Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

René Büscher

Ein Safety-Konzept für Airborne Embedded Systems

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

René Büscher

Ein Safety-Konzept für Airborne Embedded Systems

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Thomas Lehmann
Zweitgutachter : Prof. Dr. Bettina Buth

Abgegeben am 23.05.2014

René Büscher

Thema der Bachelorthesis

Ein Safety-Konzept für Airborne Embedded Systems

Stichworte

Sicherheit, Systemarchitekturen, Gefahren, Risiken, Analysen, FTA, UAV, Autonomes System, Drohne, VHDL, ModelSim, Hardware Simulation, Hardwareentwurf, CPDL, FPGA, Eingebettete Systeme, Luftfahrt

Kurzzusammenfassung

Diese Arbeit beschäftigt sich in ihrem Schwerpunkt mit dem Entwurf einer Safety-Architektur für Airborne Embedded Systems. Dazu werden entsprechende Gefahren- und Risiko-Analysen, zur Beurteilung der Systemsicherheit und der dazugehörigen Komponenten, durchgeführt. Nach den Analysen werden, anhand der Ergebnisse, die Anforderungen an die Architektur definiert sowie die Safety-Anforderungen an die Komponenten. Im letzten Teil der Arbeit wird ein Entwurf der Architektur vorgestellt. Die Implementierung und Verifikation des Entwurfs wurde mithilfe von VHDL und ModelSim durchgeführt.

René Büscher

Title of the paper

A safety-concept for Airborne Embedded Systems

Keywords

safety, system architecture, hazards, risks, analysis, FTA, UAV, autonomous system, drone, VHDL, ModelSim, hardware simulation, hardware design, CPDL, FPGA, embedded systems, aviation

Abstract

In its focus, this work deals with the design of a safety architecture for airborne embedded systems. To assess the safety of the system and associated components, corresponding hazard and risk analyses were carried out. The results of the carried analysis, were used to define requirements for the architecture, as well as the safety requirements on the components. A possible architecture that suits these requirements is proposed in the last part of this work. The approached architecture was implemented in VHDL and verified with ModelSim.

Danksagung

Zunächst möchte ich mich, an dieser Stelle, bei all meinen Unterstützern bedanken. In erster Linie geht mein Dank an meine Betreuer Prof. Dr. Herr Lehmann und Prof. Dr. Frau Buth. Sie standen mir jederzeit mit Rat und Tat zur Seite und gaben mir immer die Möglichkeit mich bei Problemen und Fragen mit Ihnen zusammenzusetzen.

Ein weiterer Dank geht an meine Kommilitonen und Mitstreiter Hagen Hasberg und Alexander Rohrer. Es gab immer die Möglichkeit Probleme und Fragen in unserer kleinen Gruppe zu diskutieren. Auch wenn die Diskussionen häufig ausschweiften, waren diese immer sehr hilfreich und es kamen dadurch meist sehr gute Anregungen zustande. Ein großes „Danke“ für die schöne und heitere Zusammenarbeit.

Auch bei meiner Freundin Vera Brinkers muss ich mich herzlichst bedanken. Sie hat mich immer wieder motivieren können, an der Arbeit weiter zu schreiben, auch wenn ich mich lieber anderweitig beschäftigt hätte. Gerade im Endspurt konnte sie mich immer wieder auf ein Neues ermutigen. Sie hat mich immer unterstützt und mir den Rücken frei gehalten. Zum Korrekturlesen brachte sie viele Stunden und auch Nerven auf.

Nicht zuletzt gebührt meinen Eltern ein großer Dank. Sie haben mich während meines Studiums immer unterstützt, sowohl finanziell als auch emotional. Sie standen bei all meinen Entscheidungen immer voll und ganz hinter mir.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

»Der Anhalter sagt, es ist eine Kunst zu fliegen«, sagte Ford, »oder vielmehr ein Trick. Der Trick ist, daß man lernt, wie man sich auf den Boden schmeißt, aber daneben.« [2, S. 17]

1. Einleitung

Wozu benötigt man „Ein Safety Konzept für Airborne Embedded Systems“? Auf diese und auf andere Fragen wird es im Laufe der Arbeit eine Antwort geben. Als Einstieg wird zunächst das System sowie dessen Umfeld beschrieben werden. Anhand dieser Randbedingungen kann erkannt werden, aus welchem Grund diese Arbeit entstanden ist. Im nächsten Schritt werden die übergeordneten Ziele aufgelistet, die für den Lösungsentwurf benötigt werden. Am Ende wird der Aufbau der Arbeit beschrieben, um die Vorgehensweise nachvollziehen zu können.

1.1. Randbedingungen

Die Bedingungen sind aufgeteilt in das Umfeld, also den Teams und die Organisation des Projekts. Außerdem sollen die allgemeinen Ziele der Teams aufgezeigt werden. Die zweite Randbedingung stellt das System dar, das sich zum Teil noch in der Entwicklung befindet. Auf Grundlage des Systems baut die gesamte Arbeit auf.

1.1.1. Das Umfeld

Das Projekt Airborne Embedded System (PO AES) entstand Anfang 2013. Zentrales Ziel ist die Entwicklung einer autonom fliegenden Drohne, auch Unmanned Aerial Vehicle (UAV) genannt. Die Drohne soll später für den Katastrophenschutz eingesetzt werden. Dazu ist es angedacht eine solche Drohne, in ferner Zukunft, für den Flug in Deutschland zulassen zu können. Daher ist eines der Kernthemen die Systemsicherheit. Der Träger für das System kommt vom Department Fahrzeugtechnik und Flugzeugbau der HAW Hamburg. Das Project BWB AC20.30, im weiteren Verlauf als BWB-Team (Blended Wing Body) bezeichnet, ist für den Träger verantwortlich. Das BWB-Team besitzt bereits ein Modellflugzeug. Sie forschen an neuen Flugzeugformen und benötigen ein neues Messsystem, für den experimentellen Träger, um aerodynamische Messflüge durchführen zu können. Da sich die Interessen der Teams überschneiden, nutzen beide Teams dies, um sich gegenseitig zu unterstützen und einen Wissenstransfer zu schaffen. Das PO AES umfasst des Weiteren eine Wahlpflichteinheit für Studierende der HAW Hamburg, des Departments Informatik. Dadurch sind jedes Semester neue Studierende mit Teilaufgaben der Systementwicklung betraut.

In diesem Abschnitt wird noch einmal detaillierter auf die Verantwortungsbereiche eingegangen. Wie bereits erwähnt arbeitet das BWB-Team an der Entwicklung verschiedener Trägersysteme. Die folgende unvollständige Auflistung soll einen Überblick über die Tätigkeiten des BWB-Teams geben.

1. Entwurf und Konstruktion des Trägers und dessen Komponenten mithilfe von CAD
2. Materialerprobung
3. Verfahrenserprobung
4. Umsetzung und Herstellung des Entwurfs

Die folgende Liste zeigt einige Teilbereiche, für die das Department Informatik verantwortlich ist:

1. Entwurf und Entwicklung der Software/Hardware
2. Entwurf und Entwicklung des Flugcomputer/Logging Systems
3. Planung der IT-Infrastruktur
4. Auswahl der Sensorik und Aktorik
5. Anbindung der Sensorik und Regelalgorithmen
6. Sicherheitsanalysen des Systems

Des weiteren müssen die Schnittmengen der beiden Teams erläutert werden. Entwirft und konstruiert das BWB-Team einen neuen Träger, müssen Schnittstellen definiert werden. Dazu werden mit dem PO AES bereits die ersten Absprachen getroffen. Zum einen benötigt, dass PO AES die Informationen, was mit dem Träger später gemacht werden soll. Zum anderen muss der Informatik mitgeteilt werden, welche Teile des Systems aktiv gesteuert werden sollen. Aus diesen Informationen kann die entsprechende Sensorik und Aktorik gewählt werden. Auf der anderen Seite benötigt das BWB-Team diese Informationen, um entsprechende Komponenten in der Konstruktion zu berücksichtigen. Auch der Flugcomputer muss in der Konstruktion berücksichtigt werden. Ebenfalls wird sich gemeinsam über das Thema der Systemsicherheit ausgetauscht. Nachdem die notwendigen Informationen ausgetauscht wurden, kann das PO AES das System entwerfen, umsetzen und testen sowie das BWB-Team seinen Kernaufgaben nachgehen.

1.1.2. Das System

Zunächst sollte zu der Systemarchitektur gesagt werden des es sich lediglich um ein Konzept handelt. Die hier verwendete Architektur lehnt sich an eine interne Arbeit an [7]. Die Anforderungen und Spezifikationen der Architektur sind zum größten Teil nur handschriftlich erfasst und somit nur für Projektteilnehmer ersichtlich. Im Kapitel 6 wird noch detaillierter auf diese Problematik eingegangen. Für den Verlauf dieser Arbeit wird die in diesem Abschnitt beschriebene Architektur verwendet.

Die folgende Architektur ist somit ein erster Vorschlag. Sie kann der Abbildung 1.1 entnommen werden kann. Zentrales Element in dieser ist die sogenannte Flight Control Unit (FCU). Diese sammelt alle Informationen des Systems, speichert diese und verwendet sie zur Weiterverarbeitung und Steuerung. Diese Kernkomponente enthält unter anderem den Autopiloten und die Stabilisierungsalgorithmen. Im späteren Projektverlauf des PO AES soll es möglich sein, weitere „High Level Application“ zu integrieren, beispielsweise eine Missionsplanung.

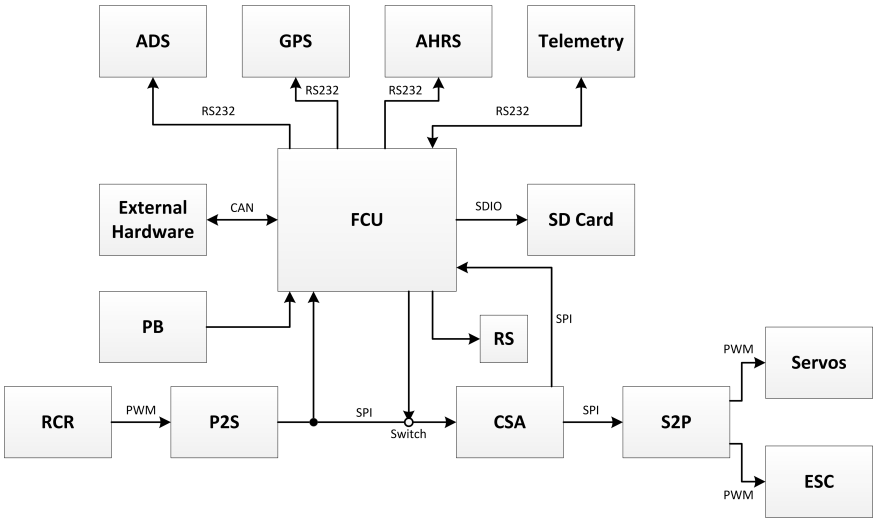

Abbildung 1.1.: Aufbau des aktuellen Gesamtsystems und die Verbindungen aller Komponenten.

Alle Daten, die von der FCU benötigt werden, ermittelt die Sensorik und gibt diese weiter. Diese besteht aus einem Air Data Sensor (ADS), einem Attitude Heading Reference System (AHRS) und einem Global Positioning System (GPS). Über ein Controller Area Network (CAN) kann weitere externe Hardware an das System angebunden werden. Die ADS wird zum Messen der Windgeschwindigkeiten, Windrichtungen und des barometrischen Drucks verwendet. Das AHRS kann die Roll-, Nick- und Gierwinkeländerungen sowie Beschleunigungen und den Kurs ermitteln. Durch den GPS-Sensor kann die Position des autonomen Systems bestimmt werden, aber auch andere Daten, wie zum Beispiel die Höhe, Geschwindigkeit oder Uhrzeit. Die Telemetrie soll wichtige Systemdaten direkt zu einer Ground Control Station (GCS) senden. Außerdem ist es vorgesehen Befehle von der GCS an die FCU zu übermitteln.

Bei der Komponente Radio Control Receiver (RCR) handelt es sich um einen handelsüblichen Empfänger für Funksignale einer Modellbau-Fernbedienung. Der RCR wandelt die Funksignale in PWM Signale (Pulsweitenmodulation) um. Diese Signale werden an die PWM to SPI (P2S) weitergereicht und dort in ein serielles SPI-Signal umgewandelt (Serial Pheriphal Interface). Die Daten werden via SPI an die FCU und an den CSA (Control Surface Allocator)

weitergegeben. Der CSA wandelt die Steuerbefehle der Fernbedienung oder der FCU in Steuersignale für die Aktorik um. Die Aktorik wird im späteren Verlauf erläutert. Die Ergebnisse, des CSA, werden via SPI an die FCU, zur Aufzeichnung, und an die SPI to PWM (S2P) gereicht. Diese decodiert die SPI-Daten und erzeugt daraus die benötigten PWM-Signale für die Aktorik. Letztere besteht zum einen Teil aus den Servomotoren, um den Roll-, Nick- und Gierwinken zu beeinflussen, und den ESC (Electronic Speed Controller) zur Steuerung der Triebwerke. Des Weiteren gibt es ein Power Board (PB), das die Versorgungsspannung der Leistungselektronik (ESC) und der restlichen Komponenten stetig prüft. An dem PB werden alle Akkus angeschlossen und von dort aus wird die Spannung an alle anderen Komponenten verteilt. Entsprechende Pegelwandler sind dort vorgesehen, um alle Komponenten mit der richtigen Spannung zu speisen. Außerdem ist ein Rescue System (RS) vorgesehen, welches durch die FCU oder den RCR ausgelöst werden kann und den Träger sicher zum Boden transportieren soll.

Für den Verlauf der Arbeit ist es nicht weiter notwendig, auf die konkrete Hardware einzugehen. Es genügt eine abstrakte Übersicht der Architektur. Der Aufbau soll lediglich einen Überblick über das System geben. Außerdem wird dieser im späteren Verlauf für die Definition der Systemzustände (Kapitel 3) und für die Gefahren- und Risikoanalysen (Kapitel 4) benötigt. Diese Architektur wird später (Kapitel 5) erweitert, um entsprechende Maßnahmen zur Erhöhung der Sicherheit einzuleiten.

1.2. Ziele und Motivation

Die Ziele der Teams sind bereits bekannt. Generell soll eine autonom fliegende Drohne entwickelt werden. Ein solches ist üblicherweise ein komplexes und sicherheitskritisches Echtzeitsystem. Bei solchen Systemen sollte daher ein besonderes Augenmerk auf die Sicherheit, sowohl Safety als auch Security, gelegt werden. Es sollte besonders darauf geachtet werden, dass die Drohne weder ihrem Umfeld noch sich selber Schaden zufügen kann. Grundsätzlich genießt der letzte Punkt einen geringeren Stellenwert, ein entsprechendes Interesse, das sich der Träger auch selber nicht zerstört, ist dennoch gegeben. Besonders unter dem Gesichtspunkt, dass es sich bei dem Träger um einen Prototypen handelt.

Anhand der beschriebenen Systemarchitektur (Abbildung 1.1), kann direkt erkannt werden, dass dieses System vor keinerlei Ausfällen geschützt ist. Daher ist die Betrachtung und Analyse der möglichen Fehler und Risiken ein großer Abschnitt der Arbeit. Aus den ermittelten Gefahren und Risiken kann eine mögliche Safety-Architektur abgeleitet und verifiziert werden. Ziel der Architektur soll die Erhöhung der Gesamtsicherheit sein und die Toleranz gegenüber einem Ausfall. Entsprechend können folgende Ziele konkret festgehalten werden:

- Systemzustände ermitteln und spezifizieren

- Systemgefahren ermitteln und dokumentieren
- Risiken der Komponenten ermitteln
- Komponenten klassifizieren
- Entwurf einer Safety-Architektur zur Risikominderung
- Safety-Anforderungen für Systemkomponenten dokumentieren
- Implementierung und Test des ersten Lösungsentwurfs

Nach dem die generellen Ziele festgelegt sind, wird nun die persönliche Motivation beschrieben. Sicherheitskritische Systeme faszinieren mich bereits seit geraumer Zeit. Ebenfalls habe ich ein starkes Interesse am Systems Engineering und dem Entwurf von Hardware und eingebetteten Echtzeitsystemen. Aus diesem Grund habe ich mich 2013 entschlossen bei dem PO AES mit zu wirken. Innerhalb dieser Zeit wuchs mein Interesse an der Entwicklung autonomer Systeme, genauer Drohnen, mitzuwirken. Besonders da das Projekt alle meine Interessen in einem Umfeld zusammenfasst. Durch diese Arbeit hatte ich die Möglichkeit viele neue Erfahrungen in der Analyse und Entwicklung sicherheitskritischer Systeme zu sammeln. Außerdem kann ich das erlernte Basiswissen im Verlauf meines Masterstudiums weiter nutzen und vertiefen.

1.3. Aufbau der Arbeit

Im ersten Schritt der Arbeit wurden bereits die Randbedingungen sowie die generellen Ziele beschrieben und festgelegt. Der nächste Schritt ist die Definition grundlegender Begriffe zum Themenumfeld der Sicherheit. Ebenfalls werden allgemeine Ideen zur Erhöhung der Sicherheit beschrieben und es werden Beispielarchitekturen und Open Source Projekte betrachtet und beurteilt. Dadurch sollen die notwendigen Grundlagen zum Bearbeiten des Themas geschaffen werden. Außerdem werden in diesem Abschnitt Gefahren- und Risikoanalysen vorgestellt, die im weiteren Verlauf zur Systemanalyse verwendet werden (Kapitel 2). Im Anschluss daran wird das gegebene System analysiert und die logischen Systemzustände erfasst und spezifiziert (Kapitel 3). Für die folgende Gefahrenanalyse werden die vorher ermittelten Zustände verwendet. Nach der Gefahrenanalyse werden die Risiken ermittelt und kategorisiert. Die verwendete Risikoanalyse entspricht einer stark verallgemeinerten Variante und weicht deutlich von der Praxis ab (Kapitel 4). Der letzte Bearbeitungsschritt widmet sich dem Entwurf und Test der Safety-Architektur, die auf Basis der gesammelten Analyseergebnisse entwickelt wird (Kapitel 5). Im Anschluss daran wird das Fazit gezogen und das Ergebnis zusammengefasst. Außerdem werden die möglichen Aussichten und weiterführende und parallel laufende Arbeiten beschrieben (Kapitel 6).

2. Grundlagen

In diesem Kapitel sollen die Grundlagen zum Themenumfeld erläutert werden. Zunächst wird ganz allgemein auf das Themenfeld Sicherheit, im Sinne von Safety, eingegangen und es werden die grundlegenden Begrifflichkeiten definiert. Im weiteren Verlauf wird der Begriff Sicherheit detailliert betrachtet und bekannte Heran- und Vorgehensweisen vorgestellt. Des Weiteren werden bestehenden autonomen Systemen und Luftfahrsysteme betrachtet. Bei den autonomen Systemen handelt es sich um Open Source Projekte, da es zu kommerziellen Systemen kaum Informationen gibt. Ziel des Kapitels ist es relevante Begrifflichkeiten und Konzepte, zum besseren Verständnis der Arbeit, zu definieren und ein besseres Verständnis, für das Thema Sicherheit, aufzubauen. Außerdem werden Umsetzungen betrachtet, die als Anregung für das Safety-Konzept genutzt werden können.

2.1. Begriffe

Der erste Abschnitt geht auf die grundlegenden Begrifflichkeiten zum Thema Sicherheit ein. Mit dem soeben verwendeten Begriff Sicherheit wird bereits das erste Problem deutlich. Der Begriff ist mehrdeutig.

> „**Security** - the world must not harm the system.“ [3, S. 81]

> „**Safety** is a property of a system that it will not endanger human life or the environment.“ [26, S. 2]

In dieser Arbeit wird der Begriff Sicherheit immer mit der Definition von Safety verwendet, ansonsten wird auf eine anderweitige Verwendung explizit hingewiesen.
Ebenfalls ein wichtiger Begriff, wenn man über Sicherheit spricht, ist der Fehler. Hier ergibt sich dasselbe Problem wie beim Begriff Sicherheit.

> „A **fault** is a defect within the system.
> An **error** is a deviation from the required operation of the system or subsystem.
> A **system failure** occurs when the system fails to perform its required function.“ [26, S. 12]

Im weiteren Verlauf werden die englischen Begriffe, nach obiger Definition, einfach übernommen und für die Arbeit „eingedeutscht“. Dadurch soll die Mehrdeutigkeit vermieden werden. Ein Fault kann in der Art, Dauer und dem Ausmaß [26, S. 114] unterschieden werden und beschreibt die Ursachen. Ein Error beschreibt die Abweichung vom Soll und ein Failure den konkreten Ausfall. Das bedeutet, dass ein oder mehrere Faults zu einem Error und ein oder mehrere Errors zu einem Failure führen können. Durch den oben beschriebenen Failure können Gefahren, im englischen Hazards, entstehen.

> „A **hazard** is a situation in which there is actual or potential danger to people or to the environment.“ [26, S. 33]

In sicherheitskritischen Systemen versucht man, die Gefahren zu identifizieren. Dazu gibt es verschiedene Analysemethoden. Die Methoden werden im späteren Verlauf noch genauer besprochen. Zu jeder Gefahr existiert ein Risiko. Dieses muss ebenfalls analysiert und bewertet werden. Auf die Gefahren und Risiken wird in Kapitel 4 noch genauer eingegangen.

> „**Risk** is a combination of the frequency or probability of a specified hazardous event, and is consequence.“ [26, S. 60]

Gefahren und Risiken sind grundlegende Bewertungskriterien für sichere Systeme. Gefahren und Risiken sollten bekannt sein, um Systeme so zu entwerfen, dass sie einem diese tolerieren können. Außerdem kann dadurch festgelegt werden was für das System gefährlich ist. Zusammengefasst entstehen aus Fehlern die Gefahren und zu jeder Gefahr existiert ein entsprechendes Risiko. Eine Gefahr eines Systems ist beispielsweise der Ausfall einer Komponente. Das Risiko dazu ist im Regelfall ein Wert, der festlegen soll, ob ein solcher Ausfall tolerierbar ist.

Im nächsten Schritt werden wichtige Kennwerte eines sicherheitskritischen Systems definiert. Dabei gelten die beschriebenen Werte nicht ausschließlich für solche Systeme. Die aufgeführten Kennwerte können zur Beurteilung ermittelt oder als Anforderung für Systeme festgelegt werden. Die wesentlichen Begriffe sind die Zuverlässigkeit (Dependability), Verfügbarkeit (Availability), Wartbarkeit (Maintainability) und Ausfallsicherheit (Reliability).

> „**Reliability** is the probability of a component, or system, functioning correctly over a given period of time under a given set of operating conditions.“ [26, S. 20]

> „The **availability** of a system is the probability that the system will be functioning correctly at any given time.“ [26, S. 21]

> „**Dependability** is a property of a system that justifies placing one’s reliance on it.“ [26, S. 24]

> „**Maintainability** is the ability of a system to be maintained.“ [26, S. 24]

Die Kennwerte sind nicht zwingend konstant, sondern können vom Systemzustand abhängig sein. Ein Beispiel ist ein Flugzeug. Steht dieses in einer Halle und ist nicht in Betrieb, ist es zwar hoch zuverlässig, aber nicht verfügbar. Befindet es sich wiederum im Flug, ist es weniger zuverlässig, aber hoch verfügbar. Ein weiterer Aspekt ist die Integrität, auf die in Kapitel 4 noch detaillierter eingegangen wird.

> „The **integrity** of a system is its ability to detect faults in its own operation and to inform a human operator.“ [26, S. 22]

All diese Kennwerte sind hier meist im Zusammenhang zu einem System beschrieben, gelten analog dazu gleichermaßen für Teilsysteme oder Komponenten. Die Kennwerte sind Eigenschaften des Systems und können zur Klassifikation herangezogen werden.

Weitere Begriffe, die besonders im Zusammenhang mit sicherheitskritischen Systemen stehen, lauten Fail-Safe, Safe-State, Safe-Live und Fail-Operational.

> **Safe-State** bezeichnet einen Zustand in dem das System Personen und seiner Umgebung keinen Schaden zufügen kann. [26, S. 22]
>
> Ein System ist **Fail-Safe**, wenn es nach einem Failure immer eine Safe-State erreichen wird. [26, S. 22]
>
> **Safe-Life** sind Kriterien die Aussagen wie lange ein System noch weiter funktionieren muss, auch mit eingeschränkter Funktionalität. Ziel ist das erreichen eines Safe-State.
>
> **Fail-Operational** ist eine Eigenschaft die aussagt das ein System trotz Ausfall weiterarbeiten kann, zum Beispiel durch Redundanz.

Die hier beschriebenen Begriffe spiegeln sich häufig in den Anforderungen wieder und sollten beim Systementwurf beachtet werden. Bei sicherheitskritischen Systemen ist es außerdem üblich, sogenannte Sicherheitsanforderungen zu definieren. Durch eine solche Kategorisierung soll die Darstellung der Anforderungen übersichtlicher werden, was ein menschliches Versagen reduzieren kann. Auch zur Trennung der Zuständigkeiten ist dies hilfreich. Außerdem werden Anforderungen meist noch weiter untergliedert, in funktional und nicht-funktional. Dadurch sollen die tatsächlichen Anforderungen an die Funktionen eines Systems getrennt von den Anforderungen aufgestellt werden, die beispielsweise gesellschaftliche Grundzüge haben. In den Anforderungen sollten klare Systemgrenzen festgelegt werden und das geplante Umfeld sollte dort beschrieben werden. Ein System kann nicht unter alle Einflüssen und in jedem Zustand sicherstellen können, das es einwandfrei funktioniert. Daher ist es unentbehrlich klare Abgrenzungen zu spezifizieren.

Die in diesem Kapitel beschriebenen Definitionen gelten, wie hier definiert, im weiteren Verlauf der Arbeit. An Stellen, wo eine andere oder angepasste Definition verwendet wird, steht ein entsprechender Hinweis und eine Erläuterung.

2.2. Sicherheit

In diesem Kapitel werden Vorgehensweisen und Betrachtungsweisen zur Sicherheit beschrieben und diskutiert. Es wird nur auf die Sicherheit im Sinne von Safety eingegangen. Die Security stellt keinen Schwerpunkt dieser Arbeit und somit auch nicht in diesem Abschnitt dar. Dies hat keinerlei Relevanz für die Arbeit. Zunächst werden Sicherheitsphilosophien vorgestellt, beziehungsweise vergleichbare Aussagen. Daran angeknüpft werden bekannte Normen für das Sicherheitsumfeld besprochen, da diese im heutigen technischen Bereich allgegenwärtig sind. Der letzte Abschnitt widmet sich den Sicherheitsarchitekturen. Man erhält einen Einblick in die theoretischen und praktischen Ansätze zur Erhöhung der Sicherheit. Es soll aufgezeigt werden, was beim Entwurf, berücksichtigt werden sollte und wie Sicherheitsprobleme theoretisch gelöst werden können.

