Anspruch dieser Arbeit ist, dem Prinzip des „Inherent System Safety Approach“ gerecht zu werden, welches auch betont, dass das zu entwickelnde System im *Gesamtkontext* zu betrachten ist und ein „*Inherent Safe System Design*“ fordert. Dafür müssen bereits in der Vor-Projektphase alle äußeren Einflüsse auf das System systematisch-methodisch identifiziert und analysiert werden. Daher ist der erste Schritt der Konzeptionierung, die Festlegung der Systemgrenzen und die Identifikation der System-Umgebung und des System-Umfeldes, um davon ausgehend einen inhärent sicheren System-Entwurf so genau wie nur möglich definieren zu können. Dieser System-Entwurf ist Hauptbestandteil des technischen Konzepts und basiert auf den betrieblichen Anforderungen sowie den Rahmenbedingungen der System-Umgebung und des System-Umfeldes.

Der Prozess der Konzeptionierung ist in 4 grundlegende Basis-Aktivitäten untergliedert. Im ersten Schritt werden die System-Umgebung und das System-Umfeld identifiziert [KON_BA1]. Das heißt, es muss das zu entwickelnde System definiert sowie seine System-Umgebung und sein System-Umfeld identifiziert werden. Darauf basierend werden die Top-Level Requirements abgeleitet und spezifiziert [KON_BA2]. Anschließend kann damit der erste System-Entwurf entwickelt werden [KON_BA3]. Die Spezifikation des technischen Konzepts [KON_BA4] liefert den Output des

Konzeptionierungs-Prozesses. Die nachfolgende Tabelle 5 fasst die erforderlichen Basis-Aktivitäten der Konzeptionierung nochmals zusammen.

KON_BA1	System-Umgebung und System-Umfeld identifizieren
KON_BA2	Top-Level Requirements spezifizieren
KON_BA3	System-Entwurf entwickeln
KON_BA4	Technisches Konzept spezifizieren und freigeben

Tabelle 5: Basis-Aktivitäten der Konzeptionierung

System-Umgebung und System-Umfeld identifizieren [KON_BA1]

Die Identifikation des System-Umfeldes umfasst 6 Teilschritte:

1. Systemgrenzen festlegen [KON_BA1.1]
2. Unmittelbare System-Umgebung identifizieren [KON_BA1.2]
3. Mittelbare System-Umgebung identifizieren [KON_BA1.3]
4. System-Einbettung identifizieren [KON_BA1.4]
5. System-Umwelten identifizieren [KON_BA1.5]
6. Einsatzbereich identifizieren [KON_BA1.6]

Im ersten Teilschritt werden die Systemgrenzen festgelegt [KON_BA1.1], indem das System als leeres Rechteck dargestellt wird. Dabei symbolisieren die Seiten des Rechtecks die Systemgrenzen. Alles, was außerhalb dieses Rechtecks liegt, zählt zur System-Umgebung. Die Systemabgrenzung ist der erste und wichtigste Schritt, da sich darauf aufbauend die unmittelbare System-Umgebung bestimmen lässt. Es mag sein, dass dies noch nicht die endgültige Abgrenzung ist, und man in der Folge erkennt, dass das eine oder andere Element in das System aufzunehmen bzw. aus dem System zu entfernen ist. Eine klare Festlegung, die zumindest bis auf weiteres Gültigkeit hat, ist jedoch insbesondere für die nachfolgende Hazard-Analyse wesentlich.

Nach der System-Abgrenzung wird die System-Umgebung definiert, welche in die unmittelbare und in die mittelbare System-Umgebung unterteilt wird. In der unmittelbaren System-Umgebung [KON_BA1.2] befinden sich alle direkt angrenzenden anderen Systeme. Von diesen anderen Systemen können die Top-Level Funktionen abgeleitet werden. Danach wird die mittelbare System-Umgebung [KON_BA1.3] identifiziert, das sind jene anderen Systeme, die nicht direkt an das zu entwickelnde System angrenzen. Damit wird auch ersichtlich, wie das System in seiner Ziel-Umgebung eingebettet ist (hierarchisches Konzept) [KON_BA1.4].

Ein Beispiel soll die Vorgehensweise genauer darstellen: Das System sei ein „ABS-Steuergerät“ für ein Auto. Zweck des ABS-Steuergerätes ist es, beim Bremsen blockierende Räder zu verhindern, um die Lenkfähigkeit des Fahrzeugs aufrecht zu erhalten. Die direkte System-Umgebung besteht aus den Raddrehzahlsensoren, dem Bremspedal, dem Hydroaggregat und einem CAN-Bus. All diese Systeme aus der unmittelbaren System-Umgebung sind mit dem ABS-Steuergerät physikalisch verbunden.

Das ABS-Steuergerät ist wiederum im übergeordneten System „Fahrwerk“ integriert, das Fahrwerk ist Teil des Gesamtsystems „Auto“. Das heißt, dass alle anderen Systeme, die sich im Fahrwerk bzw. im Gesamtsystem Auto befinden und nicht mit dem ABS-Steuergerät direkt verbunden sind, zu den mittelbar angrenzenden Systemen zählen. Das sind beispielsweise das EBV-Steuergerät[49] oder das ESP-Steuergerät[50] im Fahrwerk oder andere Systeme auf Fahrzeug-Ebene. Auch diese anderen nicht direkt mit dem ABS-Steuergerät verbundenen Systeme können Einfluss auf das ABS-Steuergerät nehmen und sind in die Analyse-Aktivitäten mit einzubeziehen.

In den nächsten beiden Schritten werden noch die Systemumwelten identifiziert [KON_BA1.5], und der genaue Einsatzbereich [KON_BA1.6] definiert. Zu den System-Umwelten zählen beispielsweise klimatische Verhältnisse, chemische Einwirkungen oder kosmische Strahlungen. Der Einsatzbereich gibt zum Beispiel die Einsatzdauer und -situationen an, aber auch gesetzliche Vorgaben und die politische Lage der Einsatzregionen.

Top-Level Requirements spezifizieren [KON_BA2]

Im zweiten Prozessschritt des Konzeptionierungs-Prozesses werden die Top-Level Requirements spezifiziert. Die Top-Level Requirements sind in die Top-Level Funktionen, in die Funktionalen Top-Level Requirements sowie in die nicht-funktionalen Top-Level Requirements und in die Top-Level Interface Requirements unterteilt. Diese Basis-Aktivität zwei ist in 4 Teilschritte unterteilt:

1. Top-Level Funktionen definieren [KON_BA2.1]
2. Funktionale Top-Level Requirements definieren [KON_BA2.2]
3. Nicht-funktionale Top-Level Requirements definieren [KON_BA2.3]
4. Top-Level Interface Requirements definieren [KON_BA2.4]

Zu den Top-Level Requirements zählen in erster Linie die Top-Level Funktionen und die Funktionalen Top-Level Requirements, also das funktionale Konzept nach Ropohl (siehe dazu Kapitel 2.1.1). Für die Identifikation der Top-Level Funktionen [KON_BA2.1] werden die unmittelbar angrenzenden Systeme aus der System-Umgebung herangezogen. Es wird analysiert, welches dieser unmittelbar angrenzenden Systeme, das zu entwickelnde System zu einer Handlung anregt, das heißt, „wer triggert das zu entwickelnde System?“. Die System-Trigger stellen in diesem Fall die Top-Level Funktionen dar. Das heißt, das zu entwickelte System muss nun darauf basierend ein entsprechendes Service an die System-Umgebung liefern, welches zu spezifizieren ist. Anhand des Triggers und des geforderten Services können nun auch die funktionalen System Requirements [KON_BA2.2] abgeleitet werden. Die nachfolgende Tabelle 6 zeigt ein Beispiel:

BNS ID	**System-Trigger (Top-Level Funktionen)**	**Service (Funktionale Top-Level Requirements)**
Brems-pedal	Bremspedal wird gedrückt	Wenn das Bremspedal gedrückt wird, muss das „ABS-Steuergerät“ über das Hydroaggregat den Bremsvor-

[49] EBV … Elektronische Bremskraft Verteilung

[50] ESP … Elektronisches Stabilitätsprogramm

		gang solange zulassen, bis die Raddrehzahlsensoren einen Schlupf erkennen oder das Bremspedal losgelassen wird.

Tabelle 6: Herleiten der Top-Level Funktionen und der funktionalen Top-Level Requirements

Ist das zu entwickelnde System abgegrenzt und seine System-Umgebung identifiziert, können davon ausgehend die nicht-funktionalen Requirements abgeleitet [KON_BA2.3] werden. Im nächsten Schritt sind die Top-Level Interface Requirements, also die externen Schnittstellen zur unmittelbaren System-Umgebung, zu definieren [KON_BA2.4]. Die Schnittstellen regeln die Beziehungen zwischen dem zu entwickelnden System und den unmittelbar benachbarten Systemen. Grundsätzlich werden an dieser Stelle Schnittstellen als tatsächliche Verbindungen der interagierenden Systeme verstanden, so Fischer [Fi_06, S.15]. Sie beeinflussen das Verhalten der verbundenen Systeme – System mit Systemen aus der System-Umgebung – als auch das Verhalten des gesamten Systems, so Biethahn und Mucksch [BiMu_04]. Über die definierten Schnittstellen wird der Übergang von einem zum anderen System gesichert, argumentiert Fischer [Fi_06, S.15].

Die Anforderungen der externen Schnittstellen müssen eindeutig spezifiziert werden und sind in dieser Phase erstmals unabhängig von der Realisierung des Systems zu betrachten. Das interne Verhalten des zu entwickelnden Systems liegt jedoch für das/die Schnittstellen nutzende(n) externe(n) Systeme in diesem Schritt nicht im Fokus und soll deshalb noch nicht in Betracht gezogen werden.

Derartige Schnittstellen zu anderen Systemen in der unmittelbaren System-Umgebung können analoge Eingangs-Schnittstellen zur Messung von physikalischen Größen, wie beispielsweise einer Batteriespannung sein oder Ausgangs-Schnittstellen zur Steuerung eines Motors. Ebenso können das digitale Schnittstellen sein, über die beispielsweise digitale Sensor-Signale bewertet bzw. digitale Stellglieder (z.B. Abgasrückführventil) gesteuert werden oder Bus-Schnittstellen, wie beispielsweise Profi-, CAN- oder Flexray-Bus.

System-Entwurf entwickeln [KON_BA3]

Mit der System-Abgrenzung und der Identifikation der System-Umgebung und des System-Umfelds auf dem richtigen Abstraktionslevel und der klaren Definition der Schnittstellen nach außen werden die wichtigsten Rahmenbedingungen für die Definition des zu entwickelnden Systems geschaffen. Das zu entwickelnde System besteht nun aus einer Menge von System-Elementen, welche in dieser Arbeit als Sub-Systeme bezeichnet werden. Solche Sub-Systeme sind beispielsweise Hardware- und/oder Software-Elemente, die im nächsten Detaillierungsschritt als Komponenten bezeichnet werden. Diese Untergliederung ist notwendig, damit auf Sub-System-Ebene die Hardware und die Software in ihrer Architektur nochmals unterteilt werden können. Damit besteht die Möglichkeit, auch auf dieser Ebene eine Kapselung von sicherheitskritischen Komponenten vorzunehmen zu können.

Ausgehend von den definierten externen Schnittstellen kann nun die interne System-Struktur festgelegt [KON_BA3.1] werden. Das systematische Vorgehen ist nun in drei Teilschritte aufgeteilt:

1. System-Inhalte definieren [KON_BA3.1]
2. Interne Schnittstellen definieren [KON_BA3.2]
3. Funktionale Top-Level Requirements verifizieren [KON_BA3.3]

Im ersten Teilschritt werden die System-Inhalte definiert [KON_BA3.1], indem von jeder externen Schnittstelle ausgehend die für die Schnittstelle erforderlichen Sub-Systeme definiert werden, siehe dazu Abbildung 25. Das sind im Falle des ABS-Beispiels der Ein-/Ausgabe-Treiber (I/O Driver) für Raddrehzahl-Sensoren und der Bremspedal-Sensor, der Hydroaggregat-Treiber (Aggregate Driver), und der CAN-Bus Treiber (CAN BUS Driver). Im nächsten Schritt wird die Steuerung (Controller) definiert. Sind alle Sub-Systeme definiert, können die internen Schnittstellen festgelegt werden [KON_BA3.2]. Die nachfolgende Abbildung 25 zeigt einen konzeptionellen Vorschlag, also den System-Entwurf für das elektrische ABS-Steuergerät. Dies ist lediglich ein fiktives Beispiel und erhebt keinen Anspruch auf Richtigkeit und Vollständigkeit. Die Steuerung selbst kann in einer nächsten Stufe wiederum in Hardware- und Software- Architektur unterteilt werden, die dann Architektur-Komponenten besitzen.

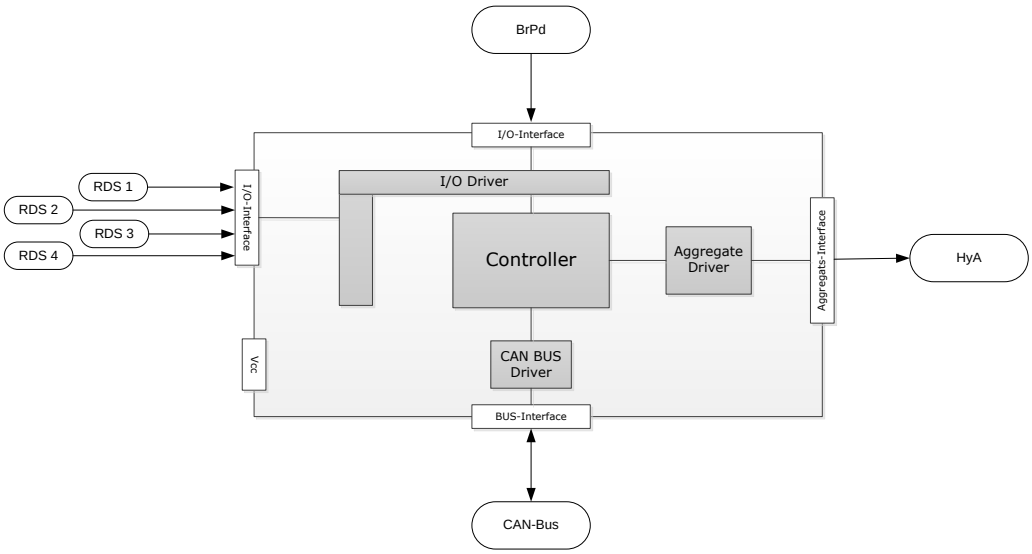


Abbildung 25: Entwicklung der Systemstruktur über externe Schnittstellen (Beispiel)

Anhand dieser Darstellung können die funktionalen System Requirements, die in Tabelle 6 dargestellt sind, verifiziert [KON_BA3.3] werden. Ist die PHI, die im nächsten Kapitel beschrieben wird, durchgeführt und sind die daraus spezifizierten Safety Requirements im System-Entwurf realisiert, müssen auch diese verifiziert werden.

Technisches Konzept spezifizieren und freigeben [KON_BA4]

Die letzte Basis-Aktivität der Konzeptionierung fordert die genaue Spezifikation des technischen Konzepts, da dieses für die Preliminary Hazard-Identifikation herangezogen wird. Das fertige technische Konzept soll in einem multidisziplinären Team (z.B. Safety-Engineer, Hard- und Software-Engineer, System-Architekten) einem Review unterzogen werden. Nach positivem Review-Ergebnis wird das technische Konzept freigegeben.

3.1.3 Preliminary Hazard Identification (PHI)

Die Preliminary Hazard Identification (PHI) beinhaltet die Safety-Aktivitäten im Problemraum. Ziel der PHI ist es, die organisatorischen Safety-Maßnahmen zu definieren und zu planen sowie die zentralen Hazards, im Problemraum als Top-Level Hazards[51] bezeichnet, zu identifizieren, zu bewerten und zu klassifizieren. Um brauchbare Analyse-Ergebnisse und Lösungsansätze erzielen zu können, ist ausführliches Informationsmaterial aus der System-Umgebung und dem System-Umfeld sowie ein modularer System-Entwurf notwendig. Wie auch Leveson [Lev_95] betont, ist die Qualität der Analyse-Ergebnisse stark von der Qualität des Modells, welches das reale System abbildet, abhängig.

Es sei erwähnt, dass im nachfolgenden Modellierungsraum die Hazard-Analyse in einem höheren Detaillierungsgrad ausgeführt wird, da für eine vollständige Hazard-Analyse erst einmal die Requirements aller Stakeholder eingeholt werden müssen, sowie das detaillierte System Design verfügbar sein muss, was zum Zeitpunkt der PHI noch nicht der Fall ist.

Die PHI ist in sechs grundlegende Basis-Aktivitäten untergliedert. Das ist im ersten Prozessschritt die Identifikation der organisatorischen Safety-Maßnahmen [PHI_BA1], um zu verstehen was von Regulatoren und Zertifizierungsbehörden sowie anderer Stakeholder gefordert ist. Danach werden auf System-Ebene die Top-Level Hazards identifiziert [PHI_BA2]. Die Herausforderung im Prozessschritt drei ist, diese Hazards soweit als möglich, im Sinne der Hazard Avoidance [PHI_BA3], zu vermeiden. Ist das nicht möglich, wird der Prozessschritt 4 eingeleitet, das Safety-Risiko-Assessment [PHI_BA4], welches die nicht vermeidbaren Hazards bewertet und einer entsprechenden Sicherheitsanforderungsstufe zuordnet. Das Safety-Konzept [PHI_BA5] dokumentiert die Risiko-Minimierungsmaßnahmen. Der PHI-Report [PHI_BA6] schließt den Prozess der Preliminary Hazard Identification ab. Die nachfolgende Tabelle 7 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im Prozess der PHI gefordert sind.

PHI_BA1	Organisatorische Safety-Maßnahmen identifizieren
PHI_BA2	Top-Level Hazards identifizieren
PHI_BA3	Hazard Avoidance durchführen
PHI_BA4	Safety-Risiko-Assessment durchführen
PHI_BA5	Safety-Konzept erstellen
PHI_BA6	PHI-Report verfassen und kommunizieren

Tabelle 7: Basis-Aktivitäten der PHI

[51] Hazards mit gleichen Ergebnissen werden zu einer generischen Hazard-Kategorie zusammengefasst und als Top-Level Hazards bezeichnet. Das erleichtert die Übersichtlichkeit auf Top-Level Ebene [Eric1_11, S. 437].

Organisatorische Safety-Maßnahmen identifizieren [PHI_BA1]

Die erste Basis-Aktivität der PHI ist in 5 Teilschritte aufgeteilt:

1. Anzuwendende Safety-Normen verstehen [PHI_BA1.1]
2. Erste Kontakte mit Regulatoren und/oder Zertifizierungsbehörden aufnehmen [PHI_BA1.2]
3. Zertifizierungs-Prozess festlegen [PHI_BA1.3]
4. Zertifizierungsaufwände abschätzen [PHI_BA1.4]
5. Zertifizierung planen [PHI_BA1.5]

Im ersten Teilschritt wird die anzuwendende Safety-Norm einer genauen Betrachtung unterzogen [PHI_BA1.1], da Safety-Normen meist unterschiedliche Sicherheits-Integritäts-Anforderungen und Risiko-Bewertungsmodelle fordern. Im nächsten Teilschritt ist eine erste Kontaktaufnahme mit den zuständigen Regulatoren oder den Zertifizierungsbehörden empfohlen [PHI_BA1.2]. Damit kann der Zertifizierungs-Prozess sowie die spezifischen und individuellen Forderungen der Zertifizierungsbehörden und deren Zertifizierer konkretisiert werden [PHI_BA1.3]. Darauf basierend können die Aufwände für die Zertifizierung genau abgeschätzt [PHI_BA1.4] und das Zertifizierungsdatum sowie die Vor- und Nachbereitungsarbeiten in die Grob-Planung aufgenommen [PHI_BA1.5] werden. Des Weiteren sind die daraus resultierenden Safety Audits in der Grob-Planung zu berücksichtigen.

Top-Level Hazards identifizieren [PHI_BA2]

Im zweiten Prozessschritt werden die Top-Level Hazards auf Basis des technischen Konzepts identifiziert, analysiert und eindeutig beschrieben. Grundvoraussetzung ist dabei, dass das Analyse-Team die System-Umgebung und das System-Umfeld genau kennt. Die Basis-Aktivität zwei der PHI ist in 5 Teilschritte untergliedert:

1. Top-Level Funktionen auf mögliche Hazards untersuchen [PHI_BA2.1]
2. Unmittelbare System-Umgebung auf mögliche Hazards untersuchen [PHI_BA2.2]
3. Mittelbare System-Umgebung auf mögliche Hazards untersuchen [PHI_BA2.3]
4. Gefahrenquellen und gefährliche Situationen identifizieren [PHI_BA2.4]
5. Unfallsauslösende Ereignisse betrachten [PHI_BA2.5]

Wesentlich bei der Identifikation der Top-Level Hazards ist die Wahl der Analyse-Methode. Dies begründet sich wie folgt: Ericson klassifiziert alle in [Eric2_05, S. 44] beschriebenen Analyse-Methoden und ordnet diese den System-Lifecycle Phasen zu. Das sind die Phasen "Concept Definition", "Preliminary Design", "Detailed Design", "Test", "Production", "Operation and Disposal". Der Problemraum im ISaPro® ist vergleichbar mit der Phase der „Concept Definition". Es handelt sich hierbei um die Phase vor der eigentlichen Entwicklung, d.h. in der Vor-Projektphase. Die einzige Methode, die Ericson der „Concept Definition" zuordnet, ist die PHL-Methode (siehe dazu auch Kapitel 2.5.1 „Preliminary Hazard List"). Die PHL hat zum Ziel, möglichst frühzeitig viele potentielle Hazards zu identifizieren.

Laut Ericson [Eric2_05, S.56] ist die PHL-Methode „similar to a brainstorming session", geht aber über ein reines Brainstorming weit hinaus, da über bereits bestehende Checklisten, Hazards

gesammelt und darauf basierend die Auswirkungen (Effekte) eingeschätzt werden. Dennoch ist die PHL für den „Inherent System Safety Approach“ nicht optimal geeignet, da ein Brainstorming zuwenig Systematik hat. Dadurch können bereits im Vorfeld Lücken entstehen oder Fehleinschätzungen passieren. Des Weiteren ist es im Rahmen der PHL nicht vorgesehen, eine vorläufige Einschätzung der Sicherheitsanforderungsstufe zu ermitteln, andererseits wird schon auf Hardware-Elemente und Software-Funktionen verwiesen, die zu diesem Zeitpunkt noch gar nicht konkretisiert sein müssen. Darüber hinaus existieren derartige Hazard- und Unfall-Checklisten in vielen Unternehmen noch nicht. Speziell im Automotive-Bereich sind diese nicht vorhanden, da die ISO 26262 erst seit Ende 2011 verpflichtend eingeführt wurde.

Im Rahmen des Safety-Vorgehensmodells ist es nötig, eine vorläufige, aber gut argumentierte Sicherheitsanforderungsstufe identifizieren zu können. Diese wird benötigt, um die daraus geforderterten Aktivitäten in den einzelnen Phasen des Projektes ausreichend genau abschätzen und planen zu können. Gleichzeitig sind im Problemraum viele Informationen, auf die die PHL aufsetzt, noch nicht verfügbar. So geht die Preliminary Hazard List einerseits zu stark ins Detail, andererseits liefert sie nicht die gewünschten Ergebnisse, die aus Sicht des Problemraumes erforderlich sind. Aus diesem Grund konzentriert sich der Prozess der Preliminary Hazard Identification (PHI) auf die identifizierten Funktionen und die externen Einflussfaktoren (exogene Hazards), die einen Hazard bzw. einen Unfall auslösen können.

Das von der PHI vorgeschlagene Vorgehen ist nun folgendes: Im ersten Teilschritt werden die in der Konzeptionierung identifizierten Top-Level Funktionen (TLF) (siehe Kapitel 3.1.2) auf mögliche Hazards analysiert [PHI_BA2.1]. Dabei werden für jede TLF mögliche Abweichungen systematisch untersucht und bewertet. Anhand des Hazard Triangles wird überprüft, ob es sich tatsächlich um einen Hazard handelt oder um eine Failure.

Im nächsten Teilschritt werden alle externen Einflussfaktoren und -bereiche auf das zu entwickelnde System, in Bezug auf exogene Hazards, untersucht. Dabei werden die Interaktionen mit anderen Systemen aus der unmittelbaren System-Umgebung betrachtet und auf mögliche Hazards analysiert [PHI_BA2.2]. Der dritte Teilschritt beschäftigt sich mit möglichen Gefährdungen aus der mittelbaren System-Umgebung [PHI_BA2.3], das sind beispielsweise Vibrationen, Intensität oder giftige Substanzen. Im Teilschritt 4 werden gefährliche Situationen (Operational Modes) analysiert [PHI_BA2.4]. Teilschritt 5 untersucht unfallauslösende Ereignisse [PHI_BA2.5], die durch menschliches Versagen, prozedurale Fehler oder Fehlmechanismen verursacht werden können. Unfallauslösende Ereignisse können aber auch Ausfälle anderer Systeme sein oder Störeinflüsse aus der System-Umgebung. Es können aber auch gefährliche Situationen im System-Umfeld sein, wie beispielsweise eine nasse Fahrbahn oder schwer befahrbares Gelände.

Hazard Avoidance durchführen [PHI_BA3]

Inhärent sicher im Sinne der „Hazard Avoidance“ bedeutet Hazards zu vermeiden. Mit Hilfe des Hazard Triangle wird analysiert, ob das Hazard Element eliminiert werden kann und/oder ob die vorhandenen Initiation Mechanism auf null reduziert werden können, siehe dazu Kapitel 2.1.4 „Hazard Triangle“. Können Hazards vermieden werden so muss das Safety-Risiko-Assessment nicht mehr durchgeführt werden, da der Hazard nicht mehr vorhanden ist. Für diejenigen Hazards, die

nicht vermieden werden können, ist das Safety-Risiko-Assessment erforderlich. Wird dabei das Risiko als nicht-zulässig eingeschätzt, so sind mögliche Design-Lösungen zu finden, die das Auftreten von Hazards, in gefährlichen Situationen, *verhindern*, im Sinne der Hazard Prevention. Sind beide Lösungsschritte nicht möglich, so ist das Hazard Controlling, im Sinne der Funktionalen Sicherheit anzuwenden. Dabei wird eine Sicherheitsfunktion (Safety Function) spezifiziert und ein sicherer Zustand (Safe State) des Systems definiert.

Safety-Risiko-Assessment durchführen [PHI_BA4]

Die Basis-Aktivität für das Safety-Risiko-Assessment, besteht aus 4 Teilschritten:

1. Safety-Risiko-Bewertung durchführen [PHI_BA4.1]
2. Sicherheitsanforderungsstufe bestimmen [PHI_BA4.2]
3. Safety-Goals definieren [PHI_BA4.3]
4. Safety Requirements ableiten [PHI_BA4.4]
5. Safety-Maßnahmen definieren [PHI_BA4.5]

Die Durchführung des Safety-Risiko-Assessments ist der nächste logische Prozessschritt, der das Risiko jedes noch vorhandenen Hazards nach den Vorgaben der anzuwendenden Norm bewertet [PHI_BA4.1], um im zweiten Teilschritt die geforderte Sicherheitsanforderungsstufe [PHI_BA4.2] für die Systementwicklung definieren zu können. Das Vorgehen bei der Safety-Risiko-Bewertung kann von Safety-Norm zu Safety-Norm unterschiedlich sein. Die Bewertung und die darauf basierende Definition der Sicherheitsanforderungsstufe ist wesentlich, da diese großen Einfluss auf die Projektplanung und die Aufwände bei der Entwicklung des Systems hat. Zwei Beispiele sollen das verdeutlichen. So schreibt etwa die ISO 26262 [ISO26_11] beim System Design ab Sicherheitsanforderungsstufe ASIL C Inspections[52] anstatt Walkthroughs[53] vor, wie es der ASIL B fordert. Die IEC 61508 [IE08_10] fordert ab Sicherheitsanforderungsstufe SIL 3 eine vom Projektteam unabhängige Abteilung zur Durchführung von Safety Assessments, SIL 2 würde hier nur eine unabhängige Person fordern.

Auf Basis der identifizierten Hazards und der Sicherheitsanforderungsstufe werden im nächsten Schritt die Safety Goals [PHI_BA4.3] definiert, von denen die Safety Requirements [PHI_BA4.4] abgeleitet werden. Auf den Top-Level Safety Requirements werden die dafür erforderlichen Safety-Maßnahmen [PHI_BA4.5] festgelegt, welche die bewerteten Safety-Risiken auf ein akzeptables Level reduzieren.

Safety-Konzept erstellen [PHI_BA5]

Das Safety-Konzept ist die Zusammenführung der Ergebnisse aus den durchgeführten Safety-Analysen mit den darin definierten Safety Goals, den davon abgeleiteten Safety Requirements und Safety-Maßnahmen (Safety Mechanism). Besonders wichtig dabei ist, dass auch die Maßnahmen

[52] Eine Inspection ist nach der [ISO26_11] eine Überprüfung von Arbeitsprodukten nach einer formalen Prozedur.

[53] Ein Walk-through ist nach der [ISO26_11] eine systematische Überprüfung von Arbeitsprodukten und weniger formal als eine Inspection.

und/oder Bedingungen die zur Hazard Avoidance beigetragen haben entsprechend begründet und ausführlich dokumentiert sind. Darüber hinaus muss das Safety-Konzept auch die vorläufigen Design-Lösungen zur Hazard-Prevention genau begründen. Die Hazard-Controlling Mechanismen mit dem dafür erforderlichen „Safe State“ sind ebenso zu beschreiben. Vorläufige Design-Lösungen sind beispielsweise, redundante Systeme, Watchdog Counter oder Warnkonzepte.

PHI_Report verfassen und kommunizieren [PHI_BA6]

Die Ergebnisse der PHI werden in einem Report zusammengefasst und an das Projektmanagement sowie an alle relevanten Stakeholder berichtet. Damit wird der aktuelle Safety-Status offiziell kommuniziert und sorgt für eine einheitliche Sichtweise aller beteiligten Personen. Das ist besonders wichtig, da damit sichergestellt werden kann, dass, aus Sicht der beteiligten Personen, auf nichts vergessen wurde. Der Report soll auch sicherstellen, dass alle noch notwendigen Leistungen in der Planung berücksichtigt werden.

3.1.4 Zusammenfassung des Problemraums

Zusammenfassend wird der Ablauf des Problemraums anhand eines Flussdiagramms (Flow-Chart) in Abbildung 26 noch einmal in komprimierter Weise dargestellt. Der Ablauf beginnt mit der Spezifikation des Projektumfanges, also dem Statement of Work (SoW), das die Anforderungen des Kunden bzw. der relevanten Stakeholder auf Businesslevel beschreibt. Darauf basierend wird die Projekt-Kontext-Analyse (PKA) durchgeführt, um weitere Anforderungen und Rahmenbedingungen identifizieren zu können. Basierend auf dem SoW und der PKA kann die Projekt-Prozesslandschaft (PPL) entwickelt werden, die das strategische Vorgehen über den gesamten Projektverlauf darstellt. Darauffolgend wird die Konzeptionierung durchgeführt, in der das System abgegrenzt, die System-Umgebung identifiziert und der erste System-Entwurf ausgearbeitet wird. Das Resultat der Konzeptionierung ist das technische Konzept.

Basierend auf dem technischen Konzept beginnt die Durchführung der PHI, in der die Top-Level Hazards identifiziert werden. Diese sollen, im Sinne des Inherent System Safety Approach, soweit als möglich vermieden werden. Diejenigen die nicht vermieden werden können, werden einem Safety-Risiko-Assessment unterzogen. Darauf basierend wird für nicht vermeidbare Hazards, die Sicherheitsanforderungsstufe, das Safety Goal und die Safety Requirements ausgearbeitet sowie dafür notwendigen Safety-Maßnahmen definiert.

Anhand der Ergebnisse der PHI entscheidet sich, ob Änderungen im SoW, in der PKA und der PPL notwendig sind. Wenn ja, dann müssen die entsprechenden Basis-Aktivitäten in der Projekt-Initialisierung nochmals durchlaufen werden. Ergeben sich in der PHI zusätzliche Safety Requirements, so sind auch diese in einem nochmaligen Durchlauf, in der Konzeptionierung zu berücksichtigt und müssen in der PHI einer nochmaligen Safety-Analyse unterzogen werden. Im letzten Schritt erfolgt die Zusammenfassung aller drei Konzepte zum Makro-Konzept. Dieses muss unter Konfigurationsmanagement gestellt und entsprechend abgelegt werden.

Durch dieses systematisch-methodische Vorgehen können bereits in der Vor-Projektphase, also im Problemraum, Flawed Requirements vermieden werden. Damit wird der Grundstein für eine genaue Kostenschätzung gelegt sowie die Voraussetzungen für ein Inherent Safe System Design und eine dafür erforderliche Planung geliefert. Darüber hinaus kann mit den erarbeiteten Konzepten eine recht genaue Initial-Planung durchgeführt werden. Diese ist für eine realistische Kostenschätzung sehr wichtig.

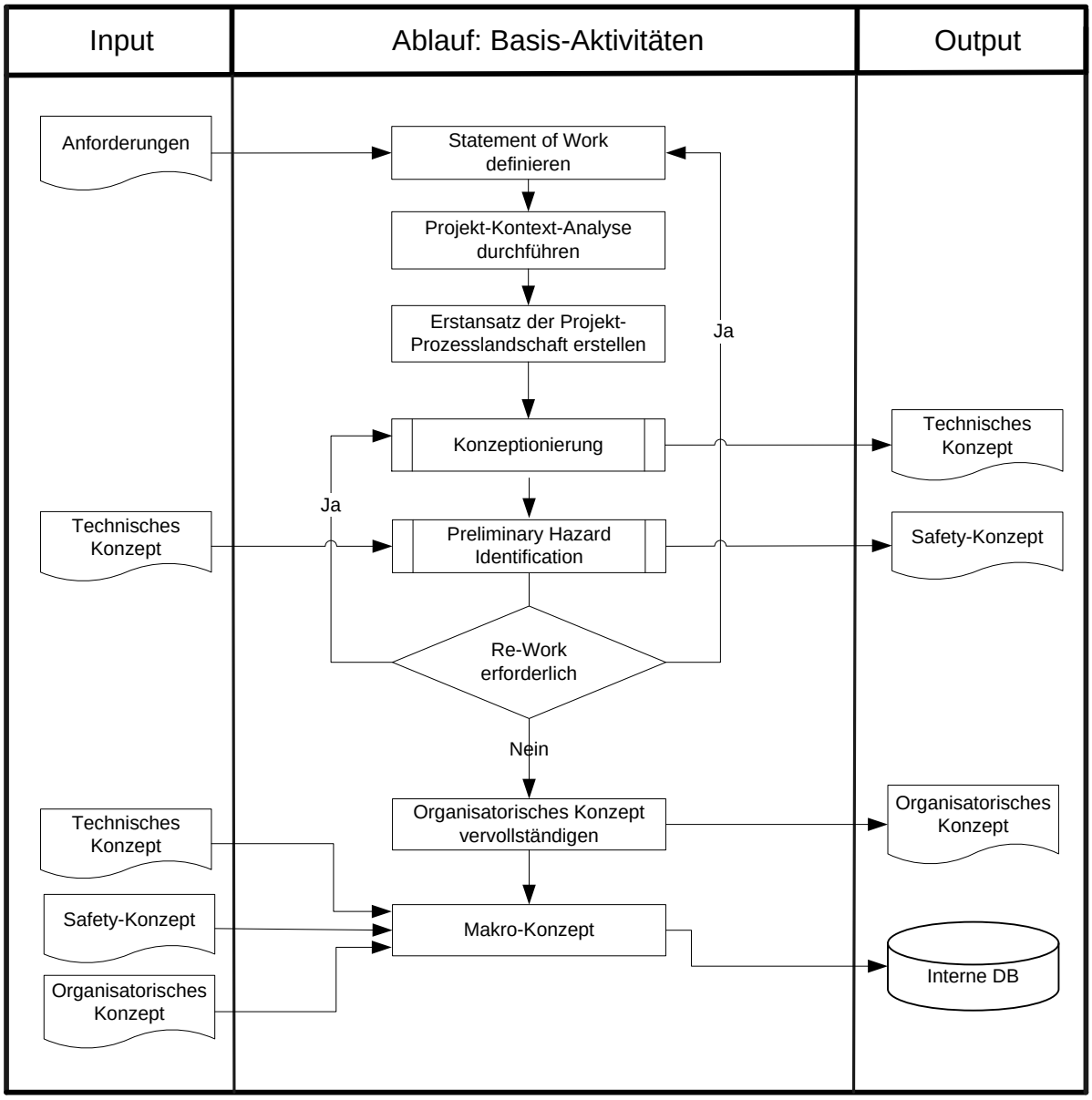


Abbildung 26: Ablauf des Problemraums als Flow-Chart

3.2 Modellierungsraum

Zweck des Modellierungsraums ist die Entwicklung des „Inherent Safe System Designs“, basierend auf den systematisch hergeleiteten Anforderungen aller identifizierten Stakeholder. Um dieses Inherent Safe System Design auch richtig umsetzen zu können, ist eine detaillierte, auf alle Disziplinen abgestimmte Planung zur Realisierung des Projektvorhabens über den gesamten Projekt-Lifecycle erforderlich. Die nachfolgende Abbildung 27 zeigt das Vorgehen im Modellierungsraum auf Prozessebene.

Input in den Modellierungsraum ist das Makro-Konzept aus dem Problemraum, bestehend aus dem organisatorischen, dem technischen und dem Safety Konzept. Output des Modellierungsraums sind einerseits die Spezifikation der System Requirements und das System Design, andererseits das Projekthandbuch mit allen erforderlichen Detailplanungen. Darin müssen die Engineering-, Safety- und Support-Aktivitäten (wie Qualitätssicherung, Konfigurationsmanagement, Verifikation und Validation) genau geplant sein. Wenn größere Projekte eine separate Planung der Safety- und Support-Aktivitäten erfordern, so müssen diese mit der Projektplanung verlinkt bzw. abgeglichen sein. Wesentlich sind auch in dieser Phase die Wechselwirkungen der parallel ablaufenden Prozesse, welche durch die Pfeile zwischen den Prozessen in der nachfolgenden Abbildung 27 dargestellt sind.

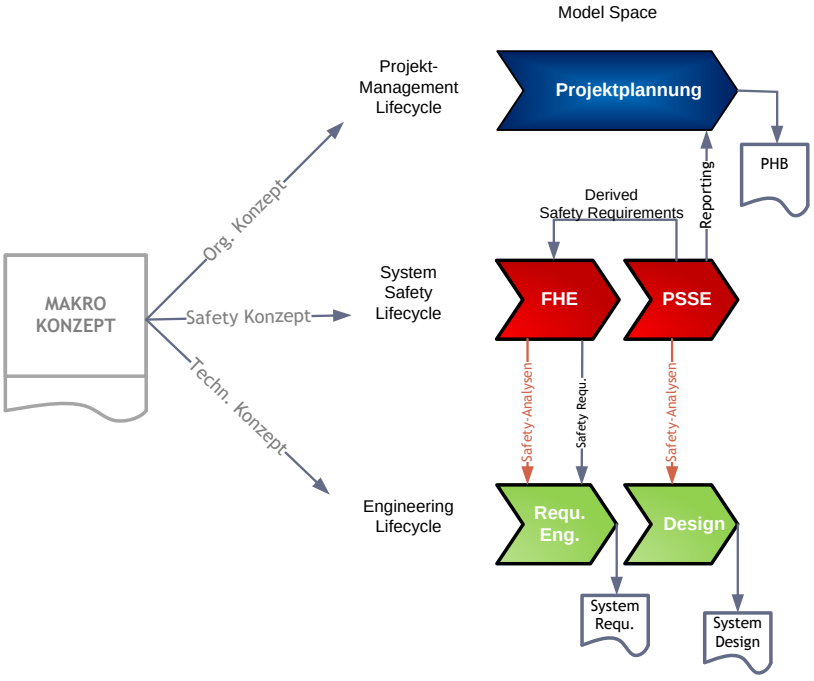


Abbildung 27: Prozesse im Modellierungsraum

Auf Projektmanagement-Ebene startet der Prozess der Projektplanung, welcher durch den Projektstart-Workshop ausgelöst wird. Der Prozess der Projektplanung strukturiert die Planungsaktivitäten und fordert als Output ein detailliertes Projekthandbuch (PHB). Damit kann einerseits das „Commitment[54]“ der Projektteam-Mitglieder sowie aller relevanter Stakeholder zum Projektvorhaben eingeholt und andererseits können damit die genauen Projektkosten berechnet werden. Die Freigabe des PHBs schließt den Projektplanungs-Prozess ab.

Im Requirements Engineering-Prozess werden die Anforderungen aller identifizierten Stakeholder des Projekts erhoben sowie die Rahmenbedingungen der System-Umgebung und des System-Umfeldes identifiziert und davon die technischen System Requirements abgeleitet. Die funktionalen System Requirements werden dann im Zuge der Functional Hazard Evaluation (FHE) einer detaillierteren Safety-Analyse unterzogen. Das Ergebnis dieser Analyse können zusätzliche Safety Requirements sein, die wiederum in die System Requirements Spezifikation eingesteuert werden müssen. Die Auswirkungen auf die Planung werden im Zuge des Reportings mit dem Projektplanungs-Prozess synchronisiert. Im nächsten Schritt werden die spezifizierten System Requirements nach den Vorgaben des System Design Prozesses realisiert. Das fertige System Design (SyD) wird im Zuge der Preliminary System Safety Evaluation (PSSE) mehreren Safety-Analysen unterzogen. Die Ergebnisse der Analysen werden wiederum an das Projektmanagement berichtet, in welchem die Planung entsprechend anpasst wird. Das freigegebene PHB schließt den Modellierungsraum ab.

Analog zum Problemraum wird auch hier der Modellierungsraum mit seinen Teil-Prozessen in seiner zeitlichen Abfolge dargestellt. Abbildung 28 zeigt die Schlüsselaktivitäten der oben beschriebenen Prozesse. Die Reihenfolge der Abarbeitung ist zwar prinzipiell flexibel und iterativ, die Darstellung zeigt jedoch einen idealtypischen Ablauf, der zeitlich von links nach rechts geordnet ist. In den nachfolgenden Unterkapiteln werden die einzelnen Lifecycle-Modelle sowie deren Wechselwirkungen zueinander im Detail beschrieben. Dieses modellhafte Vorgehen bedarf in der Praxis meist einer gewissen Anpassung.

[54] Unter dem Begriff „Commitment“ wird im Projektmanagement die Verpflichtung zur Selbstverpflichtung verstanden.

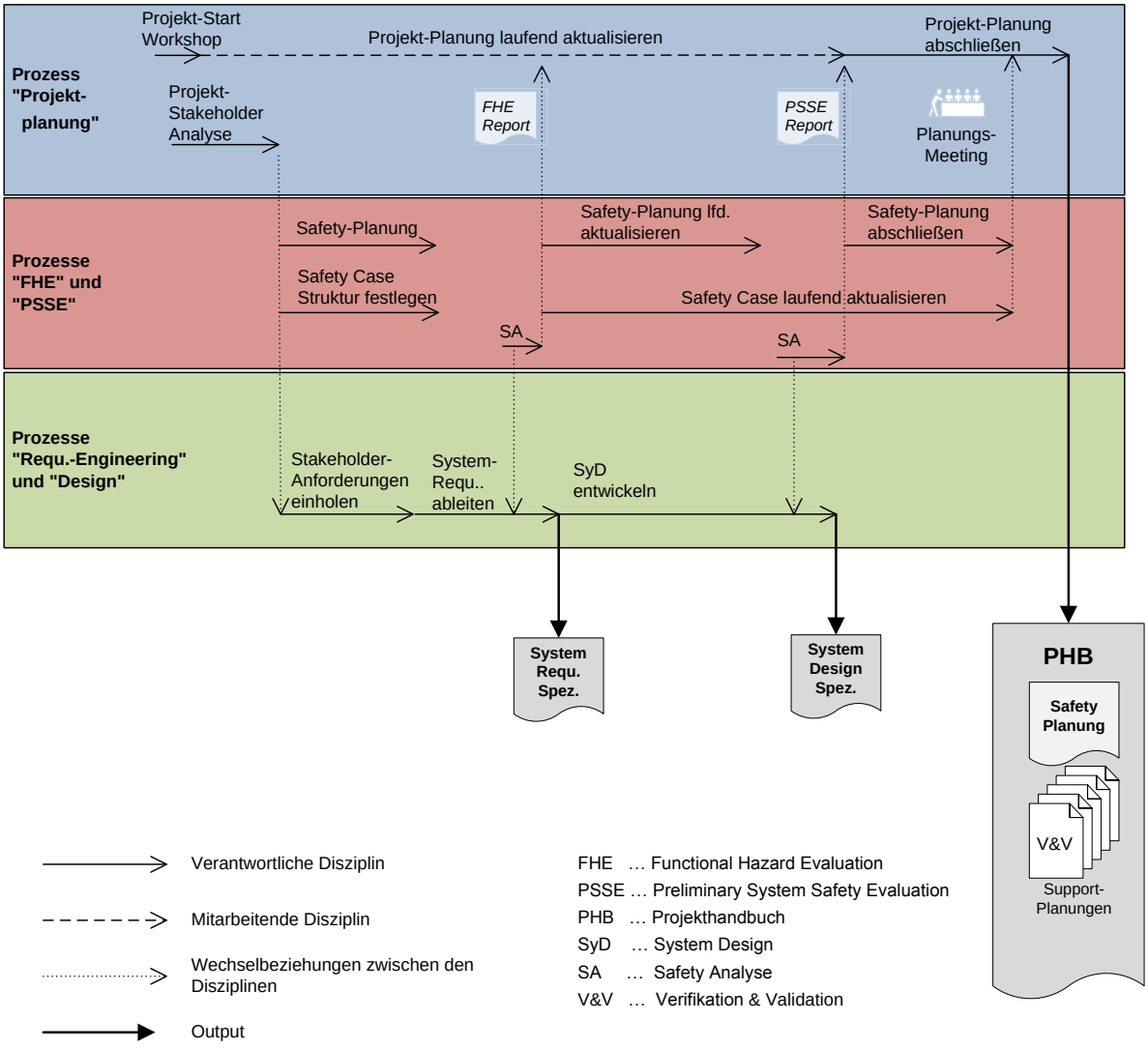


Abbildung 28: Zeitlich-systematischer Ablauf des Modellierungsraumes

Der Modellierungsraum startet, wie schon einführend erwähnt, mit einem Projektstart-Workshop, in dem der aktuelle Status aus dem Problemraum dargestellt wird und die nachfolgenden Schritte anhand der aktuellen PPL geplant bzw. durchgeführt werden können. Auf Basis der Projekt Stakeholder Analyse ergeben sich weitere Planungsschritte, dazu zählt auch die Detail-Planung der Safety-Aktivitäten. Des Weiteren wird die Grundstruktur für den Sicherheitsnachweis (Safety Case) aufgebaut.

Nachdem die Projekt-Stakeholder identifiziert wurden, werden im Engineering Lifecycle die Anforderungen eingeholt und davon die System Requirements abgeleitet. Die funktionalen System Requirements werden anschließend einer Safety-Analyse (SA) unterzogen. Dabei wird die Sicherheitsanforderungsstufe neu bewertet, und, wenn erforderlich, die Safety Requirements ergänzt bzw. erweitert. Die Analyse-Ergebnisse und die aktuelle Sicherheitsanforderungsstufe werden dem

Projektmanagement anhand des FHE-Reports kommuniziert. Darauf basierend wird die Planung, wenn erforderlich, verfeinert bzw. erweitert.

Mit der System Requirements Spezifikation (inkl. Safety Requirements) wird das System Design entwickelt und weiteren Safety-Analysen unterzogen. Ergeben sich daraus erweiterte/zusätzliche Safety Requirements, so müssen diese wiederum in die aktuelle System Requirements Spezifikation eingepflegt werden. Diese sind im System Design zu berücksichtigen und müssen im Zuge eines nochmaligen Durchlaufs der PSSE überprüft werden. Die Ergebnisse der gesamten Safety-Analysen werden dem Projektmanagement, anhand des PSSE-Reports berichtet, damit die Planung nach den Erfordernissen angeglichen werden kann. Die Konsolidierung und der Abschluss der Planung erfolgt im Planungs-Meeting, die Planungs-Ergebnisse werden im Projekthandbuch dokumentiert und freigegeben.

3.2.1 Projektplanung

Ziel der Projektplanung ist ein systematisch-methodisch vollständig durchdachtes und durchgeplantes Projekt. Damit können die Projekt-Leistungen genau abgeschätzt und die Projekt-Kosten berechnet werden. Dies ist nötig, da damit ein Vergleich mit den initial berechneten Kosten aus dem Problemraum (Projekt-Initialisierung) durchgeführt werden kann. Bei größeren Abweichungen besteht in dieser Phase des Projekts noch die Möglichkeit, die Kosten neu zu verhandeln, das Projekt zu unterbrechen um den Leistungsumfang zu reduzieren, oder, wenn nötig, das Projekt abzubrechen. Dafür wurde Stage Gate 2 eingeführt, da dies einer rigoroseren Entscheidung bedarf.

Die Projektplanung ist in 12 Basis-Aktivitäten untergliedert. Beginnend mit dem Projektstart-Workshop [PjP_BA1] wird die Planungsphase eingeleitet. Die nachfolgende Identifikation der Projekt-Stakeholder [PjP_BA2] liefert die Basis für die organisatorischen Einflüsse und Rahmenbedingungen, aber auch für das Requirements Engineering und ist wesentlich für die Aktualisierung der Planungsaktivitäten [PjP_BA3]. Die Projektplanung ist auch abhängig von den Safety-Aktivitäten der Safety-Planung [PjP_BA4] und von den Aktivitäten der Support-Planung [PjP_BA5]. Die Ergebnisse der vorhergehenden Basis-Aktivitäten und die PPL sowie das SoW aus dem Problemraum liefern die Inputs für die Leistungsplanung [PjP_BA6]. Die definierten Leistungen in den Arbeitspaketen des Projektstrukturplans stellen die Basis für die Leistungs-Abschätzungen [PjP_BA7] dar. Diese werden für die zeitliche Planung [PjP_BA8] herangezogen. Danach wird die Projekt-Organisation festgelegt [PjP_BA9] und das Projekt-Risiko eingeschätzt [PjP_BA10]. Mit der Einführung des Konfigurationsmanagements [PjP_BA11] und der damit verbundenen Installation des Change Control Boards kann das Projekthandbuch fertiggestellt und freigegeben werden [PjP_BA12]. Die nachfolgende Tabelle 8 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im Prozess der Projektplanung gefordert sind.

PjP_BA1	Projektstart-Workshop durchführen
PjP_BA2	Projekt-Stakeholder identifizieren
PjP_BA3	Projektplanung laufend aktualisieren
PjP_BA4	Safety Plan(ung) erstellen/integrieren
PjP_BA5	Pläne (Support-Prozesse) erstellen/integrieren
PjP_BA6	Leistungsplanung durchführen
PjP_BA7	Leistungs-Abschätzungen durchführen
PjP_BA8	Zeitliche Planung und Kostenplanung durchführen
PjP_BA9	Projektorganisation festlegen
PjP_BA10	Projekt-Risikomanagement durchführen
PjP_BA11	Konfigurationsmanagement definieren und institutionalisieren
PjP_BA12	PHB erstellen und freigeben

Tabelle 8: Basis-Aktivitäten der Projektplanung

Projektstart-Workshop durchführen [PjP_BA1]

Der Projektstart-Workshop ist wichtig, um das initiale Projekt-Team über den aktuellen Status des Projektvorhabens zu informieren und damit alle Projektteammitglieder auf einen einheitlichen Stand zu bringen. Dabei wird das Makro-Konzept vorgestellt und dieses in Bezug auf mögliche Änderungen aktualisiert. Das Scope of Work (SoW) beschreibt den Projekt-Umfang, der im Zuge der gesamten Planungsphase immer wieder aktualisiert wird. Die Projekt-Kontext-Analyse (PKA) ist ein wesentlicher Input für die Projekt-Stakeholder-Analyse, da darin schon in der Projekt-Initialisierung wichtige Stakeholder und deren Anforderungen erhoben sowie wichtige Rahmenbedingungen an das Projekt identifiziert wurden. Die Projekt-Prozesslandschaft (PPL) vermittelt dem Projektteam einen strategischen Gesamtüberblick über das Vorhaben. Sie beschreibt anhand der dargestellten Prozesse die Basis-Aktivitäten, die im Projekt zu leisten sind und zeigt die wichtigsten Ereignisse (Audits, Assessments, Stage Gates etc.) über den gesamten Projektverlauf auf. Damit ist unter anderem klar erkennbar, was die nächsten Schritte im Projektablauf sind. Das technische Konzept gibt einen Überblick über das zu entwickelnde System, seine System-Umgebung und seinem Projektumfeld. Das Safety-Konzept beschreibt die Top-Level Hazards, die Sicherheitsanforderungsstufe und die Top-Level Safety Requirements sowie die Safety-Maßnahmen im System-Entwurf. Dieser Überblick liefert die Ausgangsbasis für alle Projektteammitglieder. Das

ist besonders wichtig, da davon ausgehend die Aktivitäten im Modellierungsraum aufsetzen sollen. An der Schnittstelle vom Problemraum zum Modellierungsraum dürfen keine Lücken entstehen.

Projekt-Stakeholder identifizieren [PjP_BA2]

Ein wichtiger nächster Schritt im Modellierungsraum ist die Identifikation der Projekt-Stakeholder. Ziel und Zweck dieser Analyse ist die Identifikation aller Personen(gruppen), die den Projektablauf in irgendeiner Weise beeinflussen können, um darauf basierend entsprechende Maßnahmen planen zu können. Dieser Einfluss kann sowohl im positiven wie auch im negativen Sinn erfolgen. Es sind alle Stakeholder zu identifizieren, die am Erfolg des Projektes Interesse haben, aber auch solche, die das Projekt behindern können. Diese Analyse ist eine wichtige Voraussetzung für eine vollständige Projektplanung. Starke Einflussfaktoren im Zuge der Planung sind auch die technischen Anforderungen der Stakeholder, die im Zuge des Requirements Engineering erhoben werden.

Planung laufend aktualisieren [PjP_BA3]

Einen wesentlichen Einfluss auf die Projektplanung haben die Zwischenergebnisse aus den parallel laufenden Engineering- und System Safety-Aktivitäten. Die Ergebnisse aus FHE und PSSE fließen in Form von Reportings[55] - FHE-Report und PSSE-Report - in den Planungs-Prozess ein, wie in Abbildung 28 exemplarisch dargestellt. Die Safety-Analysen bewerten die Sicherheitsanforderungsstufe dabei immer wieder neu. Falls sich diese gegenüber der im Problemraum identifizierten Sicherheitsanforderungsstufe geändert hat, muss die Planung auf den aktuellen Stand gebracht werden.

Safety-Plan(ung) erstellen/integrieren [PjP_BA4]
Pläne (Support-Prozesse) erstellen/integrieren [PjP_BA5]

Werden die Safety-Aktivitäten separat geplant, so sind diese mit der übergeordneten Projektplanung zu synchronisieren und die wichtigsten Eckdaten in die Projektplanung aufzunehmen [PjP_BA4]. Dasselbe gilt auch für die Planung der Support-Aktivitäten [PjP_BA5], Qualitätssicherung, Konfigurationsmanagements, Verifikation und Validation.

Leistungsplanung durchführen [PjP_BA6]

Eine der wichtigsten Basis-Aktivitäten in der Projektplanung ist die Leistungsplanung. Davon werden alle weiteren Planungs-Aktivitäten abgeleitet. Die Hauptaufgaben des SoW sowie die PPL liefern die Grundlage für die Leistungsplanung, die im Projektstrukturplan (PSP) dargestellt wird. Die PPL gibt die Struktur für den PSP, über den gesamten Projektverlauf, vor. Die nachfolgende Abbildung 29 zeigt den von der PPL abgeleiteten PSP mit den dafür erforderlichen Projektphasen und den darunterliegenden Arbeitspaketen. Der PSP gliedert die Gesamtleistung in plan- und kontrollierbare Teil-Leistungen. Der adäquate Detaillierungsgrad ist immer von der jeweiligen

[55] Reportings (FHE-Report, PSSE-Report) enthalten die Ergebnisse der Safety-Analysen, wie z.B. Safety Requirements, Sicherheitsanforderungsstufe, aber auch Lösungsansätze auf der Design-Ebene, welche sich auf die Detail-Planung auswirken.

Projektsituation abhängig. Prinzipiell wird so weit ins Detail geplant, bis plan- und abschätzbare Arbeitspakete vorhanden sind.

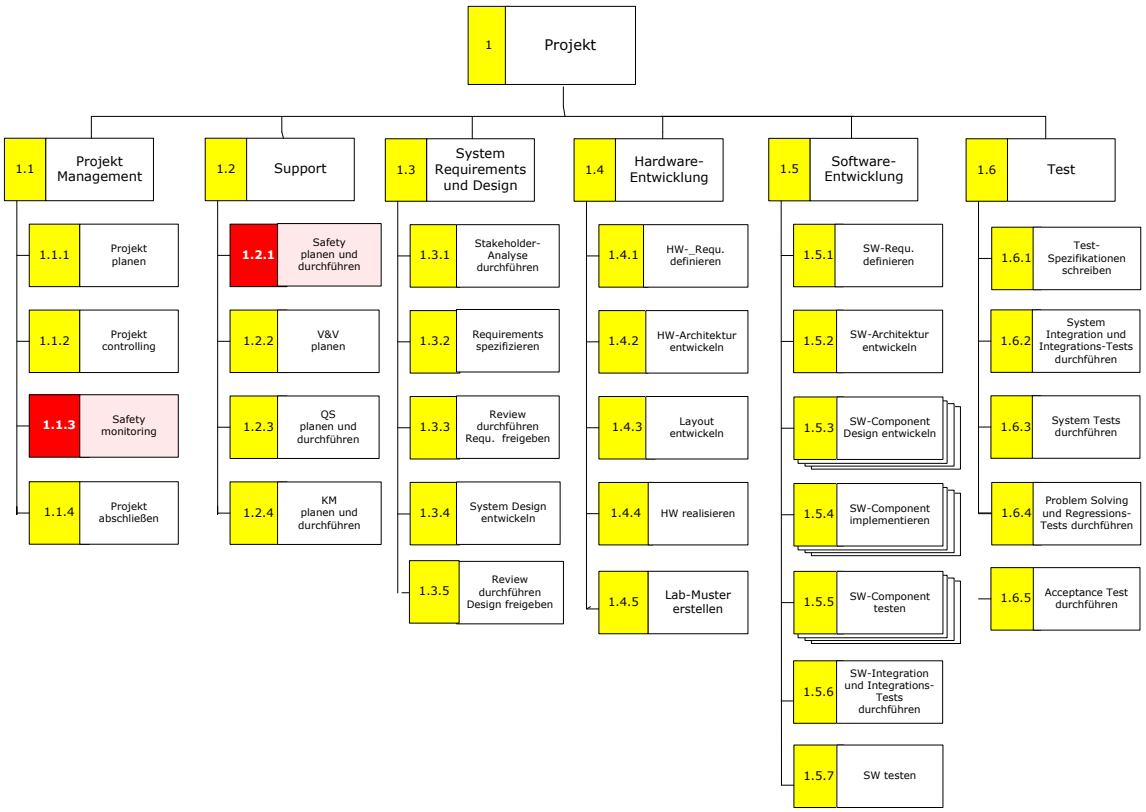


Abbildung 29: Generisches PSP-Pattern

Der dargestellte PSP wurde von der generischen PPL aus Abbildung 24 abgeleitet und kann damit als generisches PSP-Pattern wiederverwendet werden. Das PSP-Pattern besteht aus den Phasen Projektmanagement, Support, System Requirements und Design sowie Hardware-Entwicklung, Software-Entwicklung, der Test-Phase und den darunter angesiedelten Arbeitspaketen.

Die Projektmanagement-Phase ist direkt von den Prozessen des Projektmanagement-Lifecycles abgeleitet und besteht aus den Arbeitspakten „Projekt planen“, „Projekt controlling“ und „Projekt abschließen“. Das Safety-Vorgehensmodell integriert das Arbeitspaket „Safety monitoring“, in dem die Safety-Aktivitäten laufend überwacht werden. Die Phase „Support“ enthält alle unterstützenden Leistungen, die im Projektverlauf erforderlich sind. In dieser Phase wird das Arbeitspaket „Safety planen und durchführen“ dargestellt. Darin werden alle durchzuführenden Safety-Aktivitäten über den gesamten System Safety Lifecycle geplant. Dazu zählen auch Safety Assessments und Safety Audits. Arbeitspaket-Verantwortlicher ist hier der Safety-Engineer. Bei größeren Projekten, in denen eine größere Safety-Planung erforderlich ist, kann dafür eine eigene Phase dargestellt werden.

Zur Support-Phase zählen zudem noch die Arbeitspakete „V&V planen“, „Qualitätssicherung planen und durchführen“ und „Konfigurationsmanagement planen und durchführen“. Im Arbeitspaket „V&V planen“ werden die im Projekt erforderlichen Verifikations- und Validations-Aktivitäten, wie beispielsweise die zu planenden Reviews, Inspections und Tests über den gesamten Projekt-

Lifecycle geplant. Die Durchführung der Aktivitäten ist in eigenen Arbeitspaketen, wie beispielsweise „Review durchführen“, dargestellt bzw. in anderen Arbeitspaketen integriert, wie beispielsweise „System-Test durchführen“. Um Doppelaktivitäten zu vermeiden und den Arbeitsaufwand zu limitieren werden im Arbeitspaket „V&V planen“ auch die erforderlichen Safety Test-Aktivitäten mitgeplant und in den entsprechenden Arbeitspaketen in der Testphase durchgeführt.

Das Arbeitspaket „Qualitätssicherung planen und durchführen“ organisiert die von der ISO 15504 geforderten Assessments über den gesamten Projektverlauf. Die ISO 26262 [ISO26_11, Teil 2, Kapitel 6.4.8] fordert diesbezüglich die Durchführung von „Prozess-Audits“ und versteht darunter das gleiche Vorgehen wie die ISO 15504 [ISO15_11, Teil 2] unter dem Begriff „Qualitäts-Assessment“. Dabei handelt es sich um eine Bewertung der geforderten Prozesse und um die Überprüfung, ob diese im Projekt eingehalten werden. Diese Assessments bzw. Audits sind, je nach Sicherheitsanforderungsstufe, von einer unabhängigen Person bzw. Abteilung oder Institution vorzunehmen. Trotzdem die beiden Normen hier unterschiedliche Begriffe verwenden und auch unterschiedliche Prozesse überprüfen, stellt diese Arbeit den Anspruch, das Vorgehen zu vereinheitlichen.

Nachdem alle für das Projekt erforderlichen Prozesse mit der PPL abgedeckt werden, reicht ein Assessment, nach dem Vorgehen der ISO/IEC 15504 [ISO15_11, Teil 2], aus. Damit kann das geforderte Audit der ISO 26262 sowie das Assessment der ISO/IEC 15504 von einem Assessor bzw. einem Assessment-Team abgedeckt werden. Hiermit können sowohl die Effizienz gesteigert als auch Kosten eingespart werden. Dies ist jedoch nur möglich, da der ISaPro® sowohl die Aktivitäten der geforderten Safety-Norm als auch die Aktivitäten der ISO 15504 [ISO15_11, Teil 5] in einem Rahmenwerk mit definierten Prozessen abbildet. Wäre das Rahmenwerk des ISaPro® nicht vorhanden, so müssten die Prozesse des jeweiligen Safety Lifecycles – der von Norm zu Norm unterschiedlich ist – nach der jeweils erforderlichen Safety-Norm sowie die zusätzlichen Prozesse der ISO/IEC 15504 separat einem Audit und einem Assessment unterzogen werden.

Die IEC 61508 sowie die ISO 26262 fordern das Konfigurationsmanagement. Die IEC 61508 fordert es in Bezug auf Änderungen nach der Sicherheits-Validierung, die ISO 26262 über den gesamten Lifecycle. Löw [LPP_10, S.151] bestärkt diese Forderung damit, dass die im Projektverlauf erzeugten Arbeitsprodukte unter Konfigurationskontrolle stehen müssen. Die ISO/IEC 15504 stellt diesbezüglich einen eigenen Prozess zur Verfügung und zielt damit auf die Bewahrung der Integrität der Arbeitsprodukte in einem Projekt ab. Das heißt, es ist ein Konfigurationsmanagementsystem einzurichten, das die Arbeitsprodukte identifiziert und über das gesamte Projekt entsprechend lenkt. Das Arbeitspaket „Konfigurationsmanagement planen und durchführen“ enthält alle dafür erforderlichen Aktivitäten, siehe dazu auch Kapitel 2.4.1 “Konfigurationsmanagement“. Die Aktivitäten dafür werden vom Konfigurationsmanagement-Prozess abgeleitet. Die Planung enthält somit die Identifikation der unter dem Konfigurationsmanagement stehenden Arbeitsprodukte sowie die Planung der erforderlichen Baselines und Konfigurationsmanagement-Audits, wie diese auch schon in der PPL (siehe Abbildung 24) im Problemraum dargestellt wurden.

Die erforderlichen Aktivitäten für das System Requirements Engineering und die Entwicklung des System Designs, sind im PSP auf eine Phase zusammengeführt. Handelt es sich um größere

Projekte, können diese auch als eigene Phasen dargestellt werden. Die erforderlichen Basis-Aktivitäten die zur Erstellung der System Requirements Spezifikation erforderlich sind, werden vom Prozess Requirements Engineering abgeleitet. Die Basis-Aktivitäten zur Entwicklung für das System Design, werden vom gleichnamigen Prozess abgeleitet. Der in der PPL dargestellte Hardware- und der Software-Lifecycle wird im PSP in auf zwei Phasen aufgeteilt, da dafür mehrere Arbeitspakete notwendig sind. Die System-Integration und der System-Test sind in der PSP-Phase Test repräsentiert.

Da die Arbeitspaket-Inhalte von bewährten Prozessen abgeleitet werden, kann damit die Qualität der Planung stark erhöht werden. Wesentlich dabei ist, dass mit bewährten und reifen Prozessen „Process Flaws“ vermieden werden können. Damit kann im Modellierungsraum eine qualitativ hochwertige Planung für den weiteren Projektverlauf bestimmt werden.

Leistungs-Abschätzungen durchführen [PjP_BA7]

Der Projektstrukturplan mit seinen Arbeitspakten liefert die Basis für die Abschätzungen der zu erbringenden Leistungen im Projekt. Dabei wird jedes Arbeitspaket von den zuständigen Experten separat abgeschätzt. Damit wird die Basis für eine realistische Planung gelegt, die sich sowohl auf die zeitliche Planung als auch auf die Kostenplanung auswirkt. Werden die Abschätzungs-Ergebnisse am Projekt-Abschluss evaluiert, korrigiert und als Metrik-Daten abgelegt, so können diese in weiteren Projekten wiederverwendet werden. Damit kann die Projekt-Qualität erhöht und Planungslücken vermieden werden. Die standardisierten Prozesse des ISaPro® bieten dabei die Möglichkeit, die Metrik-Daten zu vereinheitlichen und je nach Erfordernis, diese zu verfeinern. Es sei an dieser Stelle noch erwähnt, dass für Abschätzungen im Software-Bereich professionelle Methoden und Software-Metriken, wie beispielsweise von Parthasarathy [Part_07] vorgeschlagen, verwendet werden sollen.

Zeitliche Planung und Kostenplanung durchführen [PjP_BA8]

Wie bereits erwähnt, werden die Zeit- und Kostenplanung von den abgeschätzten Leistungen der einzelnen Arbeitspakete des PSP abgeleitet. Es ist besonders wichtig, dass im Falle eines Re-Planings, die Reihenfolge - Arbeitspaket-Abschätzung, zeitliche Planung, Kostenplanung - einzuhalten ist. Damit kann sichergestellt werden, dass die Planungsaktivitäten in sich konsistent und lückenlos sind. Auf die Zeit- und Kostenplanung wird in dieser Arbeit nicht näher eingegangen, da sie zu den Standardaufgaben des Projektmanagements gehören und daher zum Neuigkeitsgrad dieser Arbeit wenig beitragen.