2.2.1. Sicherheitsphilosophien

Welche Bedeutungen haben Philosophien im Bereich der Sicherheit? Laut Duden bedeutet Philosophie „persönliche Art und Weise, das Leben und die Dinge zu betrachten“ [43]. In diesem Bezug kann man auch die Sicherheit des Systems betrachten. Nach obiger Bedeutung ist die Sicherheit somit zunächst ein persönliches „Gefühl“ und Entwickler von sicherheitskritischen Systemen nutzen häufig ihre Erfahrungen und ihr Gefühl zum Beurteilen von Schwächen und möglichen Problemen. Dadurch ist selbstverständlich kein sicheres System gegeben, aber gerade Ingenieure mit Jahrzehnten an Erfahrungen sehen Probleme häufig bevor sie entstehen.

Eine ganz bekannte Aussage ist Murphys Gesetz (Murphy’s law) „Whatever can go wrong will go wrong.“ [45]. Diese Lebensweisheit kam vom amerikanischen Ingenieur, Edward A. Murphy, und sie wird heute immer noch im Zusammenhang von komplexen Systemen und dessen Sicherheit gern zitiert.

Auf den ersten Blick haben die Philosophien keinen direkten praktischen Nutzen für die Sicherheit. Grundlegend werden über Philosophien zu meist Werte und daran Regeln abgeleitet. Es sollte daher berücksichtigt werden das, beim Finden von Lösungen, nicht nur technische, sondern auch gesellschaftliche Aspekte eine sehr wichtige Rolle spielen. Beim Entwerfen von Systemen spielt auch das Team und das Teamklima eine wichtige Rolle. Es sollte ein offener Umgang gepflegt werden. Teammitglieder sollten gemachte Fehler und bekannte Probleme

nicht verschweigen, sondern zugeben und mit den anderen Mitgliedern Lösungen diskutieren und finden. Auch Erfahrungen und ein Bewusstsein für das, was man macht, sind essenziell. Alle Beteiligten tragen zumeist eine große Verantwortung, die ihnen bewusst sein sollte.

2.2.2. Technische Vorgehensweise

In diesem Abschnitt werden allgemeine technische Ansätze betrachtet. Es entspricht einer kleinen Übersicht üblicher Techniken zur Erhöhung der Systemsicherheit. Der Oberbegriff für diese Techniken ist das System Health Management, kurz SHM. Es besteht im Allgemeinen aus [15, Vorwort]:

- „Fail-Safe or fail-operational component/subsystem designs
- „Designing out“ identified failure modes
- Design margins (power, propulsion, supplies, etc.)
- Fault accommodation
- Redundant components and subsystems
- Redundancy management techniques enabling fault tolerance for flight-critical functions
- Fault detection, isolation and recovery, FDIR
- Fault protection.“ [15, Vorwort]

Eine weitere und ähnliche Übersicht über den Inhalt von SHM ist folgende. Diese gilt speziell für verschiedene Luftfahrzeuge [15, Vorwort]:

- „Efficient fault detection, isolation and recovery
- Prediction of impending failures or functional degradation
- Increased reliability and availability of mission systems
- Enhanced vehicle situational awareness for crews
- Condition-based and just-in-time maintenance practices
- Efficient ground processing and increased asset availability“ [15, Vorwort]

Anhand dieser beiden Aufzählung kann zum einen erkannt werden, welche Techniken und Teilgebiete der Systemsicherheit und des Systems Engineering benötigt und verwendet werden. Zum anderen wird gezeigt, welche Eigenschaften an die beteiligten Komponenten vorausgesetzt werden und welche Anforderungen an das System zu stellen sind. Man kann ebenfalls erkennen, dass es unmöglich ist, all diese Bereiche in einer Arbeit unterzubringen.

Durch die verschiedenen Ansätze kann ein System deutlich sicherer entworfen werden. Ein Teilbereich des SHM ist das System Health Monitoring (SH Monitoring). Hierbei steht die Überwachung der Komponenten und dessen Verbindungen (häufig als Links bezeichnet) im Vordergrund. Die Datenmenge, die der Überwachung bereitgestellt wird, sollte ebenfalls gesondert gespeichert werden. Ein sogenanntes Logging ermöglicht eine nachträgliche Auswertung der

Daten. Durch solche Maßnahmen können Fehler zum einen schon erkannt werden, bevor sie auftreten und zum Anderen kann nach einem Ausfall die Ursache leichter gefunden werden. Alle Daten werden zentral zusammengetragen. Entweder werden diese durch einen Computer analysiert und verwertet oder visualisiert und durch einen menschlichen Überwacher verwertet. Auch die Kombination ist möglich. Zum Beispiel ermittelt der Computer einen Fehler und zeigt das Problem dem Überwacher an. Dieser muss nun entscheiden, was zu tun ist. Ein Monitoring kann nicht immer den Ausfall des verhindern. Dafür kann es frühzeitig auf Probleme hinweisen und durch ein entsprechendes Logging kann ein Ausfall besser nachvollzogen werden. Es erleichtert das Identifizieren der Schwachstelle und das Beheben des Problems.

In technischen Systemen unter sicherheitskritischen Bedingungen werden meist ganze Systeme, Teilsysteme oder einzelne Komponenten redundant ausgelegt. Dadurch soll verhindert werden, das aus einem Fault ein Failure entsteht. Es soll eine gewisse Toleranz bezüglich dem Ausfall einzelner Komponenten erreicht werden. Im Bereich der Informatik unterscheidet man zwischen Hardware- und Softwareredundanz. Der Unterschied liegt im Wesentlichen darin, dass entweder die Software oder die Hardware mehrfach ausgelegt wird. In beiden Fällen werden die Ergebnisse verglichen oder auf eine andere Art und Weise auf Fehler geprüft. Es gibt verschiedene Wege ein System redundant auszulegen, ob nun die Verwendung von mehreren gleichartigen Komponenten oder zum Beispiel durch sogenannte rekonfigurierbare Systemen. Mehr Informationen dazu können, unter anderem, in [26, S. 124] und in [28, S. 446] nachgeschlagen werden. Ein weiterer Aspekt bei allen redundanten Systemen ist die sogenannte „Design Diversity“ [26, S. 126]. Durch die Design-Vielfalt sollen und systematische Faults verhindert werden, da Komponenten nach diesem Prinzip von verschiedenen Teams entwickelt werden. Dabei sollten die Spezifikationen aber gleich sein. Dies schützt nicht vor Fehlern in der Spezifikation. Weitere Probleme sind hier zum einen die deutlich höheren Kosten und der menschliche Faktor, da die Entwickler die Spezifikationen unterschiedlich interpretieren könnten. Außerdem könnte es möglich sein das am Ende, das gleiche entwickelt wurde.

Durch regelmäßige Wartung können Systeme ebenfalls sicherer werden. Besonders im Umfeld von Flugzeugen ist die regelmäßige Wartung ein Muss. Jedes Bauteil im Flugzeug hat meist eine festgeschriebene Lebensdauer, nach dem es ausgetauscht werden muss, unabhängig davon, ob es noch voll intakt ist. Dadurch soll dem Systemausfall vorgebeugt werden. Ein Problem, das hierdurch aber auftreten kann, ist das durch die häufige Wartung und das Arbeiten an dem System, Fehler durch den Menschen, zum Beispiel beim Einbau, gemacht werden. Um dieses Problem zu minimieren, wird meist ein sogenannter „Built-In-Test“ genutzt. Dies bedeutet, dass es nach dem Austausch einen Testdurchlauf gibt, wodurch Fehler beim Einbau direkt gefunden werden können [26, S. 103].

Die bisher beschriebenen Vorgehensweisen sollen nur eine Orientierung geben. Es sind Ideen und Ansätze, die in Systemen kombiniert werden können. Es sollte zudem beachtet werden, dass diese Vorgehensweisen nicht das Allheilmittel sind, sondern nur gegen bestimmte Fehlerklassen helfen. In der Tabelle 2.1 wird eine Auswahl von verschiedenen Fehlerklassen

präsentiert. Zu jeder Klasse wurde eine Beschreibung und ein Lösungsvorschlag hinzugefügt. Insgesamt wurden auch bei Weitem nicht alle Vorgehensweisen und Ideen gezeigt. In [15] sind viele weitere Ansätze und Bedingungen, speziell für die Luft- und Raumfahrt, beschrieben.

Fehlerklasse	**Beschreibung**	**Mögliche Gegenmaßnahme**
Systematic/Design Fault	Allgemeine Entwurfsfehler, entspricht einer Kategorie von Fehlern.	Vier-Augen-Prinzip, Reviews der Entwürfe, Testen, Prüfen der Spezifikationen
Intermittent Fault	Faults die erscheinen, wieder verschwinden und später wieder auftauchen. Beispielsweise durch Korrosion eines Kontaktes.	SH Monitoring, Wartung, passende Materialwahl beim Entwurf, Rekonfiguration
Permanent Fault	Fehler existiert auf unbestimmte Zeit.	Beheben des Fehlers. Keine andere Möglichkeit, da der Fehler von Anfang im System war.
Random Fault	Ein Fault durch/in eine/r Komponente.	Erkennung durch SH Monitoring, Tolerierbar durch Redundanz oder Rekonfiguration.
Transient Fault	Ein Fault der zu irgendeinem Zeitpunkt auftritt und zu einem beliebigen zeitlichen Punkt verschwindet, zum Beispiel durch Strahlung.	Redundanz, angepasstes Hardwaredesign, Prüfsummen, Rekonfiguration

Tabelle 2.1.: Tabellarische Darstellung von verschiedenen Fehlerklassen, inklusive Beschreibung, und einer Lösungsidee um einem Fault entgegen zu wirken [26, S. 114ff].

2.2.3. Sicherheitsnormen

Für sicherheitskritische Systeme gibt es eine Vielzahl von verschiedenen Normen. Viele von ihnen gelten für ein spezielles technisches Umfeld. Einen guten Überblick über verschiedenste Normen gibt es in [23]. Im Grunde besteht eine Norm aus verschiedenen Regeln und Leitlinien. Sie soll den Teams bei der Umsetzung und Dokumentation unterstützen. Manche Normen enthalten zudem konkrete Beispiele als Hilfestellung und es sind manchmal Kennwerte festgelegt oder Verfahren beschrieben. Produkte, die später einer Zertifizierung unterzogen werden, müssen nachweisen können, dass nach entsprechenden Normen gearbeitet wurde. Zu dieser Prüfung werden die erstellten Dokumente, Methodiken und Arbeitsabläufe betrachtet und bewertet. Normen spiele beispielsweise beim Erstellen von Spezifikationen für Dritte eine wich-

tige Rolle. Häufig und gerade im Bereich der Sicherheit werden Normen für die Produkthaftung herangezogen. Anhand dieser wird geprüft, welche beteiligte Institution nicht korrekt gearbeitet hat. Außerdem sind sie teilweise gesetzliche Vorgabe.

Die Ausgangsnorm für viele sicherheitskritische Systeme ist die IEC 61508 beziehungsweise beim Deutschen Institut für Normung die DIN EN 61508 (VDE 0803). Der Titel der Norm lautet: „Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme“. Sie besteht aus insgesamt sieben Teilen und einem Beiblatt mit grundlegenden Hinweisen. Im Folgenden eine Übersicht der Teile:

„Teil 1: Allgemeine Anforderungen
Teil 2: Anforderungen an sicherheitsbezogenen elektrischer/elektronischer/programmierbarer elektronischer Systeme
Teil 3: Anforderungen an Software
Teil 4: Begriffe und Abkürzungen
Teil 5: Beispiel zur Ermittlung der Stufe der Sicherheitsintegrität (safety integrity level)
Teil 6: Anwendungsrichtlinie für IEC 61508-2 und IEC 61508-3
Teil 7: Anwendungshinweise über Verfahren und Maßnahmen“ [12, S. 12]

Im späteren Verlauf der Arbeit (Kapitel 4) wird die DIN EN 61508-5 noch detaillierter betrachtet, da diese als Grundlage für die Gefahren und Risikoanalyse verwendet wird. Die DIN EN 61508 ist generisch und kann für jedes technische Gebiet verwendet werden. Alternativ können auch spezialisierte Normen auf dieser aufbauen [12, S. 14].

Im Folgenden werden ein paar bekannt und wichtige Normen für die Systemsicherheit vorgestellt. Für die Automotive Industrie gibt es die ISO 26262. Es handelt sich dabei um eine spezialisierte Norm, die aus der DIN EN 61508 abgeleitet wurde. Sie besteht insgesamt aus zehn Teilen und wurde weniger als Regelwerk geschrieben, sondern beschäftigt sich eher mit den Anforderungen und Methoden zwischen Automobilhersteller und dessen Zulieferer. [21, S. 2f].

Mit der Sicherheit von Eisenbahnen befassen sich die Normen DIN EN 50128 [8] und DIN EN 50129 [9]. Beide Normen sind von der DIN EN 61508 abgeleitet und beschreiben die Regeln für Hardware und Software für Systeme auf der Strecken- und Zugseite.

Auch für die Luftfahrt gibt es Normen, die sich unter anderen mit dem Themenschwerpunkt Sicherheit beschäftigen. Die Radio Technical Commission for Aeronautics (RTCA) verwaltet und definiert eine Vielzahl der Normen und Standards für den Luftfahrtbereich. Mitglieder des Komitees sind die Mehrheit der Großunternehmen aus dem Luftfahrtbereich sowie Organisationen, Hochschulen und Behörden. Interessant sind hier aus Sicht der Informatik die DO-178C

und DO-254. Die DO-178C (Software Considerations in Airborne Systems and Equipment Certification) beschreibt die Software im Umfeld der Luftfahrt, die DO-254 (Design Assurance Guidance for Airborne Electronic Hardware) die Hardware. Die Normen gleichen vom grundlegenden Aufbau, den bereits erwähnten. Von der konzeptionellen Phase und der Beschreibung der Prozesse über verschiedene Analysen bis hin zur Wartung werden hier entsprechende Regeln festgelegt.

Auch im militärischen Bereich und der Raumfahrt gibt es angepasste Normen und Richtlinien, zum Beispiel die United States Military Standards. Diese sind häufig sehr spezielle Regelwerke und zum Teil umfangreicher. Meist entsprechen die Standards einem höheren Sicherheitsstandard. Der höhere Sicherheitsanspruch entspricht häufig einem höherem Anspruch an die Komponenten und Systeme. Nicht immer ist hier, wie es beispielsweise in den bereits vorgestellten Bereichen der Fall ist, der Mensch das wichtigste Gut, sondern die Maschine. Ein gutes Beispiel ist der Astronaut. Dieser ist im Weltraum kaum vor Strahlung geschützt, die Systeme sehr wohl. Eine Übersicht über den Qualitäts- und Sicherheitsanspruch an Bauteile im Luft- und Raumfahrtbereich kann in der Tabelle in [28, S. 449] eingesehen werden.

2.2.4. Safety-Architekturen

In allen kritischen Bereichen werden verschiedene Maßnahmen verwendet, um die Sicherheit zu erhöhen. Etabliert haben sich Architekturen, die speziell für die Erhöhung der Systemsicherheit, entworfen wurden. Im Folgenden werden Standardmuster vorgestellt, die in verschiedensten Bereichen verwendet werden können. Bereits erwähnt wurden redundante Systeme sowie das SH Monitoring. Dies sind meist grundlegende Muster, die innerhalb solcher Architekturen verwendet werden. Die hier beschriebenen Architekturen stellen implementierbare Beispiele dar.

Generische Ideen

Eine erste Lösungsidee ist es ein System fehlertolerant zu entwerfen, und zwar durch einen nicht programmierbaren Sicherheitskanal (vgl. Abbildung 2.1). Nicht programmierbar ist zum Beispiel ein Relais, welches so lange geschlossen gehalten wird, bis ein Sensor die Spannung abfallen lässt und das Relais sich öffnet. Ein sehr anschauliches Beispiel dazu kann bei [26, S. 149] gefunden werden. Es stellt eine Art Rückversicherung dar, falls ein Teil des programmierbaren Systems ausfällt. Das Ganze funktioniert folgendermaßen: Ein System sendet die Sensordaten an eine programmierbare Komponente. Die Komponente entscheidet anhand der Daten, welche Bewegung die Aktorik durchführen soll. Wenn nun die programmierbare Komponente ausfällt, kann die Aktorik nicht mehr gesteuert werden. Dieses Risiko könnte minimiert werden, indem ein weiterer Sensor dieselbe oder eine vergleichbare Messung durchführt. Beim

Überschreiten eines Grenzwertes wird das System direkt in einen sicheren Zustand versetzt. Diese Architektur kann natürlich nicht auf jedes Problem übertragen werden. Diese Architektur verwendet Redundanz und „Design Deversity“ um die Systemsicherheit zu erhöhen.

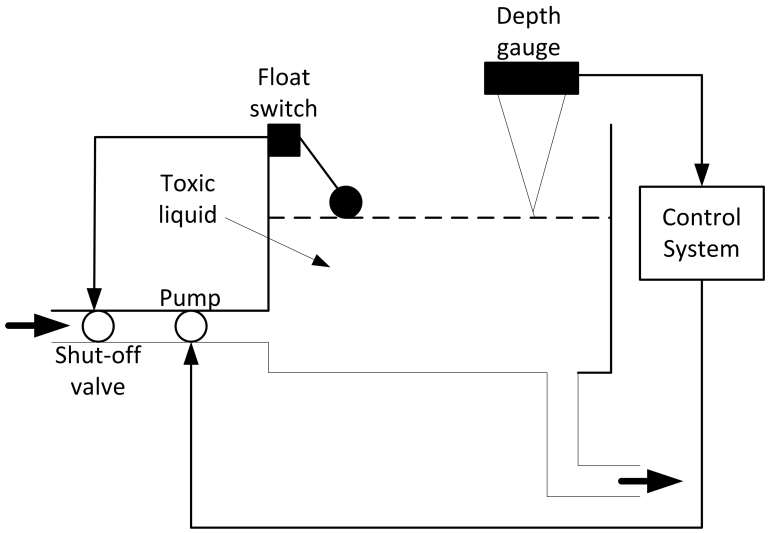

Abbildung 2.1.: Beispiel einer Safety-Architektur mit einem nicht-programmierbaren Sicherheitskanal [26, S. 150].

Eine vergleichbare Architektur ist die sogenannte Emergency Shutdown Architecture (vgl. Abbildung 2.2). Eine kurze Beschreibung mit Beispielen kann bei [4, S. 40] gefunden werden. Im Grunde funktioniert diese Architektur ähnlich wie die zuvor beschriebene. Es gibt ebenfalls ein Kontrollsystem. Dieses wird von einem Emergency Shutdown System überwacht, welches gegebenenfalls das System in einen sicheren Zustand bringen kann. Letztere Komponente sollte ebenfalls die Möglichkeit haben, von außen oder auch anderen Komponenten ausgelöst zu werden. Ein weiteres kleines System überwacht das Emergency Shutdown System. Im Falle, dass dieses ausfällt, kann das zusätzliche System das restliche Gesamtsystem in einen sicheren Zustand versetzen. Dies Architektur nutzt SH Monitoring und Redundanz, um das System sicherer zu machen. Ein Problem der Emergency Shutdown Architecture ist die Überwachung des Überwachers auch bekannt als Zitat: „Who watches the watchmen“. Um diese Problematik zu lösen, könnte man die Architektur dahin gehend erweitern, dass sich die Teilsysteme gegenseitig überwachen.

Interessant sind die X-by-Wire Systeme, da diese immer häufiger in Vehikeln jeder Art verbaut werden. X-by-Wire bedeutet, dass zwischen den Bedienelementen und der Aktorik nicht mehr entsprechende mechanische Verbindungen existieren, sondern die Steuerbefehle als Signal (meist elektrisch) zu der jeweiligen Aktorik übermittelt werden. Alle daran beteiligten Subsys-

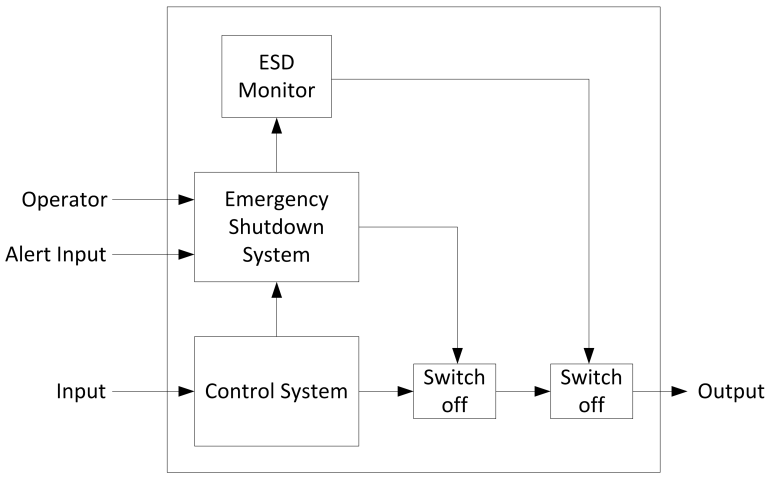

Abbildung 2.2.: Beispiel einer Emergency Shutdown Architecture [4, S. 41].

teme und Komponenten müssen entsprechend sicher sein, um die Funktionalität gewährleisten zu können. Ein passendes Beispiel aus der Praxis ist die Architektur eines Fly-by-Wire Systems. Dies wird in [4, S. 42] beschrieben (vgl. Abbildung 2.3). Es sind zwei gleiche aber unabhängige Steuerungen vorgesehen, linker und rechter Pilot. Dazu kommen jeweils zwei Steuersysteme, die die Steuersignale von beiden Piloten entgegennehmen und jeweils eigene unabhängige Sensoren besitzen. Die Sensoren ermitteln Informationen über das System und dessen Zustand. Beide Systeme sind miteinander verbunden sowie mit dem Autopiloten. Letzteres Teilsystem gibt entsprechende Steuerbefehle weiter. Selbstverständlich wird auch innerhalb der übergeordneten Komponenten mit verschiedenen Methodiken, des SHM, gearbeitet um die einzelnen Komponenten sicher zu entwerfen. Dies Architektur ist ein Beispiel für die mehrfache Verwendung von redundanten Komponenten.

Es existieren noch viele weitere Architekturen. Zum Beispiel solche, die die Fehlertoleranz eines Systems erhöhen sollen. Ein solches Konzept wäre beispielsweise der redundante Aufbau eines Systems. Das Konzept der Redundanz wurde bereits angeschnitten. Des Weiteren existieren spezielle Architekturen, die die Robustheit eines Systems erhöhen sollen. Dazu muss klar sein, was genau mit robust gemeint ist. Eine Art wäre die Robustheit vor Strahlung. Ein Beispiel dafür wird im Kapitel 5 noch einmal aufgegriffen. Für den hier vorgestellten Lösungsentwurf spielen diese keine Rolle.

2.2.5. Zusammenfassung

Beim Entwerfen von sicherheitskritischen Systemen sind Erfahrungen und viel Wissen über das System unentbehrlich. Wenn diese Erfahrungen fehlen oder passende Kennwerte von den

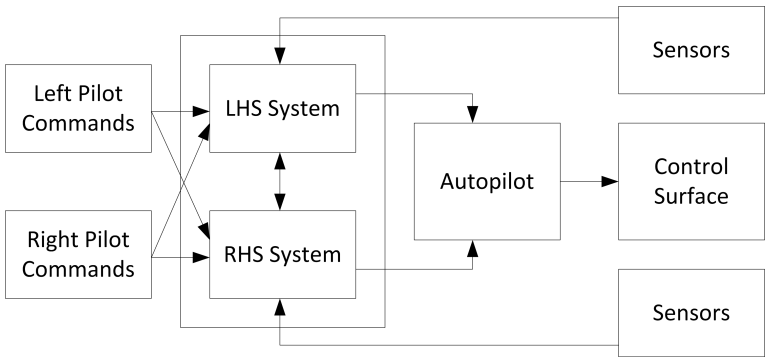

Abbildung 2.3.: Beispielarchitektur eines Fly-By-Wire Systems in Verbindung mit dem Autopiloten [4, S. 42].

Komponenten unbekannt sind, kann versucht werden, dies durch Architekturen zu kompensieren, die bereits erfolgreich eingesetzt werden. Die Mittel, die durch SHM zur Verfügung gestellt werden, sind ebenfalls eine gute Orientierungshilfe. Ein Passendes SH Monitoring und das Loggen sämtlicher Systeminformationen sollte definitiv immer berücksichtigt werden. Gerade bei der Auswertung und der Fehlersuche ist es ein sehr hilfreiches und mächtiges Instrument. Selbstverständlich nur, wenn es entsprechend sinnvoll implementiert und eingesetzt wird.

2.3. Systeme

Im folgenden Kapitel werden zunächst einige Opensource Projekte vorgestellt, die sich bereits mit UAV´s beschäftigen. Diese bieten mindestens eine Software Plattform. Andere Projekte besitzen zum Teil auch eigene Hardware Plattformen. Der zweite Teil betrachtet dann ein Luftfahrsystem im Allgemeinen und geht später auf konkrete Systeme aus der Luft- und Raumfahrt ein. Durch das Betrachten dieser verschiedenen Systeme erhält man einen Überblick über den Stand der Technik. Außerdem können diese als Orientierung und Anregung für das Safety-Konzept verwendet werden.

2.3.1. Autonome Systeme

Inzwischen existieren viele Projekte, die sich mit UAV´s beschäftigen. All diese Projekte zu begutachten und zu bewerten, würde den Umfang dieser Arbeit sprengen. Daher wurden nur Projekte mit besonderen Merkmalen oder großer Akzeptanz geprüft. Der Schwerpunkt bei der Prüfung lag auf den Betriebszuständen und den Sicherheitsmerkmalen.

Paparazzi

Als erstes Projekt sei Paparazzi genannt. Dies ist eines der ältesten Projekte. Das System besitzt drei Betriebsmodi: Manual, Auto 1 und Auto 2. Im Modus Manual wird das Flugzeug über den Transmitter gesteuert. Auto 1 wird verwendet, um den manuellen Flug zu stabilisieren. Der Modus Auto 2 ist der autonome Flug [40]. Im Bereich der Sicherheit hat man sich hier bereits früh Gedanken gemacht. Zu Beginn des Projektes wurde ein Rückfallsystem verwendet (Fly-By-Wire). Ein Pilot hatte immer einen „Safety Link“, um das Flugzeug im Fehlerfall manuell steuern zu können [6, S. 2f] und [5, S. 1]. Auf ihrer Website beschreiben sie dies auch als „Failsafe Switch“. Dort werden weitere Sicherheitsmechanismen oder besser gesagt Sicherheitsregeln beschrieben. Man hat die Möglichkeit, zu beschreiben, was im Falle eines sicherheitskritischen Ereignisses passieren soll. Ein Beispiel einer solchen Regel wäre: Batteriestand kritisch, zurück zur „Home Position“ [41].

PX4 Autopilot

Ein deutlich jüngeres Projekt ist PX4 Autopilot. Es wird von der ETH Zürich getragen. Im Grunde besitzt auch dieses System drei Betriebsmodi: Manual, Assisted und Auto. Diese sind vergleichbar mit den Modi von Paparazzi: Manual, Auto1 und Auto2. Im Detail sind die Zustände Assisted sowie Auto weiter spezialisiert. Dies geschieht über Sogenannte Switche. Dadurch können präzisere Vorgaben gemacht werden. Außerdem gibt es einen Automaten, der das Startverhalten des Systems bis zum Beginn der Mission beschreibt. Ebenfalls dort enthalten ist das Verhalten bei einem Fehler oder Neustart des Systems [32]. Textuell werden dort die Transitionen zwischen den Zuständen beschrieben. Um die Sicherheit des Systems zu gewährleisten, wurde ein extra Board entwickelt (PX4IO) [34]. Dieses erlaubt das überschreiben der Steuersignale der Aktorik durch einen RC-Receiver. Dieses Board ist ausschließlich für die Ein- und Ausgabe zuständig. Für die Steuerung gibt es das PX4FMU [33], welches auch ohne das PX4IO das System steuern kann.

Openpilot

Das nächste Projekt heißt Openpilot und besitzt ebenfalls drei Zustände. Die verschiedenen Modi können noch weiter konfiguriert werden, was vergleichbar mit dem PX4 Autopilot wäre. Dies geschieht über eine Konfigurationsoberfläche [44]. Interessanter ist hier die Implementierungsweise der Zustände. Openpilot ist ähnlich einem Mikrokernel Betriebssystem aufgebaut. Somit stellt jeder Modus ein spezielles Modul dar, welches an den Kernel angebunden wird [38]. An den Kernel werden auch alle Sensoren und Aktoren als auch weitere Features, angebunden. Im Gegensatz zu den anderen Projekten existieren bei Openpilot keine Sicherheitslösungen, durch zusätzliche Hardware oder regelbasiert (vgl. Paparazzi). Dafür gibt es zwei

Plug-ins für die Ground Control Station (GCS), die erwähnenswert sind. Das erste Plug-in heißt Notify. Es wird in der GCS eingebunden und kann dort ebenfalls konfiguriert werden. Durch das Erstellen von Regeln können akustische (Warn-)Hinweise den Operator auf bestimmte Situationen und Probleme aufmerksam machen [37]. Das System Health Plug-in wird ebenfalls in die GCS intrigiert. Es bietet eine grafische Darstellung aller wichtigen Komponenten des Systems. Anhand dieser Übersicht können schnell Fehler und Probleme erkannt werden [42]. Ein erkanntes Fehlerverhalten, durch die Plug-ins, führt zu keiner Fehlerbehandlung oder etwas Vergleichbares. Der Fehler wird lediglich akustisch oder optisch angezeigt.

NG-UAVP

Das letzte Projekt, welches im Detail betrachtet wurde, ist NG-UAVP. Zustände oder Modi, wie es in den anderen Projekten gibt, sind hier in einer ganz anderen Art implementiert. Das Projekt setzt auf sogenannte Controller. Ein Controller ist eine Standard Schnittstelle. Somit kann man ohne größere Probleme einen eigenen Controller programmieren und einbinden. Es werden Controller mitgeliefert, die bereits manuelles, stabilisierendes und autonomes Fliegen ermöglichen [36]. Zum Thema Sicherheit wurden sich nicht viele Gedanken gemacht. Separate Hardware, zur Erhöhung der Sicherheit, existiert nicht. Eine andere Eigenschaft, die entsprechend die Sicherheit geringfügig erhöhen könnte, ist die Verwendung von Status-LEDs und akustischen Signalen. Dadurch können Fehler frühzeitig erkannt werden [39]. Solche Maßnahmen werden auch, zum Teil, von den anderen Projekten verwendet aber nicht extra erwähnt. Ebenfalls werden die dadurch erkannten Fehler für keinerlei Fehlerbehandlung verwendet. Abschließend kann, zu diesem Projekt, gesagt werden das bezüglich der Sicherheit nichts Konkretes beachtet wurde.

2.3.2. Luftfahrtsysteme

In diesem Kapitel geht es um Luftfahrtsysteme im Allgemeinen. Zunächst werden die verschiedenen Zustände, in denen sich ein reguläres Flugzeug befinden kann, beschrieben. Danach wird ganz allgemein ein Autopilot und dessen Zustände betrachtet. Im letzten Abschnitt wird kurz auf die Sicherheit bekannter Luftfahrtsysteme eingegangen. Es soll ein Einblick in das praktische Umfeld der Luft- und Raumfahrt gegeben werden.

Flugzustände eines Flugzeugs

Im Grunde kann der Flug eines Flugzeugs in sechs verschiedene Phasen eingeordnet werden. Die Phasen können der Abbildung 2.4 entnommen werden.

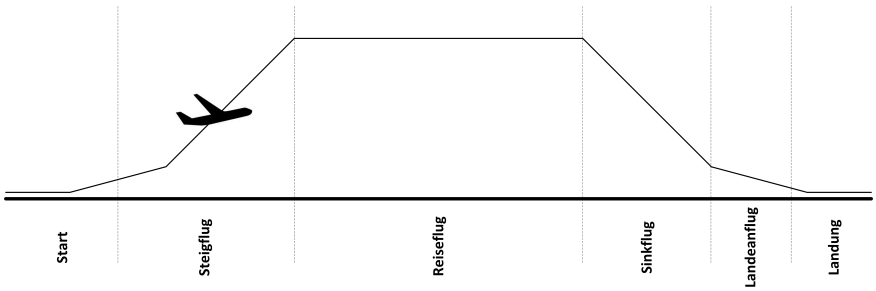

Abbildung 2.4.: Übersichtliche Darstellung, der Flugphasen, eines Großraumflugzeugs. Idee und Umsetzung nach [17, S. 347 und S. 1417].

Die Bedingungen, die im Detail für einen Phasenwechsel verantwortlich sind, können hier [16] entnommen werden. Ebenfalls existieren bei einem normalen Flugzeug verschiedenste Zustände. Ein heutiges Flugzeug fliegt im Regelfall mit einem Autopiloten. Aber es ist immer noch möglich, das Flugzeug manuell zu steuern. Wobei die direkte manuelle Steuerung meist nur noch bei kleineren Maschinen durchführbar ist. Ein heutiges Flugzeug wird normalerweise durch Fly-by-Wire angesteuert, also durch elektronische Unterstützung. Die Steuersignale des Piloten werden geregelt beziehungsweise gedämpft. Wenn wir nun einen Vergleich zu den im vorherigen Kapitel beschriebenen Projekten ziehen, sehen wir, dass ein normales Flugzeug ebenfalls die Modi manuell, stabilisierend sowie autonom besitzt.

Beispiele sicherer Luftfahrtsysteme

Im Anschluss werden noch kurz einige Beispiele aus der Luft- und Raumfahrt aufgelistet.

In diesem Abschnitt werden einige Beispiele, sicherer Systeme beschrieben. Dabei wird hauptsächlich auf die originalen Arbeiten verwiesen. Es werden nur Besonderheiten herausgestellt. Die Architekturen, die vorgestellt werden, sind alle sehr interessant und es ist lohnenswert sich diese genauer zu betrachten. Aber in unserem Umfeld (Kapitel 1) und anhand der aktuellen Systemkonfigurationen sind diese stark überdimensioniert. Daher werden sie nur am Rande betrachtet. Anhand dieser Beispiele kann gesehen werden wie ein Teil der vorgestellten Techniken in konkreten Systemen verwendet werden.

Eine sehr gute Übersicht für Luft- und Raumfahrzeuge bietet die Arbeit [29, S. 483]. Es werden generelle Eigenschaften und Sicherheitsbedingungen für Luft- und Raumfahrzeuge beschrieben. Im Anschluss daran werden konkrete Architekturen vorgestellt, unter anderem vom Mars Pathfinder, Airbus A330/A340/A380 und der Boing 777. Beispielsweise wird der Einsatz von Software Redundanz beim Mars Pathfinder ausführlich beschrieben, als Alternative zur Hardware Redundanz. Bei Luftfahrtsystemen ist die Ausarbeitung etwas weniger umfangreich.

Einen guten zeitlichen Überblick und die entsprechenden genutzten SHM Features gibt die Tabelle 29.2 in [28, S. 479]. Um auch einen allgemeinen Eindruck zum Health Management kommerzieller Flugzeuge zu erhalten, kann noch [18, S. 588] empfohlen werden.

2.3.3. Zusammenfassung

Alle Opensource Projekte besitzen grundlegend dieselben drei Betriebszustände, manuell, stabilisierend und autonom, beziehungsweise alle können auf diese drei Zustände zurückgeführt werden. Die Unterschiede befinden sich in der Implementierung. Vergleichbar sind diese ebenfalls mit dem Umfeld der Luftfahrt und den entsprechenden Autopiloten. Diese einfachen Zustände sind dort ebenfalls enthalten. Zum Bereich der Sicherheit können nicht alle vorgestellten Projekte etwas beitragen. Bei den meisten Projekten existieren akustische oder optische Signalgeber, um auf Fehler hinzuweisen. Das bedeutet nicht, dass ein erkannter Fehler auch entsprechend behandelt wird. Signalgeber könnten auch für unser System interessant werden. Separate Hardware für den Einsatz eines Safety Piloten ist von der Idee her sehr gut. Problematisch ist nur, dass der Ausfall einzelner oder mehrerer Komponenten in den Projekten nicht berücksichtigt wird. Die Betrachtung der wirklich „Großen“ Systeme, im letzten Abschnitt war sehr interessant. Aber für das hier gegebene System sind diese stark überdimensioniert.

2.4. Analysemethoden

In diesem Kapitel sollen zum einen Vorgehensweisen zur Gefahrenanalyse vorgestellt werden und zum anderen das Vorgehen einer Risikoanalyse beschrieben werden. Diese dienen als Grundlage für die im späteren Verlauf der Arbeit durchgeführte Gefahren- und Risikoanalyse im Kapitel 4.

2.4.1. Gefahrenanalyse

Im folgenden Abschnitt werden die wichtigsten und gängigsten Gefahrenanalysen beschrieben und kurz deren Nutzen erläutert.

HAZOP-Studie

HAZOP steht für Hazard and Operability Studies. In der Studie werden durch ein interdisziplinäres Expertenteam die (Teil-)Systeme betrachtet. Das Team ermittelt die Verbindungen zwischen den Komponenten und prüft die Abhängigkeiten. Dabei werden in einer Diskussion,

über die Parameter der Links und Attribute der Komponenten, Leitwörter verwendet um Gefahren zu identifizieren. Die Leitworte sollen das Team auf zu beachtende Aspekte und Probleme aufmerksam machen. Ziel ist es, mögliche Abweichungen vom normalen Betrieb zu finden und die aus den Abweichungen resultierenden Gefahren abzuleiten. Es ist nicht das Ziel, alle möglichen Ursachen zu finden, diese sollen durch eine FTA oder FMEA gefunden werden. Die Tabelle 2.2 zeigt ein anschauliches Beispiel einer einfachen HAZOP-Studie. [26, S. 39ff], [4, S. 61ff]
Ein deutsches Verfahren, welches mit dem hier beschriebenen vergleichbar ist, wäre das PAAG-Verfahren (Prognose von Störungen, Auffinden der Ursache, Abschätzen der Auswirkungen, Gegenmaßnahmen bewerten) [35].

Item	Interconnection	Attribute	Guide word	Cause	Consequence	Recommendation
1	Sensor supply line	Supply voltage	No	PSU, regulator or cable fault	Lack of sensor signal detected and system shuts down	
2			More	Regulator fault	Possible damage to sensor	Consider overvoltage protection
3			Less	PSU or regulator fault	Incorrect temperature reading	Include voltage monitoring
4		Sensor current	More	Sensor fault	Incorrect temperature reading, possible loading of supply	Monitor supply current
5			Less	Sensor fault	Incorrect temperature reading	As above
6	Sensor output	Voltage	No	PSU, sensor or cable fault	Lack of sensor signal detected and system shuts down	
7			More	Sensor fault	Temperature reading too high ? results in decrease in plant efficiency	Consider use of duplicate sensor
8			Less	Sensor mounted incorrectly or sensor failure	Temperature reading too low ? could result in overheating and possible plant failure	As above

Tabelle 2.2.: Beispiel einer HAZOP-Studie aus [26, S. 43].

FMEA

FMEA steht für Failure Modes and Effects Analysis. Diese Technik wird in der Designphase eines Projektes eingesetzt, um mögliche Auswirkungen auf die zu analysierende Komponente und angrenzende Systeme zu finden. Dabei sind die Failures der Komponente bereits bekannt

und dienen als Ausgangspunkt für die Analyse. Die Durchführung der Analyse verläuft tabellarisch. Es wird für ein, mehrere oder alle Systeme oder Komponenten ermittelt, welche Failures möglich sind. Zu allen gefundenen Failures kann festgehalten werden, welche Ursachen zugrunde liegen könnten. Die Auflistung der Ursachen, ist kein Muss und die Liste muss nicht vollständig sein. Außerdem werden Effekte, die auf die Umgebung und das System wirken können, in der Tabelle festgehalten. Ein Beispiel einer FMEA zeigt die Tabelle 2.3. [26, S. 38f], [4, S. 60f]

FMEA for a microswitch						
Ref No	**Unit**	**Failure Mode**	**Possible cause**	**Local effects**	**System effects**	**Remedial action**
1	Tool guard switch	Open-circuit contacts	(a) faulty component (b) excessive current (c) extreme temperature	Failure to detect tool guard in place	Prevents use of machine ? system fails safe	Select switch for high reliability and low probability of dangerous failure Rigid quality control on switch procurement
2		Short-circuit contacts	(a) faulty component (b) excessive current	System incorrectly senses guard to be closed	Allows machine to be used when guard is absent ? dangerous failure	Modify software to detect switch failure and take appropriate action
3		Excessive switch-bounce	(a) ageing effects (b) prolonged high currents	Slight delay in sensing state of guard	Negligible	Ensure hardware design prevents excessive current through switch

Tabelle 2.3.: Beispiel eines FMEAs aus [26, S. 38].

FTA

FTA steht für Fault Tree Analysis, im deutschen Sprachraum auch als Fehlerbaumanalyse bezeichnet wird. Dies ist eine Methodik zur Ermittlung der möglichen Faults die für den Failure eines Systems verantwortlich sind. Die Analyse beginnt mit dem Ausfall einer Komponente oder des Systems, dem sogenannten Top Level Event (TLE). Von daraus wird versucht, ein Baum bis zu den auslösenden Events aufzuspannen, den sogenannten Basic Events (BE). Des Weiteren existieren noch Intermediate Events (IE). Diese können aus mehreren BE und/oder IE bestehen. Ein IE wird immer über ein logisches Gater „AND“ und „OR“ erzeugt. Ein Beispiel eines FTs kann der Abbildung 2.5 entnommen werden. Problematisch ist die mögliche Größe eines solchen Baumes, besonders bei großen Systemen. Als Randbedingung sollten die verwendeten Komponenten und der Systemzustand, für eine solche Analyse, abgegrenzt und festgelegt werden. Daher ist der Einsatz eines FTA meist erst im späteren Projektverlauf

sinnvoll, da eine entsprechende Systemstruktur sowie detaillierte Informationen zu dieser, gegeben sein sollte. [26, S. 43ff], [4, S. 54ff]

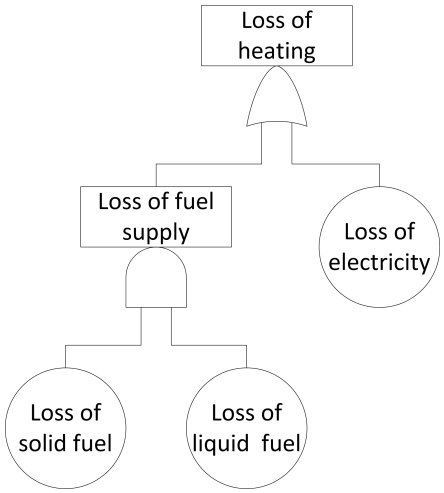

Abbildung 2.5.: Beispiel eines Fault Trees aus [26, S. 65].

ETA

ETA steht für Event Tree Analysis, im Deutschen auch bezeichnet als Ergebnisbaumanalyse. Ausgehend von einem Ereignis, das Initialereignis, werden alle möglichen Auswirkungen auf ein System oder eine Komponente untersucht. Der Baum wird von links nach rechts aufgebaut. Dabei besitzt jeder Knoten zwei Äste. Der obere Zweig beschreibt das erfolgreiche Verhalten und der untere das Scheitern. Ein Beispiel eines ETs kann der Abbildung 2.6 entnommen werden. Ein Problem dieser Analyse ist ebenfalls die mögliche Größe des Baumes und die sich daraus ergebende schlechte Übersichtlichkeit und Lesbarkeit. Für diese Methodik müssen die Auslöser bekannt sein. Daher wird die Analyse meist in späteren Design Phasen verwendet, nachdem beispielsweise die Systemanforderungen definiert wurden. [26, S. 35f], [4, S. 64f]

Weitere Methoden

Die bisher vorgestellten Analyse-Methoden entsprechen den Standardverfahren im Bereich der Sicherheit. Inzwischen gibt es aber auch andere Verfahren, die meist auf den grundlegenden Analysen aufsetzen, diese aber um spezielle Features erweitern. Immer wichtiger wird auch das Zusammenspiel von Safety und Security, da es möglich ist, dass durch eine Lücke in der

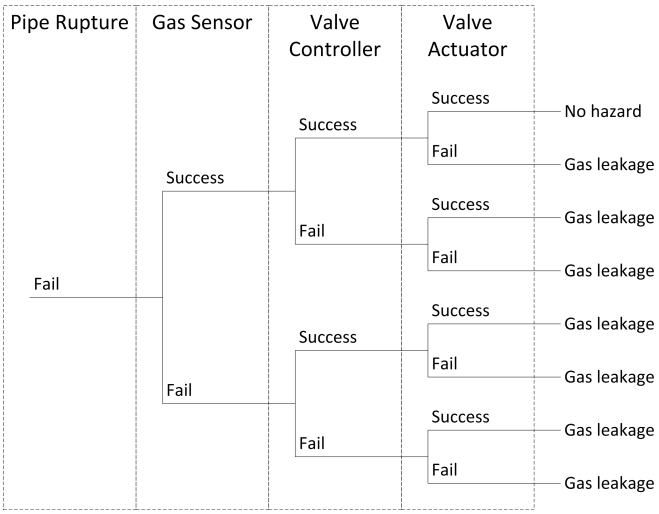

Abbildung 2.6.: Beispiel eines Event Trees aus [4, S. 65].

Security eines Systems, Probleme mit der Safety eines Systems einhergehen. Vorgehensweisen zur Security Analyse eines Systems sind Attack Trees (AT) oder Fault Trees for Security (FTFS). Auf diese Analyse-Methoden soll, in dieser Arbeit, nicht weiter eingegangen werden. Ein Vorgehen, das Security Probleme sucht, die die Safety eines Systems beeinflussen, ist in [25] beschrieben. Eine weitere Arbeit beschäftigt sich mit sicherheitskritischen Cyber Physical Systems (CPS) unter dem Aspekt der Safety und Security [22].

2.4.2. Risikoanalyse

Im folgenden Abschnitt wird die Risikoanalyse beschrieben. Die theoretischen Grundlagen basieren auf [26, S. 59ff] und der Norm DIN EN 61508-5 [14].

Eine Risikoanalyse besteht meist aus verschiedenen Phasen. Da die möglichen Gefahren bereits ermittelt wurden, sollte nun das Risiko bewertet und quantifiziert werden. Dazu wird zunächst das Vorgehen zur Risikoanalyse ganz allgemein beschrieben. Dazu sollten zunächst die Begriffe des Unfalls und Vorfalls definiert werden.

> „An **accident** is an unintended event or sequence of events that causes death, injury environmental or material damage.“ [26, S. 59]
>
> „An **incident** (or near miss) is an unintended event or sequence of events that does not result in loss, but, under different circumstances, has the potential to do so.“ [26, S. 60]

Anhand dieser Begriffe können Gefahren und die daraus entstehenden Risiken klar abgegrenzt werden. Im Grunde gilt Folgendes: Zu jedem Unfall existiert mindestens ein Vorfall. Im nächsten Schritt sollten die Risiken klassifiziert werden. Zur Klassifizierung wird folgende Formel verwendet:

$$\text{„}Risk = Serverity * Frequency\text{“ [26, S. 60]}$$

Entsprechend muss die Schwere und Häufigkeit definiert sein. Die Definition ist meist abhängig vom Anwendungsgebiet sowie den Anforderungen. Zum Beispiel existieren Komponenten mit niedrigen, hohen oder kontinuierlichen Anforderungen. Die DIN EN 61508 unterscheidet die Häufigkeiten folgendermaßen in absteigender Reihenfolge:

- Häufig
- Wahrscheinlich
- Gelegentlich
- Gering
- Unwahrscheinlich
- Nicht glaubhaft [14, S. 28]

Dabei legt die Norm nicht fest, was sich konkret hinter diesen Begriffen verbirgt. Das muss, je nach Anwendung und Umfeld, gesondert definiert werden. Zur Schwere legt die Norm ebenfalls die Grundbegriffe fest:

- Katastrophal
- Kritisch
- Begrenzt
- Geringfügig [14, S. 28]

Auch diese sind von der Norm nicht definiert und müssen der Anwendung und dem Umfeld entsprechend beschrieben werden. Durch die Verwendung einer Matrix (Tabelle 2.4) ist es nun möglich, das Risiko einer Klasse zu zuordnen.

Häufigkeit	**Auswirkung**			
	katastrophal	kritisch	begrenzt	geringfügig
häufig	I	I	I	II
wahrscheinlich	I	I	II	III
gelegentlich	I	II	III	III
gering	II	III	III	IV
unwahrscheinlich	III	III	IV	IV
nicht glaubhaft	IV	IV	IV	IV

Tabelle 2.4.: Matrix zur Ermittlung der Risikoklasse nach DIN EN 61508 [14, S. 28].

Aus der Tabelle 2.4 kann die Klasse nun einfach abgelesen werden. Der folgenden Tabelle 2.5 können die Definitionen der Risikoklassen entnommen werden.

Risikoklassen	**Interpretation**
Klasse I	nicht tolerierbares Risiko
Klasse II	unerwünschtes Risiko, das nur tolerierbar ist, wenn ein Risikominderung nicht durchführbar ist oder die Kosten der Minderung unverhältnismäßig hoch im Vergleich zur erzielten Verbesserung sind
Klasse III	tolerierbares Risiko, wenn die Kosten einer Risikominderung die erreichbare Verbesserung übersteigen
Klasse IV	vernachlässigbares Risiko

Tabelle 2.5.: Tabellarische Auflistung der Risikoklassen und der jeweiligen Beschreibung nach DIN EN 61508 [14, S. 28].

Zu beachten ist bei dem hier beschriebenen Vorgehen, dass sich die DIN EN 61508 nicht auf bestimmte Werte festlegt. Da es sich, wie in Kapitel 2 beschrieben, um eine sehr generische Norm handelt, müssen die verschiedenen Werte, je nach Anwendung und Umfeld, konkretisiert werden. Andere Normen, die bereits konkret für ein Fachgebiet definiert wurden, enthalten zum Teil passende oder angenäherte Werte.

Nach der Klassifizierung der Risiken kann die Akzeptanz dieser betrachtet werden. Dabei sollte grundlegend geprüft werden, ob ein Risiko tolerierbar ist. Ist dies nicht der Fall, muss geprüft werden, wie es möglicherweise reduziert werden kann. Ziel ist es, ein akzeptables Restrisiko zu erreichen. Dementsprechend müssen die Maßnahmen zur Risikoreduzierung ebenfalls geprüft werden. Ein Restrisiko ist in jedem System vorhanden. Zur Ermittlung der Akzeptanz existieren verschiedene Prinzipien. Ein Beispiel ist ALARP (as low as is reasonably). Übersetzt man ALARP ins Deutsche, bedeute es: „so niedrig, wie vernünftigerweise praktikabel“. Allgemein gesagt sollen die Risiken auf ein Maß reduziert werden, in dem die Sicherheit am höchsten ist [26, S. 68].

Der letzte Abschnitt widmet sich der Integritätsbestimmung. Dazu werden in den verschiedenen Sicherheitsnormen sogenannte Sicherheitsintegritätslevel verwendet. Die DIN EN 61508 kennt insgesamt vier verschiedene Level. Die Safety Integrity Level (SIL) werden vom höchsten SIL-4 bis zum niedrigsten SIL-1 aufgeteilt. Im Regelfall enthalten alle Normen eine solche oder vergleichbare Ausführung. In der Tabelle 2.6 sind die SIL Level der DIN EN 61508 für Komponenten und Systeme mit niedrigen Anforderungsraten. In der Tabelle 2.7 sind die Level für alle Systeme und Komponenten für hohe oder kontinuierliche Anforderungsraten aufgelistet. In [27] zeigt die Tabelle 2 einen Vergleich der domänenspezifischen Integritätslevel. Im Kern beschäftigt sich dies mit der Erweiterung der bekannten Level.

Sicherheits-Integritätslevel (SIL)	**Mittlere Wahrscheinlichkeit eines gefahrbringenden Ausfalls bei Anforderung der Sicherheitsfunktion (PFD_{avg})**
4	$\geq 10^{-5} bis < 10^{-4}$
3	$\geq 10^{-4} bis < 10^{-3}$
2	$\geq 10^{-3} bis < 10^{-2}$
1	$\geq 10^{-2} bis < 10^{-1}$

Tabelle 2.6.: SI-Level für Systeme mit niedrigen Anforderungsraten nach DIN EN 61508 [13, S. 36].