Projektorganisation festlegen [PjP_BA9]

Aus den Planungsergebnissen, welche auch die Rahmenbedingungen der jeweiligen Sicherheitsanforderungsstufe zu berücksichtigen haben, können die erforderlichen Kompetenzen (Skills und Erfahrungen) der Projektteammitglieder abgeleitet werden. Damit können die erforderlichen Projektrollen besetzt und die Projektorganisation festgelegt werden. Die Qualifikation der Projektteammitglieder wird durch den Prozess der Qualitätssicherung in den geforderten Assessments überprüft. Damit kann die organisatorische Machbarkeit auch im Projektverlauf

überprüft werden. Die notwendig Qualifikation ist bei Projekten im sicherheitskritischen Bereich besonders wichtig, da damit „Human Errors“ vermieden werden können.

Projekt-Risikomanagement durchführen [PjP_BA10]

Die bis hierher durchgeführten Planungsaktivitäten liefern die Basis für das Projekt-Risikomanagement. Es ist hier wichtig, dass die Projekt-Risiken nicht mit den technischen Safety-Risiken verwechselt oder vermischt werden. Obwohl technische Safety-Risiken, auf der organisatorischen Ebene der Projekt-Risiken ihre Wirkung zeigen können. Diese Risiken sind jedoch auf Projektmanagement-Ebene zu erkennen, wo auch die entsprechenden Maßnahmen zu treffen sind.

Jenny [Jen_09] definiert, dass das Projekt-Risikomanagement als Teil der Projektabwicklung zu sehen ist. Dabei ist das bewusste Einbeziehen und Bewältigen von möglichen, projektbezogenen Störfällen in der Projektabwicklung, auf Grundlage der systematischen Erfassung, Bewertung und Verfolgung von projektbezogenen Risiken, zu sehen. Jenny spricht in diesem Zusammenhang auch von der gezielten Wahrnehmung und Umsetzung von Chancen.

Des Weiteren unterscheidet Jenny zwischen Umsetzungsrisiken, Funktionsrisiken und sozialen Risiken im Projekt. Zum Beispiel werden *Umsetzungsrisiken* durch schlechte Planung und zu wenig geplanten Zeitaufwand für die Durchführungsaktivitäten verursacht. *Funktionsrisiken* durch mangelnde Auslegung der Anforderungen in Bezug auf Menge, Umfang, Funktion und Qualität. *Soziale Risiken* werden durch unzureichende Ziele-Definition und mangelnde Planung verursacht. Mit dem Safety-Vorgehensmodell wird in diesen Bereichen präventive Vorsorge getragen.

Auf das Projekt-Risikomanagement wird hier nicht näher eingegangen, da es zum State-of-the-Art im Projektmanagement gehört und in dieser Arbeit keinen Neuigkeitswert darstellt.

Konfigurationsmanagement definieren und institutionalisieren [PjP_BA11]

Spätestens zu diesem Zeitpunkt muss das Konfigurationsmanagement institutionalisiert werden (siehe dazu auch Kapitel 2.4.1). Nachdem die Output-Dokumente des Modellierungsraums freigegeben sind muss bereits die erste Baseline gezogen werden, in der das Projekthandbuch, die Requirements Spezifikation, das System Design und die Ergebnisse der Safety-Analysen eingefroren werden. Auch können nun alle weiteren wichtigen Arbeitsprodukte, die aufgrund der detaillierten Planung nun bekannt sind, miteinbezogen werden.

Des Weiteren muss das Change Control Board (CCB) eingerichtet werden. Ab der Freigabe der Projekt-Realisierung im Stage Gate 2 stehen damit alle Änderungen unter Change Control. Das heißt, alle Änderungen, über den gesamten Projektverlauf, müssen nun über das CCB genehmigt, entsprechend gesteuert und dokumentiert werden. Die Dokumentation der Änderungen hat auch wesentlichen Einfluss auf die Safety-Aktivitäten sowie auf den Safety Case. Werden Änderungen durchgeführt, so müssen diese nochmaligen Safety-Analysen unterzogen werden und sind im Safety Case darzustellen.

PHB erstellen und freigeben [PjP_BA12]

Sind die System Requirements Spezifikation und ein Inherent Safe System Design freigegeben sowie alle Safety-Analysen abgeschlossen, können die Planungsaktivitäten komplettiert werden. Dabei sind die freigegebenen Support-Aktivitäten bzw. deren Pläne in die Projektplanung zu integrieren und im PHB darzustellen. Die geplanten Aktivitäten werden danach im Planungs-Meeting mit dem Projektteam konsolidiert. Das fertig gestellte PHB wird danach einem Review unterzogen, mit allen betroffen Stakeholdern. Committen sich alle zur Planung, so kann das PHB freigegeben werden. Das Commitment, also die „Verpflichtung zur Selbstverpflichtung“ ist an dieser Stelle besonders wichtig, da damit der geplante Projektverlauf nicht mehr in Frage gestellt werden kann.

3.2.2 Requirements Engineering

Im Requirements Engineering Prozess werden die Anforderungen der Stakeholder an das zu entwickelnde System erhoben. Diese müssen mit den bereits identifizierten Top-Level Requirements aus dem Problemraum auf Konsistenz geprüft und wenn nötig erweitert bzw. verfeinert werden. Ziel des Requirements Engineering-Prozesses im Safety-Vorgehensmodell ist es, durch das systematisch-methodische Vorgehen, Lücken in den Requirements (Flawed Requirements) zu vermeiden.

Im gesamten Vorgehen spielen Safety Requirements eine besondere Rolle. Die System Requirements Spezifikation, in welcher die Safety Requirements enthalten sein sollen, liefert die Grundlage für die Entwicklung des Inherent Safe System Designs. Eine stringente systematische Vorgehensweise ist hier besonders wichtig, da kein Lösungsansatz im Design ohne dazugehörige System Requirements vorhanden sein darf.

Der Requirements Engineering-Prozess besteht aus 6 Basis-Aktivitäten. Er beginnt mit der Einholung der Stakeholder-Anforderungen und -Wünsche [RqE_BA1]. Darauf aufbauend werden die Stakeholder-Anforderungen und die Top-Level Requirements analysiert und konsolidiert [RqE_BA2]. Danach werden die System Requirements abgeleitet und beschrieben [RqE_BA3]. Im Prozessschritt 4 werden die System Requirements evaluiert [RqE_BA4]. Nach positiver Evaluierung werden die funktionalen System Requirements in der FHE einer Safety-Analyse unterzogen [RqE_BA5]. Falls dabei zusätzliche System Requirements identifiziert werden, müssen diese in die System Requirements Spezifikation eingearbeitet werden. Danach wird diese einem Review unterzogen, freigegeben und allen davon betroffenen Stakeholdern kommuniziert [RqE_BA6]. Die nachfolgende Tabelle 9 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im Requirements Engineering-Prozess gefordert sind.

RqE_BA1	Stakeholder-Anforderungen und –wünsche einholen
RqE_BA2	Stakeholder-Anforderungen und Top-Level Requirements analysieren und konsolidieren
RqE_BA3	System Requirements entwickeln/ableiten und beschreiben

RqE_BA4	System Requirements evaluieren
RqE_BA5	FHE durchführen
RqE_BA6	System Requirements freigeben und kommunizieren

Tabelle 9: Basis-Aktivitäten des Requirements Engineerings

Das Safety-Vorgehensmodell fordert eine hinreichende Anforderungsermittlung, welche durch ein systematisch-methodisches Vorgehen unterstützt wird. Werden in dieser Phase Anforderungen vergessen, so können sie weder einer Safety-Analyse unterzogen werden, noch können erforderlichen Safety Requirements im System Design berücksichtigt werden. Die Forderung nach einer hinreichenden Anforderungsermittlung ist wichtig, da diese Auswirkungen auf die Validation des System Designs haben. Ehret [Ehr_03] begründet dies damit: "The validation aims to give a design team confidence that their design fulfils the safety requirements, before the hardware of the system is built"[56].

Wichtig ist dabei auch die genaue Planung der dafür erforderlichen Aufwände, die für die Realisierung der spezifizierten Requirements notwendig sind. Ist nicht entsprechend Zeit dafür eingeplant, so können die Requirements auch nicht in der geforderten Qualität realisiert werden. Darüber hinaus entstehen, wie auch schon in der Projektplanung [PjP_BA10] erwähnt, auf Projektebene Umsetzungsrisiken, die im Risikomanagement oft erst viel zu spät erkannt werden. Mit einer systematischen Anforderungsermittlung und einer abgeglichenen Planung zur Realisierung, wie es das Safety-Vorgehensmodell vorgibt, können schon in sehr frühen Projektphasen „Flawed Requirements" und Projekt-Risiken vermieden werden. Eine genaue dokumentierte und strukturierte System Requirements Spezifikation liefern die notwendigen Argumente für die dafür nötigen Leistungen, die in der Planung berücksichtigt werden müssen.

Stakeholder-Anforderungen und -wünsche einholen [RqE_BA1]

Die Anforderungsanalyse beginnt mit der Ermittlung der Anforderungen aller definierten Stakeholder, die in der Projektplanung bereits identifizierten wurden (siehe dazu [PjP_BA2]). Hier ist es wichtig, sowohl auf der Projektmanagement-Ebene als auch auf der Engineering-Ebene die gleiche Klientel zu bedienen. Auch hier ist eine ganz eindeutige Abgrenzung notwendig, wer zu den Projekt-Stakeholdern gehört und wer nicht. Anforderungen der Systemumgebung müssen ebenso berücksichtigt werden und mit den Anforderungen der Stakeholder abgeglichen werden. So können, beispielsweise aufgrund genauerer Informationen des Einsatzbereichs, weitere Anforderungen in Bezug auf die geographischen und klimatischen Verhältnisse abgeleitet werden. Auch können über genauere Kenntnisse der gesetzlichen Rahmenbedingungen aus verschiedenen Ländern zusätzliche rechtliche Anforderungen entstehen. Gibt die Stakeholder Analyse zu wenig Auskunft über das

[56] "Die Validierung hat zum Ziel, dem Design Team Vertrauen zu geben, dass ihr Design die Sicherheitsanforderungen erfüllt, bevor die Hardware für das System gebaut wird." (Übersetzung des Verfassers)

User-Verhalten, so ist es notwendig, auch dieses genau zu untersuchen. Das Ergebnis kann sich dabei auf die Anforderungen des User-Interfaces auswirken, aber auch ein entsprechend richtiges Verhalten vom User fordern.

Stakeholder-Anforderungen und Top-Level Requirements analysieren und konsolidieren [RqE_BA2]

Im nächsten Prozessschritt werden die Stakeholder-Anforderungen analysiert und konsolidiert. Anforderungen auf Projekt-Ebene, wie beispielsweise geforderte Normen und Standards, oder Assessments und Audits, werden den Projekt-Zielen im Projekthandbuch zugeordnet, da sich diese auf mehrere Disziplinen auswirken können und damit über die Hauptaufgaben in die Leistungsplanung systematisch einfließen. Anforderungen an das System werden in definierten Kapiteln in der System Requirements Spezifikation beschrieben. Dabei müssen die Ergebnisse aus dem Problemraum mitberücksichtigt und mitanalysiert bzw. mitkonsolidiert werden, sodass hier kein Lücken entstehen und Überlappungen sichtbar werden.

System Requirements entwickeln/ableiten und beschreiben [RqE_BA3]

Die nachfolgende Abbildung 30 zeigt, wie die Top-Level Requirements aus dem Problemraum und die Stakeholder Anforderungen aus dem Modellierungs-Raum zusammengeführt und im Prozessschritt [RqE_BA2] analysiert und konsolidiert werden. Nach der Konsolidierung werden sie den vordefinierten Kapiteln der System Requirements Spezifikation zugeordnet. Es wird dabei grundsätzlich zwischen nicht-Safety-relevanten und Safety-relevanten Requirements unterschieden. Diese Trennung ist notwendig, um einerseits die Kriterien (Safety Requirements) für die „Safety Validation“ – wie sie die ISO 26262 [ISO26_11, Teil 4, Kapitel 9] und die IEC 61508 [IEC08_10, Teil 1, Kapitel 7.14] fordern – verfügbar zu haben, und andererseits die notwendige Priorisierung der Requirements vornehmen zu können, wie es auch Lee und Anderson [LA_90, S.33-34] empfehlen: „A specification on the top of the hierarchy imposes more stringent requirements on the design of the system than a specification lower hierarchy. For instance, one specification (top of the hierarchy) could cover critical system services, where a failure of such a kind of services may lead to a catastrophe. Then a second specification covers essential system services, where a failure would impair the system capabilities but not the safety, […]“[57]. Das heißt, Safety-Requirements bekommen hier die höchst Priorität und müssen auch in der System Requirements Spezifikation entsprechend erkennbar sein. Dies wird durch die Kapitelstruktur abgedeckt.

[57] „Eine Spezifikation höherer Priorität verlangt eine stringentere Erfüllung der Anforderungen an das Design des Systems, als eine niedrigere Priorität. Zum Beispiel kann eine Spezifikation höherer Priorität eher kritische System-Services abdecken, die zu einer Katastrophe führen können. Als eine niedriger priorisierte Spezifikation, die grundlegende Services zu erfüllen hat, die zwar die Systemfähigkeit beeinträchtigt, jedoch nicht die Sicherheit.“ (Übersetzung des Verfassers)

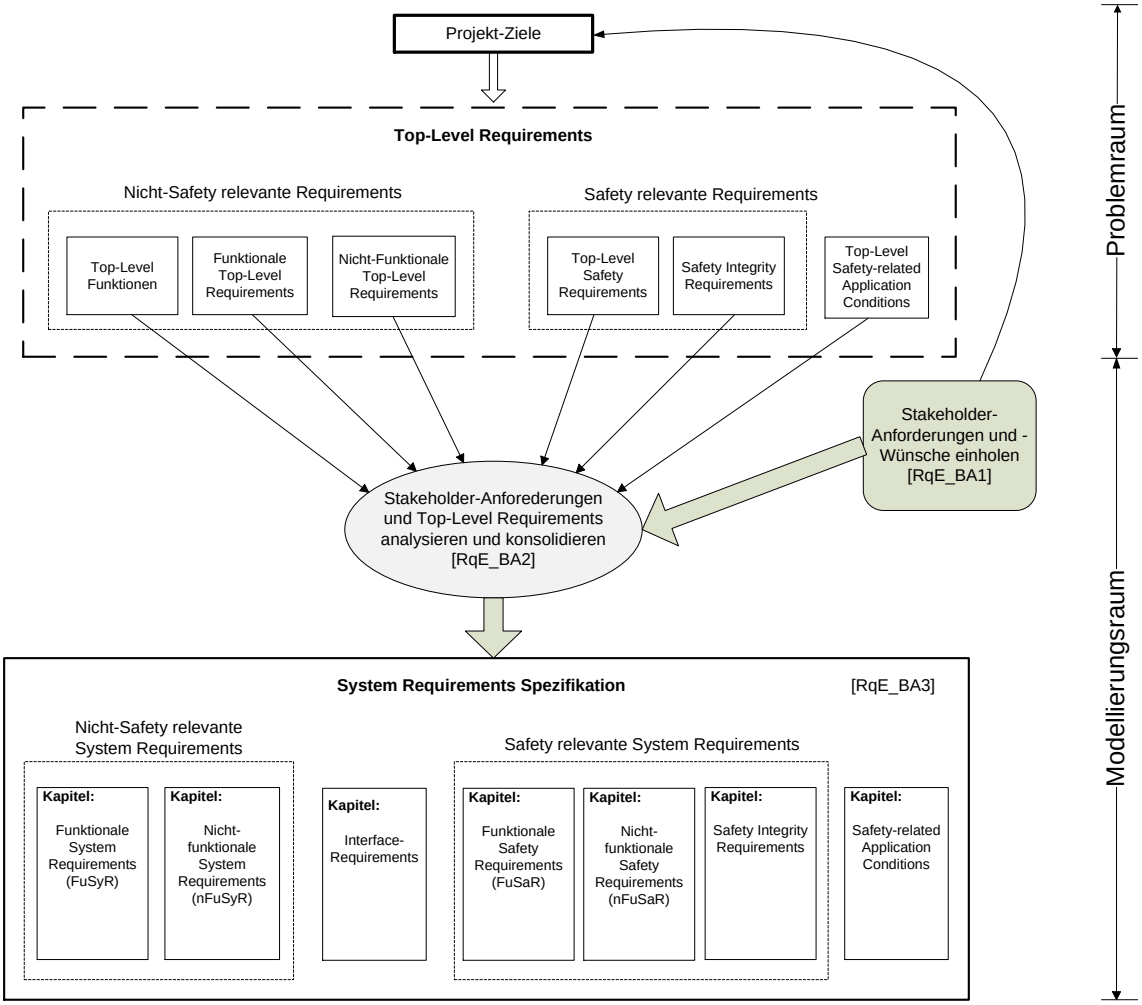


Abbildung 30: Requirements-Taxonomie

Beide Kategorien, also die nicht-Safety-relevanten und die safety-relevanten Requirements, werden somit, soweit in dieser Phase möglich, in funktionale und nicht-funktionale Requirements unterteilt. Die Unterscheidung definiert Sommerville [Som_95, Seite 64] so: "A requirement may be a *functional requirement*, that is, it describes a system service or function. Alternatively, it may be a *non-functional requirement.* A non-functional requirement is a constraint placed on the system (for instance, the required response time) or on the development process (such as the use of a specific language standard)."[58] Man kann also sagen, ein funktionales Requirement bezieht sich direkt auf die geforderte Funktion oder die zu leistenden Services eines Systems. Nicht-funktionale Requirements sind eher qualitative Anforderungen an das System bzw. dessen Funktionen, wie Antwortzeit, Benutzbarkeit, Erlernbarkeit, Stabilität oder auch Sicherheit im Sinne von Security und

[58] "Ein Requirement kann ein *funktionales Requirement* sein, das heißt, es beschreibt einen Systemservice oder eine Systemfunktion. Alternativ kann es ein *nicht-funktionales Requirement* sein. Ein nicht-funktionales Requirement ist eine Einschränkung einer Systemfunktion (beispielsweise die erforderliche Antwortzeit) oder des Entwicklungsprozesses (wie die Verwendung eines bestimmten Sprachstandards)." (Übersetzung des Verfassers)

natürlich auch Safety. Safety Integrity Requirements, also die geforderte Sicherheits-anforderungsstufe oder auch geforderte Standards, sind also als nicht-funktonale Requirements zu klassifizieren.

Die nicht-safety-relevanten Requirements werden in die funktionalen System Requirements (FuSyR) und in die nicht-funktionalen System Requirements (nFuSyR) unterteilt; ebenso die safety relevanten Requirements, *soweit möglich*, in die funktionalen Safety Requirements (FuSaR) und in die nicht-funktionalen Safety Requirements (nFuSaR). Ist nicht eindeutig ersichtlich, ob es sich um ein funktionales Safety Requirement handelt, so wird es in dieser Phase vorerst als nicht-funktionales Safety Requirement bezeichnet. Dieser Schritt ist wichtig, da ein Safety Requirement durch Operationalisierung auch zu einer Pre- oder Post-Condition anderer Requirements werden kann. Wichtig bei der Spezifikation der System Requirements sind die Safety-Integrity Requirements sowie die externen Interface-Requirements und die Safety-related Application Conditions (SAC).

Auch in dieser Phase des Lifecycles ist diese Unterscheidung unbedingt notwendig, da die Requirements gezielt den Safety-Analysen zugeführt werden sollen. Funktionale Requirements, die einen externen Trigger besitzen, werden in der FHE auf mögliche gefährliche Fehlfunktionen bzw. Funktionsausfälle analysiert.

Bei der Strukturierung der System Requirements wird das Systemkonzept von Ropohl [Rop_09] (siehe Kapitel 2.1.1 „Grundbegriffe der technischen Systemtheorie“) berücksichtigt, um eine einheitliche Struktur bei der Entwicklung des System Designs und den Safety-Analysen aufzuweisen. Die System Requirements werden nach dem funktionalen, dem strukturalen und dem hierarchischen Systemkonzept eingeteilt. Die funktionalen Requirements stellen das funktionale Systemkonzept dar. Das strukturale Systemkonzept wird durch die internen Sub-Systeme des Systems dargestellt, denen die Requirements on Elements zugeteilt werden. Das hierarchische Systemkonzept beschreibt die Einbettung des Systems in ein Super-System, hier werden Requirements der externen Schnittstellen zu anderen Systemen beschrieben. Diese Unterscheidung ist wichtig, da bei den Safety-Analysen im nächsten Kapitel die gleiche Struktur gewählt wird.

Da die Abgrenzung zwischen funktionalen und nicht-funktionalen Requirements auf System-Ebene nicht immer eindeutig zu ziehen ist, wird hier ein weiterer Schritt definiert, der diese Unterscheidung klar herausarbeitet. Dieser Schritt wird in dieser Arbeit mit „Operationalisierung“ bezeichnet. Die Operationalisierung führt zu klar getrennten funktionalen und nicht-funktionalen Requirements.

Anhand des Trigger-Service Modells kann die Unterscheidung zwischen funktionalen und nicht-funktionalen Requirements getroffen werden. Das heißt, gibt es einen eindeutigen Trigger, der das System von außen anstößt und darauf basierend das System ein Service liefern muss, handelt es sich um ein funktionales Requirement. Ein Beispiel soll das verdeutlichen. Ein Requirement auf oberster Ebene könnte sein: „Das zu entwickelnde System muss die Möglichkeit eines Software-Updates zur Verfügung stellen.“ In der Operationalisierung würde daraus ein klares funktionales Requirement werden: „Wenn das Maintenance-Personal den ODC-Stecker ansteckt und über das Bus-System den Befehl zum Software-Update sendet (Trigger), muss das zu entwickelnde System die Software-Update Prozedur starten und durchführen (Service).“

Ist nicht eindeutig erkennbar woher der Trigger kommt, ist das Requirement vorerst als nicht-funktionales Requirement zu spezifizieren. Auch Requirements mit internem Trigger werden in dieser Phase als nicht-funktionale Requirements formuliert, da die genauen Systeminhalte (interner Systemaufbau) noch nicht ausspezifiziert sind. Dieser Schritt ist notwendig, weil in der FHE nur die funktionalen Requirements in Bezug auf ihre Systemumgebung untersucht und damit die zu analysierenden Requirements klar abgegrenzt werden.

System Requirements evaluieren [RqE_BA4]

Sind alle definierten Requirements in die System Requirements Spezifikation eingearbeitet, so werden sie einer Evaluierung unterzogen. Die spezifizierten System Requirements werden auf Basis des technischen System-Entwurfs und des Shell-Modells evaluiert. Darüber hinaus werden die Erfüllung der geforderten Rahmenbedingungen und die der gesetzlichen Vorgaben kontrolliert und die Machbarkeit der System Requirements nochmals in Frage stellt. Erst wenn diese Kriterien erfüllt sind, kann die FHE durchgeführt werden.

FHE durchführen [RqE_BA5]

Nach positiver Evaluierung der System Requirements werden die funktionalen Requirement einer Safety-Analyse unterzogen, die im nachfolgenden Kapitel ausführlich beschrieben wird. Werden in der FHE noch weitere (Safety-)Requirements definiert, so müssen diese in die System Requirements Spezifikation eingearbeitet und neuerlich evaluiert werden.

System Requirements freigeben und kommunizieren [RqE_BA6]

Im letzten Prozessschritt wird, mit allen von den System Requirements betroffenen Stakeholdern, ein Review durchgeführt. Je nach definierter Sicherheitsanforderungsstufe kann das jedoch auch ein Walkthrough oder eine Inspection sein. Damit wird sichergestellt, dass die spezifizierten Requirements Gültigkeit haben und so akzeptiert wurden. Die Stakeholder bestätigen damit die Vollständigkeit und Richtigkeit ihrer Anforderungen, die System-Architekten deren Umsetzbarkeit. Damit können die System Requirements zur Umsetzung freigegeben werden. Nach der offiziellen Freigabe, wird die System Requirements Spezifikation allen betroffenen Stakeholdern kommuniziert. Mit diesem Vorgehen wird auch an dieser Stelle den Flawed Requirements entgegengewirkt.

3.2.3 Functional Hazard Evaluation (FHE)

Ziel der Functional Hazard Evaluation ist es, die funktionalen Requirements in ihrer System-Umgebung einer detaillierten und ausreichenden Safety-Analyse zu unterziehen. Der Bezug vom zu entwickelnden System auf das Gesamtsystem und zum Systemumfeld soll dabei hergestellt werden. Dabei werden, Fehlfunktionen sowie Funktionsausfälle und deren Auswirkungen auf die System-Umgebung und dessen System-Umfeld analysiert. Damit soll sichergestellt werden, dass gefährliche Fehlfunktionen im operationalen Bereich verhindert werden. So soll auch gewährleistet werden, dass die funktionalen System Requirements keine Lücken in Bezug auf ihre Systemumgebung aufweisen.

Dieser Prozess liefert einen wesentlichen Beitrag für ein Inherent Safe System Design und dessen Systemverhalten in seinem System-Umfeld.

In der FHE sind 5 Prozessschritte spezifiziert. Der erste Prozessschritt fordert die Safety-Planung [FHE_BA1], die vom Safety-Ingenieur durchzuführen ist. Prozessschritt zwei fordert die Festlegung der Struktur des Safety Cases [FHE_BA2]. Darauffolgend werden die funktionalen Anforderungen in Bezug auf die System-Umgebung und das System-Umfeld auf mögliche Hazards untersucht [FHE_BA3]. Die identifizierten Hazards werden im nächsten Prozessschritt auf Hazard Avoidance analysiert [FHE_BA4], wie er schon im Problemraum in der PHI beschrieben wurde. Diejenigen Hazards, die nicht vermieden werden können werden müssen einem Safety-Risiko-Assessment unterzogen [FHE_BA5] werden, in der die Sicherheitsanforderungsstufe festgelegt wird und die Safety Requirements definiert werden. Der Prozessschritt 6 den kommunizierten FHE-Report [FHE_BA6]. Die nachfolgende Tabelle 10 gibt einen Gesamtüberblick über die Basis-Aktivitäten im Prozess der FHE.

FHE_BA1	Safety-Planung durchführen
FHE_BA2	Safety Case Struktur festlegen
FHE_BA3	Hazards identifizieren
FHE_BA4	Hazard Avoidance durchführen
FHE_BA5	Safety-Risiko-Assessment durchführen
FHE_BA6	FHE-Report verfassen und kommunizieren

Tabelle 10: Basis-Aktivitäten der FHE

Safety-Planung durchführen [FHE_BA1]

Im Prozessschritt der Safety-Planung werden die Ergebnisse der Stakeholder-Analyse aus Safety-Sicht betrachtet und die dafür nötigen Planungsaktivitäten durchgeführt. Wenn nötig, müssen hier weitere Informationen von den Stakeholdern, im Speziellen von den Zertifizierungsbehörden bzw. unabhängigen Assessoren, eingeholt werden. Damit können die zu leistenden Aktivitäten in die Safety-Planung aufgenommen werden. Das sind beispielsweise der dafür erforderlichen Zertifizierungsprozess sowie die dafür notwendigen Confirmation Measures (Audits, Assessments und Confirmation Reviews), aber auch die Erstellung der geforderten Evidenzen für den Safety Case.

Safety Case Struktur festlegen [FHE_BA2]

Der Safety Case und dessen Dokumentationsstruktur muss schon in der Projektplanungsphase festgelegt werden. Ein vollständiger systematisch geführter Safety Case soll beweisen, dass das System für seinen Einsatzzweck ausreichend sicher ist. Um diesen Beweis führen zu können, müssen von Beginn an alle dafür relevanten Arbeitsprodukte - die für den Safety Case als Evidenzen

dienen - identifiziert und unter Konfigurationsmanagement gestellt werden. Damit wird die Integrität der Evidenzen für den Safety Case sichergestellt und kann jederzeit nachvollzogen werden. Wichtig dabei ist, dass die Struktur des Safety Cases - mit Goals, Argumenten und Evidenzen - so genau als möglich vordefiniert ist. Hier leistet die PPL eine sehr große Hilfestellung, da davon die Struktur des Projektverlaufs und alle erforderlichen Arbeitsprodukte abgeleitet werden können. Ist die Struktur klar dargestellt und diese auf das Projektvorgehen (PPL) abgestimmt, brauchen die Evidenzen, im Projektverlauf, „nur" mehr entsprechend eingepflegt werden, (siehe dazu Kapitel" 2.6 „Safety Case").

Hazards identifizieren [FHE_BA3]

In der FHE des Safety-Vorgehensmodells werden nach der Erstellung der System Requirements schon Safety-Analysen gefordert. Die Analysen basieren auf den, in dieser Phase identifizierten, funktionalen Requirements. Dabei werden die bis dato spezifizierten System-Funktionen auf mögliche Hazards analysiert. Hierfür eignet sich wiederum eine explorative Hazard Analyse, wie beispielsweise die HAZOP. Damit kann auf die, in der PHI durchgeführten Analyse nahtlos aufgesetzt werden. Zur Identifikation der Auswirkungen werden die System-Umgebung und das System-Umfeld, mit dem darin definierten Einsatzbereich, miteinbezogen. In Bezug auf das User-Interface wird der „Foreseeable Misuse", wie ihn die IEC 61508 [IEC08_10, Teil 4, Kapitel 3.1.14] fordert, mitanalysiert.

Hazard Avoidance durchführen [FHE_BA4]

Auch in dieser Phase des Projekts wird der Versuch gestartet Hazards zu vermeiden, wie es auch im Prozessschritt 3 der PHI [PHI_BA3] schon gefordert wurde.

Safety-Risiko-Assessment durchführen [FHE_BA5]

Für die noch vorhandenen, neu identifizierten Hazards erfolgt die Durchführung des Safety-Risiko-Assessments. Auch hier wird die Sicherheitsanforderungsstufe nochmals bestimmt und darauf basierend die noch erforderlichen Safety Goals und Safety Requirements definiert sowie die notwendigen Safety-Maßnahmen abgeleitet.

FHE-Report verfassen und kommunizieren [FHE_BA6]

Die Dokumentation und Kommunikation der Ergebnisse im FHE-Report schließen den FHE-Prozess ab. Auch hier muss, basierend auf den Ergebnissen der FHE, die Planung angepasst werden.

3.2.4 System Design

Ziel dieses Prozesses ist, basierend auf den System Requirements (mit den darin inkludierten Safety Requirements) und dem System-Entwurf aus dem Problemraum, ein Inherent Safe System Design (SyD) zu entwickeln. Das SyD beschreibt die Sub-Systeme und die Schnittstellen, aus denen das System besteht. Die System Requirements können damit den definierten Sub-Systemen zugewiesen und weiter ausspezifiziert werden. Das SyD, welches hier im Modellierungsraum entwickelt wird, kann bei komplexen Systemen im Lösungsraum noch einen weiteren Detaillierungsgrad fordern.

Dabei werden die Sub-Systeme des SyD in weitere Hardware- und/oder Software-Komponenten unterteilt und ausspezifiziert.

Der System Design Prozess des ISaPro® ist in 6 Basis-Aktivitäten gegliedert. Das ist im ersten Prozessschritt die Entwicklung des System Designs [SyD_BA1], in dem externe Schnittstellen, Sub-Systeme und interne Schnittstellen definiert und spezifiziert werden. Danach werden alternative Lösungen evaluiert [SyD_BA2], um sicherzustellen, dass eine hinreichend sichere Design-Lösung gefunden wurde. Wird die Design-Lösung als „safe“ bestätigt, dann werden die System Requirements den Sub-Systemen des SyD zugeteilt [SyD_BA3]. Hier spricht das Safety-Vorgehensmodell von „Requirements on Elements“. Anschließend wird das System Design anhand der funktionalen Requirements verifiziert [SyD_BA4], um die Vollständigkeit des SyD und die Traceability zu den Requirements sicherzustellen. Damit ist die Basis zur Durchführung der PSSE gelegt, in der, basierend auf dem SyD, Safety-Analysen durchgeführt werden. Wird nun festgestellt, dass das SyD nicht inhärent sicher ist, müssen weitere Safety Requirements, hier als „*Derived Safety Requirements*“ bezeichnet, definiert und in das SyD eingearbeitet werden [SyD_BA5]. Die Freigabe und Kommunikation [SyD_BA6] des SyD schließt den Prozess ab. Die nachfolgende Tabelle 11 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im SyD Prozess gefordert sind.

SyD_BA1	System Design Architektur entwickeln
SyD_BA2	Alternative Lösungen evaluieren
SyD_BA3	System Requirements den Sub-Systemen zuteilen
SyD_BA4	System Design verifizieren
SyD_BA5	PSSE durchführen
SyD_BA6	System Design freigeben und kommunizieren

Tabelle 11: Basis-Aktivitäten des System Design Prozesses

System Design Architektur entwickeln [SyD_BA1]

Der Prozess des SyD legt im ersten Prozessschritt die System-Architektur fest. Dabei ist der technische System-Entwurf aus dem Problemraum ein wichtiger Input. Ausgangsbasis für den ersten Prozessschritt sind die definierten System Requirements, von denen, unter Berücksichtigung des System-Entwurfs, die externen Schnittstellen abgeleitet und ausspezifiziert werden. Die externen Schnittstellen sind Ausgangspunkt für die Definition der Sub-Systeme. Sind alle erforderlichen Sub-Systeme bestimmt, lassen sich die internen Schnittstellten festlegen. Es ist bei der Entwicklung des SyD besonders darauf zu achten, dass es struktural aufgebaut ist, wie in Kapitel 2.1.1 „Grundbegriffe der technischen Systemtheorie“ beschrieben. Damit bietet SyD die Möglichkeit, durch Dekomposition die sicherheitskritischen Sub-Systeme zu kapseln.

Alternative Lösungen evaluieren [SyD_BA2]

In diesem Prozessschritt fordert das Safety-Vorgehensmodell die Evaluierung des SyD anhand von alternativen Lösungen. Dabei sollen, wenn vorhandene, ähnliche Lösungen analysiert, oder alternative Lösungen angedacht werden, um, wie schon erwähnt, durch Dekomposition sicherheitskritische Funktionen in eigene Sub-Systemen kapseln zu können.

System Requirements den Sub-Systemen zuteilen [SyD_BA3]

In diesem Prozessschritt werden die System Requirements den Sub-Systemen zugeteilt und werden damit als „Requirements on Elements (RoE)“ bezeichnet. Bei der Zuteilung müssen die RoE entsprechend ausspezifiziert werden. Können bestimmte System Requirements nicht zugeteilt werden, so müssen sie als Safety Application Conditions spezifiziert und den betroffenen Stakeholdern zugewiesen werden. Hier ist es wichtig, dass dieser Part eindeutig spezifiziert wird und die Bestätigung vom betroffenen Stakeholder eingeholt wird.

System Design verifizieren [SyD_BA4]

In diesem Prozessschritt wird die funktionale Sichtweise des SyD mit der strukturalen Sichtweise verifiziert. Dabei werden alle definierten funktionalen Requirements im System Design verifiziert. Damit wird einerseits sichergestellt, dass alle System Requirements in das SyD eingearbeitet sind und andererseits wird die Design-Entscheidung analysiert und bewertet.

PSSE durchführen [SyD_BA5]

Das fertig entwickelte SyD wird nun, anhand der PSSE, weiteren Safety-Analysen unterzogen, welche im nachfolgenden Kapitel genau beschrieben sind. Werden dabei zusätzliche Hazards identifiziert, so werden dafür sogenannte „Derived Safety Requirements“ definiert, die in die System Requirements Spezifikation und in das SyD eingearbeitet werden müssen. Dafür ist ein Iterationszyklus über die Prozesse, Requirements Engineering und FHE notwendig, wie es in der nachfolgenden Abbildung 31 dargestellt ist.

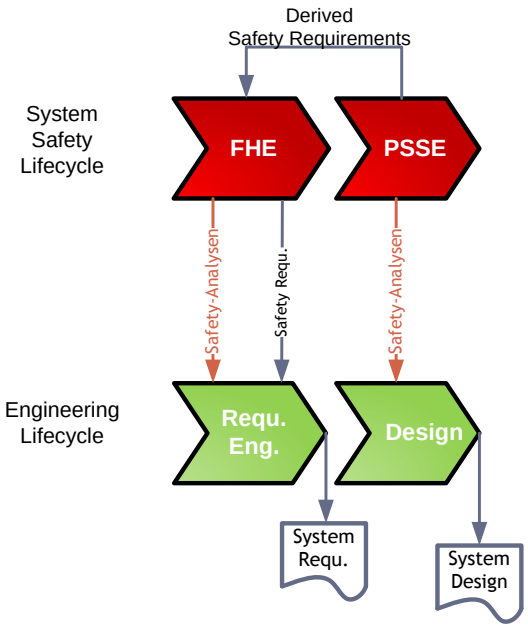


Abbildung 31: Derived Safety Requirements im Modellierungsraum

Wichtig dabei ist, dass die FHE und die PSSE die Wirksamkeit des Derived Safety Requirement auch bestätigen und dass die definierten Maßnahmen im SyD berücksichtigt werden. Bei komplexeren Systemen können hier auch mehrere Iterationen notwendig sein.

System Design freigeben und kommunizieren [SyD_BA6]

Der letzte Prozessschritt fordert die Freigabe und Kommunikation des SyD. Je nach Sicherheitsanforderungsstufe werden auch hier ein Walkthrough, ein Reviews oder eine Inspection gefordert. Dabei sollen alle betroffenen Stakeholder, wie beispielsweise die Hard- und Software-Entwickler, der verantwortliche Test-Engineer und der Safety-Engineer, aber auch der Requirements-Engineer teilnehmen. Damit wird sichergestellt, dass die geforderten Requirements im SyD richtig umgesetzt wurden und dass das SyD in Hard- und Software realisiert werden kann. Darüber hinaus wird vom Test-Engineer die Integrations- und Testbarkeit bestätigt.

3.2.5 Preliminary System Safety Evaluation (PSSE)

Um eine hinreichende Evaluierung durchführen zu können, ist ein gut spezifiziertes und strukturiertes SyD, wie es im vorigen Kapitel beschrieben wurde, notwendig, was wiederum eine ausreichende System Requirements Spezifikation voraussetzt. Die Evaluierung soll in diesem Kapitel durch das systematisch-methodische Vorgehen sowie durch den modularen Aufbau mit strukturaler, funktionaler und hierarchischer System-Architektur, nach Ropohl [Rop_09], sichergestellt werden. Die Preliminary System Safety Evaluation (PSSE) überprüft nun, ob die in der System Requirements Spezifikation definierten Safety Requirements im SyD entsprechend umgesetzt sind und ob das SyD selbst die geforderte Systemsicherheit erreicht.

Die PSSE besteht aus 7 Prozessschritten und ist einer der wichtigsten Prozesse im gesamten Projektablauf. Im ersten Prozessschritt werden die Design-Lösungen evaluiert [PSSE_BA1]. Dabei wird überprüft, ob jeder identifizierte Hazard ein Safety Goal und mindestens ein Safety Requirement besitzt und ob jedes Safety Requirement im System Design realisiert wurde. Nach positiver Bewertung, wird das SyD selbst weiteren Safety-Analysen [PSSE_BA2] unterzogen. Ziel dieser Analysen ist es, endogene Hazards sowie die gefährlichen Contributory Hazards zu identifizieren. Danach werden die Safety-Integritäts-Anforderungen spezifiziert [PSSE_BA3]. Erfüllt das Design alle Anforderung und sind keine Änderungen mehr nötig, wird das System Design validiert [PSSE_BA4]. Dabei wird überprüft, ob die Safety Goals erfüllt und alle Safety Requirements richtig umsetzt wurden. Anschließend wird die Safety-Planung fertiggestellt [PSSE_BA5]. Die Erweiterung des Safety Cases [PSSE_BA6] und der PSSE-Report [PSSE_BA7] schließen die PSSE ab Die nachfolgende Tabelle 12 gibt einen Gesamtüberblick über die Basis-Aktivitäten des PSSE Prozesses.

PSSE_BA1	System Design-Lösung evaluieren
PSSE _BA2	Safety-Analysen durchführen
PSSE _BA3	Safety Integrity Requirements spezifizieren
PSSE _BA4	System Design validieren
PSSE _BA5	Safety-Planung fertigstellen
PSSE _BA6	Safety Case weiterführen
PSSE_BA7	PSSE-Report verfassen und kommunizieren

Tabelle 12: Basis-Aktivitäten der PSSE

System Design Lösung evaluieren [PSSE_BA1]

Im ersten Prozessschritt „Design-Lösung evaluieren“ [PSSE_BA1] wird die Traceability, beginnend beim Hazard über das Safety Goal und das oder die Safety Requirements bis hin zur Design-Lösung, überprüft. Dabei wird sichergestellt, dass jedem Hazard ein Safety Goal und diesem Safety Goal ein oder mehrere Safety Requirements zugeordnet sind und dass diese eine adäquate und bewährte Design-Lösungen besitzen. Damit wird nachgewiesen, dass von den identifizierten Hazards bis zur Design-Lösung keine Lücken (*Flawed Requirements*) vorhanden sind. Ebenso wird überprüft, ob die Design-Lösung das/die Safety Requirement(s) erfüllt, ob das/die Safety Requirement(s) das Safety Goal erfüllen und ob das Safety Goal den Hazard verhindert, vermeidet oder mitigiert. Damit wird nachgewiesen, dass ein gefährlicher System Accident, der durch Hazards ausgelöst werden kann, durch ein Inherent Safe System Design verhindert wird. Die nachfolgende Abbildung 32 symbolisiert den ersten Prozessschritt.

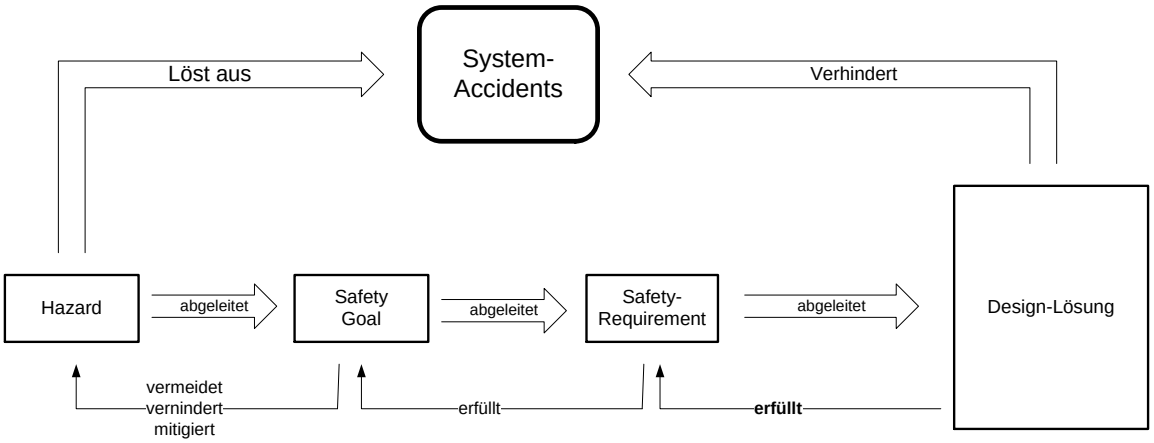


Abbildung 32: Evaluierung der System Design Lösungen

Safety-Analysen durchführen [PSSE_BA2]

Erst wenn die Traceability vom Hazard bis zur Design-Lösung sichergestellt ist, wird das SyD selbst auf Hazards analysiert. Um ein Inherent Safe System Design hinreichend spezifizieren zu können, muss das SyD ganzheitlich analysiert werden. Das heißt, es ist auch wichtig das SyD in seiner System-Umgebung und in seinem System-Umfeld zu betrachten. Dabei wird auch hier, genauso wie bei der System-Entwicklung im Prozess SyD, nach dem Systemkonzept von Ropohl [Rop_09] vorgegangen (siehe Kapitel 2.1.1). Das Safety-Vorgehensmodell fordert hier eine explorative, eine induktive und eine deduktive Analyse. Mit der induktiven Analyse-Methode wird das strukturale, mit der deduktiven das hierarchische und mit der explorativen das funktionale System-Konzept analysiert. Darüber hinaus sollen noch die gefährlichen Contributory Hazards identifiziert werden. Darauf basierend werden, wenn nötig, Derived Safety Requirements definiert.

Dieser Prozessschritt [PSSE_BA2] ist in 5 Teilschritten unterteilt:

1. Funktionales System-Konzept analysieren [PSSE_BA2.1]
2. Strukturale Sichtweise analysieren [PSSE_BA2.2]
3. Hierarchisches System-Konzept analysieren [PSSE_BA2.3]
4. Contributory Hazards identifizieren [PSSE_BA2.4]
5. Derived Safety Requirements definieren [PSSE_BA2.5]

Im ersten Teilschritt werden die noch offenen operationalisierten funktionalen System Requirements einer Safety-Analyse unterzogen [PSSE_BA2.1]. Dabei wird eine explorative Analyse-Methode angewendet.

Im zweiten Teilschritt wird das *strukturale System-Konzept analysiert* [PSSE_BA2.2]. Dabei werden die physikalischen System-Elemente des SyD auf mögliche System-Ausfälle mit einer induktiven Analyse-Methode untersucht. Damit werden system-interne Fehler, jedes analysierten System-Elements und deren Auswirkungen sichtbar.

Im dritten Teilschritt wird das *hierarchische System-Konzept analysiert* [PSSE_BA2.3]. Dabei wird ausgehend von einem gefährlichen Ereignis deduktiv zu den möglichen Fehlerursachen analysiert.

Es wird vom Hazard an der Gesamtsystemgrenze über die gesamte definierte System-Umgebung des Gesamtsystems bis zu den im SyD definierten System-Elementen zurück analysiert. Damit wird das SyD aus einem völlig anderen Blickwinkel betrachtet.

Wesentlich ist bei der Anwendung der beiden letzten Methoden, der induktiven und der deduktiven, dass gefährliche Fehlerursachen aus verschiedenen Blickwinkeln identifiziert werden können. Damit ist die Wahrscheinlichkeit hoch, dass möglichst viele Fehlerursachen identifiziert werden. Dieses Vorgehen ist auch von vielen Safety-Normen so empfohlen.

Im vierten Teilschritt wird das funktionale System-Konzept [PSSE_BA2.4] mit der „Contributory Hazard Analysis (CHA)“ untersucht. Die CHA ist das Herzstück der Analyse-Methoden im Safety-Vorgehensmodell, es dient der Identifikation von Contributory Hazards, die zu schwerwiegenden System Accidents führen können. System Accidents werden durch multi-kausale Hazards hervorgerufen. Die Federal Aviation Authority FAA [FAA7_00] bezeichnet diese Art der Hazards als „Contributory Hazards“. Aus diesem Anlass wurde die hier entwickelte Analyse-Methode auch mit „Contributory Hazard Analysis“ bezeichnet. Mit der CHA werden prognostizierte Fehlfunktionen der Safety Requirements on Element (SRoE) und deren Kombinationsmöglichkeiten, auch mit anderen identifizierten gefährlichen Fehlern, betrachtet. Die damit identifizierten gefährlichen Ergebnisse werden mit Contributory Hazards bezeichnet, da sie sich von den klassischen Hazards wesentlich unterscheiden.

Als erstes wird die *systeminterne funktionale Sichtweise* des SyD untersucht, indem die Safety Requirements on Elements (SRoE) auf mögliche Fehlfunktionen analysiert werden. Dabei wird folgendermaßen vorgegangen: Durch Negation der Verben der zugeordneten funktionalen SRoE werden Failures, also Fehlfunktionen, auf Ebene des System-Elements, prognostiziert. Danach werden die identifizierten Failures jedes betrachteten System-Elements bis auf die System-Ebene durchanalysiert. Alle diese Failures der einzelnen System-Elemente werden nun auf der nächsten System-Ebene als Errors betrachtet. Diese Errors werden aus Systemsicht weiter untersucht, ob sie in einer bestimmten Kombination zu einer Failure bzw. einem Hazards an die nächste Systemgrenze propagieren können. Das nachfolgende Beispiel, welches in Abbildung 33 dargestellt wird, soll den Sachverhalt näher beschreiben.

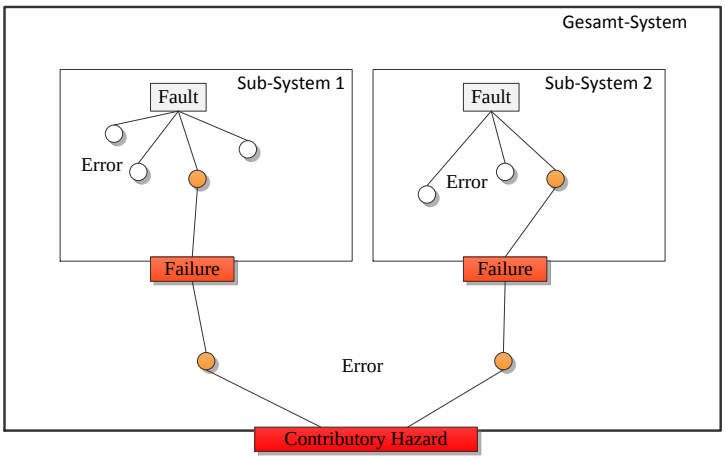


Abbildung 33: Fehlerfortpflanzung bis zum Contributory Hazard

In diesem Beispiel besteht das System aus zwei Sub-Systemen. Auf dieser Ebene werden die System-Elemente als Sub-Systeme bezeichnet, weil es sich hier um die erste untergeordnete System-Ebene handelt und die Methode auch auf tieferen Ebenen angewendet werden kann. Jedes dieser Sub-Systeme enthält nun je einen Fault. Jeder dieser Faults führt im jeweiligen Sub-System zu mehreren Errors. Einer dieser Errors, in jedem Sub-System, wird zu einer Failure an der jeweiligen Sub-Systemgrenze propagiert. Auf Gesamtsystem Ebene werden die beiden Failures wiederum als Errors betrachtet. Die beiden Errors auf der System-Ebene treffen nun zu einem bestimmten Zeitpunkt, also quasi-zufällig (siehe dazu 4.1.3 „Fehlerklassen“), zusammen und bewirken auf Gesamtsystem Ebene eine Failure. Ist diese Failure gefährlich, so wird sie als Contributory-Hazard bezeichnet.

Als nächstes werden nun auch noch Hardware-Fehler mitbetrachtet. Dafür werden bereits identifizierten Hardware-Failure aus der induktiven Analyse, die auf System-Ebene als Hardware-Error wirken, herangezogen. Das heißt, es werden Hardware-Errors mit den Software-Error in den verschiedenen Kombinationsmöglichkeiten untersucht und deren Auswirkungen auf System-Ebene betrachtet. Auch hier werden die gefährlichen Kombinationen als Contributory-Hazards gezählt.

Darauffolgenden werden die gefährlichen Einflüsse aus der System-Umgebung und dem System-Umfeld noch mitanalysiert. Das sind beispielsweise fehlerhafte benachbarte Systeme, menschliche Fehlhandlungen oder Witterungseinflüsse. Hier werden die Analyse-Ergebnisse aus der PHI herangezogen, siehe dazu Kapitel 3.1.3 „Preliminary Hazard Identification (PHI)“, Teilschritt zwei [PHI_BA2]. Und auch hier werden die gefährlichen Kombinationen als Contributory-Hazards gezählt. Diese Bezeichnung der Contributory-Hazards ist hier notwendig damit sie von den klassischen Hazards, also den gefährlichen Einzel-Fehlern, unterschieden werden können.

Im letzten Teilschritt [PSSE_BA2.5] werden für alle identifizierten Hazards aus Teilschritt [PSSE_BA2.1], [PSSE_BA2.2] und [PSSE_BA2.3] aber auch für die Contributory-Hazards aus Teilschritt [PSSE_BA2.4] nach geeigneten Lösungen gesucht. Aber auch hier müssen diese von Safety-Goals und Safety-Requirements abgedeckt sein. Diese spezielle Basis-Aktivität kann dabei mehrere Iterationen im Modellierungsraum auslösen.

Safety Integrity Requirements spezifizieren [PSSE_BA3]

Im diesem Prozessschritt werden anhand der definierten Sicherheitsanforderungsstufe die von der Safety-Norm geforderten Safety Integrity Requirements festgelegt. Diese spezifischen Requirements sind den anzuwendenden Safety-Normen zu entnehmen und können von Sicherheitsanforderungs-stufe zu Sicherheitsanforderungsstufe unterschiedlich sein. Prinzipiell steigt der Anforderungsgrad mit zunehmender Sicherheitsanforderungsstufe. Die geforderten Safety Integrity Requirements sind beispielsweise anzuwendende Software Architektur-Methoden, Review-Techniken oder Test-Methoden. Bei höheren Sicherheitsanforderungsstufen wird beispielsweise die Graceful Degradation zum Error-Handling bei der Entwicklung der Software Architektur gefordert, bei Review-Techniken sind es Inspections und bei den Test-Methoden Fault Injection Tests und/oder Back-to-Back Tests. Die genaue Ausprägung dieser Forderungen ist festzulegen, da diese wesentliche Auswirkungen auf den Leistungsaufwand und auf die geforderten Kompetenzen der Projektteammitglieder haben kann.

System Design validieren [PSSE_BA4]

Im diesem Prozessschritt wird das System Design validiert, dabei wird das SyD durch ein Exptерten-Team, nach Möglichkeit auch mit dem Kunden und Zertifizierern, statisch überprüft, ob die Safety Goals angemessen sind und ob im SyD alle Safety Requirements korrekt umgesetzt wurden. Mit der Validierung des System Designs soll der Beweis erbracht werden, dass das Design Inherent Safe ist.

Safety-Planung fertigstellen [PSSE_BA5]

Sind alle Prozessschritte positiv abgeschlossen und keine weiteren Iterationen mehr notwendig, wird die Safety-Planung fertiggestellt. Dabei müssen die erforderlichen Safety-Assessments, Safety-Audits sowie die Safety-Zertifizierung mit Vor- und Nachbereitung mitberücksichtigt werden. Die Verifikationen der Safety-Requirements und die Validationen der Safety-Goals müssen in die System-Test bzw. Acceptance-Test Aktivitäten integriert werden. Die Safety-Planung ist, je nach Projektgröße, in die Projektplanung zu integrieren oder mit dieser zu synchronisieren, wenn sie als eigene Planung geführt wird.

Safety Case weiterführen [PSSE_BA6]

Basierend auf der in der FHE festgelegten Safety Case Struktur werden die Ergebnisse aus der PSSE in die vordefinierte Struktur als Evidenzen eingepflegt. Das sind vorwiegend die Ergebnisse aus den Safety-Analysen und die Dokumente die aufgrund der Ergebnisse geändert werden mussten.

PSSE-Report verfassen und kommunizieren [PSSE_BA7]

Die Ergebnisse der PSSE werden anhand des PSSE-Reports an das Projektmanagement sowie an alle betroffenen Stakeholder berichtet. Dieser Report soll auch sicherstellen, dass alle noch notwendigen Leistungen in der Planung berücksichtigt werden.

3.2.6 Zusammenfassung des Modellierungsraums

Die nachfolgende Abbildung 34 fasst den Ablauf im Modellierungsraum nochmals zusammenfassen. Der Modellierungsraum beginnt mit dem Projektstart-Workshop, in dem alle am Projekt beteiligten Personen auf den gleichen Wissensstand gebracht werden. Im Zuge der Projekt Stakeholder Analyse werden alle Einflussfaktoren, die auf das Projekt einwirken können, identifiziert, um frühzeitige Problemfelder, aber auch Potentiale erkennen zu können. Darauf basierend werden im Requirements Engineering die Anforderungen an das zu entwickelnde System spezifiziert. Das Ergebnis ist die System-Requiements Spezifikation, die in weiterer Folge einer Safety-Analyse in der FHE unterzogen wird. Die System Requirements-Spezifikation und der FHE-Report sind wichtige Outputs für die Detail-Planung. Wurden in der Safety-Analyse weitere Safety Requirements identifiziert, müssen diese wiederum in die System Requirements-Spezifikation eingesteuert werden und es folgt eine weitere Iteration.

Wenn die System Requirements einen definierten Reifegrad besitzen, also wenn sie evaluiert und freigegeben sind, kann darauf basierend das SyD entwickelt werden. Das SyD wird in der PSSE weiteren Safety-Analysen unterzogen. Die PSSE ist der wichtigste Prozess im gesamten

Projektablauf, hier werden die Weichen für ein Inherent Safe System Design gestellt. Das heißt, es muss anhand des Traceability-Checks nachgewiesen werden, dass keine Requirements-Lücke mehr identifiziert wurde und dass für jedes geforderte System Requirement eine Lösung im SyD existiert. Werden in der Preliminary System Safety Evaluation (PSSE) noch weitere Derived Safety Requirements identifiziert, so müssen diese wiederum in die System Requirements Spezifikation eingearbeitet werden. Handelt es sich um ein funktionales Requirement, so muss dieses in der FHE einer Safety-Analyse unterzogen werden und danach im SyD entsprechend realisiert werden.

Ist das System Design freigegeben, kann darauf basierend die Planung im Projekthandbuch fertiggestellt werden. Damit werden auch die Kriterien zur Freigabe für den weiteren Projektverlauf erfüllt. Alle Output-Dokumente werden im Zuge des Konfigurationsmanagements eingefroren (baselining) und in der Projekt-Datenbank abgelegt. Alle nun geforderten Änderungen und Erweiterungen im weiteren Projektverlauf werden nur mehr kontrolliert über das Change Control Board durchgeführt.

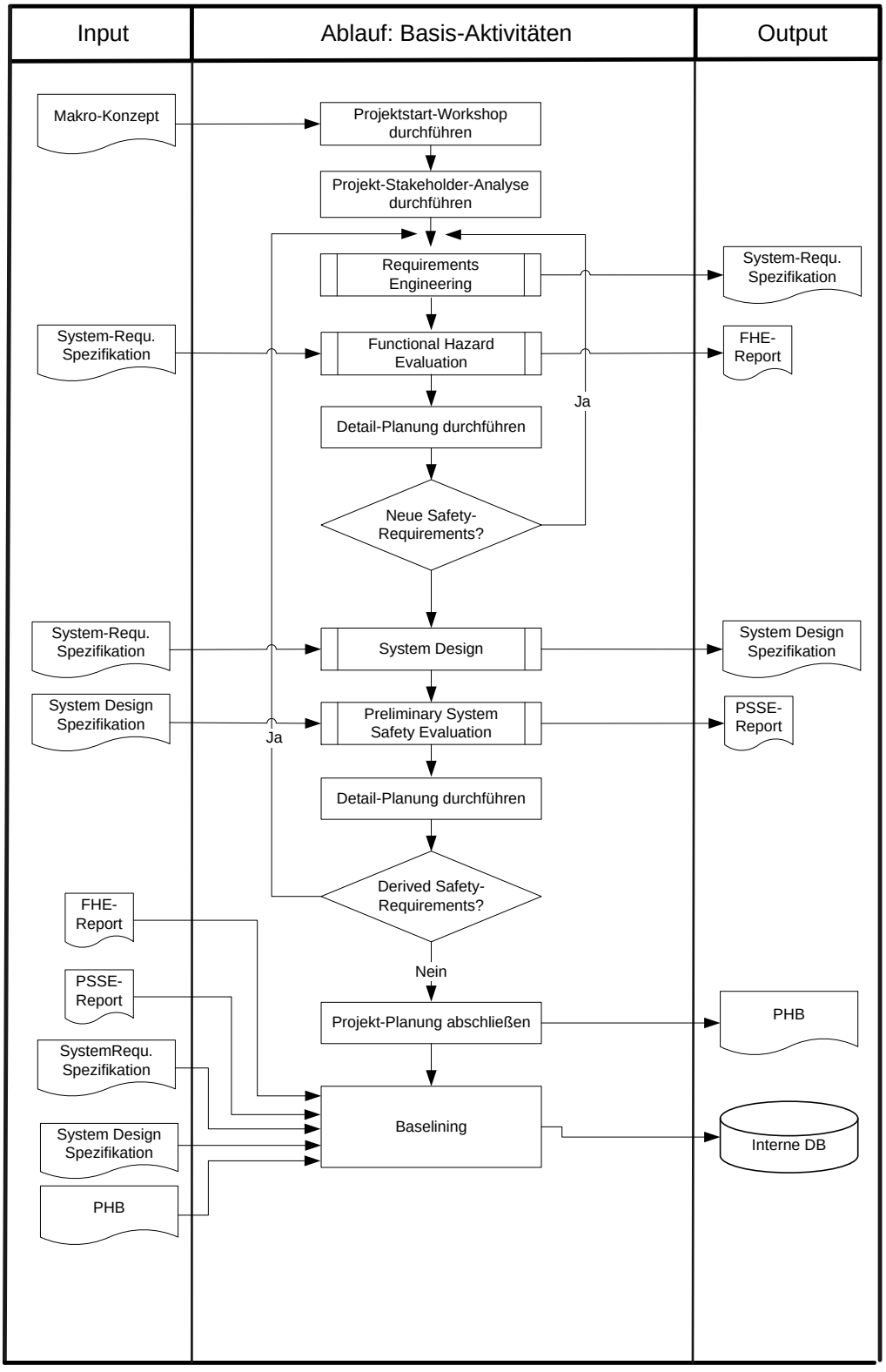


Abbildung 34: Ablauf des Modellierungsraums als Flow-Chart

4. Neue Modelle und Methoden

Im Mittelpunkt dieser Arbeit steht das Safety-Vorgehensmodell, welches auf klassischen, erweiterten, aber auch auf neu entwickelten Modellen und Methoden aufbaut. Diejenigen Modelle und Methoden, die für diese Arbeit erweitert bzw. neu entwickelt wurden, werden in diesem Kapitel beschrieben. Dies ist das Modell der erweiterten Fehlerkette, die den Wirkmechanismus von einer Grundursache bis zum Unfall durchgängig darstellt. Ein zweites, für diese Arbeit wesentliches Modell, ist das „Shell-Modell“. Es verschafft einen ganzheitlichen System-Überblick, der es erlaubt Safety-Analysen über verschiedenen Systemgrenzen untersuchen zu können. Erweiterte und neue Analyse-Methoden sowie die Methoden des Risiko-Assessments und das Safety-Case Pattern schließen dieses Kapitel ab.

4.1 Umfassende Fehlerkette und Fehlerklassen

Im Folgenden wird auf die bekannte Fehlerkette von Avizienis und Laprie [AvLa_04] eingegangen, die dann in beide Richtungen weiter entwickelt wird, sodass sie schließlich eine vollständige Kausalkette von der Grundursache bis zur letzten für diese Arbeit relevanten Auswirkung bildet. Diese erweiterte Fehlerkette bildet dann die Basis für weitere Überlegungen in dieser Arbeit.

4.1.1 Erweiterung der Fehlerkette

Die in Kapitel 2.1.2 dargestellte Fehlerkette aus [AvLa_04] beschreibt die Fehlerwirkmechanismen in einem schon bestehenden, sich im Betrieb befindlichen System. Das heißt, diese Fehlerkette ist ein Modell, welches nur für einen bestimmten Teil des Lifecycles anwendbar ist, über den es nicht hinausgeht. Die Ursachen für Fehler in einem System, d.h. die Mechanismen der Entstehung, sind nicht im Modell enthalten. Auch die für diese Arbeit wesentlichen potentiellen Auswirkungen der Fehler, beispielsweise ob ein Fehler einen Unfall auslösen kann oder nicht, sind nicht in diesem Modell enthalten.

Es erscheint jedoch aus mehreren Gründen sinnvoll, auch diese Teile in die Fehlerkette zu integrieren. Ein Grund ist, dass eine solche erweiterte Fehlerkette ein verbessertes Verständnis der Zusammenhänge liefert. Dies wiederum ermöglicht es, gezielte Maßnahmen gegen die Auswirkungen der Fehler zu treffen. Ein weiterer Grund ist, dass innerhalb der erweiterten Fehlerkette eine einheitliche Terminologie definiert wird, die Missverständnisse zwischen Begriffen

zu reduzieren hilft. Der wichtigste Grund für die Erweiterung der Fehlerkette ist jedoch die ganzheitliche Betrachtung der Fehlerwirkmechanismen, beginnend mit den „Root Causes“ (Grundursachen) bis zur Auswirkung, die tatsächlich Mensch und/oder Umwelt gefährdet, also aus Safety-Sicht relevant ist. Diese Grundursachen liegen zumeist in der Entwicklung eines Systems, und hier vor allem an mangelhaften, fehlerhaften oder schlicht fehlenden Requirements. Dafür gibt es mehrere Belege, die nachfolgend angeführt werden.

Lutz [Lutz_01] untersuchte die Root-Causes von safety-relevanten Software-Errors in safety-kritischen Embedded Systems, welche bei der Integration und beim System-Test, der Raumfahrt-Software Voyager und Galileo, aufgezeigt wurden. Diese empirische Studie hat gezeigt, dass Fehler im sicherheitskritischen Bereich in diesem Kontext hauptsächlich aus folgenden Gründen entstehen: „(1) discrepancies between the documented requirements specifications and the requirements needed for correct functioning of the system and (2) misunderstandings of the software‘s interface with the rest of the system.“[59]

Analysen wie diese sind natürlich mit einer gewissen Skepsis zu betrachten, da der Anwendungsbereich möglicherweise nicht breit genug ist und die Ergebnisse nur für eine sehr limitierte Systemklasse gelten. Das Faktum, dass Mängel in Requirements für einen großen Teil der schwerwiegenden Fehler in einem System verantwortlich sind, ist jedoch in zahlreichen Studien nachgewiesen und kann deshalb als gesichert angenommen werden. Das gilt vor allem für softwarelastige Systeme: Leffingwell und Widrig [Leff_03, Seite 7] sprechen von „accumulating evidence that many of the most common, most serious problems associated with software development are related to requirements“[60] und ziehen für diese Schlussfolgerung zahlreiche Studien heran.

Requirements sind eine abstrakte Beschreibung eines Systems, die jedoch von allen „Stakeholdern“ verstanden werden müssen. Sie dienen damit als zentrales Kommunikationstool. Mängel in Requirements sind damit durch Mängel in dieser Kommunikation begründet, was wiederum mit mangelhaften Prozessen zusammenhängt bzw. generell auf Managementprobleme hinweist. Diese Überlegungen sind die Basis für die Erweiterung der Fehlerkette im folgenden Kapitel.

Nakajo und Kume [NaKu_91] haben schon 1991 eine partielle Erweiterung der Fehlerkette in diesem Sinne vorgeschlagen. Deren Fehlerkette beginnt mit einem „Inappropriate Management of Work System“, welches (unter anderem) zu „Process Flaws“ führt, die wiederum Human Errors begünstigen. Auch [NaKu_91] haben eine empirische Untersuchung durchgeführt, die zu sehr ähnlichen Ergebnissen kommt wie Lutz [Lutz_01]: 56,9% der System Failures wurden durch Interface Faults verursacht, 33,3% durch Function Faults und nur 9,8% durch Internal Faults. Die hier beschriebenen Interface und Function Faults sind letztlich auch durch Probleme in Requirements bzw. durch mangelnde Kommunikation bedingt. Lediglich die „Internal Faults“

[59] "(1) Diskrepanzen zwischen den dokumentierten Anforderungsspezifikationen und den Anforderungen, die tatsächlich für das korrekte Funktionieren benötigt werden, und (2) Missverständnisse des Software-Interfaces mit dem Rest des Systems. (Übersetzung des Verfassers)

[60] "Zunehmende Hinweise, dass viele der häufigsten, schwerwiegendsten Probleme in der Softwareentwicklung mit den Anforderungen zu tun haben" (Übersetzung des Verfassers)

können nicht auf Probleme in den Requirements zurückgeführt werden. Eine Schlussfolgerung von Nakajo und Kume [NaKu_91] ist, dass diese menschlichen Fehler letztlich auf mangelhaftes Management zurückgeführt werden können:

> "A human error occurring during software development is first detected as 'system failure'; however, the 'system failures' vary widely. Tracing back to the source of a human error reveals that 'inappropriate work system management' was ultimately responsible for the error, although the details surrounding this state vary and are strongly dependent on the organization's characteristics."[61]

4.1.2 Erweiterte Fehlerkette

Die Fehlerkette von Laprie [AvLa_04] (siehe Kapitel 2.1.2) erfährt in dieser Arbeit eine beidseitige Erweiterung, um einerseits die Ursachen für Fehler genauer zu identifizieren und andererseits die (potentiell sicherheitskritischen) Auswirkungen auf die Umgebung im Detail zu beschreiben. Diese Erweiterung der Fehlerkette basiert teilweise auf den Erkenntnissen von Nakajo und Kume [NaKu_91], wurde aber noch weiter vertieft.