Sicherheits-Intigritätslevel (SIL)	**Mittlere Wahrscheinlichkeit eines gefahrbringenden Ausfalls der Sicherheitsfunktion (PFH)**
4	$\geq 10^{-9} bis < 10^{-8}$
3	$\geq 10^{-8} bis < 10^{-7}$
2	$\geq 10^{-7} bis < 10^{-6}$
1	$\geq 10^{-6} bis < 10^{-5}$

Tabelle 2.7.: SI-Level für Systeme mit hohen oder kontinuierlichen Anforderungsraten nach DIN EN 61508
[13, S. 37].

Die Tabellen 2.6 und 2.7 enthalten die PFD (probability of dangerous failure on demand, dt. Wahrscheinlichkeit des Versagens bei Anforderung) oder PFH (probability of dangerous failure per hour, dt. Wahrscheinlichkeit des Versagens pro Stunde). Diese Werte beschreiben die mittlere Wahrscheinlichkeit eines Ausfalls, der gefahrbringend eingestuft ist. Dabei müssen diese Werte für jede Komponente definiert werden. Ein ganzes System besitzt somit das niedrigste SIL entsprechend des niedrigsten SIL einer seiner Komponenten. Das SIL wird meist zur Beurteilung der notwendigen Gesamtsicherheit eines Systems betrachtet. Außerdem kann festgelegt werden, welches SIL ein System in einem konkreten Umfeld mindestens benötigt und somit welche Stufe die Komponenten mindestens benötigen. Außerdem können die Level als Anforderungen an das System festgehalten werden, sodass ein entsprechender Entwurf, inklusive der Maßnahmen zum Einhalten, zustand kommt.

2.4.3. Zusammenfassung

Durch dieses Kapitel sollte ein kurzer Einblick in die Methodiken der Gefahren- und Risikoanalysen gegeben werden. Bei den Gefahrenanalysen ist es in der Praxis üblich, dass eine Kombination aus den verschiedenen Analyse Methoden verwendet wird. Die Analysen besitzen verschiedene Stärken und Schwächen und sollten in unterschiedliche Projektphasen eingesetzt werden. Bei den Risikoanalysen ist es zumeist problematisch, die Wertigkeiten richtig zu bestimmen. Zum Teil ist dies sogar unmöglich. Gerade diese Anforderung an die Analyse ist recht problematisch für ihren Einsatz.

3. Anforderungsanalyse

In diesem Kapitel werden die Systemzustände des Gesamtsystems identifiziert und definiert. Dazu werden zunächst die Ziele der Analyse besprochen. Im Anschluss werden die ermittelten Zustände spezifiziert. Außerdem wird der Mindestumfang an aktiven Komponenten, je Zustand, angegeben. Die Ermittlung ist für die anschließende Systemanalyse (Kapitel 3) sowie das spätere Safety-Konzept (Kapitel 5) wichtig. Es handelt sich hierbei also grundsätzlich um die gestellten Anforderungen an das System als Grundlage für die nachfolgenden Kapitel.

3.1. Problemstellung und Ziele

Zum einen soll durch die nachfolgende Beschreibung eine klare Spezifikation des Systems festgelegt werden. Auf diese Spezifikation bauen entsprechend die nachfolgenden Kapitel auf. Diese Basisaussagen werden für die Arbeit benötigt, da sie bisher noch nicht eindeutig festgeschrieben wurden. Dadurch soll es zum einen ermöglicht werden, qualitativ gute Aussagen zur Systemsicherheit zu machen. Zum anderen kann anhand dieser und den benötigten Komponenten, je Zustand, die Gefahren und Risiken einfacher analysiert werden. Außerdem kann dadurch eine erste Abgrenzung vorgenommen werden, da das Safety-Konzept nicht für alle Systemzustände gelten soll. Des Weiteren ist es hilfreich, da ein Zustandsautomat eine abstrakte Sicht auf das System bietet. Dadurch ist das System meist einfacher zu verstehen.

3.2. Spezifikation

Die Abbildung 3.1 zeigt den Zustandsautomaten des Gesamtsystems. Es sind nur Zustände auf erster Ebene definiert worden. Innerhalb dieser Zustände können auch weitere Subautomaten hierarchisch definiert werden. Um die grundlegenden Risiken des Systems darzustellen, reicht eine solche abstrakte Sicht. Die hier beschriebenen Systemzustände entsprechen den Grundzuständen manuell, stabilisierend und autonom, die ebenfalls bei dem bereits vorgestellten System vorhanden sind (Kapitel 2). Der Zustandsraum wurde noch erweitert, um die Zustände am Boden, vergleichbar mit den Flugphasen eines Flugzeugs (Vorfeld, Rollen) und es wurden außerdem entsprechende Zustände für die Sicherheit eingeführt. Die Zustände

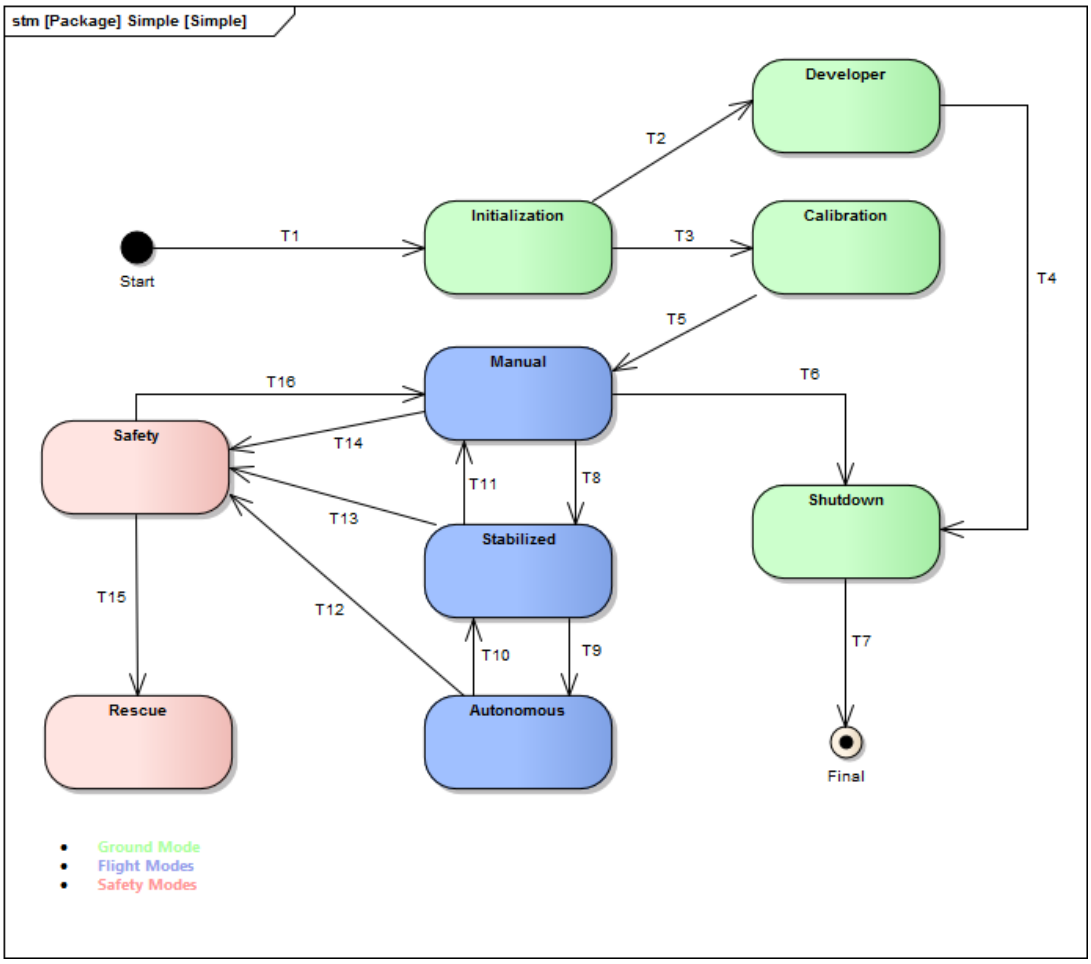

Abbildung 3.1.: Spezifikation der aktuellen logischen Systemzustände: Ground-, Flight- und Safety-Modes.

wurden in drei Kategorien unterteilt: Ground Modes, Flight Modes und Safety Modes. Auf die entsprechenden Zustände und dessen Transitionen wird im weiteren Verlauf detailliert eingegangen. Die Transitionen wurden in der Grafik durchnummeriert, als T1 - Tn. Es wurde keine Guards an den Kanten eingetragen. Der Grund dafür ist die eher unübersichtliche Darstellung, da es viele Abhängigkeiten gibt. In diesem Kapitel werden die Bedingungen und Transitionen „wörtlich“ beschrieben und ein entsprechender Hinweis, um welche Transition es geht, wurde hinzugefügt.

Einen Vorschlag zu den Systemzuständen wurde bereits von Arne Richter [19, S. 26ff] vorgestellt. Dabei wurden in diesem lediglich die verschiedenen Flug Modi beschrieben. Die dortigen Modi sind außerdem nach neusten Erkenntnissen nicht alle, wie dort definiert, umsetzbar. Der Stabilisierte Flugmodus und der Autonome Flugmodus sind vergleichbar mit den Zuständen „Stabilized“ und „Autonomous“. Der begrenzte Steuerungsmodus kann grundsätzlich mit dem Zustand „Manual“ verglichen werden. Der vorgeschlagene Modi Safemode-Reines Messsystem wiederum kann gar nicht umgesetzt werden, da es bei vielen Trägern nicht möglich ist, die Steuersignale eins zu eins auf die Aktorik zu übertragen. Daher wurden im Folgenden entsprechende Anpassungen für die Systemsicherheit vorgenommen. Des Weiteren wurde das Auslösen des RS bisher nicht im Detail betrachtet. Die Beschreibung von Arne Richter bezüglich der Modi war sehr allgemein und kann daher übertragen werden. In dieser Spezifikation wird zusätzlich noch angegeben, welche Komponenten benötigt werden. Somit kann abschließend festgehalten werden, dass die Modi Stabilisierte Flugmodus, Autonomer Flugmodus und der begrenzte Steuerungsmodus grundsätzlich auf den bereits von Arne Richter beschriebenen Bedingungen aufbaut und diese um die genutzten Komponenten erweitert. Zusätzlich wurden entsprechende Safety Mode hinzugefügt, die speziell für das Safety-Konzept interessant sind. Die Ground Modes wurden nur der Vollständigkeit halber eingefügt.

Komponenten

Der Systemaufbau auf Komponentenebene ist in Kapitel 1 beschrieben. In den beschriebenen Zuständen und besonders für die späteren Analysen (siehe Kapitel 4) sind die Komponenten entscheidend. Jeder Zustand benötigt verschiedene Komponenten, um einen reibungslosen Betrieb zu gewährleisten. Für jeden Zustand wird immer eine funktionierende Spannungsversorgung vorausgesetzt. Eine Ausnahme ist der Zustand „Rescue“ der Safety Modes. Das bedeutet, das das Power Board keine extra Erwähnung in der folgenden Spezifikation hat, sondern immer aktiv sein muss.
Die Komponente, beziehungsweise genauer der Sensor, AHRS ermöglicht als alleinstehend einen autonomen Flug durchzuführen. Es werden nicht zwingend weitere Sensoren benötigt. Der alleinige Einsatz sollte nur im Notfall notwendig sein, da es merkbare Abweichungen und Ungenauigkeiten gibt.

3.2.1. Ground Modes

An die Ground Modes gibt es keine speziellen Anforderungen. Der Vollständigkeit halber werden diese in diesem Abschnitt kurz angeschnitten. Grundsätzlich kann gesagt werden, dass es sich bei diesen Modes um die Initialisierung der Komponenten handelt. Da sie im weiteren Verlauf nicht benötigt werden, handelt es sich bei diesen Modes um einen Vorschlag, der noch konkretisiert werden muss. Eine Grundbedingung für diese ist, dass sich der Träger am Boden befindet und nicht in Bewegung ist. Das Rollen gehört zu dem „Manual“ Zustand und wird entsprechend den Flight Modes zugeordnet.

Nach dem Einschalten des Systems befindet sich dieses im Zustand „Initialization“ (über T1). In dem Zustand sollte zunächst ein Power On Self Test (POST) durchgeführt werden. Wenn dieser Vorgang abgeschlossen ist, wechselt der Automat in den Zustand „Calibration“, unter Bedingung, dass alle Komponenten des Systems betriebsbereit sind (über T3). Der „Calibration“ Zustand dient der Kalibrierung der Sensorik und Aktorik. Wie eine solche Kalibrierung im Detail aus zu sehen hat, soll ebenfalls nicht hier erläutert werden. Nach der Kalibrierung geht der Automat über in die Flight Modes. Ein alternativer Pfad wäre die Transition T2. Dies schaltet nur, wenn ein Entwicklerkabel mit dem System verbunden ist. Dann wechselt das System in den Zustand „Developer“. Zu beachten ist, dass für einen solchen Wechsel keine Voraussetzungen gegeben sind. Der „Developer“ Zustand ist hier nicht von zentraler Bedeutung, aber bereits vorgesehen. In diesem Modus soll ein erweiterter Zugriff zur Verfügung stehen, um zum Beispiel das System zu debuggen oder neu zu flashen.

Der letzte Zustand der zu den Ground Modes gehört ist der „Shutdown“. Dieser ist aus dem Flight Mode „Manual“ (über T6) sowie dem Ground Mode „Developer“ (über T4) erreichbar. In diesem Zustand werden die Routinen zum Abschalten des Systems ausgeführt, bevor das System ordnungsgemäß herunterfährt. Was in diesem Zusammenhang ordnungsgemäß bedeuten soll, wird hier nicht besprochen werden, da es für den weiteren Verlauf nicht notwendig ist.

Die Ground Modes haben in dieser Arbeit keine zentrale Bedeutung. Sie wurden der Vollständigkeit halber eingefügt. Im weiteren Verlauf des PO AES müssen dies noch detailliert spezifiziert werden.

3.2.2. Flight Modes

Die Flight Modes bestehen aus genau drei Zuständen. Der Zustand „Manuel“ richtet sich dabei an den Begrenzten Steuerungsmode aus [19, S. 28]. Im Unterschied zu der dortigen Beschreibung ist es nicht zwingend notwendig, dass die Steuerwerte manipuliert werden, da diese vom Träger abhängig sind. Der Zustand „Stabilized“ basiert grundsätzlich auf dem Stabilisierten

Flugmodus aus [19, S. 28f]. Im Detail bedeutet dies, dass der Flug des Piloten mit der Hilfe von Dämpfern stabilisiert wird. Der Zustand „Autonomous“ basiert grundsätzlich auf dem Autonomen Flugmodus aus [19, S. 29f]. Das bedeutet, dass in diesem ein autonomer Flug durchgeführt wird, der alleine unter der Kontrolle der FCU erfolgt. Außerdem sollen entsprechende Soll-Vorgaben über eine Funkverbindung übertragen werden können. Somit können die entsprechenden Anforderungen von Arne Richter übernommen werden, außer ein paar kleine Anpassungen.

In diesen Zuständen befindet sich der Träger entweder am Boden, im Folgenden als Rollen bezeichnet, oder in der Luft. Am Boden ist der Träger bereits vollständig initialisiert und bereit für den Flug. Der Name Flight Modes wirkt vielleicht etwas unpassend, da aber die explizite Unterscheidung zwischen Flug und Rollen recht umständlich ist, wurden die beiden Abschnitte zusammengefasst.

Im „Manual“ Zustand befindet sich der Träger, wenn er am Rollen ist, während des Start- und Landevorgangs, sowie im manuell gesteuerten Flug. Anders gesagt soll in diesem Zustand ein externer Pilot die Kontrolle über den Träger haben. Die Komponenten, die in diesem Zustand zwingend benötigt werden sind: RCR, CSA, Aktorik, P2S, S2P. Unter dieser Bedingung kann auch T5 schalten. Im Falle, dass einer dieser Komponenten nicht mehr zur Verfügung steht oder ein anderer Failure präsent ist, kann das System in den Zustand „Safety“ der Safety Modes wechseln (über T14), welche im weiteren Verlauf besprochen werden. Eine Transition zum Zustand „Stabilized“ ist möglich, wenn die Komponenten RCR, CSA, Aktorik, P2S, S2P sowie zusätzlich noch der AHRS und die FCU zur Verfügung stehen. Dies sind ebenfalls alle Komponenten, die für den ordnungsgemäßen Betrieb des Systems in diesem Zustand benötigt werden. Unter dieser Bedingung kann auch T8 schalten. Fällt einer der Komponenten aus, oder ist ein Failure präsent, kann das System in den Zustand „Safety“ wechseln (über T13). Ansonsten besteht die Möglichkeit, dass das System zurück zum „Manual“ Zustand wechselt . Dabei gelten für T11 die gleichen Bedingungen wie für T5. Ebenfalls ist ein Wechsel zum Zustand „Autonomous“ möglich. Für einen sicheren Betrieb in diesem wird der CSA, die Aktorik, der S2P, der AHRS und die FCU benötigt. Somit ist es nur möglich T9 zu schalten, wenn die entsprechenden Komponenten fehlerfrei sind. Wie in den bereits spezifizierten Zuständen kann bei einem Failure, das System in den Zustand „Safety“ wechseln (über T12). Alternativ dazu kann zurück in den „Stabilized“ Zustand gewechselt werden. Dabei gelten dieselben Bedingungen für T10 wie die bereits beschriebenen für T8.

Die als Flight Modes spezifizierten Zustände sind vergleichbar mit den bereits in Kapitel 2.3 vorgestellten Modi. Die interessante Erweiterung sind die Safety Modes, die als Nächstes besprochen werden. Ein Ziel der Zustandsraumerweiterung ist es, die Systemsicherheit zu erhöhen. Durch die Safety Modes soll es ermöglicht werden, bei einem Failure, den Systemzustand neu zu beurteilen und entsprechende Maßnahmen einzuleiten. Der Wechsel von Zustand „Manual“ zum Zustand „Autonomous“, sowie anderes herum, geht über den Zustand „Stabilized“. Dadurch sollen sogenannte Ausreißer verhindert werden. Fliegt der Träger beispielsweise eine

steile Rechtskurve und der Pilot hat seinen „Steuerstick“ in der Position einer steilen Linkskurve stehen, soll dadurch ein möglichst stabiler Wechsel erfolgen. Außerdem muss man sich in Zukunft noch Gedanken zur Zustandsanzeige machen, zum Beispiel akustisch oder optisch, damit der Sicherungspilot vom Wechsel nicht überrascht wird.

3.2.3. Safety Modes

Die separaten Safety Modes werden als Erweiterung zu den bereits bekannten Modes vorgeschlagen. Dabei kann hier auf keinen Vorschlag aufgesetzt werden. Der Zustand „Safety“ ist bei einem Ausfall die erste Anlaufstelle. Der Zustand soll prüfen, ob ein Weiterflug möglich ist. Dies ist auch die Anforderung an den Zustand. Falls ein Weiterflug nicht möglich ist, muss versucht werden auf einem anderen Weg einen Safe-State zu erreichen. Ein Safe-State soll mithilfe des Zustandes „Rescue“ ermöglicht werden. Der Zustand ist die letzte mögliche Instanz und soll ein Auslösen des RS ermöglichen, unter möglichst allen Bedingungen. Dies ist auch die Anforderung an diesen Zustand.

Die beiden Zustände „Safety“ und „Rescue“ der Safety Modes können nur aus den Flight Modes erreicht werden (über T12, T13 und T14). Dies beruht auf der Annahme, dass der Träger in einem Safe-State ist, solange er sich am Boden befindet. Der zentrale Zustand der Safety Modes ist „Safety“. An die Transitionen T12, T13 und T14 sind keine weiteren Bedingungen geknüpft. Wenn das System in dem Zustand „Safety“ angelangt ist, prüft der Zustand, ob folgende Komponenten funktionsfähig sind: RCR, CSA, Aktorik, P2S, S2P. Ist dies der Fall, besteht die Möglichkeit den Träger manuell sicher zu landen und ein Safe-Live ist gewährleistet. Ein Wechsel über T16 in den Zustand „Manual“ ist somit möglich. Für die Transition gelten die eben beschriebenen Bedingungen. Ist ein solcher Wechsel nicht möglich, geht das System in den Zustand „Rescue“ (über T15). In diesem wird das Rescue System ausgelöst und die Triebwerke abgeschaltet. Der Träger befindet sich nun in einem Safe-State. Diesen Zustand kann das System nicht aus eigener Kraft verlassen, da unklar ist, ob zum Beispiel die Struktur des Trägers, eine Systemkomponente oder etwas anderes beschädigt ist. Der gesamte Träger und alle Bestandteile sollten nach einem solchen Ausfall zunächst ausgiebig geprüft werden und es sollten die Log-Dateien und des SH Monitoring analysiert werden.

Der Zustandsraum der Safety Modes ist sehr kompakt und einfach gehalten. Dieser Raum ist extra überschaubar und mit einer geringen Anzahl von Links entworfen worden. Komplexe Systeme und Regeln sind zu meist fehlerträchtiger, da es häufig schwieriger ist, diese in den Griff zu bekommen.

4. Gefahren- und Risikoanalyse

In diesem Kapitel sollen die Gefahren und Risiken des Systems und seiner Komponenten ermittelt, analysiert und kategorisiert werden. Anhand dieser Daten können die entsprechenden Gegenmaßnahmen beschrieben werden. Zunächst werden die Ziele definiert, um das Vorgehen zu begründen. Im Anschluss daran wird mithilfe der FTA die Gefahrenanalyse durchgeführt. Dies geschieht auf Basis der der Systemzustände aus Kapitel 3. Im letzten Abschnitt werden die Ergebnisse aus der vorhergegangenen FT-Analyse zur Ermittlung der Risiken genutzt. Die ermittelten Daten ermöglichen es, das System besser beurteilen zu können. Diese Ergebnisse zeigen die Sicherheitsprobleme des Systems und anhand dieser können passende Lösungsentwürfe erarbeitet werden. Außerdem wird aufgezeigt, welche Komponenten besonders gefährdet sind.

4.1. Ziele

Grundlegend sollen in diesem Kapitel die Gefahren sowie Risiken ermittelt und eingeordnet werden. In einem sicherheitskritischen System sind dies Vorgehensweisen, die zwingend notwendig sind. Um ein mögliches Konzept zur Erhöhung der Sicherheit vorzuschlagen, müssen die hier diskutierten Gefahren und Risiken bekannt sein. Außerdem muss sich das Safety-Konzept später abgrenzen können, da es unmöglich ist, die „Eierlegende Wollmilchsau“ zu entwerfen. Es ist nur möglich, einen Teil der kritischen Pfade entgegen zu wirken. Alle anderen Pfade wurden aber analytisch erfasst und können als Anforderungen an andere Komponenten und Teilsysteme weiter gegeben werden.

4.2. Gefahrenanalyse

Momentan sind die Bedingungen, die ein Systemausfall zur Folge haben, noch vollkommen unbekannt. Es sollte daher zunächst ermittelt werden, welche Faults einen Ausfall einer Komponente und letztlich den Systemausfall zur Folge haben. Aus dieser Sicht heraus bietet sich die FTA an und wird für diese Arbeit verwendet. Die Grundlagen für die Analyse kann dem Kapitel 2 entnommen werden. Außerdem muss man sich durch die Verwendung der FTA zunächst

nur auf die für den Ausfall relevanten Systemteile konzentrieren. Des Weiteren wurden bereits entsprechende Systemzustände, sowie alle nötigen Abgrenzungen und Anforderungen (Kapitel 3) definiert. Das gewählte Verfahren ist simpel in der Anwendung und für das Verständnis Dritter einfacher. Bei der Größe des gegebenen Systems ist die Analyse noch übersichtlich. Eingegrenzt ist die Analyse auf die drei Grundzustände der Flight Modes, dem Zustand „Manual, „Stabilized“ und „Autonomous“. Es gibt verschiedene Arten der Fault Trees (FT). Einen Überblick gibt [24]. Im Folgenden wird nach der im vorherigen Abschnitt beschriebenen Fehlerbaumanalyse vorgegangen. Zusätzlich zur Erstellung der FTs werden sogenannte Minimal Cut Sets (MCS, auf Deutsch Minimalschnitt) zu jedem FT angefertigt. Die MCSs entsprechen einer Menge von BEs, die notwendig sind, um das TLE zu erreichen. Ein MCS wird durch das Aufstellen der booleschen Gleichung zum FT ermittelt. Ein Beispiel zu den MCS kann in [1] nachgelesen werden.

4.2.1. Fault Tree Analyse

Die Analyse besteht insgesamt aus drei Teilen, für den Zustand „Manual“, den Zustand „Stabilized“ und den Zustand „Autonomous“. Es wurde jeweils ein FT und der dazugehörige MCS angefertigt. Der erste FT zum Zustand „Manual kann der Abbildung 4.1 entnommen werden. Die Ermittlung des MCS kann der Tabelle 4.1 entnommen werden.

Zu erkennen ist auf den ersten Blick, die Kette, die durch einen Fault oder Failure in Gang gesetzt wird und zum Ausfall des Gesamtsystems führen kann. Löst beispielsweise BE7 aus, kann der ordnungsgemäße Ablauf im Zustand „Manual“ nicht mehr gewährleistet werden. Die hier angegebenen „System faults“ stellen einen Software oder Hardware Fault dar. Das BE8 stellt einen Fault des RCR dar, darunter fällt ebenfalls der Fault „Kein Signal“. Das bedeutet, dass der RCR keinen Kontakt zum Piloten hat. Diese Komponente wurde als BE definiert, da sie keiner Eigenentwicklung unterliegt, sondern eingekauft wurde. Dass Gleiche gilt für die Aktorik (BE4). Das IE1 entspricht dem Failure des PBs. Das IE1 ist als Ausfall in allen weiteren FTs vorhanden (Abbildung 4.2 und 4.3). Das PB fällt aus, wenn einer der beiden Akkus nicht mehr funktioniert oder keine Versorgungsspannung mehr liefern kann. Eine zweite Möglichkeit wäre es, dass das Board selber einen Fault, hat.

Anhand der MCS für den Zustand „Manual“ kann erkannt werden, dass nur eine Komponente ausfallen muss, damit es zum Failure des Zustandes kommt. Es müssen diesbezüglich definitiv Maßnahmen eingeleitet werden.

Der zweite FT zeigt die Abhängigkeiten im Zustand „Stabilized“. Der FT kann der Abbildung 4.2 entnommen und der ermittelte MCS der Tabelle 4.2.