Prinzipiell kann man diese vollständige Fehlerkette – wie in Abbildung 35 dargestellt – in drei Bereiche unterteilen:

1. Der Bereich der Fehlerursachen („Root Causes"), also die kausalen Faktoren, warum überhaupt Fehler in einem System vorhanden sind bzw. wie sie in das System gelangen. Diese Ursachen liegen in der Umgebung, in der ein System entwickelt bzw. gefertigt wird („Development Environment").
2. Der Bereich der Fehler im System selbst. Dieser Bereich ist mit der Fehlerkette aus [AvLa_04] gut beschrieben.
3. Die Auswirkungen der Fehler auf die „World", in der das System eingesetzt wird, also die operationelle Umgebung (Operational System Environment), sowie fehlerverursachende Einwirkungen auf das System von dieser.

[61] "Ein menschlicher Fehler während der Softwareentwicklung wird zuerst als 'Systemfehler' entdeckt. Die 'Systemfehler' variieren jedoch stark. Wenn man jedoch die Ursache des menschlichen Fehlers analysiert, stellt sich oft heraus, dass ein 'ungeeignetes Arbeits-Management' letztlich verantwortlich für den Fehler war, wenn auch die Details dieses Zustands je nach Organisation stark variieren und von den Charakteristiken der Organisation abhängen." (Übersetzung des Verfassers)

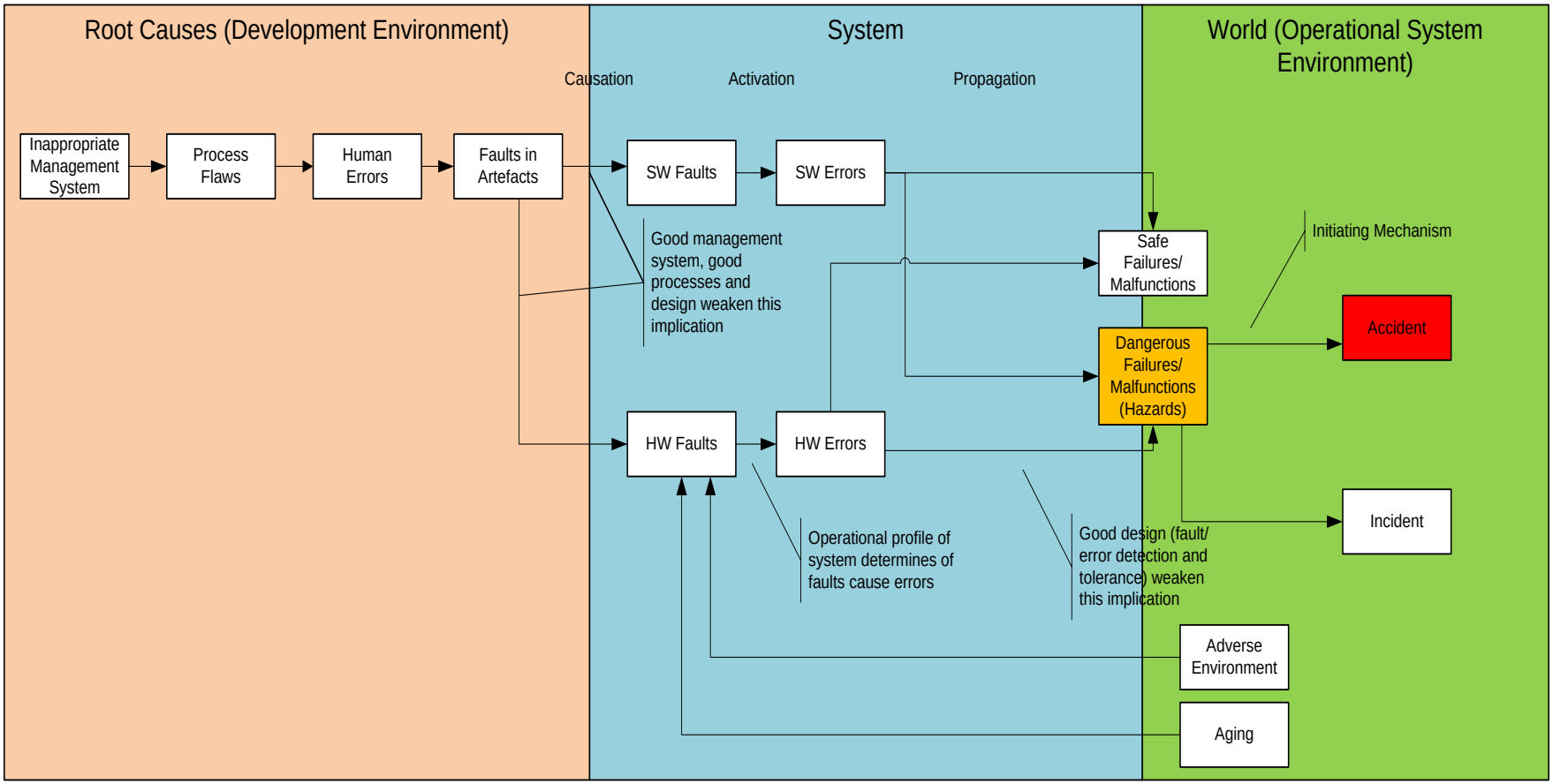


Abbildung 35: Erweiterte Fehlerkette für elektronische Systeme

Im Folgenden werden die drei Bereiche „Root Causes“, „System“ und „World“ genauer beschrieben.

Root Causes: Der Bereich der Fehlerursachen kann folgendermaßen beschrieben werden: Ein schlechtes Managementsystem führt zu schlechten, mangelhaften oder fehlenden Prozessen, also allgemein zu „process flaws“. Schlechte Prozesse begünstigen Fehler und Fehlentscheidungen von Menschen („human errors“), da beispielsweise nicht sichergestellt ist, dass alle erforderlichen Informationen rechtzeitig bzw. zum richtigen Zeitpunkt einwandfrei verfügbar sind. Von weiterer Bedeutung ist dabei auch ein einheitliches Vorgehen beziehungsweise eine einheitliche Arbeitsweise. Wenn diese nicht gegeben ist, entstehen Fehler. Menschliche Fehler sind prinzipiell unvermeidbar, und selbst mit ausgereiften Prozessen und bei Routineaufgaben begehen Menschen Fehler. Sollte es jedoch keine definierten Prozesse geben und die Aufgaben damit nicht zur Routine gehören, ist die Fehlerrate für menschliche Fehler wesentlich höher. Smith [Smi_00] beinhaltet in Appendix 6 beispielsweise eine ausführliche Tabelle, in der die Fehlerrate für „non-routine tasks“ um eine Größenordnung größer ist. Menschliche Fehler führen nun zu Fehlern in den diversen Arbeitsprodukten (Dokumente, Zwischenprodukte, Fehler im Hardware-Layout etc.), die im Zuge der Entwicklung produziert werden. Diese Fehler in den Arbeitsprodukten können nun wiederum durch reife Prozesse (wie beispielsweise Dokumenten- und Code-Reviews, Inspektionen) entdeckt werden, bevor sie sich als Fehler im System, entweder in der Hardware oder in der Software, manifestieren.

System: Dieser Teil der Fehlerkette entspricht der Beschreibung in Avinzienis und Laprie [AvLa_04]. In der erweiterten Fehlerkette wurde jedoch noch zwischen Software- und Hardwarefehlern unterschieden. Ob ein Fehler in einen Fehlzustand mündet, hängt von der Benutzung des Systems ab, also vom „operational profile“. Genauso hängt es vom operational profile ab, ob ein Fehlzustand zu einem Ausfall oder einer Fehlfunktion (failure oder malfunction) führt. Durch Error-Detection Mechanismen können beispielsweise Fehlzustände erkannt werden, und das System kann einem Ausfall oder einer Fehlfunktion noch entgegenwirken (beispielsweise durch Umschalten auf eine redundante Komponente als Reaktion auf einen Fehlzustand in einer Komponente). Ausfälle oder Fehlfunktionen können nun sicherheitskritisch („dangerous“) oder nicht sicherheitskritisch („safe“) sein. Ein „dangerous failure“ wird typischerweise als Hazard bezeichnet.

World: In der operationellen Umgebung können nun Umwelteinflüsse wie Temperatur, Erschütterungen etc. oder auch einfach Alterung zu Hardwarefehlern führen. Softwarefehler hingegen entstehen nicht durch die operationelle Umgebung, sondern sind schon von vornherein in dem System enthalten – dieser Unterschied ist wesentlich für das Verständnis von zufälligen beziehungsweis systematischen Fehlern. In der operationellen Umgebung kann ein Hazard nun – je nach Umgebungsbedingungen bzw. „Initiating Mechanism“ – zu einem Vorfall (Incident) oder Unfall (Accident) führen. Der Unfall ist das Ereignis, bei dem tatsächlich Schaden an Mensch und/oder Umwelt entsteht, und welches es letztlich zu vermeiden gilt.

Diese neue Darstellung der Fehlerkette hilft nun, die Wirkmechanismen und Kausalbeziehungen zu illustrieren, die zu Unfällen führen, und Gegenmaßnahmen gezielt zu planen. Außerdem definiert diese erweiterte Fehlerkette die in dieser Arbeit verwendete Terminologie und deren Zusammenhänge. Prinzipiell gibt es zwei Möglichkeiten, die Fortpflanzung der Fehler innerhalb der

Fehlerkette zu unterbrechen: Einerseits kann man bei den einzelnen Gliedern der Fehlerkette ansetzen und diese eliminieren bzw. reduzieren, beispielsweise durch reife Prozesse die „process flaws“ reduzieren oder man „schwächt“ die Implikationspfeile zwischen den Gliedern, zum Beispiel indem man durch „Error Detection“-Mechanismen verhindert, dass ein Error zu einem Failure/Malfunction wird.

4.1.3 Fehlerklassen

Die Fehlerklassen „zufällige Fehler“ und „systematische Fehler“ sind für das Verständnis der erweiterten Fehlerkette und der Problematik der System Accidents notwendig, deshalb sollen auch diese kurz beschrieben werden, um das Gesamtbild abzurunden.

Zufällige Fehler sind solche, deren Auftreten nur mittels einer Wahrscheinlichkeitsdichteverteilung vorausgesagt werden können und die während des Betriebes entstehen. Diese Fehler sind ausschließlich in der Hardware zu finden – sie werden durch nicht vorhersehbare Umwelteinflüsse bzw. Alterungserscheinungen verursacht. Die zugehörigen zufälligen Ausfälle sind dann solche Ausfälle, die als Resultat von Zufallsfehlern entstehen.

Systematische Fehler sind solche, die schon während der Entwicklung oder Fertigung eines Systems in das System einfließen (wie beispielsweise Designfehler oder Programmierfehler) und somit während des Betriebes schon von Beginn an vorhanden sind. Diese Fehler können dann über die oben beschriebene Fehlerkette aktiviert werden und letztlich zu Ausfällen propagieren. Die daraus resultierenden Ausfälle sind somit systematische Ausfälle.

In komplexen Systemen ist auch der Wirkmechanismus von systematischen Fehlern oft sehr komplex. Der Ausfall tritt beispielsweise nur dann auf, wenn zeitgleich mehrere Bedingungen gegeben sind, deren spezifische Kombination aber sehr selten ist. Solche systematischen Fehler sind somit theoretisch vorhersehbar, da sie immer unter diesen spezifischen Bedingungen auftreten. Sind diese seltenen, gleichzeitig eintretenden Bedingungen jedoch nicht bekannt (was durch die Komplexität und die Seltenheit der einzelnen Bedingungen leicht der Fall sein kann), so *scheint* ein solcher Ausfall rein zufällig zustande zu kommen. Ausfälle dieser Art sind somit theoretisch vorhersehbar, in der Praxis jedoch von vollkommen zufälligen Ausfällen nicht unterscheidbar. In diesem Fall kann man also von *„quasizufälligen“ Ausfällen* sprechen. Als Beispiel sei ein Systemabsturz eines komplexen Betriebssystems genannt, in dem ein Programmierfehler (fault) vorliegt. Wann und unter welchen Umständen dieser sich aber so auswirkt, dass das System abstürzt, ist nicht vorhersehbar. Der Benutzer nimmt nur wahr, dass das System hin und wieder abstürzt. Für den Benutzer erscheint der Ausfall also zufällig, obwohl ein systematischer Fehler dahintersteckt.

Darüber hinaus ist zu beachten, dass der Zusammenhang zwischen Fehlern und Ausfällen kein 1:1, sondern ein m:n Zusammenhang ist. Ein Fehler kann also keinen, einen oder mehrere Ausfälle hervorrufen, während ein Ausfall durch einen oder auch durch mehrere Fehler verursacht werden kann. Wie in Holzmann [Hol_07] und Leveson [Lev_00] beschrieben, sind die aus Safety-Sicht besonders gefährlichen Ausfälle jene, die von mehreren Fehlern hervorgerufen werden, deren kombiniertes Auftreten schlicht nicht vorhergesehen wurde. Ein Fehler in diesem Sinne ist nicht nur

die Nichterfüllung eines vorhandenen Requirements, sondern auch Fehler in Requirements (fehlerhafte, fehlende, widersprüchliche… Requirements) selbst.

4.2 Shell-Modell

Um das zu entwickelnde System in seinem Kontext besser erfassen zu können, wurde im Zuge dieser Arbeit eine anschauliche graphisch basierte Darstellungs- und Analyse-Methode entwickelt. Diese Methode wird als „*Shell-Modell*“ bezeichnet. Es dient der *systematischen* Identifikation aller im Kontext stehenden Elemente und Eigenschaften, die auf das zu entwickelnde System einwirken können, die aber auch durch dieses beeinflusst werden können. Das zu entwickelnde System wird im Shell-Modell als System Under Consideration (SUC) bezeichnet. In der Analyse wird das SUC, mit seiner gesamten System-Umgebung und seinem System-Umfeld in sogenannte Schalen unterteilt, die als „Shells“ bezeichnet werden. Jede einzelne Shell wird mit den darin befindlichen Elementen vom Analyse-Team einer systematischen Analyse unterzogen. Mit dem Shell-Modell können die funktionalen sowie die nicht-funktionalen Requirements, die an das System gestellt werden, identifiziert werden. Des Weiteren können darauf basierend exogene Hazards[62] frühzeitig erkannt werden. Einwirkungen von außen können sich auf das Systemverhalten beispielsweise so auswirken, dass dadurch ein Fehler initiiert wird und dieser, wie in Kapitel 2.1.4 „Hazard Triangle“ nach Ericson [Eric2_05] beschrieben, einen System-Hazard verursacht, welcher wiederum zu einem Unfall führen kann. Eine Systematik stellt sicher, dass die Vorgangsweise wiederholbar ist.

Der große Vorteil am Shell-Modell ist auch, dass der Diskussionsprozess im Analyseteam – im Gegensatz zu herkömmlichen Methoden – aufgrund der sequentiellen Diskussion der einzelnen Shells sehr geordnet abläuft. Alle Diskussionsteilnehmer haben die gleiche ganzheitliche Sicht des SUC vor Augen und die Analyse gestaltet sich durch diese Systematisierung sehr effizient.

Das innovative Element im Shell-Modell ist, dass im Gegensatz zu herkömmlichen Analyse-Methoden wie z.B. dem Brainstorming oder dem „Treffen von Annahmen“, nicht nur das SUC, sondern das SUC in seiner gesamten Systemumgebung systematisch betrachtet und analysiert wird. Damit kann, wie in Kapitel 2.1.1 „Grundbegriffe der technischen Systemtheorie“ beschrieben, das hierarchische Systemkonzept des zu entwickelnden Systems, über alle Systemgrenzen hinweg, bis zur Systemumgebung und seinem Einsatzbereich betrachtet werden.

Das Shell-Modell bietet auch die Möglichkeit, das funktionale Systemkonzept mit seinen System-Funktionen sowie das strukturale Systemkonzept mit seinen Sub-Systemen und deren internen und externen Schnittstellen herzuleiten und sichtbar zu machen.

Um das Verständnis zu fördern, wird in den nachfolgenden Unterkapiteln der Aufbau des Shell-Modell von außen nach innen beschrieben und danach die Anwendung, die jedoch von innen nach außen verläuft. Diese Darstellung wurde bewusst so gewählt, damit ein gewisses Grundverständnis

[62] Exogene Hazards sind Gefährdungen, die das System durch äußere Ursachen (z.B. Witterung, andere Systeme) so beeinflussen, dass das System an seiner definierten Systemgrenze eine Gefährdung darstellt. Das Gegenteil sind endogene Hazards, die aus systeminneren Ursachen entstehen und das System in einen gefährlichen Zustand bringen.

über das Modell aufgebaut werden kann, bevor auf die etwas intensivere Funktionsweise des Modells eingegangen wird.

4.2.1 Aufbau des Shell-Modells

Das Shell-Modell kann, je nach Anwendung, aus mehreren Shells bestehen, minimal sind jedoch vier notwendig. In diesem Sub-Kapitel werden das Prinzip des Modells sowie die wichtigsten Shells genau beschrieben. Die nachfolgende Abbildung 36 zeigt den grundlegenden Aufbau, also das Pattern[63] des Shell-Modells.

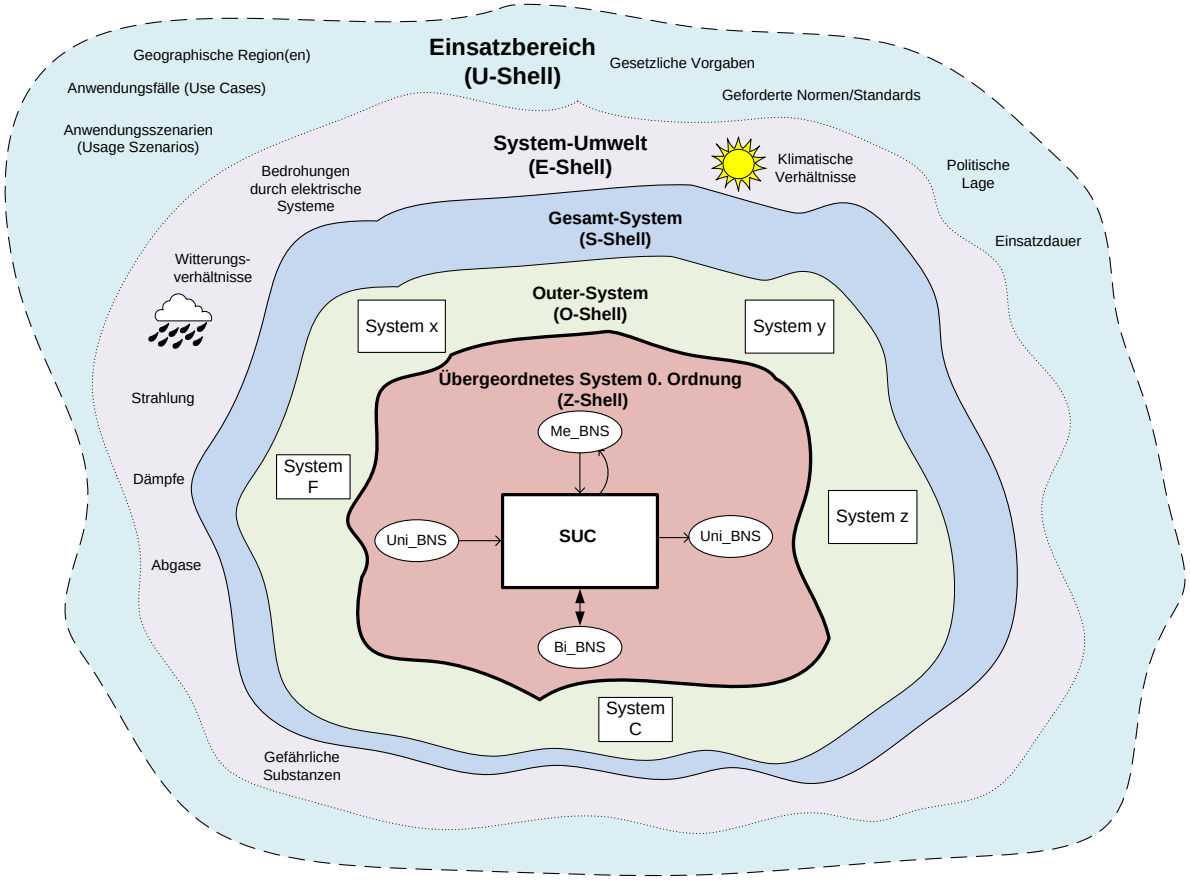


Abbildung 36: Shell-Modell

Die äußerste Shell definiert den *Einsatzbereich* des Systems und wird mit Usage-Shell (U-Shell) bezeichnet. Sie enthält Einflussfaktoren, die durch den Einsatzbereich und die Einsatzdauer des SUC bedingt sind. Das sind Rahmenbedingungen, die das SUC einhalten bzw. berücksichtigen muss. Geforderte Normen und Standards können hier identifiziert werden oder gesetzliche Vorgaben, wie es auch die ISO 26262 [ISO26_11] im Teil 3, Kapitel 3.4.1.c „legal requirements (especially laws

[63] Unter der Bezeichnung „Pattern“ wird in dieser Arbeit eine Vorlage verstanden, die in einem bestimmten Zusammenhang immer wieder zur Problemlösung angewendet werden kann. Es muss jedoch immer an die jeweilige Situation angepasst werden.

and regulations), national and international standards", fordert. Das können beispielsweise Umweltauflagen oder Prozessreifegradmodelle sein, wie beispielsweise das in den USA geforderte CMMI- oder in Europa das SPICE-Modell. Auch die politische Lage[64] kann großen Einfluss auf die Systementwicklung nehmen. In der U-Shell werden die verschiedenen Anwendungsszenarien (Usage Scenarios), und Use Cases definiert.

Die darunter liegende Shell beschreibt die *System-Umwelt*, als Environmental-Shell (E-Shell) bezeichnet. Damit sind die Umweltfaktoren sowie deren Einflüsse auf das SUC gemeint. Dabei werden beispielsweise die Witterung und klimatischen Verhältnisse sowie physikalische oder chemische Einwirkungen identifiziert, wie beispielsweise Vibration, Strahlung und Gase sowie gefährliche Substanzen oder Komponenten im System-Umfeld.

In der nächsten Shell wird das definierte *Gesamtsystem* beschrieben, in der Darstellung wird die Gesamtsystemgrenze mit System Border Shell (S-Shell) bezeichnet. Diese S-Shell beschreibt das Interface zur Umwelt (E-Shell). Dabei kann das SUC auch das Gesamtsystem sein. Dies ist der Fall, wenn vom Projektvorhaben keine übergeordneten Systeme definiert sind. In diesem Fall stellt sich natürlich die Frage, ob die Anwendung des Shell-Modells noch sinnvoll ist.

Das SUC kann aber auch hierarchisch in mehrere übergeordnete Systeme eingebettet sein. In diesem Fall wird die übergeordnete Shell beziehungsweise als O-Shell (Outer Shell) bezeichnet. Es ist zu analysieren, in welcher Ebene gegenüber dem Gesamtsystem, das SUC integriert werden soll. Das heißt, das SUC kann beispielsweise Teil eines übergeordneten Systems sein, welches wiederum Teil eines Supersystems ist. Um die übergeordneten Systeme einordnen zu können, wird jeder Hierarchie-Stufe zwischen Gesamtsystem und System eine Ordnungszahl (O_1 … O_n, $n \geq 1$) zugeordnet. Die innerste Shell wird als Zero-Shell, kurz Z-Shell, bezeichnet. Sie beschreibt die unmittelbare Umgebung des SUC. Die nachfolgende Tabelle 13 gibt einen Gesamtüberblick mit kurzer Beschreibung der einzelnen Shells.

Es soll an dieser Stelle noch festgehalten werden, dass alles was sich außerhalb des Gesamtsystems befindet, also die E-Shell und die U-Shell, als *System-Umfeld* bezeichnet wird. Alles was innerhalb des Gesamtsystems dargestellt wird, also die Outer-Shells (> O_1), wird als *System-Umgebung* bezeichnet wird

[64] Das tolerierbare Risiko wird auf gesellschaftlicher Basis bestimmt und berücksichtigt gesellschaftliche und politische Faktoren. [IEC08_10, Teil 5, Kap. A5]

Shell	Beschreibung
SUC	In der innersten Shell befindet sich das System Under Consideration (SUC), also das zu entwickelnde System.
Z-Shell	Die Zero-Shell (Z-Shell) stellt die unmittelbare Systemumgebung des SUC dar. Darin sind das SUC und dessen unmittelbar benachbarten Systeme enthalten, die mit dem SUC direkt kommunizieren. Das heißt, hier bestehen eine oder mehrere physikalische Schnittstellen vom SUC zu den benachbarten Systemen (BNS). Die BNSe können sein: - Menschliche BNSe (Me_BNS) bzw. Bedienelemente oder Repräsentanten - Uni-direktionale BNSe (Uni_BNS) und - Bi-direktionale BNSe (Bi_BNS) Die Z-Shell ist damit die übergeordnete Shell 0.Ordnung, also quasi die O_0-Shell.
O-Shell	Die Outer-Shells der Ordnung 1 oder darüber, stellen die mittelbare Umgebung des SUC dar. In den O-Shells > 0 sind übergeordnete Systeme dargestellt, in denen das SUC zum Einsatz kommt. Diese Shells können eine beliebige Anzahl sein ($O_1 \dots O_n$, $n \geq 1$). Anmerkung: O_0 = Z-Shell Die Systeme in den jeweiligen O-Shells sind mit dem SUC nicht direkt verbunden, können auf dieses jedoch indirekt einwirken. Ist keine O-Shell vorhanden, dann ist das SUC das Gesamtsystem (O_n = S) Es gibt somit immer mindestens eine O-Shell, in diesem Fall die S-Shell.
S-Shell	Die System Border Shell (S-Shell) betrachtet das Gesamtsystem oder auch das Supersystem, in dem das SUC eingebettet ist. Das Gesamtsystem kann aus mehreren Sub-Systemen bestehen. Die S-Shell ist praktisch die äußerste O-Shell und bildet das Interface zum Environment, also zur E-Shell.
E-Shell	Die Shell der System-Umwelten, also die Environmental-Shell (E-Shell), identifiziert die Umweltbedingungen des jeweiligen Einsatzbereiches, der auf das SUC Einfluss nimmt. Das sind beispielweise: • Klima und Wetter • Chemische Einwirkungen (Gas, Dunst, Rauch, …) • Physikalische Einwirkungen (Geschwindigkeit, Erschütterung, ...) • Elektromagnetische Strahlung • Kosmische Strahlung • etc.
U-Shell	Die Usage-Shell (U-Shell) stellt den Einsatzbereich dar, in der die Rahmenbedingungen des SUC und der Einsatzbereich (U-Shell) definiert werden. • Gesetzliche Vorgaben • politische Lage (z.B. Stabilität der gesetzlichen Lage) • Normen und Standards • geographische Regionen • Usage Scenarios (z.B. Fahrsituationen bei Fahrzeugen) • Use Cases (z.B. spezifische Anwendungsfälle) • Dauer des Einsatzes • etc.

Tabelle 13: Inhalte der einzelnen Shells im Shell-Modell

Ein Beispiel aus dem Automotive-Bereich soll die Betrachtung verdeutlichen. Das zu entwickelnde System ist ein „ABS-Steuergerät“. Als Gesamtsystem wird das Fahrzeug definiert, welches in den Antriebsstrang, das Fahrwerk und in die Karosserie unterteilt wird. Das ABS-Steuergerät befindet sich im Fahrwerk.

Das „ABS-Steuergerät“ wird laut der oben angeführten Beschreibung als SUC bezeichnet und befindet sich in der Z-Shell. Das Fahrwerk bildet die O_1-Shell und das Fahrzeug die S-Shell. Diese drei Shells bilden die System-Umgebung. Das Fahrzeug wird von den System-Umwelten beeinflusst, die in der E-Shell dargestellt werden. Die Rahmenbedingungen werden durch den Einsatzbereich, also der U-Shell, vorgegeben.

Wird nun aber beispielsweise nur die Software für das ABS-Steuergerät entwickelt und die Hardware ist bereits vorhanden, so wird die ABS-Software als SUC bezeichnet und befindet sich in der Z-Shell. Das ABS-Steuergerät wird in die O_1-Shell verschoben und wird damit das übergeordnete System erster Ordnung. Der Antriebsstrang wird in die O_2-Shell verschoben und ist somit übergeordnetes System der zweiten Ordnung. Das Fahrzeug bleibt das Gesamtsystem in der S-Shell. Die System-Umwelt mit E-Shell und U-Shell bleibt gleich.

4.2.2 Identifikation der unmittelbaren System-Umgebung

In diesem Unterkapitel wird das systematische Vorgehen bei bei der Identifikation der unmittelbaren System-Umgebung, also die genauen Inhalte der Z-Shell sowie die Definition des SUC beschrieben. Das SUC wird dabei durch ein leeres Rechteck skizziert und weist damit die innersten Systemgrenzen aus. Damit kann zwischen Innen und Außen unterschieden werden, was bei der Anwendung des Shell-Modells besonders wichtig ist, da alles, was sich außerhalb des Systems befindet, untersucht werden soll. Darauf basierend wird später das Innere abgeleitet, also das Konzept bzw. das Design des zu entwickelnden Systems.

Im ersten Schritt wird die *Z-Shell* definiert, d.h. es werden die direkt benachbarten Systeme sowie der/die User beziehungsweise die Bedienelemente identifiziert. Es wird somit die unmittelbare System-Umgebung identifiziert und definiert. Dabei werden die direkt angrenzenden Systeme, welche eine physikalische Verbindung zum entwickelnden System haben, identifiziert. Diese direkt angrenzenden Systeme werden hier als **B**e**N**achbarte **S**ysteme (BNS) bezeichnet. BNS können andere technische Systeme sein, kann aber auch der User sein. Wird eines der technischen BNS von Menschen bedient, so ist dieser als menschliches benachbartes System darzustellen, da das Verhalten der User im sicherheitsrelevanten Bereich von großer Bedeutung ist. Die Schnittstellen zu den BNS sind dabei als besonders kritisch zu betrachten, da das Verhalten des BNS nicht immer eindeutig definiert ist. Sie sind einerseits schwierig zu beschreiben, da in der Projektinitialisierungs-phase, also der Vor-Projektphase, oft keine genauen Kenntnisse über das jeweilige Verhalten des BNS vorliegen und andererseits nie wirklich klar ist, wie sich das Verhalten im Laufe der Zeit verändern kann. Dies kann durch Veränderung des User-Verhaltens bedingt sein, aber auch durch den Alterungsprozess der Hardware beeinflusst werden, oder durch Software-Änderungen. Auch die System-Umwelt kann hier Einfluss nehmen.

Diese benachbarten Systeme lassen sind grundsätzlich in drei Typen einteilen:

1. Unidirektionale BNS (technische Systeme mit unidirektionalem Interface)
2. Bidirektionale BNS (technische Systeme mit bidirektionalem Interface)
3. Menschliche BNS (User)

Der Grund für diese Unterteilung liegt darin, dass diese Arten von BNS substantiell unterschiedliche Eigenschaften besitzen, welche in der Folge zu berücksichtigen sind.

Unidirektionale BNS sind Systeme die, wie der Name schon sagt, nur in *eine Richtung* agieren. Das heißt, hier werden Daten vom BNS abgefragt oder Daten zum BNS abgesetzt. Ein unidirektionales BNS kann beispielsweise ein Sensor sein, der zyklisch abgefragt wird oder ein Aktuator, der vom System gesteuert wird.

Bidirektionale BNS sind komplexere Systeme, die in beide Richtungen kommunizieren, eher ein dynamisches Verhalten aufweisen und einen Request-Response Dialog führen. Ein bidirektionales BNS kann beispielsweise ein Bussystem sein. Bei dieser Kategorie von BNSen spielt das Zeitverhalten eine wesentliche Rolle, da ein erforderlicher „Response“ nur für einen genau definierten Zeitrahmen zur Verfügung steht. Hier ist bei der System-Entwicklung darauf zu achten, dass die „Antwortzeit“ des zu entwickelnden Systems genau spezifiziert wird.

Menschliche BNS sind Menschen, die das System benutzen, und somit einen direkten Einfluss auf das System haben. Die Interaktion zwischen Mensch und System muss über ein definiertes User Interface erfolgen. Selbst wenn dieses User Interface nicht Teil des zu entwickelten Systems ist, sollte der User und sein Verhalten in diesem Schritt schon mitbetrachtet werden. Die direkt benachbarten Systeme und Benutzer werden in Abbildung 37 in der Systematik der Shell-Modell Methode dargestellt.

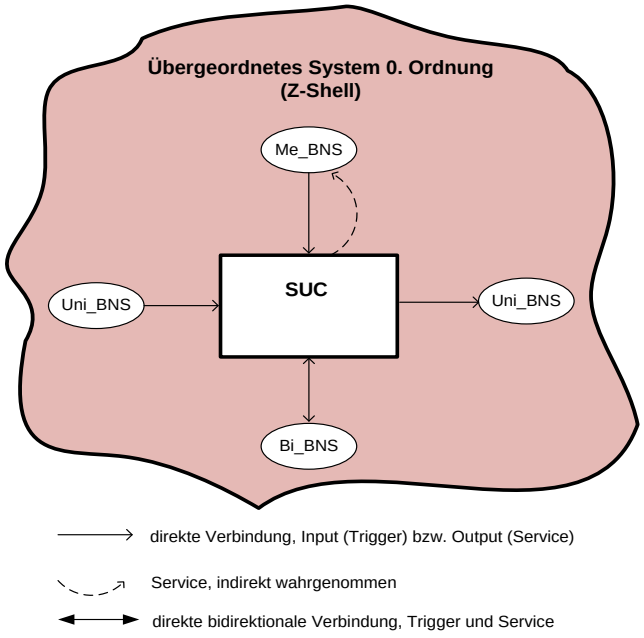


Abbildung 37: Kontext-Diagram in der Z-Shell

Sind alle BNS identifiziert, so werden im nächsten Schritt die System-Funktionen definiert. Die hierfür gewählte Sichtweise entspricht dem in Kapitel 2.1.1 vorgestelltem funktionalen Systemkonzept, wobei jedoch anstatt von „Input“ und „Output“ die Begriffe „Trigger“ und „Service“ verwendet werden. Das heißt, als erstes werden die Trigger, die auf das zu entwickelnde System einwirken, identifiziert. Ein Trigger ist hier als ein Ereignis zu verstehen, welches eine Funktionalität im System auslösen soll. Diese daraufhin vom System zur Verfügung gestellte Funktionalität ist das „Service“, welches das System an seine System-Umgebung liefern soll. Die einfachste Form dieses „Trigger-Service“ Modells, wäre beispielsweise, wenn ein Benutzer einen Schalter schließt (Trigger) und das System daraufhin eine Funktionalität aktiviert, beispielsweise ein Licht eingeschaltet wird. Trigger können von Menschen, benachbarten Systemen wie Sensoren oder Bussystemen oder auch vom System selbst (wie beispielsweise durch ein internes Timeout) initiiert werden. Für jeden Trigger soll nun ein Szenario überlegt werden, welches beschreibt, wie das System darauf reagieren sollte, beispielsweise welches Service das System erbringen muss.

Zusammenfassend kann gesagt werden, dass das Shell-Modell notwendig ist, um den Systemzweck zu schärfen, die damit verbundenen Anforderungen an das SUC im Gesamtkontext zu erkennen und um die exogenen Hazards identifizieren zu können. Das Shell-Modell zeigt eine ganzheitliche Betrachtung der System-Umgebung und fordert eine klar definierte Systematik zur Identifikation der Anforderungen. Die ganzheitliche Systembetrachtung wird mit dem Shell-Modell klar dargestellt, es reduziert „Flawed Requirements“ und damit auch das Risiko vor Fehlinterpretationen und es vereinfacht die Hazard-Analysen.

4.3 Projekt-Stakeholder-Analyse

Jedes Projekt wird von seinem Umfeld beeinflusst, ebenso kann das Projekt wiederum sein Umfeld beeinflussen – diese Einflussbereiche müssen in Projekten analysiert werden. Im sicherheits-relevanten Bereich sind diese Einflussfaktoren besonders wichtig, da sie nicht nur das Projektergebnis maßgeblich mitbestimmen, sondern auch ihre Auswirkungen in Bezug auf die Systemsicherheit ausschlaggebend sind. Das können beispielsweise Lieferanten sein, von denen die Organisation abhängig ist, die lediglich nichtzertifizierte COTS-Komponenten mit vielen Konfigurationsmöglichkeiten zur Verfügung stellen. Hier ergibt sich ein erheblicher Aufwand, um diese Komponenten letztlich im sicherheitskritischen Umfeld einsetzen zu können.

Die deutsche Gesellschaft für Projektmanagement [GPM_05] definiert den Begriff des Projektumfeldes wie folgt und referenziert dabei auf die International Competence Baseline (ICB) der International Project Management Association (IPMA): „Laut ICB (IPMA Competence Baseline) ist das Projektumfeld die Umgebung, in der das Projekt formuliert, bewertet und durchgeführt wird und die das Projekt direkt und indirekt beeinflussen und/oder von dessen Auswirkungen betroffen ist. Diese äußeren Einflüsse können „physische, ökologische, gesellschaftliche, psychologische, kulturelle, politische, wirtschaftliche, finanzielle, juristische, vertragliche, organisatorische, technologische und ästhetische Faktoren sein“

Die Projektumfeldanalyse nach Patzak [Pat_04, S.68ff] geht ähnlich vor, sie dient der Erfassung von Einflussfaktoren auf das Projekt und der frühzeitigen Erkennung von Potentialen und

Problemfeldern. Von der Projektumfeldanalyse werden entsprechende Aktivitäten für den Projektmanager zur Optimierung der Beziehungen zu den jeweiligen Umwelten abgeleitet. Typische Einflussgrößen sind Personen, Personengruppen, Organisationen oder andere Institutionen, die durch ihr Handeln bzw. Unterlassen den Projektablauf beeinflussen können.

Im ersten Schritt werden alle Einflussgrößen aus dem Projektumfeld erfasst und einer detaillierten Analyse unterzogen. Entsprechende Maßnahmen werden abgeleitet und in der Projektplanung berücksichtigt. Die nachfolgende Abbildung 38 zeigt ein Beispiel eines Projektumfeldes.

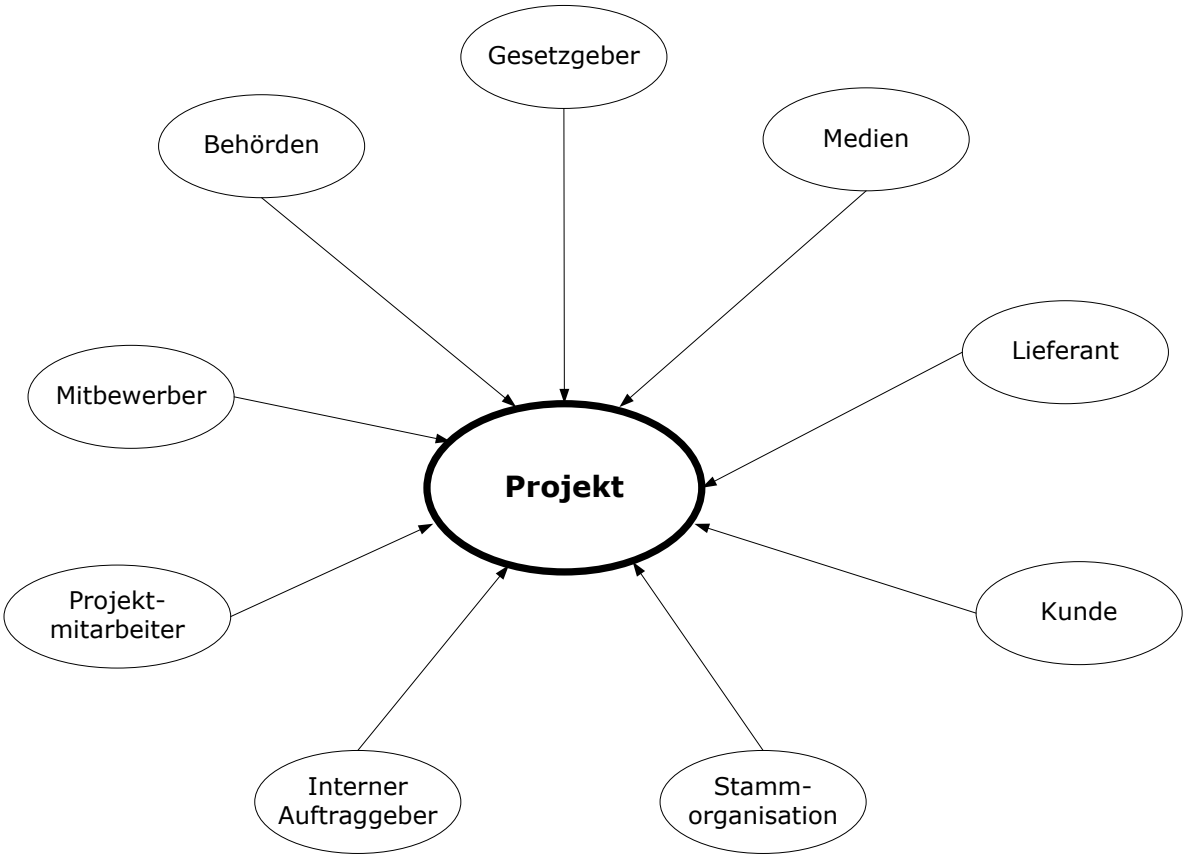


Abbildung 38: Stakeholderanalyse (Beispiel)

Neben dem Begriff der Projektumfeldanalyse verwendet das weltweit anerkannte Project Management Institute (PMI) im PMBOK Guide[65] [PMB_13, S.29] den Begriff der Stakeholder-Analyse. Das PMBOK bezeichnet den Project Stakeholder als "individual, group, or organization who may affect, be affected by, or perceive itself to be affected by a decision, activity or outcome of a project. Stakeholders may be actively involved in the project or have interests that may be

[65] Der PMBOK Guide wurde 1998 vom American National Standard Institute (ANSI) und vom Institute of Electrical and Electronics Engineers (IEEE) als Standard anerkannt.

positively or negatively affected by the performance or completion of the project."[66]. Folgende Aufgaben legt das PMBOK zur Stakeholder-Analyse fest: "The project team identifies internal and external, positive and negative, and performing and advising stakeholders in order to determine the project requirements and the expectations of all parties involved."[67] In der 5. Auflage des PMBOK Guide wurde das Stakeholder-Management als eigenes Wissensgebiet etabliert.

Die ISO 10006 [IS10_03] spricht beim Begriff Stakeholder von projektinteressierten Parteien. Partsch [Par_10] bezieht sich dabei auf Personen oder Organisationen, die ein potentielles Interesse an dem zukünftigen, zu entwickelnden System haben und somit auch Anforderungen an das System stellen können. Stakeholder können nach [Par_10] beispielsweise Interessensträger, Wissensträger, Projektbeteiligte oder Systembetroffene sein.

Im Requirements Engineering wird eine ähnliche Analyse-Methode angewendet. Die neue ISO/IEC 15504 Teil 5 [ISO15_11] (2011 als FDIS-Version[68] veröffentlicht) definiert den Prozess „Stakeholder Requirements Definition" und begründet den Prozess-Zweck folgendermaßen: „The purpose of the Stakeholder requirements definition process is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment.[69]"

Da es sich hier sowohl auf Projektmanagement-Ebene als auch auf Requirements-Ebene um die Einflussfaktoren handelt, die einerseits auf der organisatorischen Ebene und andererseits auf das technische System Einfluss nehmen, wird in dieser Arbeit das Vorgehen der Analyse dieser Einflussfaktoren über beide Bereiche systematisiert. Damit sollen Doppelaktivitäten vermieden und Synergien zwischen dem Projektmanagement und dem Requirements Engineering genutzt werden. Das Vorgehen wird in dieser Arbeit als „*Stakeholder-Analyse*" bezeichnet.

Das heißt, die im Projektteam identifizierten Stakeholder werden einerseits vom Projektmanager zur Optimierung der Beziehungen zwischen Stakeholder und Projekt herangezogen und andererseits vom Requirements Ingenieur zur Erhebung der Anforderungen verwendet. Bei der Erhebung der Anforderungen gilt es, die Anforderungen der identifizierten Stakeholder an das zu entwickelnde System zu ermitteln, zu analysieren, zu verstehen und zu dokumentieren. Dies bedarf einer intensiven Kommunikation mit den Stakeholdern in mehreren Feedbackzyklen.

Ziel dieser erweiterten Methode ist es, einerseits möglichst alle Einflussfaktoren die auf der organsatorischen Ebene auf das Projekt einwirken, identifizieren zu können und andererseits möglichst alle Anforderungen an das System genau festlegen zu können. Eine große Hilfestellung leistet hier das Shell-Modell (siehe Kapitel 4.2), das über die einzelnen Shells in systematischer

---

[66] "Individuum, Gruppe oder Organisation, welche eine Entscheidung im Projekt, eine Aktivität im Projekt oder das Ergebnis des Projektes beeinflussen können, davon betroffen sein können, oder sich davon betroffen fühlen können." (Übersetzung des Verfassers)

[67] "Das Projektteam identifiziert interne und externe, positive und negative, handelnde und beratende Stakeholder, um die Anforderungen und Erwartungen aller involvierten Parteien zu ermitteln." (Übersetzung des Verfassers)

[68] FDIS … Final Draft International Standard

[69] „Zweck der Stakeholder Anforderungs-Definition ist die Definition der Anforderungen an ein System in seinem definierten Umfeld, die vom Benutzer bzw. Stakeholdern gefordert werden." (Übersetzung des Verfassers)

Weise die Einflussbereiche darstellt. Dabei werden nicht nur die technischen Anforderungen an das System identifiziert, sondern auch Rahmenbedingungen und Einschränkungen, die beispielsweise aus der Shell des Einsatzbereiches identifiziert werden. Das können gesetzliche Auflagen aus diversen Ländern oder Anforderungen von einzuhaltenden Normen und Standards sein.

Mit diesem Vorgehen lassen sich der Projekt-Umfang und der Umfang des zu entwickelnden Systems festmachen. Eine gewisse Vollständigkeit der Anforderungen kann dabei nur gewährleistet werden, wenn die Anforderungen systematisch identifiziert, analysiert, konsolidiert, dokumentiert und einem Review mit allen betroffenen Stakeholdern unterzogen werden. Damit wird auch der Gefahr des „Scope Creep“[70] entgegengewirkt.

Sind Anforderungen für komplexe, abstrakte Systeme zu erheben, bedarf es fundierter Kenntnisse im Requirements Engineering. Diese Aufgabe muss dann von ausgebildeten Requirements-Ingenieuren durchgeführt werden. Neben den externen Stakeholdern sind auch interne Stakeholder, wie z.B. System-Architek, Entwickler und Tester in die Betrachtung mit einzubeziehen. Lassen sich nicht alle relevanten Anforderungen erheben, kann weder die Funktionalität des Systems vollumfänglich gewährleistet, noch dessen Leistungsfähigkeit und Sicherheit dargestellt werden. Auch kann es dadurch zu Inkonsistenzen zwischen unterschiedlichen Spezifikationen kommen, da diese eventuell auf unterschiedlichen Annahmen basieren.

4.4 Extended Functional Failure Analysis

Es wird hier nochmals kurz auf die FFA eingegangen, da diese für eine optimale Anwendung, im speziellen in der Preliminary Hazard Identification (PHI), also im Problemraum, noch erweitert werden muss. Die grundsätzliche vorgeschlagene Vorgangsweise nach Mc Dermid [McD_02] ist in Kapitel 2.5.2 beschrieben. Das heißt, hier werden Funktionen mit den Guidewords kombiniert, um mögliche Fehlfunktionen, also Failures/Malfunctions/Deviations, identifizieren zu können.

Um die FFA in der PHI sinnvoll anwenden zu können, muss sie erweitert werden. Diese erweiterte FFA ergänzt die Elemente der Hazard Analysis nach dem Hazard Triangle und wird in dieser Arbeit als „extended Functional Failure Analysis, eFFA“ bezeichnet. Zusätzlich stehen im Rahmen der eFFA auch noch folgende Informationen aus den Ergebnissen der Projekt-Initialisierung und der Konzeptionierung zur Verfügung: Lessons Learned aus früheren Projekten (sofern verfügbar), Hazard- und/oder Unfall-Checklisten, die aus anderen Projekten oder aus dem Anwendungsbereich bekannt sind, sowie die durch das Shell-Modell (Kapitel 4.2) identifizierten erwarteten Umgebungsbedingungen. Diese Zusatzinformationen dienen der erweiterten Betrachtung bei der Identifikation von Failures. Damit werden einerseits Failures nicht fälschlicherweise ignoriert, andererseits aber auch nicht unrealistische Szenarien als zu realistisch eingeschätzt. Die nachfolgende Abbildung 39 gibt einen Gesamt-Überblick, über die eFFA.

[70] Scope Creep … schleichender Inhalts- und Umfangszuwachs in einem Projekt

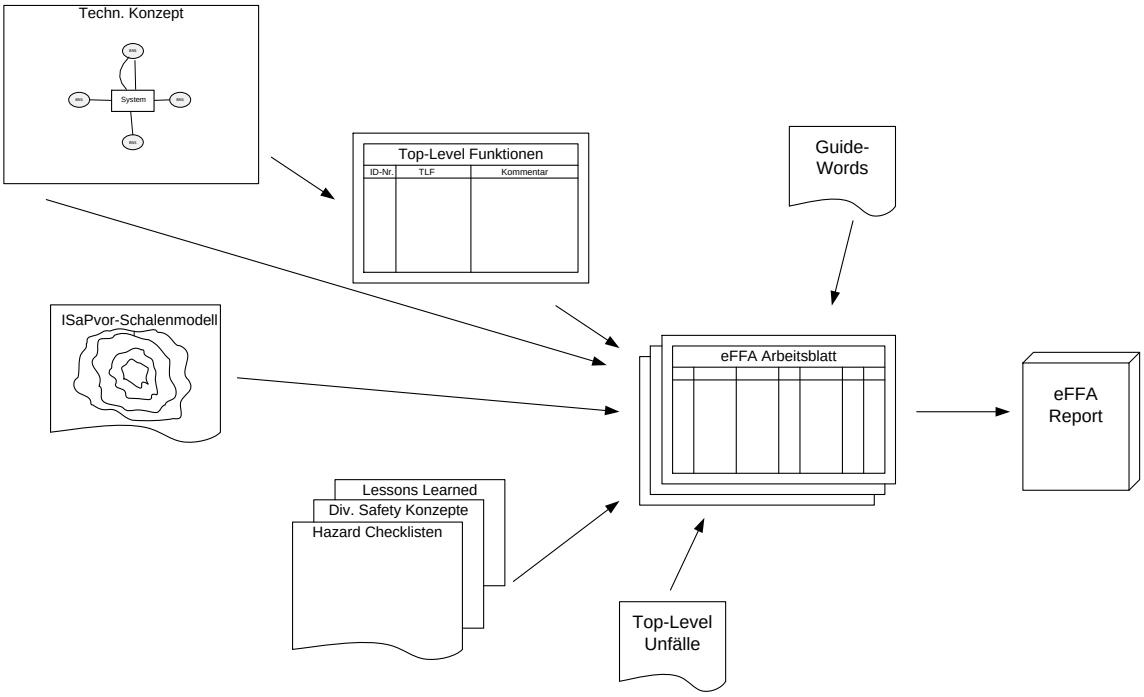


Abbildung 39: Erweiterte Functional Failure Analysis (eFFA)

Wichtig bei der Anwendung der Anwendung der eFFA ist die richtige Auswahl der Guidewords. Redmill [Red_99] begründet das so: "Guide words may be interpreted differently in different industries, at different stages of the system's life cycle, and when applied to different design representations." Die eFFA wird beim Safety-Vorgehensmodell in der PHI eingesetzt, daher ist es wichtig, nur wenige Guidewords zu verwenden, um die Analyse so effizient wie möglich zu halten. Für die richtige Auswahl der Guidewords wird das technische Konzept herangezogen. Nachfolgend sind diejenigen Guidewords aufgelistet, die dafür in Frage kommen, was aber nicht heißt, dass die Auswahl nicht erweitert werden kann.

Wie in Kapitel 2.5.2 gezeigt, ist das Guideset von Mc Dermid [McD_02] auf Top-Level Funktionen, wie sie in Kapitel 3.1.2 Konzeptionierung verwendet werden, gut anwendbar:

- Function not provided when required (omission)
- Function provided when not intended (commission);
- Function provided incorrectly (incorrect).

Soll das Zeitverhalten von Schnittstellen analysiert werden, empfiehlt Redmill [Red_99] die Anwendung folgender Guidewords:

- Early: something happens earlier in time than intended
- Late: something happens later in time than intended
- Before: something happens earlier in a sequence than intended
- After: something happens later in a sequence than intended

Geht es um den richtigen Datenfluss, so werden die nachfolgenden Guidewords angewendet:

- More data rate: data may be passed at a higher rate than intended
- More data quantity: More data may be passed than intended

Es sei auch erwähnt, je länger die Liste an Guidewords ist, desto mehr Funktion-Guideword Kombinationen müssen untersucht werden, und desto vollständiger ist die Analyse, desto aufwendiger ist dieses allerdings auch. Aus diesem Grund sind genaue Kenntnisse über das System, die Systemumgebung und das System-Umfeld wichtig, damit die richtigen Guidewords ausgewählt werden können. Da die Ressourcen in der Vor-Projektphase häufig sehr eingeschränkt zur Verfügung stehen, werden maximal vier Guidewords angewendet.

Bei jeder Funktion-Guideword Kombination wird nun folgende Analyse durchgeführt:

1. Ist die resultierende Fehlfunktion realistisch und plausibel?
   Die Beantwortung dieser Frage ist nicht vollständig objektiv möglich, da zu diesen frühen Phasen noch keine Quantifizierung vorgenommen werden kann. Letztlich hängt sie auch stark von der Expertise der an der eFFA teilnehmenden Experten ab. Um jedoch trotzdem eine möglichst exakte Analyse durchzuführen, sollten bei der Beantwortung dieser Frage Erfahrungen aus vergangenen Projekten („Lessons Learned“, Checklists) sowie die möglichen Betriebsmodi und Umgebungsbedingungen aus dem Shell-Modell, einfließen.
2. Ist die resultierende Fehlfunktion gefährlich?
   Wenn ja, dann wird sie als Hazard klassifiziert. Mithilfe des Hazard-Triangles, wie in Kapitel 2.1.4 beschrieben, kann entschieden werden, ob eine Fehlfunktion ein Hazard ist. Auch hier gilt dasselbe wie bei Frage 1. Um die Gefährlichkeit einzuschätzen ist insbesondere das Shell-Modell relevant, da alle möglichen „Targets/Threats“ (siehe Kapitel 2.1.4) vorkommen, und man somit bestimmen kann, ob die Fehlfunktion ein Gefahrenpotential birgt.
3. Wie schwerwiegend und wahrscheinlich ist ein resultierender Unfall?
   Daraus ergibt sich das Risiko, welches eine vorläufige Einschätzung der Sicherheitsanforderungsstufe erlaubt.

Die ermittelten Fehlfunktionen werden im nächsten Schritt anhand des Hazard-Triangles analysiert und dementsprechend als Hazard oder nicht Hazard, also als Failure klassifiziert. Die nachfolgende Abbildung 40 zeigt das Template der eFFA.

extended Functional Failure Analysis							
Failure Identifikation				**Hazard Analyse und Definition**			
#	**Funktion**	**Guideword**	**Failure**	**Hazard Element**	**Trigger**	**Konsequenzen / Accident**	**Hazard**
		Omission					
		Commission					
		Incorrect					

Abbildung 40: eFFA Template

Das Template ist in die „Failure Identifikation“ und die „Hazard Analyse und Definition“ unterteilt. Die Failure Identifikation funktioniert genauso wie die FFA von Mc Dermid (siehe Kapitel 2.5.2). Die identifizierten Failures werden in der Folge einer Hazard Analysis nach dem Hazard Triangle (sieh Kapitel 2.1.4) unterzogen.

4.5 Exogene Hazard Analysis (exHA)

Es wurde im Zuge dieser Arbeit die exogenen Hazard Analysis entwickelt, da es bis dato keine Analyse-Methode gibt, die sich dieser Problematik widmet. Exogene Hazards sind Gefährdungen die vom zu entwickelnden System ausgehen, deren Ursachen jedoch durch die Systemumgebung ausgelöst werden. Um ein Inherent Safe System Design entwickeln zu können, muss das zu entwickelnde System gegen solche Einflussfaktoren resistent sein.

Die Analyse-Methode ist in zwei Bereiche geteilt, das ist die „Failure Identifikation“ und die „Hazard Analyse“. Die Failure Identifikation untersucht im ersten Schritt alle benachbarten System und deren exogene Einflüsse auf das zu entwickelnde System. Basierend darauf wird die Reaktion des Systems analysiert und deren unmittelbaren Auswirkungen auf den System-Kontext. Dafür wird das Shell Modell herangezogen, um über alle Systemgrenzen, bis zur System-Umwelt (U-Shell), die unmittelbaren Auswirkungen darstellen zu können. Mit der Hazard Analysis, kann festgestellt werden, ob es sich um einen Hazard handelt oder nicht. Dafür wird das Hazard Triangle herangezogen. Können Hazard Element und Initiation Mechanism (IM) identifiziert werden, so handelt es sich um einen Hazard. Dies hat zur Folge, dass die Konsequenzen, ausgehend von den unmittelbaren Auswirkungen, anhand der erweiterten Fehlerkette (siehe Kapitel 4.1.2) identifiziert werden müssen. Dabei werden die Konsequenzen bis hin zum Accident untersucht. Die nachfolgende Abbildung 41 zeigt das Template für die exogene Hazard Analyse.

Exogene Hazard Analysis								
Failure Identifikation				Hazard Analyse				
#	Benachbartes System	Exogene Einflüsse/Failure	Unmittelbare Auswirkungen	Hazard Element	Trigger	Konsequenzen / Accident	Top-Level	Kommentar
1								
2								
3								
4								
5								
6								
7								

Abbildung 41: Exogenen Hazard Analyse (exHA) Template

Wie in der Abbildung ersichtlich, werden alle benachbarten Systeme (siehe Spalte 2) und deren exogenen Einflüsse (Spalte 3) analysiert. Danach werden die unmittelbaren Auswirkungen an der innersten Systemgrenze identifiziert (Spalte 4). Jeder Failure wird danach untersucht, ob er ein Hazard-Element (Spalte 5) und ein oder mehrere IM (Spalte 6) vorhanden sind, die einen Accident (Spalte 7) auslösen könnten.

4.6 Hazard-Analyse und Safety-Risiko-Assessment

Die „Hazard Analysis“ und das „Risk Assessment“ sind das Herzstück von Safety-Normen. Sie dienen der Identifikation von Hazards und der Kategorisierung von gefährlichen Ereignissen. Dabei liefern sie die Grundelemente zur Bestimmung der Sicherheitsanforderungsstufe sowie zur Definition der Sicherheitsfunktionen, in der ISO 26262 [ISO26_11] der Safety Goals[71]. Die beiden Normen lassen dabei einen sehr großen Interpretationsspielraum zu. Die IEC 61508 [IEC08_10] fordert beispielsweise im normativen Teil 1 lediglich, dass eine Hazard- und Risikoanalyse durzuführen ist. Sie verweist auf den informativen Teil 5, in dem verschiedene Teilschritte für die Bestimmung der Sicherheitsanforderungsstufe beschrieben sind, liefert aber kein systematisches Vorgehen und keine konkret anzuwendende Methode, wie Hazards identifiziert werden sollen. Die [ISO26_11] empfiehlt im Teil 3, Kapitel 7.4.2.2.1 die Anwendung von Brainstorming, Checklisten, FMEA und Feldstudien.

Beide Normen zeigen, wie bei der Safety-Risiko-Bewertung vorzugehen ist, im Besonderen die ISO 26262. Sie ist bei der Bestimmung der Sicherheitsanforderungsstufe, hier als Automotive Safety Integrity Level (ASIL) bezeichnet, sehr genau und liefert dafür detaillierte Parameter. In beiden Normen findet sich jedoch kein systematisches Vorgehen, wie Hazards identifiziert, analysiert und definiert werden. Ein nicht-systematisches Vorgehen kann zu großen Lücken bei der Hazard-Analyse in frühen Entwicklungsphasen führen und Fehleinschätzungen zur Folge haben.

Wird darüber hinaus die Hazard-Analyse selbst nicht ordnungsgemäß durchgeführt, kann dies zur Folge haben, dass sie nicht vollständig (Hazards wurden übersehen, Flawed Requirements) oder unrealistisch (nichtexistente bzw. zu vernachlässigende und dadurch falsche Hazards wurden identifiziert) ist. Eine mangelhafte Bewertung der Hazards führt zu nicht korrekt abgeleiteten Safety Goals, zu einer nicht korrekten Sicherheitsanforderungsstufe und letztlich zu fehlenden oder fehlerhaften Maßnahmen bei der Implementierung.

Ein weiteres Grundproblem liegt auch in der mangelnden Erfahrung der Entwicklungs-Ingenieure bei der Anwendung der Analyse-Methoden. So argumentiert auch Ständer [Stä_10, S. 10], dass die Safety-Analyse Tätigkeiten meist von verschiedenen ungeübten Anwendern durchgeführt werden. Daraus folgend gibt es unterschiedliche Ergebnisse, die stark von der subjektiven Meinung der Anwender abhängig sind. Die Anwendung von meist klassischen Methoden des Qualitätsmanagements, wie sie beispielsweise von der ISO 26262 vorgeschlagen werden, verstärkt die Problematik, da sich damit wiederum nur Failure identifizieren lassen und keine Hazards. Auch hier kann ein systematisches Vorgehen, wie das Safety-Vorgehensmodell, die Problematik eindämmen.

In vielen Fällen findet die FMEA Anwendung bei der Hazard- und Risikoanalyse, die individuell je nach Anwendungsfall und Kenntnis der Anwender erweitert wird. Die FMEA ist eine Methode mit der Failures (siehe Kapitel 2.5.5) identifiziert und bewertet werden, ebenso wie beim Einsatz von der

[71] Ein Safety Goal ist ein Safety Top-Level Requirement, welches das Resultat der Hazard Analysis und des Risiko Assessments ist [ISO26_11].

HAZOP oder FFA. Der Name HAZOP spricht zwar von einer Hazard and Operability Analyse, aber wenn man das Vorgehen genau betrachtet, werden in der Praxis lediglich Failures anhand von Guidewords identifiziert. Eine Failure muss aber nicht unbedingt gefährliche Auswirkungen haben und darf nicht ungeprüft als Hazard bezeichnet werden.

Die grundlegende Frage, die sich hier stellt, ist also, nach welchen Kriterien man eine Failure einem Hazard zuordnet bzw. welche Aspekte muss eine Hazard-Beschreibung beinhalten, damit man ihn klar von einer Failure unterscheiden kann? Diese Frage soll im nachfolgenden Kapitel beantwortet werden, das sich der Anwendung der systematischen Hazard-Identifikation, -Analyse und -Beschreibung sowie dem darauffolgendem Safety-Risiko-Assessment widmet. Es werden in weiterer Folge die Ausdrücke „Hazard-Identification“ und „Risk Assessment“ verwendet, da es in erster Linie um die Identifikation von Hazards geht und in zweiter Linie um die Bewertung des Risikos, der darauf basierenden Festlegung der Sicherheitsanforderungsstufe und der Definition der Sicherheitsziele (Safety Goals). Hier spricht das Safety-Vorgehensmodell vom Assessment des Risikos. Dieses Kapitel konzentriert sich dann auch auf das systematische Vorgehen bei der Analyse der Root-Causes. Als Basis dafür dienen die erweiterte Fehlerkette aus Kapitel 4.1.2 und das Hazard Triangle aus Kapitel 2.1.4 sowie das Shell-Modell aus Kapitel 4.2.

4.6.1 Hazard-Identifikation und -Analyse

Betrachtet man die erweiterte Fehlerkette, so ist der Ausgangspunkt bei der Hazard-Identification die Failure an der Systemgrenze. Anhand der nachfolgenden Abbildung 42 sollen diese Zusammenhänge nochmals kurz dargestellt werden. Wie in Kapitel 4.1.2 beschrieben, wird eine Failure durch einen Fault, der sich vorerst systemintern als Error auswirkt, verursacht. Eine Failure ist eine Abweichung von der gewünschten Funktionalität, die als Fehlfunktion oder Ausfall an der Systemgrenze sichtbar wird. Die Identifikation einer Failure erfolgt unter der Anwendung von den genannten klassischen Analyse-Methoden, wie beispielsweise der FFA, der HAZOP oder der FMEA. Das Ergebnis sind identifizierte Failure auf System-Ebene. Hat die Failure gefährliche Auswirkungen, so wird sie als Dangerous Failure, also als Hazard, klassifiziert. Failures werden nun anhand der Elemente des Hazard-Triangle (siehe Kapitel 2.1.4) analysiert. Im ersten Schritt werden die Hazard-Elemente identifiziert. Können keine Hazard-Elemente identifiziert werden, so handelt es sich um keinen Hazard, sondern um eine Failure. Im zweiten Schritt werden mögliche Ereignisse, also IM (Initiation Mechanism), festgemacht, die von einer gefährlichen Failure zu einem Unfall führen können. Die im dritten Schritt identifizierten gefährdeten Lebewesen und Umwelten zeigen die Auswirkungen eines Unfalls. Kann dieser Zusammenhang von Hazard-Element, IM und Thread dargestellt werden, so ist ein Hazard klar begründet.

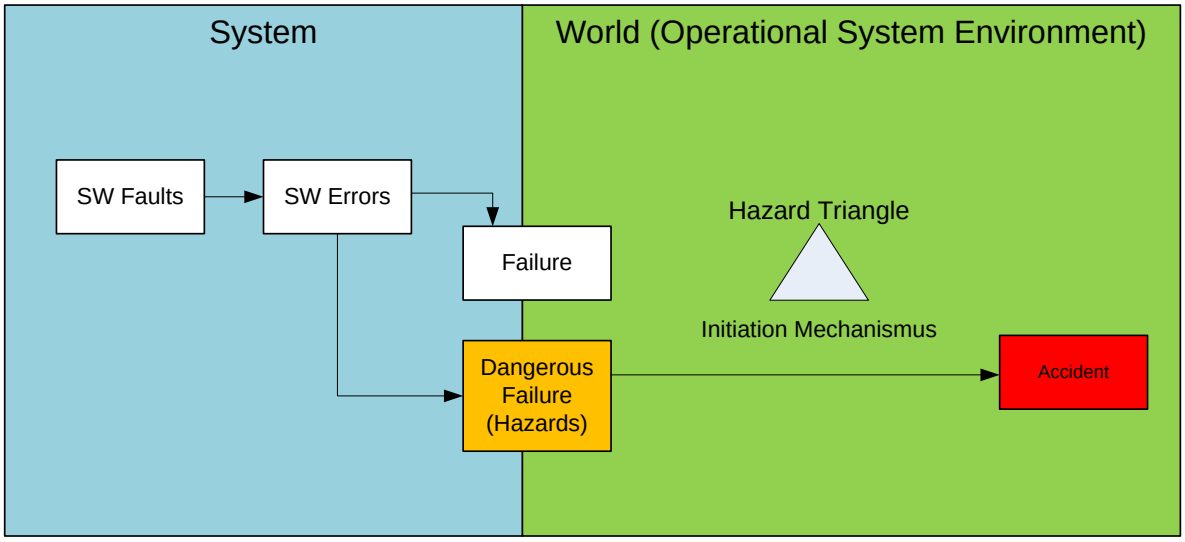


Abbildung 42: System und Operational System Environment in der Erweiterten Fehlerkette

Als Beispiel soll hier das Abblendlicht eines Autos analysiert werden. Die Failure wäre das Flackern des Abblendlichtes. Ein flackerndes Abblendlicht muss jedoch nicht unbedingt ein Hazard sein. Die Failure wird erst zum Hazard, wenn der Beschreibung der Failure ein oder mehrere Hazard-Elemente hinzugefügt werden können, wie beispielsweise, das Abblendlicht flackert bei einer nächtlichen Fahrt auf einer Landstraße bei einer *Geschwindigkeit von größer 80 km/h*. Das Hazard-Element ist hier die Geschwindigkeit. In dieser Beschreibung können nun auch die Umstände, die zu einem Unfall führen, festgelegt werden. In diesem Fall wäre es eine kurvenreiche Strecke, der Initiation Mechanism (IM), der auch eine plötzlich auftauchende Nebelzone sein kann. Es kann aber auch starker Gegenverkehr sein, der hier einen Unfall auslösen kann. Wie aus diesem Beispiel hervorgeht, können verschiedene IM, aber auch eine Kette von IM einen Unfall auslösen. In weiterer Folge müssen all diese Möglichkeiten bei der Hazard-Identifikation berücksichtigt werden. Die Einflussfaktoren, die hier als IM wirken können, können aus dem Operational System Environment, abgeleitet werden.

Eine eindeutige Beschreibung eines Hazards, welcher die oben angeführten Kriterien erfüllt, hat maßgeblichen Einfluss auf das Safety-Risiko-Assessment und auf die Definition der Safety Goals. Die Definition der Safety Goals wirken sich wiederum auf das Design und damit auf die Sicherheit des Systems aus. Abbildung 42 zeigt auch denjenigen Teil der erweiterten Fehlerkette, der bei der Erarbeitung des Vorgehens für die Identifikation und Bewertung von Hazards herangezogen wurde. Das Hazard-Triangle, im Pfad vom Hazard zum Accident, symbolisiert die Kriterien, die nachweisen, dass eine Failure zu einem Hazard wird und in Folge zu einem Unfall/Accident führen kann.

4.6.2 Failure Analyse und Hazard Beschreibung

Wie im vorherigen Kapitel beschrieben, werden Failures anhand von diversen Analyse-Methoden identifiziert und mithilfe des Hazards-Triangle einem Hazard zugeordnet. Sind die Hazards erstmals identifiziert und analysiert, so müssen diese im nächsten Schritt eindeutig beschrieben werden. Dabei ist wichtig, dass in der Beschreibung des Hazards die Hazard-Elemente und Initiation Mechanism (IM) enthalten sind. Im oben angeführten Beispiel des flackernden Lichtes würde die Hazard-Definition folgendermaßen lauten:

> Ein plötzlich auftretendes flackerndes Abblendlicht eines Fahrzeugs mit *hoher Geschwindigkeit* auf einer *nicht beleuchteten kurvenreichen Landstraße.*

Die Konsequenzen werden in der Analyse angeführt, in der Hazard-Beschreibung jedoch nicht erwähnt, da sie nicht Teil des Hazards sind, sondern nur dessen Auswirkungen/Konsequenzen aufzeigen. Die Konsequenzen liefern jedoch ein zusätzliches Argument, dass es sich um einen potentiellen Hazard handelt. Eine eindeutige Hazard-Beschreibung ist auch notwendig, damit diese von jedermann verstanden wird.

Die nachfolgende Abbildung 14 zeigt das Template mit Beispielen zur Hazard-Analyse und Hazard-Beschreibung, ausgehend von einem Failure.

Nr.	**Failure oder Hazard?**	**Hazard Element€**	**IM**	**Konsequenzen / Unfall**	**Hazard Beschreibung**
1	Flackerndes Abblendlicht	Hohe Geschwindigkeit	Kurvenreiche Strecke Nächtliche Fahrt	Fahrzeug verlässt bei starker Kurve die Fahrbahn	Plötzlich auftretendes flackerndes Abblend-licht eines Fahrzeugs mit hoher Geschwindig-keit auf einer kurven-reichen Landstraße
2	Unbeabsichtigtes Bremsmoment tritt ein	Auf der Landstraße bei *hoher Geschwindigkeit*	Dichter Verkehr	Fahrer verliert Kontrolle, Lebensgefährliche Verletzungen	Ein unbeabsichtigtes Bremsmoment tritt bei der Fahrt auf einer Landstraße bei hoher Geschwindigkeit und dichtem Verkehr ein
3	Drehmoment des Motors wird plötzlich hochgeregelt, Fahrzeug beschleunigt	Ungewollte *Beschleunigung* auf der Autobahn	Keine Überhol-möglichkeit	Fahrzeug fährt vorderem Auto auf, Massen-karambolage Lebensgefährliche Verletzungen	Das Drehmoment des Motors wird bei der Fahrt auf der Autobahn plötzlich hochgeregelt, das Fahrzeug beschleunigt von 100 auf 150 km/h innerhalb von 15 Sekunden und es besteht keine Überholmöglichkeit

4	Sitz-Justage schaltet sich plötzlich ein und fährt den Fahrersitz nach vor	Der Fahrersitz wird während der Fahrt auf einer Landstraße bei einer *Geschwindigkeit von 100 km/h* plötzlich nach vorne gefahren	Bremse wird vor einer Kurve verzögert erreicht	Fahrer kann nicht rechtzeitig bremsen, Fahrzeug fliegt in der Kurve raus Schwere Lebensgefährliche Verletzungen	Der Fahrersitz wird während der Fahrt auf einer kurvenreichen Landstraße bei einer Geschwindigkeit von 100 km/h plötzlich nach vorne gefahren
4	Airbag löst plötzlich ungewollt aus	Airbag löst während der Fahrt auf der Autobahn bei einer *Geschwindigkeit von 130 km/h* aus	Fahrzeug weicht von der Fahrbahn ab	Fahrzeug verursacht Massenkaram-bolage auf der Autobahn Schwere Lebensgefährliche Verletzungen	Airbag löst während der Fahrt auf der Autobahn bei einer Geschwindigkeit von 130 km/h aus

Tabelle 14: Failure Analyse und Hazard Beschreibung anhand des Hazard Triangles

4.6.3 Safety Risiko Assessment

Nach der Hazard-Identification erfolgt das Risk Assessment, darauf aufbauend werden die Maßnahmen zur Risiko-Vermeidung bzw. Minimierung abgeleitet. Nach dem Risk Assessment, das stark von der anzuwendenden Safety-Norm abhängig ist, wird die Sicherheitsanforderungsstufe bestimmt. Darauf basierend werden wiederum die Sicherheitsziele (Safety Goals) festgelegt und die Safety Requirements abgeleitet. All diese Teil-Aktivitäten lassen sich im Safety-Vorgehensmodell unter dem Begriff „Safety Risk Assessment“ zusammenfassen. Dieses Kapitel konzentriert sich vorwiegend auf das Safety Risk Assessment, da diese in den Safety-Normen sehr unterschiedlich ist.

Die eindeutige Beschreibung der Hazards ist für die Bewertung des Risikos unbedingt notwendig. Damit kann der subjektive Interpretationsspielraum stark eingeschränkt werden. Die Anzahl der IM sind wichtige Indikatoren bei der Analyse der Eintrittswahrscheinlichkeit, die Threats/Targets zeigen die Konsequenzen auf und beschreiben den Unfall (Accidents).

Zur Bewertung des Risikos gibt es eine Vielzahl von Verfahren. Einige Safety-Normen fordern ein definiertes Verfahren, wie beispielsweise die ISO 26262, das nachfolgend beschrieben wird. Andere, wie die IEC 61508, lassen hier viel Interpretationsspielraum und schlagen lediglich im informativen Bereich der Norm ein mögliches Vorgehen vor.

ISO 26262

Die ISO 26262 [ISO26_11] fordert bei der Safety-Risiko-Bewertung ein stringentes Vorgehen. Sie definiert eine Risiko-Matrix mit drei fest definierten Einflussfaktoren. Anhand dieser Einfluss-faktoren werden die Sicherheitsanforderungsstufen, welche die [ISO26_11] als Automotive Safety Integrity Level (ASIL) bezeichnet, festgelegt. Bei den Risikoklassen ASIL A bis ASIL D sind die

entsprechenden Anforderungen der [ISO26_11] bei der Umsetzung der Safety Goals einzuhalten. Außer den ASILs enthält die Risiko-Matrix auch noch die Felder „QM“. Diese sind keine Sicherheitsanforderungsstufen, sondern QM steht für Qualitätsmanagement und drückt aus, dass bei der Realisierung der definierten Maßnahmen ein Qualitätsmanagementsystem, wie beispielsweise die Forderungen der ISO/TS 16949 [VDA_04] und/oder der ISO 15504 (SPICE) [ISO15_06], einzuhalten sind. Das heißt, dass in diesem Fall keine Sicherheits-Anforderungen (Safety Integrity Requirements) von der [ISO26_11] vorgeschrieben sind.

Die nachfolgende Abbildung 43 zeigt die Risiko-Matrix der ISO 26262 mit den Einflussfaktoren und den Zuordnungen der ASILs bzw. QM. Die drei zu verwendenden Einflussfaktoren stehen für:

- Beherrschbarkeit (Controllability), welche angibt, wie gut ein Verletzen des Safety Goals durch den User beherrscht werden kann.
- Schwere (Severity), welche die Schwere der Auswirkung angibt, die durch die Verletzung des Sicherheitszieles entsteht.
- Exposition (Exposure), welche die Wahrscheinlichkeit/Häufigkeit des Aufenthalts in einer Fahr- oder Betriebssituation angibt, in der eine Gefährdung von beteiligten Personen, wie Fahrer, Insassen oder weitere Verkehrsteilnehmer auftreten kann.

ASIL Determination				
		C1	**C2**	**C3**
S1	**E1**	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	**E1**	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	**E1**	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Abbildung 43: Risiko-Matrix der ISO 26262 [ISO26_11]

Die Controllability ist in der Risiko-Matrix waagrecht aufgetragen und in drei Stufen unterteilt, das sind die Controllability des Fahrers, die Severity (Schwere) und in die Exposure. Die Controlability ist wiederum in vier weitere Kategorien unterteilt, das sind:

- C0: Im Allgemeinen beherrschbar
- C1: Einfach beherrschbar
- C2: Normalerweise beherrschbar
- C3: Schwierig oder nicht beherrschbar

Die Severity, also die Schwere des Unfalls, ist in der Risiko-Matrix senkrecht aufgetragen und in vier Stufen eingeteilt:

- S0: Keine Verletzungen
- S1: Leichte bis mittlere Verletzungen
- S2: Schwere Verletzungen, Überleben wahrscheinlich
- S3: Lebensgefährliche Verletzungen, Überleben unwahrscheinlich

Die höchste Klasse S3 geht dabei von lebensbedrohlichen Verletzungen mit niedriger Überlebenschance von mindestens einer Person aus. Diese Klassifizierung entspricht der Abbriviated Injury Scale (AIS) der Medizin.

Die fünfstufige Exposure ist in der Risiko-Matrix ebenfalls senkrecht aufgetragen und den einzelnen Stufen der Severity untergliedert:

- E0: Unvorstellbar (Incredible)
- E1: Sehr niedrige Wahrscheinlichkeit (Very low probability)
- E2: Niedrige Wahrscheinlichkeit (Low probability)
- E3: Mittlere Wahrscheinlichkeit (Medium probability)
- E4: Hohe Wahrscheinlichkeit (High probability)

Zusammenfassend wird laut ISO 26262 zur Ableitung des ASILs also die Schwere S (Severity) eines potentiellen Unfalls bewertet, sowie die Wahrscheinlichkeit, dass ein Ausfall zu diesem Unfall führt. Die Wahrscheinlichkeit setzt sich aus den beiden Komponenten C (Controllability) und E (Exposure) zusammen.

IEC 61508 und Alternativen

Die IEC 61508 fordert kein bestimmtes Vorgehen zur Bestimmung des Safety Integrity Levels (SIL). Im Teil 5 der Norm werden jedoch informativ mehrere Methoden zur SIL-Bestimmung angegeben, die je nach Situation angewendet werden können. Dabei sind qualitative wie auch quantitative Methoden zulässig.

Da sich diese Arbeit mit Software-lastigen Systemen auseinandersetzt, soll hier noch auf zwei Methoden eingegangen werden, die insbesondere für diese Systeme anwendbar sind. Es handelt sich um die Vorschläge aus dem MIL-STD 882E [MIL_05] und der Norm ED-153 [ED_153].

Software-lastige Systeme sind generell anders zu behandeln als Hardware-lastige Systeme, da Software lediglich aufgrund von systematischen Fehlern ausfällt (siehe Kapitel 4.1.3). Damit ist das Bestimmen von Ausfallwahrscheinlichkeiten wesentlich diffiziler. Das Risiko, welches mit Software-lastigen Systemen verbunden ist, lässt sich nicht mit Hilfe von Wahrscheinlichkeitsrechnungen kalkulieren, da Software-Ausfälle schwer bis gar nicht quantifizierbar sind. Das wird auch vom MIL-STD 882E [MIL_05, S.14] so gesehen, wo folgende Aussage zu finden ist:

> "The assessment of risk for software, and consequently software-controlled or software-intensive systems, cannot rely solely on the risk severity and

> probability. Determining the probability of failure of a single software function is difficult at best and cannot be based on historical data."[72]

Aus diesem Grund schlägt der MIL-STD 882E [MIL_05, S.14ff] auch eine alternative Möglichkeit das Risiko von Software zu betrachten vor, die nicht auf der Ermittlung von Wahrscheinlichkeiten eines Softwareausfalls beruht. Anstatt dessen wird ermittelt, wie schwerwiegend die Folgen eines Software-Hazards sein können und wie „stark", i.e. autonom der Beitrag der Software zu dem unerwünschten Ereignis ist: „Therefore, another approach shall be used for the assessment of software's contributions to system risk that considers the potential risk severity and the degree of control that software exercises over the hardware."[73]

Dieser „degree of control" der Software wird in fünf „Software Control Categories" unterteilt: Der höchste (5) bedeutet, dass die Software völlig autonom handelt, der niedrigste (1) dass die Software keinen Einfluss hat. Die nachfolgende Abbildung zeigt die Software Control Categories aus dem [MIL_05].

[72] „Die Beurteilung des Risikos für Software und somit softwaregesteuerter oder softwareintensiver Systeme können nicht alleine durch die Schwere und die Wahrscheinlichkeit des Risikos bewertet werden. Die Bestimmung der Wahrscheinlichkeit eines Ausfalls einer einzelnen Software-Funktion ist bestenfalls schwierig und kann nicht auf historischen Daten beruhen." (Übersetzung des Verfassers)

[73] „Daher wird ein anderer Ansatz für die Bewertung des System-Risikos, mit Software, verwendet, dabei wird die potentielle Schwere des Risikos und der Grad der Kontrolle die die Software über die Hardware ausübt." (Übersetzung des Verfassers)

SOFTWARE CONTROL CATEGORIES		
Level	**Name**	**Description**
1	Autonomous (AT)	• Software functionality that exercises autonomous control authority over potentially safety-significant hardware systems, subsystems or components without the possibility of predetermined safe detection and intervention by control entity to preclude the occurence of a mishap or harzard. (This definition includes complex system/software functionality with multiple subsytems, interaction parallel processors, multiple interfaces, and safety-critical functions that are time critical.)
2	Semi-Autonomous (SAT)	• Software functionality that exercises control autohority over potentially safety-significant hardware systems, subsytems or components, allowing time for predetermined safe detection and intervention by independent safety mechanism to mitigate or control the mishap or hazard. (This definition includes the control of moderately complex system/software functionality, no parallel processing, or few interfaces, but other safety systems/mechanisms can partially mitigate. System and software fault detection and annunciation notifies the control entity of the need for required safety actions.)
		• Software item that displays safety-significant information requiring immediate operator entity to execute a predetermined action for mitigation or control over a mishap or hazard. Software exception, failure, fault or delay will allow, or fail to prevent, mishap occurence. (This definition assumes that the safety-critical display information may be time-critical, but the time available does not exceed the time reqired for adequate control entity response and hazard control.)
3	Redundant Fault Tolerant (RFT)	• Software functionality that issues commands over safety-significant hardware systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. (This definition assumes that there is adequate fault detection, annunciation, tolerance, and system recovery to prevent the hazard occurence if software fails, malfunctions, or degrades. Ther are redundant sources of safety-significant information, and mitigating functionality can respond within any time-critical period.)
4	Influential	• Software that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection and display.
		• Software generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.
5	No Safety Impact (NSI)	• Software functionality that does not posses comman or control authority over safety-significant hardware systems, subsystems or components and does not provide safety-significant information. Software does not provide safety-significant or time sensitive data or information that requires control entity interation. Software does nocht transport or resolve communication or safety-significant or time sensitive data.

Abbildung 44: Software Control Categories aus dem MIL-STD 882E [MIL_05, S.15]

Aus der Matrix Severity vs. Software Control Category ergibt sich der Software Criticality Index (SwCI) (siehe Abbildung 45). Die SwCIs spiegeln somit die Kritikalität der Software wider. Die Matrix ist jedoch nicht mit einer Risikomatrix zu verwechseln, in der sich akzeptierbare und nicht akzeptierbare Bereiche befinden. Die Matrix sagt lediglich, wie kritisch die Software ist, und damit wie verlässlich sie sein muss: Je kritischer die Software, desto verlässlicher muss sie sein. Das heißt gleichzeitig, dass kritische Software besonders rigoros entwickelt und getestet werden muss. Dies ist analog zum Konzept der SILs (auf Software bezogen).

SOFTWARE SAFETY CRITICALITY MATRIX				
	SEVERITY CATEGORY			
SOFTWARE CONTROL CATEGORY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SwCI 1	SwCI 1	SwCI 3	SwCI 4
2	SwCI 1	SwCI 2	SwCI 3	SwCI 4
3	SwCI 2	SwCI 3	SwCI 4	SwCI 4
4	SwCI 3	SwCI 4	SwCI 4	SwCI 4
5	SwCI 5	SwCI 5	SwCI 5	SwCI 5

SwCI	Level of Rigor Tasks
SwCI 1	Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing.
SwCI 2	Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing.
SwCI 3	Program shall perform analysis of requirements and architecture; and conduct in-depth safety-specific testing.
SwCI 4	Program shall conduct safety-specific testing.
SwCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

Abbildung 45: Software Safety Criticality Matrix des MIL-STD 882E [MIL-STD 882E]

Für jeden Software Criticality Index müssen nun genaue Vorgaben spezifiziert werden, die bei der Entwicklung der Software eingehalten werden müssen – das ist genau das, was der SIL bei der Softwarenentwicklung vorgibt.

Einen ähnlichen Weg geht auch der Standard ED153 [ED_153], der den „SWAL“ (Software Assurance Level), der auch wieder ein ähnliches Konzept wie der SIL ist, aus Severity und dem Beitrag der Software zum unerwünschten Effekt ermittelt (siehe Abbildung 46). Hier wird einerseits - wie oben - der potentielle Schaden, den die Software verursachen kann, evaluiert (Severity, hier in vier Klassen eingeteilt von 1 bis 4). Auf der anderen Seite wird evaluiert, wie wahrscheinlich es ist, dass bei Auftreten des Softwarefehlers dieser Schaden eintritt. Es sei bemerkt, dass dies nichts mit der Wahrscheinlichkeit des Auftretens des Softwarefehlers selbst zu tun hat, sondern die

Wahrscheinlichkeit des unerwünschten Effekts unter der Bedingung des Auftretens evaluiert wird („likelihood of generating such an effect“).

Effect Severity Class / Likelihood of generating such an effect (Pe x Ph)	1	2	3	4
Very Possible	SWAL1	SWAL2	SWAL3	SWAL4
Possible	SWAL2	SWAL3	SWAL3	SWAL4
Very Unlikely	SWAL3	SWAL3	SWAL4	SWAL4
Extremely Unlikely	SWAL4	SWAL4	SWAL4	SWAL4

Abbildung 46: SWAL nach ED-153 [ED_153]

Die (informativen) Methoden zur SIL-Ermittlung der IEC 61508 (Teil 5) sind allgemein gehalten und deshalb eben nicht explizit für Softwaresysteme konzipiert. Es wird deshalb in dieser Arbeit vorgeschlagen, den Weg der oben genannten Normen [MIL_05] und [ED_153] zur Ermittlung des Beitrag der Software zum unerwünschten Ereignis zu gehen, um darauf basierend einen SIL abzuleiten. Eine solche Möglichkeit SILs abzuleiten ist in der nachfolgenden Abbildung 47 dargestellt.

Likelihood of generating such an effect / Effect of Software Failure	Catastrophic	Critical	Marginal	Negligible
Very Possible	SIL4	SIL3	SIL2	SIL1
Possible	SIL3	SIL2	SIL2	SIL1
Very Unlikely	SIL2	SIL2	SIL1	SIL1
Extremely Unlikely	SIL1	SIL1	SIL1	SIL1

Abbildung 47: Möglichkeit der SIL-Ableitung

Nach der Bewertung der Sicherheitsanforderungsstufe wird das weitere Vorgehen, wie schon eingehend beschrieben, einheitlich fortgesetzt. Das heißt, die nächsten Schritte stellen die Definition des Sicherheitszieles und die Ableitung der Safety Requirements dar.

4.7 Safety Case Pattern

Die IEC 61508 verlangt explizit keinen Safety Case, sondern einen Konformitätsnachweis: „To conform to this standard it shall be demonstrated that all the relevant requirements have been satisfied to the required criteria specified (for example safety integrity level) and therefore, for each clause or subclause, all the objectives have been met." Das heißt, in einem Projekt muss dargelegt werden, dass die Anforderungen den festgelegten Kriterien entsprechen und damit die Ziele jedes Abschnitts oder Unterabschnitt erreicht werden können.

Die ISO 26262 [ISO26_11, Teil 3, Punkt 1.106] fordert hingegen ausdrücklich einen Safety Case – „argument that the safety requirements for an item (1.69) are complete and satisfied by evidence compiled from work products of the safety activities during development". Das heißt, die ISO 26262 verlangt als Sicherheitsnachweis ein *Argument*, dass die definierten Safety Requirements eines Items komplett umgesetzt wurden, dafür sind die entsprechenden Arbeitsprodukte der Safety-Aktivitäten über die gesamte Entwicklung als *Beweis* zu liefern.

Es liegen also in der Literatur zwei völlig unterschiedliche Ansätze vor, um zu beweisen, dass das zu entwickelnde System ausreichend sicher ist. Um hier ein einheitliches Vorgehen zu schaffen, wird der Safety Case anhand der GSN festgelegt, die bereits in Kapitel 2.6 beschrieben wurde. Die ISO 26262 spricht beim Safety Case von einem Nachweis, dass das System (Item) keine unzumutbaren Risiken enthält und das im Zusammenhang mit jedem gefährlichen Ereignis (hazardous event), das durch eine Fehlfunktion (malfunctioning behaviour) des Items ausgelöst wird. Dies wird erreicht durch die Definition von Safety Goals, welche über die Safety Requirements und Safety-Maßnahmen die unzumutbaren Risiken vermeiden. Die Norm spricht hier lediglich von endogenen Hazards und bezieht exogene Hazards nicht ein. Der Fokus sollte jedoch auf der Beurteilung von endogenen und exogenen Hazards liegen, wobei die Argumentationsstruktur so aufgebaut ist, dass sie in jedem weiteren Entwicklungsprojekt wiederverwendet werden kann.

Im Folgenden wird ein Template (Vorlage) für einen Safety Case (Sicherheitsnachweis) vorgestellt. Die darin vorhandene Safety Case Argumentation ist in der Goal Structuring Notation (GSN) (siehe Kapitel 2.6 „Safety Case") formuliert. Die Vorlage enthält verschiedene Safety Case Pattern, also Mustervorlagen, die in den jeweiligen Phasen der PPL des jeweiligen Projekts angewendet werden können und nach dem Safety-Vorgehensmodell strukturiert sind.

Die gesamte Vorlage stellt eine Argumentation dar, die zeigen soll, dass ein definiertes System für eine gegebene Anwendung in seiner definierten System-Umgebung und seinem System-Umfeld ausreichend sicher ist. Die Vorlage hat zumindest zwei Zwecke:

- Die Vorlage stellt ein generisches deduktives Argument dar, welches die Wahrheit obiger Aussage zeigen soll. Durch das Lesen der GSN-Struktur von oben nach unten, d.h. vom Top-Level Goal G0 zu den einzelnen Solutions an den „Blättern" der Baumstruktur, wird das Argument für den Leser nachvollziehbar.
- Die Vorlage zeigt an ihren „Blättern", den Solutions, welche Arbeitsprodukte und Analyse-Artefakte nötig sind, um die GSN zu vervollständigen. Die Solutions wie auch die Context-Elemente sind in der Regel Dokumente oder Teile von Dokumenten, die die entsprechenden

Informationen enthalten. Das Safety-Vorgehensmodell (siehe Kapitel 3) ist so strukturiert, dass sichergestellt ist, dass die benötigten Arbeitsprodukte für die Solutions und Context-Elemente vorhanden sind.

Die Vorlage ist dazu gedacht, unverändert für ein beliebiges System übernommen zu werden. In Einzelfällen kann es aber nötig und sinnvoll sein, sie zu adaptieren oder zu erweitern. Im Folgenden wird der in Abbildung 48 dargestellte Aufbau des Top-Level Patterns, also die GSN-Struktur auf Top-Level Eben, d.h. im Problemraum, beschrieben.

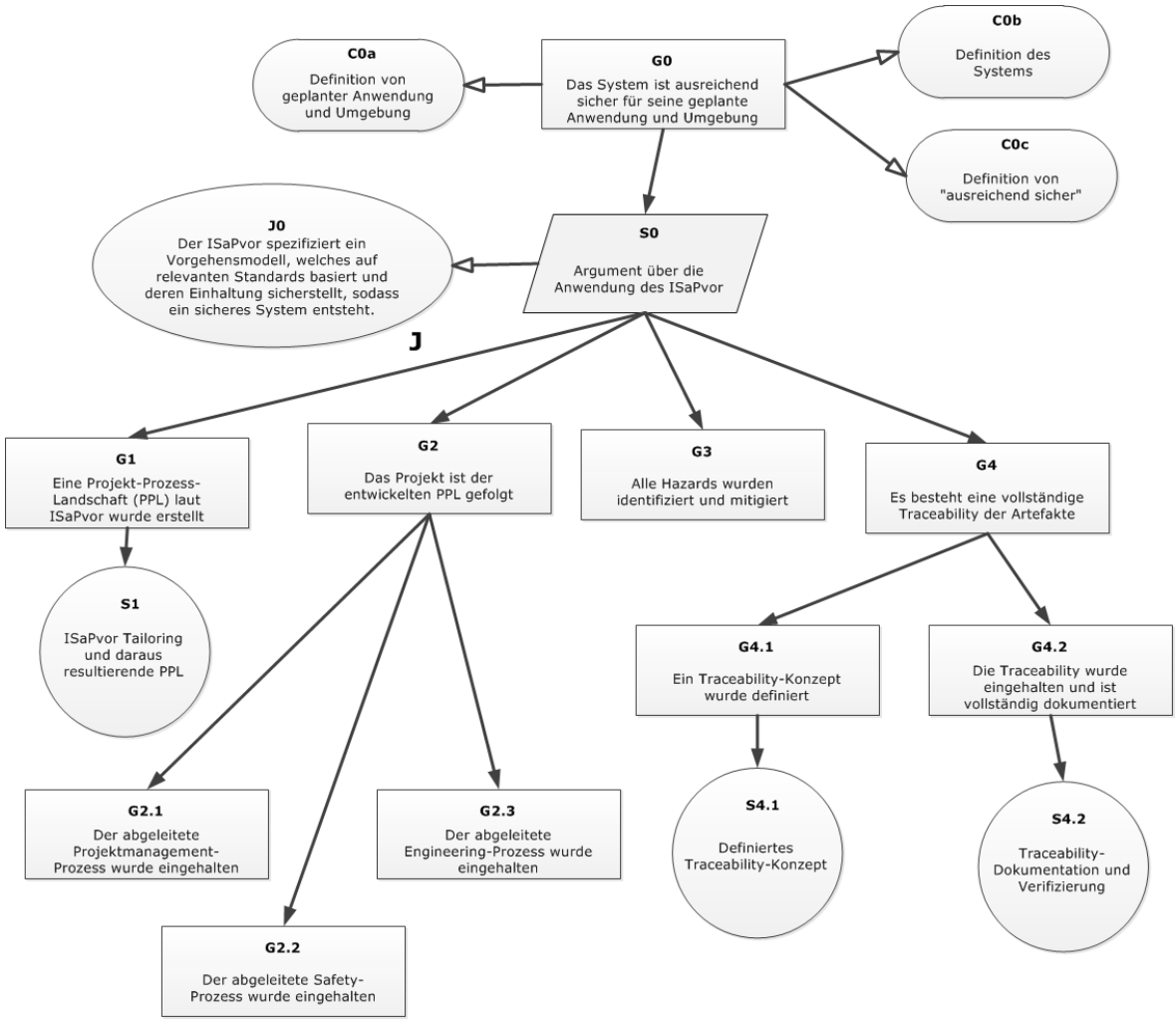


Abbildung 48: Top-Level Safety Case Pattern

Das Argument beginnt mit der zu beweisenden Aussage (G0), nämlich dass ein gegebenes System für seine geplante Anwendung und Umgebung ausreichend sicher ist. Um die Überprüfbarkeit dieser Aussage verifizieren zu können, müssen das System, die geplante Anwendung und Umgebung klar definiert sein. Diese Context-Informationen liefert das Shell-Modell des Safety-Vorgehensmodells, siehe dazu Kapitel 4.2 „Shell-Modell“. Aus diesem Grund stellt das Safety-Vorgehensmodell sicher, dass diese Definitionen schon zu Beginn verfügbar sind.

Die gesamte Argumentationsstrategie (S0) beruht auf der Justification (J0) die begründet, dass durch die Anwendung des Safety-Vorgehensmodells die benötigten Artefakte für die Solutions und Context-Elemente vorhanden sind. Das Safety-Vorgehensmodell stellt darüber hinaus sicher, dass ein sicherheitskritisches System korrekt entwickelt wird, was in dieser Arbeit begründet wird. Es ist eine bewährte Strategie, sowohl direkte als auch indirekte Hinweise für die Sicherheit eines Systems zu identifizieren: Indirekte Hinweise beruhen meist auf einem korrekt eingehaltenen Prozess bei der Systementwicklung, direkte Hinweise sind vom entwickelten System direkt gewonnene Aussagen. So können die folgenden Sub-Goals G1 und G2 als indirekte Aussagen, die Sub-Goals G3 und G4 als direkte Aussagen über das System klassifiziert werden.

G1 und G2 beziehen sich also auf den Prozess. Wie in Kapitel 3 beschrieben, muss vorerst der korrekte Prozess definiert werden (G1), und dann dieser in all seinen Ausprägungen befolgt werden (G2, bzw. G2.1, G2.2, G2.3).

G3 und G4 beziehen sich auf das entwickelte System selbst, nämlich einerseits auf die Identifikation und Behandlung aller (soweit das möglich ist) Hazards, und andererseits auf eine vollständig vorhandene Traceability, um die Nachvollziehbarkeit und Überprüfbarkeit zu gewährleisten.

C0a: intended environment: kommt aus Anwendung des Shell-Modells

C0b: Systemdefinition kommt aus Systemabgrenzung

C0c: adequately safe kommt aus Hazard-Analyse und safety goals

J0: Justification ist die vorliegende Arbeit

G1/G2 sind sozusagen der „indirekte“ Teil der Evidences

G3/G4 ist der direkte Teil der Evidences

5. Anwendungsbeispiel: Side-Stand

Damit das bisher theoretisch beschriebene systematische Vorgehen besser veranschaulicht werden kann, soll als Anwendungsbeispiel ein einfaches System entwickelt werden. Es wurde bewusst ein sehr einfaches Beispiel gewählt, um einerseits das systematisch-methodische Vorgehen nach dem Safety-Vorgehensmodells besser demonstrieren zu können und andererseits die erforderlichen Aufwände für Entwicklungen im sicherheitsrelevanten Bereich eindeutig sichtbar zu machen. Darüber hinaus wäre die Darstellung eines komplexeren Beispiels ohne geeignetes Konfigurations-management-Tool hier auch nicht möglich, mit einem Tool wäre die Systematik wiederum nicht eindeutig sichtbar. Anhand dieses Beispiels soll ersichtlich werden, dass für ein „Inherent Safe System Design“ und dessen Realisierung ein derartiges Safety-Vorgehensmodell unumgänglich ist.

Es soll eine Elektronik für einen elektrischen Seitenständer für Motorräder, in dieser Arbeit als „Side-Stand“ betitelt, entwickelt werden. Die zu entwickelnde Elektronik, also das System Under Control (SUC) ist das „Side-Stand Control System (SSCS)“. Da es keine eigene Safety-Norm für Motorräder gibt, soll die ISO 26262 dafür herangezogen werden. Auch wenn Motorräder nicht in die Zuständigkeit der ISO 26262 gehören, wurde dieses Beispiel bewusst gewählt, um das Vorgehen an einen verallgemeinerten Anwendungsfall verifizieren zu können.

Das SSCS, bestehend aus Hardware und Software, und wird nach dem Safety-Vorgehensmodell, wie im Kapitel 3 beschrieben, entwickelt. Damit soll gezeigt werden, dass das detailliert skizzierte Safety-Vorgehensmodell vielfältig anwendbar ist. Anfangs existiert nur eine sehr rudimentäre Systemidee, wie in folgender Abbildung 49 dargestellt:

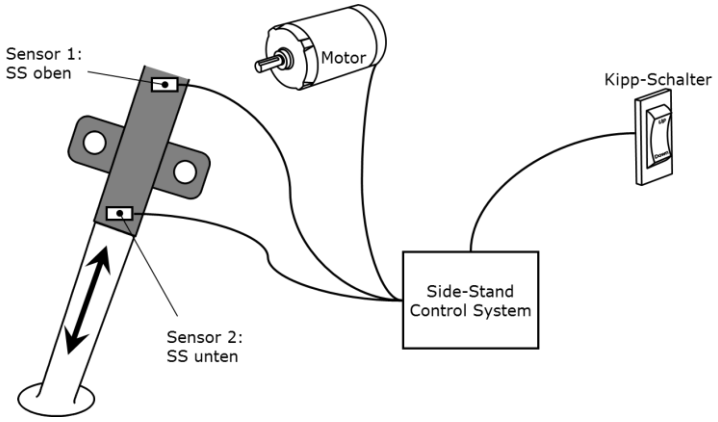


Abbildung 49: Projektidee des elektrisch ausfahrbaren Side-Stands

Der Side-Stand soll mit einem einfachen Kipp-Schalter, welcher am Lenker des Motorrads befestigt wird, über einen Motor elektrisch ein- und ausgefahren werden können. Zwei Sensoren (S1 und S2) signalisieren die Endpositionen, „Side-Stand oben“ und „Side-Stand unten“. Gesteuert wird der Side-Stand über das Side-Stand Control System (SSCS). Um die Lesbarkeit der Spezifikationstabellen und Abbildungen zu erleichtern werden in den nachfolgenden Kapiteln das Motorrad mit MR und der Side-Stand mit SS abgekürzt.

Damit die Ablaufstruktur besser nachvollziehbar ist, werden in jedem Prozess die einzelnen Prozessschritte nochmals tabellarisch dargestellt. Darüber hinaus wird der Ablauf der jeweilig behandelten Basis-Aktivität in der Überschriften dargestellt.

5.1 Side-Stand: Problemraum – Makro-Konzept Entwicklung

Ziel des Problemraumes ist die Erstellung des Makro-Konzepts, welches den ganzheitlichen Projektumfang in der Vor-Projektphase darstellt. Der Problemraum soll einen ganzheitlichen Überblick über das Projekt-Vorhaben schaffen.

In den ersten Schritten wird nach dem Prozess der Projekt-Initialisierung (siehe Kapitel 3.1.1) der Projekt-Umfang im Scope of Work (SoW) beschrieben, die Projekt-Kontext-Analyse (PKA) durchgeführt und die Projekt-Prozesslandschaft (PPL) definiert. Im nächsten Schritt wird, basierend auf den Ergebnissen der Projekt-Initialisierung das technische Konzept (siehe Kapitel 3.1.2) erstellt. Dabei wird das zu entwickelnden Systems in seiner Systemumgebung anhand des Shell-Modells dargestellt und darauf basierend die Top-Level Requirements (TLR) identifiziert. Die Ergebnisse liefern die Basis für die Preliminary Hazard-Identifikation (siehe Kapitel 3.1.3). Dabei werden die Top-Level Hazards (TLHa) identifiziert, Hazards im Sinne der Hazard Avoidance soweit als möglich vermieden. Für die noch vorhandenen Hazards wird das Safety-Risiko-Assessment durchgeführt und damit der vorläufige, im Fall der ISO 26262, ASIL (Automotive Safety Integrity Level) eingeschätzt. Basierend darauf werden die Safety Goals definiert sowie die Safety Requirements spezifiziert. Die Safety Requirements, wenn erforderlich, haben nun direkten Einfluss auf die Konzeptionierung. Darauf aufbauend müssen Safety-Maßnahmen, im Sinne der Hazard-Prevention bzw. der des Hazard-Controllings gesetzt werden. Das können zusätzliche technische Lösungen im System-Entwurf bedeuten. Die Ergebnisse können aber auch auf die vorläufigen Planungsmaßnahmen in der Projekt-Initialisierung Einfluss nehmen.

Das heißt, dass ein Durchlauf oder mehrere Durchläufe der drei Prozesse erforderlich sein können, wie in

Abbildung 26 im Kapitel 3.1.4 „Zusammenfassung des Problemraums“ dargestellt. Basierend auf den Ergebnissen des Problemraums wird im Stage Gate 1 die Entscheidung über die Durchführung oder Nicht-Durchführung des Projekts getroffen.

5.1.1 Side-Stand: Projekt-Initialisierung

Die Projekt-Initialisierung ist der Ausgangspunkt für die Entwicklung des elektrisch ausfahrbaren Side-Stands (SS). Beginnend mit dem Scope of Work (SoW) [PjI_BA1] wird das Projektvorhaben genau beschrieben. Darauf basierend wird die Projekt-Kontext-Analyse durchgeführt [PjI_BA2] und die Projekt-Prozesslandschaft (PPL) erstellt [PjI_BA3] (siehe dazu Kapitel 3.1.1). Mit dem SoW, der PKA und der PPL kann die Konzeptionierung und danach die PHI [PjI_BA4] beauftragt werden. Im Zuge der Erarbeitung des Konzepts und der darauffolgenden Durchführung der Safety-Analysen in der PHI kann es vorkommen, dass das SoW und die PPL sowie auch der technische Entwurf immer wieder ergänzt und angepasst werden müssen. Mit den abgeschlossenen Verfeinerungen erfolgt die Fertigstellung des organisatorischen Konzepts und das Output-Dokument des Problemraums, also das Makro-Konzept, kann erstellt und freigegeben werden [PjI_BA6]. Die nachfolgende Tabelle 15 gibt einen Gesamtüberblick über die 6 Basis-Aktivitäten, die in der Projekt-Initialisierung gefordert sind.

PjI_BA1	Scope of Work (SoW) definieren
PjI_BA2	Projekt-Kontext-Analyse (PKA) durchführen
PjI_BA3	Erstansatz der Projekt-Prozesslandschaft (PPL) erstellen
PjI_BA4	Konzeptionierung und PHI beauftragen/veranlassen
PjI_BA5	Organisatorisches Konzept erstellen
PjI_BA6	Makro-Konzept erstellen und freigeben

Tabelle 15: Basis-Aktivitäten der Projekt-Initialisierung im SSCS

Scope of Work (SoW) definieren [PjI_BA1]

Die erste Basis-Aktivität der Projekt-Initialisierung [PjI_BA1] besteht aus 7 Teilschritten:

1. Projektbeschreibung erstellen [PjI_BA1.1]
2. Projekt-Zweck definieren [Pj_BA1.2]
3. Operational Requirements spezifizieren [PjI_BA1.3]
4. Definierte(s) Lieferobjekte definieren [PjI_BA1.4]
5. Projekt-Ziele definieren [PjI_BA1.5]
6. Projekt-Nicht-Ziele definieren [PjI_BA1.6]
7. Hauptaufgaben im Projekt festlegen [PjI_BA1.7]

In der nachfolgenden Abbildung 50 ist das SoW für den elektrisch ausfahrbaren Side-Stand dargestellt. Darin wird im ersten Teilschritt das Projektvorhaben [PjI_BA1.1] beschrieben. In diesem Fall handelt es sich um eine Neu-Entwicklung einer Side-Stand Elektronik, die aus Hard- und Software besteht. Diese soll nach den Vorgaben einer Safety- und einer Prozess-Norm entwickelt werden in diesem Fall ist das die ISO 26262 und die ISO/IEC 15504. Zweck des Projekts [PjI_BA1.2] ist die Erschließung eines neuen Marktsegments. Die betrieblichen Anforderungen

[PjI_BA1.3], auch operational Requirements genannt, liefern mit „Side-Stand einfahren“ und „Side-Stand ausfahren“ die Basis-Funktionen. Das Lieferobjekt [PjI_BA1.4], das am Ende des Projekts abgeliefert werden soll, ist die Side-Stand Elektronik, mit Side-Stand Control System (SSCS) bezeichnet.

Die Projekt-Ziele [PjI_BA1.5] legen fest, was am Ende des Projekts erreicht werden soll und geben damit die Entscheidungsgrundlage für die Planung und Durchführung vor. Ziel ist auch der Nachweis der Normkonformität nach der ISO 26262 und der ISO/IEC 15504. Die Nicht-Ziele [PjI_BA1.6] stellen klar, das weder ein Gehäuse noch die Mechanik, Teil des Projektvorhabens sind. Von den Projekt-Zielen werden die Hauptaufgaben [PjI_BA1.7] des Projekts abgeleitet, welche die Basis für die PPL bilden und damit auch die Phasen des Projektstrukturplans vorgeben.

<table>
<tr><th colspan="2">Scope of Work</th></tr>
<tr><td colspan="2">In diesem Projekt findet eine Neu-Entwicklung einer Elektronik (Hardware und Software) für einen elektrisch aus- und einfahrbaren Side-Stand für Motorräder statt. Bei der Entwicklung wird die Einhaltung der ISO 26262 und die Konformität zur ISO/IEC 15504, nach SPICE Level 2, gefordert.

Projektzweck:
Erschließung eines neuen Marktsegmentes.

Betriebliche Anforderungen / Operationale Requirements:
Der Side-Stand soll mit einem Kipp-Schalter, der sich am Motorradgriff linker Hand befindet, aus- und eingefahren werden können.

Lieferobjekt:
Side-Stand Control System (SSCS), bestehend aus Hardware und Software.</td></tr>
<tr><td>Projektstarttermin: n.a.</td><td>Projektendtermin: n.a.</td></tr>
<tr><td>Projektziele:
> SSCS für elektrisch ausfahrbaren Side-Stand für Motorräder ist entwickelt
> Normkonformität ist hergestellt
o ISO 26262
o ISO/IEC 15504 (SPICE-Level 2)</td><td>Nicht-Projektziele:
> Mechanische Entwicklung
> Gehäuse der Elektronik</td></tr>
<tr><td>Hauptaufgaben:
1. Projektmanagement
2. System Requirements
3. System Design
4. Hardware-Entwicklung
5. Software-Entwicklung
6. Test
7. ISO 26262 Audit
8. SPICE-Level 2 Assessment</td><td>Projekt-Aufwände:
n.a.

Projekt-Kosten:
n.a.</td></tr>
<tr><td>Projekt-Risiken:
n.a.</td><td></td></tr>
</table>

Abbildung 50: Scope of Work des SSCS im Problemraum (Status 1st draft)

In diesem Beispiel werden die Projekt-Aufwände und Projekt-Kosten sowie die Projekt-Risiken nicht bearbeitet, da sie, wie schon im Kapitel 3.1.1 erwähnt, als State-of-the-Art Aktivitäten betrachtet werden und auch keinen wesentlichen Beitrag im Vorgehen leisten.

Projekt-Kontext-Analyse (PKA) durchführen [PjI_BA2]

Der Prozessschritt der PKA strukturiert das Vorgehen in 7 Teilschritten:

1. Projekt-Inputs identifizieren [PjI_BA2.1]
2. Nach-Projektphase analysieren [Pj_BA2.2]
3. Andere Projekte im Unternehmen analysieren [PjI_BA2.3]
4. Safety-Management im Unternehmen und Safety-Kultur darstellen [PjI_BA2.4]
5. Qualitätsmanagementsystem darstellen [PjI_BA2.5]
6. Identifikation der wichtigsten Stakeholder [PjI_BA2.6]
7. Zuständige Behörden, Regulatoren und Zertifizierungseinrichtungen identifizieren [PjI_BA2.7]

Da es sich beim SSCS um ein fiktives Beispiel handelt, kann die Projekt-Kontext-Analyse nur bedingt demonstriert werden. Als Projekt-Inputs [PjI_BA2.1] wurde zuvor die *Hazard-Checkliste* (siehe Anhang A) identifiziert. Die Nach-Projektphase [PjI_BA2.2] bezieht sich lediglich auf die Fertigung des SSCS, die Integration mit der Mechanik und die Montage sowie die Maintenance-Aktivitäten der Elektronik. Die Unternehmenskultur [PjI_BA2.4] mit der von der ISO 26262 geforderten Safety-Kultur und dem Qualitätsmanagementsystem wird als gegeben angenommen und in der Projektdarstellung entsprechend berücksichtigt. Die restlichen Punkte „andere Projekte“ [PjI_BA2.3] und „Qualitätsmanagement Systeme betrachten“ [PjI_BA2.5] sowie die „Identifikation der zuständigen Behörden“ [PjI_BA2.7] fließen in die Projekt-Kontext-Analyse nicht mit ein, da sie in diesem fiktiven Beispiel keine Anwendung finden. Die „Identifikation der Stakeholder“ [PjI_BA2.6] wird, da es sich hier auch ausschließlich um beispielhafte Annahmen handelt, in die Konzeptionierung verlegt.

Erstansatz der Projekt-Prozesslandschaft (PPL) erstellen [PjI_BA3]

Die Basis-Aktivität drei zur Erstellung der PPL [PjI_BA3] ist in 5 Teilschritte unterteilt:

1. Prozesse von den Hauptaufgaben des SoW ableiten und in einem V-Modell darstellen [PjI_BA3.1]
2. Engineering Lifecycle vom V-Modell ableiten [Pj_BA3.2]
3. System Safety Lifecycle darstellen und Safety-Prozesse den Engineering-Prozessen zuordnen [PjI_BA3.3]
4. Support-Prozesse in die PPL einbinden [PjI_BA3.4]
5. Projektmanagement-Lifecycle einbinden [PjI_BA3.5]

Die Projekt-Prozesslandschaft (PPL) wird auf Basis der definierten Hauptaufgaben [PjI_BA3.1] im Projekt, d.h. aus dem Scope of Work (SoW), hergeleitet. Das sind in diesem Fall die Ermittlung der System Requirements, das Erstellen des System Design, die Hardware- und Software-Entwicklung und die verschiedenen Tests. Für jede Hauptaufgabe werden die entsprechenden Prozesse aus dem

ISaPro® herangezogen und in einem V-Modell dargestellt. Findet sich für bestimmte Hauptaufgaben kein Prozess, so werden dafür fiktive Prozesse, wie schon in Kapitel 3.1.1 erwähnt, angeführt.

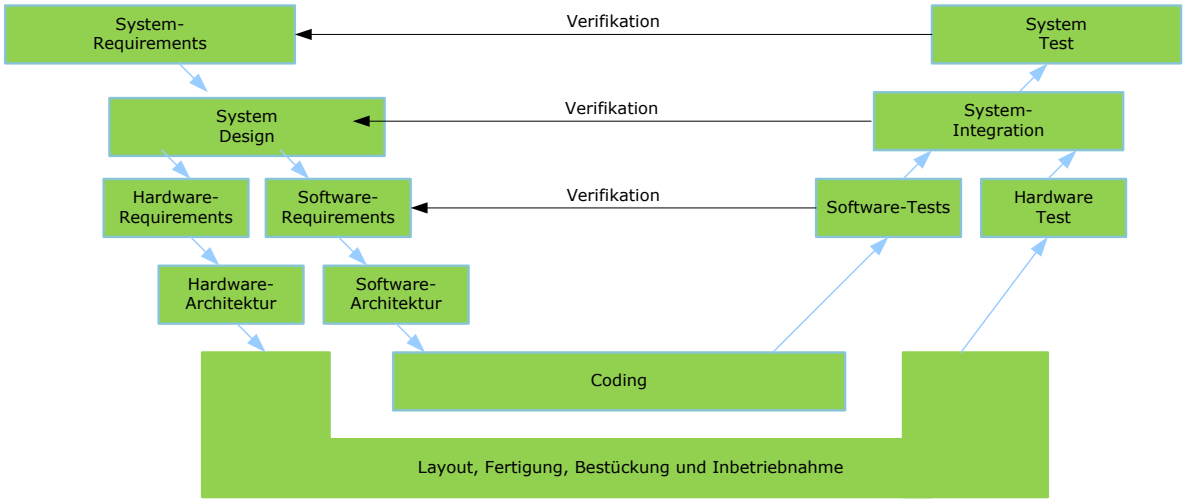


Abbildung 51: V-Modell des SSCS

Nachdem der elektronische Side-Stand auch keine definitiven Kunden hat, bilden die System Requirements den Ausgangspunkt des Vorgehens. Davon kann das System Design abgeleitet werden. Danach teilt sich der Systems-Engineering Lifecycle in die Bereiche Hardware- und Software-Engineering auf. Im Software-Engineering Lifecycle werden nun vom System Design die Software-Requirements abgeleitet, welche die Basis für die Software-Architektur bilden. Aufgrund der gering eingeschätzten Komplexität kann direkt die Implementierung erfolgen. Nach abgeschlossener Implementierung wird die Software getestet. Parallel zum Software Engineering läuft das Hardware-Engineering. Auch hier werden die Hardware-Anforderungen vom System Design abgeleitet, welche zur Entwicklung der Hardware-Architektur dienen. Darauf basierend wird die Hardware realisiert, also das Layout entwickelt, gefertigt und der gefertigte Print bestückt. Die Hardware-Inbetriebnahme passiert noch im gleichen Prozess. Der Hardware-Test Prozess schließt das Hardware-Engineering ab. Danach beginnt die Integration auf System-Ebene, d.h. es werden Hardware und Software gemeinsam in Betrieb genommen und der sogenannte Integrationstest durchgeführt, indem das Hardware-Software Interface getestet wird. Die System-Entwicklung schließt mit dem System-Test ab.

Nach der Definition des V-Modells wird dieses aufgeklappt und die Engineering-Prozesse als Engineering Lifecycle [PjI_BA3.2] dargestellt. Danach werden die Safety- [PjI_BA3.3], die Support- [PjM_BA3.4] und die Projektmanagement-Prozesse [PjM_BA3.5] integriert und als PPL dargestellt (siehe dazu Kapitel 3.1.1).

Die Prozesse der Verifikation und Validation (V&V) unterstützen die Planung der Qualitätskontrollen, wie beispielsweise der durchzuführenden Reviews und der Test-Aktivitäten.

Da die Entwicklung nach den Anforderungen der ISO/IEC 15504 nach SPICE-Level 2 erfolgen soll, wird der Prozess „Qualitätssicherung“ vorausgesetzt. Hier fordert die ISO/IEC 15504 eine

unabhängige Instanz, die überwacht, ob die für dieses Projekt definierten Prozesse eingehalten werden und die dabei geplanten Arbeitsprodukte in entsprechender Qualität vorhanden sind. Anhand der PPL ist gut ersichtlich, wie viele Arbeitsprodukte anfallen. Um die Integrität der Arbeitsprodukte herzustellen und diese aufrechtzuerhalten wird der Konfigurationsmanagement-Prozess benötigt, den auch SPICE-Level 2 fordert. Als letztes werden die Stage-Gates, Audits, Assessments und Baselines eingeplant. Die nachfolgende Abbildung 52 zeigt die Projekt-Prozesslandschaft für die Entwicklung des Side-Stand Control Systems.

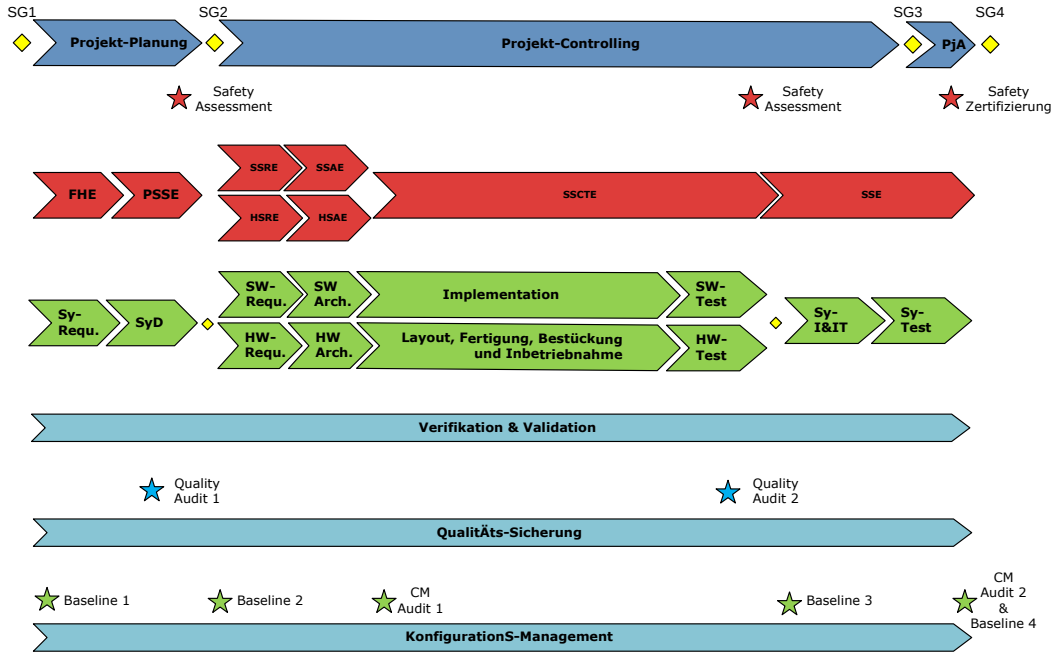


Abbildung 52: PPL des SSCS (1st draft)

Konzeptionierung und PHI beauftragen/veranlassen [PjI_BA4]

Das SoW, die Ergebnisse aus der PKA und die PPL sind wichtige Inputs für die Beauftragung der Konzeptionierung und der nachfolgenden PHI, die in den nächsten Kapiteln beschrieben werden. Die Ergebnisse dieser beiden Prozesse, also das technische Konzept und das Safety-Konzept, haben direkten Einfluss auf die Planungsaktivitäten in der Projekt-Initialisierung.

Organisatorisches Konzept erstellen [PjI_BA5]

Makro-Konzept erstellen und freigeben [PjI_BA6]

Das organisatorische Konzept wird, wie zuvor erwähnt, nach der Durchführung der Konzeptionierung und der PHI erstellt. Um die in Kapitel 3 vorgegebene Kapitelstruktur, in der Reihenfolge Projekt-Initialisierung, Konzeptionierung und PHI, auch hier einhalten zu können, wurde das Kapitel 5.1.4 „Side-Stand: Abschluss des Problemraums“ eingeführt, in dem das gesamte und endgültige Makro-Konzept [PjI_BA6] dargestellt wird.

5.1.2 Side-Stand: Konzeptionierung

In der Konzeptionierung soll nun die System-Umgebung und das System-Umfeld identifiziert, die Top-Level Requirements spezifiziert und das SSCS soweit als möglich konkretisiert werden. Die Inhalte bilden das technische Konzept. Der System-Entwurf des SSCS – welcher Teil des technischen Konzepts ist – soll zeigen, wie die Projekt-Idee technisch realisiert werden kann.

Die Konzeptionierung ist, wie schon in Kapitel 3.1.2 beschrieben, in vier grundlegende Basis-Aktivitäten untergliedert. Das beginnt mit der *Identifikation der System-Umgebung und des System-Umfeldes* [KON_BA1]. Basierend darauf werden die *Top-Level Requirements spezifiziert* [KON_BA2] und danach der *System-Entwurf entwickelt* [KON_BA3]. Die *Spezifikation und Freigabe des technischen Konzepts* [KON_BA4] schließt die Konzeptionierung ab. Die nachfolgende Tabelle 16 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die in der Konzeptionierung gefordert sind.

KON_BA1	System-Umgebung und System-Umfeld identifizieren
KON_BA2	Top-Level Requirements spezifizieren
KON_BA3	System-Entwurf entwickeln
KON_BA4	Technisches Konzept spezifizieren und freigeben

Tabelle 16: SSCS: Basis-Aktivitäten der Konzeptionierung im SSCS

Hierfür wird das Shell-Modell (siehe dazu Kapitel 4.2) herangezogen, es dient der systematischen Identifikation aller im Kontext stehenden Elemente und Eigenschaften, die auf das SSCS einwirken können. Damit sollen die Anforderungen an das SSCS nahezu lückenlos identifiziert werden.

System-Umgebung und System-Umfeld identifizieren [KON_BA1]

Die erste Basis-Aktivität der Konzeptionierung [PjI_BA1] ist in 6 Teilschritte unterteilt:

1. Systemgrenzen festlegen [KON_BA1.1]
2. Unmittelbare System-Umgebung identifizieren [KON_BA1.2]
3. Mittelbare System-Umgebung identifizieren [KON_BA1.3]
4. System-Einbettung identifizieren [KON_BA1.4]
5. System-Umwelten identifizieren [KON_BA1.5]
6. Einsatzbereich identifizieren [KON_BA1.6]

Wie in Kapitel 3.1.2 beschrieben, ist die System-Abgrenzung [KON_BA1.1] der erste und wichtigste Schritt, da darauf basierend die System-Umgebung identifiziert wird sowie die Top-Level Requirements (TLR) abgeleitet werden. Das Side-Stand Control System (SSCS) wird zunächst einmal als leeres Rechteck dargestellt, da zu diesem Zeitpunkt noch keine System-Inhalte bekannt sind. Alles was außerhalb dieses Rechtecks, also des SSCS, ist, gilt es zu identifizieren. Die unmittelbare System-Umgebung wird im nächsten Teilschritt festgelegt [KON_BA1.2]. Dabei werden die direkt angrenzenden Systeme gelistet. Zu den benachbarten Systemen (BNS) gehören

(siehe Kapitel 3.1.2) alle angrenzenden Systeme, die in *direkter Verbindung* mit dem SSCS stehen. Das sind in diesem Fall der Fahrer als menschliches benachbartes System (Me_BNS), die Sensoren als unidirektionale benachbarte Systeme (Uni_BNS) und der Motor zum Antrieb des Side-Stands als weiteres unidirektionales benachbartes System. Im Me_BNS bedient der Fahrer einen Kipp-Schalter (KS) zum Ein- und Ausfahren des Side-Stands. Damit ist die Z-Shell identifiziert

Als übergeordnetes System (O-Shell) ist in diesem Fall das Motorrad zu sehen, welches auch gleich das Gesamtsystem (O-Shell = S-Shell) darstellt, da das SSCS in keinem anderen übergeordneten System, außer dem Gesamtsystem Motorrad, eingebettet ist. Damit wird in der S-Shell die unmittelbare System-Umgebung identifiziert [KON_BA1.3], da das SSCS in keinem weiteren übergeordneten System eingebettet ist [KON_BA1.4]. Zu den wichtigsten Elementen in der mittelbaren System-Umgebung zählen der Motor, der Auspuff, die Räder und der Tank. Die aufgezählten Elemente beeinflussen das SSCS allerdings nur mittelbar, also indirekt. Die nachfolgende Abbildung 53 zeigt das Shell-Modell des SSCSs und seine unmittelbare System-Umgebung sowie dessen System-Umfeld, mit System-Umwelten und Einsatzbereich.

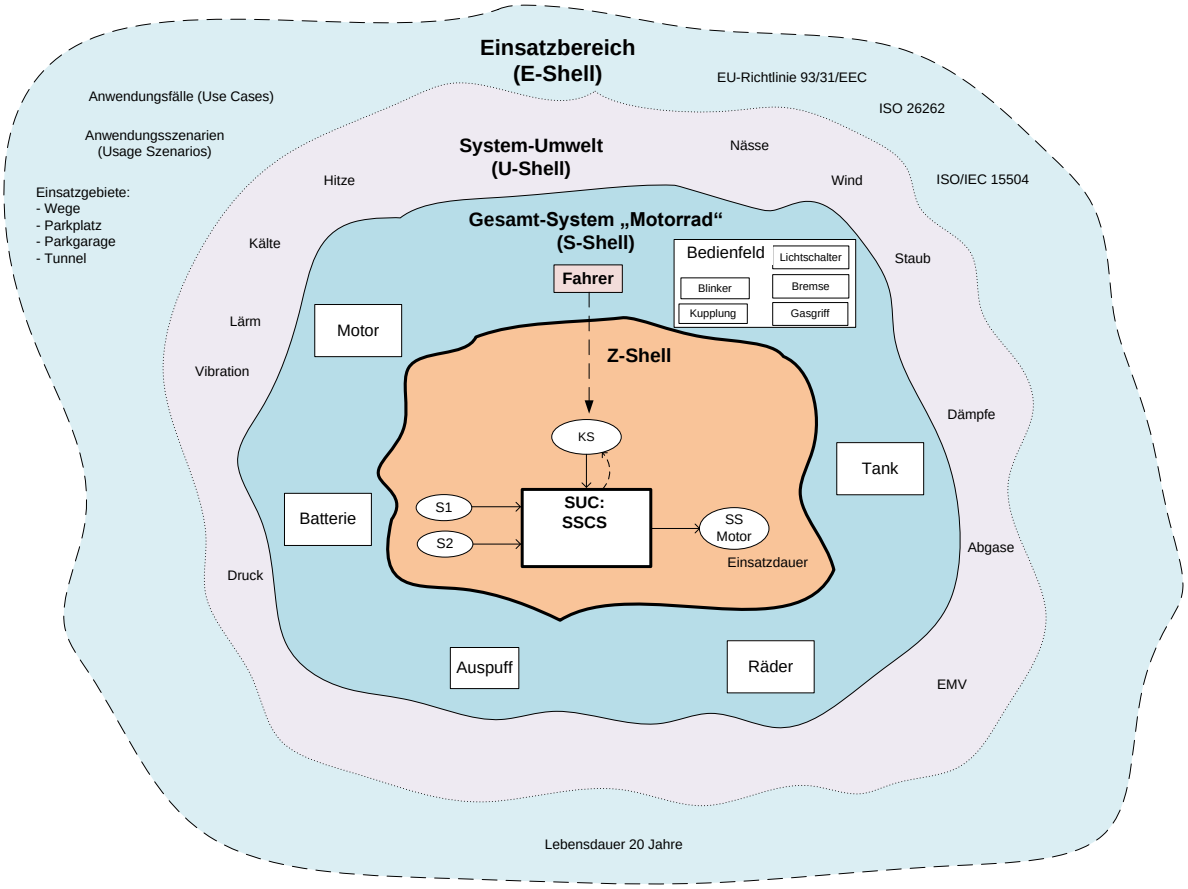


Abbildung 53: Shell-Modell des SSCS

Diese indirekten Einflussfaktoren die auf das SSCS einwirken können sind Vibration, Dämpfe, Abgase, Lärm, elektromagnetische Entladungen und elektromagnetische Interferenzen[74]. All diese Einflussfaktoren, direkt sowie indirekt, sind auf exogene Hazards zu analysieren. Damit können bereits in der Vor-Projektphase, im Sinne der Hazard Avoidance, mögliche Hazards vermieden werden. Wenn dies nicht möglich ist, so sind mögliche technische Lösungen, im Sinne der Hazard Prevention oder des Hazard Controllings, im System-Entwurf zu berücksichtigen.

Bei der Identifikation der BNS sollen die Bedienelemente des Motorrads (MR) ebenfalls mitgedacht werden, da sie Einfluss auf das User-Verhalten haben. Hier ist die Positionierung und Ausführung des Kipp-Schalters ein wichtiger Aspekt. Alle anderen Bedienelemente der mittelbaren System-Umgebung sind ebenso zu betrachten, da sie Einfluss auf die Bedienung des Side-Stands nehmen. Dazu gehören die Kupplung und der Blinker-Schalter, die beide mit der linken Hand bedient werden, sowie der Bremshebel für die Vorderradbremse und der Gasdrehgriff, die mit der rechten Hand bedient werden. Die Beleuchtung wird von der rechten Hand aus- und eingeschaltet.

Als System-Umwelten (U-Shell) [KON_BA1.5] wurden Hitze, Kälte, Luftfeuchtigkeit, Wasser, Nässe, Wind, Abgase, Staub, Dämpfe, Lärm, Druck und EMI identifiziert. Als Einsatzbereich (E-Shell) [KON_BA1.6] werden die Länder der europäischen Union ausgewählt. Die hierfür gesetzlichen Vorgaben/Bestimmungen und die geforderten Normen und Standards sind die ISO 26262 und die ISO/IEC 15504. Als Einsatzgebiete wurden Straßen, Wege, Parkplatz/Parkgarage und Tunnel definiert. Als Lebensdauer der SSCSs wurden 20 Jahre angenommen. In der nachfolgenden Tabelle 17 sind die identifizierten Elemente und Informationen aus der System-Umgebung sowie die Einflussfaktoren aus dem System-Umfeld, mit System-Umwelten und Einsatzbereich, nochmals zusammengefasst.

„Shell“	**Inhalt**
SUC	Side-Stand Control System (SSCS)
Z-Shell	Side-Stand Control System inklusive aller direkt kommunizierenden Systeme • Kippschalter K1 „Side-Stand aus und einfahren“ • Sensor S1 „Side-Stand oben“ (U_BNS) • Sensor S2 „Side-Stand unten“ (U_BNS) • Side-Stand Motor (U_BNS) • Versorgungsspannung • Elektrische Leitungen
S-Shell	Gesamtes Motorrad mit Fahrer und Beifahrer. Zu den wichtigsten indirekt beeinflussenden Systemen des Motorrads zählen: • Fahrer • Batterie (Versorgungsspannung)

[74] EMI Elektromagnetische Interferenz
Bekommt das SSCS ein Metallgehäuse, so können elektromagnetische Wellen an der Oberfläche reflektiert und überlagert werden. In der Physik bezeichnet man das als Interferenz: Wellenberge addieren sich, Wellental und Wellenberg löschen sich gegenseitig aus. Derartige Störungen, verursacht durch elektromagnetische Signale, können bei Übertragungskanälen die Integrität der Daten reduzieren und dadurch die Fehlerrate erhöhen.

	• Motor (Vibration, Hitze, Lärm) • Auspuff (Abgase) • Räder (Vibration und Stöße) • Tank (gefährliche Substanz) Das Motorrad selbst kann dann noch verursachen: • Elektromagnetische Entladungen und • Elektromagnetische Interferenzen Bedienfeld des Motorrads • Kupplung (linke Hand) • Blinker Schalter (linke Hand) • Bremshebel (rechte Hand) • Gasdrehgriff (rechte Hand) • Beleuchtung (rechte Hand) • Gangschaltung (rechter Fuß)
E-Shell	System-Umwelten: • Geschwindigkeit • Verkehr • Hitze und Kälte (Temperaturbereich -40°C bis +80°C) • Motor (Verbrennungsmotor in der Nähe!) • Luftfeuchtigkeit • Wasser und Nässe • Wind • Abgase • Staub • Dämpfe • Lärm • Druck • EMI Schutz nötig wegen anderen elektronischen Geräten
U-Shell	Einsatzbereich: • Region: Europäische Union • Unmittelbarer Einsatzbereich: o Straße ▪ Autobahn ▪ Landstraße ▪ Wege o Parkplatz und Parkgarage o Tunnel • Kraftfahrzeugsdurchführungsverordnung

	• ISO 26262 • ISO/IEC 15504 • 20 Jahre Lebensdauer

Tabelle 17: Ergebnisse aus der Shell-Analyse des SSCS

Top-Level Requirements spezifizieren [KON_BA2]

Nachdem die System-Umgebung identifiziert wurde, werden von der *unmittelbaren* System-Umgebung die Top-Level Requirements abgeleitet. Die Basis-Aktivität dieses Prozessschritts ist in 4 Teilschritte unterteilt:

1. Top-Level Funktionen definieren [KON_BA2.1]
2. Funktionale Top-Level Requirements definieren [KON_BA2.2]
3. Nicht-funktionale Top-Level Requirements definieren [KON_BA2.3]
4. Top-Level Interface Requirements definieren [KON_BA2.4]

Die Ermittlung der Top-Level Funktionen (TLF) [KON_BA2.1] basiert auf den identifizierten unmittelbaren benachbarten Systemen, die einfach aus dem Shell-Modell herauszulesen sind (siehe Abbildung 54):

- Me_BNS Fahrer: Der Fahrer ist ein menschliches benachbartes System, welcher den Kipp-Schalter benutzt.
- Uni_BNS Sensoren: Die beiden Sensoren sind unidirektional kommunizierende Systeme, die einen voll eingefahrenen und einen voll ausgefahrenen Side-Stand an das SSCS signalisieren.
- Uni_BNS Motor: Der Motor als unidirektional kommunizierendes System erhält vom SSCS Befehle zum Ein- und Ausfahren des Side-Stands.