Die IE2, IE3 und IE5 entsprechen grundsätzlich den IEs aus dem FT des Zustandes „Manual“. An dem IE3 wurde zusätzlich der IE4 in den Baum intrigiert. Dieser Zweig stellt den Ausfall der

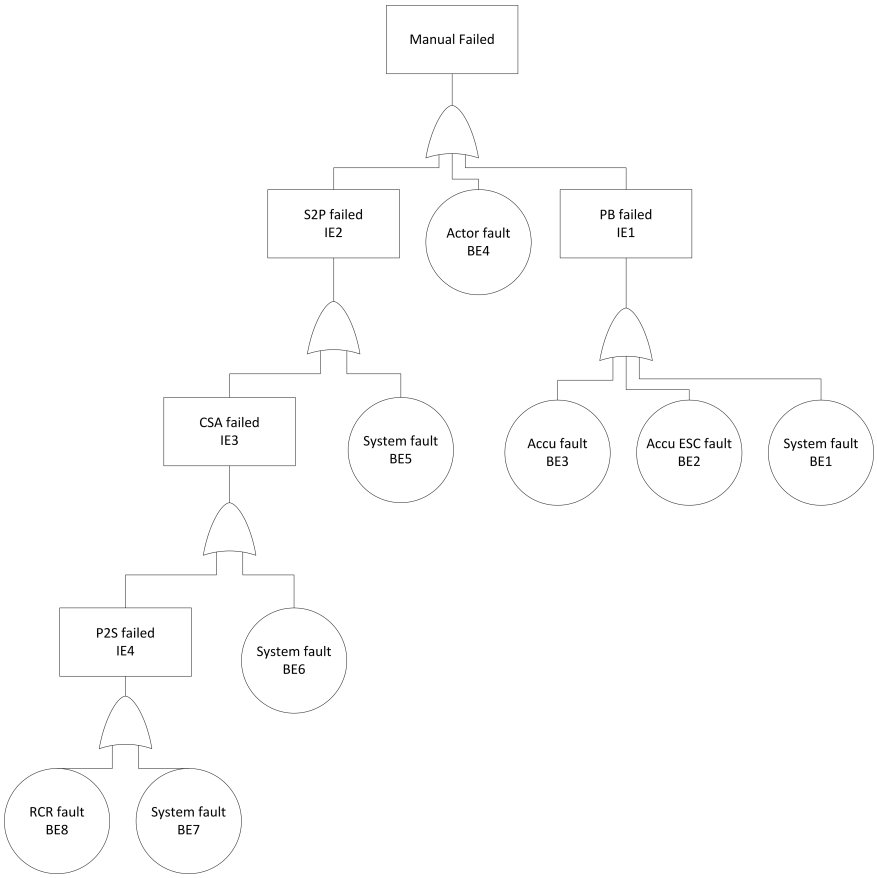

Abbildung 4.1.: Fault Tree des Zustandes „Manual“.

Schritt	**Boolescher Ausdruck**
1	$TLE = IE1 \vee BE4 \vee IE2$
2	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee (IE3 \vee BE5)$
3	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee (IE4 \vee BE6)$
4	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee BE6 \vee (BE8 \vee BE7)$
5	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee BE6 \vee BE8 \vee BE7$

Tabelle 4.1.: Minimal Cut Set des Zustandes „Manual“.

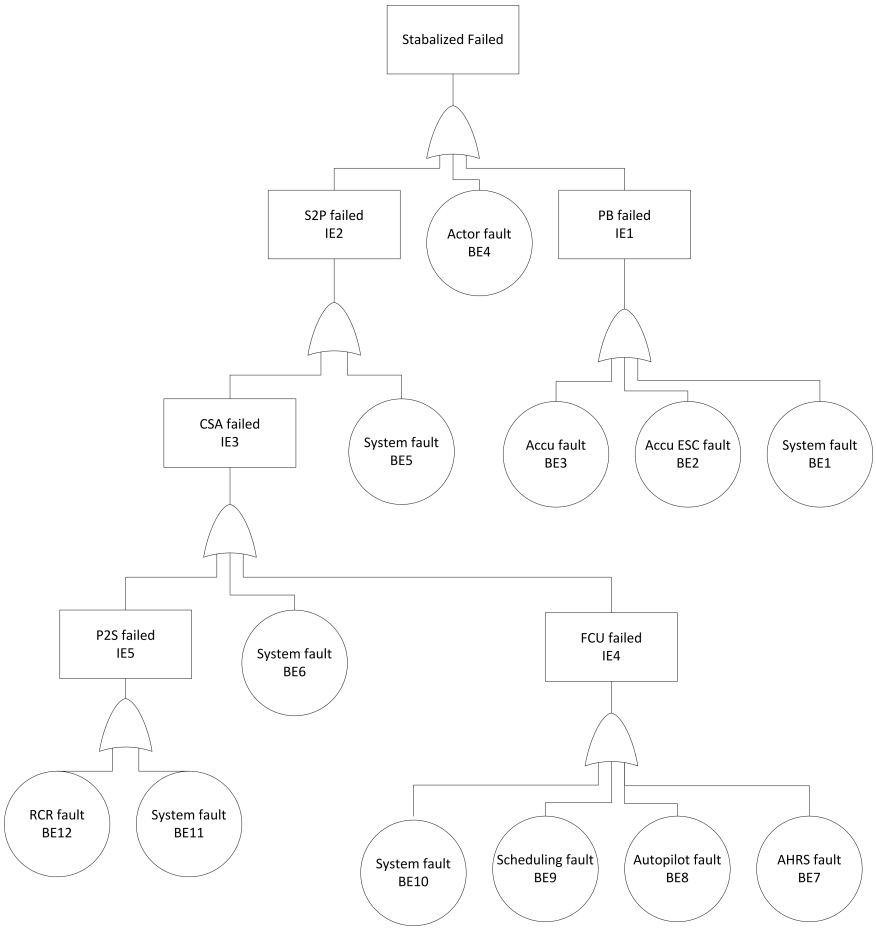

Abbildung 4.2.: Fault Tree des Zustandes „Stabilized“.

FCU dar. Die FCU kann ebenfalls durch einen Hardware oder Software Fault ausfallen (BE10). Weitere Möglichkeiten, die sich speziell auf systematische Faults beziehen, ist der Ausfall des Autopilots (BE8) oder ein Problem beim Scheduling (BE9). Dabei handelt es sich um Software Faults, die zu einem Ausfall der Komponenten führen kann. Eine weitere spezielle Abhängigkeit stellt die AHRS (BE7) dar. Diese Komponente wurde ebenfalls eingekauft und unterliegt keiner Eigenentwicklung.

Schritt	Boolescher Ausdruck
1	$TLE = IE1 \vee BE4 \vee IE2$
2	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee (IE3 \vee BE5)$
3	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee (IE5 \vee BE6 \vee IE4)$
4	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee ((BE11 \vee BE12) \vee BE6 \vee (BE7 \vee BE8 \vee BE9 \vee BE10))$
5	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee BE11 \vee BE12 \vee BE6 \vee BE7 \vee BE8 \vee BE9 \vee BE10$

Tabelle 4.2.: Minimal Cut Set des Zustandes „Stabilized“.

Hier kann man die gleiche Problematik wie im Zustand „Manual“ erkennen. Fällt eine Komponente aus, ist ein Weiterflug innerhalb des Zustandes „Stabilized“ unmöglich. Eine andere Eigenschaft kann dem FT entnommen werden. Wenn nur die FCU (IE4) ausfällt, ist es immerhin möglich, im Zustand „Manual“ weiter zu fliegen. Umgekehrt gilt dasselbe. Im Falle das nur der P2S (IE5) ausfallen würde, könnte im Zustand „Autonmous“ weitergeflogen werden.

Der dritte und letzte FT zeigt die Abhängigkeiten im Zustand „Autonomous“. Der FT zu diesem Mode kann der Abbildung 4.3 entnommen werden und der dazugehörige MCS der Tabelle 4.3. Grundsätzlich ist der FT auch mit dem des Manual Mode vergleichbar. Es wurde lediglich der IE4 (P2S failed) des FT des Zustandes Zustand „Manual“ gegen den IE4 (FCU failed) aus dem FT des Zustandes „Stabilized“ ausgetauscht. Ansonsten stellt der FT des Zustandes „Autonomous“ eine Kopie des FT des Zustandes „Manual“ dar.

Alle Faults (BEs) und Failures (IEs), die in dem FT des Zustandes „Autonomous“ verwendet werden, wurden bereits in den analysierten FTs beschrieben. Es ändert sich grundsätzlich nur die Konstellation der Events. Der MCS zeigt die gleiche Problematik. Fällt eine der Komponenten oder ein Teilsystem aus, führt dies letztlich zum Ausfall des Zustandes „Autonomous“.

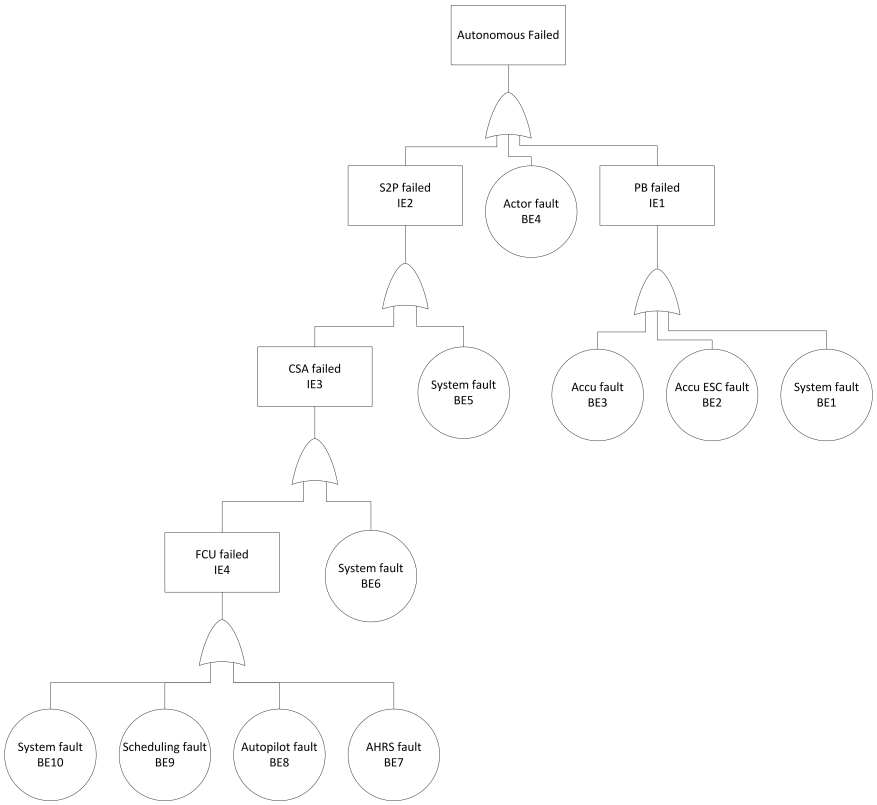

Abbildung 4.3.: Fault Tree des Zustandes „Autonomous“.

Schritt	Boolescher Ausdruck
1	$TLE = IE1 \vee BE4 \vee IE2$
2	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee (IE3 \vee BE5)$
3	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee (IE4 \vee BE6)$
4	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee BE6 \vee (BE7 \vee BE8 \vee BE9 \vee BE10)$
5	$TLE = BE1 \vee BE2 \vee BE3 \vee BE4 \vee BE5 \vee BE6 \vee BE7 \vee BE8 \vee BE9 \vee BE10$

Tabelle 4.3.: Minimal Cut Set des Zustandes „Manual“.

4.2.2. Zusammenfassung

Zusammenfassend kann gesagt werden, dass alle relevanten Komponenten direkte Auswirkungen auf die Zuverlässigkeit und Verfügbarkeit des Systems haben. Das Ergebnis zeigt, dass der Ausfall einer wichtigen Komponente den Systemausfall zur Folge hat. Das Erreichen eines Safe-States ist somit fast unmöglich. Es ist zum Teil möglich den Flug in einem alternativen Mode fortzuführen, um dadurch ein Safe-State zu erreichen. Aber der größere Teil der Komponenten stellt einen „Single Point of Failure“ dar. Anhand der Analyse kann nun erkannt werden, warum das System ein Safety-Konzept benötigt. Außerdem ist im System bereits ein Rescue System vorgesehen (siehe Kapitel 1). Zurzeit kann nicht in allen Situationen sichergestellt werden, dass das RS auch ausgelöst wird. Mit dem hier erlangten Wissen zum System und dem möglichen Verhalten im Fall eines Failures kann im nächsten Schritt das Risiko für die Gefahren und Komponenten bestimmt werden.

Die FT-Analyse war für den ersten Durchlauf die richtige Methode. Die grundlegenden Faults die zum System-Failure führen können, konnten durch die Analyse ermittelt werden und sind nun festgehalten. Anhand der erstellten FTs konnten nutzbare Alternativzustände aufgezeigt werden, in denen ein weiterer Betrieb möglich wäre. Die Problematiken des Systems sind durch die FTs übersichtlich dargestellt. Außerdem können die ermittelten Faults im Safety-Konzept verwendet werden, um entsprechende Gegenmaßnahmen zu beschreiben. Trotzdem sind noch weitere Analysen notwendig.

4.3. Risikoanalyse

Nachdem die Gefahren analysiert und bestimmt sind, können zu diesen und den beteiligten Komponenten die entsprechenden Risiken ermittelt werden. Das Vorgehen richtet sich nach den im Kapitel 2 vorgestellte Methode. Dabei wird keine komplette Analyse durchgeführt, sondern diese in ihrer Form und ihrem Umfang an die gegebenen Bedingungen angepasst und stark vereinfacht.

4.3.1. Durchführung

Nun werden die Gefahren, die durch die FTA ermittelt wurden und die dazugehörigen Komponenten, in ihrem Risiko bewertet. Dabei wird von der zunächst beschriebenen Strategie der Risikoanalyse abgewichen. Dies beruht auf der Problematik der mangelnden fachlichen Erfahrungen, die Kennwerte für einzelne Komponenten festzulegen. Zu beachten ist auch, dass selbst im praktischen Bereich die Werte der Risikoanalyse nicht immer mit Zahlen erfasst werden. Die Erfassung kann zum einen nicht möglich sein oder zum anderen entsteht dadurch

kein praktischer Nutzen durch das Kennen der Werte. Grundlegend wird in diesem Abschnitt davon ausgegangen, dass Komponenten des Systems, die selber entworfen und entwickelt werden, weniger robust und damit anfälliger für einen Ausfall sind, als diese, die eingekauft wurden und schon mehrere Jahre in verschiedenen technischen Bereichen erfolgreich eingesetzt werden.

Zunächst sollten die Risiken klassifiziert werden. Es kann gesagt werden, dass jeder Ausfall „Kritisch“ ist, da es keinerlei Redundanz gibt oder andere Maßnahmen zur Erhöhung der Sicherheit ergriffen wurden. Um die Risiken genauer beschreiben zu können, werden die Komponenten zunächst aufgeteilt. Die folgenden Komponenten werden zwingend in jedem Mode benötigt: CSA, S2P und PB. Daher wird ein Ausfall dieser Komponenten im Folgenden nicht nur als „Kritisch“ eingestuft, sondern als „Katastrophal“. Ein Ausfall einer dieser Komponenten hat den direkten Ausfall des Systems zur Folge. Die restlichen Komponenten werden jeweils in maximal zwei Modes benötigt: FCU, P2S, AHRS und RCR. Daher wurden diese als „Kritisch“ eingestuft, da ein Ausfall durch einen alternativen Zustand abgefedert werden könnte.

Eine Besonderheit stellt hier der Zustand „Stabilized“ dar. Dieser benötigt im Wesentlichen alle kritischen Komponenten. Anhand der Notwendigkeit wurde entschieden, welche Schwere der Komponentenausfall nach sich ziehen würde. Weiter wurde unterschieden, dass Komponenten, die keiner eignen Entwicklung unterliegen, deutlich seltener ausfallen können, als solche, die innerhalb des Projektes entworfen wurden. Auf dieser Aussage wurde festgelegt, dass die Ausfallwahrscheinlichkeit der AHRS oder des RCR „Gering“ ist. Alle anderen Komponenten unterliegen der Eigenentwicklung und die Annahme ist, dass solche Komponenten häufiger ausfallen. Dementsprechend wurde der Ausfall als „Gelegentlich“ eingestuft. In folgender Tabelle 4.4 wurden alle Komponenten aufgelistet. Um das Risiko zu identifizieren, wurde die Schwere und Frequenz pro Komponente angegeben.

Generell muss des Weiteren noch erwähnt werden, dass im Folgenden immer der Ausfall der gesamten Komponente betrachtet wird. In der aktuellen Architektur und den Beschreibungen der Komponenten ist keine Fehlertoleranz vorgesehen. Das bedeutet, dass eine Fehlfunktion einer Komponente im Regelfall in einem Failure dieser resultiert. Daher ist es nicht notwendig, dass die Fehlfunktionen gesondert betrachtet werden.

Die hier verwendeten Begriffe „Katastrophal“, „Kritisch“ und „Gelegentlich“, „Gering“ haben keine konkrete Wertigkeit, sondern dienen lediglich der klaren Abgrenzung und sollen die Unterscheidung ermöglichen. Dabei gilt Folgendes:

$$Katastrophal \gg Kritisch$$

und

$$Gelegentlich \gg Gering$$

Komponente	Schwere	Häufigkeit
CSA	Katastrophal	Gelegentlich
S2P	Katastrophal	Gelegentlich
PB	Katastrophal	Gelegentlich
FCU	Kritisch	Gelegentlich
P2S	Kritisch	Gelegentlich
AHRS	Kritisch	Gering
RCR	Kritisch	Gering

Tabelle 4.4.: Zuordnung der ermittelten Schwere und Häufigkeit, je Komponente, des AES Systems.

Nach dem Einteilen der Komponenten kann durch die Tabelle 4.5, die durch die DIN EN 61508 bereitgestellt wird, jedes dieser Risiken klassifiziert werden. Dabei ergeben sich folgende Klassen je Komponente:

Komponente	Klasse
CSA	I
S2P	I
PB	I
FCU	II
P2S	II
AHRS	III
RCR	III

Tabelle 4.5.: Festlegen der Risikoklassen, je Komponente, des AES Systems.

Anhand dieser Klassen ist es nun möglich, die Akzeptanz zu ermitteln und die nötigen Gegenmaßnahmen zu diskutieren. Dies geschieht unter anderem unter Zuhilfenahme folgender Tabelle 2.5. Die Akzeptanz in unserem Umfeld und anhand des Systems wird schriftlich festgehalten und sieht folgendermaßen aus:
Definitiv nicht tolerierbar ist momentan der Ausfall des CSA, der S2P sowie des PB. Weiter sollte beachtet werden, dass die FCU und P2S auch nicht tolerierbar sind und es auch nicht

unverhältnismäßig hoch wäre, dies sicherzustellen. Im Gegensatz dazu sollte in der ersten Iteration des Projektes die AHRS sowie der RCR als tolerierbar eingestuft werden. Unter anderem unter der Aussage, dass die Kosten hierfür deutlich höher wären und der Mehrwert gering.

4.3.2. Zusammenfassung

Grundlegend muss zu der hier durchgeführten Analyse zunächst gesagt werden, dass diese sehr stark vereinfacht wurde. Daraus ergibt sich eine nur kleine Aussagekraft. Grundsätzlich orientiert sich die Analyse am im Kapitel 2 vorgestellten Konzept. Es wurden viele Dinge sehr stark verallgemeinert. Der Grund dafür ist ein zu geringes Wissen über die Eigenschaften der Komponenten und eine zu geringe Erfahrung des Autors. Diese stark verallgemeinerte Analyse hat als solche keine starke Aussagekraft. Die Tabellen können dennoch als Hilfestellung genutzt werden, da zumindest ein Problem hier noch einmal deutlich sichtbar wird: Die Komponenten CSA, S2P und PB werden unter allen Umständen benötigt und es gibt keinerlei Möglichkeit einen Ausfall zu tolerieren. Ansonsten sollte eine solche Analyse ein weiteres Mal durchgeführt werden, wenn das System konkretisiert wurde und somit bessere Annahmen getroffen werden können.

5. Safe-Life Board

In diesem Kapitel wird auf Basis der Kapitel 3 und 4 ein Lösungsentwurf entwickelt, implementiert und verifiziert. Zu Beginn werden die Ergebnisse aus den eben genannten Kapiteln verwendet, um Anforderungen bestimmen. Nach dem Festlegen der Anforderungen werden zunächst einige Hardware-Plattformen betrachtet, auf die die Problemlösungen aufsetzen kann. Nach der Auswahl der Plattform ist es möglich, im nächsten Schritt den Lösungsvorschlag und die Schnittstellen zu spezifizieren. Des Weiteren müssen die Anforderungen an die anderen Komponenten festgelegt werden. In den beiden letzten Schritten wird auf Basis der Anforderungen und der Spezifikation ein Entwurf vorgestellt und implementiert. Im Anschluss wird dieser verifiziert. Das Ergebnis stellt das Safety-Konzept dar.

5.1. Anforderungen

Es werden zunächst die entsprechenden Anforderungen abgeleitet. Dies geschieht, wie bereits erwähnt, auf Basis der Kapitel 3 und 4. Die Anforderungen können der Tabelle 5.1 entnommen werden. Dabei besteht jede Anforderung aus einer eindeutigen Kennung, um die Zuordnung zu gewährleisten, und einer schriftlichen Erläuterung.

Die Anforderungen aus der Tabelle 5.1 wurden auf Basis von vier verschiedenen Umständen ermittelt:

- Allgemeine Anforderungen
- Anforderungen aus Ermittlung von Systemzuständen
- Anforderungen aus der Gefahrenanalyse
- Anforderungen aus der Risikoanalyse

Zum einen gibt es Aussagen zu generellen Bedingungen, die berücksichtigt werden müssen. Schaut man beispielsweise in den Bereich der Luftfahrt oder Eisenbahnen, wird immer noch ein Mensch als letzte Instanz eingesetzt. Ebenfalls wird eine solche in den ersten Schritten für das hier betrachtete System vorausgesetzt. Des weiteren soll hier der Umschaltvorgang zwischen FCU und RCR erfolgen. Als Basis für den Umschaltvorgang dienen die in Kapitel 3 vorgestellten Systemzustände. Außerdem war bisher unklar, wie das RS konkret ausgelöst werden kann. Dazu wird ein Konzept vorgestellt. Ziel der hier entworfenen Komponente soll

#	**Anforderung**
SLB-01	Realisierung des Umschaltvorgangs zwischen den einzelnen Modes. Berücksichtigung der benötigten Komponenten, bevor ein Schaltvorgang durchgeführt wird.
SLB-02	Kommunikationsschnittstellen zum RS bereitstellen. Es soll sichergestellt werden, dass als letzte mögliche Konsequenz eines Systemausfalls das RS ausgelöst wird, um einen Safe-State zu erreichen.
SLB-03	Das SLB soll eine zentrale Überwachung der einzelnen Komponenten ermöglichen. Es muss zu jeder zu überwachenden Komponente mindestens eine Schnittstelle bereitstellen, bei der sich die Komponente in regelmäßigen Abständen melden muss.
SLB-04	Das SLB sollte die eigene Funktionalität sicherstellen. Das bedeutet, dass das SLB sich selber Überwachung können sollte und bei einem ermittelten Ausfall Maßnahmen einleiten sollte.
SLB-05	Der Pilot (Safety-Link) genießt die höchste Priorität und muss immer die Möglichkeit haben einzugreifen.
SLB-06	Alle ermittelten Gefahren und Risiken müssen beim Entwurf beachtet werden. Das bedeutet nicht, dass für alle Problematiken eine Lösung entworfen werden muss, dennoch müssen diese entsprechend an andere Komponenten verteilt werden.
SLB-07	Es sollte getestet werden, ob alle Gefahren, Risiken und sonstige Entwurfsentscheidungen berücksichtigt wurden. Wurden sie nicht beachtet, ist dies schriftlich festzuhalten.
SLB-08	Das Ergebnis muss das Restrisiko des Gesamtsystems reduzieren.
SLB-09	Sicherstellung, dass eine Auslösung des RS nicht mehr zurückgenommen werden kann.
SLB-10	Vorschlag, wie die entworfene Komponente in das Gesamtsystem intrigiert werden kann und Begründung der Entscheidung.
SLB-11	Ein Fehler im Zustand „Stabilized“ oder Zustand „Autonomous“ muss nicht zwingend zum Auslösen des RS führen, es sollte zunächst eine Steuerung durch den Zustand „Manual“ versucht werden.

Tabelle 5.1.: Auflistung der Anforderungen an das SLB.

die Erhöhung der Gesamtsicherheit sein. Dazu müssen die ermittelten Gefahren sowie Risiken aus dem Kapitel 4 berücksichtigt werden. Da es sich hierbei um ein Sicherheitskonzept handelt, sollte selbstverständlich alles Nötige getan werden, um vom Entwurf über die Implementierung, bis hin zum Testen und der Verifikation, den Sicherheitsgedanken immer im Vordergrund zu halten. Eine der Leitaussagen ist daher: „Keep it simpel“.

5.1.1. Zusammenfassung

Die ermittelten Anforderungen sollten beim Safety-Konzept, also dem Lösungsentwurf, entsprechend berücksichtigt werden. Durch die Anforderungen soll versucht werden den Arbeitsumfang einzugrenzen. Ziel soll es zunächst sein die generelle Sicherheitsproblematik zu lösen. Durch Änderungen an der Systemarchitektur könnte es möglich sein das die hier definierten Anforderungen nicht mehr mit dem Lösungsentwurf übereinstimmen. Daher kann gesagt werden, dass entsprechende Änderungen zunächst in den Anforderungen berücksichtigt werden müssen, um im nächsten Schritt die Anpassungen am Entwurf durchzuführen. Basierend auf den Anforderungen ist es nun möglich eine geeignete Plattform auszuwählen, um darauf das konkrete Konzept zu entwickeln, implementieren und zu testen.

5.2. Recherche

Im folgenden Kapitel werden zunächst verschiedene Hardware-Plattformen betrachtet, die für eingebettete Systeme typisch sind. Außerdem wurde ein bereits umgesetztes Safety-Konzept betrachtet, welches einen Lösungsvorschlag auf Basis eines CPLD entspricht.