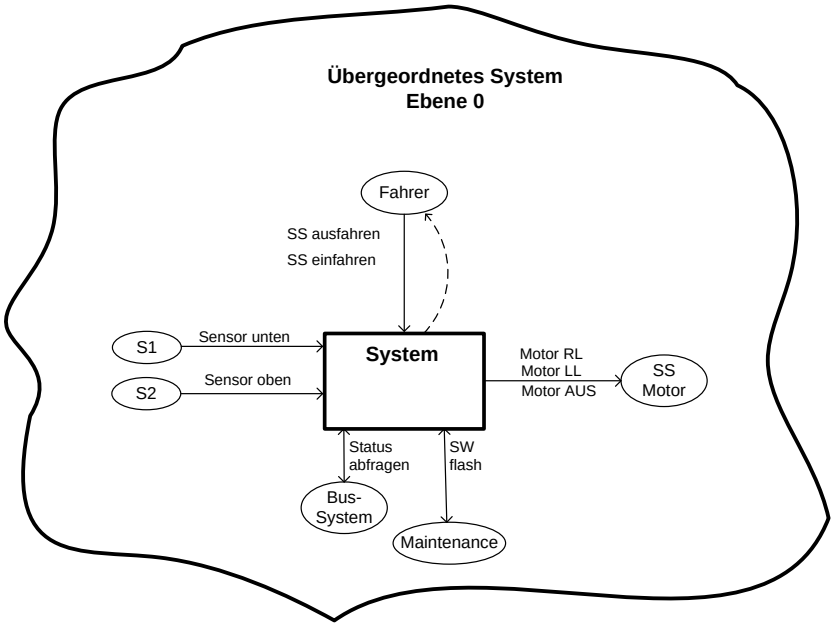


Abbildung 54: Z-Shell des SSCS

Darauf basierend werden die externen Trigger und Services abgeleitet, wie in Tabelle 18 dargestellt. Die beiden Trigger „SS ausfahren“ und „SS einfahren“ sind zugleich auch die Top-Level Funktionen. Die Services, die das SSCS leisten muss, sind „Motor RL“, Motor LL“ und „Motor AUS“, womit die Mechanik des Side-Stands angesteuert wird. Anhand der Trigger und Services können nun die funktionalen Top-Level Requirements [KON_BA2.2] ausspezifiziert werden. Die funktionalen Top-Level Requirements werden unter anderem auch zur Verifikation des technischen Konzepts benötigt.

BNS	**System-Trigger (Top-Level Funktionen)**	**Service (Funktionale Top-Level Requirements)**
Fahrer	SS ausfahren (KS in Richtung „SS ausfahren“ drücken)	Wenn der Fahrer den KS in Stellung „SS ausfahren“ drückt, muss das SSCS den Motor in Richtung „SS ausfahren“ solange ansteuern, bis Sensor S1 „SS unten“ erreicht wird.
Fahrer	SS einfahren (KS in Richtung „SS einfahren“ drücken)	Wenn der Fahrer den KS in Stellung „SS einfahren“ drückt, muss das SSCS den Motor in Richtung „SS einfahren“ solange ansteuern, bis Sensor S2 „SS oben“ erreicht wird.

Tabelle 18: Top-Level Funktionen und Funktionale Top-Level Requirements des SSCS

Aus der System-Umgebung des Shell-Modells können die nicht-funktionalen Requirements hergeleitet [KON_BA2.3] werden. Die Definition der Top-Level Interface Requirements, also der externen Schnittstellen des SSCS, erfolgt im nächsten Teilschritt [KON_BA2.4]. Das sind ein I/O- und ein Motor-Interface.

System-Entwurf entwickeln [KON_BA3]

Die Definition des System-Entwurfs ist in drei Teilschritte unterteilt:

1. System-Inhalte definieren [KON_BA3.1]
2. Interne Schnittstellen definieren [KON_BA3.2]
3. Funktionale Top-Level Requirements verifizieren [KON_BA3.3]

Ausgehend von den Top-Level Interface Requirements werden nun die benötigten System-Inhalte [KON_BA3.1] des SSCS abgeleitet. Für die Ein-/Ausgabe-Schnittstelle (I/O Interface) werden Treiber-Bausteine (I/O Driver)festgelegt. Für die Schnittstelle zum Motor (Motor Interface) wird ein Motor-Treiber (Motor Driver) definiert. Als Steuer-Element ist eine Steuerung (Controller) angedacht. Nach der Definition der Sub-Systeme können die internen Schnittstellen definiert [KON_BA3.2] werden. Des Weiteren muss noch die Versorgungsspannung, die von der Batterie des Motorrads kommt, definiert und die dafür erforderliche Schnittstelle (Vcc Interface) spezifiziert werden. Die nachfolgende Abbildung zeigt den System-Entwurf des SSCS für den elektronisch ausfahrbaren Side-Stand.

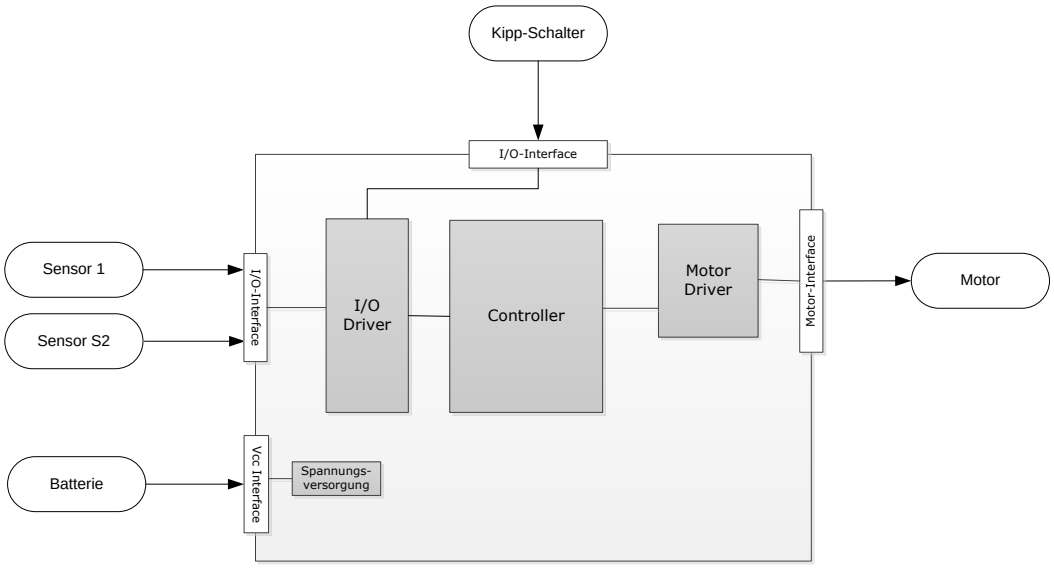


Abbildung 55: Technischer System-Entwurf des SSCS

Der System-Entwurf mit den direkt angrenzenden Systemen bietet nun die Möglichkeit, die funktionalen Top-Level Requirements zu verifizieren [KON_BA3.3]. Dabei werden die spezifizierten funktionalen Top-Level-Requirements, wie sie in Tabelle 18 dargestellt sind, am System-Entwurf in Abbildung 55 nachvollzogen. Damit kann festgestellt werden, ob im System-Entwurf ein Element fehlt bzw. ob die funktionalen Top-Level Requirements lückenhaft sind. Dieses Vorgehen ist erforderlich, bevor die ersten Safety-Analysen durchgeführt werden. Damit wird auch die Traceability zwischen den Funktionalen Top-Level Requirements und dem System-Entwurf sichergestellt. Des Weiteren bietet diese Verifikation die Möglichkeit der Qualitätskontrolle auf der Ebene des System-Entwurfs.

Technisches Konzept spezifizieren und freigeben [KON_BA4]

Der letzte Prozessschritt fordert die Spezifikation und Freigabe des technischen Konzepts. Inhalt des technischen Konzepts sind die Top-Level Requirements, der System-Entwurf und die Spezifikation der System-Umgebung und des System-Umfeldes, anhand des Shell-Modells. Die Spezifikation der Top-Level Requirements beinhaltet die Top-Level Funktionen, die Funktionalen Top-Level Requirements, die Nicht-funktionalen Top-Level Requirements und die Top-Level Interface Requirements.

5.1.3 Side-Stand: Preliminary Hazard Identification (PHI)

Die Vorarbeiten, welche von der Projekt-Initialisierung und der Konzeptionierung geleistet wurden sind wichtige Grundvoraussetzungen für die Preliminary Hazard Identification (PHI). Umso genauer und detaillierten das technische Konzept ausgearbeitet ist, umso effektiver können die Safety-Analysen durchgeführt werden. Wichtig ist dabei auch die, in der Projekt-Initialisierung definierte, anzuwendende Safety-Norm, da sie das Modell für die Safety-Risiko-Bewertung vorgibt. Für die Durchführung der Safety-Analysen ist das technische Konzept ein wichtiger Bestandteil.

Der Prozess der PHI fordert sechs durchzuführende Basis-Aktivitäten. Das ist im ersten Prozessschritt die Identifikation der organisatorischen Safety-Maßnahmen [PHI_BA1], um zu verstehen was von Regulatoren und Zertifizierungsorganisationen gefordert wird. Da es sich beim Side-Stand um ein fiktives Beispiel handelt und daher keine Zertifizierung durchgeführt wird, wird auf diesen Prozessschritt nicht näher eingegangen. Der nächste Prozessschritt fordert die Identifikation der Top-Level Hazards [PHI_BA2]. Darauf basierend wird die Hazard Avoidance [PHI_BA3] angestrebt, in der nach Lösungsmöglichkeiten gesucht wird, Hazards zu vermeiden. Hazards die nicht vermieden werden können, werden einem Safety-Risiko-Assessment [PHI_BA4] unterzogen. Basierend auf den erarbeiteten Lösungen, hier als Safety-Maßnahmen bezeichnet, wird das Safety-Konzept erstellt [PHI_BA5]. Der Safety-Report [PHI_BA5] schließt die Aktivitäten der PHI ab. Die nachfolgende Tabelle 19 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die in der Preliminary Hazard Identification gefordert sind.

PHI_BA1	Organisatorische Safety-Maßnahmen identifizieren
PHI_BA2	Top-Level Hazards identifizieren
PHI_BA3	Hazard Avoidance durchführen
PHI_BA4	Safety-Risiko-Assessment durchführen
PHI_BA5	Safety-Konzept erstellen
PHI_BA6	PHI-Report verfassen und kommunizieren

Tabelle 19: Basis-Aktivitäten der PHI im SSCS

Top-Level Hazards identifizieren [PHI_BA2]

Im nächsten Schritt werden nun, anhand des technischen Konzepts und anhand der Inhalte der System-Umgebung die Top-Level Hazards identifiziert. Die zweite Basis-Aktivität der PHI ist in 5 Teilschritte untergliedert:

1. Top-Level Funktionen auf mögliche Hazards untersuchen [PHI_BA2.1]
2. Unmittelbare System-Umgebung auf mögliche Hazards untersuchen [PHI_BA2.2]
3. Mittelbare System-Umgebung auf mögliche Hazards untersuchen [PHI_BA2.3]
4. Gefahrenquellen und gefährliche Situationen identifizieren [PHI_BA2.4]
5. Unfallsauslösende Ereignisse betrachten [PHI_BA2.5]

Nachfolgend werden zwei Safety-Analyse Methoden angewendet, welche die oben angeführten Teilschritte komplett abdecken. Das ist die „Extended Functional Failure Analysis“ (eFFA), die in Kapitel 4.4 beschrieben ist, und die „Exogene Hazard Analysis“ (exHA) aus Kapitel 4.5.

Zur Identifikation von Fehlfunktionen wird die eFFA herangezogen, da sie in Bezug auf Funktionalitäten, in diesem Fall der Top-Level Funktionen (TLF) [PHI_BA2.1], besonders wirksam ist. Damit wird der Teilschritt [PHI_BA2.1] abgedeckt, und teilweise auch [PHI_BA2.4] und [PHI_BA2.5]. Die nachfolgende Abbildung 19 zeigt die Analyse-Ergebnisse.

extended Functional Failure Analysis (eFFA)							
Failure Identifikation				**Hazard Analyse und Definition**			
#	**Funktion**	**Guideword**	**Failure**	**Hazard Element**	**Initiation Mechanism**	**Konsequenzen / Accident**	**Top-Level Hazard**
1	**SS ausfahren**	Omission	SS fährt nicht aus	---	---	---	---
2		Commission	unbeabsichtigtes Ausfahren des SS	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebens-gefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt bei hoher Geschwindigkeit
3		Incorrect	nur teilweises Ausfahren des SS	Gewicht des Motorrads	Personen in unmittelbarer Nähe SS ist nicht komplett ausgefahren	Personen (Fahrer, Mitfahrer oder Passanten) werden verletzt	teilweise ausgefahrener SS beim Parken lässt das MR kippen
5			SS wird einge-fahren anstatt ausgefahren	---	---	---	---
#	**Funktion**	**Guideword**	**Failure**	**Hazard Element**	**Initiation Mechanism**	**Konsequenzen / Accident**	**Top-Level Hazard**
6	**SS einfahren**	Omission	SS wird nicht eingefahren	Geschwindigkeit	Linkskurve	Fahrer stürzt und wird lebens-gefährlich verletzt	Fahrer fährt mit nicht voll-ständig eingefahrenem SS los
7		Commission	Unbeabsichtigtes Einfahren des SS	Gewicht des Motorrads	Personen in unmittelbarer Nähe SS fährt ein	Personen (Fahrer, Mitfahrer oder Passanten) werden verletzt	unbeabsichtigtes Einfahren des SS während des Parkens lässt das MR kippen
8		Incorrect	SS wird nur teilweise eingefahren	Geschwindig-keit	Linkskurve	Fahrer stürzt und wird lebens-gefährlich verletzt	Fahrer fährt mit nicht voll-ständig eingefahrenem SS los

Tabelle 20: Extended Functional Failure Analysis (SSCS)

Bei der Durchführung der eFFA wurden vier Top-Level Hazards (TLHa) identifiziert, die nachfolgend gelistet sind:

- **TLHa1:** Unbeabsichtigtes (teilweises) Ausfahren des Side-Stand während der Fahrt bei hoher Geschwindigkeit
- **TLHa2:** Fahrer fährt mit nicht vollständig eingefahrenem Side-Stand los
- **TLHa3:** Unbeabsichtigtes Einfahren des Side-Stand in der Parkposition
- **TLHa4:** Nur teilweise ausgefahrener Side-Stand lässt das Motorrad beim Parken kippen

Zur Identifikation der exogenen Hazards über die gesamte System-Umgebung wird die exogene Hazard Analysis (exHA) herangezogen. Mit der exHA, in Kombination mit dem Shell-Modell, können die Teilschritte [PHI_BA2.2] bis [PHI_BA2.5] vollständig abgedeckt werden. Die nachfolgende Abbildung zeigt die Ergebnisse der exHA:

Exogene Hazard Analysis							
Failure Identifikation				**Hazard Analyse und Definition**			
#	**Benachbartes System**	**Exogene Einflüsse/Failure**	**Unmittelbare Auswirkungen**	**Hazard Element**	**Initiation Mechanism**	**Konsequenzen / Accident**	**Top-Level Hazard**
1	**Kipp-Schalter (KS)**	Unterbrechung	keine	---	---	---	---
2		Vibrationen des MR verursachen Kurzschluss	SS fährt während der Fahrt ein	---	---	---	---
3			SS fährt während der Fahrt aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
4		Vibrationen des MR verursachen einen Kurzschluss der KS-Kontakte --> SS fährt aus	SS fährt während der Fahrt aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
5		Vibrationen des MR verursachen einen Kurzschluss der KS-Kontakte --> SS fährt ein	SS fährt während der Fahrt ein	---	---	---	---
6	**Verdrahtungs-leitungen**	Störspitzen auf den Verdrahtungsleitungen	Unerwünschte Signale am I/O-Interface des SSCS verursachen ein Ausfahren des SS	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
7		Unterbrechung	keine Funktion	---	---	---	---
8		Kurzschluss	SS fährt während der Fahrt aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
9	**Sensor S1 "oben"**	Kaputter S1: Unterbrechung	Bei komplett eingefahrenem SS wird der SS-Motor nicht abgeschaltet --> Motorr läuft heiß und wird beschädigt	---	---	---	---
10		Kaputter S1: Kurzschluss	Motor wird nicht eingeschaltet	---	---	---	---
11		S1 ist falsch positioniert/montiert	SS ist nicht ganz eingefahren	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	Fahrer fährt mit nicht ganz eingefahrenem SS los **--> wie TLHa2**
12	**Sensor S2 "unten"**	Kaputter S2: Unterbrechung	Bei komplett ausgefahrenem SS wird der SS-Motor nicht abgeschaltet --> Motorr läuft heiß und wird beschädigt	---	---	---	
13		Kaputter S2: Kurzschluss	Motor wird nicht ausgeschaltet --> Motorr läuft heiß und wird beschädigt	---	---	---	
14		S2 falsch ist positioniert/montiert	SS ist nicht ganz ausgefahren	Gewicht des MR	MR abstellen	MR verletzt Fahrer	nicht vollständig ausgefahrener SS bringt das MR beim abstellen ins kippen **--> wie TLHa4**

#	Benachbartes System	Exogene Einflüsse/Failure	Unmittelbare Auswirkungen	Hazard Element	Initiation Mechanism	Konsequenzen / Accident	Top-Level Hazard
15	**Versorgungs-spannung**	Versorgungs-spannung fällt aus	keine Reaktion des SS	---	---	---	---
16		Spannungs-schwankungen	SSCS führt nicht zulässige Funktionen aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
17		Störspitzen auf der Versorgungsspannung	SSCS führt nicht zulässige Funktionen aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
18	**Mechanik**	SS-Mechanik bleibt beim Ausfahren stecken	SS fährt nicht komplett aus, Motor läuft heiß und wird beschädigt	Gewicht des MR	MR abstellen	MR verletzt Fahrer	nicht vollständig ausgefahrener SS bringt das MR beim abstellen ins kippen **--> wie TLHa4**
19		SS-Mechanik bleibt beim Einfahren stecken	SS fährt nicht komplett ein, Motor läuft heiß und wird beschädigt	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	Fahrer fährt mit nicht ganz eingefahrenem SS los **--> ähnlich wie TLH2**
20	**Fahrer**	Irrtümliches betätigen des KS während der Fahrt in Richtung "SS ausfahren"	Side-Stand fährt während der Fahrt aus	Geschwindigkeit	Linkskurve	Fahrer stürtzt und wird lebensgefährlich verletzt	unbeabsichtigtes Ausfahren des SS während der Fahrt **--> wie TLHa1**
21		Irrtümliches betätigen des KS während der Fahrt in Richtung "SS einfahren"	SS-Motor wird angesteuert, obwohl SS-Mechanik auf Anschlag steht	---	---	---	---

Tabelle 21: Ergebnisse der Exogenen Hazard Analysis (SSCS)

Da die Ursachen der exogenen Hazards in der System-Umgebung liegen, werden diese durch die exHA vorgegeben. Hier werden nochmals zusammenfassend mögliche Ursachen beschrieben.

Als exogener Hazard wurde auch hier das „*unerwünschte Ausfahren des Side-Stands während der Fahrt“* (#3, #4, #6, #8, #16, #17, #20) identifiziert. Dieser Top-Level Hazard ist mit TLHa1 aus der eFFA ident, hat in diesem Fall jedoch andere Ursachen:

- Defekter KS1 (Kurzschluss)
- Prellen der KS1-Kontakte, verursacht durch Vibrationen des Motorrads
- Kurzschluss der Verdrahtungsleitungen
- Störspitzen auf den Verdrahtungsleitungen (z.B. durch EMI)
- Störspitzen auf der Versorgungsspannung
- Fahrer drückt irrtümlich den KS1

Ein „*defekter Kipp-Schalter KS1“* in Kombination mit einem zweiten exogenen Einflussfaktor, der *Vibration des Motorrads*, kann ein Prellen des defekten KS-Kontaktes verursachen und damit den SS ausfahren. Das kann aber auch durch Störspitzen oder Kurzschlüsse der Leitungen verursacht werden. Als weitere Ursache lässt sich ein „*irrtümliches Drücken des Kipp-Schalters“* durch den Fahrer prognostizieren, was durch eine ungünstige Montage des KS begünstigt werden kann.

Des Weiteren wurde der *„nicht vollständig eingefahrene Side-Stand“* (#11) identifiziert, der dem TLHa2 aus der eFFA gleicht, mit folgenden Ursachen:

- Falsch montierter S1
- SS-Mechanik bleibt beim Einfahren stecken
- SS bleibt beim Einfahren stehen (verursacht durch Spannungsschwankung oder Störungen)

Ein falsch montierter Sensor S1 „SS oben“ oder ein mechanischer Defekt der SS-Mechanik können gefährliche Auswirkungen haben. Auch Störungen und Schwankungen in der Versorgungsspannung können das SSCS in einen undefinierten Zustand bringen und sich gefährlich auswirken.

Der TLHa3 wurde aufgrund der Methodik der exHA hingegen nicht gefunden.

Der TLHa4 wurde auch in der exHA identifiziert, bei dem der „*Side-Stand nicht komplett ausgefahren“* (#14, #18) wird und das Motorrad beim Abstellen kippt und dadurch den Fahrer oder Passanten verletzen kann. Es wurden dafür folgende Ursachen identifiziert:

- Falsch montierter S2
- SS-Mechanik bleibt beim Ausfahren stecken
- SS bleibt beim Ausfahren stehen (verursacht durch Spannungsschwankung oder Störungen)

Die Ursachen sind die gleichen wie beim identifizierten Top-Level Hazard #11, ein *falsch montierter Sensor* und ein *mechanischer Defekt des Side-Stands*, verursacht durch Verschmutzung oder Korrosion der Mechanik. Auch hier können Störungen und Schwankungen in der Versorgungsspannung das SSCS in einen undefinierten Zustand bringen und sich gefährlich auswirken.

Darüber hinaus wurden weitere Einflussbereiche aus Mechanik, Elektrik und Umwelt über das Shell-Modell identifiziert. In verschiedenen Kombinationen mit anderen benachbarten Systemen konnten weitere Failure identifiziert werden, die jedoch zu keinen zusätzlichen Hazards führen können. Jedenfalls dienen als Hilfestellung zur gesicherten Abdeckung dieser Ursachen bereits auch definierte Vorgaben aus dem Qualitätsmanagement. Maßnahmen dafür sind beispielsweise kurzschlussfeste Klemmen (permanente Versorgungsspannung und Fahrzeugmasse) sowie der Verpolungsschutz oder ein maximal zulässige Ruhestrom von 100µA. Im mechanischen Bereich ist das die Anforderung über die Beständigkeit gegenüber Vibration und mechanischem Stoß. Um exogene Hazards bzw. Fehler im Umweltbereich (wie z.B. Witterungseinflüsse oder EMI) zu vermeiden, werden beispielsweise Temperaturschock- und Verweiltests durchgeführt, oder EMV-Maßnahmen ergriffen. Da diese Qualitäts-Anforderungen zum State-of-the-Art bei den Entwicklungen von Steuergeräten nach Reif [Rei_12, S.239ff] gehören sind diese Ursachen damit weitestgehend entschärft. Diese Qualitäts-Anforderungen werden in dieser Arbeit nicht näher betrachtet.

Hazard Avoidance durchführen [PHI_BA3]

Die Herausforderung bei dieser Basis-Aktivität besteht darin Hazards soweit als möglich zu vermeiden. Bei den identifizierten Hazards im SS-Beispiel können durch eine Safety Application

Condition[75], die an die Mechanik gerichtet ist, die exogenen Hazards „Fahrer fährt mit nicht ganz eingefahrenen SS los“ und „Nicht vollständig ausgefahrener SS bringt das MR beim Absteigen ins Kippen“, verursacht durch „falsch positionierte Sensoren“, vermieden werden. In diesem Fall wurde der Initiation Mechanism eliminiert.

- **TLSAC1**: Die Montage-Position der Sensoren muss mechanisch sichergestellt sein, sodass ein Verrücken nicht möglich ist.

Darüber hinaus können weitere Forderungen, im Sinne von TLSACs an den Kippschalter KS und an die Mechanik gestellt werden.

Safety-Risiko-Assessment durchführen [PHI_BA4]

Das Safety-Risiko-Assessment ist in die nachfolgenden 5 Teilschritte unterteilt:

1. Risiko-Bewertung durchführen [PHI_BA4.1]
2. Sicherheitsanforderungsstufe bestimmen [PHI_BA4.2]
3. Safety-Goals definieren [PHI_BA4.3]
4. Safety Requirements ableiten [PHI_BA4.4]
5. Safety Maßnahmen definieren [PHI_BA4.5]

Die identifizierten Top-Level Hazards werden anhand der Risiko-Matrix der ISO 26262 (siehe dazu 4.6.3 „Safety Risiko Assessment“ der „ISO 26262“) bewertet [PHI_BA4.1]. Für TLHa1 wurde die Controllability mit C3 bewertet, da ein ausgefahrener Side-Stand in einer Linkskurve unwillkürlich zum Sturz führt und das Motorrad nicht mehr beherrscht werden kann. Ein derartiger Sturz bei hoher Geschwindigkeit führt zu lebensgefährlichen Verletzungen oder kann sogar tödlich enden, daher lässt sich die Severity mit S3 bewerten. Die Exposure wurde mit E4 bewertet, da das Auftreten einer Linkskurve unvermeidlich ist. Aus der Risiko-Tabelle der ISO 26262 (siehe Abbildung 43) kann der entsprechend ASIL [PHI_BA4.2] entnommen werden. TLHa1 wird daher ein ASIL D zugeordnet. TLHa2 wird mit C3, S2, E3 bewertet und entspricht einem ASIL B. TLHa3 und TLHa4 erhält die Bewertungen C2, S2 und E2 und ergibt QM. Die nachfolgende Tabelle zeigt die Bewertung im Detail sowie die zugeordneten Top-Level Safety Goals (TLSaGo) [PHI_BA4.3].

TLHa-Nr.	Top-Level Hazard	Controllability Severity Exposure	ASIL	Top-Level Safety Goals
TLHa1	Unbeabsichtigtes Ausfahren des SS während der Fahrt	C3 S3 E4	D	TLSaGo1: SS darf während der Fahrt nicht ausfahren

[75] SACs (deutsch "sicherheitsbezogene Anwendungsbedingungen") sind laut EN 50129 [CEN29_03, S. 27] "Regeln, Bedingungen und Einschränkungen, […] die bei der Anwendung des/der Systems/Teilsystems/Einrichtung eingehalten werden müssen."

TLHa2	Fahrer fährt mit nicht vollständig eingefahrenem SS los	C3 S2 E3	B	TLSaGo2: Mit nicht vollständig eingefahrenem SS muss das Losfahren blockiert sein
TLHa3	Unbeabsichtigtes Einfahren des SS in der Parkposition	C2 S2 E2	QM	TLSaGo3: In der Parkposition muss das Einfahren des SS blockiert sein
TLHa4	Nicht vollständig ausgefahrener SS bringt das MR beim Abstellen ins Kippen	C2 S2 E2	QM	TLSaGo4: Der SS muss nach dem Ein- bzw. Ausfahren immer in einer definierten End-Position stehen

Tabelle 22: Ergebnisse der Safety-Risiko-Bewertung des SSCS im Problemraum

Von den Top-Level Safety Goals werden die erforderlichen Top-Level Safety Requirements (TLSaR) abgleitet [PHI_BA4.4]. In diesem Fall können drei TLSaR alle Top-Level Safety Goals erfüllen. Die ausspezifizierten TLSaRs lauten:

- **TLSaR1:** Das SSCS darf das Ein- bzw. Auszufahren nur erlauben, wenn das Motorrad steht *und* die Gangschaltung in neutraler Stellung ist.
- **TLSaR2:** Das SSCS muss die Endposition des Side-Stands detektieren und bei nicht eindeutiger Positions-Detektion eine „Warnung“ sowie ein "Losfahrverbot" an das MR übermitteln.
- **TLSaR3:** Das SSCS muss das Einfahren verhindern, wenn die Zündung des MR ausgeschaltet ist.

Basierend auf den Top-Level Requirements werden die Safety-Maßnahmen zur Reduktion des Risiko-Levels definiert. Im Sinne der Hazard Prevention wurde die Top-Level Safety-Maßnahme TLSaMa1 definiert.

- **TLSaMa1:** Das SSCS muss ein Bus-System zum Motorrad zur Verfügung stellen, um den Status zwischen Motorrad und Side-Stand auszutauschen zu können.

TLSaMa1 definiert die Erweiterung des technischen Konzepts, hier ist ein Bus-System gefordert. Das Bus-System soll eine bidirektional kommunizierende Schnittstelle zum Motorrad herstellen. Diese Schnittstelle bietet damit die Möglichkeit, betriebliche Zustände zwischen Motorrad und Side-Stand auszutauschen. Damit können in gefährlichen Situationen Hazards *verhindert* werden. In diesem Fall soll das Ein- bzw. Auszufahren blockiert werden, wenn das Motorrad fährt *und/oder* die Gangschaltung nicht in neutraler Stellung ist. Damit wird das Top-Level Safety Requirement TLSaR1 erfüllt und der Top-Level Hazard TLHa1, im Sinne der Hazard Prevention, vermieden.
Auch TLHa2 kann mit der TLSaMa1, im Sinne des Hazard Controllings, abgedeckt werden. Dabei werden bei nicht eindeutiger Detektion der Endposition des Side-Stands, über das Bus-System, eine Warnung und ein Losfahrverbot versandt. Damit wird TLSaR2 und das Risiko des TLH2 reduziert.

Auch TLHa3 und TLHa4 können mit der TLSaMa1 abgedeckt werden.
Darüber hinaus sollen Spannungsschwankungen mit einer Spannungsüberwachungs-Hardware „controlled“ werden. Dadurch kann das SSCS, im Sinne des Hazard Controllings, den sicheren Zustand „Side-Stand muss Position halten“, herstellen.

- **TLSaMa2:** Das SSCS muss die Versorgungsspannung laufend überwachen, um Spannungsschwankungen detektieren zu können.

Mit der TLSaMa2 wird das TLSaR4 erfüllt und verhindert damit auch ein unbeabsichtigtes Ausfahren des Side-Stands (TLHa1), verursacht durch Spannungsschwankungen.

Die notwendigen Safety-Maßnahmen werden in einem zweiten Durchlauf der Konzeptionierung (siehe dazu
Abbildung 26 „Ablauf des Problemraums als Flow-Chart“ in Kapitel 3.1.4) eingearbeitet. Im Kapitel 5.1.4 „Side-Stand: Abschluss des Problemraums“ wird das End-Ergebnis des technischen Konzepts, mit dem organisatorischen und dem technischen Konzept nochmals dargestellt.

Safety-Konzept erstellen [PHI_BA5]

Das Safety-Konzept beinhaltet die Analyse-Ergebnisse aus der eFFA (Tabelle 20) und aus der exHA (Tabelle 21) sowie der Ergebnisse aus dem Safety-Risiko-Assessment (Tabelle 22) mit den identifizierten ASILs und den davon abgeleiteten Safety Requirements. Darüber hinaus werden die definierten Top-Level Safety-Maßnahmen dokumentiert. Auch die Safety Application Conditions müssen im Safety-Konzept dokumentiert werden.

PHI-Report verfassen und kommunizieren [PHI_BA6]

Die Ergebnisse der PHI werden im PHI-Report zusammengefasst und an das Projektmanagement berichtet. Damit wird sichergestellt, dass alle erforderlichen Aktivitäten, zur Abdeckung der Safety-Maßnahmen mitgeplant werden.

5.1.4 Side-Stand: Abschluss des Problemraums

Nach der Einarbeitung der Top-Level Safety-Maßnahmen in das technisch Konzept und der nochmaligen Durchführung der PHI wird das organisatorische, das technische und das Safety-Konzept zum Makro-Konzept vereint. Die überarbeiteten Konzepte werden als Abschluss des Problemraums nochmals zusammenfassend dargestellt.

Organisatorisches Konzept:

Im Zuge der Rework-Schleife wurde das Scope of Work (SoW) verfeinert. Dabei wurde die Projektbeschreibung konkretisiert, die Projekt-Ziele mit der Forderung nach ASIL D erweitert und die Hauptaufgaben mit den dafür erforderlichen Confirmation Measures und Safety Case ergänzt. Die nachfolgende Abbildung 56 zeigt das vollständige SoW.

Scope of Work

In diesem Projekt findet eine Neu-Entwicklung einer Elektronik für einen elektrisch aus- und einfahrbaren Side-Stand für Motorräder statt. Entwickelt werden Hard- und Software nach den Anforderungen der ISO 26262 nach ASIL D. Als systematisches Vorgehen dient das Safety-Vorgehensmodell, dabei wird nach den Prozessen des Integrativen Safety Prozesses ISaPro® gearbeitet. Das zu entwickelnde System wird als Side-Stand Control System (SSCS) bezeichnet.

Projektzweck: Erschließung eines neuen Marktsegmentes.

Betriebliche Anforderungen / Operationale Requirements:
Side-Stand soll mit einem Kipp-Schalter aus- und eingefahren werden können.

Lieferobjekt:
Side-Stand Control System (SSCS), bestehend aus Hardware und Software.

Projektziele:	Nicht-Projektziele:
> SSCS für elektrisch aus- und einfahrbaren Side-Stand für Motorräder ist entwickelt > SSCS ist nach den Anforderungen der ISO 26262 ASIL D entwickelt > Normkonformität und Sicherheitsnachweis ist hergestellt o ISO 26262 o ISO/IEC 15504 (SPICE-Level 2)	> Mechanische Entwicklung des Side-Stands > Gehäuse der Elektronik
Hauptaufgaben: 1. Projektmanagement 2. System Requirements 3. System Design 4. Hardware-Entwicklung 5. Software-Entwicklung 6. Test 7. Confirmation Measures 8. Safety Case 9. SPICE-Level 2 Assessment	Projekt-Leistungen: n.a. Projekt-Kosten: n.a.
Projekt-Risiken: n.a.	

Abbildung 56: Statement of Work des SSCS (Letztstand)

Die auf das SoW abgestimmte Projekt-Prozesslandschaft (PPL) ist nachfolgend abgebildet. In der zeitlichen Abfolge sind die wichtigsten Events (Assessments, Audits, Stage Gates etc.) des Projekts dargestellt.

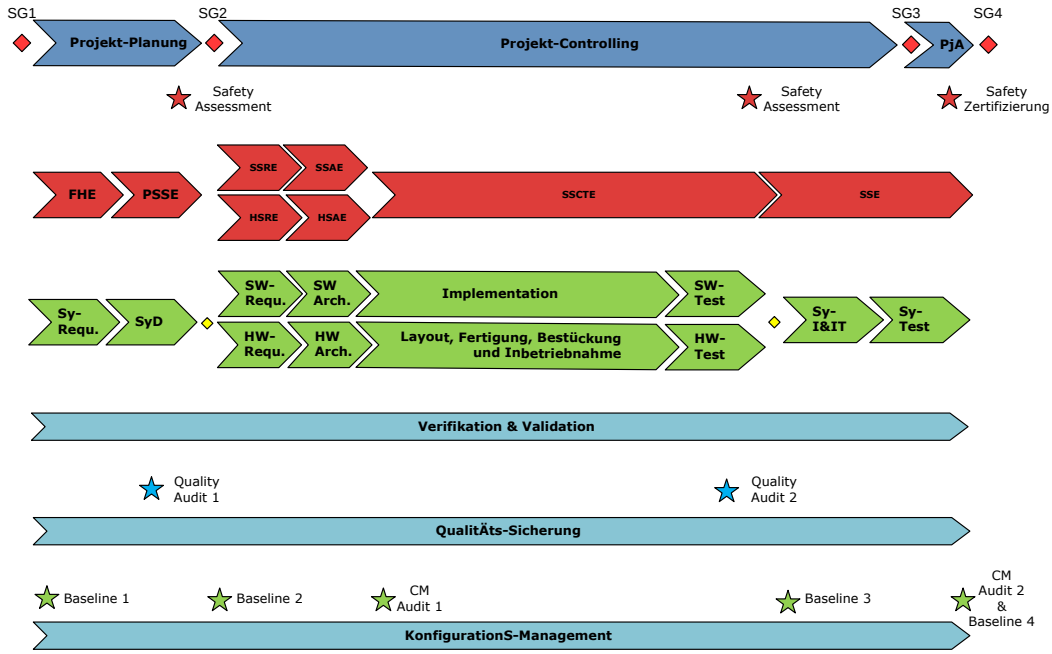


Abbildung 57: PPL des SSCS im Problemraum (Letztstand)

Safety-Konzept

Die nachfolgende Tabelle zeigt die Ergebnisse der Safety-Analyse mit den 5 Top-Level Hazards und den nach der ISO 26262 zugeordneten ASILs sowie den Safety Goals und den Top-Level Safety Requirements. Der höchste ASIL ist ASIL D, der nun für das komplette SSCS als gültig angenommen werden muss. Da sich die Anforderungen des ASIL D nach der ISO 26262 auf die inhaltliche Planung und die erforderlichen Kenntnisse der Projektteammitglieder stark auswirkt, wurde der ASIL D in die Projekt-Ziele mit aufgenommen. Von den Safety Goals wurden 4 Top-Level Safety Requirements abgeleitet.

Top-Level Hazards	ASIL	Safety Goals	Top-Level Safety Requirements
TLHa 1: Unbeabsichtigtes Ausfahren des SS während der Fahrt [eFFA #2] [eHA #3, #4, #6, #8]	D	**TLSaGo1:** SS darf während der Fahrt nicht ausfahren	**TLSaR1:** Das SSCS darf das Ausfahren nur erlauben, wenn das MR steht und die GS in neutraler Stellung ist.
TLHa 2: Fahrer fährt mit nicht vollständig eingefahrenem SS los [eFFA #6, #8] [eHA #11]	B	**TLSaGo2:** Mit nicht vollständig eingefahrenem SS muss das Losfahren blockiert sein.	**TLSaR2:** Das SSCS muss die Endposition des Side-Stands detektieren und bei nicht eindeutiger Positions-Detektion eine „Warnung“ sowie ein "Losfahrverbot" an das MR übermitteln.

TLHa 3: Unbeabsichtigtes Einfahren des SS in der Parkposition. [eFFA #7]	QM	**TLSaGo3:** In der Parkposition muss das Einfahren des SS blockiert sein.	**TLSaR3:** Das SSCS muss das Einfahren verhindern, wenn die Zündung des MR ausgeschaltet ist.
TLHa 4: Nicht vollständig ausgefahrener SS bringt das MR beim Abstellen ins Kippen [eFFA #3] [eHA #14]	QM	**TLSaGo4:** Der SS muss nach dem Ein- bzw. Ausfahren immer in einer definierten End-Position stehen.	**Siehe TLSaR2:**
TLHa5: Durch Spannungsschwankungen und Störspitzen wird der Status des SSCS gestört, dadurch nimmt das I/O einen undefinierten Zustand ein. [exHA#12, #13]	B	**TLSaGo5:** Das SSCS muss gegen Spannungs-schwankungen und Störspitzen resistent sein.	**TLSaR4:** Das SSCS muss die Versorgungs-spannung laufend überwachen und bei einem Abfall der Versorgungsspannung bzw. bei Spannungsschwankungen den SS in der vorhandenen Stellung halten und eine „Warnung“ an das MR übermitteln.

Tabelle 23: Top-Level Hazards, ASIL, Safety Goals und -Requirements des SSCS

Des Weiteren wird die Top-Level Safety Application Condition definiert, da die mechanische Position der Sensoren von der SSCS nicht kontrolliert werden kann und sichergestellt werden muss, dass die Sensoren richtig an den beiden Endpositionen montiert sind.

- **TLSAC1:** Die Montage-Position der Sensoren muss mechanisch sichergestellt sein, sodass ein Verrücken nicht möglich ist.

Technisches Konzept

Das technische Konzept wurde ursprünglich systematisch mit dem Shell-Modell hergeleitet und wird nach der PHI um die TLSaR 1 bis 3 erweitert. Dabei wurden folgende Top-Level Safety-Maßnahmen für das SSCS definiert:

- **TLSaMa1:** Das SSCS muss ein Bus-System zum Motorrad zur Verfügung stellen, um den Status zwischen Motorrad und Side-Stand auszutauschen zu können
- **TLSaMa2:** Das SSCS muss die Versorgungsspannung laufend überwachen, um Spannungsschwankungen detektieren zu können.

Die nachfolgende Abbildung 58 zeigt den System-Entwurf des technischen Konzepts mit den geforderten Maßnahmen.

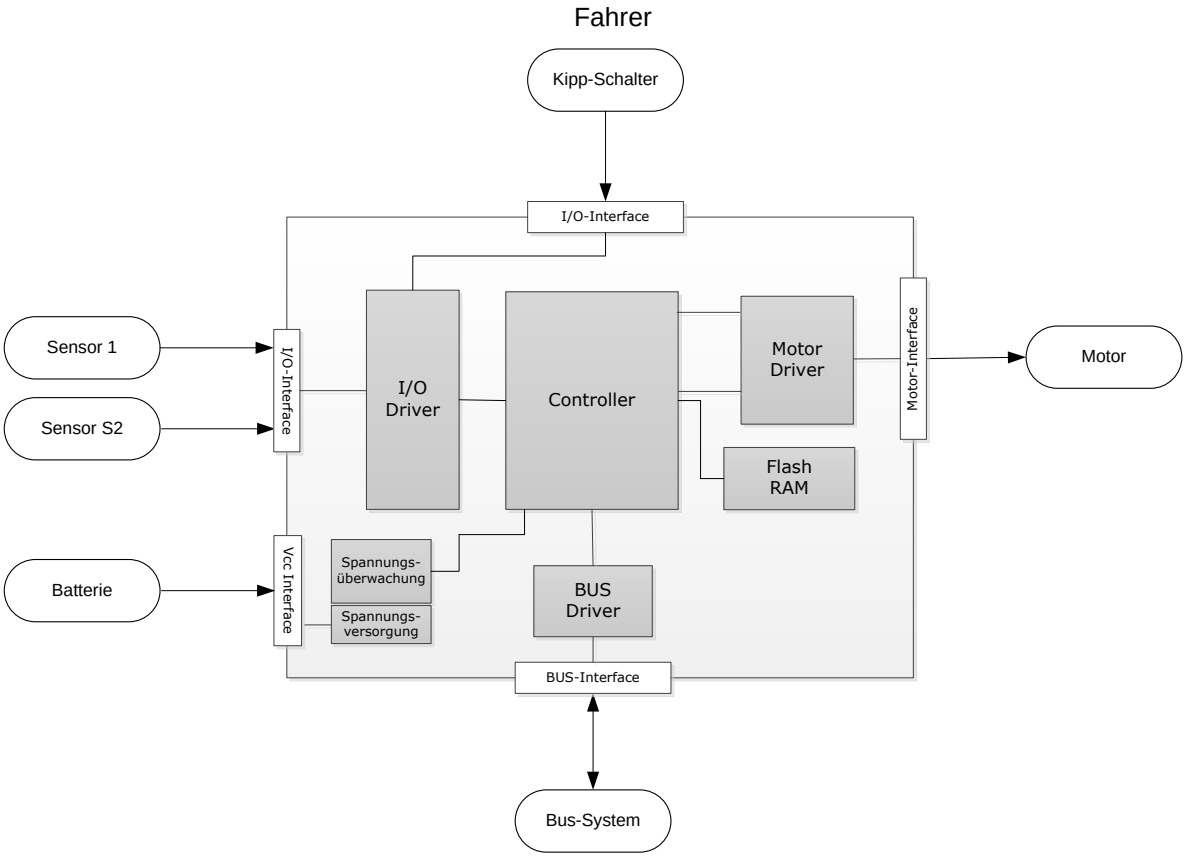


Abbildung 58: Erweitertes technisches Konzept mit Bus-Schnittstelle und Bus-Treiber des SSCS

Damit müssen die Top-Level Safety Application Conditions (TLSAC) an das Motorrad bzw. den Motorradhersteller erweitert werden. Zur besseren Übersicht werden nochmals alle TLSACs dargestellt:

- **TLSAC1:** Die Montage-Position der Sensoren muss mechanisch sichergestellt sein, sodass ein Verrücken nicht möglich ist.
- **TLSAC2:** Das Motorrad muss ein bidirektionales Bus-System zum SSCS zur Verfügung stellen.
- **TLSAC3:** Das Motorrad muss auf Anforderung des SSCSs über das Bus-System die geforderten Status-Informationen zur Verfügung stellen.
    - Motorrad steht bzw. Motorrad fährt
    - Zündung ist eingeschaltet bzw. ausgeschaltet
    - Gangschaltung ist in neutraler Stellung bzw. nicht in neutraler Stellung
- **TLSAC4:** Motorrad muss ein Warnsignal vom Bus-System übernehmen können und den Fahrer über ein akustisches Signal warnen.

Die Ergebnisse des Problemraums zeigen, dass mit einer systematischen Vorgehensweise, wie es das Safety-Vorgehensmodell in dieser Phase fordert, bereits Ergebnisse im Sinne des „Inherent System Safety Approach“ erzielt werden konnten. Es wurde durch das systematisch-methodische Vorgehen und den reifen Prozessen Projektinitialisierung, PHI und Konzeptionierung, das disziplinenübergreifende Denken unterstützt und damit die gefährlichen „Flawed Requirements“ verhindert. Darüber hinaus wurde ein weiterer Aspekt des Inherent System Safety Approach erfüllt, das ist das Denken in Safety Requirements und nicht in Safety-Funktionen wie es die IEC 61508 fordert.

5.2 Modellierungsraum – Projekthandbuch und Design Spezifikationen

Input für den Modellierungsraum ist das Makro-Konzept aus dem Problemraum. Output des Modellierungsraums ist auf der organisatorischen Ebene die vollständige Planung des Projekts anhand des Projekthandbuches (PHB), inklusive aller erforderlichen Planungen der Support-Aktivitäten. Auf der Engineering Ebene wird ein Inherent Safe System Design angestrebt, dass von der System Requirements Spezifikation mit den darin integrierten Safety Requirements abgeleitet wird. Auf der Safety-Ebene sind es die Ergebnisse der Safety-Analysen aus FHE und PSSE sowie die geplanten Safety-Aktivitäten über den gesamten Projektverlauf und die Struktur des Safety Cases. Eine definierte Struktur des Safety Cases muss spätestens in dieser Phase des Projekts eingeführt werden, um die erforderlichen Nachweise (Evidences) strukturiert einpflegen und darstellen zu können.

Auch im Modellierungsraum stehen das systematisch-methodische Vorgehen und die dafür erforderlichen Wechselwirkungen der Disziplinen Engineering, Safety und Projektmanagement im Mittelpunkt. Es wird das gesamte Vorgehen anhand des zeitlich-systematischen Ablaufes, wie er in Kapitel 3.2 dargestellt wurde (siehe dazu Abbildung 28), verfolgt.

5.2.1 Side-Stand: Projektplanung

Die Projektplanung beginnt mit dem Projektstart-Workshop [PjP_BA1], in dem das Projektvorhaben vorgestellt wird. Im Prozessschritt der Projekt-Stakeholder Identifikation [PjP_BA2] werden alle relevanten Personen, Personengruppen und Institutionen identifiziert, die das Projekt auf der organisatorischen Ebene beeinflussen können. Der nächste Prozessschritt fordert die laufende Aktualisierung der Planungsaktivitäten, das sind die Projektplanung [PjP_BA3], die Safety-Planung [PjP_BA4] und die Planung der Support-Aktivitäten [PjP_BA5]. Auf Projektmanagement-Ebene wird die Projektplanung mit der Leistungsplanung [PjP_BA6] im Projektstrukturplan (PSP) konkretisiert. Auf Basis der Arbeitspakete im PSP können danach die erforderlichen Leistungen abgeschätzt [PjP_BA7] werden. Anhand dieser Ergebnisse kann die zeitliche Planung sowie die Kostenplanung [PjP_BA8] durchgeführt werden. Die Leistungs- und die Zeitplanung sind wiederum die Basis für die Ressourcenplanung und die Definition der dafür erforderlichen Projekt-Organisation [PjP_BA9]. Basierend auf den nun vorhandenen Planungsaktivitäten werden die Projekt-Risiken eingeschätzt [PjP_BA10]. Mit der Einführung des Konfigurationsmanagements [PjP_BA11] und der Fertigstellung des Projekthandbuchs [PjP_BA12] wird der Projektplanungs-

prozess abgeschlossen. Die nachfolgende Tabelle 24 gibt einen Gesamtüberblick über die 12 Basis-Aktivitäten, die in der Projektplanung gefordert sind.

PjP_BA1	Projektstart-Workshop durchführen
PjP_BA2	Projekt-Stakeholder identifizieren
PjP_BA3	Projektplanung laufend aktualisieren
PjP_BA4	Safety Plan(ung) erstellen/integrieren
PjP_BA5	Pläne (Support-Prozesse) erstellen/integrieren
PjP_BA6	Leistungsplanung durchführen
PjP_BA7	Leistungs-Abschätzungen durchführen
PjP_BA8	Zeitliche Planung und Kostenplanung durchführen
PjP_BA9	Projektorganisation festlegen
PjP_BA10	Projekt-Risikomanagement durchführen
PjP_BA11	Konfigurationsmanagement definieren und institutionalisieren
PjP_BA12	PHB erstellen und freigeben

Tabelle 24: Basis-Aktivitäten der Projektplanung im SSCS

Da viele der angeführten Basis-Aktivitäten, sowie auch die oben erwähnte erste Basis-Aktivität, zum State-of-the-Art des Projektmanagements zählen und diese wenig Neuigkeitswert darstellen, werden nur die für das Vorgehen wesentlichen Basis-Aktivitäten im Detail beschrieben.

Projektstart-Workshop durchführen [PjP_BA1]

Der Projektstart-Workshop kann als offizielles „Projekt-Kickoff“ gesehen werden, um rasch produktiv arbeiten zu können. Dabei wird das gesamte Projektteam auf einen einheitlichen Wissensstand gebracht. Da es sich beim Side-Stand Beispiel um ein fiktives Projekt handelt, wird dieser Prozessschritt nicht weiter behandelt. Der Projekt-Start Workshop soll jedoch trotzdem erwähnt sein, da ein einheitlicher Wissensstand einen wesentlichen Beitrag zur Reduktion von Human-Errors (siehe dazu Kapitel 4.1.1 „Erweiterung der Fehlerkette“) im Projekt beträgt. Damit können Flawed Requirements auf der organisatorischen Ebene schon zu Projektbeginn vermieden werden.

Projekt-Stakeholder identifizieren [PjP_BA2]

Im zweiten Prozessschritt der Projektplanung erfolgt die Projekt-Stakeholder-Analyse [PjP_BA2] nach dem in Kapitel 4.3 dargestellten Vorgehen. In der Projekt-Stakeholder-Analyse werden alle

Stakeholder, d.h. alle ökonomischen Teilhaber und andere Personen, die Interesse am Projektverlauf haben und diesen beeinflussen können, identifiziert. In der nachfolgenden Abbildung 59 sind die für das Side-Stand Beispiel identifizierten Stakeholder graphisch dargestellt.

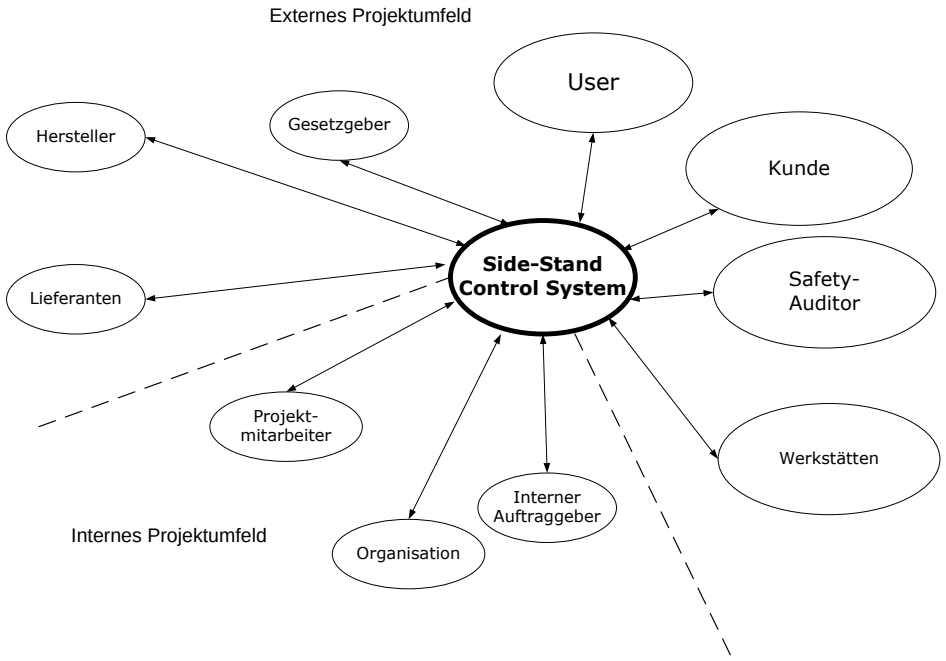


Abbildung 59: Stakeholderanalyse (SSCS)

Im Mittelpunkt der Grafik steht das Projekt, also in diesem Fall das Side-Stand Control Systems (SSCS)“, umgeben von seinem Projektumfeld, also den Stakeholdern. Die Distanz zwischen Projekt und Stakeholdern demonstriert den Beeinflussungsgrad. Umso geringer die Distanz ist, umso größer ist der Beeinflussungsgrad auf das Projekt. Die Größe der Stakeholder-Bubbles stellt die Bedeutung des Stakeholders dar. Eine Darstellungsform, wie sie in Abbildung 59 aufgeführt ist, gibt einen ganzheitlichen Überblick, anhand dessen der Einfluss und die Priorität der einzelnen Stakeholder einfach und schnell erkennbar sind.

Zur Illustration des Vorgehens wurden exemplarisch drei Stakeholder ausgewählt, die sowohl den organisatorischen als auch den technischen Einflussbereich demonstrieren sollen. Damit kann die integrative Betrachtung und die Wechselwirkungen der einzelnen Disziplinen im Vorgehen gut dargestellt werden. Als Stakeholder wurden der Safety Auditor, die Service-Werkstätten und die Motorrad-Hersteller ausgewählt.

Planung laufend aktualisieren [PjP_BA3]

Safety-Plan(ung) erstellen/integrieren [PjP_BA4]

Pläne (Support-Prozesse) erstellen/integrieren [PjP_BA5]

Basierend auf der Stakeholder-Analyse und den Zwischen-Ergebnissen der diversen Safety-Analysen aus FHE und PSSE, werden bereits vorhandene Planungen über den gesamten Ablauf im Modellierungsraum laufend aktualisiert. Die angeführten Prozessschritte sind dabei iterativ zu

betrachten, da sie je nach Projektverlauf und in Abhängigkeit von den Analyse-Ergebnissen im Modellierungsraum immer wieder angepasst werden müssen.

Leistungsplanung durchführen [PjP_BA6]

Die Leistungsplanung wird im Projektstrukturplan (PSP) dargestellt. Die Struktur des PSP wird durch die PPL (siehe dazu Abbildung 57 „PPL des SSCS “) vorgegeben und gliedert somit den gesamten Projektverlauf. Die Basis-Aktivitäten der in der PPL dargestellten Prozesse liefern die Kern-Inhalte für die zu planenden Arbeitspakete. Damit werden Abläufe standardisiert und den „Process Flaws“ entgegengewirkt. Die Prozesse sind auf die Anforderungen der ISO 26262 und der ISO/IEC 15505 (SPICE-Level 2) abgestimmt, womit die „Compliance“ nachgewiesen werden kann. Die nachfolgende Abbildung 60 zeigt den PSP des Side-Stand Control Systems (SSCS).

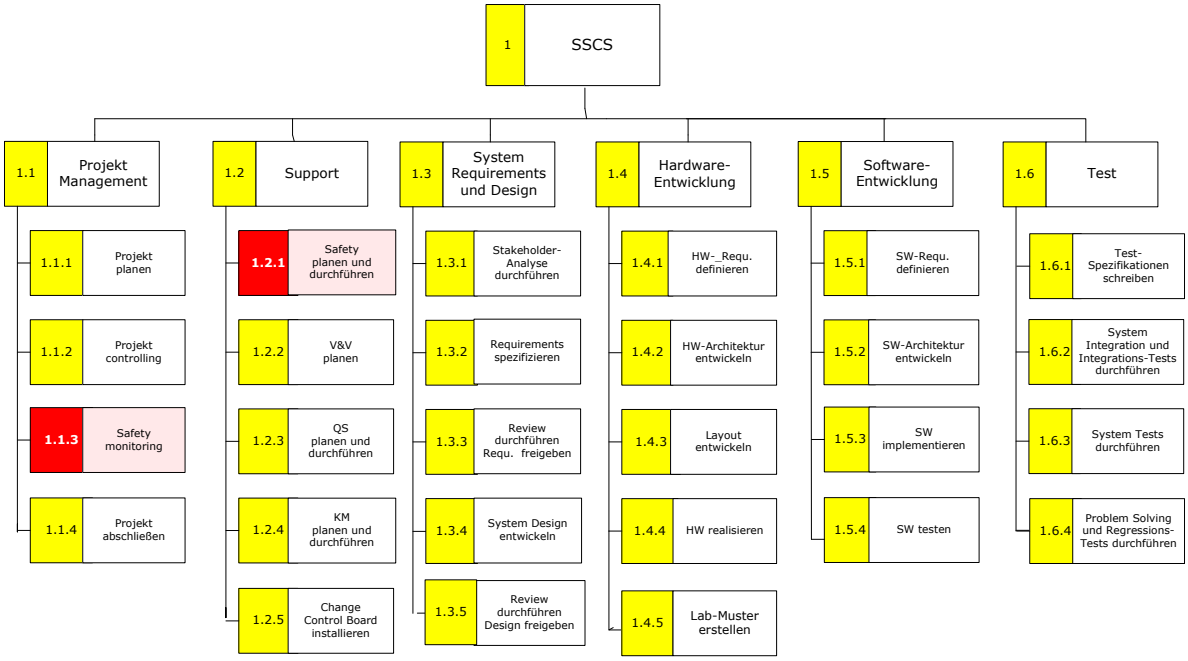


Abbildung 60: Projektstrukturplan (SSCS)

Der PSP gliedert sich in die sechs Projektphasen, „Projekt Management“, „Support“, „System Requirements und Design“, „Hardware Entwicklung“, „Software-Entwicklung“ und „Test“. Die Phase „Projektmanagement“, besteht aus den Arbeitspaketen „Projekt planen“, „Projekt controlling“, „Safety monitoring“ und „Projekt abschließen“. Diese Phase bildet die Grundstruktur des PSP. In der Projektphase „Support“ werden alle Support-Aktivitäten, die durch die Support-Prozesse in der PPL dargestellt sind, geplant. Die Safety-Aktivitäten mit den Safety-Analysen, den Confirmation Measures und der Zertifizierung sowie die Erstellung und Aktualisierung des Safety Cases werden mit dem Arbeitspaket „Safety planen und durchführen“ abgedeckt. Im Arbeitspaket „V&V planen“ werden alle erforderlichen Reviews, Inspections und Test-Aktivitäten über das gesamte Projekt geplant. So fordert es auch die ISO 15504 [ISO15_08] in den Prozessen Verifikation und Validation. Die Compliance zum SPICE-Level 2 sowie die dafür erforderlichen Assessments werden durch das Arbeitspaket „QS planen und durchführen“ abgedeckt. Das

Arbeitspaket „KM-planen und durchführen“ deckt die Aufgaben des Konfigurationsmanagements im Projekt ab. Das sind beim SSCS die Dokumentenlenkung, das Baselining und die Durchführung von Konfigurations-Audits. Das Arbeitspaket „Change Control Board installieren“ institutionalisiert das Änderungsmanagement im Projekt.

Die Arbeitspakete der Phasen „System Requirements und Design“ sowie „Hardware Entwicklung“, „Software Entwicklung“ und „Test“ basieren auf den Basis-Aktivitäten der gleichnamigen Prozesse und auf den Anforderungen der System Requirements sowie auf den Forderungen der geforderten Sicherheitsanforderungsstufe.

Wie aus diesem Beispiel ersichtlich ist, können mit der systematischen Vorgehensweise die generischen Planungs-Patterns des Safety-Vorgehensmodells (SoW, PPL und PSP) angewendet werden, womit Planungsfehler weitgehend vermieden werden können.

Leistungs-Abschätzungen durchführen [PjP_BA7]

Zeitliche Planung und Kostenplanung durchführen [PjP_BA8]

Projektorganisation festlegen [PjP_BA9]

Projekt-Risikomanagement durchführen [PjP_BA10]

Konfigurationsmanagement definieren und institutionalisieren [PjP_BA11]

PHB erstellen und freigeben [PjP_BA12]

Anhand der spezifizierten Arbeitspakete im PSP werden in den nächsten Schritten die durchzuführenden Leistungen der einzelnen Arbeitspakete abgeschätzt [PjP_BA7]. Die Abschätzungsergebnisse liefern die Grundlage für die zeitliche Planung [PjP_BA8], die nach der Struktur der PPL ausgerichtet ist. In den Arbeitspaketen werden die Arbeitspaket-Verantwortlichen und die mitarbeitenden Personen definiert. Das liefert die Basis für die Ressourcenplanung im Projekt womit die Projektorganisation definiert werden kann [PjP_BA9]. Das dafür erforderliche Wissen sowie die benötigte Erfahrung der Projektteammitglieder muss durch den Projektmanager sichergestellt werden. In den geplanten Assessments werden diese vom Qualitätsverantwortlichen überprüft. Damit sollen „Human Errors“, wie sie in Kapitel 4.1.1 „Erweiterung der Fehlerkette“ dargestellt sind, weitgehend vermieden werden. In den darauffolgenden Schritten wird das Projekt-Risikomanagement durchgeführt [PjP_BA10] und das Change Management eingeführt [PjP_BA11]. Dabei wird ein Change Control-Board im Projekt installiert, um Änderungen strukturiert und nachvollziehbar durchführen zu können. Die Fertigstellung des Projekthandbuchs und dessen Freigabe [PjP_BA12] schließen die Projektplanung ab.

5.2.2 Side-Stand: Requirements Engineering

Der Requirements Engineering-Prozess fordert 6 Basis-Aktivitäten. Im ersten Prozessschritt werden die Stakeholder-Anforderungen und -Wünsche eingeholt [RqE_BA1]. Anschließend werden die Stakeholder-Anforderungen und die Top-Level Requirements analysiert und konsolidiert [RqE_BA2] und in die System Requirements übergeführt. Die System Requirements werden danach entsprechend ausspezifiziert, d.h. entwickelt und beschrieben [RqE_BA3] und im darauffolgenden

Prozessschritt evaluiert [RqE_BA4]. Nach positiver Evaluierung werden die funktionalen System Requirements in der FHE einer Safety-Analyse unterzogen [RqE_BA5]. Falls dabei zusätzliche System Requirements identifiziert werden, müssen diese in die System Requirements Spezifikation eingearbeitet werden. Im letzten Prozessschritt werden die System Requirements einem Review unterzogen und freigegeben [RqE_BA6] sowie allen davon betroffenen Stakeholdern kommuniziert. Die nachfolgende Tabelle 25 gibt einen Gesamtüberblick über die 6 Basis-Aktivitäten, die im Requirements Engineering gefordert sind.

RqE_BA1	Stakeholder-Anforderungen und –wünsche einholen
RqE_BA2	Stakeholder-Anforderungen und Top-Level Requirements analysieren und konsolidieren
RqE_BA3	System Requirements entwickeln/ableiten und beschreiben
RqE_BA4	System Requirements evaluieren
RqE_BA5	FHE durchführen
RqE_BA6	System Requirements freigeben und kommunizieren

Tabelle 25: Basis-Aktivitäten des Requirements Engineerings im SSCS

Stakeholder-Anforderungen und -wünsche einholen [RqE_BA1]

Stakeholder-Anforderungen und Top-Level Requirements analysieren und konsolidieren [RqE_BA2]

System Requirements entwickeln/ableiten und beschreiben [RqE_BA3]

Auf Basis der im Planungsprozess identifizierten Projekt-Stakeholder (siehe dazu Abbildung 59) werden die Anforderungen und Wünsche dieser eingeholt [RqE_BA1], unter Berücksichtigung der bereits bestehenden Top-Level Requirements aus dem Problemraum. Im nächsten Prozessschritt werden nun die Anforderungen, der drei, im Planungsprozess ausgewählten Stakeholder, analysiert [RqE_BA2], den entsprechenden Bearbeitungsgebieten zugeordnet und beschrieben [RqE_BA3]. Die nachfolgenden Tabellen demonstrieren das Vorgehen im Detail. Die Tabellen sind so aufgebaut, dass in der ersten Spalte das ursprüngliche Requirement klar ersichtlich ist. In der zweiten Spalte ist die Anforderungskategorie mit dem ausspezifiziertem Requirement dargestellt und in der dritten Spalte die entsprechende Zuordnung.

Als erstes werden Anforderungen des Gesetzgebers eingeholt, analysiert, zugeteilt und damit fixiert. Die nachfolgenden Tabelle 26 demonstriert das Vorgehen.

Stakeholder	Gesetzliche Anforderung	Zuordnung
Gesetzgeber	EU-Richtlinie für Motorräder Council Directive 93/31/IEEE Paragraph 3.1.2: Die EU-Richtlinie verbietet, dass das MR mit nicht komplett eingefahrenem Side-Stand losfahren kann!	Projekt-Ziel (PHB) Wird auch durch **TLSaR1** abgedeckt
Gesetzgeber	EU-Richtlinie für Motorräder Council Directive 93/31/IEEE Paragraph 3.1.1.4: Der Side-Stand darf nicht automatisch einfahren, wenn der Neigungswinkel unbeabsichtigt verändert wird oder das Fahrzeug unbeaufsichtigt in seiner Parkposition steht.	Projekt-Ziel (PHB) Wird auch durch **TLSaR3** abgedeckt

Tabelle 26: Gesetzliche Anforderungen (SSCS)

Die gesetzlichen Anforderungen müssen auf Gesamtsystemebene einer Validation unterzogen werden, also auf Ebene des Motorrads. Da das SSCS die Gesamtsystemebene zwar betrachtet aber der Fahrzeughersteller für die Validierung verantwortlich ist, wird in diesem Projektverlauf keine Validation geplant. Da die dafür erforderlichen Safety-Analysen sowie die Entwicklungs- und Verifikationsmaßnahmen durchgeführt werden müssen, werden die beiden Direktiven aufgrund ihrer Wichtigkeit in die Projektziele aufgenommen. Damit werden sie über die Systematik des Vorgehensmodells in die Hauptaufgaben übernommen und fließen danach über den PSP in die entsprechenden Arbeitspakete ein. Das sind vordergründig die Arbeitspakete 1.6.1 „Test-Spezifikation schreiben“ und 1.6.2 „System Test durchführen“. Mit der Durchführung des Acceptance Tests beim Hersteller, werden die Anforderungen dann schlussendlich validiert.

In der nachfolgenden Tabelle 27 werden die Anforderungen des Safety Auditors dargestellt.

Stakeholder	Norm-Forderungen	Zuordnung
Safety Auditor	Durchführung von Safety Assessments	Projekt-Ziel (PHB)
Safety Auditor	Durchführung von Safety Audits	Projekt-Ziel (PHB)
Safety Auditor	Durchführung von Confirmation-Reviews	Projekt-Ziel (PHB)
Safety Auditor	Safety Case führen	Projekt-Ziel (PHB)
Safety Auditor	Die Organisation muss nach der ISO 26262 [ISO26_11, Teil 2, Kapitel 5.4.2 „Safety Culture“] eine gelebte Safety-Kultur nachweisen.	Darstellung im PHB
Safety Auditor	Die Organisation muss nach der ISO 26262 [ISO26_11, Teil 2, Kapitel 5.4.4.1 „Quality Management during the safety lifecycle“] ein gelebtes Qualitätsmanagement nach der ISO 9001[ISO09_08] oder ISO 16949 [VDA_04] nachweisen.	Darstellung im PHB

Tabelle 27: Forderungen des Safety Auditors (SSCS)

Der Safety Auditor fordert die von der ISO 26262 [ISO26_11, Teil 2, Kapitel 6.2] vorgeschriebenen Safety Audits, -Assessments und Confirmation Reviews. Auch diese Anforderungen werden in die Projektziele aufgenommen, die wiederum über die Hauptaufgaben in die entsprechenden Arbeitspakete des PSPs einfließen. Hier ist in erster Linie das Arbeitspaket 1.2.1 „Safety planen und durchführen“ betroffen. Darüber hinaus prüft der Safety Auditor, ob ein institutionalisiertes Qualitätsmanagementsystem im Unternehmen vorhanden ist und eine Safety-Kultur gelebt wird. Diese Punkte fordert die ISO 26262 in [ISO26_11, Teil 2, Kapitel 5.4.4.1][76]. Der Nachweis wird im Projekthandbuch in einem eigenen Kapitel dargestellt.

Die Anforderungen der Service-Werkstätten sind in der nachfolgenden Tabelle 28 dargestellt.

Stakeholder	**Anforderungen der Service-Werkstätten**	**Zuordnung**
Werkstätten	Das SSCS muss die Möglichkeit der Software-Aktualisierung bieten.	System Requirement **nFuSyR_1**
Werkstätten	Das SSCS muss die Möglichkeit einer Software-Diagnose bieten.	System Requirement **nFuSyR_2**
Werkstätten	Das SSCS muss einen Verpolungsschutz besitzen und gegen die doppelte Versorgungsspannung geschützt sein	nFuSyR_4

Tabelle 28: Anforderungen der Service-Werkstätten (SSCS)

Damit aktuelle Software-Versionen im SSCS aktualisiert werden können, fordern die *Service-Werkstätten* eine Schnittstelle dafür. Darüber hinaus soll das SSCS eine Diagnose-Software zur Fehlerdetektion zur Verfügung stellen, mit der vom SSCS diagnostizierte Fehler in einem Fehlerspeicher zwischengespeichert werden können. Damit wird den Service-Werkstätten die Möglichkeit geboten, beim Service oder bei einer Reparatur bessere und effektivere Diagnosen durchführen zu können. Darüber hinaus werden ein Verpolungsschutz sowie ein Schutz gegen eine erhöhte Versorgungsspannung gefordert. Die Anforderungen der Werkstätten werden den nicht-funktionalen System Requirements nFuSyR_1, nFuSyR_2 und nFuSyR_4 zugeteilt, da sie noch nicht operationalisiert sind.

Die Anforderungen der Hersteller sind in Tabelle 29 dargestellt.

Stakeholder / Quelle	**Anforderungen der Hersteller**	**Zuordnung**
Hersteller	Das SSCS muss die Möglichkeit der End-of-Line Programmierung bieten.	System Requirement **nFuSyR_1**

[76] The organizations involved in the execution of the safety lifecycle shall have an operational quality management system complying with a quality management standard, such as ISO/TS 16949, ISO 9001, or equivalent.

Hersteller	Das SSCS muss die Möglichkeit bieten, End-of-Line Tests durchzuführen.	System Requirement **nFuSyR_3**

Tabelle 29: Anforderungen der Hersteller (SSCS)

Die *Motorrad-Hersteller* fordern eine „End-of-Line Programmierung“ und einen „End-of-Line Test“ für den Fertigungsbereich. Bei der End-of-Line Programmierung handelt es sich um das erstmalige Software-Update des SSCS. Beim End-of-Line Test werden vom Fertigungs-Computer, über das Bus-System, im SSCS interne Test-Prozeduren aufgerufen. Nach der automatischen Durchführung dieser internen Tests soll das Ergebnis an den Fertigungs-Computer rückgemeldet werden. Die erste Anforderung wird mit dem nFuSyR_1 aus Tabelle 28 abgedeckt, für die zweite Anforderung wurde ein weiteres nicht-funktionales System Requirement nFuSyR_3 definiert.

In der nachfolgenden Tabelle 30 sind die funktionalen System Requirements gelistet, die bereits im Problemraum identifiziert wurden. In dieser Phase des Projekts werden sie von funktionalen Top-Level Requirements (FuTLR) zu funktionalen System Requirements (FuSyR) transferiert. Dieser Transfer ist wichtig, da damit die Traceability von den Stakeholder-Anforderungen zu den technischen System Requirements nachgewiesen wird.

Stakeholder / Quelle	**Funktionale Anforderungen**	**Zuordnung**
[FuTLR1]	SS ausfahren: Wenn der Fahrer den KS in Richtung „“SS ausfahren“ drückt, muss das SSCS den Motor in Richtung „Side-Stand ausfahren“ ansteuern, bis Sensor S1 „SS unten“ erreicht ist.	Funktionales System Requirement **FuSyR_1**
[FuTLR2]	SS einfahren: Wenn der Fahrer den Kipp-Schalter in Richtung „“SS einfahren“ drückt, muss das SSCS den Motor in Richtung „Side-Stand einfahren“ ansteuern, bis Sensor S2 „SS oben“ erreicht ist.	Funktionales System Requirement **FuSyR_2**

Tabelle 30: Funktionale Requirements (SSCS)

Die Safety Requirements werden ebenso mit den Top-level Safety Requirements (TLSaR) auf Konsistenz geprüft und in nicht-funktionalen Safety Requirements (nFuSaR) transferiert. Da sie keinen eindeutigen Trigger besitzen, wurden sie den nicht-funktionalen Safety Requirements zugeordnet, wie in der nachfolgende Tabelle 31 dargestellt.

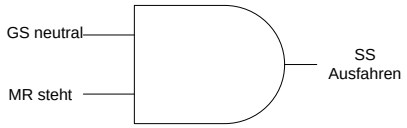
Stakeholder / Quelle	**Safety Requirements**	**Zuordnung**
[TLSaR1]	Das SSCS darf das Ausfahren des SS nur erlauben, wenn das MR steht und die GS in neutraler Stellung ist.  	Nicht-funktionales Safety Requirement **nFuSaR_1**
[TLSaR2]	Das SSCS muss die Endposition des Side-Stands detektieren und bei nicht eindeutiger Positions-Detektion eine „Warnung“ sowie ein "Losfahrverbot" an das MR übermitteln.	Nicht-funktionales Safety Requirement **nFuSaR_2**
[TLSaR3]	Das SSCS muss das Einfahren verhindern, wenn die Zündung des MR ausgeschaltet ist.	Nicht-funktionales Safety Requirement **nFuSaR_3**
[TLSaR4]	Das SSCS muss die Versorgungsspannung laufend überwachen und bei einem Abfall der Versorgungsspannung bzw. bei Spannungsschwankungen den SS in der vorhandenen Stellung halten und eine „Warnung“ an das MR übermitteln.	Nicht-funktionales Safety Requirement **nFuSaR_4**

Tabelle 31: Safety Requirements (SSCS)

Von den Top-Level Safety Requirements TLSaR1 bis TLSaR4 wird auch das Interface-Requirement eInRq_1 abgeleitet und das erforderliche Übertragungsprotokoll festgelegt, siehe dazu Tabelle 32.

Stakeholder / Quelle	**Interface-Requirements**	**Zuordnung**
[TLSaR1] [TLSaR2] [TLSaR3] [TLSaR4]	Das SSCS muss eine Bus-Schnittstelle zur Verfügung stellen, über die der aktuelle Status des MRs und des SSs ausgetauscht werden kann. Geforderter Informationsaustauch: • MR-Status - MR steht - MR fährt - Gangschaltung ist in neutraler Stellung - Gangschaltung ist nicht in neutraler Stellung - Zündung ist eingeschaltet - Zündung ist ausgeschaltet • Information vom SS an das MR: - Losfahrverbot übermitteln - Warnung	Externes Interface Requirement **eIntRq_1:**

Tabelle 32: Interface-Requirements (SSCS)

Die Top-Level Safety Application Conditions (TLSAC) aus dem Problemraum werden ebenso in die Safety-Application Conditions (SAC) transferiert und in die System Requirements Spezifikation übernommen.

Stakeholder / Quelle	**Safety Application Condition**	**Zuordnung**
[TLSAC1]	Der Hersteller der SS-Mechanik muss dafür sorgen, dass die Sensoren S1 und S2 eine fixe Montageposition in der SS-Mechanik besitzen.	Safety Application Condition **SAC_1**
[TLSAC2]	Das MR muss ein bidirektionales Bus-System zum SSCS zur Verfügung stellen.	Safety Application Condition **SAC_2**
[TLSAC3] [TLSAC4]	Das MR muss auf Anforderung des SSCSs über das Bus-System die geforderten Status-Informationen zur Verfügung stellen und die Forderungen des SSCS ausführen: Geforderter Informationsaustauch: • MR-Status - MR steht - MR fährt - Gangschaltung ist in neutraler Stellung - Gangschaltung ist nicht in neutraler Stellung - Zündung ist eingeschaltet - Zündung ist ausgeschaltet - • Information vom SS an das MR: - Losfahren verhindern - Warnung	Safety Application Condition **SAC_3**
[PSSE_BA2.1]	Wird vom SSCS die keine Side-Stand Information übermittelt, muss das MR das Losfahren verhindern.	Safety Application Condition **SAC_4**

Tabelle 33: Safety Application Conditions an die Stakeholder (SSCS)

System Requirements evaluieren [RqE_BA4]

Alle spezifizierten System Requirements werden in diesem Prozessschritt, auf Basis des technischen System-Entwurfs aus dem Problemraum und des aktuellen Shell-Modells, evaluiert. Danach wird die technische Erfüllung der beiden EU-Richtlinien für Motorräder (Council Directive 93/31/IEEE Paragraph 3.1.2 und Paragraph 3.1.1.4) geprüft und die technische Machbarkeit bestätigt. Damit wird die System Requirements Spezifikation an die FHE zur funktionalen Safety-Analyse übergeben.

FHE durchführen[RqE_BA5]

Die evaluierten funktionalen System Requirements werden einer Safety-Analyse unterzogen. Das nachfolgende Unterkapitel 5.2.3 „Side-Stand: Functional Hazard Evaluation (FHE)“ beschreibt die Analyse im Detail.

System Requirements freigeben und kommunizieren [RqE_BA6]

Nach durchgeführter FHE, musste noch ein Derived Safety Requirement (nFuSaR_5: zeitliche Limitierung des Side-Stand Ein- bzw. Ausfahrens) eingearbeitet werden. Nach einer weiteren Iteration wurde die System Requirements Spezifikation freigegeben.

5.2.3 Side-Stand: Functional Hazard Evaluation (FHE)

Ziel der Functional Hazard Evaluation ist es, die funktionalen Requirements in ihrer System-Umgebung und dessen System-Umfeld einer detaillierten Safety-Analyse zu unterziehen. Das Shell-Modell bietet dabei die Möglichkeit Fehlfunktionen und Funktionsausfälle bis in den Einsatzbereich zu analysieren.

Die FHE ist in 6 Prozessschritte untergliedert und beginnt mit der Safety-Planung [FHE_BA1]. Im zweiten Prozessschritt wird die Struktur des Safety Cases festgelegt [FHE_BA2]. Das Safety Case Patterns (siehe dazu Kapitel 4.7) gibt dabei die Grundstruktur vor. Im Prozessschritt drei werden, basierend auf den funktionalen Anforderungen, mögliche Hazard identifiziert [FHE_BA3] und auf Hazard Avoidance analysiert [FHE_BA4]. Die nicht vermeidbaren Hazards werden darauffolgend einem Safety-Risiko-Assessment [FHE_BA5] unterzogen. Im letzten Prozessschritt wird der FHE-Report verfasst und kommuniziert [FHE_BA6]. Die nachfolgende Tabelle 34 gibt einen Gesamtüberblick über die geforderten Basis-Aktivitäten:

FHE_BA1	Safety-Planung durchführen
FHE_BA2	Safety Case Struktur festlegen
FHE_BA3	Hazards identifizieren
FHE_BA4	Hazards Avoidance durchführen
FHE_BA5	Safety-Risiko-Assessment durchführen
FHE_BA6	FHE-Report verfassen und kommunizieren

Tabelle 34: Basis-Aktivitäten der FHE im SSCS

Safety-Planung durchführen [FHE_BA1]

Basierend auf der Stakeholder-Analyse aus dem Requirements-Engineering wird der Kontakt mit den Zertifizierungsbehörden intensiviert. Dabei werden zentrale Informationen in Bezug auf die Zertifizierung und der notwendigen Confirmation Measures sowie der Forderungen und Evidenzen

für den Safety Case eingeholt. Die dafür erforderlichen Maßnahmen werden in die Planung aufgenommen. Da es sich in diesem Fall um ein fiktives Beispiel handelt und die „Confirmation Measures“ bereits in der PPL berücksichtigt sind, wird diese Basis-Aktivität nicht näher behandelt.

Safety Case Struktur festlegen [FHE_BA2]

Basierend auf den bisherigen Ergebnissen ist es zu diesem Zeitpunkt sinnvoll die Initialisierung der Safety Case Struktur zu erarbeiten. Dies basiert auf dem Safety Case Top-Level Pattern, wie es in Kapitel 4.7 „Safety Case Pattern“ beschrieben ist. Die nachfolgende Abbildung 61 zeigt das angepasste Top-Level Pattern für das SSCS.

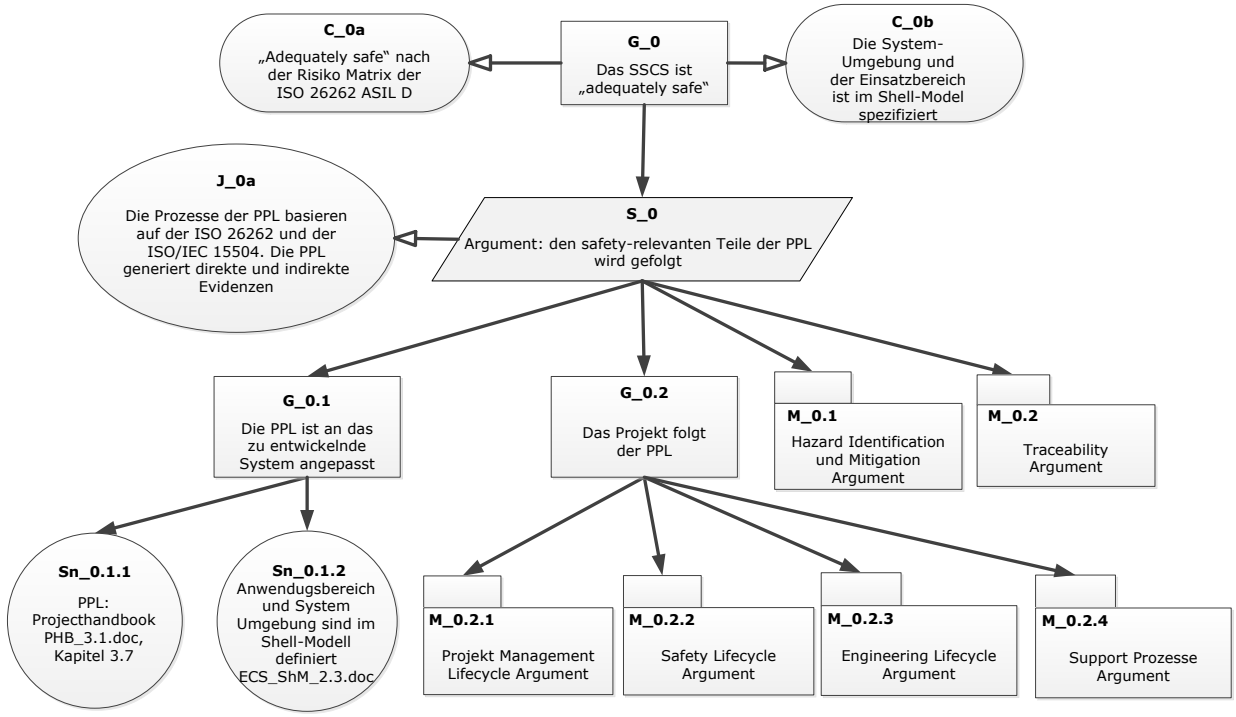


Abbildung 61: Safety Case Grund-Struktur (SSCS)

Der GSN-Baum ist von oben nach unten zu lesen und beginnt mit dem Top-Level Goal. Das Top-Level Goal G_0 sagt aus, dass das SSCS „adequately safe“, also ausreichend sicher ist. Dies wird durch die Strategie S_0 – dass das Projekt nach der PPL des SSCS (siehe dazu Abbildung 52) abgearbeitet wird – und den dargestellten Sub-Goals G_0.1 und G_0.2 durch indirekte Evidenzen argumentiert. Zusätzlich wird G_0 mit den Modulen M_0.1 und M0.2, durch direkte Evidenzen, argumentiert.

Das Goal G_0 weist auch im angeführten Context C_0b, anhand des Shell-Modells, aus, wie das SSCS abgegrenzt ist, wo das SSCS eingebettet ist, wie dessen System-Umgebung aussieht und in welchem Umfeld sich das SSCS betrieben wird bzw. wo dessen Einsatzbereiche liegen. Context C_0a beschreibt den Kontext zur ISO 26262 [ISO26_11], gearbeitet wird nach der normenspezifischen Risiko-Matrix.

Das Sub-Goal G_0.1 hat zum Ziel, zu beweisen, dass die PPL an die Projektsituation des SSCS angepasst wurde. Dieser Nachweis wird durch die Solution Sn_0.1.1 erbracht, das heißt, als Beweis wird das Projekthandbuch (PHB) herangezogen, in dem im Kapitel 3.7 im PHB die anzuwendende PPL beschrieben ist. Sn_0.1.2 liefert das Shell-Modell als Beweis für die Einbettung des SSCS im Gesamtsystem Motorrad und dessen Einsatzbereich, welches auch Auswirkungen auf die PPL des SSCS hat.

Das Sub-Goal G_0.2 fordert, dass nach der projektspezifischen PPL, welche im PHB beschrieben ist, gearbeitet wird. Dies wird in den nachfolgenden Modulen Projekt Management M_0.2.1, System Safety Lifecycle M_0.2.2, Engineering Lifecycle M_0.2.3 und den Support-Prozessen M_0.2.4 dargestellt. Diese Module befinden sich noch innerhalb des Parent-Goals, wobei die angeführten Module dieses Goal unterstützen. Diese Module sind als eigene GSN-Bäume ausgeführt und haben die positive Eigenschaft, dass sie anhand dieser Struktur immer wieder verwendet werden können. Die einzelnen Module enthalten eigene Goals, die durch Argumente und Evidenzen unterstützt werden. Die Traceability der Zwischenprodukte/Artefakte und die Hazard-Mitigation werden als separate Sub-Tree Module ausgewiesen, um deren Wichtigkeit in der Argumentation zu demonstrieren. Die nachfolgende Abbildung 62 zeigt den GSN-Baum für die Hazard-Mitigation.

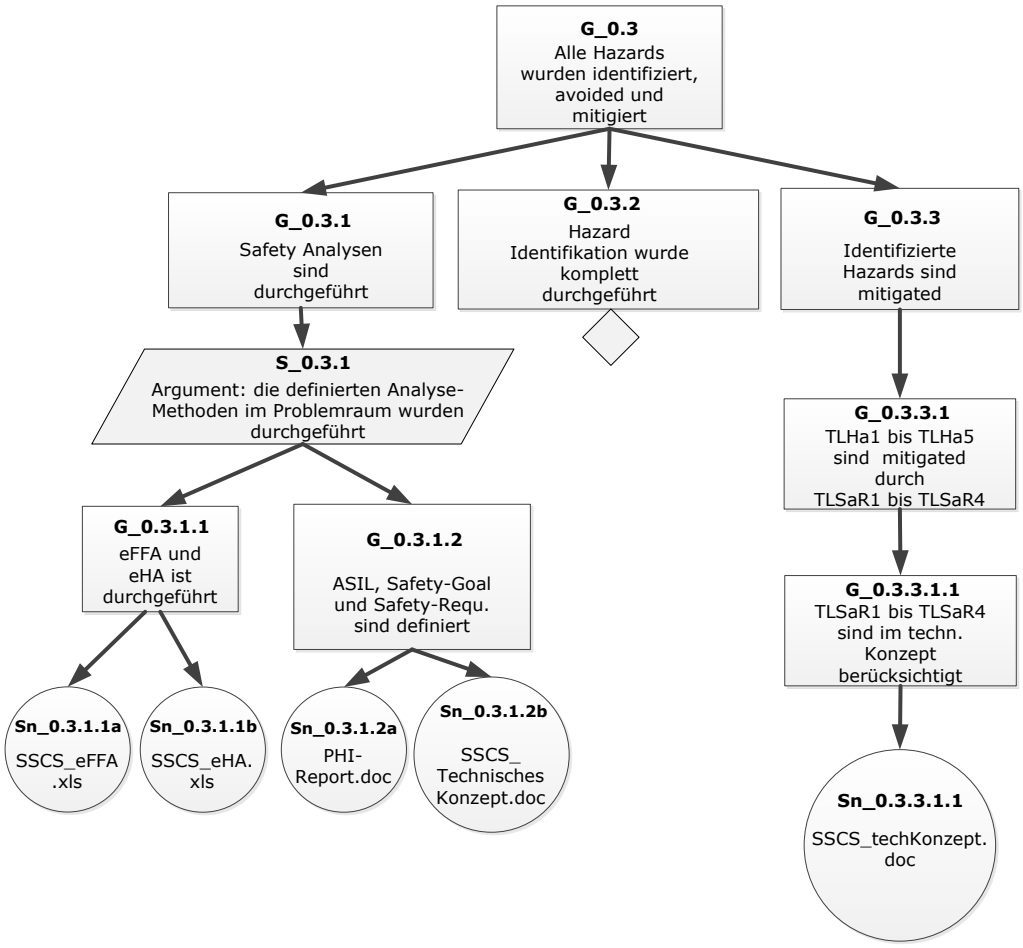


Abbildung 62: GSN Safety Case Hazard-Identifikation und Mitigation (SSCS)

Das Goal G_0.3 beschreibt, dass alle Hazards identifiziert und entsprechend mitigiert sind. Es ist in weitere Sub-Goals unterteilt, wobei G_0.3.1 auf die angewendeten Safety-Analysen ausgerichtet ist, G_0.3.2 beschreibt die Forderung, dass alle Hazards identifiziert wurden und G_0.3.3, dass die identifizierten Hazards auch auf einen akzeptablen Level limitiert wurden. Goal G_0.3.1 wird über die angewendete Analyse-Methode und dem -Ergebnis argumentiert. Die durchgeführten Methoden sind mit G_0.3.1.1 durch die durchgeführte eFFA und eHA argumentiert und wird mit den Excel-Dokumenten SSCS_FFA.xls und SSCS_eHA.xls bewiesen. Goal G_0.3.1.2 setzt das Ziel, dass der ASIL eingeschätzt wurde, das Safety Goal definiert und davon die Safety Requirements abgeleitet wurden. Die Evidence/Solution Sn_0.3.1.2a beweist mit dem PHI-Report das gesamte Ergebnis der Safety-Analyse, in dem der ASIL, die Safety Goals und die Safety Requirements an das Projektmanagement berichtet werden. Sn_0.3.1.2b liefert als Evidence das technische Konzept, in dem die Safety Requirements spezifiziert sind.

Hazards Identifizieren [FHE_BA3]

Im nächsten Schritt wird basierend auf den spezifizierten funktionalen System Requirements eine detaillierte Safety-Analyse durchgeführt. Die FHE empfiehlt in diesem Fall die Anwendung der Hazard and Operability Analysis, kurz HAZOP genannt (siehe Kapitel 2.5.3). Die HAZOP hat den Vorteil, dass sie auf der Extended Functional Failure Analysis (eFFA), aus dem Problemraum, nahezu nahtlos aufsetzen kann. Die HAZOP ist, wie die eFFA, eine explorative Analyse-Methode und auf Funktionen anwendbar. Im ersten Schritt werden die Guidewords für die HAZOP definiert/ausgewählt. Für den Side-Stand werden die Guidewords aus Tabelle 1 in Kapitel 2.5.3 „Hazard and Operability Analysis “ benutzt, die in der nachfolgenden Tabelle 35 nochmals angeführt ist.

HAZOP Guidewords	HAZOP Leitworte	Bedeutung
No (not, none)	kein, nicht	Völlige Verneinung der Sollfunktion
More	mehr	Quantitativer Zuwachs
Less	weniger	Quantitative Abnahme
As well as	sowohl als auch	Qualitativer Zuwachs
Part of	teilweise, zum Teil	Qualitative Abnahme
Reverse	umgekehrt	Logisches Gegenteil der Sollfunktion
Other than	anders als	Völliger Austausch, andere Betriebszustände
Early	zu früh	Verfrühte Reaktion
Late	zu spät	Verspätete Reaktion
Before	vor	Früher als geplant / erwartet (Reihenfolge)
After	nach	Später als geplant / erwartet (Reihenfolge)

Tabelle 35: Liste von HAZOP Guidewords für das SSCS

Die Guidewords sollen nun auf die identifizierten Funktionen „SS ausfahren“ und „SS einfahren“ angewendet werden. Zusätzlich sollen dabei auch der operationale Bereich betrachtet werden. Die ISO 26262 spricht in diesem Fall von „Operational Situations“. Im konkreten Fall fordert die ISO 26262 die Analyse der Gefährdungen in allen „Operating Modes“ und „Operational Situations“

[ISO26_11, Teil 3, Kapitel 7.4.4.1]: „The operational situations and operating modes in which an item's malfunctioning behaviour is able to trigger hazards shall be described.“[77]. Dies kann zu einer großen Anzahl von Kombinationen führen. Um diese Anzahl handhabbar zu machen, wird wie folgt vorgegangen.

Vorerst werden die Operating Modes ermittelt. Dies kann mithilfe der bis dato aus dem Requirements Engineering ermittelten funktionalen Requirements aus Tabelle 30 erfolgen. Anhand dieser funktionalen Requirements wird zur besseren Übersicht ein Zustandsgraph des SSCS erstellt, aus dem die Operating Modes klar hervorgehen (siehe Abbildung 63).

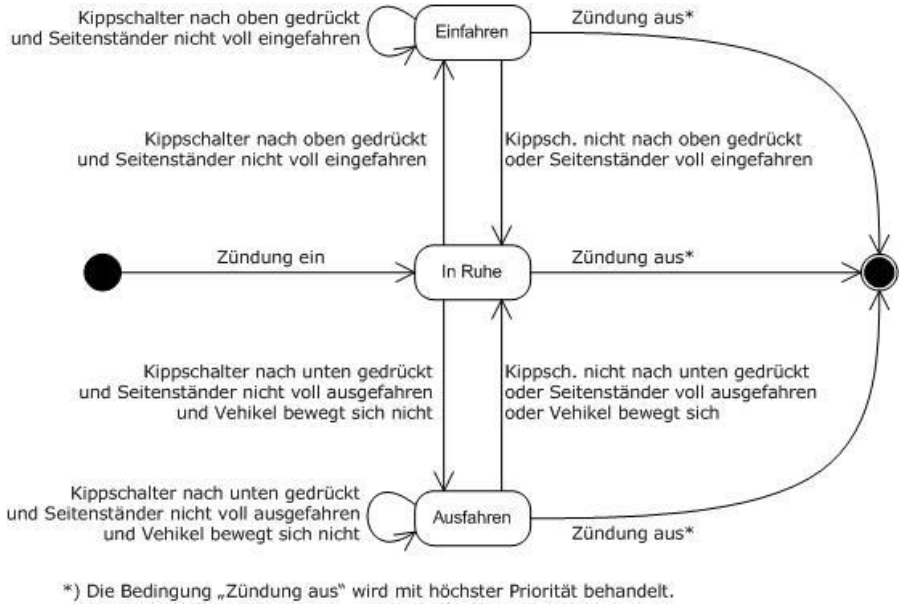


Abbildung 63: Zustandsgraph mit Operating Modes (SSCS)

Aus dem Zustandsgraphen sind nun alle Operating Modes (OpMod) klar erkennbar. Außer den bereits identifizierten OpMod „SS ausfahren“ und „SS einfahren“ ist nun ein zusätzlicher OpMod „SS hält Position“ zu erkennen, der in der HAZOP mitanalysiert wird. Des Weiteren wird noch der von der IEC 61508 [IEC08_10, Teil 4, Kapitel 3.1.14] und der ISO 26262 [ISO26_11, Teil 3, Kapitel 7.4.3.7] geforderte „Foreseeable Misuse“[78] analysiert. Die HAZOP wird nun nur auf die OpMod angewendet und danach ausschließlich die daraus identifizierten Hazards mit den Operational Situations (OpSit) kombiniert, um den Aufwand zu reduzieren.

[77] „Die operationalen Situationen und die Betriebszustände, in welchen die Betrachtungseinheit gefährliche Fehlfunktionen (Hazards) auslösen kann, sind zu beschreiben.“ (Übersetzung des Verfassers)

[78] “Use of a product, process or service in a way not intended by the supplier, but which may result from readily predictable human behaviour. “
„Die Verwendung eines Produkts, Verfahrens oder einer Dienstleistung in einer Weise, die vom Lieferanten so nicht vorgesehen ist, die sich aber aus absehbarem menschlichem Verhalten leicht ergeben kann.“ (Übersetzung des Verfassers)

#	Operating Mode	Guideword	Failure Mode / Fehlfunktion
1	SS ausfahren	kein, nicht	Side-Stand fährt nicht aus
2		mehr	Side-Stand fährt zu weit aus
3		weniger	Side-Stand fährt zu wenig weit aus
4		sowohl als auch	Side-Stand fährt ein und aus
5		teilweise, zum Teil	Side-Stand fährt zu wenig weit aus
6		umgekehrt	Side-Stand fährt ein statt aus
7		anders als	Side-Stand fährt nicht aus oder ein
8		zu früh	Side-Stand fährt vor Stillstand des Fahrzeugs aus
9		zu spät	Side-Stand fährt nach Fahrtbeginn aus
10		vor	Side-Stand fährt vor Stillstand des Fahrzeugs aus
11		nach	Side-Stand fährt nach Fahrtbeginn aus
12	SS einfahren	kein, nicht	Side-Stand fährt nicht ein
13		mehr	Side-Stand fährt zu weit ein
14		weniger	Side-Stand fährt zu wenig weit ein
15		sowohl als auch	Side-Stand fährt ein und aus
16		teilweise, zum Teil	Side-Stand fährt zu wenig weit ein
17		umgekehrt	Side-Stand fährt aus statt ein
18		anders als	Side-Stand fährt nicht ein oder aus
19		zu früh	Side-Stand fährt während des Parkens ein
20		zu spät	Side-Stand fährt erst nach Fahrtbeginn ein
21		vor	Side-Stand fährt während des Parkens ein
22		nach	Side-Stand fährt erst nach Fahrtbeginn ein
23	SS hält Position	kein, nicht	Side-Stand fährt ungewollt ein oder aus
24		mehr	nicht möglich
25		weniger	Side-Stand fährt ungewollt ein oder aus
26		sowohl als auch	Side-Stand fährt ungewollt ein oder aus
27		teilweise, zum Teil	Side-Stand fährt ungewollt ein oder aus
28		umgekehrt	Side-Stand fährt ungewollt ein oder aus
29		anders als	Side-Stand fährt ungewollt ein oder aus
30		zu früh	Side-Stand bleibt beim Ein-/Ausfahren stecken
31		zu spät	Side-Stand fährt ungewollt noch etwas ein oder aus
32		vor	Side-Stand bleibt beim Ein-/Ausfahren stecken
33		nach	Side-Stand fährt ungewollt noch etwas ein oder aus
#	**Foreseeable Misuse**		**Failure Mode / Fehlfunktion**
34	SS ausfahren		Fahrt wird mit ausgefahrenem Side-Stand begonnen
35			Side-Stand wird während der Fahrt ausgefahren

Tabelle 36: HAZOP-Ergebnisse (SSCS)

Aus der HAZOP ist ersichtlich, dass nicht alle Kombinationen sinnvoll sind, beziehungsweis manche Fehlfunktionen mehrfach vorhanden sind.

Nun müssen diese potentiell gefährlichen Fehlfunktionen mit den Operational Situations (OpSit) kombiniert werden, wie von der ISO 26262 gefordert [ISO26_11, Teil 3, Punkt 7.4.2.2 und 7.4.2.3]. Diese Fahrsituationen sind auf Fahrzeugebene zu bestimmen. Mithilfe des Shell-Modells, in dem der Einsatzbereich definiert wurde, wurden folgende Fahrsituationen (Operational Situation) identifiziert.

Nr.	Operational Situation	Beschreibung
1	Fahren	Fahrzeug bewegt sich, der Motor läuft und bewegt die Räder
2	Rollen	Fahrzeug bewegt sich, der Motor läuft nicht und die Räder rollen
3	Stillstand	Fahrzeug bewegt sich nicht, die Räder rollen nicht
4	Parken	Fahrzeug steht am Side-Stand, Motor läuft nicht

Tabelle 37: Operational Situations auf Fahrzeugebene (SSCS)

Hazard Avoidance durchführen [FHE_BA4]

Basierend auf den Analysen des vorhergehenden Prozessschritts, konnten keine weiteren Hazard identifiziert werden. Daher ist die Hazard Avoidance hier nicht anwendbar.

Safety-Risiko-Assessment durchführen [FHE_BA5]

In der nachfolgenden Tabelle 38 sind diejenigen Failures dargestellt, die zu neuen Erkenntnissen führten. Es ist an dieser Stelle noch zu erwähnen, dass die Top-Level Safety Goals (TLSaGo1 bis TLSaGo5) aus dem Problemraum in diesem Prozessschritt nochmals analysiert und auf Gültigkeit überprüft und danach in die Safety Goals (SaGo_1 bis SaGo_5) transferiert wurden.

Failure	Hazard	ASIL	Safety Goals	Safety Requirements
Failure 4: SS fährt ein und aus	**Ha_2:** Fahrer fährt mit nicht vollständig eingefahrenem SS los. [TLHa2]	B	**SaGo_6:** SS muss innerhalb < 5 Sekunden ein bzw. ausgefahren sein und muss die Position halten	**nFuSaR_5:** Wenn beim Ein- bzw. Ausfahren die Endposition innerhalb < 5 Sekunden nicht erreicht wird, muss das SSCS den SS in die ursprüngliche Position bringen und eine „Warnung“ an das MR übermitteln.
Failure 30: SS bleibt beim Ein-/Ausfahren stecken	**Ha_2:** Fahrer fährt mit nicht vollständig eingefahrenem SS los. [TLHa2]	B	Siehe **SaGo_6:**	Siehe **nFuSaR_5:**
Failure 35: SS wird während der Fahrt beabsichtigt ausgefahren	**Ha_1:** Unbeabsichtigtes Ausfahren des SS während der Fahrt bei hoher Geschwindigkeit. [TLHa_1]	D	**SG_1:** SS darf während der Fahrt nicht ausfahren.	**nFuSaR_1:** Das SSCS darf das Ausfahren des SS nur dann erlauben, wenn das MR steht und die GS in neutraler Stellung ist. [TLSaR1]

Tabelle 38: Ergebnisse aus der FHE (SSCS)

Die identifizierte Failure 30 führt zum bereits vorhandenen Hazard 2 „Fahrer fährt mit nicht vollständigem Side-Stand los“. Im Zuge der Analyse wurde dafür ein weiteres wichtiges Safety Goal SaGo_6 definiert. Aufgrund dieses neuen Safety Goals wurde das nicht-funktionalen Requirement nFuSaR_5 definiert. Dies ist eine wichtige Erkenntnis, da man damit auch die Failure 4 abgedeckt wird. Die Failure 35, identifiziert durch den Foreseeable Misuse, führt zum Hazard Ha_1 und wird daher mit dem nicht-funktionalen Safety Requirement nFuSaR_1 abgedeckt.

Die Kombination der gefährlichen Fehlfunktionen mit den Operational Situations ermöglicht es nun, die Safety-Risiko-Bewertung der ISO 26262 zu finalisieren, bzw. die Ergebnisse aus der Projekt-Initialisierungsphase zu untermauern bzw. zu revidieren. Es wird hier in diesem Beispiel nur auf die gefährlichste Kombination eingegangen. Die gefährlichste Kombination ist die Fehlfunktion „Side-Stand fährt ungewollt ein oder aus“ in der Operational Situation „Fahren“. Die Safety-Risiko-Bewertung ergibt eine Severity S3, eine Exposure E4 und eine Controllability C3. Dies führt laut ISO 26262 zu einem ASIL D.

Es ist hier deutlich ersichtlich, dass der schon vorläufig abgeschätzte ASIL aus dem Problemraum bestätigt werden kann. Der Aufwand dafür ist jedoch wesentlich geringer als die von der ISO 26262 durchzuführende Hazard and Risk Analysis. Der Vorteil einer vorläufigen ASIL Abschätzung im Problemraum ist, die Aufwände schon in der Vorprojektphase gut einschätzen zu können. Außerdem wird durch diese Vorarbeiten die vollständige Hazard-Analyse in der FHE entsprechend beschleunigt, sodass insgesamt von keiner signifikanten Erhöhung des Aufwandes ausgegangen werden muss. Ein weiteres Argument für die vorläufige ASIL-Bestimmung ist das Vermeiden von potentiellen (teuren) Fehlern. Dadurch wird auch das gesamte Projektrisiko gesenkt.

FHE-Report verfassen und kommunizieren [FHE_BA6]

Der FHE-Report beinhaltet die oben angeführten Ergebnisse und erfordert in diesem Fall die Einarbeitung des nFuSaR_5 und einen nochmaligen Durchlauf. Nach positivem Durchlauf wird ein erneuter FHE-Report notwendig, um den genauen Hergang zu dokumentieren. Damit werden die Maßnahmen zur Hazard-Mitigation dokumentiert.

5.2.4 Side-Stand: System Design

Der System Design Prozess verfeinert in dieser Phase den technischen System-Entwurf des SSCS anhand von 6 Basis-Aktivitäten. Das ist einmal die Entwicklung der Architektur des System Designs (SyD) [SyD_BA1] auf Basis der definierten System Requirements Spezifikation. Im darauf folgenden Prozessschritt werden alternative Design-Lösungen [SyD_BA2] betrachtet, um damit ein optimales und sicheres SyD darstellen zu können. Danach werden die System Requirements den Sub-Systemen zugeteilt [SyD_BA3]. Ist das SyD vollständig entwickelt, wird es mit den System-Funktionen verifiziert [SyD_BA4], um die Konsistenz und Traceability der System Requirements mit dem SyD sicherzustellen. Eine positive Verifikation liefert die Basis für die Durchführung der PSSE [SyD_BA5], in der das SyD mehreren Safety-Analysen unterzogen wird. Wird in der PSSE festgestellt, dass das SyD nicht inhärent sicher ist, müssen weitere Safety Requirements, in dieser

Arbeit als Derived Safety Requirements (DSaR) bezeichnet, definiert und in das SyD eingearbeitet werden. Dabei muss der gesamte Zyklus des Modellierungsraums solange wiederholt werden, bis kein zusätzlicher Hazard mehr identifiziert werden kann. Erst danach wird das SyD freigegeben und an die davon betroffenen Personen kommuniziert [SyD_BA6].Die nachfolgende Tabelle 39 gibt einen Gesamtüberblick über die Basis-Aktivitäten, die im Design-Prozess gefordert sind.

SyD_BA1	System Design Architektur entwickeln
SyD_BA2	Alternative Lösungen evaluieren
SyD_BA3	System Requirements den Sub-Systemen zuteilen
SyD_BA4	System Design verifizieren
SyD_BA5	PSSE durchführen
SyD_BA6	System Design freigeben und kommunizieren

Tabelle 39: Basis-Aktivitäten des System Designs (SSCS)

System Design Architektur entwickeln [SyD_BA1]

Die Basis für das SyD des SSCS ist die System Requirements Spezifikation und der System-Entwurf des technischen Konzepts. Der Ablauf bei der Entwicklung des SyD ist ähnlich dem Vorgehen wie beim System-Entwurf im Problemraum. Beim Vorgehen im Modellierungsraum muss nun jedes System Requirement einzeln betrachtet und einer konkreten Lösung zugeführt werden. Als erstes wird für jedes funktionale System Requirement, die dafür erforderliche externe Schnittstelle konkretisiert und darauf basierend die dafür notwendigen Sub-Systeme festgelegt. Sind für die externen Schnittstellen Software-Protokolle erforderlich, so müssen diese definiert werden. Das gleiche Vorgehen wird dann auf die nicht-funktionalen System Requirements angewendet. Die nachfolgende Tabelle 40 zeigt das Vorgehen für die funktionalen und nicht-funktionalen System Requirements.

Identifikation der Schnittstellen und Sub-Systeme				
SyR-ID	**System-Requirement**	**ext. Hardware Schnittstelle**	**Hardware Sub-System**	**Software Protokoll**
FuSyR_1	Wenn der Fahrer den KS in Richtung "SS ausfahren" drückt, muss das SSCS den Motor in Richtung "SS ausfahren " einsteuern, bis der KS losgelassen wird oder bis S1 "SS unten" erreicht ist.	I/O-Interface Motor-Interface	I/O-Driver Motor-Driver: H-Bridge	n.a.
FuSyR_2	Wenn der Fahrer den KS in Richtung "SS einfahren" drückt, muss das SSCS den Motor in Richtung "SS einfahren " ansteuern, bis der KS losgelassen wird oder bis S2 "SS oben" erreicht ist.			
nFuSyR_1	Das SSCS muss die Möglichkeit der SW-Aktualisierung bieten.	OBD Interface	CAN Bus Transceiver	CAN Bus Protokoll
nFuSyR_2	Das SSCS muss die Möglichkeit einer SW-Diagnose bieten.			
nFuSyR_3	Das SSCS muss die Möglichkeit bieten, End-of-Line Tests durchzuführen.			
nFuSyR_4	Das SSCS muss einen Verpolungsschutz besitzen und gegen die doppelte Versorgungsspannung geschützt sein.	Vcc Interface	Reverse Polarity Protection	n.a.

Tabelle 40: Systematische Identifikation der System-Schnittstellen und Sub-Systeme (SSCS)

Hier wurden eine Ein-/Ausgabe-Schnittstelle (I/O-Interface), eine Schnittstelle zum Motor (Motor Interface) und der CAN-Bus (CAN Bus Interface) als Kommunikationsschnittstelle zum Motorrad konkretisiert. Die nicht-funktionalen System Requirements nFuSyR1 bis nFuSyR3 fordern hier noch eine zusätzliche externe Hardware-Schnittstelle (OBD-Interface), die im System Design berücksichtigt werden muss. Für die Forderung des Verpolungsschutzes wurde ein „Reverse Polarity Protection Sub-System“ ausgewählt. Das soll bei falschem Anschluss der Versorgungsspannung Schäden am SSCS verhindern.

Als nächstes werden die Safety Requirements betrachtet, welche in der nachfolgenden Tabelle 41 dargestellt sind.

SyR-ID	System-Requirement	ext. Hardware Schnittstelle	Hardware Sub-System	Software Protokoll
nFuSaR_1	Das SSCS darf das Ausfahren des SS nur erlauben, wenn das MR steht und die GS in neutraler Stellung ist.	CAN Bus Interface	CAN Bus Transceiver	CAN Bus Protokoll
nFuSaR_2	Das SSCS muss die Endposition des SS detektieren und bei nicht eindeutiger Positions-Detektion eine „Warnung“ sowie ein "Losfahrverbot" an das MR übermitteln.	I/O-Interface CAN Bus Interface	I/O-Driver CAN Bus Transceiver	CAN Bus Protokoll
nFuSaR_3	Das SSCS muss das Einfahren verhindern, wenn die Zündung des Motorrads ausgeschaltet ist.	CAN Bus Interface	CAN-Bus Treiber	CAN Bus Protokoll
nFuSaR_4	Das SSCS muss die Versorgungsspannung laufend überwachen und bei einem Abfall der Versorgungsspannung bzw. bei Spannungsschwankungen den SS in der vorhandenen Stellung halten und und eine „Warnung“ an das MR übermitteln.	Vcc Interface CAN Bus Interface	Voltage Controlled Divider CAN Bus Transceiver	CAN Bus Protokoll
nFuSaR_5	Wenn beim Ein- bzw. Ausfahren die Endposition innerhalb < 5 Sekunden nicht erreicht wird, muss das SSCS den SS in die ursprüngliche Position bringen und eine „Warnung“ an das MR übermitteln.	CAN Bus Interface	CAN Bus Transceiver	CAN Bus Protokoll
nFuSaR_10	siehe nFuSaR_1			
eIntRq_1	Das SSCS muss eine Bus-Schnittstelle zur Verfügung stellen, über die der aktuelle Status des MRs und des SSs ausgetauscht werden kann. Geforderter Informationsaustauch: • MR-Status - MR steht - MR fährt - Gangschaltung ist in neutraler Stellung - Gangschaltung ist nicht in neutraler Stellung - Zündung ist eingeschaltet - Zündung ist ausgeschaltet • Information vom SS an das MR: - Losfahrverbot übermitteln - Warnung	CAN Bus Interface	CAN Bus Transceiver	CAN Bus Protokoll

Tabelle 41: Identifikation der System-Schnittstellen und Sub-Systeme (SSCS)

Als Bus-System wurde der CAN Bus gewählt, mit CAN Bus Interface und CAN Bus Transceiver. Für den CAN Bus wurde das bereits bestehende und standardisierte CAN-Bus Software-Protokoll ausgewählt. Der CAN Bus Transceiver bietet als Übertragungseinheit spezielle Diagnosefunktionen an. Das sind die Kommunikationsschnittstelle zum Motorrad und die Schnittstelle für den Hersteller bzw. die Service-Werkstätten. Für die zweite Schnittstelle wurde die On-Board-Diagnose Schnittstelle (OBD Interface) ausgewählt. Das OBD Interface ist ein Fahrzeugdiagnosesystem, mit

der ein Fehlerspeicher ausgelesen und Software-Updates durchgeführt werden können. Des Weiteren kann der CAN Bus in den Fail Safe Modus wechseln und damit Fehlerfunktionen, die vom CAN Bus selbst verursacht werden, vermeiden. Für das nFuSaR_4 wurde eine „Voltage Polarity Protection“ definiert, die in das Sub-System „Voltage Controlled Divider“ (nFuSaR_4) integriert wird. Damit deckt das Sub-System folgende Punkte ab:

- Voltage Polarity Check: Ein Verpolungsschutz verhindert Schäden am SSCS, wenn die Versorgungsspannung falsch angeschlossen wird
- Supply Regulator: Liefert eine geregelte Versorgungsspannung für die einzelnen Sub-Systeme (12 V, 5V, 3,3V)
- Voltage Divider: Stellt im Fall Spannungsschwankungen einen definierten Pegel zur Verfügung

Als Motor Driver wurde eine H-Bridge mit Polaritätsumkehr gewählt. Für die Steuerung wurde ein 32-Bit Mikroprozessor (ARM Cortex-A8) ausgewählt, der im Automotive-Bereich breiten Einsatz findet.

Mit diesem Vorgehen werden Inkonsistenzen, Lücken und Überlappungen im System Design vermieden und es wird sichergestellt, dass jedes System Requirement berücksichtigt wurde. Darüber hinaus kann die Traceabiltity vom System Requirement über die Schnittstellen bis zum Sub-System darstellt werden. Die Vermeidung von Flawed Requirements ist hier ein starkes Argument für ein Inherent Safe System Design.

Alternative Lösungen evaluieren [SyD_BA2]

Bei der Forderung nach alternativen Lösungen steht beim Side-Stand Beispiel die ASIL-Reduktion im Mittelpunkt. Durch den modularen Aufbau des SyD bietet sich die Möglichkeit sicherheitskritische Funktionen zu kapseln. Das heißt, mit dieser Methodik kann der ASIL auf die einzelnen Sub-Systeme unterteilt werden. Hier wird ersichtlich, wie wichtig ein strukturales Design (siehe dazu Kapitel 2.1.1) mit klar abgegrenzten, möglichst unabhängigen Sub-Systemen ist.

In diesem Fall, bietet die ISO 26262 explizit eine Möglichkeit, die ASILs einzelner Sub-Systeme mit Hilfe der „ASIL-Decomposition“ (ISO 26262, Teil 10) aufzuteilen und gegebenenfalls den ASIL zu reduzieren. So kann ein ASIL D in folgende ASILs aufgeteilt werden:

- ASIL A und ASIL C
- ASIL B und ASIL B
- ASIL D und QM (kein ASIL)

Die nachfolgende Abbildung 64 zeigt den alternativen Lösungsansatz, ASIL D in mehrere niedrigere ASILs zu unterteilen. Dabei wird die ursprüngliche Steuerung in zwei voneinander unabhängige Steuerungen aufgeteilt, in die Control Unit und in den Expansion Guardian. Beide Steuerelemente werden mit dem bereits definierten 32-Bit Mikroprozessor ARM Cortex A8 ausgestattet. Die nachfolgende Abbildung 64 zeigt das System Design mit den eingearbeiteten Requirements.

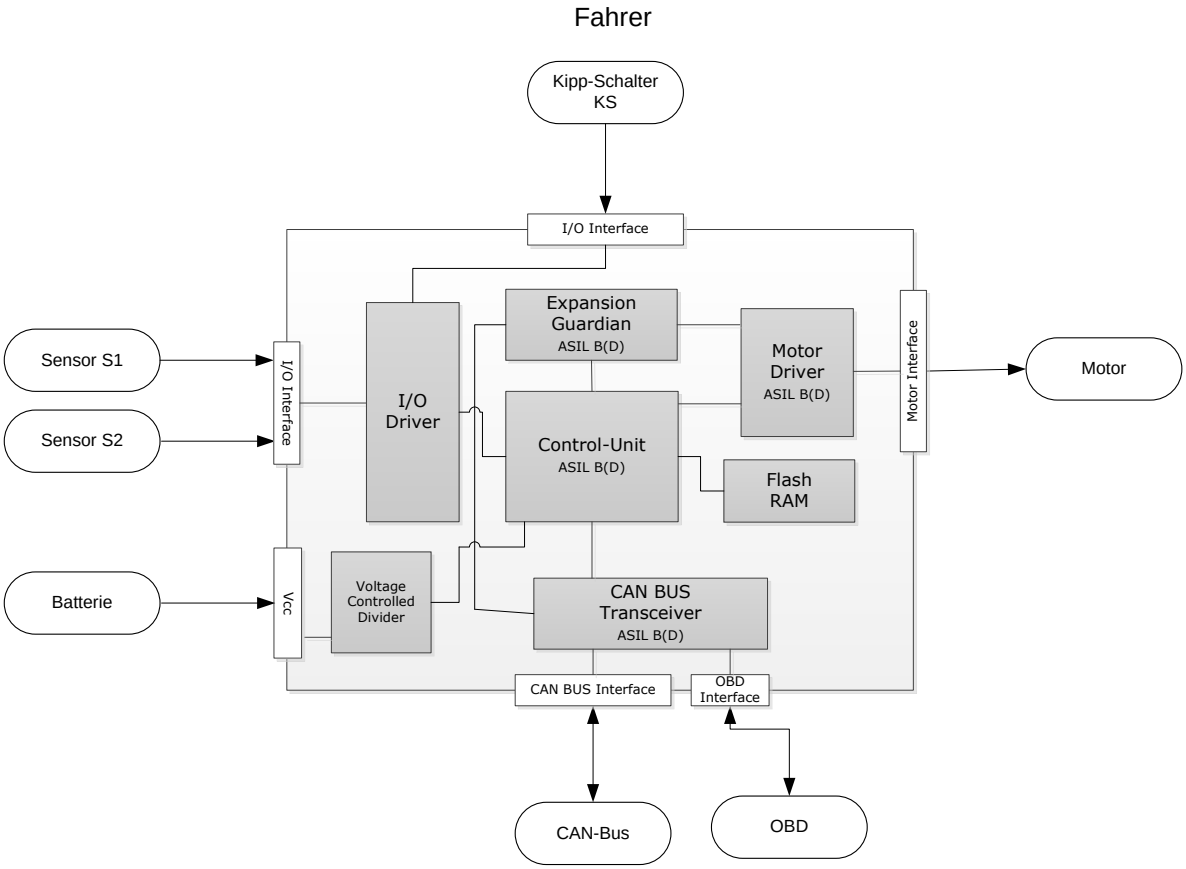


Abbildung 64: System Design des SSCS

Die Aufgaben der Control Unit (CU) und des Expansion Guardian (EG) sind nun folgendermaßen aufgeteilt: Der Side-Stand kann nur ausfahren, wenn der neu eingeführte EG dies erlaubt und die CU das Kommando dazu gibt. So kann der Side-Stand nur ausfahren, wenn zwei voneinander unabhängige Elemente dies zulassen. Auf diese Weise lässt sich der ASIL auf diese Sub-Systeme aufteilen.

Die CU ist nun das komplexere der beiden Elemente, da sie die gesamte Steuerungslogik beinhaltet. Der EG ist ein simples Sub-System, welches lediglich den Status des Motorrads abfragen muss und dann das Ausfahren des Side-Stands verhindert oder erlaubt. So kann der (komplexeren) CU ein niedrigerer ASIL zugeordnet werden, was den Entwicklungsaufwand reduziert. Damit werden im Sinne der Hazard Prevention mögliche endogene Hazards mit dem gekapselten EG *verhindert*. Somit wird im Sinne des Inherent System Safety Approach die Sicherheit in das Design hineinentwickelt.

System Requirements den Sub-Systemen zuteilen [SyD_BA3]

Im nächsten Prozessschritt werden System Requirements den Sub-Systemen zugeteilt und wenn nötig noch genauer ausspezifiziert. Wichtig dabei ist, dass die zugeteilten „Requirements on Elements (RoE)“ bzw. „Safety Requirements on Elements (SRoE)“ soweit als möglich gekapselt bleiben und wenn nötig durch entsprechende Trigger operationalisiert werden. Können sie nicht

operationalisiert werden, so müssen sie als Pre- oder Post-Conditions anderen funktionalen System Requirements zugeordnet werden. Die nachfolgenden Tabellen zeigen die Zuordnung der System Requirements zu den beiden Sub-Systemen CU und EG. Die logische Verknüpfung im spezifizierten Requirement wird anhand von Großbuchstaben, am Beginn der Requirement-Abkürzung angefügt. Dadurch bekommen die zugeordneten Requirements der Sub-Systeme (RoE bzw. SRoE) eine neue Identifikationsbezeichnung (ID) und können anhand der neuen ID eindeutig zugeordnet werden. Dies ist auch notwendig, um die Rückwärts-Traceability sichtbar zu machen. Diese sogenannten „Naming Conventions" fallen in den Aufgabenbereich des Konfigurationsmanagements und werden in der Praxis meist von einem Tool realisiert, daher wird in der weiteren Folge nicht näher darauf eingegangen.

Als erstes werden nun die funktionalen System Requirement FuSyR_1 „SS ausfahren" und FuSyR_2 „SS einfahren" der CU zugewiesen und entsprechend für das Sub-System CU ausspezifiziert, wie in Tabelle 42 ausgeführt.

SyR-ID	ASIL	Requirement on Element (RoE)	Allocation	RoE-ID
FuSyR_1		SS ausfahren Wenn die CU „SS ausfahren" detektiert, muss sie den Motortreiber in Richtung „SS ausfahren" ansteuern, bis Sensor S1 „SS unten" detektiert wird.	Control Unit	CU_FuSyR_1
FuSyR_2		SS einfahren Wenn die CU „SS einfahren" detektiert, muss sie den Motortreiber in Richtung „SS einfahren" ansteuern, bis Sensor S2 „SS oben" detektiert wird.	Control Unit	CU_FuSyR_2

Tabelle 42: Zuteilung der funktionale System Requirements FuSyR_1 und FuSyR_2 (SSCS)

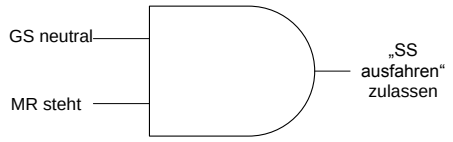
Als nächstes werden die nicht-funktionalen System Requirements zugewiesen, siehe dazu Tabelle 43. Das nicht-funktionale System Requirement nFuSyR_1 wird über die OBD Hardware-Schnittstelle durch den CAN-Bus Befehl „SW-Update" getriggert und wird dadurch operationalisiert und der CU zugeordnet. Damit wird es vom nicht-funktionalen zum funktionalen System Requirement CU_FuSyR_3. Die Diagnoseabfrage (nFuSyR_2) und der End-of-Line Test (nFuSyR_3) werden hier auch operationalisiert. Das nicht-funktionale System Requirement nFuSyR_4 für den Verpolungschutz wurde bereits in Hardware realisiert.

SyR ID	ASIL	Requirement on Element (RoE)	Allocation	SW-Requ.
nFuSyR_1		SW-Update Wenn die CU über den CAN-Bus den Befehl „SW-Update" detektiert, muss die CU den Daten-Transfer starten.	Control Unit	CU_FuSyR_3

nFuSyR_2		Diagnose-Abfrage Wenn die CU über den CAN-Bus den Befehl „Diagnose-Abfrage“ detektiert, muss die CU die Diagnose-Daten übertragen.	Control Unit	CU_FuSyR_4
nFuSyR_3		End-of-Line Test anfordern Wenn die CU über den CAN-Bus den Befehl „End-of-Line Test“ detektiert, muss die CU den Test durchführen und das Ergebnis über den CAN-Bus übertragen.	Control Unit	CU_FuSyR_5
nFuSyR_4		Verpolungsschutz	Hardware-Lösung	n.a.

Tabelle 43: Zuteilung der nicht-funktionalen System Requirements nFuSyR_1 bis nFuSyR_4 (SSCS)

Als nächstes werden die Safety Requirements den Sub-Systemen zugeteilt und damit als Safety Requirements on Element (SRoE) bezeichnet, siehe dazu Tabelle 44. Das nicht-funktionale Safety Requirement nFuSaR_1, welches den Motorrad-Status abfragt, wird auf die CU_SaR_1.1 „SS ausfahren zulassen“ und auf EG_SaR_1.1 „SS ausfahren freigeben“ aufgeteilt. Beide SRoE werden durch zyklische interne Trigger operationalisiert. Damit wird der Motorrad-Status, sowohl von der CU als auch von der EG, automatisch und laufend überwacht. Durch die zyklische Abfrage des MR-Status kann die Überwachungsroutine in der CU entkoppelt laufen und limitiert damit die Komplexität des CU_FuSyR_1 „SS ausfahren“.

SaR ID	ASIL	Safety Requirement on Element (SRoE)	Allocation	SW-Requ.
nFuSaR_1	B(D)	SS ein- und ausfahren zulassen Wenn die CU über den CAN-Bus „GS neutral“ UND „MR steht“ detektiert, soll das Ausfahren zugelassen werden.  Die Abfrage wird zyklisch von einem internen Trigger der CU gestartet.	Control Unit	CU_SaR_1.1

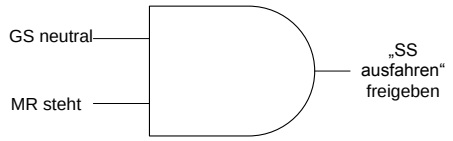
	B(D)	<u>SS ausfahren freigeben</u> Wenn der EG über den CAN-Bus „GS neutral“ UND „MR steht“, soll der EG das Ausfahren freigeben.  Die Abfrage wird zyklisch von einem internen Trigger des EG gestartet.	Expansion Guardian	EG_SaR_1.1

Tabelle 44: Zuteilung des nicht-funktionalen Safety Requirements nFuSaR_1 (SSCS)

Die nachfolgende Tabelle 45 zeigt die Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_2. Das nFuSaR_2 „Endposition des SS detektieren“ wird durch das Einschalten der Zündung und/oder durch die Befehle Ein- oder Ausfahren getriggert und damit als SRoE für die CU operationalisiert.

SaR ID	ASIL	Safety Requirement on Element (SRoE)	Allocation	SW-Requ.
nFuSaR_2	B	<u>Endposition des SS detektieren</u> Wenn die CU, das Einschalten der Zündung detektiert ODER „SS einfahren“ ODER „SS ausfahren“ detektiert, muss sie die Endposition der Sensoren S1 und S2 abfragen. Kann keine der beiden SS Endposition detektiert werden, muss sie eine „Warnung“ UND ein „Losfahrverbot“ über den CAN-Bus an das MR senden UND die Position halten.	Control Unit	CU_SaR_2.1

Tabelle 45: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_2 (SSCS)

Das nicht-funktionale Safety Requirement nFuSaR_3 (siehe dazu Tabelle 46) fordert, dass das Einfahren des Side-Stands verhindert werden muss, wenn die Zündung ausgeschalten ist. Hier wurden eine Hardware- und eine Software-Lösung definiert. Bei der Hardware-Lösung wird mit dem Ausschalten der Zündung die Versorgung des Motor-Drivers abgeschaltet und nur die Versorgung für den Sleep-Mode der CU und des EG aufrechterhalten. Damit kann die H-Bridge den Motor nicht mehr ansteuern. Durch den EG wird zusätzlich noch das Ein- bzw. Ausfahren gesperrt, das durch den Sleep-Mode getriggert wird.

SaR ID	ASIL	Safety Requirement on Element (SRoE)	Allocation	SW-Requ.
nFuSaR_3	QM	Status Zündung abfragen Wenn sich der EG im Sleep-Mode befindet, muss er das Ein- oder Ausfahren sperren.	Expansion Guardian	EG_SaR_3.1

Tabelle 46: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_3 (SSCS)

Das nFuSaR_4 „Versorgungsspannung überwachen“ wird durch die Hardware mit dem Sub-System „Voltage Controlled Driver“ gelöst, siehe dazu Tabelle 47.

Eine weitere Lösung wurde aufgrund der Interface-Analyse gefordert. Der Ausfall des Voltage Controlled Dividers (VCD) soll zusätzlich überwacht werden. Hier wird durch einen internen Trigger in der CU das nFuSaR_4 operationalisiert. Die CU soll nun zyklisch den VCD abfragen, ob seine Funktionsfähigkeit noch gewährleistet ist. Das Design des VCD muss diesbezüglich konstruiert werden und eine Leitung dafür zur Verfügung stellen. Ist die Funktionsfähigkeit nicht gegeben, sendet die CU über den CAN-Bus eine Warnung und hält die Side-Stand Position.

SaR ID	ASIL	Safety Requirement on Element (SRoE)	Allocation	SW-Requ.
nFuSaR_4	B	Versorgungsspannung überwachen Wenn die CU keine definierten Signale vom „Voltage Controlled Divider“ detektiert, muss sie den SS im aktuellen Zustand halten und über den CAN-Bus eine Warnung ausgeben.	Control Unit	CU_SaR_4.1

Tabelle 47: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_4 (SSCS)

Das nicht-funktionale Safety Requirements nFuSaR_5 kann durch eine Post-Condition abgedeckt werden, siehe dazu Tabelle 48. Dieses Requirement wird mit dem Befehl „SS einfahren“ bzw. „SS ausfahren“ wirksam. Dabei wird ein interner Timer aufgezogen und während der Funktion Ein- bzw. Ausfahren zyklisch abgefragt. Läuft der Timer ab, so wird der Side-Stand in seine ursprüngliche Position gebracht und über den CAN-Bus eine Warnung ausgegeben. Auch hier ist der EG über das SRoE EG_SaR_2.2 aktiv.

SaR ID	ASIL	Safety Requirement on Element (SRoE)	Allocation	SW-Requ.
nFuSaR_5	QM	Wenn die CU den Motortreiber in Richtung „SS ausfahren“ ODER „SS einfahren“ ansteuert, muss die CU einen internen 5 Sekunden Timer starten und die Aus- bzw. Einfahrzeit überwachen. Wird der entsprechende Sensor innerhalb von 5 Sekunden nicht erreicht, muss sie den SS wieder einfahren UND über den CAN-Bus eine „Warnung“ ausgeben.	Control Unit	*Post-Condition zu CU_FuSyR_1*

Tabelle 48: Zuteilung der nicht-funktionalen Safety Requirements nFuSaR_5 (SSCS)

Aus den spezifizierten Requirements on Elements sind die Anforderungen an die CU sowie an den EG nun klar ersichtlich. Durch die Operationalisierung und Kapselung der Funktionalen Requirements wird die Komplexität der Steuerfunktionen reduziert und übersichtlicher. Damit werden systematische Fehler, in diesem Fall Human Errors, Faults in Artefakts und SW Faults, weitgehend vermieden. Mit diesen Maßnahmen wird die inhärente Systemsicherheit erhöht.

System Design verifizieren [SyD_BA4]

Bei der Verifizierung des SyD wurde jedes spezifizierte Requirement anhand des nun aktualisierten System Designs verifiziert. Damit wurde das System Design auf mögliche inhärente Fehler analysiert. Das SyD wird mit positivem Verifikations-Ergebnis zur Preliminary System Safety Evaluation (PSSE) freigegeben.

PSSE durchführen [SyD_BA5]

Die PSSE ist im nächsten Kapitel 5.2.5 ausführlich beschrieben.

System Design freigeben und kommunizieren [SyD_BA6]

Der SyD wird nach Einarbeitung der DSaR, einer nochmaligen Iteration im Modellierungsraum und einem positiven Review als „Inherent Safe System Design“ freigegeben.

5.2.5 Side-Stand: Preliminary System Safety Evaluation (PSSE)

Im ersten Prozessschritt der PSSE wird das System Design (SyD) evaluiert [PSSE_BA1]. Bei dieser Evaluierung werden alle Safety Requirements und die darauf basierenden Design-Lösungen überprüft. Danach wird das SyD selbst mehreren Safety-Analysen [PSSE_BA2] unterzogen. Wird dabei das SyD als inhärent sicher eingestuft und ist der ASIL fixiert, werden die Safety-Integritäts-Anforderungen, in diesem Fal die Safety Integrity Requirements, spezifiziert [PSSE_BA3]. Erfüllt das SyD alle Anforderungen und sind keine Änderungen mehr nötig, wird das SyD validiert [PSSE_BA4]. Dabei wird überprüft, ob das SyD die spezifizierten Safety Goals erfüllt und ob alle Safety Requirements im SyD wirksam umgesetzt wurden. Anschließend kann die Safety-Planung fertiggestellt [PSSE_BA5] werden. Die Erweiterung des Safety Cases [PSSE_BA6] und der PSSE-Report [PSSE_BA7] schließen die Preliminary System Safety Evaluation ab. Die nachfolgende Tabelle 49 gibt einen Gesamtüberblick über die 7 Basis-Aktivitäten, die in der PSSE gefordert sind.

PSSE_BA1	System Design Lösungen evaluieren
PSSE_BA2	Safety-Analysen durchführen
PSSE_BA3	Safety Integrity Requirements spezifizieren
PSSE_BA4	System Design validieren
PSSE_BA5	Safety-Planung fertigstellen

PSSE_BA6	Safety Case weiterführen
PSSE_BA7	PSSE-Report verfassen und kommunizieren

Tabelle 49: Basis-Aktivitäten der PSSE im SSCS

System Design Lösungen Evaluieren [PSSE_BA1]

Die nachfolgende Tabelle 50 zeigt einen Ausschnitt aus der Evaluierung der System Design-Lösung, nach dem Vorgehen wie sie im Kapitel 3.2.5 genauer beschrieben ist. Das komplette Ergebnis ist im Anhang C dargestellt. Die Tabelle zeigt die Traceability vom Hazard über das Safety Goal und den Safety Requirements bis hin zur System Design Lösung. Darüber hinaus werden in der Tabelle noch die den Sub-Systemen zugeordneten Safety Requirements on Elements (SRoE) dargestellt. Die Methodik überprüft und stellt sicher, dass für jeden Hazard eine Design-Lösung vorhanden ist.