5.2.1. Plattform

Zur Erfüllung der in Tabelle 5.1 aufgestellten Anforderungen, wird eine Plattform benötigt. Im Folgenden werden drei verschiedene Plattformen betrachtet und bewertet, um letztlich einen der vorgestellten Vorschläge begründet auszuwählen.

Mikrocontroller

Mikrocontroller bestehen im Wesentlichen aus einem Prozessor sowie verschiedener integrierter Peripherie. Grundlegend könnte man nun davon ausgehen, dass ein solches System perfekt für das gegebene Problem geeignet ist. Es bietet die benötigte Peripherie. Die Programmierung erfolgt normalerweise durch eine Hochsprache, wodurch das Ganze ebenfalls sehr

flexibel ist. Des Weiteren existieren inzwischen Controller, die extra für den Einsatz in sicherheitskritischen Systemen geeignet sind. Ein Beispiel für einen solchen Mikrocontroller ist der AURIX^{TM}TriCore^{TM}der Firma Infineon [30]. Dabei setzen solche Chips intern und bei ihrer Peripherie bereits auf Redundanz sowie redundante zeitversetzte Ausführung des Sourcecodes (genannt Delayed Lookstep). Auf der anderen Seite ist ein Mikrocontroller ganz allgemein betrachtet für das hier zu lösende Problem deutlich überdimensioniert. Unter dieser Bedingung kann man nicht behaupten, dass die Lösung möglichst simpel gehalten wurde. Zwar die notwendige Peripherie bereits implementiert, was einem beim Entwurf Zeit spart. Aber auf der anderen Seite muss sich bei dem Einsatz eines solchen Gedanken zu einem Scheduling gemacht werden. Außerdem ist das Weiterleiten eines Signals, auf dieser Plattform, nicht einfach in der Umsetzung.

FPGA

FPGA steht für Field Programmable Gate Array. Grundlegend hat man hier die Möglichkeit durch die Rekonfiguration der internen Strukturen verschiedenste digitale Schaltungen zu realisieren. Dabei ist der Skalierung bei aktuellen Geräten kaum noch Grenzen gesetzt, von einfachsten IO-Funktionen, bis hin zur Synthese ganzer Mikroprozessoren, als sogenannten Softcores. Grundsätzlich ist es nicht der Sinn eines FPGAs einen ASIC nach zu bilden. FPGAs werden im Regelfall für Aufgaben eingesetzt, die sehr gut parallelisiert werden können oder als Hilfsprozessor für einen klassischen Mikrocontroller dienen. In diesem können auf jeden Fall die Anforderungen auf eine einfache Art und Weise integriert werden. Ein anderes Problem ist, das ein FPGA zusätzlich noch Flashspeicher benötigt, um nach einem Spannungsverlust die Konfigurationsdaten zu laden. Auch die Ladezeit sollte hier beachtet werden, da ein allgemeiner Irrglaube darüber herrscht, dass ein FPGA nach dem Einschalten sofort arbeitet.

CPLD

CPLD steht für Complex Programmable Logic Device. Zum größten Teil bestehen CPLDs aus einer AND/OR Matrix. Außerdem gibt es Ein- und Ausgabeblöcke, die zum Teil mit zusätzlichem Speicher verbunden werden. Im direkten Vergleich zu einem FPGA ist ein CPLD deutlich simpler gefertigt. Es ist möglich, sehr genaue Zeitaussagen zu den Schaltvorgängen zu machen. Natürlich kann ein CPLD bei der Rechengeschwindigkeit nicht mit einem FPGA mithalten und die Komplexität der zu synthetisierenden Schaltung ist ebenfalls begrenzt. Ein anderer großer Pluspunkt ist der grundlegende Aufbau. Ein CPLD ist EEPROM (Electrically Erasable Programmable Read-Only Memory) basiert. Somit muss bei einem CPLD kein extra Speicher verwendet werden und nach dem Starten ist das Programm sofort geladen.

Rudimentäre Schaltung

Eine weitere Möglichkeit wäre der Aufbau einer einfachen logischen Schaltung. Eine viel simplere Implementierung kann fast nicht mehr erreicht werden. Ein großes Problem wiederum ist die sehr geringe Flexibilität. Ebenfalls ist die Größe einer solchen Schaltung nicht vergleichbar mit der eines kleinen Chips in Form eines Mikrocontrollers, FPGAs oder CPLDs. Ein Vorteil, der auch den noch heutigen Einsatz solcher Aufbauten rechtfertigt, ist die Resistenz gegenüber Strahlung oder anderen Umwelteinflüssen. Daher werden solche groben Raster in der Raumfahrt für besonders kritische Komponenten immer noch gerne verwendet. Für unser System ist diese Betrachtung momentan noch nicht relevant, da wir uns keine Gedanken über Strahlungsresistenz machen müssen.

5.2.2. SLB Plattform

Die Auswahl der Plattform wurde auf ein CPLD festgelegt. Die Begründung dafür ist recht einfach. Zum einen können alle benötigten Anforderungen und Mechanismen sehr einfach durch VHDL (Very High Speed Integrated Circuit Hardware Description Language) beschrieben, simuliert und synthetisiert werden. Dadurch wird die Lösungen in den räumlichen Ausmaßen sehr kompakt. Des Weiteren werden außer das CPLD keine anderen Bauteile benötigt, um ein alleinstehendes Board zu entwerfen. Nachdem die Spannungsversorgung eingeschaltet wurde, ist es sofort startbereit und die Schaltzeiten sind relativ konstant. Damit können genaue Zeitaussagen getroffen werden. In der Arbeit [10] wurde ebenfalls ein CPLD für die Entwicklung eines Fail-Safe Systems verwendet. Ein weiterer Vorteil, der unter anderem aus dieser Arbeit hervorgeht, ist die Testbarkeit. Bei Mikrocontrollern beispielsweise gibt es eine viel Zahl von Modulen, die von der Software automatische verwendet werden, und entsprechende Probleme, mit diesen, müssten getestet werden. Ein CPLD basiertes System ist entsprechend für das zu lösende Problem spezialisiert und es enthält nur Module, welche auch wirklich benötigt werden.

Für unsere Zwecke soll ein CoolRunner-II CPLD der Firma Xilinx [31] zum Einsatz kommen. Die Ressourcen dieser Chipfamilie sollten zur Implementierung der gewollten Features ausreichen. Die Auswahl des Chips verlief recht einfach. Zum einen haben wir an der Hochschule Zugang zu der Entwicklungssoftware von Xilinx. Dadurch ist die Möglichkeit einen anderen Hersteller zu verwenden, recht begrenzt und wäre deutlich umständlicher. Zum anderen ist die Auswahl von CPLDs inzwischen überschaubar. Der gewählte Chip repräsentiert dabei den aktuellen Stand der Technik. Der Schwerpunkt der Hersteller liegt momentan bei der Entwicklung von FPGAs oder Mikrocontrollern und seit jüngster Zeit die Kombination beider Welten.

5.3. Design

In diesem Kapitel wird der Lösungsentwurf des Safety-Konzepts beschrieben. Im ersten Schritt werden die Schnittstellen detailliert beschrieben. Des weiteren werden aus den Schnittstellen entsprechende Anforderungen an die anderen Komponenten hergeleitet, die im weiteren Verlauf ebenfalls aufgelistet sind. Hierbei wird das SLB zunächst als Blackbox betrachtet. Erst im zweiten Teil des Kapitels wird der interne Aufbau des SLB ausführlich beschrieben. In diesem Abschnitt wird der Aufbau der Module beschrieben. Außerdem wird auf die Schnittstellen zwischen den Modulen untereinander eingegangen. Des weiteren kann dort eingesehen werden, wie die vorher beschriebenen Schnittstellen innerhalb der internen Struktur weitergereicht und die Signale verwendet werden.

5.3.1. Schnittstellen

Der Aufbau der Komponenten und dessen Links können der Abbildung 5.1 entnommen werden. Die Schnittstellen der Komponenten können grundsätzlich in mehrere Kategorien unterteilt werden. Zum einen existieren die Safety-Links für die Komponenten:

- P2S Watchdog
- PB Watchdog
- FCU Watchdog
- CSA Watchdog
- S2P Watchdog

Bei diesen Links handelt es sich um jeweils einen einfachen Pin. Dieser wird innerhalb des SLB auf einen Reset des Watchdog-Counter geführt. Dabei wird durch diese Art des Aufbaues sichergestellt, dass das SLB weiß ob in seinem Umfeld noch alles in Ordnung ist. Durch die verwendeten Watchdogs ist jede Komponente gezwungen, sich regelmäßig beim SLB zu melden. Weitere Schnittstellen sind solche, die eine direkte Kommunikation mit dem SLB ermöglichen. Dabei handelt es sich ebenfalls um einfache Signallösungen. Für jede Komponente gibt es die Möglichkeit ein Fehlverhalten direkt an das SLB zu melden. Dadurch muss nicht das Auslaufen des jeweiligen Watchdog (WD) abgewartet werden, wenn der Komponente ein möglicher Ausfall bereits bekannt ist. Folgende Signal Benennungen sind dafür vorgesehen:

- P2S Failure
- PB Failure
- FCU Failure
- CSA Failure
- S2P Failure

Wie bereits erwähnt, gibt es noch weitere Links zur direkten Kommunikation. In der folgenden Auflistung wurden jeweils die Benennung aufgelistet. Im darauf folgenden Abschnitt wird jedes Signal kurz erläutert.

- RCR No Signal
- RCR Switch
- RCR Activate RS
- FCU Activate RS
- RS Activate

Diese Schnittstellen bezeichnen eine direkte Kommunikation mit dem SLB. Das Signal „RCR No Signal“ soll angeben, ob der RCR noch mit einem Transmitter, oder anders gesagt mit dem Piloten, verbunden ist. Bei dem Signal „RCR Activate RS“ und „FCU Activate RS“ handelt es sich um die direkte Möglichkeit durch die FCU oder vom Piloten aus, das RS zu aktivieren. Der „RCR Switch“ wiederum ist ein Signal, welches angibt, ob im Manual Mode geflogen wird (Signalpegel ist ’0’) oder ob im Zustand „Stabilized“/„Autonomous“ geflogen werden soll (Signalpegel ist ’1’). Nach der Beschreibung der Flight Modes aus Kapitel 3 gibt es eine Unterscheidung zwischen drei Modes. Auf Grundlage der Anforderungen konnte das System auf zwei Zustände vereinfacht werden (SLB-05). Des Weiteren existiert noch das Signal „RS Activate“. Dieses wird vom SLB ausgegeben und soll das RS auslösen. Eine weitere Kategorie stellen die Steuersignale des Piloten und der FCU dar. Im Folgenden ist eine kurze Auflistung dieser zu sehen:

- RCR SPI Input
- FCU SPI Input
- FCU SPI Output
- CSA SPI Output

Dabei ist ein SPI-Signal nicht nur ein einziges Signal, sondern besteht aus insgesamt vier Signalen. Zum einen aus einem Takt (Clock, kurz CLK), einem Auswahlsignal (Chip Select, kurz CS) und zwei Datenkanälen, für ausgehende (Serial Data Out, SDO) und eingehende (Serial Data In, SDI) Daten. Je nach Modus müssen die Kanäle gewechselt werden. In das SLB laufen die RCR-Steuerbefehle und die FCU-Steuerbefehle ein. Die Namen der Verbindungen lauten „RCR SPI Input“ und „FCU SPI Input“. Je nachdem, welche Komponente gerade die Steuerung des Trägers übernimmt, wird einer der SPI Streams via „CSA SPI Output“ an den CSA weitergeben. Außerdem existiert noch ein statischer Pfad, namens „FCU SPI Output“. Dabei werden immer die Steuerbefehle des Piloten an die FCU weitergeleitet, zum Zwecke des Loggings. Unter der anmahne der oben beschriebenen SPI-Kanäle, benötigt man für alle vier Signale jeweils vier Kanäle. In dem hiesigen Fall muss SPI nur unidirektional und nicht bidirektional eingesetzt werden. Daher sind für jedes Signal drei Leitungen vorgesehen: CLK, CS und SDI oder SDO (abhängig von der Richtung).

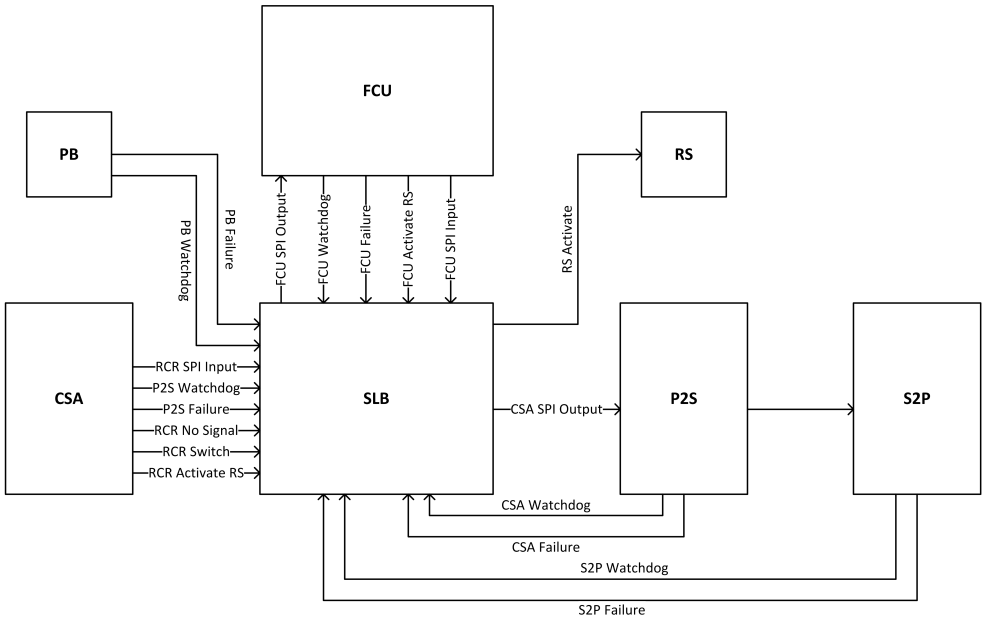

Abbildung 5.1.: Übersicht aller spezifizierten Schnittstellen und grafische Zuordnung zu den einzelnen Komponenten.

5.3.2. Safety-Anforderungen an die Komponenten

In diesem Abschnitt sollen die Anforderungen an die einzelnen Komponenten beschrieben und begründet werden. Der Tabelle 5.2 können die Anforderungen entnommen werden. Dabei hat jede Anforderung eine eindeutige Kennzeichnung, eine Beschreibung sowie eine Zuordnung zu einer Komponente. Die hier erstellten Anforderungen beziehen sich ausschließlich auf folgende Komponenten:

- FCU
- CSA
- PB
- P2S
- S2P
- RS

Generell kann für jede Komponente festgehalten werden, dass sie einen Fehler in ihrer Umgebung, also an einer an ihr angeschlossenen Komponente, erkennen muss. Die Erkennung aller möglichen Fehler, zwischen den einzelnen Komponenten, wäre viel zu aufwendig. Dabei können hier die Fehlermöglichkeiten selbstverständlich sehr vielfältig sein und die Erkennung dadurch sehr unterschiedlich und zum Teil recht umständlich. Beim Erkennen einer solchen

Problematik, welcher Auslöser es auch immer sein mag, und beim Feststellen des dadurch folgenden Versagens kann die Komponente nun direkt eine Rückmeldung an das SLB geben. Dazu steht jeder Komponente ein direkter Kommunikationskanal zur Verfügung. Des Weiteren existiert ein zweiter Safety-Link. Dazu setzt das SLB für jede Komponente einen Watchtdog zur Prüfung der Verfügbarkeit ein. Jede Komponente muss sich nun innerhalb von maximal 10 ms bei dem SLB melden, um den intern laufenden Watchdog zurückzusetzen. Die 10 ms wurde vorerst einfach als Wert festgelegt und haben keinen tieferen Sinn. Dieser kann beim konkreten Entwurf angepasst werden. Dabei muss stets ein Signalwechsel erfolgen. Ausgenommen von den eben beschriebenen Sicherheitslinks ist das RS. Bei dem hier verwendeten WD handelt es sich präziser um einen Time-Out-Watchdog. Dabei zählt ein Zähler von einem bestimmten Wert runter. Wenn der Wert null erreicht ist ein Fehler aufgetreten. Dies kann nur durch ein Reset des WD, genauer des Zählers, verhindert werden.

Eine höhere Anzahl an Anforderungen werden an die P2S und FCU gestellt. Die P2S muss zusätzlich drei weitere Signale erzeugen. Sie soll feststellen, ob der Pilot noch Signale sendet, beziehungsweise ob dieser noch mit dem RCR verbunden ist. Dazu soll der entsprechende Status via „RCR No Signal“ weitergereicht werden. Zum Wechseln der Zustände soll ebenfalls eine Schnittstelle zur Verfügung stehen. Auf Grundlage der in Tabelle 5.1 festgehaltenen Anforderung (SLB-05) wird die Unterscheidung von lediglich zweier Zuständen nötig. Wenn somit der Pin der Schnittstelle logisch ’0’ ist, wird der Zustand „Manual“ ausgeführt. Bei einer logischen ’1’, die an dem entsprechenden Pin anliegt, handelt es sich um den Zustand „Stabilized“ oder den Zustand „Autonomous“. Die P2S sowie die FCU besitzen jeweils noch ein Signal, welches das Auslösen des RS direkt ermöglichen soll. Wenn einer dieser Links auf eine logischen ’1’ geschaltet wird, soll das RS ausgelöst werden. Die FCU kann nur das RS direkt auslösen, wenn sie über die Steuerung des Trägers verfügt.

Die letzten relevanten Schnittstellen sind die Kanäle zur Datenübertragung via SPI. Dabei müssen die passenden drei SPI-Kanäle der Pfeilrichtung entsprechend der Abbildung 5.1 an das SLB weitergereicht werden. Ein Pfeil in das SLB herein bedeutet, das die vorher geschaltete Komponente ein SDO-Kanal bereitstellen muss. Ein Pfeil aus dem SLB heraus steht für ein SDI-Kanal. Intern werden alle Maßnahmen zum Umschalten getroffen. In der folgenden Tabelle 5.2 werden alle Anforderungen an die umliegenden Komponenten übersichtlich dargestellt.

Bisher wurden die kritischen Komponenten AHRS und RCR nicht betrachtet. Das liegt grundsätzlich daran, dass das SLB keine direkte Verantwortung für diese trägt und es keine Möglichkeit hat, diese zu prüfen. Die Verantwortung wurde entsprechend der Zugehörigkeit an die beiden Subsysteme, der FCU und dem P2S, weitergeben. Somit ergibt sich die Notwendigkeit, die benötigten Prüfungen durch die Komponenten durchzuführen. Die hier beschriebene Anforderung kann auf E-SLB-04 der Tabelle 5.2 zurück geführt werden.

#	Anforderung	Komponente
E-SLB-01	Wenn keine RCR-Daten empfangen werden können, muss das „RCR No Signal“ eine logische ’1’ sein, ansonsten logisch ’0’.	P2S
E-SLB-02	Unterscheidung des Zustandes „Manual“ durch logisch ’0’ und des Zustandes „Stabilized“/„Autonomous“ durch eine logische ’1’. Durch die gegebenen Anforderungen aus Tabelle 5.1 ist keine weitere Unterscheidung nötig.	P2S
E-SLB-03	Jedes Watchdog Signal muss spätestens alle 10 ms durch ein Signalwechsel zurückgesetzt werden.	FCU, CSA, S2P, P2S, PB
E-SLB-04	Jede Komponente muss für sich selber die Umgebung prüfen und entsprechende Sicherungsmaßnahmen intern implementieren. Bei einem Fehler kann dies durch eine logische 1, durch ein direkt verbundenes Signal, dem SLB mitgeteilt werden.	Alle
E-SLB-05	Das RS-Signal, ausgehend vom SLB, ist „Low-Active“. Nach dem Einschaltvorgang wird das Signal direkt auf eine logische ’1’ gesetzt. Eine weitere Bedingung ist das Abschalten der Triebwerke, oder kappen der Spannungsversorgung beim Auslösen des RS.	RS, Hinweis
E-SLB-06	Das direkte Auslösen des RS durch die FCU oder den Piloten erfolgt durch eine logische ’1’ am SLB und ist nur im zugehörigen Flight Mode durchführbar.	FCU, P2S
E-SLB-07	Für jede SPI-Schnittstelle sind genau drei Pins am SLB vorgesehen. Dabei wird die kleinste Pinnummer für den Takt, die größte für die Daten und die mittlere für das Chipselect, verwendet. Die Richtung kann der Abbildung 5.1 entnommen werden.	P2S, FCU, CSA
E-SLB-08	Das RS muss auch ohne Spannungsversorgung ein Auslösen des Sicherungssystems ermöglichen. Eine Idee wäre ein Schalter, der ohne eine Spannungsversorgung deaktiviert wird und das System somit aktiviert.	RS
E-SLB-09	Es muss eine Möglichkeit geben, beim Auslösen des RS ebenfalls die Triebwerke abzuschalten oder die Versorgungsspannung dieser zu unterbrechen.	PB, ESC, Träger Allgemein

Tabelle 5.2.: Auflistung aller Safety-Anforderungen an externe Komponenten.

Zusammenfassung

Die in diesem Abschnitt beschriebenen Safety-Anforderungen gelten für die entsprechenden angegebenen Komponenten. Der Lösungsentwurf basiert ebenfalls auf diesen und setzt die entsprechenden Anforderungen voraus, um einen ordnungsgemäßen Betrieb zu ermöglichen. Werden Anpassungen an der Architektur vorgenommen, kann es möglich sein, dass entsprechende neue Anforderungen festgelegt werden müssen. Ein wichtiger Punkt, der berücksichtigt werden sollte, ist das Kommunizieren der Anforderungen mit den Komponenten-Entwickler und die Sicherstellung, dass sich alle Beteiligten an diese halten. Die Nichteinhaltung der Anforderungen würde die Sicherheit negativ beeinflussen können oder es könnte zu Problemen und ungewollten Ausfällen des Systems kommen.

5.3.3. Interner Aufbau

In diesem Abschnitt wird der interne Lösungsentwurf beschrieben. Dieser basiert auf der Emergency Shutdown Architecture, die bereits in Kapitel 2.2.4 vorgestellt wurde. Der globale Aufbau kann der Abbildung 5.2 entnommen werden. Im Vergleich Es werden zur Lösung zwei CPLDs, Master und Slave verwendet. Beide haben einen separaten Kanal, um das RS zu aktivieren. Dieser Kanal wird mithilfe eines externen NOR-Gatters geprüft. Das Master CPLD beinhaltet die gesamte Logik zur Prüfung der Komponenten und zum Wechseln des Modes. Das Slave CPLD überprüft nur das Master CPLD, ob dieses noch funktionstüchtig ist. Um das Problem „Who watches the Watchmen“ zu lösen, überwacht das Master CPLD ebenfalls das Slave CPLD. Beide Komponenten müssen immer aktiv sein. Dabei wird hier nur die grundsätzliche Funktionalität der CPLDs geprüft und nicht die der Makrozellen. Somit wird lediglich der Ausfall der gesamten Hardware geprüft. Ein kleinerer Detailgrad macht auch keinen Sinn, da die entsprechende Prüfung kompliziert und umfangreich wäre und die erreichte Sicherheitserhöhung dazu nicht in Relation steht. Ab einem bestimmten Grad sollte man den Komponenten vertrauen. Durch eine ausgiebige Prüfung und umfangreiche Tests sollte bereits die Anzahl der systematischen Faults ausreichend reduziert werden können.

Auch zu den beiden Safety-Anforderungen E-SLB-08 und E-SLB-09 der Tabelle 5.2 muss abschließend noch erwähnt werden, dass im aktuellen Entwurf nur ein RS Signal erzeugt wird. Dieses muss, je nach konkreter Umsetzung des RS und dem möglichen Abschalten der Triebwerke, noch angepasst werden. Die aktuelle Entwurf verwendet nur genau ein Signal. Das Signal ist „Low-Active“ (durch das abschließende NOR-Gatter, zu sehen in Abbildung 5.2). Die Entwurfsentscheidung für die Verwendung eines „Low-Active“ Signals beruht auf der Tatsache, dass wenn die Spannungsversorgung ausfällt, die CPLDs ebenfalls ausfallen würden. Ein „Low-Active“ Signal würde das Auslösen des RS trotzdem sicherstellen.

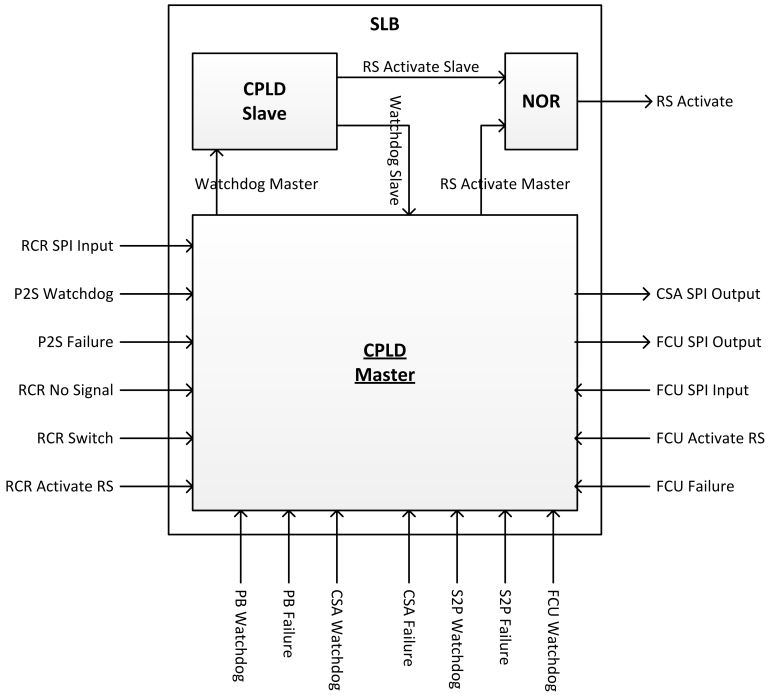

Abbildung 5.2.: Globale Sicht auf das SLB-Board mit allen ein- und ausgehenden Signalen und den Signalen zwischen den Modulen des SLB.