Hazard	**Safety Goal**	**Safety Requirement**	**Design-Lösung**	**Safety Requirement on Elelement**
Ha_1: Unbeabsichtigtes Ausfahren des SS während der Fahrt [TLHa1].	**SaGo_1:** SS darf während der Fahrt nicht ausfahren	**nFuSaR_1:** Das SSCS darf das Ausfahren des SS nur erlauben, wenn über das Bus-System [eIntRq_1] gemeldet wird: - MR steht UND - GS ist in neutraler Stellung [TLSaR1]	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Transceiver - EG - CU	**Expansion Guardian** **EG_SaR_1.1** Wenn der EG über den CAN-Bus „GS neutral“ UND „Motorrad steht“ detektiert, soll der EG das Ausfahren freigeben. **Control Unit** **CU_SaR_1.1** Wenn die CU über den CAN-Bus „GS neutral“ UND „MR steht“ detektiert, soll das funktionale SW-Requirement CU_FuSyR_1 zugelassen werden.

Tabelle 50: Hazard-Traceability-Matrix des SSCS

Safety-Analysen durchführen [PSSE_BA2]

Voraussetzung für die Durchführung der Safety-Analysen auf Basis des SyDs ist, dass die Traceability jedes Hazards über das Safety-Goal und den Safety Requirements, bis hin zur Design-Lösung nachgewiesen ist. Nur so kann eine effektive Hazard-Analyse durchgeführt werden. Ist hier keine Durchgängigkeit sichtbar, so besteht die Gefahr, dass die Analysen auf falschen Design-Lösungen angesetzt werden. Ist die Traceability komplett nachgewiesen, kann eine wirksame Hazard-Analyse durchgeführt werden. Dabei wird das SyD nach den drei konzeptionellen Sichtweisen nach Ropohl [Rop_09] analysiert. Damit lässt sich das systematisch-methodische Vorgehen auch hier nahtlos fortsetzen. Dieser Prozessschritt ist in 5 Teilschritte unterteilt:

1. Funktionales System-Konzept analysieren [PSSE_BA2.1]
2. Strukturales System-Konzept analysieren [PSSE_BA2.2]
3. Hierarchisches System-Konzept analysieren [PSSE_BA2.3]
4. Contributory Hazards identifizieren [PSSE_BA2.4]
5. Derived Safety Requirements definieren [PSSE_BA2.5]

Im ersten Teilschritt wurden die im System Design Prozess operationalisierten System Requirements CU_FuSyR_3 (Software update), CU_FuSyR_4 (Diagnose Software) und CU_FuSyR_5 (End-of-Line Test) einer funktionalen Safety-Analyse unterzogen [PSSE_BA2.1]. Dabei wurden keine weiteren Hazards identifiziert, deshalb wird auf die Ergebnisse dieser Analyse hier auch nicht näher eingegangen.

Im zweiten Teilschritt wird das Strukturale System-Konzept, also die physikalischen Sub-Systeme des SyD, analysiert [PSSE_BA2.2]. Hier kommt eine induktive Analyse-Methode zum Einsatz, in diesem Fall wurde die FMEA angewendet. Damit können Ausfälle der definierten Sub-Systeme analysiert werden. Der unmittelbare Effekt dieser Ausfälle wird dabei auf Sub-System-Ebene dargestellt sowie der Effekt auf Gesamtsystem-Ebene prognostiziert. Die nachfolgende Tabelle 51 zeigt die Analyse-Ergebnisse.

#	HW-Element	Failure	Unmittelbarer Effekt auf Sub-System-Ebene	Effekt auf Gesamtsystem-Ebene	Kommentar/ Maßnahmen
1	CU	HW-Pin der CU ist defekt”	HW-Pin: Ausfahren ist immer log. “0” (Ausfahren ist gesperrt)	Ausfahren des SS kann nicht durchgeführt werden	Kein Hazard
2			HW-Pin: Ausfahren ist immer log. “1” (Ausfahren bewirken)	Steht das MR und ist die GS in neutraler Stellung, fährt der SS aus und fährt wieder ein, wenn die GS wieder eingelegt wird.	Zeigt auf SSCS-Ebene keine Wirkung, wenn das MR fährt bzw. die GS in neutraler Stellung ist, da das Ausfahren durch den EG verhindert wird.
3		CAN-Bus Pin funktioniert nicht	CU kann vom CAN-Bus keine Daten empfangen und keine	SS kann mit MR-nicht kommunizieren. Warnung und	SSCS muss den SS-Status halten

			Daten senden.	Losfahrverbot kann nicht an das MR gesendet werden	
4	EG	HW-Pin des EG ist defekt	HW-Pin: „Ausfahren freigeben“ ist immer auf log. „0“ (Ausfahren ist blockiert)	EG blockiert das Ausfahren ständig	
5			HW-Pin: „Ausfahren freigeben“ ist immer auf log. „1“ (Ausfahren freigegeben)	Einfahren funktioniert nicht	EG gibt das Ausfahren immer frei, trotzdem wird der SS nicht ausgefahren, weil die CU das Ausfahren nur zulässt, wenn über den CAN-Bus „MR steht und GS in neutraler Stellung“ kommuniziert wird.
6		CAN-Bus Pin funktioniert nicht	EG kann vom CAN-Bus keine Daten empfangen und keine Daten senden.	SS kann mit MR-nicht kommunizieren.	EG muss das Ausfahren blockieren
7	Motor Driver	Motor Driver ist defekt	Output Pins sind inaktiv	Motor wird nicht angesteuert	
8			Output Pins steuern den Motor ständig an	Motor Driver fährt SS während der Fahrt aus	ASIL B Hardware einsetzen
9	CAN-Bus Treiber	CAN-Bus Treiber funktioniert nicht	Siehe #3 und #6	Siehe #3 und #6	Siehe #3 und #6
10	Voltage Controlled Divider	Stabilisierung der Versorgungs-spannungen funktionieren nicht	Versorgungsspannung ist instabil	SS fährt während der Fahrt aus	ASIL B Hardware einsetzen

Tabelle 51: FMEA des SSCS

Bei der Anwendung der FMEA ist ersichtlich, dass keine neuen Hazards mehr identifiziert werden konnten. Hier wird auch deutlich sichtbar, wie wirksam die Safety-Analysen der vorherigen Phasen waren. Da sich die FMEA nur auf das hierarchische System-Konzept beschränkt, konnten sehr strukturiert mögliche Hardware-Fehlerursachen für den schon bekannten Hazard „SS fährt während der Fahrt unerwünscht aus“ identifiziert werden. Das sind in diesem Fall die beiden Sub-Systeme Motor Driver und Voltage Controlled Divider. Beschränkt man sich bei der Definition der Safety-

Maßnahmen auf die klassische Vorgehensweise der Safety-Normen, so ist für die anzuwendende Hardware die definierte Ausfallsrate für ASIL B aus der ISO 26262, für die beiden Sub-Systeme, festzulegen.

Im dritten Teilschritt wird das Hierarchische System-Konzept analysiert [PSSE_BA2.3]. Dabei wird eine deduktive Analyse-Methode angewendet, bei der, ausgehend von den gefährlichen Ereignissen, zu möglichen Fehlerursachen analysiert wird. Hierfür wurde die Fault Tree Analyse (FTA) angewendet. In der nachfolgenden Abbildung 65 ist der wichtigste Fault Tree modelliert. Also auch hier der Hazard des unerwünschten Ausfahrens während der Fahrt, der das Safety Goal SaGo_1 verletzt.

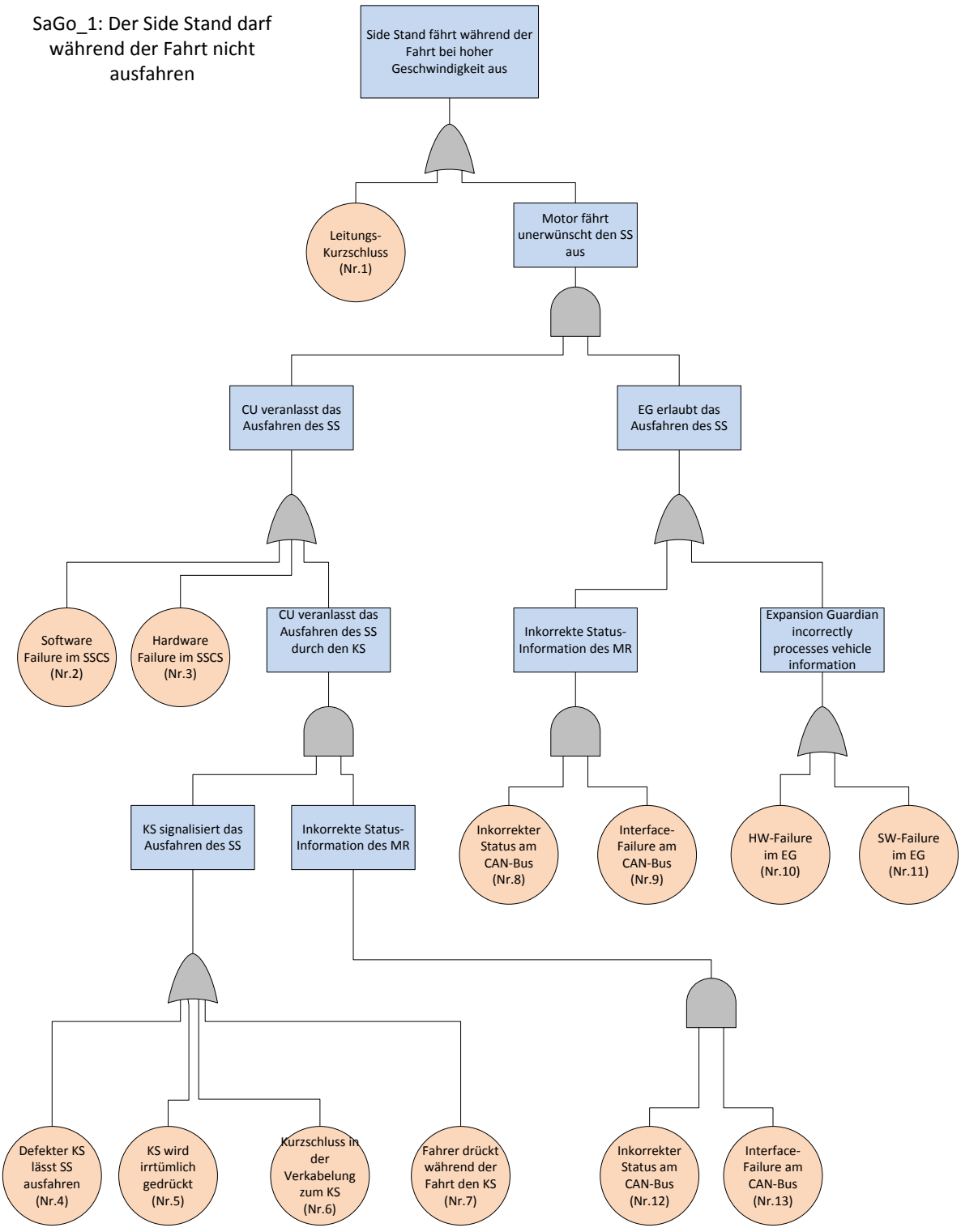


Abbildung 65: Fault Tree des SSCS der die Verletzung des SaGo_1 modelliert

Der Fault Tree ist nun folgendermaßen aufgebaut: Eine unmittelbare Ursache wäre, wenn der Motor direkt durch einen Leitungskurzschluss im Hardware-Teil angesteuert wird und der Side-Stand ausfährt. Die zweite unmittelbare Ursache wäre, dass das SSCS den Motor von sich aus ausfährt. Dazu müssen zumindest zwei Fehler passieren, nämlich, dass die CU das Ausfahren bewirkt und der EG das Ausfahren erlaubt. Das kann hier nicht nur durch Software-Fehler, sondern auch durch Hardware-Fehler verursacht werden. Im Falle von Hardware-Fehlern können das auch die in der FMEA identifizierten defekten Output-Pins der beiden Steuerelemente CU und EG sein.

Der rein qualitative Fault Tree, wie er hier dargestellt ist, ermöglicht nun eine Diskussion zu dem vorgeschlagenen Design und stellt *Single Points of Failure* dar. Der einzige Single Point of Failure ist die direkte Ansteuerung des Motors und die Möglichkeit für Kurzschlüsse. Somit ist ersichtlich, dass diese Ansteuerung ein potentieller Schwachpunkt ist. Es ist nötig, diese Ansteuerung so zu konzipieren, dass die Wahrscheinlichkeit für einen Kurzschluss möglichst gering ist.

In der zweiten Ebene werden die Fehlerkombinationen sichtbar. Hier ist es die Kombination EG und CU, deren Wahrscheinlichkeit zwar als gering eingeschätzt wird, jedoch sehr gefährlich ist. Diese gefährliche Situation wird im nächsten Teilschritt noch weiter analysiert.

Im vierten Teilschritt wird das SyD auf Contributory Hazards (C-Hazards) untersucht [PSSE_BA2.4]. Als erstes werden prognostizierte Fehlfunktionen der SRoE analysiert. Im Beispiel des Side-Stands liegt die Konzentration auf dem, in der FTA identifizierten Single Point of Failure. Mit der FTA ist auch die Fehlerkombination in der zweiten Ebene als gefährlich identifiziert und soll daher noch intensiver untersucht werden. Als erstes werden die *Safety Requirements on Element* CU_SaR_1.1 und EG_SaR_1.1 auf Fehlermöglichkeiten analysiert. Dabei werden die Verben im jeweiligen SRoE negiert, um damit auf Sub-System-Ebene Fehlfunktionen prognostizieren zu können, siehe dazu die nachfolgende Tabelle 52.

Rq-ID.	**Safety Requirement on Element (SRoE)**	**Fehlfunktion**
CU_SaR_1.1	SS ausfahren zulassen Wenn die CU über den CAN-Bus „GS neutral“ UND „MR steht“ detektiert, soll das Ausfahren zugelassen werden.	CU bewirkt das Ausfahren immer
EG_SaR_1.1	SS ausfahren freigeben Wenn der EG über den CAN-Bus „GS neutral“ UND „MR steht“ detektiert soll der EG das Ausfahren freigeben	EG gibt das Ausfahren immer frei

Tabelle 52: Fehlfunktionen der SRoE im SSCS

Nun werden die identifizierten Fehlfunktionen, nach dem Modell der Fehlerkette, durchanalysiert. Die nachfolgende Abbildung 66 zeigt die prognostizierte Fehlerfortpflanzung von der Sub-System-

Ebene bis zum SSCS. Mit dem Fehler-Modell wird sichtbar, wie sich Software-Fehler fortpflanzen können.

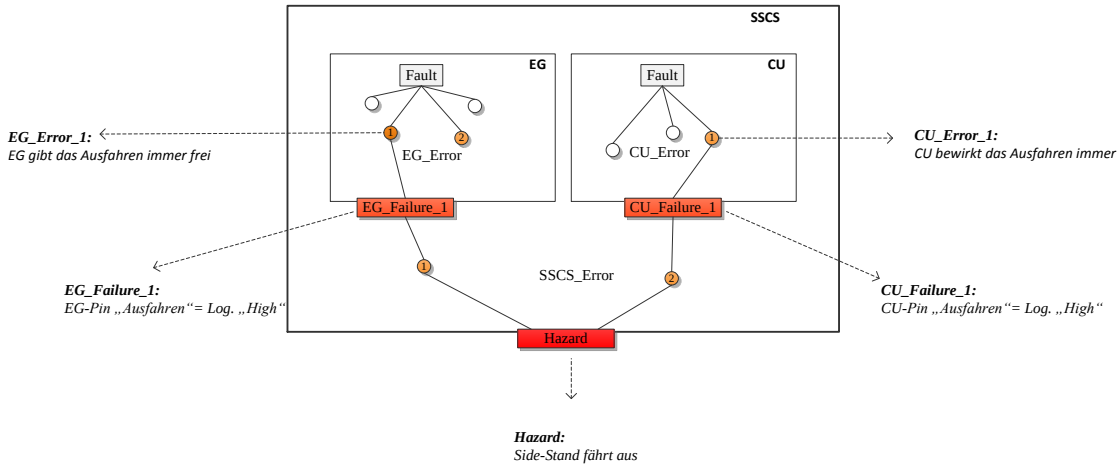


Abbildung 66: SS-Failure-Chain, Contributory Hazard 1 “SS fährt aus” (SSCS)

Anhand dieser Darstellungsform kann der Vorgang folgendermaßen nachverfolgt werden: Durch einen Fault im EG wird EG_Error_1 ausgelöst. Er symbolisiert den internen Zustand des Sub-Systems, der nun das Ausfahren des Side-Stands immer frei gibt. Dieser propagiert an der Systemgrenze des EG zum EG_Failure_1, da dieser den Output-Pin „Ausfahren“ des EG auf logisch „high“ setzt und damit an der Sub-System Grenze sichtbar bzw. messbar wird. Auf der Ebene des SSCS ist dieser jedoch noch nicht sichtbar und propagiert auf dieser Betrachtungsebene zum SSCS_Error_1.

Wird im Sub-System CU das Ausfahren des Side-Stands durch einen Fault bewirkt, so wird damit der CU_Error_1 aktiviert und propagiert sofort zum CU_Failure 1 auf Ebene der CU. Auf Ebene des SSCS propagiert er zum SSCS_Error_2. Durch das Zusammentreffen der beiden Errors SSCS_Error_1 und SSCS_Error_2, zu einem bestimmten Zeitpunkt und über eine bestimmte Zeitspanne, werden diese in ihrer Kombination an der Systemgrenze SSCS sichtbar, da der Motortreiber angesteuert wird. Auf Gesamtsystem-Ebene (Ebene Motorrad) würde nun der Side-Stand plötzlich ausfahren. Das ergibt einen gefährlichen Hazard, hervorgerufen durch die Kombination zweier Failures, der hier mit Contributory Hazard (C-Hazard) bezeichnet wird.

Nun werden als nächstes die beiden gefährlichen Hardware-Fehler #2 und #5 aus der FMEA mitbetrachtet:

- #2: CU HW-Pin „SS ausfahren“ ist immer auf log. „1“
- #5: EG HW-Pin „SS ausfahren freigeben“ ist immer auf log. „1“

Da die beiden *Hardware-Fehler* auch im Fehler-Modell interpretierbar sind, können die unterschiedlichen Fehler-Szenarien damit weiter durchanalysiert werden. Mit dieser Darstellung im Fehlermodell werden die Zusammenhänge der Fehlfunktionen beider Sub-Systeme an deren Sub-Systemgrenzen sowie die Hardware-Fehler in Kombination und deren Auswirkungen am SSCS deutlich erkennbar und nachvollziehbar. Damit wird ersichtlich, dass mit der Anzahl der möglichen

Fehlerursachen und deren Kombinationsmöglichkeiten die Eintrittswahrscheinlichkeit des C-Hazards erhöht wird.

Als nächstes werden die exogenen Einflüsse auf Basis des Shell-Modells und der Ergebnisse der exogenen Hazard-Analyse identifiziert. Als erstes wird die Z-Shell betrachtet, also *die unmittelbare System-Umgebung* und deren gefährlichen Einflüsse auf das SSCS. Hier werden die Ergebnisse der exogenen Hazard Analysis aus dem Problemraum herangezogen. Das sind beim Side-Stand Beispiel der Kippschalter und die elektrischen Leitungen zum Kippschalter, die bei einem Kurzschluss das Ausfahren des Side-Stands bewirken. Als nächstes wird die S-Shell, also die *mittelbare System-Umgebung* analysiert. Hier wurde der Forseeable Misuse identifiziert, also der Fahrer, der den Kipp-Schalter bewusst drückt. Aus der E-Shell wurden keine weiteren gefährlichen Einflüsse identifiziert. Betrachtet man diese Einflüsse in Kombination mit den bereits identifizierten Fehlern, so wird die Wahrscheinlichkeit für das Auftreten des C-Hazards noch weiter erhöht.

Bevor nun auf den nächsten Teilschritt [PSSE_BA2.5] eingegangen wird, soll der Zusammenhang zu den System Accidents noch diskutiert werden. System Accidents haben nach Leveson [Lev_03] ihre Ursachen in den Flawed Requirements. Diese Flawed Requirements rufen gefährliche inhärente Systemzustände hervor, die an der betrachteten Systemgrenze überwiegend *nicht* sichtbar sind. Erst wenn sie durch eine quasi-zufällige Kombination so zusammentreffen, sodass eine Failure hervorgerufen wird, zeigt diese an der Systemgrenze ihre Wirkung. Ist diese Failure gefährlich, so handelt es sich um einen C-Hazard. Da durch die Anwendung der klassischen Analyse-Methoden nur Hazards identifiziert werden können und davon nur ein geringer Anteil, im Sinne der Hazard Avoidance, vermieden werden kann, ist noch ein hohes Potential für C-Hazards vorhanden. Dies begründet sich darin, dass hinter einer Hazard Prevention oder einem Hazard Controlling meist weitere Elemente und Mechanismen notwendig sind, die das Potential für zusätzliche Fehler in der zweiten Ebene bedeuten. Damit wird ersichtlich, dass das aus Normensicht „ausreichend sichere System“ das Potential für System Accidents besitzt.

Betrachtet man nun die gegebene Situation des SSCS, so wird deutlich sichtbar, dass auch schon einfache technische Systeme das Potential für System Accidents haben. Auch die Motive dafür, wie das Zusammentreffen fehlerhafter Elemente und die geringere Eintrittswahrscheinlichkeit, trifft auf das SSCS zu.

Mit der systematischen Abarbeitung der ersten drei Safety-Analysen und der darauffolgenden Contributory Hazard Analysis konnte das Potential für einen System Accidents sichtbar gemacht werden. Die Systematik bei der Durchführung dieser Analysen, zuerst induktiv und dann deduktiv, deckten weitere mögliche Ursachen auf. Mit der nachfolgenden CHA konnten die wichtigsten davon, die einen System Accident hervorrufen können, ausfindig gemacht werden. Die Herausforderung des nächsten Teilschrittes ist nun, dafür entsprechende Derived Safety Requirements so zu definieren [PSSE_BA2.5], dass gezielt Lösungsmöglichkeiten eingeleitet werden können, welche die Auftrittswahrscheinlichkeit des C-Hazards verringert.

Die Fehlerursachen, die einen System Accident auslösen können, sollen nun nach Möglichkeit vermieden (Fault Avoidance) werden. Auch hier steht die „Avoidance“ im Vordergrund. In diesem Fall handelt es sich jedoch um eine Fault Avoidance und nicht um die Hazard Avoidance, da bei einem C-Hazard immer die Kombination mehrere Fehler wirkt. Daher kann mit der Fault Avoidance

nur die Eintrittswahrscheinlichkeit des C-Hazards verringert werden. Ist eine Fault Avoidance nicht möglich, so kommt als nächstes die Failure Detection zur Anwendung, mit der Failure detektiert werden und durch zusätzliche Maßnahmen die Wahrscheinlichkeit des C-Hazards verringert werden soll. Bei derartigen Maßnahmen muss die Effektivität bewertet werden, da damit ein weiteres Fehler-Potential hervorgerufen wird. Das heißt, es ist immer zu untersuchen, ob das Potential für einen C-Hazard damit auch wirklich verringert und nicht erhöht wird.

Für das SSCS wurden folgende Lösungen gewählt:

- **SAC_5:** Der Kippschalter des Side-Stands muss am Motorrad so positioniert sein, dass er während der Fahrt nicht gedrückt werden kann.
- **SAC_6:** Die Leitungen des Kippschalters müssen am Motorrad so geführt werden, dass kein Kurzschluss durch Quetschen verursacht werden kann
- **HWRq_1:** Die Leitungsführung für die Kippschalter-Signale muss auf der Leiterplatte so gewählt werden, dass kein Kurzschluss möglich ist.
- **HWRq_2:** Die Hardware-Architektur muss schaltungstechnisch so konzipiert sein, das bei einem Hardware-Fehler (FMEA: #2, #5) der EG bzw. die CU den Fehler detektiert und ein Ausfahren während der Fahrt vermeidet. Eine Warnung wird an das Motorrad übermittelt.
- **HWRq_3:** Der Voltage Controlled Divider muss eine Leitung zum EG führen über der die Funktionstüchtigkeit übermittelt wird. Ist die Funktionstüchtigkeit des VCD nicht gegeben, muss das Ausfahren gesperrt bleiben.
- **CU_nFuSyR_1.0:** Die Software Architekturen der CU muss modular aufgebaut sein und einer Contributory Hazard Analysis unterzogen werden.
- **EG_nFuSyR_1.0:** Die Software Architekturen des EG muss modular aufgebaut sein und einer Contributory Hazard Analysis unterzogen werden.
- **CU_nFuSyR_2.0:** Alle detektierten Fehler sind im Fehlerspeicher des SSCS abzulegen

Mit den beiden Safety Application Conditions SAC_5 und SAC_6 sowie mit dem Hardware Requirement HWRq_1 kann eine Fault-Avoidance erzielt und damit die Eintrittswahrscheinlichkeit des C-Hazards um mehr als 50% reduziert werden.

Mit dem Hardware Requirement HWRq_2 wird eine inhärent sichere Hardware angestrebt. Ein Lösungsvorschlag wird im Anhang D: „SSCS Hardware-Lösungsvorschlag für den HWReq_2“. Mit dem Hardware Requirement HWRq_3 soll das Ausfahren des Side-Stands während der Fahrt verhindert werden, wenn der Voltage Controlled Divider ausfällt. Darüber hinaus ist mit definierten Requiements on Elements CU_nFuSyR_1.0 und EG_nFuSyR_1.0 der modulare Aufbau der Software Architektur gefordert, die ebenso einer CHA unterzogen werden sollen. CU_nFuSyR_2.0 fordert, dass alle detektierten Fehler im Fehlerspeicher des SSCS abzulegen sind. Mit den definierten Safety Application Conditions und den Requirements on Element kann die Auftrittswahrscheinlichkeit eines C-Hazards stark eingeschränkt werden.

Safety Integrity Requirements spezifizieren [PSSE_BA3]

Der geforderte Automotive Safety Integrity Level ist beim SSCS ASIL B. Die Safety Integrity Requirements sind in diesem Fall aus ISO 26262 zu entnehmen. Da der Umfang dieser Safety Integrity Requirement sehr groß ist, wird er hier nicht dargestellt. Es ist jedoch zu erwähnen, dass die Anforderungen an das System Design vom Safety-Vorgehensmodell für ASIL B erfüllt wurden.

System Design validieren [PSSE_BA4]

In diesem Prozessschritt soll das System Design von einem Experten-Team, mit dem Kunden und nach Möglichkeit mit dem Zertifizierer validiert werden. Dabei steht die Erfüllung der Safety Goals im Mittelpunkt. Dieses Experten-Review wurden beim Side-Stand Beispiel mit Experten und ExpertInnen der Firma Frequentis durchgeführt. Das System Design wurde als ein Inherent Safe System Design freigegeben.

Safety-Planung fertigstellen [PSSE_BA5]

Da beim SSCS die Safety-Aktivitäten in die Projektplanung integriert werden, ist dieser Prozessschritt nicht erforderlich. Es ist jedoch hier zu erwähnen, dass die erforderlichen Aufwände für die Derived Safety Requirements in die Projektplanung mit aufzunehmen sind.

Safety Case weiterführen [PSSE_BA6]

Der erweiterte Safety Case ist in den nachfolgenden Abbildungen dargestellt. Im ersten Schritt wurde Goal G_0.3 erweitert, das aussagt, dass alle Hazards identifiziert und mitigiert sind. In der nachfolgenden Abbildung 67 ist die „Erweiterte GSN, Goal G_0.3.1 Safety Analysen durchgeführt“ dargestellt.

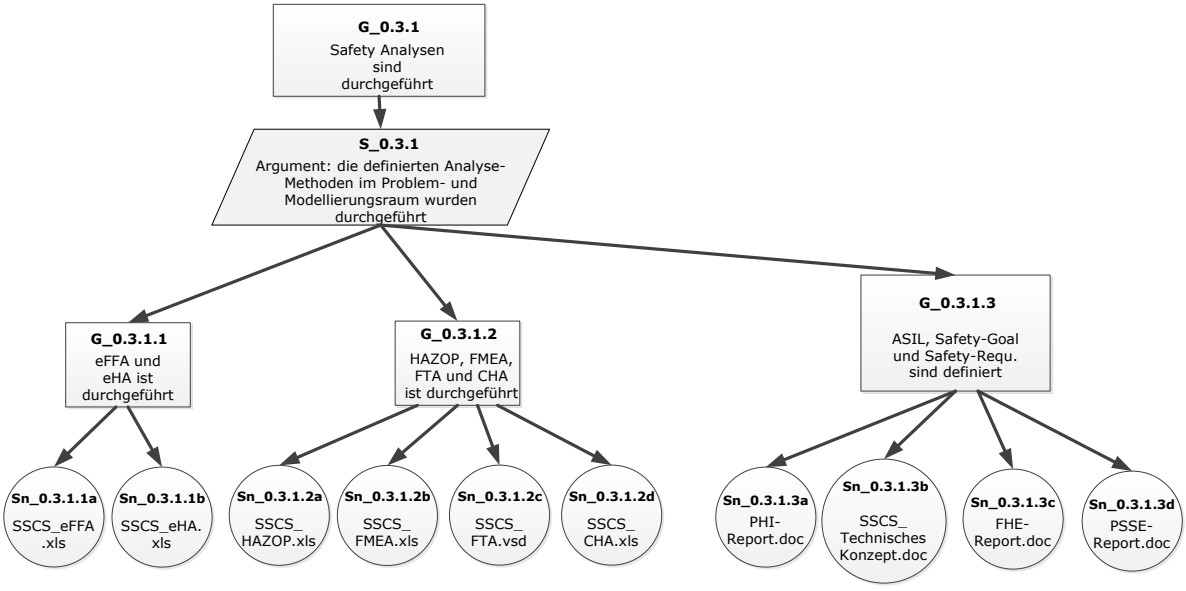


Abbildung 67: Erweiterte GSN, Goal G_0.3.1 Safety Analysen durchgeführt (SSCS)

Die neu durchgeführten Analyse-Methoden sind im Goal G_0.3.1.2 angeführt und durch die Evidenzen Sn_3.1.2a bis Sn_3.1.2d direkt argumentiert. Die Evidenzen Sn_0.3.1.3c „FHE-Report“ und Sn_0.3.1.3d „PSSE-Report“ liefern die Evidenzen für Goal G_3.1.3

Die nachfolgende Abbildung 68 erweitert das Goal G_0.3.2 mit den Evidenzen Sn_0.3.2.1c bis Sn_0.3.2.1f mit den Dokumnetationen der durchgeführten Safety-Analysen, welche im Modellierungsraum durchgeführt wurden.

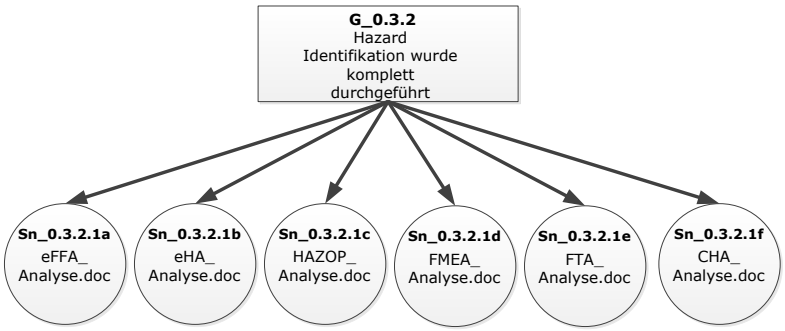


Abbildung 68: Erweiterte GSN, Goal G_0.3.2 Hazard Identifikation komplett (SSCS)

Abbildung 69 zeigt Goal G_0.3.3, dass die identifizierten Hazards mitigiert wurden. Es mag verwundern, dass die Safety Application Conditions SAC_1, SAC_5 und SAC_6 hier nicht als Hazard Avoidance angeführt sind. Das begründet sich damit, dass SACs nicht in den Safety Case des SSCS fallen, sondern auf Gesamt-Systemebene dargestellt werden müssen.

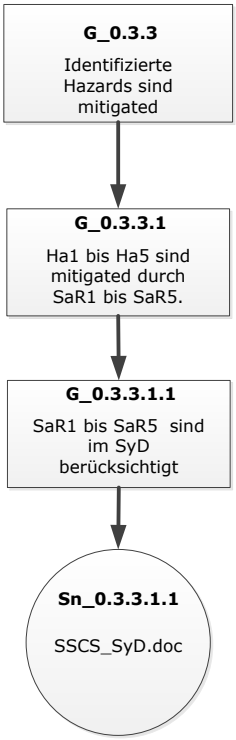


Abbildung 69: Erweiterte GSN, Goal G_0.3.3 Identifizierte Hazards mitigiert (SSCS)

In der nachfolgenden Abbildung 70 ist der Nachweis der Hazard Traceability dargestellt.

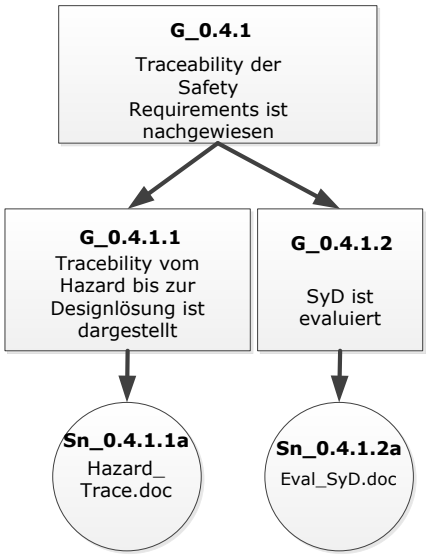


Abbildung 70: Erweiterte GSN, Goal G_0.4.1 Traceability der Safety Requiements (SSCS)

Goal G_0.4.1 ist auf die Sub-Goals G_4.1.1 und G_4.1.2 aufgeteilt. Mit G_4.1.1 wird anhand der Hazard Traceability Matrix (siehe Tabelle 50) nachgewiesen, dass jeder Hazard ein Safety Goal besitzt und vom Safety Goal mindestens ein Safety Requirement abgeleitet wurde. Darüber hinaus zeigt die Traceability Matrix auch noch die Design-Lösung mit dem Safety Requirements on Element. Die Traceability Matrix ist in Sn_0.4.1.1a als Evidence dargestellt. Mit Sub-Goal G_4.1.2 wird das evaluierte System Design dargestellt, womit der Nachweise eines Inherent Safe System Designs erbracht wird.

PSSE-Report verfassen und kommunizieren [PSSE_BA7]

Der PSSE-Report schließt die PSSE ab.

5.2.6 Side-Stand: Abschluss im Modellierungsraum

Im Modellierungsraum wird der Grundstein für das Inherent Safe System gelegt. Alles was in dieser Phase nicht geplant wird, wirkt sich auf den weiteren Projektverlauf und auf die Systemsicherheit aus. Daher ist ein systematisch-methodisches Vorgehen, wie es das Safety-Vorgehensmodell definiert, unbedingt notwendig. Basierend auf der Stakeholder-Identifikation und Stakeholder-Analyse werden alle erforderlichen Anforderungen erhoben. Das systematisch-methodische Vorgehen lässt hier wenig Platz für Flawed Requirements. Bereits in der Phase des Requirements Enginieering werden die ersten Safety-Analysen mit Hilfe der FHE durchgeführt, um die notwendigen Safety Requirements im SyD berücksichtigen zu können. Die Traceability der Anforderungen wird einerseits von der Stakeholderquelle über Planung bis zur Requirements Spezifikation dargestellt. Andererseits wird die Traceability von den Top-Level Hazards über den

ASIL und den Safety Goals bis hin zur Safety Requirements Spezifikation beschrieben. Die Spezifikation der System Requirements legt die Basis für das SyD. Auch in dieser Phase wird die Systematik weiterverfolgt, indem jedes System Requirement systematisch in ein SyD übergeführt wird. Der Aufbau des SyD wurde nach dem Prinzip von Rophol aufgabaut, damit wird einerseits die Möglichkeit einer ASIL-Decomposition geschaffen und andererseits können Safety Requirements gekapselt werden. Danach wurden die System Requirements den Sub-Systemen zugeteilt und verfeinert. In der PSSE wurde das SyD nochmals eingehend analysiert und mit Derived Safety Requirements zu einem Inherent Safe System Design weiterentwickelt.

6. Zusammenfassung und Schlussfolgerungen

Diese Arbeit hat sich zum Ziel gesetzt, ein Safety-Vorgehensmodell zur Konzeption und Entwicklung von sicherheitskritischen Systemen vorzuschlagen. Dabei wurde ein neuer Ansatz gewählt, der sogenannte „Inherent System Safety Approach". Dieser Ansatz gibt dem Safety-Vorgehensmodell wichtige Leitlinien vor. Das Wort „Inherent" drückt dabei aus, dass bei der Durchführung der Safety-Analyse, die *Hazard Avoidance* höchste Priorität hat. Können Hazards nicht vermieden werden; dann sollen sie in gefährlichen Situationen, im Sinne der *Hazard Prevention*, verhindert werden. Ist auch dies nicht möglich, kommt das *Hazard Controlling*, im Sinne der Funktionalen Sicherheit, zum Einsatz. Mit „System Safety" werden die Disziplinen Management und Engineering ausgedrückt. Darüber hinaus wird damit aber auch der Blick auf das Ganze gefordert, sowohl auf der organisatorischen Ebene, mit der PPL, als auch auf der System-Ebene, mit dem Shell-Modell. Auf der System-Ebene ist der „Design-in-Approach" gefordert, also, dass die Sicherheit in das System hineinentwickelt werden soll. Dabei darf allerdings der Blick auf das Ganze nie verloren gehen, was mit dem Begriff „Integrated Sum-of-the-Parts" zum Ausdruck kommen soll. Mit diesem Ansatz soll der Paradigmenwechsel von der Funktionalen Sicherheit zur Inhärenten Systemsicherheit eingeleitet werden.

Oberstes Ziel dieses Ansatzes ist es, den gefährlichen „System Accidents" entgegenzuwirken. Das Safety-Vorgehensmodell soll dabei helfen die potentiellen „Flawed Requirements", Hazards und vor allem Contributory-Hazards in sehr frühen Lifecycle-Phasen zu vermeiden, um damit die Grundlage zur Entwicklung eines „Inherent Safe System Designs" legen zu können. Für die zuverlässige Realisierung solch eines Inherent Safe System Designs sorgt der Projektplanungsprozess der auch Teil des Safety-Vorgehensmodells ist. Damit zeigt das Modell auch seine Wirkung über die Entwicklungsphase hinaus, das Inherent Safe System Design entsprechend in die Realisierung überführen zu können.

Das Safety-Vorgehensmodell selbst hat den Anspruch *ganzheitlich* zu sein. Mit der Projekt-Prozesslandschaft (PPL) schafft das Vorgehensmodell einen ganzheitlichen Überblick über das gesamte Projektvorhaben, mit all seinen dafür erforderlichen Interaktionen. Es ist damit, im Gegensatz zu anderen Modellen, nicht auf einzelne Disziplinen, wie Projektmanagement, Engineering und Safety, ausgerichtet. Das Safety-Vorgehensmodell integriert diese Sichten und betont die Notwendigkeit der Interaktionen zwischen den verschiedenen Disziplinen. Mit Hilfe dieser integrierten Sichtweise und des darauf basierenden, systematisch-methodischen Vorgehens, ist es möglich, Process Flaws weitgehend zu vermeiden. Mit einer ganzheitlichen Systemsicht, welche durch das Shell-Modell ermöglicht wird, ist eine wichtige Grundlage für die

Konzeptionierung des System-Entwurfs gelegt. Mit speziellen Analyse-Methoden werden Ursachen für System Accidents identifiziert, das liefert die Basis zur Entwicklung des Inherent Safe System Designs.

Die Grundlage dafür liefert die „Erweiterte Fehlerkette“ (Kapitel 4.1). Dieses Modell ordnet die verschiedenen Ursachen den Bereichen zu, wo sie entstehen, also der Entwicklungsumgebung des Systems, dem System selbst oder dessen System-Umgebung und -Umwelt. Außerdem wird an der erweiterten Fehlerkette deutlich, dass letztlich alle Fehlerursachen menschlicher Natur sind, die mit einem guten Managementsystem bzw. geeigneten Prozessen in ihrer Wahrscheinlichkeit reduziert werden können. Die erweiterte Fehlerkette führt zu einem besseren Verständnis der Wirkmechanismen, die schlussendlich zu einem System-Accident führen können. In sicherheitskritischen Systemen geht es letztlich um die Vermeidung aller Ursachen, die zu einem Accident, also zu Schaden an Mensch und Umwelt, führen.

Das Shell-Modell stellt das zu betrachtende System in seiner gesamten Umgebung dar, um die Einflüsse auf das System und Wirkmechanismen vom System in die System-Umgebung besser analysieren und verstehen zu können. Es kommt damit dem Anspruch des Inherent System Safety Approaches sehr nahe, möglichst ganzheitlich das System zu betrachten. Das Shell-Modell bietet mit speziell entwickelten Analyse-Methoden die Möglichkeit, Einflüsse von außen zu betrachten, um daraus exogene Hazards ermitteln und vermeiden zu können. Auf dieses Modell und die damit verbundene Anwendungsempfehlung greift das Safety-Vorgehensmodell an mehreren Stellen im Projektablauf zurück. Das Shell-Modell hat das Ziel, eine möglichst vollständige Hazard-Analyse zu ermöglichen.

Mit der Fokussierung auf Konzept und Entwicklung, wird die systematisch-methodische Analyse von den Anforderungen an das zu entwickelnde System verstärkt, um schon zu Beginn mögliche „Flawed Requirements“ zu identifizieren und somit zu vermeiden. Zu diesem Zweck wurden in dieser Arbeit zusätzlich neue Methoden entwickelt und vorhandene erweitert, um diese Analysen überhaupt erst zu ermöglichen. Durch die daraus resultierenden, möglichst korrekten Anforderungen können in der Folge sicherere Designs, ja sogar „Inherent Safe System Designs“ entwickelt werden. Die solcher Art sicheren Designs vermeiden verstärkt viele Ausfälle, Fehlfunktionen und Abweichungen der gewünschten System-Funktionen. Dadurch wird die Anzahl an inhärenten System-Fehlern deutlich reduziert. Dies führt in weiterer Folge zur Vermeidung der möglichen Contributory Hazards, was wiederum eine Reduktion der schwerwiegenden Unfälle, also der System Accidents, bewirkt.

Der Aufbau des Safety-Vorgehensmodells, wie er in dieser Arbeit beschrieben wurde, hat also hohes Potential zur Entwicklung eines Inherent Safe System Designs. Es sei an dieser Stelle jedoch bemerkt, dass das Safety-Vorgehensmodell kein simples „Rezept“ ist, das für alle Fälle immer gleich durchzuführen ist. Das Safety-Vorgehensmodell erfordert immer eine intelligente Anpassung an die jeweilige konkrete Projekt-Situation. Es gibt klare Richtlinien vor, die als Startpunkt zu sehen sind, von denen aber immer wieder Anpassungen notwendig sind.

Das Safety-Vorgehensmodell teilt den Lifecycle grob in den Problemraum, den Modellierungsraum und den Lösungsraum ein. Wie in dieser Arbeit ersichtlich ist, liegt der Schwerpunkt des Safety-Vorgehensmodells auf dem Problemraum, also der Vor-Projektphase und dem Modellierungsraum.

Der Lösungsraum, also der Teil des Lifecycles in dem das System tatsächlich realisiert wird, ist nicht mehr Teil dieser Arbeit. Er ist zwar aus Engineering Sicht vermutlich der „greifbarste“, hier sind aber die wesentlichen Entscheidungen im Vorfeld schon getroffen worden und wenn nicht, dann nur deshalb weil die vorangehenden Phasen vernachlässigt wurden. Somit wurde in dieser Arbeit ein starker Fokus auf den Problemraum gelegt – dies ist der Teil des Lifecycles, in dem das Projekt noch gar nicht beauftragt wurde. In sicherheitskritischen Systementwicklungen ist diese Phase, in die beispielsweise eine Feasibility Study, aber auch die Einschätzung der Sicherheitsanforderungsstufe und ein grobes Safety Konzept fallen, essentiell. Die Kosten und Ressourcen für ein Projekt hängen von diesen Faktoren direkt ab. Des Weiteren wird ein Fokus auf den Modellierungsraum gelegt, wo das System aus technischer Sicht entwickelt wird und alle relevanten Safety Analysen durchgeführt werden müssen. So soll ein Inherent Safe System Design entstehen, welches dann im Lösungsraum umgesetzt werden muss.

6.1 Ergebnisse und Diskussion

Das Beispiel des Side-Stands demonstriert die praktische Anwendung des Safety-Vorgehensmodells. Dabei wird ersichtlich, wie komplex die Entwicklung eines sicherheitskritischen Systems ist, obwohl das Beispiel auf den ersten Blick einfach erscheinen mag. Das Safety-Vorgehensmodell bringt die notwendige Systematik in die Systementwicklung und macht die verborgene Komplexität sichtbar. Die Vorgehensweise im Safety-Vorgehensmodell, wie sie in Kapitel 3 beschrieben ist, wirkt aufgrund seiner Darstellungsform sehr sequentiell. In Wahrheit handelt es sich zwar um eine sequentielle Darstellung (eine andere Darstellung würde die Verständlichkeit mindern), jedoch nicht notwendigerweise um eine sequentielle Vorgehensweise. Die Struktur des Safety-Vorgehensmodells ist dabei so aufgebaut, dass erforderliche Iterationszyklen immer möglich sind und der Ablauf dabei nicht verletzt wird. Mit den zyklisch durchgeführten Safety-Analysen und den damit verbundenen Derived Safety Requirements werden Iterationen im Modellierungsraum angeregt. Mit diesen Iterationszyklen wird das System Design immer wieder verbessert, bis hin zu einem Inherent Safe System Design.

Wie das Side-Stand Beispiel zeigt, kann mit einem Inherent Safe System Design die Eintrittswahrscheinlichkeit von System Accidents stark reduziert werden. Das Vorgehen war folgendes: Mit der Anwendung der traditionellen Analyse-Methoden, einer induktiven und einer deduktiven Methode, konnten mehrere Fehler-Ursachen und der Single Point of Failure identifiziert werden. Mit diesen Ergebnissen wurden in der Contributory Hazard Analysis gefährliche Fehler-Kombinationen analysiert und ein Contributory-Hazard (C-Hazard) identifiziert. Die Ursachen für diesen C-Hazard boten die Möglichkeit der Fault-Avoidance, die wirksamste Möglichkeit den System Accidents entgegenzuwirken. Da dies nur sehr eingeschränkt möglich ist wurde für die restlichen gefährlichen Fehler-Kombinationen die Failure-Prevention angewendet. Diese zeigt sehr wohl auch seine Wirkungen, es ist dabei jedoch darauf zu achten, dass bei den dafür erforderlichen Safety-Maßnahmen kein zusätzliches Fehler-Potential implementiert wird. Wie aus dem Side-Stand Beispiel auch eindeutig ersichtlich ist, kann die Wahrscheinlichkeit von System Accidents durch ein Inherent Safe System Design stark reduziert, jedoch nicht komplett ausgeschlossen werden.

Das Safety-Vorgehensmodell baut bei den Safety-Analysen auf drei generischen Konzepten, der Hazard Avoidance, der Hazard Prevention und der Hazard Control auf. Die These der Inherent System Safety will den Fokus auf das erste dieser drei Konzepte lenken, die Avoidance. Die Hazard Prevention wird eingesetzt, wenn die Hazard Avoidance nicht möglich ist und das Hazard-Controlling, wenn keine Hazard Prevention möglich ist.

Um potentielle „Flawed Requirements" frühzeitig erkennen und vermeiden zu können, reicht ein systematisches Vorgehen alleine nicht aus. Der richtige Methoden-Einsatz zum richtigen Zeitpunkt spielt hier eine wesentliche Rolle. Das heißt, es ist ein systematisch-methodisches Vorgehensmodell gefordert. Wie im Side-Stand Beispiel deutlich erkennbar ist, konnten bereits in der Vor-Projektphase die wesentlichen Hazards identifiziert werden. Das Shell-Modell lieferte dabei den zu analysierenden System-Kontext. Mit der Erweiterten FFA wurden die Top-Level Funktionen analysiert und mit der Exogenen Hazard Analysis die äußeren System-Einflüsse. Zwei Analyse-Methoden die effizient eingesetzt werden konnten und dabei gute Ergebnisse lieferten.

Da eine vollständige System Requirements Spezifikation eine wesentliche Voraussetzung für Entwicklung eines Inherent Safe System Designs ist, wurden im Modellierungsraum alle definierten System Requirements zugeordnet und danach systematisch analysiert. Mit der HAZOP wurden in der FHE die funktionalen System Requirements nochmals eingehend analysiert, bevor sie zur Entwicklung des System Designs freigegeben wurden. Im Zuge der Preliminary System Safety Evaluation wurde das System Design auf endogene Hazards analysiert. Dabei wurde eine induktive (FMEA) und eine deduktive (FTA) Analyse-Methode angewendet, um mögliche Fehlerursachen aus zwei verschiedenen Blickwinkeln, identifizieren zu können. Die Ergebnisse lieferten die Basis für Contributory Hazard Analysis, mit der die gefährlichen Fehlerkombinationen identifiziert wurden, welche ein hohes Potential für System Accidents darstellen. Mit der Fault-Avoidance und der Failure-Prevention konnte die Eintrittswahrscheinlichkeit von System Accidents stark reduziert werden.

Für die Realisierung eines „Inherent Safe Systems" sorgt der Projektplanungsprozess. Im Sinne des Inherent Safe System Approach beginnt die Planung bereits in der Vor-Projektphase, also im Problemraum. Hier wurde im Side-Stand Beispiel, mit der initialen Erstellung der Projekt-Prozesslandschaft der organisatorische Gesamtüberblick geschaffen. Dieser wurde während der Projektplanung verfeinert und war Basis für die Planungsstruktur. Die PPL ist eine wichtige Darstellungsform, mit der das Projekt in seiner Gesamtheit für alle Beteiligten begreifbar gemacht werden kann. Der Projektplanungsprozess im Modellierungsraum konkretisiert die erforderlichen Leistungen im Projekt. Für eine lückenlose Planung ist auch hier die vorgegebene Systematik der einzelnen Planungsschritte einzuhalten. Mit der Standardisierung der Abläufe können „Process Flaws" im Planungsprozess vermieden werden.

Ein nachvollziehbarer Sicherheitsnachweis (Safety Case), mit dem ein „Inherent Safe System" nachgewiesen werden soll, ist von einer definierten Ablaufstruktur abhängig. Das Safety-Vorgehensmodell stellt mit der Projekt-Prozesslandschaft (PPL) diese Struktur sicher. Mit den in der PPL definierten Arbeitsprodukten können auch die Evidenzen sichergestellt werden. Der geforderte System-Kontext wird mit dem Shell-Modell nachgewiesen.

Mit der Goal Structuring Notation (GSN) von Kelly [Kel_98] wurde ein Safety Case Pattern erarbeitet das an die konkrete Projektsituation angepasst werden kann. Um ein Safe System, im Sinne des „Inherent System Safety Approach“ nachweisen zu können, sind jedoch noch die Nachweise für die Hazard Identification und Risk-Mitigation sowie der Nachweis der Hazard-Traceability notwendig. Dafür wurden im Safety Case Pattern eigene Goals angelegt. Die Evidenzen für die Hazard Identification und die Risk-Mitigation liefern die Ergebnisse der Safety-Analysen. Die Evidenzen für die Hazard-Traceability wird mit der Hazard Traceability Matrix aus der PSSE nachgewiesen. Mit der Evaluierung der System Design Lösungen, im System Design Prozess, wird das Inherent Safe Design nachgewiesen.

Mit Hilfe dieser integrierten Sichtweise und des darauf basierenden, systematisch-methodischen Vorgehens, ist es nun möglich Systeme zu entwickeln, die in ihrer Gesamtheit schon wesentlich sicherer konzipiert sind. Dabei ist klar, dass auch ein „Inherent System Safety Approach“ das Restrisiko nicht komplett beseitigen kann. Außerdem können auch nur eine geringe Anzahl aller identifizierten Hazards, im Sinne der „Avoidance“ vermieden werden, womit auch die Hazard-Prevention und das Hazard Controlling weiterhin benötigt werden, um die größtmögliche Sicherheit des entwickelnden Systems zu gewährleisten.

Der „Inherent System Safety Approach“, welcher im Safety-Vorgehensmodell verankert ist, führt zu einem „Inherent System Safety Thinking”, also auch zu einer Erweiterung der Safety-Kultur. Entsprechend der erweiterten Fehlerkette, die ja bis zum Management Approach gedacht ist, kann so die Sicherheit schon von Anfang an ins System integriert werden.

6.2 Nutzen der Erkenntnisse aus der Dissertation

Schon heute, und bestimmt verstärkt in Zukunft, zeigen sich die negativen Auswirkungen schwerwiegender Unfälle auf die Wirtschaft. Somit steigt die wirtschaftliche Notwendigkeit für verbesserte Safety-Maßnahmen, um kostenintensive Auswirkungen, getrieben durch Sensibilisierung der Gesellschaft und Konsequenzen aus Vorfällen, Unfällen und juristischen Prozessen, entgegenzuwirken. Als stichwortartige Beispiele dienen hierzu die vermehrten Rückrufaktionen in der Automobil-Branche (Toyota, etc.), Unfallentschädigungen und Prozesskosten im Bereich der Flugindustrie (Malaysian Air, etc.) und im Bahnwesen (DB Eschede, Kaprun, etc.), sowie die daraus resultierenden Image- und Verkaufsverluste.

Der Paradigmenwechsel hin zum „Inherent System Safety Approach“ soll damit den Safety-Notwendigkeiten zukünftiger, noch komplexerer Systeme gerecht werden, und zwar über den aktuellen State-of-the-Art hinaus, welcher bereits heute Schwierigkeiten zeigt, mit der aktuellen Komplexität schritthalten zu können.

Der neue Ansatz dieser Dissertation, ein „Inherent System Safety Approach“ im technischen Umfeld zu leben, führt zwangsweise zu einer Konzentration auf die Konzept- und Entwicklungsphase von industriellen Entwicklungs-Projekten und will dort zu grundlegenden Verbesserungen bis hin zu einem Paradigmenwechsel führen.

Anhang A: Checklisten

Checkliste Einsatzbereich

- Rechtliche Anforderungen (Gesetze)
- Normen, Standards, State-of-the-Art
- Dauer des Einsatzes
- Region, Land, Ort
- Politische Lage, wirtschaftliche Lage…

Checkliste Systemumwelt

Folgende Liste wurde aus mehreren Quellen ([Def56_96] Part 2 Annex B, [Eric2_05], Appendix C)

- Umwelteinflüsse
    - Temperatur, Hitze, Kälte
    - Druck
    - Wind
    - Luftfeuchtigkeit
    - Wasser
    - Vibrationen
    - Elektromagnetische Einflüsse
    - Strahlung
    - Dämpfe, Gase
- Gefährliche Komponenten
    - flammbare Substanzen
    - Laser
    - Explosives
    - erstickende, giftige oder ätzende Substanzen
    - hohe Temperaturen oder tiefkalte Flüssigkeiten
    - gefährliche Konstruktionsmaterialien
    - Drucksysteme
    - elektrische Quellen
    - ionisierende und nicht ionisierende Strahlungsquellen

  - hydraulische Hebel oder rotierende Teile
  - andere Energiequellen
  - Abgase
  - passive Hindernisse, gefährliche Oberflächen
  - gefährliche Schneide- und Press-Werkzeuge
- Sicherheitskritische Schnittstellen zwischen den verschiedenen Elementen des Systems
  - Materialkompatibilität
  - elektromagnetische Interferenz und Kompatibilität
  - unachtsame Aktivierung
  - Feuer und Explosionsgefahr bzw. Ausbreitungsverhalten
  - Hard- und Software Steuerung
- Faktoren des Betriebsgebietes
  - Fall
  - Erschütterung und Vibration, auch seismisch
  - extreme Temperaturen, Druck klimatischer Gegebenheiten
  - Lärm
  - Einwirkung giftiger oder ätzende Substanzen
  - Feuer und Explosionen
  - Insekten- oder Nager-Schäden
  - Fremdkörper und Staub
  - elektrostatische Entladungen, inkludiert Blitze
  - elektromagnetische Interferenzen
  - ionisierende und nicht ionisierende Strahlung, inkludiert Laserstrahlen
  - Fehler in unterstützenden Systemen, hydraulische Systeme
- Abgase
  - Betrieb, Test, Wartung und Notfallprozeduren
  - Betrieb im Frieden, Praxis und Krieg
  - Human Factor considerations
  - Angemessenheit und Wirksamkeit von Instruktionen, Training und Probe
  - Gesundheitsgefahren bzw. Risiko
  - Benutzerfehler, inkludiert Aktivierungsausfälle
  - Effekte durch Faktoren wie Ausrüstungslayout, Ergonomie
  - mögliche Belastung durch giftige Materialien, Geräusch und Strahlung
  - Lebenserhaltungssystem
  - Crashsicherheit, Ausgang, Rettung und Überleben
  - Reparatur und Verwertung
- Bedrohung durch programmierbare elektronische Systeme
  - Viren
  - Sicherheitslücken (security breaches)

Anhang B: ISPro® Prozesse

Name:	**PjI: Projekt-Initialisierung**
Zweck:	Die Projekt-Initialisierung wird durchgeführt, um den ProjektAufwand erheben zu können. Damit kann der Business Case beziehungsweise die Angebotserstellung konkretisiert werden.
Auslöser:	Auftrag zur Erstellung des Makro-Konzepts
Inputs:	Identifizierte Artefakte aus Vorgänger-Projekten, vorhandene Anforderungen des Kunden bzw. der Stakeholder
Aktivitäten:	• **PjI_BA1: Scope of Work (SoW) definieren** (Projekt-Umfang) 1. Projektbeschreibung erstellen 2. Projekt-Zweck definieren 3. Operational Requirements spezifizieren 4. Definierte(s) Lieferobjekte definieren 5. Projekt-Ziele definieren 6. Projekt-Nicht-Ziele definieren 7. Hauptaufgaben im Projekt festlegen • **PjI_BA2**: **Projekt-Kontext-Analyse (PKA) durchführen** 1. Projekt-Inputs identifizieren 2. Nach-Projektphase analysieren 3. Andere Projekte im Unternehmen analysieren 4. Safety-Management-System und Safety-Kultur darstellen *[ISO 26262-2: Kapitel 5.4.2]* 5. Qualitätsmanagement-System darstellen *[ISO 26262-2: Kapitel 5.4.1]* 6. Identifikation der wichtigsten Stakeholder 7. Zuständige Behörden, Regulatoren und Zertifizierungs-einrichtungen identifizieren; *[IEC 61508-1, Kapitel 7.2.2.4]* ▪ behördliche Rahmenbedingungen ▪ geforderte Normen und Standards identifizieren

	• **PjI_BA3: Erstansatz der Projekt-Prozesslandschaft (PPL) erstellen** (Projektstrategie festlegen *[MIL-STD 882E, Task 101], [ISO 15504-5: PRO.1.BP2], [ISO 26262-2, Clause 5.4.5][79])* 1. Prozesse von den Hauptaufgaben des SoW ableiten und in einem V-Modell darstellen 2. Engineering Lifecycle vom V-Modell ableiten 3. System Safety Lifecycle darstellen und Safety-Prozesse den Engineering-Prozessen zuordnen 4. Support-Prozesse in die PPL einbinden 5. Projektmanagement-Lifecycle einbinden • **PjI_BA4: Konzeptionierung und PHI beauftragen/veranlassen** • **PjI_BA5: Organisatorisches Konzept erstellen** 1. Scope of Work (SoW) verfeinern ▪ Sicherheitsanforderungsstufe in das SoW aufnehmen ▪ Hauptaufgaben nacharbeiten 2. Projekt-Prozesslandschaft (PPL) verfeinern/nacharbeiten ▪ Stage Gates in die PPL aufnehmen ▪ Erforderliche Safety Audits und –Assessments in die PPL aufnehmen ▪ Erforderliche Q-Audits in die PPL aufnehmen ▪ CM-Audits und Baselines in die PPL aufnehmen 3. Machbarkeit überprüfen (technisch und organisatorisch) 4. Projekt-Risiko einschätzen 5. Leistungen abschätzen 6. Projekt-Kosten errechnen • **PjI_BA6: Makro-Konzept erstellen und freigeben**
Methoden / Techniken:	• Scope of Work • Projekt-Kontext Analyse • Projekt-Prozesslandschaft (PPL)
Output:	Scope of Work, Projekt-Kontext-Ergebnisse, PPL
Verantwortung:	Projektmanager
Resultat:	Freigegebenes Makro-Konzept

[79] Tailoring des Safety Lifecycles der ISO 26262 kann auf Organisationsebene durchgeführt werden.

Name:	**KON: Konzeptionierung**
Zweck:	Das technische Konzept wird erstellt, um darauf basierend mögliche Gefährdungen identifizieren zu können.
Auslöser:	Auftrag zur Konzeptionierung
Inputs:	Kunden-Anforderungen, System Informationen (wenn vorhanden), vorhandene ähnliche Grob-Konzepte
Aktivitäten:	• **KON_BA1: System-Umgebung und System-Umfeld identifizieren** 1. Systemgrenzen festlegen *[IEC 61508-1, Kapitel 7.3.2.1]* 2. Unmittelbares System-Umgebung identifizieren *[IEC 61508-1, Kapitel 7.2.2.1]* 3. Mittelbare System-Umgebung identifizieren 4. System-Einbettung identifizieren (mittelbares System-Umgebung) 5. System-Umwelten identifizieren (Umweltauflagen, Witterungsverhältnisse, …) 6. Einsatzbereich identifizieren (gesetzliche Rahmenbedingungen, geforderte Normen u. Standards, Einsatzgebiete, Lebensdauer, etc.) *[IEC 61508-1, Kapitel 7.2.2.4]* • **KON_BA2: Top-Level Requirements spezifizieren** *[IEC 61508-1, Kapitel 7.2.2.1]* 1. Top-Level Funktionen definieren 2. Funktionale Top-Level Requirements definieren 3. Nicht-funktionale Requirements definieren 4. Top-Level Interface Requirements definieren • **KON_BA3: System-Entwurf entwickeln** *[IEC 61508-1, Kapitel 7.2.2.1]* 1. System-Inhalte definieren (Sub-Systeme) 2. Interne Schnittstellen definieren 3. Funktionale Top-Level Requirements verifizieren • **KON_BA4: Technisches Konzept spezifizieren und freigeben**
Methoden Techniken:	• Shell-Modell • Requirements Engineering
Output:	Shell-Modell, Tabelle der System-Umgebung, Top-Level Requirements, System-Entwurf
Verantwortung:	Systems-Engineer
Resultat:	Technisches Konzept ist freigegeben

Name:	**PHI: Preliminary Hazard Identification (PHI)**
Zweck:	Die PHI wird durchgeführt, um die Top-Level Hazards zu identifizieren und die erste Einschätzung der Sicherheitsanforderungsstufe vorzunehmen.
Auslöser:	Auftrag zur Preliminary Hazard Identification
Inputs:	Technisches Konzept, PKA, Hazard- und Unfalls-Checklisten, Shell-Modell, Lessons Learned aus Vorgänger-Projekten
Aktivitäten:	• **PHI_BA1: Organisatorische Safety-Maßnahmen identifizieren** 1. Anzuwendende Safety-Normen verstehen *[PjI_BA2.7]* 2. Erste Kontake mit Regulatoren und/oder Zertifizierungsbehörden aufnehmen 3. Zertifizierungs-Prozess festlegen *[DO 178C, Kapitel 9.0]* 4. Zertifizierungsaufwände schätzen 5. Zertifizierung planen • **PHI_BA2: Top-Level Hazards identifizieren** *[IEC 61508-1, Kapitel 7.2.2.2, 7.2.2.3]* 1. Top-Level Funktionen auf mögliche Hazards untersuchen *2.* Unmittelbare System-Umgebung auf mögliche Hazards untersuchen *[IEC 61508-1, Kapitel 7.2.2.5]* 3. Mittelbare System-Umgebung auf mögliche Hazards untersuchen (Informationen über exogene Gefährdungen beschaffen (z.B. Einsatzdauer des Systems, Vibrationen, Einsatz-Intensität, Giftige Substanzen, Explosivität, etc.)) 4. Gefahrenquellen und gefährlichen Situationen identifizieren 5. Unfall-Auslösende Ereignisse betrachten *[IEC 61508-1, Kapitel 7.2.3.5]*, wie beispielsweise: - Prozedurale Fehler - Menschliches Versagen - Fehlermechanismen - … • **PHI_BA3: Hazard Avoidance durchführen** • **PHI_BA4: Safety-Risiko-Assessment durchführen** *[IEC 61508-1, Kapitel 7.4]* 1. Safety-Risiko-Bewertung durchführen 2. Sicherheitsanforderungsstufe bestimmen 3. Safety Goals definieren 4. Safety Requirements ableiten 5. Safety-Maßnahmen definieren • **PHI_BA5: Safety-Konzept erstellen**

	• **PHI_BA6: PHI-Report verfassen und kommunizieren**
Methoden / Techniken:	• Extended FFA (eFFA) • Hazard and Operability Analysis (HAZOP) • Extended Hazard Analysis (eHA)
Output:	Top-Level Hazards, Sicherheitsanforderungsstufe, Safety-Konzept, PHI-Report, erweiterte Checklisten
Verantwortung:	Safety Engineer
Resultat:	Safety-Konzept ist freigegeben

Name:	**PjP: Projektplanung**
Zweck:	Die Projektplanung wird durchgeführt, um mit dem Projektteam das gemeinsam abgestimmte Vorgehen zu dokumentieren und das Commitment des Projektteams, des Projektauftraggebers und aller relevanten Stakeholder einzuholen.
Auslöser:	Unterzeichneter Projektantrag
Inputs:	Projektantrag, Kundenanforderungen, PHB-Vorlage, Scope of Work, Projekt-Kontext-Ergebnisse, PPL
Aktivitäten:	• **PjP_BA1: Projektstart-Workshop durchführen** • **PjP_BA2: Projekt-Stakeholder identifizieren** • **PjP_BA3: Planung laufend aktualisieren** • **PjP_BA4: Safety Plan(nung) erstellen/integrieren** • **PjP_BA5: Pläne (Support-Prozesse) erstellen/integrieren** *[ISO 15504-5: PRO.1.BP7], [CMMI: SG1, SP1.4], [DO 178C, Ch. 4.2.d]* o Validations-Plan *[ISO 26262-2: Chapter 6.4.3.3] (SUPx_LC, VEF.x)* o Verifikations-Plan *[ISO 26262-2: Chapter 6.4.3.3]* o Quality-Assurance-Plan o Configuration-Management-Plan o Integrationsplan *[ISO 26262-2: Chapter 6.4.3.3]* o etc. • **PjP _BA6: Leistungsplanung durchführen** *[ISO 15504-5: PRO.1.BP5]* • **PjP _BA7: Leistungs-Abschätzungen durchführen** *[ISO 15504-5: PRO.1.BP4]* • **PjP _BA8: Zeitliche Planung und Kostenplanung durchführen** *[ISO 15504-5: PRO.1.BP8]* • **PjP _BA9: Projektorganisation festlegen** *[ISO 15504-5: PRO.1.BP6], [ISO 15504-5: PRO.1.BP9]* Definition der o notwendigen Skills definieren o Verantwortlichkeiten festlegen *[ISO 15504: GP 2.1.4],* *[ISO 26262-2: Chapter 6.4.3]* o Befugnisse festlegen *[ISO 15504-x: GP 2.1.4]* • **PjP_BA10: Projekt-Risikomanagement durchführen** • **PjP_BA11: Konfigurationsmanagement definieren und institutionalisieren** *[DO 178C, Ch.4.2.g]* Kriterien für Planungsänderungen definieren *[DO 178C, Ch. 4.2.e]* • **PjP_BA12: PHB erstellen und freigeben** *[ISO 15504-5: PRO.1.BP10], (Review durchführen [DO 178B, Ch.4.6])*

Methoden / Techniken:	• Scope of Work • Projekt-Kontext Analyse • Projekt-Prozesslandschaft (PPL) • Projekt-Strukturplan (PSP) • Zeitplanung • Projekt-Organisation • Kostenplanung • Risiko-Management
Output:	Projekthandbuch (PHB)
Verantwortung:	Projektmanager
Resultat:	Projekthandbuch (PHB) ist freigegeben

Name:	**RqE: Requirements Engineering**
Zweck:	Das Requirements Engineering wird durchgeführt, um die Bedürfnisse und –anforderungen der Stakeholder über den gesamten System-Lifecycle des zu entwickelnden Systems zu erfassen.
Auslöser:	Zeitliche Planung
Inputs:	Makro-Konzept, Template Requirements Specification
Aktivitäten:	• **RqE_BA1: Stakeholder-Anforderungen und –wünsche einholen** *[ISO 15504: ENG.1.BP1]*, [SP1.1] o Stakeholder Identifizieren [PjP_BA1] o Anforderungen und Wünsche einholen • **RqE_BA2: Stakeholder-Anforderungen und Top-Level Requirements analysieren und konsolidieren** o Safety-Anforderungen aufnehmen [PHI] o Stakeholder-Anforderungen verstehen *[ISO 15504: ENG.1.BP2]* o Stakeholder-Anforderungen konsolidieren und zuteilen *[ISO 15504: ENG.2.BP5]* ▪ Gesetzliche Auflagen ▪ Regulatoren und Zertifizierungsbehörden ▪ Geforderte Normen & Standards ▪ Rahmenbedingungen (Standardachitekturen, zu verwendende Technologien, …) • **RqE_BA3: System Requirements entwickeln/ableiten und beschreiben** *[ISO 15504: ENG.2.BP1]* • **RqE_BA4: System Requirements evaluieren** *[ISO 15504: ENG.2.BP4]* o System Requirements analysieren *[ISO 15504: ENG.2.BP3]* o Machbarkeit überprüfen und Lösungsansatz optimieren *[ISO 15504: ENG.2.BP2]* o System-Umfeld überprüfen o Rahmenbedingungen und Einschränkungen überprüfen o Umweltauflagen überprüfen o Gesetzliche Vorgaben überprüfen o Requirements auf Inkonsistenzen, Vollständigkeit, Konflikte, Uneindeutigkeit, Verifizierbarkeit überprüfen • **RqE_BA5: FHE durchführen** • **RqE_BA6: System Requirements freigeben und kommunizieren** *[ISO 15504: ENG.1.BP3], [ISO 15504: ENG.2.BP6]*
Methoden /	T.B.D. (abhängig von der Sicherheitsanforderungsstufe)

Techniken:	
Output:	Systems Requirements Spezifikation
Verantwortung:	Requirements Engineer
Resultat:	System Requirements Spezifikation ist freigegeben

Name:	**FHE: Functional Hazard Evaluation**
Zweck:	Die FHE wird durchgeführt, um mögliche Hazards auf Basis der definierten funktionalen System Requirements in deren System-Umgebung zu identifizieren.
Auslöser:	Projektplanung
Inputs:	System Requirements Spezifikation, Shell-Modell, Safety-Norm, Standards
Aktivitäten:	• **FHE_BA1: Safety-Planung durchführen** *[ISO26_11, Teil 4, Kapitel 5.1]* o Compliance Planung *[DO 178C, Kapitel 9.1]* ▪ Zertifizierungsschritte ▪ Meetings mit Zertifizierungsstelle ▪ Geforderte Nachweise (Evidenzen) über den gesamten Projekt-Lifecycle festlegen o Confirmation Measures *[ISO26_11, Teil 2, Kapitel 6.2]* ▪ Confirmation Reviews, ▪ Audits, ▪ Assessments • **FHE_BA2: Safety Case Struktur festlegen** • **FHE_BA3: Hazards identifizieren** [IEC08_10, Teil 1, Kap.7.4.1], [ISO26_11, Teil 3, Kapitel 7.4.2.2] o Funktionen *[MIL-STD 882E, Task 208]* *(functional failure: malfunction, loss of function, degraded function, function out of timing or sequence)* o System Umgebung und Einsatzbereich o Foreseeable Misuse analysieren *[IEC08_10, Teil 4, Kapitel 3.1.14]* • **FHE_BA4: Hazard Avoidance durchführen** • **FHE_BA5: Safety-Risiko-Assessment durchführen** • **FHE_BA6: FHE-Report verfassen und kommunizieren**
Methoden/ Techniken:	• eFFA • HAZOP
Output:	Analyse-Ergebnisse Derived Safety Requirements, erweiterte Hazard List, FHE-Report
Verantwortung:	Safety Engineer
Resultat:	FHE-Report ist versendet

Name:	**SyD: System Design**
Zweck:	Das System Design wird erstellt, um aus den System Requirements die benötigten Sub-Systeme und Schnittstellen zu definieren.
Auslöser:	Projektplanung
Inputs:	System Requirements Spezifikation, Technischer System-Entwurf, Shell-Modell, andere vorhandene Konzepte, FHE-Report

Aktivitäten:	• **SyD_BA1: System Design Architektur entwickeln**
	  ○ Sub-Systeme spezifizieren *[ISO 15504: ENG.3.BP1]*
	  ○ Externe Schnittstellen spezifizieren *[ISO 15504: ENG.3.BP3]*
	  ○ Interne Schnittstellen spezifizieren *[ISO 15504: ENG.3.BP3]*
	• **SyD_BA2: Alternative Lösungen evaluieren** *[ISO 15504: ENG.3.BP5]*
	• **SyD_BA3: System Requirements den Sub-Systemen zuteilen** *[IEC08_10, Teil 1, Kapitel 7.6], [ISO 15504: ENG.3.BP2]*
	• **SyD_BA4: System Design verifizieren** *[ISO 15504: ENG.3.BP4]*
	• **SyD_BA 5: PSSE durchführen**
	• **SyD_BA6: System Design freigeben und kommunizieren** *[ISO 15504: ENG.3.BP7]*

Methoden/ Techniken:	• Peer Review[80] • Walkthrough • Fagan Inspections
Output:	System Design
Verantwortung:	Systems Engineer
Resultat:	System Design ist freigegeben

[80] Peer Reviews (Partner Reviews) sind inhaltliche Überprüfungen von ausgewählten Arbeitsergebnissen durch gleichwertige Partner.

Name:	**PSSE: Preliminary System Safety Evaluation**
Zweck:	Die PSSE wird durchgeführt, um zu überprüfen, ob die Safety Anforderungen im Design umgesetzt sind und dadurch die geforderte Systemsicherheit erreicht wird. Weiters überprüft sie, ob das Design keine zusätzlichen Hazards beinhaltet.
Auslöser:	Projektplanung, Freigabe des System Design
Inputs:	System Requirements Spezifikation, System Design
Aktivitäten:	• **PSSE_BA1: System Design Lösung evaluieren** • **PSSE_BA2: Safety-Analysen durchführen** *[ISO26_11, Teil 4, Kapitel 7.4.3.1]* 1. Funktionales System-Konzept analysieren 2. Strukturales System-Konzept analysieren 3. Hierarchisches System-Konzept analysieren 4. Contributory Hazards identifizieren 5. Derived Safety Requirements definieren • **PSSE_BA3: Safety Integritäty Requirements spezifizieren** • **PSSE_BA4: System Design validieren** *[ISO26_11, Teil 4, Kapitel 9],* *[IEC08_10, Teil 1, Kapitel 7.14]* • **PSSE_BA5: Safety-Planung fertigstellen** o Synchroniseren mit PjP • **PSSE_BA6: Safety Case weiterführen** • **PSSE_BA7: PSSE-Report verfassen und kommunizieren**
Methoden/ Techniken:	• Failure Mode and Effect Analysis (FMEA)[81] • Fault Tree Analysis (FTA) • Contributory Hazard Analysis (CHA)
Output:	Analyse-Ergebnisse, PSSE-Report, Safety Case, Derived Safety Requirements,
Verantwortung:	Safety Engineer
Resultat:	PSSE-Report ist versendet

[81] Peer Reviews (Partner Reviews) sind inhaltliche Überprüfungen von ausgewählten Arbeitsergebnissen durch gleichwertige Partner.

Anhang C: Analyseergebnisse

Hazard-Traceability-Matrix des Side-Stands

Hazard	Safety Goal	Safety Requirement	Design-Lösung	Safety Requirement on Elelement
Ha_1: Unbeabsichtigtes Ausfahren des SS während der Fahrt bei hoher Geschwindigkeit und Linkskurve [TLHa1].	**SG_1:** SS darf während der Fahrt nicht ausfahren	**nFuSaR_1:** Das SSCS darf das Ausfahren des SS nur erlauben, wenn über das Bus-System [nFuSaR5] gemeldet wird: - MR steht und - GS in neutraler Stellung [TLSaR1]	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU interne Interfaces: - EG <--> CU	**Expansion Guardian** **EG_SaR_1.1** Der EG soll das Ausfahren des Side-Stands nur dann erlauben, wenn die Gangschaltung in neutraler Stellung ist und das Motorrad steht. **Control Unit** **CU_SaR_1.1** Die CU soll das Ausfahren des Side-Stands nur dann *bewirken*, wenn das Motorrad steht und die Gangschaltung in neutraler Stellung.
Ha_2: Fahrer fährt mit nicht vollständig eingefahrenem SS los, beschleunigt und fährt eine Linkskurve [TLHa2]	**SG_2:** Mit nicht vollständig eingefahrenem SS muss das Losfahren blockiert sein.	**nFuSaR_2:** Das SSCS muss die Endposition, des Side-Stands vor und nach dem Ein- bzw. Ausfahren detektieren und bei nicht eindeutiger Positions-Detektion eine Warnung über das Bus-System [nFuSaR5] an den Fahrer	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - I/O-Treiber - CAN-Bus Treiber - EG - CU	**Control Unit** **CU_SaR2.1** Die CU muss die Endposition, des Side-Stands nach jedem Einschalten der Zündung und vor und nach jedem Ein- bzw. Ausfahren

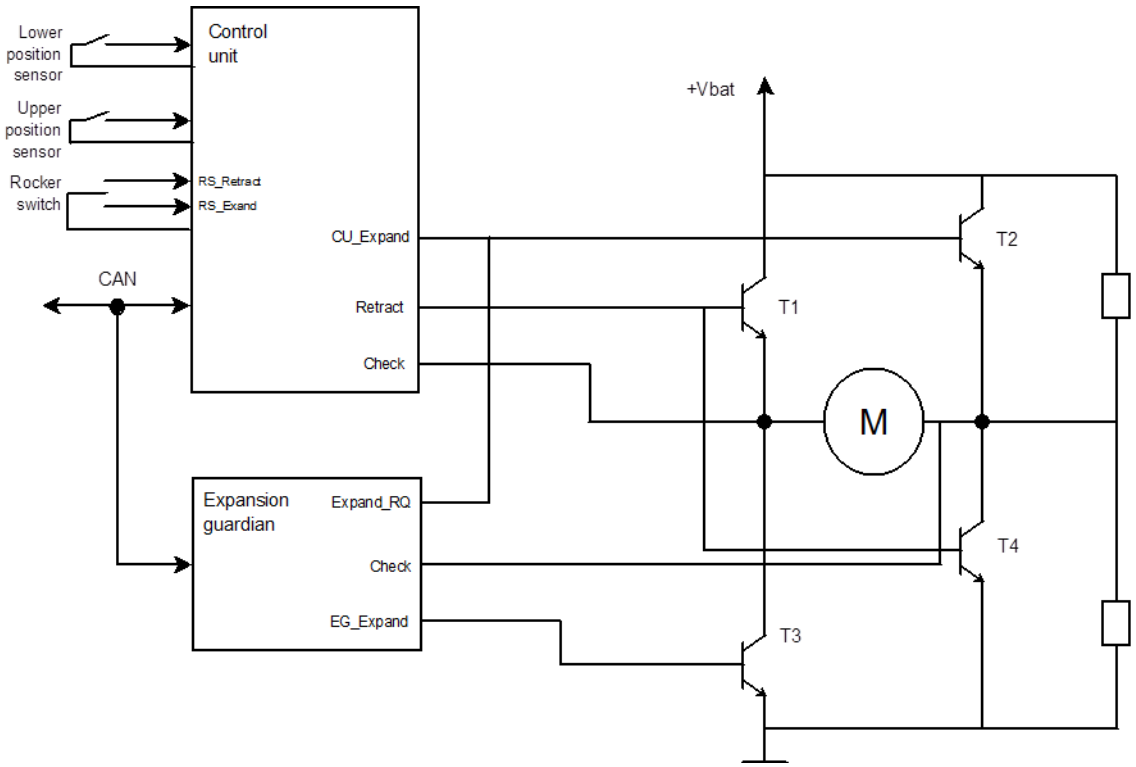
		melden. Damit wird der Fahrer aufgefordert den Kipp-Schalter nochmals in Richtung „Einfahren“ zu drücken, um den SS neu zu positionieren. Ist das nicht möglich, so muss das losfahren verhindert werden. [TLSaR2]	**interne Interfaces:** - EG <--> CU	detektieren und bei nicht eindeutiger Positions-Detektion eine Warnung via CAN-Bus an den Fahrer melden und das Losfahren verhindern.
Ha_3: Unbeabsichtigtes Einfahren des SS in der Parkposition [TLHa3].	**SG_3:** In der Parkposition muss das Einfahren des SS blockiert sein.	**nFuSaR_3:** Das SSCS muss das Einfahren verhindern, wenn über das Bus-System [nFuSaR5] die Meldung „Zündung EIN“ gemeldet wird [TLSaR3].	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**<u>Expansion Guardian</u>** **EG_SaR3.1** Der EG muss das Einfahren verhindern, wenn die Zündung des Motorrads ausgeschaltet ist. **<u>Control Unit</u>** **CU_SaR3.1** Die CU darf das Einfahren nur bewirken, wenn die Zündung des Motorrads eingeschaltet ist.
Ha_4: Nicht vollständig ausgefahrener SS bringt das Motorrad beim Abstellen ins Kippen [TLHa4]	**SG_4:** Der SS muss nach dem Ein- bzw. Ausfahren immer in einer definierten End-Position stehen.	**nFuSaR_4:** wie nFuSaR_2	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - I/O-Treiber - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**<u>Control Unit</u>** **CU_SaR4.1** wie CU_SaR2.1
Ha_5: Durch Spannungs-schwankungen und Störspitzen wird der Status des SSCS gestört, dadurch	**SG_5:** Das SSCS muss vor Spannungsschwankungen und Sörungen geschützt werden	**nFuSaR_5:** Das SSCS muss die Versorgungsspannung laufend überwachen und bei einem Abfall der Versorgungsspannung bzw. bei Spannungsschwankungen	**HW-Interface:** - Vcc-Stecker **HW-Sub-Systeme:** - Spannungs-überwachungs-	**<u>Control Unit</u>** **CU_SaR5.1** Die CU muss die Versorgungsspannung laufend überwachen und bei einem Abfall der

nehmen das I/O einen undefinierten Zustand ein. [TLHa4], [exHA#12 und exHA#13]		den Sicheren Zustand einnehmen und über das Bus-System [nFuSaR5] eine Warnung an den Fahrer melden.	baustein **interne Interfaces:** Vcc-IC --> CU	Versorgungsspannung bzw. bei Spannungsschwankungen den Sicheren Zustand einnehmen und eine Warnung via CAN-Bus an den Fahrer melden.
Ha_6: SS fährt zu wenig weit aus (#5) [exHA#9]	**SG_6:** SS muss innerhalb < 5 Sekunden ein bzw. ausgefahren sein und muss Position halten	**nFuSaR_6:** Wenn beim Ein- bzw. Ausfahren die Endposition innerhalb < 5 Sekunden nicht erreicht wird, muss das SSCS den SS in die ursprüngliche Position bringen und über das Bus-System [nFuSaR5] ein Warnsignal an den Fahrer melden.	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**Control Unit CU_SaR6.1** Wenn beim Ausfahren die Endposition innerhalb < 5 Sekunden nicht erreicht wird, muss die CU den SS in die ursprüngliche Position bringen und eine Warnung via CAN-Bus an den Fahrer melden und das Losfahre blockieren
Ha_7: SS fährt zu wenig weit ein (#14), die Fahrt wird mit ausgefahrenem SS begonnen (#34)	**SG 2:** Mit nicht vollständig eingefahrenem SS muss das Losfahren blockiert sein.	**nFuSaR_7:** wie nFuSaR_6	**HW-Interface:** CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**Control Unit CU_SaR7.1** Wenn beim Einfahren die Endposition innerhalb < 5 Sekunden nicht erreicht wird, muss die CU den SS in die ursprüngliche Position bringen und eine Warnung via CAN-Bus an den Fahrer melden.
Ha_8: SS bleibt beim Ein-/Ausfahren stecken (#30) [exHA #13 und #14]	**SG 2:** Mit nicht vollständig eingefahrenem SS muss das Losfahren blockiert sein.	**nFuSaR_8:** wie nFuSaR_6	**HW-Interface:** - CAN-Bus Interface **Protokoll:** CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG	**Control Unit CU_SaR8.1** wie CU_SaR6.1

			- CU **interne Interfaces:** - EG <--> CU	
Ha_9: SS fährt ein und aus (#4)	**SG_6:** SS muss innerhalb < 5 Sekunden ein bzw. ausgefahren sein und muss Position halten	**nFuSaR_9:** wie nFuSaR_6	**HW-Interface:** - CAN-Bus Interface **Protokoll:** - CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**Control Unit** **CU_SaR8.1** wie CU_SaR6.1
Ha_10: SS wird während der Fahrt beabsichtigt ausgefahren (#35)	**SG 1:** SS darf während der Fahrt nicht ausfahren	**nFuSaR_10:** wie nFuSaR_1	**HW-Interface:** - CAN-Bus Interface **Protokoll:** - CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	Wird durch EG_SaR_1.1 und CU_SaR_1.1 abgedeckt
		eIntRq_1: Das SSCS muss eine Bus-Schnittstelle (Bus-System) zur Verfügung stellen, über das der aktuelle Status des Fahrzeug abgerufen werden kann und über das der Side-Stand Status an das Fahrzeug kommuniziert werden kann. Geforderter Informationsaustauch: - Status des Motorrads • Motorrad steht bzw. fährt • Zündung ist eingeschaltet bzw. ausgeschaltet • Gangschaltung ist in neutraler Stellung bzw.	**HW-Interface:** - CAN-Bus Interface **Protokoll:** - CAN-Bus Protokoll **HW-Sub-Systeme:** - CAN-Bus Treiber - EG - CU **interne Interfaces:** - EG <--> CU	**Control Unit** **Cu_eIntRq_1.1** Die CU muss eine Schnittstelle zum CAN-Bus Treiber zur Verfügung stellen, über das der aktuelle Status des Fahrzeugs abgerufen werden kann und über das der Side-Stand Status an das Fahrzeug kommuniziert werden kann. **Expansion Guardian** **EG_eIntRq_1.2** Der EG muss eine Schnittstelle zum

		nicht in neutraler Stellung - Warnung an den Fahrer signalisieren - Fahren blockieren		CAN-Bus Treiber zur Verfügung stellen, über das der aktuelle Status des MR abgerufen werden kann und über das der SS-Status an das MR kommuniziert werden kann

Anhang D: SSCS HW-Lösungsvorschlag für HWReq_2



Literaturverzeichnis

[AvLa_04] Avizienis A., Laprie J.C.: Basic Concepts and Taxonomy of Dependable and Secure Computing, EEE Transactions on Dependable and secure Computing, VOL. 1, January-March 2004

[Bal_09] Balzert H.: Lehrbuch der Software-Technik, 3. Auflage, Spektrum-Verlag, 2009

[BiMu_04] Biethahn, J., Mucksch, H., Ruf W.: Ganzheitliches Informationsmanagement: Grundlagen, München, Wien, Oldenbourg-Verlag, 2004

[Bir_13] Birch J., Rivett R., Habli I., Bradshaw B., Botham J., Higham D., Jesty P., Monkhouse H., Palin R.: Safety Cases and their role in ISO 26262 Functional Safety Asssessment, Safecomp, 2013

[Bla_08] Blanchard B. S.: System Engineering Management, 4. Auflage, John Wiley & Sons Verlag, USA, 2011

[Boe_79] Boehm B.: Guidelines for verifying and validating software requirements and design Specifications, EURO IFIP, North Holland, 1979

[Bra_05] Braband J.: Risikoanalysen in der Eisenbahn-Automatisierung, 1. Auflage, Eurailpress, 2005

[BRB_93] British Railways Board: Safety of People Working on Traction and Rolling Stock, Group Standard, GM/TT0040, Issue 2, Revision A, October 1993

[CEN26_99] CENELEC EN 50126: Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS). Comité Européen de Normalisation Electrotechnique, 1999.

[CEN29_03] CENELEC EN 50129: Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling. Comité Européen de Normalisation Electrotechnique, 2003.

[Coop_02] Cooper G., Edgett S.J., Kleinschmidt: Optimizing the Stage-Gate® Process: What Best Practice Companies are Doing, Stage-Gate International and Product Development Institute Inc., Published by Research Technology Management (Indurstrial Research Institute, Inc.) Volume 45, Number 5, 2002

[Def56_07] Ministry of Defence Standard 00-56: Safety Management Requirements for Defense Systems, Issue 4, 2007

[Def56_96] Ministry of Defence Standard 00-56: Safety Management Requirements for Defense Systems, Issue 2, 1996

[Def58_00] Ministry of Defence Standard 00-58: HAZOP Studies on Containing Programmable Electronics, Part 1 and 2, Issue 2 Publication, Date 19 May 2000

[DIN25_81] DIN 25424: Fehlerbaumanalyse, Teil 1 und Teil 2, Beuth-Verlag, 1981

[DO_11] RTCA/DO-178C: Software Considerations in Airborne Systems and Equipment Certification, RTCA, Inc. 1150 18th Street, , NW, Suite 910, Washington, DC20036-3816 U.S.A, December 13, 2011

[DO2_00] RTCA/DO-254: Design Assurance Guidance for Airborne Electronic Hardware, RTCA, Inc. 1140 Connecticut Avenue, NW, Suite 1020 Washington, DC20036-4001 USA, April 19, 2000

[Drö_00] Dröschel W.: Das V-Modell 97: Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz, Oldenbourg-Verlag, 2000

[DS0_07] Ministry of Defence Standard 00-56: Safety Management Requirements for Defence Systems, Issue 4, Publication Date 01 June 2007

[ED_153] EUROCAE. Guidelines for ANS Software Safety Assurance. ED-153. August 2009.

[EGAS_13] Standardisiertes E-GAS-Überwachungskonzept für Benzin und Diesel Motorsteuerungen, Arbeitskreis EGAS, Version 5.5, 05.07.2013

[EGRW_08] Eisenberg Cl., Gildeggen R., Reuter A., Willburger A.: Produkthaftung: Kompaktwissen für Betriebswierte, Ingenieure und Juristen, Oldenbourg-Verlag, 2008

[Ehr_03] Ehret J.: Validation of Safety-Critical Distributed Real-Time Systems, Dissertation an der Fkultät für Elektrotechnik und Informatik, Technische Universität München, 2003

[ER_85] Europäische Richtlinie 85/374/EWG, Richtlinie des Rates zur Angleichung der Rechts- und Verwaltungsvorschriften der Mitgliedsstaaten über die Haftung für fehlerhafte Produkte, vom 25. Juli 1985

[Eric1_11] Ericson C.A.: Concise Encyclopedia of System Safety, 1. Auflage, John Wiley & Sons Verlag, USA, 2011

[Eric2_05] Ericson C.A.: Hazard Analysis Techniques for System Safety, John Wiley & Sons Verlag, USA, 2005

[FAA3_00] FAA System Safety Handbook, Federal Aviation Authority, Chapter 3, Principles of System Safety, December 30, 2000

[FAA7_00] FAA System Safety Handbook, Federal Aviation Authority, Chapter 7, Integrated System Hazard Analysis, December 30, 2000

[Fi_06] Fischer H.: Ein systemorientierter Ansatz zur Modularisierung von Planspielen mit dem Ziel der Komplexitätssteuerung und Integration in Standardsoftware, Dissertation, Universität Göttingen, 2006

[FTA_81] Office of Nuclear Regulatory Research: Fault Tree Handbook, NUREG-0492, Date Published: January 1981

[GMSH_12] Sundaram P., Hartfelder D.: Rigor in Automotive Safety Critical System Development, CTI-Conference ISO 26262 Conference, Detroit, June 2012

[GPM_05] Schelle H., Ottmann R., Pfeiffer A.: ProjektManager: GPM Deutsche Gesellschaft für Projektmanagement e.V., 2. Auflage, 2005

[GrBe_09] Grechenig Th., Bernhart M., Breiteneder R., Kappel K (Hrg).: Softwaretechnik. Mit Fallbeispielen aus realen Entwicklungsprojekten, Pearson Studium, München, 2009

[GSN_11] GSN Community Standard Version 1, Origin Consulting, York, November 2011

[Ham_12] Brahim Hamid, Jacob Geisel, Adel Ziani, David Gonzalez. Safety Lifecycle Development Process Modeling for Embedded Systems – Example of Railway Domain. SERENE 2012. LNCS 7527, pp. 63-75. Springer, Berlin, Heidelberg, 2012.

[HDHM_06] Hörmann, K., Dittmann L., Hindel B., Müller M.: SPICE in der Praxis, 1. Auflage, dpunkt.verlag, 2006

[HeBu_96] Heinrich L.J., Burgholzer, P.: Der Prozeß der Systemplanung, der Vorstudie und der Feinstudie, München, 1996.

[Höh_09] Höhn H., Sechser B., Dussa-Zieger K., Messnarz R., Hindel B.: Software Engineering nach Automotive SPICE, 1. Auflage, dpunkt.verlag, 2009

[HöHö_08] Höhn R., Höppner St.: Das V-Modell XT, Springer-Verlag, 2008

[Hol_07] Holzmann G. J.: Conquering Complexity, NASA/JPL for Reliable Software, 2007

[HTS_09] Tschürtz, H.: Konzept eines Integrativen Safety Prozesses. Master Thesis, Department für Wissens- und Kommunikationsmanagement. Donau-Universität Krems, Krems, 2009

[IAEA_99] Verification and Validation of Software Related to Nuclear Power Plant Instrumentation and Control, Technical Report Series Nr. 384, International Atomic Energy Agency, Vienna, 1999

[IE610_90] IEEE Std. 610.12-1990: IEEE Standards Glossary of Software Engineering Terminology, Institute of Electrical and Electronics Engineers 345 East 47th Street, NY, USA, September 28, 1990

[IEC08_10] IEC 61508, Functional Safety of electric/electronic/programmable electronic safety-related systems, Part 1-7 International Standard, IEC International Electronic Commission,Geneva, Switzerland, 2010

[IEC12_06] IEC 60812. Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA). International Standard, IEC International Electronic Commission, Switzerland, 2006

[INC_04] INCOSE (International Council on Systems Engineering): Systems Engineering Handbook, INCOSE-TP-2003-016-02, Version 2a, 1 June 2004

[IPMA_06] ICB – IPMA-Kompetenzrichtlinie, International Project Management Association, P.O. Box 1167, NL-3860 BD Nijkerk Netherlands,Version 3.0, Juni 2006

[ISO07_07] ISO/IEC FDIS 12207: Systems and Software Engineering – Software life cycle processes, International Standard, ISO International Organisation for Standardisation, Geneva, Switzerland, Second edition, 2007

[ISO09_08] EN/ISO 9001, Qualitätsmanagementsysteme – Anforderungen, ISO International Organisation for Standardisation, Geneva, Switzerland, 2008

[ISO10_03] ISO 10006 : Quality management systems – Guideline for quality management in projects, International Standard, ISO International Organisation for Standardisation, Geneva, Switzerland, 2003

[ISO12_11] ISO 12100: Sicherheit von Maschinen - Allgemeine Gestaltungssätze - Risikobeurteilung und Risikominderung, Ausgabe: 2011-03-15

[ISO15_06] International Standard ISO/IEC 15504: SPICE, Information technology – Process Assessment – Part 5: An exemplar Process Assessment Modell, 2006, Reference number ISO/IEC 15504-5:2006

[ISO15_11] International Standard ISO/IEC FDIS 15504-5: Information technology – Process assessment- Part 5: An exemplar software life cycle process assessment model, 2011, Reference number ISO/IEC FDIS 15504-5:2011

[ISO26_11] ISO 26262: Road vehicles – Functional safety, International Standard, ISO International Organisation for Standardisation, Geneva, Switzerland, 2011

[ISO88_08] International Standard ISO/IEC 15288: (International Standard – System and Software engineering – Systems and software life cycle proceses, 2008, Reference number ISO/IEC 15288:2008€, IEEE Std 15288-2008

[Jen_09] Jenny B.: Projektmanagement - Das Wissen für den Profi, vdf Hochschulverlag an der ETH Zürich, 2009

[Kam11] Roland Kammerer. Linux in Safety-Critical Applications. Open Source Automation Development Lab (OSADL). 25. Februar 2011

[Kel_07] Kelly T.: Safety Case Composition Using Contracts – Refinements based on Feedback from an Industrial Case Study, 2007

[Kel_98] Kelly T.: Arguing Safety – A Systematic Approach to Safety Case Management, Dphil Thesis YCST99-05, Department of Computer Science, University of York, UK, 1998.

[KeWe_04] Kelly T., Weaver R.: The Goal Structuring Notation – A Safety Argument Notation. Conference on Dependable Systems and Networks (DSN), Florence, Italy, 2004.

[KlAm_10] Kletz T., Amyotte P.: Process Plants: A Handbook for Inherently Safer Design, Second Edition, CRC Press, 2010

[Kle_01] Kletz T.: Learning from Accidents, Third Edition, Gulf Professional Publishing, 2001

[LA_90] Lee P.A., Anderson T.: Fault Tolerance – Principles and Practice, Volume 3 of Dependable Computing and Fault-Tolerant Systems, Second-Edition, Springer-Verlag, 1990

[Leff_03] Leffingwell D., Widrig D.: Managing Software Requirements: A Use Case Approach. Addison-Wesley Professional. 2003.

[Lev_03] Levenson N.G., Daouk M., Dulac N., Marais K.: A System Theoretic Approach for Safety Engineering, Massachusetts Institute of Technology, October 30, 2003

[Lev_11] Leveson N.G.: Engineering a Safer World: System Thinking Applied to Safety, Massachusetts Institute of Technology, Engineerin Sys-tems, 2011

[Lev_95] Leveson, N.G., Safetware: System Safety and Computers, Reading, MA: Addison-Wesley, 1995.

[LiRo_05] Liggesmeyer P., Rombach D.: Software Engineering eingebetteter Systeme, Elsivier, Spektrum Akademischer Verlag, 2005

[LPP_10] Löw P., Pabst R., Petry E.: Funktionale Sicherheit in der Praxis: Anwendungen von DIN EN 61508 und ISO/DIS 26262 bei der Entwicklung von Serienprodukten, dpunkt.verlag, 1. Auflage, 2010

[Lutz_01] Lutz R.: Analyzing Software Requirements Errors in Safety-Critical Embedded Systems, 2001

[Mas_07] Pfeiffer T., Schmitt R. (Hrsg.): Masing, Handbuch Qualitätsmanagement, Hanser-Verlag, 5. Auflage, 2007

[McD_02] John McDermid. Software Hazard and Safety Analysis. University of York, 2002.

[Men_10] Catherine Menon, Richard Hawkins, John McDermid and Tim Kelly. An Overview of the SoBP for Software in the Context of DS 00-56 Issue 4. Safety Critical Systems Symposium 2010. Springer Verlag. 2010.

[MIL_05] MIL-STD 882E: Military Standard, Department of Defense, Standard Practice for System Safety, 2005

[NaKu_91] Nakajo T., Kume H.: A Case History Analysis of Software Error Cause-Effect Rela-tionship, IEEE Trans Software Eng 17, 8. August 1991, pp. 830-838.

[PaBe_97] Pahl, G., Beitz, W.: Konstruktionslehre: Methoden und Anwendung, 4. Auflage, Springer-Verlag, 1997

[Par_10] Partsch H.: Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme, 2. Auflage, Springer-Verlag, 2010

[Part_07] Parthasarathy M.A.: Practical Software Estimation: Functional Point Methods for Insourced and Outsourced Projects, Addison-Wesley Verlag, 2007

[Pat_04] Patzak G., Rattay G.: Projektmanagement, Leitfaden zum Managen von Projekten, Projektportfolios und projektorientierten Unternehmen, Linde-Verlag, 4. Auflage, 2004.

[Pie02] Pierce R.: Preliminary assessment of Linux for safety related systems. Research Report 011. Prepared by CSE International Limited for the Health and Safety Executive 2002

[PMB_13] Project Management Institute's (PMI): A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Project Management Institute, 14 Campus Boulvard, Newton Square, Pennsylvania 19073-3299, USA, Fifth Edition, 2013

[Red_99] Redmill F., Chudleigh M., Catmur J.: System Safety: HAZOP and Software HAZOP, John Wiley & Sons Verlag, 1999

[Rei_12] Reif K.: Automobilelektronik: Eine Einführung für Ingenieure, Vieweg+Teubner Verlag, 4. Auflage, 2012

[RoMo_90] Roland H., Moriarty B.: System Safety Engineering and Management, Wiley-Interscience Publication, Second Edition, 1990

[Rop_09] Ropohl G.: Allgemeine Technologie, Eine Systemtheorie der Technik, Universitätsverlag Karlsruhe, 3. Auflage, 2009

[Ros_14] Ross H.L.: Funktionale Sicherheit im Automobil: ISO 26262, Systemsengineering auf Basis eines Sicherheitslebenszyklus und bewährten Managementsystemen, Hanser Verlag München, 2014

[Sch_98] Schildt G., Kastner W.: Prozessautomatisierung, Springer-Verlag, 1998

[ScZu_13] Schäuffele J., Zurawka Th.: Automotive Software Engineering, Springer-Verlag, 5. Auflage, 2013

[Smi_00] David J. Smith: Reliability, Maintainability and Risk, 5th edition, Butterworth Heinemann, 2000

[SmSi_04] Smith D., Simpson K.: Functional Safety, A Straightforward Guide to applying IEC 61508 and Related Standards, Elsevier-Verlag, Second Edition, 2004

[Som_95] Sommerville I,: Software Engineering. 5th edition. Addison-Wesley. 1995.

[Spr_12] John Spriggs: GSN – The Goal Structuring Notation: A Structured Approach to Presenting Arguments. Springer, 2012.

[Stä_10] Ständer T.: Eine modellbasierte Methode zur Objektivierung der Risikoanalyse nach ISO 26262, Dissertation, Universität Carolo-Wilhelmina zu Braunschweig, 2010

[Sto_96] Storey N.: Safety-Critical Computer Systems, Addison Wesley Longman, 1996

[VDA_04] Verband der deutschen Automobilindustrie VDA 6: Qualitätsmanagement in der Automobilindustrie, Teil 2, 2004

[VDA_12] Verband der deutschen Automobilindustrie VDA 4: Produkt- und Prozess-FMEA, 2012

[WaRe_11] Wallentowitz H., Reif K.:Handbuch Kraftfahrzeugelektronik: Grundlagen- Komponenten – Systeme – Anwendungen, Vieweg+Teubner Verlag, 2. Auflage, 2011

[Wer_12] Martin Werdich. FMEA – Einführung und Moderation. Springer Verlag, 2012.

[Yoo_08] Yae Han Yoon, Jae-Chon Lee, Tae-Hyun Kim, Seon-H Hong. An Integrated Process Model for the Systems Development Requiring Simultaneous Consideration of the SE Process and Safety Requirements. IEEE SysCon, Montreal, Canada, 2008.

Internetreferenzen

[AuS_13] AUTOSAR – A worldwide standard, Current developments, roll-out and outlook. http://www.autosar.org/download/papersandpresentations/AUTOSAR_a%20worldwide%20standard,%20current%20developments,%20roll-out%20and%20outlook.pdf (abgerufen am 01.08.2013)

[COIN_14] COIN-Programmlinie Aufbau: Aufbau eines Safety Kompetenzzentrums „Safety Competence Center Vienna“, Österreichische Forschungsförderungsgesellschaft (FFG), 2009-2014 https://www.ffg.at/content/coin-programmlinie-aufbau-2-ausschreibung (abgerufen am 17.09.2015)

[GCh_11] Beijing: Schwerer Unfall auf Rolltreppe kostet Menschenleben, 05.07.2011. http://german.china.org.cn/china/2011-07/05/content_22924322.htm (abgerufen am 19.09.2014)

[CMMI_11] CMMI for Development, Version CMMI-Dev. 1.3, CMU/SEI-2006-TR-008, November 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9661 (abgerufen am 15.06.2014)

[IEE_09] This Car Runs on Code, Posted 1 Feb 2009. http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code (abgerufen am 05.05.2013)

[Kap_00] Die Seilbahnkatastrophe von Kaprun, 11.11.2000. http://www.fireworld.at/cms/review.php?id=3 (abgerufen am 19.09.2014)

[MISRA_04] MISRA (Motor Industry Software Reliability Association): Guidelines for the Use of the C Language in Critical Systems, MISRA-C:2004, MIRA Ltd, October 2004 http://www.misra.org.uk (abgerufen am 01.06.2014)

[NHTSA_10] National Safety Administration: Department of Transportation Adresses Toyota Safety Issues, DOT 22-10, February 4, 2010. http://www.nhtsa.gov/PR/DOT-22-10 (abgerufen am 02.01.2014)

[Ori_11] Origin Consulting: GSN Community Standard Version 1, November 2011. http://www.goalstructuringnotation.info/ (abgerufen am 19.08.2013)

[PiMa_77] Pinto Madnes: Hundreds of people would needlessly burn to death http://www.motherjones.com/politics/1977/09/pinto-madness (abgerufen am 01.06.2014)

[Sto_09] Blockierendes Gaspedal: Toyota plant Rückruf von 3,8 Millionen Autos, 30.09.2012. http://www.spiegel.de/auto/aktuell/blockiertes-gaspedal-toyota-plant-rueckrufvon-3-8-

millionen-autos-a-652205.html
(abgerufen am 18.09.2014)

[Sha_09] Handy-Explosion tötet 20-Jährigen, 05.02.2009.
http://www.spiegel.de/netzwelt/mobil/akku-problem-handy-explosion-toetet-20-jaehrigen-a-605676.html
(abgerufen am 18.09.2014)

Glossar

Begriff	Beschreibung
Accident	(dt. Unfall) Ein Accident, im MIL-STD 882E [MIL_05] auch als Mishap bezeichnet, ist ein nicht geplantes Ereignis (Event) oder eine Serie von Events, bei dem der Umwelt, den Menschen oder dem System selbst Schaden zugefügt werden kann.
Analyse	Eine Analyse ist in ihrem Wesen die Informationsgewinnung durch das Zerlegen und Aufgliedern eines Betrachtungsgegenstandes in seine Komponenten sowie die Untersuchung der Eigenschaften und der Zusammenhänge der einzelnen Elemente [PaBe_97].
Arbeitsprodukt	Unter einem Arbeitsprodukt (work product) werden alle Artefakte verstanden die im Projektverlauf entstehen. Das sind beispielsweise Zwischenprodukte des zu entwickelnden Systems oder deren Sub-Systeme (Hardware-Module, Software-Code Teile, etc.) sowie Pläne, Spezifikationen, Dokumente, Zeichnungen, Protokolle und Berichte.
Automotive Safety Integrity Level (ASIL)	ASIL bezeichnet eines von insgesamt vier Levels zur Vermeidung von nicht tolerierbaren Restrisiken. Der ASIL verweist auf Sicherheitsmaßnahmen und notwendige Anforderungen aus der ISO 26262 für das Fahrzeugsystem (Item, 1.69) oder einzelne Elemente (1.32). Dabei steht ASIL D für das höchste und ASIL A für das niedrigste Sicherheitslevel [Ros_14].
Baseline	Eine Baseline ist ein Satz von Arbeitsprodukten, der formal geprüft und genehmigt wurde und dann als Basis für weitere Arbeiten dient und der nur durch Änderungsverfahren verändert werden kann. Eine Baseline stellt die Zuweisung einer eindeutigen Bezeichnung an eine Konfigurationseinheit oder eine Gruppe von Konfigurationseinheiten und zugehörigen Einheiten dar. Im Verlauf der Entstehung eines Produkts können mehrere Baselines verwendet werden, um seine Entwicklung und Prüfung zu lenken [CMMI_11].
Basis Aktivitäten	Basis-Aktivitäten sind vom ISaPro® definierte Prozessschritte. Das sind einerseits abgeleitete Forderungen von Normen und Standards und andererseits bewährte domain- oder unternehmensspezifische Aktivitäten.
Benutzer	Benutzer sind alle Personen, die in irgendeiner Art und Weise mit dem fertiggestelltem System zu tun haben, vorausgesetzt, dass deren Wünsche Einfluss auf das System haben.
Confirmation Measures	Confirmation Measures bestehen aus Confirmation Reviews, Safety Assessments und Safety Audits. [ISO26_11, Teil 1, 1.17]
Confirmation Review	Ein Confirmation Review liefert den Nachweis, dass das Arbeitsprodukt die Anforderungen der jeweiligen Safety-Norm erfüllt, unter der Prämisse des geforderten Unabhängigkeitslevels der Reviewer. [ISO26_11, Teil 1, 1.18]

Derived Safety Requirements	Derived Safety Requirements (DSaR) sind zusätzlich definierte Sicherheitsanforderungen, die nach dem Requirements Engineering Prozess, aufgrund neu identifizierter Hazards, definiert werden. DSaR lösen einen zusätzlichen Iterationszyklus aus, indem die DSaR spezifiziert, analysiert und realisiert werden.
Endogene Hazards	Endogene Hazards sind Gefährdungen, deren Ursachen systemintern, also endogen liegen. Das sind beispielsweise Hardware-Ausfälle und/oder Software-Fehler.
Exogener Hazard	Exogene Hazards sind Gefährdungen, deren Ursachen systemextern, also exogen liegen. Das sind beispielsweise Einflüsse aus dem System-Umfeld (z.B. Hitze, Kälte, Straßenverhältnisse) und/oder Einflüsse aus der System-Umgebung (z.B. andere Systeme wie Steuergeräte, Mechanik). Sie beeinflussen das System von außen derart, dass das System von innen nach außen an seiner definierten Systemgrenze eine Gefährdung darstellt.
Funktionale Sicherheit	Funktionale Sicherheit ist Teil der Gesamtsicherheit, bezogen auf die EUC und das EUC-Leit- oder Steuerungssystem, die von der korrekten Funktion des E/E/PE-sicherheitsbezogenen Systems, sicherheitsbezogenen Systemen anderer Technologien und externer Einrichtungen zur Risikominderung abhängt (basierend auf [IEC08_10 Teil 4, Punkt 3.1.12)
Gesamtsicherheit	Die Gesamtsicherheit bezieht sich auf alles, was die Sicherheit betrifft, das sind beispielsweiese Arbeitssicherheit, Systemsicherheit, funktionale Sicherheit, mechanische Sicherheit.
Hazard	(dt. Gefährdung) Gefahrenquelle mit dem Potential Schaden am System, an der Umwelt, an Personen oder Einrichtungen und Besitz zu verursachen (basierend auf [MIL_05]).
Item	Unter einem Item versteht die ISO 26262 ein System oder Feld von Systemen, welche eine Funktion auf Fahrzeugebene realisiert, auf welches die ISO 26262 angewendet werden soll [ISO26_11].
Latente Fehler	Latente Fehler sind Fehler, die nicht direkt zur Verletzung eines Sicherheitszieles führen, jedoch als unentdeckter Fehler in Kombination mit weiteren Fehlern eine gefahrbringenden Ausfall erzeugen.
Makro-Konzept	Ein Makro-Konzept ist ein erster Entwurf des Projekt-Vorhabens und besteht aus einem organisatorischen, einem technischen und einem Safety-Konzept. Es wird im Problemraum erstellt und dient zur Entwicklung des System-Entwurfs, zur Identifikation der Sicherheitsanforderungsstufe und zur Grob-Planung in der Vor-Projektsphase.
OEM	In der Automobilbranche wird darunter ein Unternehmen verstanden, welches als Erstausrüster bzw. Original-Equipment-Manufacturer (OEM) Produkte herstellt und unter eigenem Namen in den Handel bringt. Dieses Unternehmen kann dabei Produkte, die ein anderer Hersteller produziert, in die eigenen Produkte integrieren und unter eigenem Namen in den Handel bringen.
Projektphase	Projektphasen sind zeitlich strukturierte Abschnitte des Projektverlaufs, die von unterschiedlicher Dauer sein können. Sie werden meist mit Meilensteinen oder Stage Gates abgeschlossen, deren Erreichung ein Maß für den Projektfortschritt darstellt.
Prozess	Ein Prozess ist ein Satz zusammenhängender Tätigkeiten, die Eingaben in Ausgaben umwandeln, um einen vorgegebenen Zweck zu erreichen [CMMI_11].
Prozesslandschaft	In einer Prozesslandschaft sind jene Prozesse dargestellt, die einerseits die

	Leistung für den Kunden erbringen und andererseits auch alle Prozesse, die diese Leistungserbringung steuern, unterstützen und verbessern. Eine Prozesslandschaft teilt die Prozesse in die Kategorien Management-Prozesse, Geschäftsprozesse, unterstützende Prozesse und Mess-/Analyse- und Verbesserungsprozesse ein.
Prozessreifegradmodell	Ein Prozessreifegradmodell formuliert die Anforderungen an ein Qualitätsmanagementsystem und gibt einen Rahmen für ein Assessment vor, um den Reifegrad der Organisation festzustellen. Prozessreifegradmodelle sind zum Beispiel CMMI und SPICE (ISO 15504).
Safety Application Condition	SACs (deutsch "sicherheitsbezogene Anwendungsbedingungen") sind laut EN 50129 [CEN29_03, S. 27] "Regeln, Bedingungen und Einschränkungen, […] die bei der Anwendung des/der Systems/Teilsystems/Einrichtung eingehalten werden müssen."
Safety Assessment	Ein Safety Assessment ist eine Evaluierung der funktionalen Sicherheit, die durch das Item erreicht werden soll. [ISO26_11, Teil 2, Kapitel 6.2]
Safety Audit	Ein Safety Audit ist eine Evaluierung der Implementation des geforderten Prozesses für die Aktivitäten der funktionalen Sicherheit. [ISO26_11, Teil 2, Kapitel 6.2]
Safety Case	Ein Safety Case (Sicherheitsnachweis) ist eine schlüssige Argumentation, die die ausreichende Sicherheit eines gegebenen Systems in seiner Umgebung über seine Lebensdauer belegt.
Safety Goal	Ein Safety Goal ist nach der ISO 26262 ein Top-level Safety Requirement, welches das Resultat einer Hazard-Analyse und eines Safety-Risiko-Assessments ist [ISO26_11]. Ein Safety Goal ist den Hazards übergeordnet und kann mehrere Hazards enthalten.
Safety Lifecycle	Beschreibt alle Safety-Prozesse und deren Safety-Aktivitäten, welche notwendig sind um ein technisches System auf einen akzeptablen Risiko-Level zu halten.
Safety Measures	Safety Measures sind organisatorische Maßnahmen, die bei der Entwicklung und/oder Betrieb eines Systems notwendig sind, um die geforderte Sicherheitsanforderungsstufe zu erreichen. Dazu zählen beispielsweise Forderungen nach personeller Unabhängigkeit, Forderungen bestimmte Prozesse zu befolgen, Forderungen nach unabhängigen Überprüfungen etc.
Safety Mechanismen	Safety Mechanismen sind technische Maßnahmen, die im System umgesetzt werden müssen, um die geforderte Sicherheitsanforderungsstufe zu erreichen. Dazu zählen beispielsweise Forderungen nach Redundanz, Diversität, Backupmechanismen etc.
Safety Requirements on Elements	Safety Requirements on Elements sind den Sub-System, eines Systems, zugeteilten Safety Requirements.
Scope of Work	Ein Scope of Work (SoW) definiert den Arbeitsumfang des Projekts auf Top-Level Ebene und beinhaltetet eine kurze Projektbeschreibung mit Zweck des Projekts, Operationalen Requirements und der Definition des Lieferobjekts, das im Projekt entwickelt werden soll. Davon werden die Projektziele und Nicht-Projektziele abgeleitet. Der Projektstarttermin und Projekt-Endtermin legen dabei den zeitlichen Rahmen fest.
Service	Als Service wird das System-Verhalten bezeichnet, welches das System-Umfeld erwartet.
Sicherheitsanforderungsstufe	Maßzahl für den erforderlichen Vertrauensgrad, dass ein System seine

	spezifizierten Sicherheitseigenschaften erfüllt.
Sicherheitsbezogenes System	Ein sicherheitsbezogenes System ist ein System, das sowohl die erforderlichen Sicherheitsfunktionen ausführt, die notwendig sind, um einen sicheren Zustand für das übergeordnete System (EUC) zu erreichen und aufrecht zu erhalten. [vgl. IEC08_10]
Sicherheitsfunktion	Eine Sicherheitsfunktion ist eine Funktion die von einem E/E/EP-sicherheitsbezogenen System, einem sicherheitsbezogenen System anderer Technologie oder externer Einrichtungen zur Risiko-Minderung ausgeführt wird, mit dem Ziel, unter Berücksichtigung eines festgelegten gefährlichen Vorfalls einen sicheren Zustand für das übergeordnete System (EUC) zu erreichen oder aufrecht zu erhahlten [vgl. IEC08_10]
Sicherheitsintegrität	Die Sicherheitsintegrität ist die Wahrscheinlichkeit, dass ein sicherheitsbezogenes System die geforderte Sicherheitsfunktion unter allen festgelegten Bedingungen innerhalb eines festgelegten Zeitraums anforderungsgemäß ausführt.
Sicherheitskritisch	Ein System wird als sicherheitskritisch eingestuft, wenn es im Zuge einer Fehlfunktion, Fehlzustand oder Ausfall zu einem gefährlichen Vorfall kommt und unter Kontrolle gehalten werden muss. Hier spricht man von direkter Gefährdung.
Sicherheitsrelevant	Der Begriff "sicherheitsrelevant" wird verwendet, wenn ein Fehler im System oder eine Kombination von Fehlern, aber auch ein Zusammenwirken von Ereignissen, zu einer Bedrohung führen. Der Begriff sicherheitsrelevant bewegt sich in einem weiteren Kontext als sicherheitskritisch, da ein Einzelfehler nicht zwingend kritisch sein muss. Das Zusammenwirken von verschiedenen Fehlern und/oder Ereignissen kann hingegen gefährliche Auswirkungen haben. Hier spricht man von einer indirekten Gefährdung.
Stage Gate	Stage Gates sind klar definierte Stationen zwischen den einzelnen räumlichen Phasen (Stages) des ISaPro® und mit klaren Go- bzw. No-Go-Kriterien (Gates) hinterlegt. Stage Gates sind im Unterschied zu den Meilensteinen an Leistungsziele gebunden (z.B. Funktionen, Kosten etc.)
Stakeholder	Eine Interessensgruppen, Person oder Organisationen die ein potentielles Interesse an dem zukünftigen, zu entwickelnden, System haben und somit auch Anforderungen an das System stellen können [Par_10]. Das sind beispielsweise Kunden, Benutzer, Interessensträger, Wissensträger, Projektbeteiligte oder Systembetroffene.
System-Kontext	Mit dem System-Kontext werden die System-Umgebung und die Interaktionen mit benachbarten Systemen (technische, soziale und sozio-technische) definiert. Der Systemkontext grenzt die relevante Umgebung des Systems vom nicht relevanten Teil ab. Er legt fest, welche materiellen und immateriellen Objekte der Umgebung betrachtet werden. Schnittstellen beschreiben die Interaktion von Objekten des Systemkontextes mit dem System.
System Safety Lifecycle	Der System Safety Lifecycle besteht aus den Prozessen, Preliminary Hazard Identification (PHI), Functional Safety Evaluation (FHE), Preliminary Safety Evaluation (PSSE) und System Safety Evaluation (SSE). Der System Safety Lifecycle auf operationaler Ebene enthält noch den Prozess „Operational System Safety Evaluation“.
System-Umfeld	Das System-Umfeld besteht aus den System-Umwelten (z.B. Strahlung, Dämpfe, Witterungsverhältnisse) und dem Einsatzbereich des zu

	entwickelnden Systems.
System-Umgebung	Die System-Umgebung besteht aus der unmittelbaren und der mittelbaren System-Umgebung. In der unmittelbaren System-Umgebung befinden sich alle direkt angrenzenden anderen Systeme, die mit dem zu entwickelnden System physikalisch verbunden sind. In der mittebaren System-Umgebung befinden sich andere Systeme, die das zu entwickelnden System nur indirekt beeinflussen können (z.B. Vibration, Lärm, gefährliche Substanzen).
System-Umwelten	Zu den System-Umwelten gehören diejenigen Einflussfaktoren, die auf das zu entwickelnde System wirken. Das sind beispielsweise Witterung, chemische Reaktionen, oder elektromagnetische Entladungen.
Top-Level Hazards	Top-Level Hazards sind diejenigen Hazards, die in der Vor-Projektphase identifiziert wurden.

Abkürzungen

ABS	Antiblockiersystem
AIS	Abbriviated Injury Scale
ANSI	American National Standard Institute
ASIL	Automotive Safety Integrity Level
BA	Basis Aktivitäten
Bi_BNS	Bidirektionales benachbartes System
BNS	Benachbartes System
BrPd	Bremspedal
CAN	Control Area Network
CCB	Change Control Board
CMMI	Capability Maturity Model Integrated
COTS	Commercial of the Shelf
CU	Control Unit
DEF STAN	Defence Standard
DO	Document
E/E	Electric/Electronic
E/E/EP	Electronisch/Elektrisch/Elektronisch Programmierbar
EBV	Elektronische Bremskraft Verteilung
ECS	Equipment Control System
eFFA	Extended Functional Failure Analysis
EG	Expansion Guardian
EMI	Electromagnetic Interference
EMV	Elektromagnetische Verträglichkeit
ESP	Elektronisches Stabilitätsprogramm

EUC	Equipment under Control
exHA	Exogene Hazard Analyse
FAA	Federal Aviation Administration
FDIS	Final Draft International Standard
FFA	Functional Failure Analysis
FFG	Forschungsförderungsgesellschaft
FHE	Functional Hazard Evaluation
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
FuSyR	Funktionale System Requirement
GPM	Gesellschaft für Projektmanagement
GSN	Goal Structuring Notation
HAZOP	Hazard and Operability Study
HI	Hazard-Identifikation
HI&RA	Hazard-Identifikation and Risk Assessment
HSAE	Hardware Safety Architecture Evaluation
HSCTE	Hardware Safety Component and Test Evaluation
HSRE	Hardware Safety Requirements Evaluation
HW	Hardware
HyA	Hydroaggregat
ICB	International Competence Baseline
IEC	International Electronical Commission
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
IPMA	International Project Management Association
ISaPro®	Integrativer Safety Prozess
ISO	International Organization for Standardization
ISO/TS	International Organization for Standardization / Technical Specification
KM	Konfigurationsmanagement
KS	Kipp-Schalter
Me_BNS	Menschliches benachbartes System

MIL-STD	Military Standard
MR	Motorrad
nFuSyR	nicht-funktionalen System Requirements
NHTSA	National Highway Traffic Safety Administration
OK	Organisatorisches Konzept
PHB	Projekthandbuch
PHI	Preliminary Hazard Identification
PHL	Preliminary Hazard List
PjI	Projekt Initialisierung
PjM	Projektmanagement
PjP	Projektplanung
PKA	Projekt Kontext Analyse
PMI	Project Management Institute
PPL	Projekt-Prozesslandschaft
PSP	Projektstrukturplan
PSSE	Preliminary System Safety Evaluation
QM	Qualitätsmanagement
RA	Risk Assessment
RDS	Raddrehzahlsensoren
SA	Safety-Analyse
SAC	Safety Application Condition
SEI	Software Engineering Institute
SG	Stage Gate
SIL	Safety Integrity Level
SK	Safety Konzept
SoW	Scope of Work
SPICE	Software Process Improvement and Capability Determination
SRoE	Safety Requirements on Elements
SRS	Safety-related System (E/E/EP safety related system)
SS	Side-Stand
SSAE	Software Safety Architecture Evaluation

SSCE	Software Safety Component Evaluation
SSCS	Side-Stand Control System
SSCTE	Software Safety Code and Test Evaluation
SSE	System Safety Evaluation
SSRE	Software Safety Requirements Evaluation
SW	Software
SWAL	Software Assurance Level
SwCI	Software Criticality Index
SyD	System Design
TK	Technisches Konzept
TLF	Top-Level Funktionen
TLHa	Top-Level Hazards
TLR	Top-Level Requirements
TLSAC	Top-Level Safety Application Condition
TLSaR	Top-Level Safety Requirements
Uni_BNS	Unidirektionale benachbarte Systeme
V&V	Verifikation und Validation

Tabellenverzeichnis

Tabelle 1: Liste von HAZOP-Guidewords aus [Def58_00] 45
Tabelle 2: Beispiel einer tabellarischen FMEA 48
Tabelle 3: Basis-Aktivitäten der Projekt-Initialisierung 62
Tabelle 4: Zuordnungstabelle Engineering-Prozesse zu den Safety-Prozessen 68
Tabelle 5: Basis-Aktivitäten der Konzeptionierung 73
Tabelle 6: Herleiten der Top-Level Funktionen und der funktionalen Top-Level Requirements 75
Tabelle 7: Basis-Aktivitäten der PHI 77
Tabelle 8: Basis-Aktivitäten der Projektplanung 87
Tabelle 9: Basis-Aktivitäten des Requirements Engineerings 94
Tabelle 10: Basis-Aktivitäten der FHE 99
Tabelle 11: Basis-Aktivitäten des System Design Prozesses 101
Tabelle 12: Basis-Aktivitäten der PSSE 104
Tabelle 13: Inhalte der einzelnen Shells im Shell-Modell 120
Tabelle 14: Failure Analyse und Hazard Beschreibung anhand des Hazard Triangles 134
Tabelle 15: Basis-Aktivitäten der Projekt-Initialisierung im SSCS 146
Tabelle 16: SSCS: Basis-Aktivitäten der Konzeptionierung im SSCS 151
Tabelle 17: Ergebnisse aus der Shell-Analyse des SSCS 155
Tabelle 18: Top-Level Funktionen und Funktionale Top-Level Requirements des SSCS 156
Tabelle 19: Basis-Aktivitäten der PHI im SSCS 158
Tabelle 20: Extended Functional Failure Analysis (SSCS) 159
Tabelle 21: Ergebnisse der Exogenen Hazard Analysis (SSCS) 161
Tabelle 22: Ergebnisse der Safety-Risiko-Bewertung des SSCS im Problemraum 164
Tabelle 23: Top-Level Hazards, ASIL, Safety Goals und -Requirements des SSCS 168
Tabelle 24: Basis-Aktivitäten der Projektplanung im SSCS 171

Tabelle 25: Basis-Aktivitäten des Requirements Engineerings im SSCS 175
Tabelle 26: Gesetzliche Anforderungen (SSCS) 176
Tabelle 27: Forderungen des Safety Auditors (SSCS) 176
Tabelle 28: Anforderungen der Service-Werkstätten (SSCS) 177
Tabelle 29: Anforderungen der Hersteller (SSCS) 178
Tabelle 30: Funktionale Requirements (SSCS) 178
Tabelle 31: Safety Requirements (SSCS) 179
Tabelle 32: Interface-Requirements (SSCS) 179
Tabelle 33: Safety Application Conditions an die Stakeholder (SSCS) 180
Tabelle 34: Basis-Aktivitäten der FHE im SSCS 181
Tabelle 35: Liste von HAZOP Guidewords für das SSCS 184
Tabelle 36: HAZOP-Ergebnisse (SSCS) 186
Tabelle 37: Operational Situations auf Fahrzeugebene (SSCS) 187
Tabelle 38: Ergebnisse aus der FHE (SSCS) 187
Tabelle 39: Basis-Aktivitäten des System Designs (SSCS) 189
Tabelle 40: Systematische Identifikation der System-Schnittstellen und Sub-Systeme (SSCS) 190
Tabelle 41: Identifikation der System-Schnittstellen und Sub-Systeme (SSCS) 191
Tabelle 42: Zuteilung der funktionale System Requirements FuSyR_1 und FuSyR_2 (SSCS) 194
Tabelle 43: Zuteilung der nicht-funktionalen System Requirements nFuSyR_1 bis nFuSyR_4 (SSCS) 195
Tabelle 44: Zuteilung des nicht-funktionalen Safety Requirements nFuSaR_1 (SSCS) 196
Tabelle 45: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_2 (SSCS) 196
Tabelle 46: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_3 (SSCS) 197
Tabelle 47: Zuteilung des nicht-funktionalen Safety Requirement nFuSaR_4 (SSCS) 197
Tabelle 48: Zuteilung der nicht-funktionalen Safety Requirements nFuSaR_5 (SSCS) 197
Tabelle 49: Basis-Aktivitäten der PSSE im SSCS 199
Tabelle 50: Hazard-Traceability-Matrix des SSCS 199
Tabelle 51: FMEA des SSCS 201
Tabelle 52: Fehlfunktionen der SRoE im SSCS 204

Abbildungsverzeichnis

Abbildung 1: System Accidents und deren Ursachen 5
Abbildung 2: Konzepte der Systemtheorie ([Rop_09, S. 76] 13
Abbildung 3: Fehler-Kette nach Laprie [AvLa_04] 14
Abbildung 4: Dependability Tree [AvLa_04] 15
Abbildung 5: Hazard Triangle [Eric2_05, S. 17] 17
Abbildung 6: Safety vs. System Safety vs. Functional Safety 19
Abbildung 7: Zusammenhang EUC, ECU und SRS nach der IEC 61508 21
Abbildung 8: Safety Lifecycle der IEC 61508 [IEC08_10, Teil 1, S.18] 22
Abbildung 9: Safety Lifecycle der ISO 26262 [ISO26_11, Teil 2, S.4] 24
Abbildung 10: System Safety Process des MIL-STD 882E [MIL_05, S.9] 27
Abbildung 11: Prozess-Sequenzen für Software-Projekte nach der DO 178C [DO_11, S.22] 32
Abbildung 12: Prozess-Modell des Integrativen Safety Prozesses (ISaPro®) 33
Abbildung 13: System Safety Lifecycles basierend auf der IEC 61508 und der ISO 26262 36
Abbildung 14: Beispiel einer Fault Tree Analyse 47
Abbildung 15: GSN Symbole und deren Bedeutung, Symbolik nach [Ori_11] 49
Abbildung 16: Einfache GSN Struktur 50
Abbildung 17: Klassisches V-Modell nach Boehm 52
Abbildung 18: Prozesse im Problemraum 59
Abbildung 19: Zeitlich-systematischer Ablauf des Problemraumes 60
Abbildung 20: Scope of Work – Template 63
Abbildung 21: V-Modell, angepasst auf eine Projekt-Situation (Beispiel) 67
Abbildung 22: Engineering Lifecycle, abgeleitet vom V-Modell (Beispiel) 67
Abbildung 23: Projekt-Prozesslandschaft mit Engineering- und Safety-Prozessen (Beispiel) 70

Abbildung 24: Projekt-Prozesslandschaft komplett (Beispiel) 71
Abbildung 25: Entwicklung der Systemstruktur über externe Schnittstellen (Beispiel) 76
Abbildung 26: Ablauf des Problemraums als Flow-Chart 82
Abbildung 27: Prozesse im Modellierungsraum 83
Abbildung 28: Zeitlich-systematischer Ablauf des Modellierungsraumes 85
Abbildung 29: Generisches PSP-Pattern 89
Abbildung 30: Requirements-Taxonomie 96
Abbildung 31: Derived Safety Requirements im Modellierungsraum 103
Abbildung 32: Evaluierung der System Design Lösungen 105
Abbildung 33: Fehlerfortpflanzung bis zum Contributory Hazard 106
Abbildung 34: Ablauf des Modellierungsraums als Flow-Chart 110
Abbildung 35: Erweiterte Fehlerkette für elektronische Systeme 114
Abbildung 36: Shell-Modell 118
Abbildung 37: Kontext-Diagram in der Z-Shell 122
Abbildung 38: Stakeholderanalyse (Beispiel) 124
Abbildung 39: Erweiterte Functional Failure Analysis (eFFA) 127
Abbildung 40: eFFA Template 128
Abbildung 41: Exogenen Hazard Analyse (exHA) Template 129
Abbildung 42: System und Operational System Environment in der Erweiterten Fehlerkette 132
Abbildung 43: Risiko-Matrix der ISO 26262 [ISO26_11] 135
Abbildung 44: Software Control Categories aus dem MIL-STD 882E [MIL_05, S.15] 138
Abbildung 45: Software Safety Criticality Matrix des MIL-STD 882E [MIL-STD 882E] 139
Abbildung 46: SWAL nach ED-153 [ED_153] 140
Abbildung 47: Möglichkeit der SIL-Ableitung 140
Abbildung 48: Top-Level Safety Case Pattern 142
Abbildung 49: Projektidee des elektrisch ausfahrbaren Side-Stands 144
Abbildung 50: Scope of Work des SSCS im Problemraum (Status 1st draft) 147
Abbildung 51: V-Modell des SSCS 149
Abbildung 52: PPL des SSCS (1st draft) 150
Abbildung 53: Shell-Modell des SSCS 152
Abbildung 54: Z-Shell des SSCS 155

Abbildung 55: Technischer System-Entwurf des SSCS 157
Abbildung 56: Statement of Work des SSCS (Letztstand) 166
Abbildung 57: PPL des SSCS im Problemraum (Letztstand) 167
Abbildung 58: Erweitertes technisches Konzept mit Bus-Schnittstelle und Bus-Treiber des SSCS 169
Abbildung 59: Stakeholderanalyse (SSCS) 172
Abbildung 60: Projektstrukturplan (SSCS) 173
Abbildung 61: Safety Case Grund-Struktur (SSCS) 182
Abbildung 62: GSN Safety Case Hazard-Identifikation und Mitigation (SSCS) 183
Abbildung 63: Zustandsgraph mit Operating Modes (SSCS) 185
Abbildung 64: System Design des SSCS 193
Abbildung 65: Fault Tree des SSCS der die Verletzung des SaGo_1 modelliert 203
Abbildung 66: SS-Failure-Chain, Contributory Hazard 1 “SS fährt aus” (SSCS) 205
Abbildung 67: Erweiterte GSN, Goal G_0.3.1 Safety Analysen durchgeführt (SSCS) 208
Abbildung 68: Erweiterte GSN, Goal G_0.3.2 Hazard Identifikation komplett (SSCS) 209
Abbildung 69: Erweiterte GSN, Goal G_0.3.3 Identifizierte Hazards mitigiert (SSCS) 209
Abbildung 70: Erweiterte GSN, Goal G_0.4.1 Traceability der Safety Requiements (SSCS) 210

Lebenslauf

Hans Tschürtz
Kerschbaumgasse 3/1/502
A-1100 Wien
Geburtsdatum: 11. September 1958

Berufserfahrung

Seit 2005	FH Campus Wien Leiter des "Vienna Institute for Safety & Systems Engineering" (Leitung diverser Forschungsprojekte, COIN, MA23) Leiter des Master-Lehrgangs „Safety and Systems Engineering" Lektor in diversen technischen Studiengängen im Bereich Engineering- und Management-Disziplinen
2001 – 2005	Philips Audio Video Innovation Centre, Wien Software Release Group Manager
1997 – 2001	Philips Technology Center, Wien Project Manager
1986 – 1996	Philips Competence Center, Wien Senior Engineer Pre-Development Department, Software Development Team Leader

Ausbildung

2009 – 2015	Doktoratsstudium an der TU Wien
2008 – 2009	Donau Universität Krems Masterstudium Qualitätsmanagement, Abschluss: MSc.
2006 – 2008	Donau Universität Krems Masterstudium Prozessmanagement, Abschluss: MSc.
2001 - 2004	FH Campus Wien Studium Technisches Projekt- und Prozessmanagement, Abschluss: DI (FH)
1990 – 1992	TU Wien Universitätslehrgang Automatisierungstechnik, Abschluss: Diplom

Weiterbildung:

Certified Senior Project Manager (IPMA-Level B)

Safety Tutorials University of Cincinnati