Master CPLD

In der Abbildung 5.3 kann der Aufbau des Master CPLD betrachtet werden. Es besteht aus einem Mode Switch, einem RS Controller und einem WD-Timer. Die Komponente Mode Switch

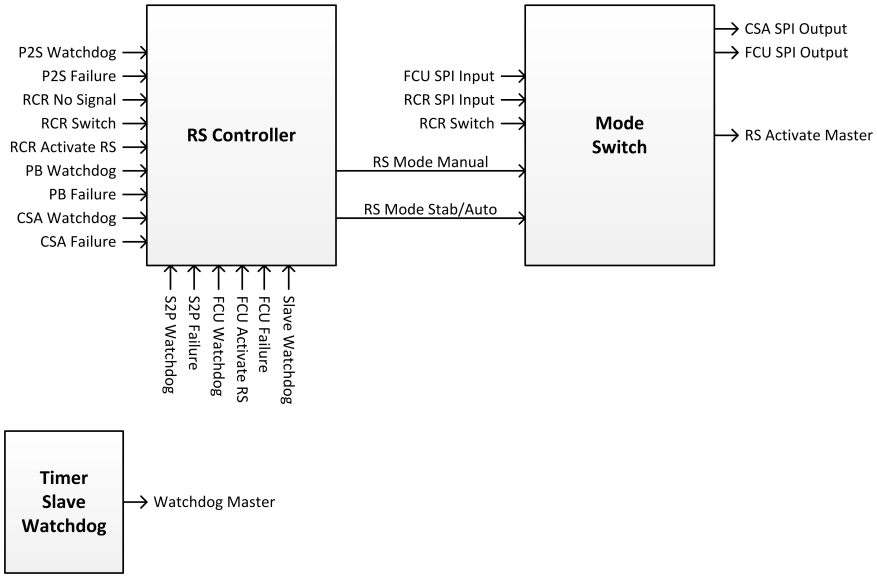

Abbildung 5.3.: Modul-Ansicht auf das Master CPLD und allen ein- sowie ausgehenden Signale.

besteht zum einen aus einem Moore-Automat, der der Abbildung 5.4 entnommen werden kann. Zum anderen aus den Multiplexern, die anhand des Modes die SPI-Signale richtig verteilen. Außerdem trifft der Automat, je nach Zustand, die Entscheidung, ob das RS ausgelöst werden soll oder nicht. Der RS Controller beinhaltet zum einen die verschiedenen WDs, die von den einzelnen zu überwachenden Komponenten zurückgesetzt werden müssen. Zum anderen entscheidet der Controller, ob ein Ausfall des Zustandes „Manual“ und/oder des Zustandes „Stabilized“/„Autonomous“ möglich ist. Diese Entscheidung wird auf Basis kombinatorischer Logik getroffen und ist aus den Wahrheitstabellen 5.3 und 5.4 abgeleitet.

Die beiden Wahrheitstabellen 5.3 und 5.4 wurden entsprechend in VHDL implementiert. Um das Ganze zu vereinfachen, wurde die Disjunktive Normalform (DNF) verwendet. Es musste somit jeweils nur das Ergebnis logisch ’0’ beider Tabellen implementiert werden, da alle anderen Ergebnisse sowieso eine logische ’1’ ergeben. Die letzte Komponente läuft komplett unabhängig von den anderen. Dabei handelt es sich um einen einfachen Timer, der im Zyklus ein Reset-Signal für das Slave CPLD erzeugt und dadurch den dort implementierten WD zurücksetzt.

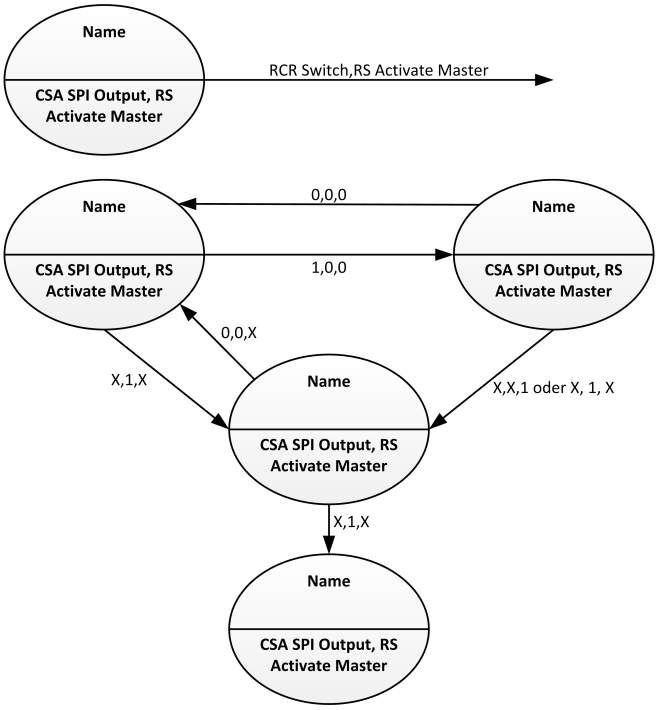

Abbildung 5.4.: Moore-Automat für den Mode-Switch. Dieser orientiert sich an den ermittelten Systemzuständen und an den gegebenen Anforderungen.

#	PB Watch-dog	PB Failure	P2S Watch-dog	P2S Failure	CSA Watch-dog	CSA Failure	S2P Watch-dog	S2P Failure	Slave Watch-dog	RCR Activa-te RS	RCR No Signal	RS Mode Manual
00	0	0	0	0	0	0	0	0	0	0	0	0
01	1	X	X	X	X	X	X	X	X	X	X	1
02	X	1	X	X	X	X	X	X	X	X	X	1
03	X	X	1	X	X	X	X	X	X	X	X	1
04	X	X	X	1	X	X	X	X	X	X	X	1
05	X	X	X	X	1	X	X	X	X	X	X	1
06	X	X	X	X	X	1	X	X	X	X	X	1
07	X	X	X	X	X	X	1	X	X	X	X	1
08	X	X	X	X	X	X	X	1	X	X	X	1
09	X	X	X	X	X	X	X	X	1	X	X	1
10	X	X	X	X	X	X	X	X	X	1	X	1
11	X	X	X	X	X	X	X	X	X	X	1	1

Tabelle 5.3.: Wahrheitstabelle des Master CPLD, speziell für den Zustand „Manual“. Synthese innerhalb des Moduls RS Controller.

#	PB Watch-dog	PB Failure	FCU Watch-dog	FCU Failure	CSA Watch-dog	CSA Failure	S2P Watch-dog	S2P Failure	Slave Watch-dog	FCU Activa-te RS	RS Mode Stab/-Auto
00	0	0	0	0	0	0	0	0	0	0	0
01	1	X	X	X	X	X	X	X	X	X	1
02	X	1	X	X	X	X	X	X	X	X	1
03	X	X	1	X	X	X	X	X	X	X	1
04	X	X	X	1	X	X	X	X	X	X	1
05	X	X	X	X	1	X	X	X	X	X	1
06	X	X	X	X	X	1	X	X	X	X	1
07	X	X	X	X	X	X	1	X	X	X	1
08	X	X	X	X	X	X	X	1	X	X	1
09	X	X	X	X	X	X	X	X	1	X	1
10	X	X	X	X	X	X	X	X	X	1	1

Tabelle 5.4.: Wahrheitstabelle des Master CPLD, speziell für den Zustand „Stabilized“/„Autonomous“. Synthese innerhalb des Moduls RS Controller.

Slave CPLD

In der Abbildung 5.5 kann der Aufbau des Slave CPLD betrachtet werden. Dieses ist deutlich simpler als das Master CPLD aufgebaut. Daher kann der Abbildung auch ein höherer Detailgrad entnommen werden. Es besteht aus zwei Komponenten, die völlig unabhängig voneinander agieren. Zum einen gibt es den Master Watchdog. Dieser muss regelmäßig durch das Master CPLD rückgesetzt werden, da sonst das RS ausgelöst wird. Zum anderen existiert ein Timer, der in regelmäßigen Abständen ein Reset-Signal zum Master CPLD sendet.

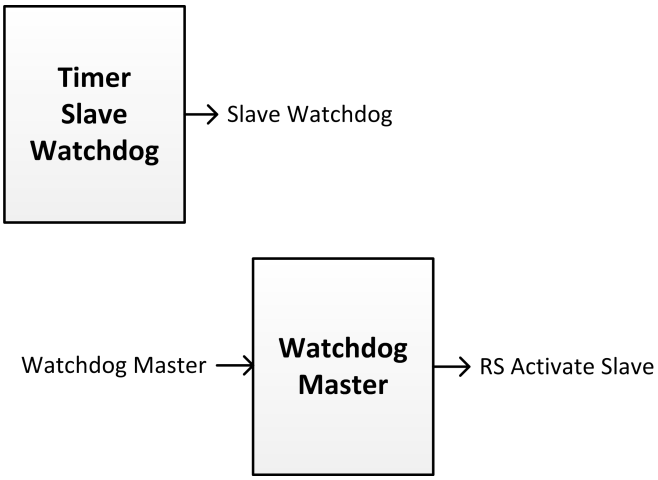

Abbildung 5.5.: Modul-Ansicht auf Slave CPLD und allen ein- sowie ausgehenden Signale.

5.4. Implementierung

In diesem Abschnitt werden Hinweise zur Implementierung des Entwurfs aufgezeigt. Die Lösung wurde mithilfe von VHDL beschrieben und dem Programm ModelSim simuliert und getestet. Auf das Testen wird im Kapitel 5.5 noch detaillierter eingegangen. Der gesamte Quellcode kann im Anhang A gefunden werden. Die Implementierung erfolgte „Bottom Up“. Das bedeutet, dass zunächst die kleinsten Komponenten, also der Watchdog und Timer, entworfen sowie getestet wurden. Danach wurden entsprechende Komponenten angelegt, die diese verwenden, um ihre spezialisierten Aufgaben zu erfüllen (RS Controller, Mode Switch). Zum Schluss wurden die Komponenten in zwei Top-Level Entwürfen zusammengefasst (Master/Slave). Hierbei gab es ansonsten keine Besonderheiten beziehungsweise nennenswerte Eigenschaften, die extra besprochen werden müssen. Die Implementierung orientiert sich direkt an der Spezifikation. Dem VHDL-Code wurde eine Konfigurationsdatei beigelegt. Diese beinhaltet alle nötigen Konstanten. Diese müssen für jedes verwendete CPLD entsprechend berechnet werden, da diese Wertigkeiten passenden für den Takt definiert sind.

Die in der Implementierung verwendeten Signale reagieren immer direkt. Das heißt, ein Eingangssignal muss nicht mindestens eine bestimmte Dauer anliegen, um gültig zu sein. Das kann bei der Hardwareumsetzung zu Problemen führen oder Störsignale könnten möglicherweise unerwünschte Schaltvorgänge auslösen. Daher sollte bei einer konkreten Hardware-Umsetzung geprüft werden, ob für die jeweiligen Eingangssignale ein Mechanismus verwendet werden sollte, der die Länge eines Signals auf eine Mindestdauer prüft und somit die Gültigkeit dieses prüft. Falls dies notwendig ist, sollten die Safety-Anforderungen an die Komponenten entsprechend angepasst werden. Außerdem handelt es sich bei dem verwendeten WD, wie bereits erwähnt, um eine, Time-Out-Watchdog. Dies ist eine der einfachsten zu implementierenden WD-Varianten. Ein besserer WD wäre ein Window Watchdog. Dieser prüft, ob der Reset innerhalb eines festgelegten Zeitfensters durchgeführt wird. Ansonsten wird der WD ausgelöst.

5.5. Test

In dem letzten Abschnitt wird auf den Test des Ergebnisses eingegangen. Im Detail wurde mithilfe von ModelSim der VHDL-Code simuliert. In diesem Abschnitt werden die Simulationsergebnisse geprüft und mit den Anforderungen sowie dem Entwurf verglichen. Es wird anhand der Simulationsergebnisse gezeigt, dass die einzelnen Module, wie im Entwurf gefordert, funktionieren. Dies ist auch ein Vorteil der Simulation. Es kann unter Umständen sehr aufwendig sein, alle Funktionalitäten in echter Hardware zu testen und zu analysieren. Beim Hardware-Entwurf hat sich eine Simulation etabliert, da es zum einen günstiger ist und zum anderen den zeitlichen Aufwand deutlich reduziert. Es wurden alle einzelnen Module vom kleinsten, bis hin

zum gesamten System, geprüft. In diesem Abschnitt werden zum einen die Ergebnisse des Slave CPLD und Master CPLD gezeigt. Außerdem wurden ebenfalls die beiden spezialisierten Module Mode Switch und RS Controller des Master CPLD geprüft. Die kleinsten Module, also der WD und der WD Timer, wurden auch geprüft, werden in diesem Abschnitt aber nicht behandelt. Die Funktionalität der beiden Module kann den anderen Simulationsergebnissen entnommen werden. Da der gesamte Quellcode angehangen wurde (Anhang A), ist es möglich, die Ergebnisse nach zu vollziehen. Die Simulation wurde für alle Komponenten in Mikrosekunden durchgeführt. Das liegt zum einen daran, dass der Simulator ansonsten viel zu lange für die Berechnungen der Ergebnisse benötigt. Zum anderen reicht diese Auflösung bei Weitem, um die Korrektheit der Ergebnisse zu prüfen.

Die Simulation wurde mithilfe von Testbenches durchgeführt. Dabei wurde das Blackbox-Verfahren gewählt. Jedes Modul wurde anhand der Schnittstellen und den Anforderungen an das Modul getestet. Es wurden innerhalb der Testbenches verschiedene Signalfolgen beschrieben. Die Signalfolgen werden über die Schnittstellen an das Modul, auch als DUT (Device Under Test) bezeichnet, weitergereicht. Außerdem werden die Rückgabewerte innerhalb der Testbench entgegengenommen. Alle erzeugten Signale der Testbench und die erhaltenen Signale vom Modul werden in der Simulation angezeigt. So kann anhand dieser die Korrektheit geprüft werden.

Mode Switch

In der Abbildung 5.6 ist eines der Simulationsergebnisse zu sehen. Dieser Abbildung kann zunächst der normale Ablauf des Mode Switch entnommen werden. Im den markierten Bereichen sind zwei Besonderheiten beschrieben. In der Markierung 1 wird gezeigt, dass im Fehlerfall der FCU ein Wechsel in den entsprechenden Zustand „Stabilized“/„Autonomous“ unmöglich ist. Im Detail kann gesehen werden, dass via RCR_Switch_Mode versucht wird, vom Zustand „Manual“ in den Zustand „Stabilized“/„Autonomous“ zu wechseln. Dies klappt nicht, da der RS_Mode_FCU noch aktiv ist (Failure in Zustand „Stabilized“/„Autonomous“).
Anhand der Markierung 2 kann beobachtet werden, wie das System aus dem Zustand „Stabilized“/„Autonomous“ in den Zustand „Safety“ wechselt, da die FCU einen Fehler hat. Das liegt daran, dass das System in im Mode_Output STAB_AUTO ist, als das Signal RS MODE FCU von ’0’ auf ’1’ wechselt. Außerdem ist erkennbar, wie der Wechsel vom Zustand „Safety“ in den Zustand „Manual“ funktioniert. Dies passiert durch den Wechsel des Signals RCR_Switch_Mode von ’1’ auf ’0’ und dadurch, dass der Mode_Output in SAFETY ist.

In der nächsten Abbildung 5.6 ist das zweite Ergebnis zu sehen. Hier wird gezeigt, wie ein Fehler (Markierung 1) im Zustand „Manual“ zum Auslösen des RS führt (Markierung 2). Dabei befindet sich das System, wie dem Signal Mode_Output zu entnehmen ist, im Zustand „Manual“. Durch den Wechsel des Signal RS_Mode_RCR von ’0’ auf ’1’ (Failure im Zustand

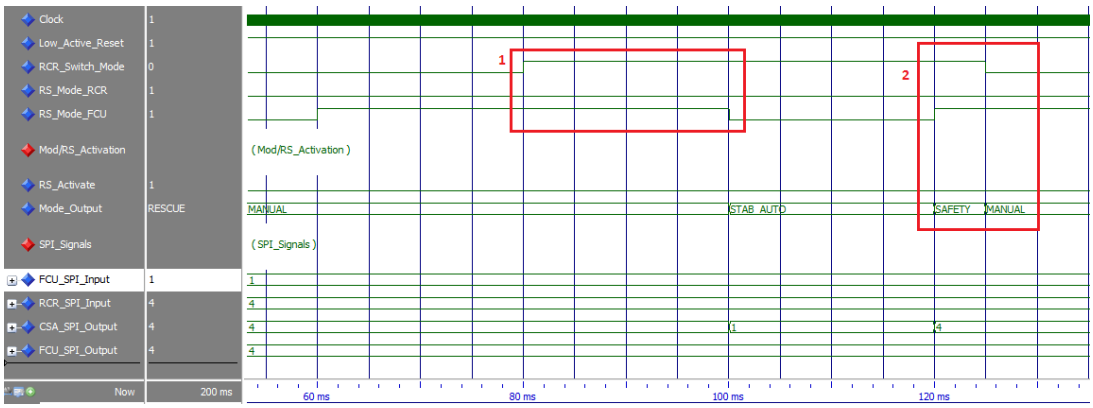

Abbildung 5.6.: Auszug aus dem Simulationsergebnis des Mode Switch Moduls.

„Manual“), wechselt der Mode_Output in den Zustand „Safety“ und prüft dort, ob es möglich ist, Manual weiter zu fliegen. Da dies nicht geht, wechselt das System in den Zustand „Rescue“ und löst das RS aus (RS_Activate von ’0’ auf ’1’).

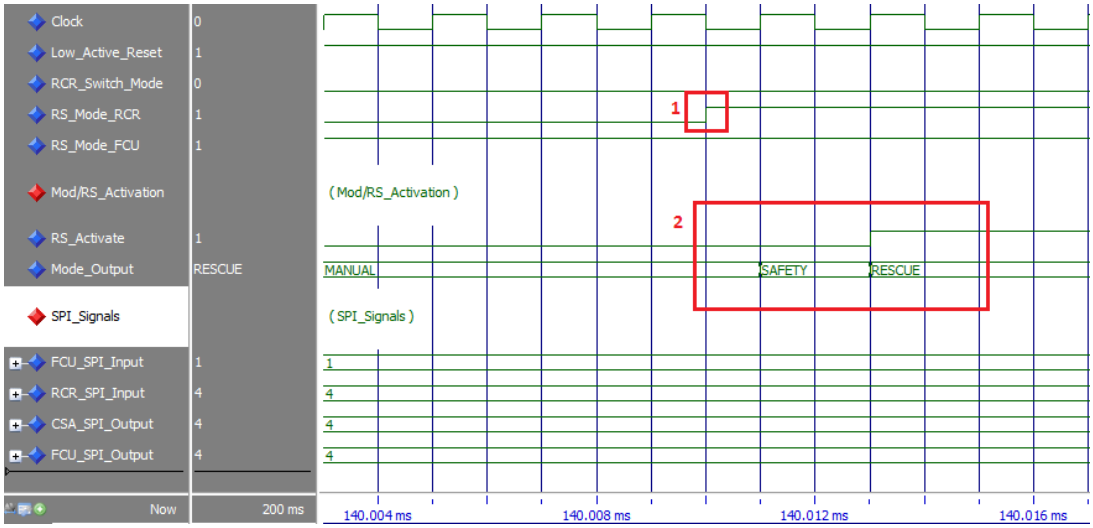

Abbildung 5.7.: Auszug aus dem Simulationsergebnis des Mode Switch Moduls. Detailansicht des Wechsels vom Zustand „Safety“ zum Zustand „Rescue“.

Die Abbildungen 5.6 und 5.7 zeigen, dass der implementierte Automat entsprechend der Anforderungen arbeitet. Auch die Steuersignale (vom Piloten oder der FCU) werden korrekt, je nach Zustand, weitergegeben. Außerdem ist es nicht möglich, das nach dem Auslösen des RS, das Signal wieder zurückgenommen wird.

RS Controller

Der RS Controller ist ein Modul mit einer Vielzahl von Ein- und Ausgängen. Der Abbildung 5.8 kann ein Simulationsergebnis des Moduls entnommen werden. Dabei kann in der Markierung 1 ein FCU Failure (Signal RS_Mode_FCU von ’0’ auf ’1’) gesehen werden, der durch das direkte Aktivieren des RS (Signal FCU_Activate_RS von ’0’ auf ’1’) und einen direkten Failure der FCU (FCU_Failure von ’0’ auf ’1’) ausgelöst wurde.
In der Markierung 2 kann ein RCR Failure (Signal RS_Mode_RCR von ’0’ auf ’1’) gesehen werden, der durch das direkte Aktivieren des RS (Signal RCR_Activate_RS von ’0’ auf ’1’) und einen direkten Failure der P2S (P2S_Failure von ’0’ auf ’1’) ausgelöst wurde.
In der Markierung 3 wiederum wird ein Ausfall beider Systeme simuliert. Die beiden Signale RS_Mode_RCR und RS_Mode_FCU wechseln von ’0’ auf ’1’. Daran kann der Ausfall beider erkannt werden. Der Auslöser ist ein direkter Failure des CSA (CSA_Failure von ’0’ auf ’1’).

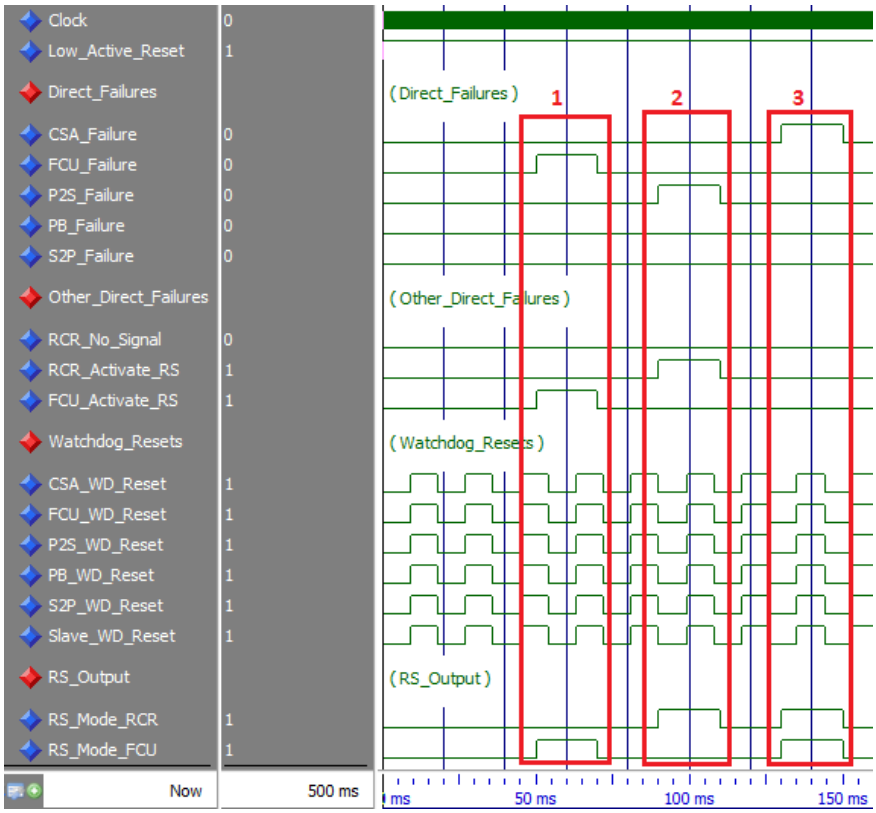

Abbildung 5.8.: Auszug aus dem Simulationsergebnis des RS Controllers.

Anhand der Abbildung 5.8 kann ein Teil des umgesetzten Entwurfs nachvollzogen werden.

Speziell die Implementierung der kombinatorischen Logik aus den Wahrheitstabellen. Außerdem laufen hier alle Safety-Links der Schnittstellendefinition zusammen und werden entsprechend ausgewertet.

Master CPLD

In der Abbildung 5.9 ist eines der Simulationsergebnisse des Master CPLD zu sehen. In diesem wird nicht das RS ausgelöst, wie der Markierung 1 entnommen werden kann (RS_Activation durchgehen auf ’0’). Der Markierung 2 kann ein normaler Wechsel vom Zustand „Manual“ zum Zustand „Stabilized“/„Autonomous“ entnommen werden. Dieser Wechsel wird durch den Signalwechsel von RCR_Mode von ’0’ auf ’1’ ausgelöst. In der Markierung 3 kann der Wechsel in den Zustand „Safety“ gesehen werden, ausgelöst durch einen FCU Failures (FCU_Failure von ’0’ auf ’1’). Auch nach der Rücknahme des FCU_Failure von ’1’ auf ’0’ bleibt die Komponente im Zustand „Safety“. Erst nachdem das Signal RCR_Mode auf eine logische ’0’ zurückgesetzt wird, wechselt das System in den Zustand „Manual“ (Markierung 4). Innerhalb der Markierung 3 kann zudem erkannt werden, dass der Pilot automatisch als primäre Steuereinheit, die Gewalt über den Träger zurück erhält (CSA_SPI_Output ist gleich RCR_SPI_Input).

In der Abbildung 5.10 ist ein weiteres Simulationsergebnis des Master CPLD zu sehen. In diesem wird zunächst ein Wechsel vom Zustand „Manual“ in den Zustand „Stabilized“/„Autonomous“ gezeigt (Markierung 1). Der Zustandswechsel wird durch einen Signalwechsel von RCR_Mode von ’0’ auf ’1’ ausgelöst. Im weiteren Verlauf bricht im Anschluss das gesamte System zusammen (Markierung 2), durch einen Ausfall des RCR Signals (RCR_No_Signal von ’0’ auf ’1’). Dadurch wechselt das System in den Zustand „Rescue“ und löst das RS aus (RS_Activation wechselt von ’0’ auf ’1’). Es wird hier gezeigt, dass der Pilot die höchste Priorität darstellt.

Slave CPLD

Der Abbildung 5.11 kann das Simulationsergebnis des Slave CPLD entnommen werden. Das Slave CPLD hat keine detaillierten Anforderungen, sondern dient lediglich der Überwachung des Master CPLD. Anhand der Markierung 1 kann die zeitliche Überschreitung des Master WDs entnommen werden. Zu sehen ist dies anhand des Signals Master_Reset_Watchdog. Dieses sollte unter normalen Umständen regelmäßig das Signal wechseln, was innerhalb der Markierung 1 nicht passiert. Dadurch wird das RS ausgelöst (Activate_RS wechselt von ’0’ auf ’1’). Im weiteren Verlauf wird gezeigt, dass die Rücknahme des RS Signals nicht möglich ist.

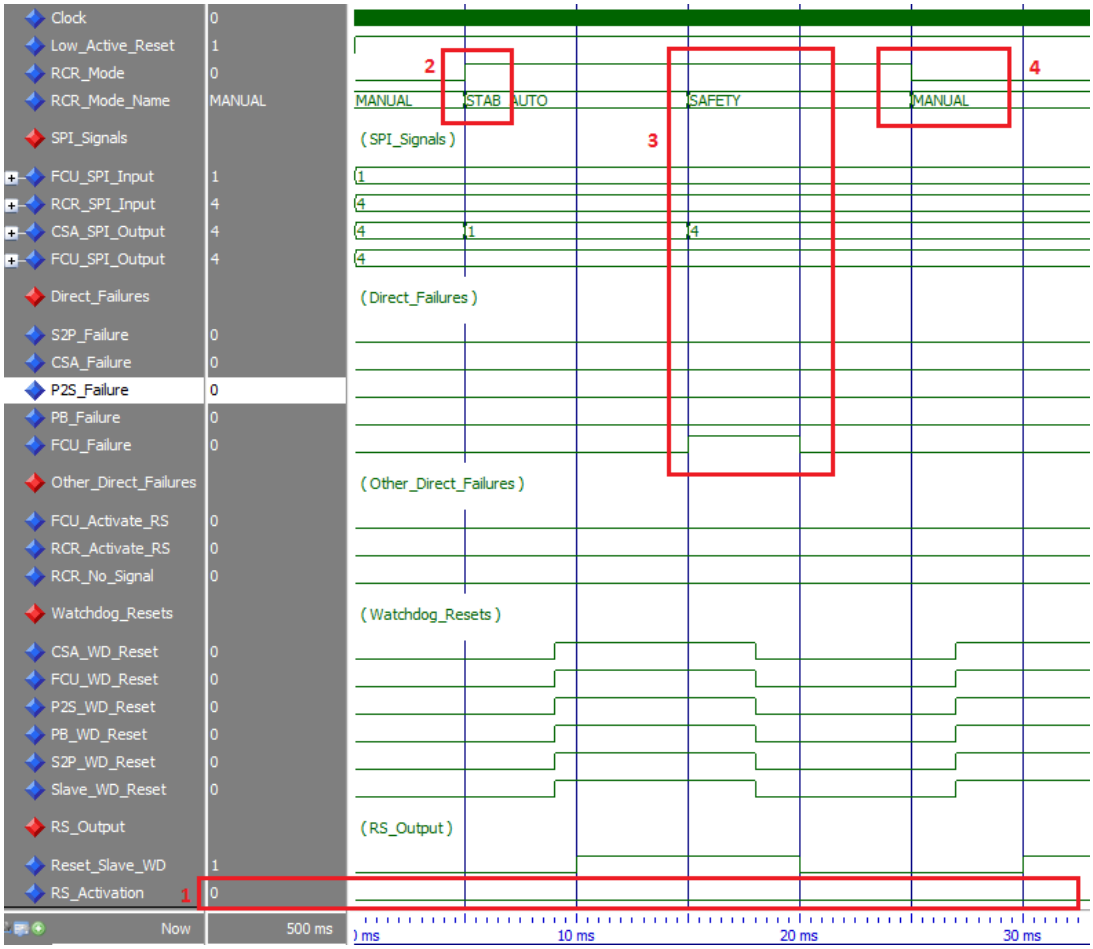

Abbildung 5.9.: Auszug aus dem Simulationsergebnis des Master CPLD. Ansicht eines Wechsels aus dem Zustand „Saftey“ in den Zustand „Manual“ zurück.

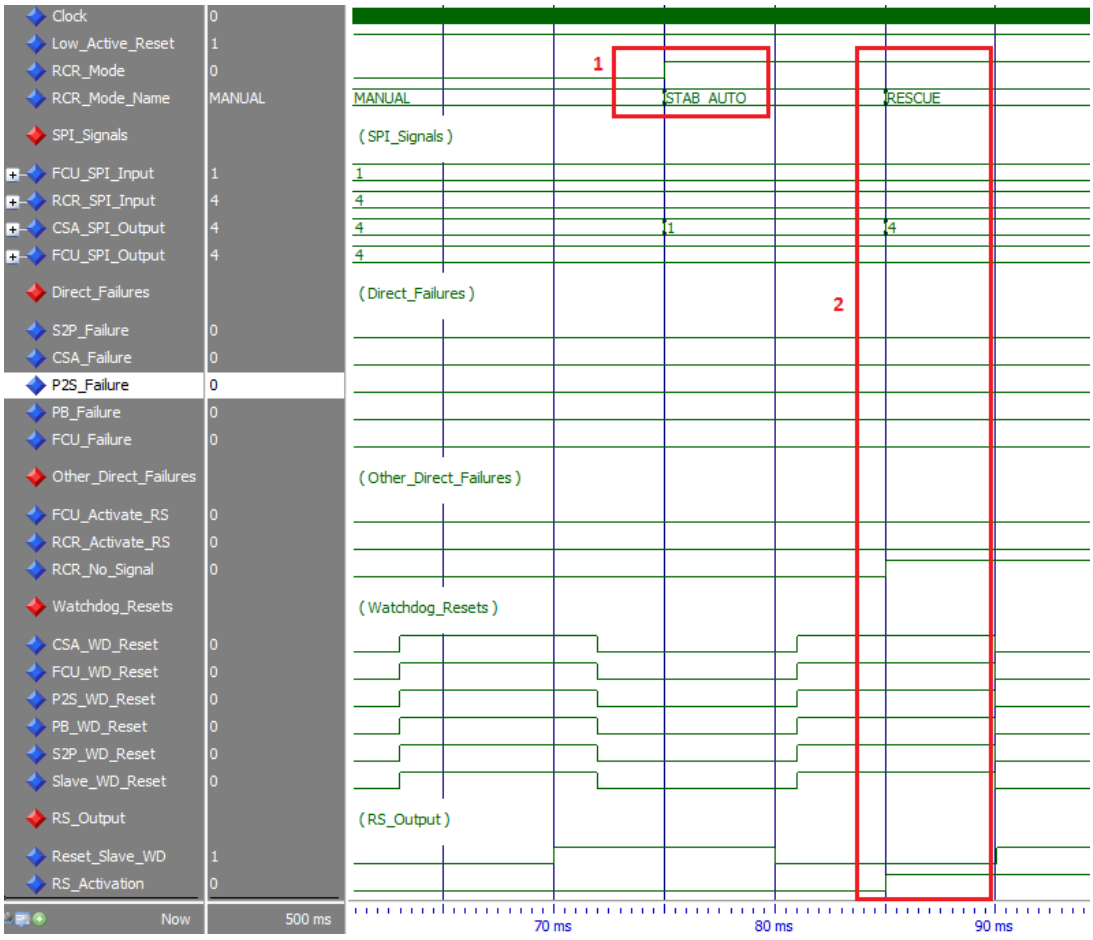

Abbildung 5.10.: Auszug aus dem Simulationsergebnis des Master CPLD. Ansicht eines Wechsels in den Zustand „Rescue“.

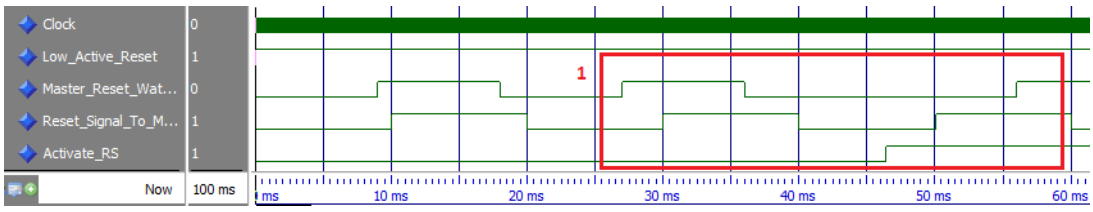

Abbildung 5.11.: Auszug aus dem Simulationsergebnis des Slave CPLD inklusive auslösen des Rescue Systems.

5.6. Zusammenfassung

Den Simulationsergebnissen kann entnommen werden, dass der Entwurf und der Lösungsvorschlag im vollem Umfang den Anforderungen entspricht sowie der gewählten Architektur. Durch die hier beschriebene Safety-Konzept ist es nun möglich, die Systemsicherheit deutlich zu erhöhen. Der Ausfall der als nicht-tolerierbar gekennzeichneten Komponenten der Risiko-Analyse aus Kapitel 4 kann nun als tolerierbar eingestuft werden. Nun müssen die Komponenten nur noch sicherstellen, dass sie sich entsprechend an die hier gestellten Safety-Anforderungen halten (Tabelle 5.2). Die letzte Forderung muss auf irgendeine Art und Weise sichergestellt werden. Ein erster Ansatz, dies sicherzustellen, wäre die Definition passender Testfälle für die Komponenten. Eine weitere Möglichkeit wäre es, eine prüfende Instanz einzuführen. Diese könnte solche Anforderungen anhand von Checklisten und Reviews prüfen. Dies sind nur erste Ideen, die als Anregung dienen. Es ist nicht Ziel eine Lösung dieser Problematik zu finden.

6. Aussicht und Zusammenfassung

In diesem Kapitel werden bestehende Probleme und weiterführende Aufgaben beschrieben. Außerdem wird auf bereits laufende oder noch benötigte Ausarbeitungen eingegangen. Am Ende wird ein kurzes Fazit gezogen und die Aussichten dieser Arbeit beschrieben.

6.1. Umsetzung des Systementwurfs

Ein nächstes Ziel ist die Umsetzung des Systementwurfs. Dazu sollte der erstellte VHDL-Code, aus Anhang A, noch ausführlich auf den CPLDs getestet werden. Die entwickelten Komponenten müssen einem Hardware-in-the-Loop (HIL) Test unterzogen werden. Des Weiteren muss ein Schaltplan sowie alle erforderlichen schematischen Zeichnungen entworfen werden. Auf Basis des Entwurfs kann ein eigenes Save-Life-Board umgesetzt werden. Ein solches Board sollte im Anschluss ebenfalls mit allen Komponenten in einem „großen“ HIL-Test ausgiebig geprüft werden. Alle Grundlagen für die nächsten Schritte können entsprechend dieser Arbeit entnommen und verwendet werden. Über konkrete Testfälle wurden sich bisher noch keine Gedanken gemacht. Es wäre auf jeden Fall sinnvoll die Tests, die bereits in der Simulation nachgewiesen wurden, auch auf der Hardware nach zu vollziehen. Dies wäre eine erste Idee für den HIL Test der SLB Komponente alleine. Da das gesamte System noch nicht besteht, könnte über den „großen“ HIL Test nur spekuliert werden.
Zwei wichtige Punkte, die bei der Hardware-Umsetzung eine Rolle spielen, wurden bereits in Kapitel 5.4 erwähnt. Es sollte zum einen geprüft werden, ob für die Signaleingänge eine Schaltung entworfen werden sollte, die prüft, ob ein Signal eine bestimmte Zeitspanne anliegt. Nur ein Signal, welches dieser Anforderung entspricht, sollte als gültig erachtet. Außerdem sollte über die Erweiterung des WD nachgedacht werden. Eine bessere Variante wäre der Window WD.

6.2. Ausstehende Arbeiten

Parallel zu dieser Arbeit entstehen zwei weitere und nennenswerte Bachelorarbeiten. Zum einen die Bachelorarbeit von *Hagen Hasberg - Ein Testkonzept für Flugregler* [11] sowie die

Arbeit von *Alexander Rohrer - Softwarearchitektur für Airborne Embedded Systems* [20]. Beide sind im gleichen Umfeld entstanden und sollen Lösungen zu den Problemen, an denen im PO AES gearbeitet wird, beitragen.

Das in Kapitel 1 vorgestellte System existiert momentan nicht komplett, sondern ist zum Teil nur spezifiziert und anhand von Anforderungen beschrieben. Die einzelnen Komponenten werden im PO AES durch mehrere studentische Gruppen entwickelt und vorhandene werden erweitert. Das hier entworfene SLB ist für das Gesamtsystem gedacht und ist daher auf die Fertigstellung der einzelnen Komponenten angewiesen. Die hier festgelegten Safety-Anforderungen aus Tabelle 5.2 müssen bei dem Entwurf oder der Weiterentwicklung der Komponenten entsprechend berücksichtigt werden. Des Weiteren wird vom BWB-Team das RS entworfen und entwickelt. Auch dieses muss sich an die Safety-Anforderungen aus Tabelle 5.2 halten. Das aktuelle Ziel im PO AES ist es somit, die Komponenten des Systems zu entwickeln. Da das System, so wie es in dieser Arbeit beschrieben ist, nicht zwingend am Ende auch entwickelt wird, ist es möglich, das entsprechende Anpassungen an dem SLB Entwurf gemacht werden müssen. Die grundlegende Idee, der hier beschriebenen Überwachung, kann auch auf ein ähnliches System übertragen werden. Auch das Umschalten der Modes kann auf eine andere Art der Implementierung übertragen werden. Je nach Umfang der Systemänderungen müssen größere oder kleinere Anpassungen an dem Entwurf durchgeführt werden. Hauptsächlich müssen die Schnittstellen an ein alternatives System angepasst werden.

Außerdem sollte über akustische und/oder optische Signalgeber nachgedacht werden. Dadurch soll der Pilot und andere Beteiligte, den aktuellen Zustand des Trägers am Boden besser erkennen können. Dies sollte bedacht werden, da dadurch die beteiligten Personen die Situationen besser einschätzen können und wesentlich besser auf eine Zustandsänderung reagieren können. Auch in der Arbeit von Arne Richter [19, S. 31] wurden bereits akustische Signalgeber vorgeschlagen.

6.3. Aussicht und Fazit

Im Großen und Ganzen kann gesagt werden, dass die Arbeit erfolgreich abgeschlossen werden konnte. Das gesteckte Ziel konnte erreicht werden und ein erstes Safety-Konzept wurde entwickelt und geprüft. Vermutlich ist dies noch nicht die finale Version, da ein Großteil der benötigten Komponenten noch nicht entwickelt wurde und es an der nötigen Infrastruktur mangelt. Aber dafür bilden die Grundlagen, Analysen und der Entwurf einen guten Ausgangspunkt für weitere Arbeiten. Außerdem konnten viele Probleme bereits identifiziert und analysiert werden. Auf dieser Basis kann die Entwicklung sehr gut fortgesetzt werden.

Auch ich bin mit meiner Arbeit im Großen und Ganzen zufrieden. Ich habe durch diese Arbeit viele neue Erfahrungen sammeln können und mich in dem Bereich der Systemsicherheit wei-

terbilden können. Ein Problem, welches ich hatte, war es in das Gebiet der Sicherheit hinein zu finden. Gerade die Bewertung und Analyse der Systeme ist mir schwergefallen. Dafür konnte ich umso mehr neue Erfahrungen sammeln. Auch das Schreiben einer solchen Arbeit war kein leichtes Unterfangen, da es mir deutlich an Übung und Erfahrung mangelte. Ich denke aber auch das ich mit dem „Rundumschlag“ sehr viel neues Wissen aufbauen konnte. Ich denke, das ich es geschafft habe ein sehr theoretisches Grundlagenthema im praktischen Einsatz zu präsentieren.
Als Grundlage für mein weiterführendes Studium kommt mir mein neues Wissen sehr entgegen. Mein Schwerpunkt soll weiterhin im Umfeld der Sicherheit liegen. Meine Interessen liegen aber ebenfalls im Systems Engineering und seit dieser Arbeit beim Entwurf von Rekonfigurierbaren Systemen. Ich werde versuchen diese verschiedenen Teilgebiete in meinen nächsten Projekten zu verbinden uns zu vertiefen.

Literaturverzeichnis

[1] *Guidelines for Chemical Process Quantitative Risk Analysis*. 2. Auflage. American Institute of Chemical Engineers, 1999. – ISBN 978–0816907205

[2] ADAMS, Douglas: *Das Leben, das Universum und der ganze Rest*. 12. Auflage. Wilhelm Heyne Verlag, 2012. – ISBN 978–3–453–14605–1

[3] AXELROD, C. W.: *Engineering Safe and Secure Software Systems*. Artech House, 2013. – ISBN 978–1–60807–472–3

[4] BOZZANO, Marco ; VILLAFIORITA, Adolfo: *Design and Safety Assessment of Critical Systems*. Auerbach Publications, 2011. – ISBN 978–1–4398–0331–8

[5] BRISSET, Pascal ; BRONZ, Murat ; GORRAZ, Michel ; TYLER, Jeremy: *YAP (Yet Another Paparazzi)*. ENAC, 2008

[6] BRISSET, Pascal ; DROUIN, Antoine ; GORRAZ, Michel ; HUARD, Pierre-Selim ; TYLER, Jeremy: *The Paparazzi Solution*. ENAC, 2006

[7] BÜSCHER, René ; HASBERG, Hagen: *FCC Hardware-Architektur - Übersicht der Hardware-Architektur des Airborne Embedded System*. Mai 2014. – Projekt - Airborne Embedded System [nicht veröffentlicht]

[8] *Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Software für Eisenbahnsteuerungs- und Überwachungssysteme; Deutsche Fassung EN50128:2011*. 2012 (DIN EN 50128)

[9] *Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme - Sicherheitsrelevante elektronische Systeme für Signaltechnik; Deutsche Fassung EN50129:2003*. 2003 (DIN EN 50129)

[10] GRIESSNIG, G. ; MADER, R. ; STEGER, C. ; WEISS, R.: A CPLD-based Safety Concept for Industrial Applications. In: *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, 2010, S. 3027–3032

[11] HASBERG, Hagen: *Ein Testkonzept für Flugregler*. vrs. 2014. – [Abgabe der Arbeit noch nicht erfolgt]

[12] *Funktionale Sicherheit sicherheitsbezogener elektrisch/ elektronischer/ programmierbaraer elektronischer Systeme - Teil 0: Funktionale Sicherheit und die IEC 61508 (IEC/TR 61508-0:2005)*. 2005 (DIN EN 61508 Beiblatt 1)

[13] *Funktionale Sicherheit sicherheitsbezogener elektrisch/ elektronischer/ programmierbaraer elektronischer Systeme - Teil 1:Allgemeine Anforderungen (IEC 61508-1:2010); Deutsche Fassung EN 61508-1:2010*. 2011 (DIN EN 61508 Teil 1)

[14] *Funktionale Sicherheit sicherheitsbezogener elektrisch/ elektronischer/ programmierbaraer elektronischer Systeme - Teil 5: Beispiele zur Ermittlung der Stufe der Sicherheitsintegrität (safety intigrity level) (IEC 61508-5:2010); Deutsche Fassung EN 61508-5:2010*. 2011 (DIN EN 61508 Teil 5)

[15] JOHNSON, Stephen B. ; GORMLEY, Thomas J. ; KESSLER, Seth S. ; MOTT, Charles D. ; PATTERSON-HINE, Ann ; RECHARD, Karl M. ; JR., Philip A. S.: *System Health Management with Aerospace Applications*. John Wiley and Sons, Ltd, 2011. – ISBN 978–0–470–74133–7

[16] KLUSSMANN, Niels ; MALIK, Arnim: *Lexikon der Luftfahrt*. 3. Auflage. Springer-Verlag Berlin Heidelberg, 2012. – ISBN 978–3–642–22500–0

[17] MENSEN, Heinrich: *Handbuch der Luftfahrt*. 2. Auflage. Springer Vieweg, 2013. – ISBN 978–3–642–34402–2

[18] PHILIP A. SCANDURA, Jr. ; CHRISTENSEN, Michael ; LUTZ, Daniel ; BIRD, Gary: Commercial Aviation Health Management. In: *System Health Management with Aerospace Applications*. John Wiley and Sons, Ltd, 2011. – ISBN 978–0–470–74133–7

[19] RICHTER, Arne M.: *Konzept und Einführung von Safety-Analysen bei Mikrocontrollerbasierten Anwendungen in UAVs*. September 2013

[20] ROHRER, Alexander: *Softwarearchitektur für Airborne Embedded Systems*. Mai 2014. – [nicht veröffentlicht]

[21] ROOS, Hans-Leo: *Funktional Sicherheit im Automobil - ISO 26262, Systemengineering auf Basis eines Sicherheitslebenszyklus und bewährter Managementsysteme*. Carl Hanser Verlag München Wien, 2014. – ISBN 978–3–446–43632–9

[22] ROTH, Michael ; LIGGESMEYER, Peter: *Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees*. SAFECOMP 2013 - Workshop DECS, 2013

[23] SMITH, David J. ; SIMPSON, Kenneth G. L.: *Safety Critical Systems Handbook - A Straightfoward Guide to Functional Safety, IEC 61508 (2010 Edition) and Related Standards*. 3. Auflage. Butterworth-Heinemann, 2011. – ISBN 978–0–08–096781–3

[24] STAMATELATOS, Michael ; VESELY, William ; DUGAN, Joanne ; FRAGOLA, Joseph ; III, Joseph M. ; RAILSBACK, Jan: *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, 2002

[25] STEINER, Max ; LIGGESMEYER, Peter: *Combination of Safety and Security Analysis - Finding Security Problems That Threaten The Safety of a System*. SAFECOMP 2013 - Workshop DECS, 2013

[26] STOREY, Neil: *Safety-Critical Computer Systems*. Pearson Education Limited, 1996. – ISBN 0–201–42787–7

[27] VERHULST, Eric ; VARA, Jose L. I. ; SPUTH, Bernhard H. ; FLORIO, Vincenzo de: *ARRL: A Criterion for Composable Safety and Systems Engineering*. SAFECOMP 2013 - Workshop SASSUR, 2013

[28] WATSON, Michael D. ; VARNAVAS, Kosta ; PATRICK, Clint ; HODGE, Ron ; BYINGTON, Carl S. ; CHAU, Savio ; BAROTH, Edmund C.: Avionics Health Management. In: *System Health Management with Aerospace Applications*. John Wiley and Sons, Ltd, 2011. – ISBN 978–0–470–74133–7

[29] ZIMPFER, Douglas J.: Flight Control Health Management. In: *System Health Management with Aerospace Applications*. John Wiley and Sons, Ltd, 2011. – ISBN 978–0–470–74133–7

[30] *AURIX^{TM} - Safety joins Performance*. http://www.infineon.com/cms/en/product/channel.html?channel=db3a30433727a44301372b2eefbb48d9, . – [Online; besucht 27. April 2014]

[31] *CoolRunner-II CPLDs*. http://www.xilinx.com/products/silicon-devices/cpld/coolrunner-ii/index.htm, . – [Online; besucht 27. April 2014]

[32] *Flight Mode*. http://pixhawk.org/users/system_modes?s[]=mode, . – [Online; besucht 3. Februar 2014]

[33] *PX4FMU Autopilot / Flight Management Unit*. http://pixhawk.org/modules/px4fmu, . – [Online; besucht 3. Februar 2014]

[34] *PX4IO Airplane/Rover Servo and I/O Module*. http://pixhawk.org/modules/px4io, . – [Online; besucht 3. Februar 2014]

[35] *Was ist PAAG / HAZOP*. http://www.tuev-sued.de/anlagen_bau_industrietechnik/technikfelder/dampf-_und_drucktechnik/stoerfallvorsorge/paag_hazop/was_ist_paag_hazop, . – [Online; besucht 14. Mai 2014]

[36] *The Controller Concept of the NG*. http://ng.uavp.ch/Documentation/Controllers, Juni 2012. – [Online; besucht 3. Februar 2014]

[37] *Notify Plugin*. http://wiki.openpilot.org/display/Doc/Notify+Plugin, November 2012. – [Online; besucht 3. Februar 2014]

[38] *OpenPilot Firmware Architecture*. http://wiki.openpilot.org/display/Doc/OpenPilot+Firmware+Architecture, Februar 2012. – [Online; besucht 3. Februar 2014]

[39] *Signals*. http://ng.uavp.ch/Documentation/Signals, Dezember 2012. – [Online; besucht 3. Februar 2014]

[40] *AutopilotModes*. http://wiki.paparazziuav.org/wiki/AutopilotModes, September 2013. – [Online; besucht 3. Februar 2014]

[41] *Failsafe*. http://wiki.paparazziuav.org/wiki/Failsafe#RC_link_failure_while_manual_%or_AUTO1, September 2013. – [Online; besucht 3. Februar 2014]

[42] *System Health Plugin*. http://wiki.openpilot.org/display/Doc/System+Health+Plugin, Dezember 2013. – [Online; besucht 3. Februar 2014]

[43] *Duden Philosophie*. http://www.duden.de/node/656105/revisions/1294685/view, 2014. – [Online; besucht 3. Februar 2014]

[44] *Flight Mode Switch Settings*. http://wiki.openpilot.org/display/Doc/Flight+Mode+Switch+Settings, Januar 2014. – [Online; besucht 3. Februar 2014]

[45] *Murphy's Law*. http://en.wikipedia.org/wiki/Murphy%27s_law, 2014. – [Online; besucht 3. Februar 2014]

A. Anhang

A.1. Inhalt der CD-ROM

Diesem Abschnitt kann eine kurze Übersicht des Inhaltes der CD-ROM entnommen werden. Alle verwendeten Programme und dessen Versionen können dem Anhang A.2 entnommen werden.

- Bachelorarbeit im PDF-Format
- LaTeX Quelldateien und BibTeX Verzeichnis
- Grafiken im Quellformat
- Enterprise Architect Projekt
- ModelSim Projekt mit allen VHDL und TCL Quelldateien
- Zwei ISE Projekte mit allen VHDL Quelldateien für die Synthese

Im Hauptordner liegt als Erste und als einzige Datei die Bachelorarbeit. Alle zusätzlichen Daten können im Ordner Anhang gefunden werden. Dort sind sie, wie in der oberen Aufzählung beschrieben, untergeordnet. Zusätzlich zum PDF-Format sind ebenfalls alle LaTeX und BibTeX Quelldaten vorhanden. Dem Enterprise Architect Projekt kann ausschließlich die Abbildung 3.1 entnommen werden. Alle anderen Abbildungen liegen als PNG-Datei und, wenn vorhanden, als VSD-Datei vor. Alle Quelldateien sind mit Kommentaren an den notwendigen Stellen versehen. Außerdem wurden zum Teil Kommentare eingefügt, durch die entsprechend eine Steigerung der Übersicht erreicht werden soll. Das ModelSim Projekt beinhaltet alle VHDL Dateien, inklusive der Testbenches und Do-Dateien (TCL Skript) für die Konfiguration der Simulation. Innerhalb dieser Projektstruktur können alle, für die Implementierung und dem Testen wichtigen Dateien, gefunden werden. Als Letztes sind noch zwei ISE Projekte vorhanden. Beide Projekte wurden für ein Synthese-Test der CPLDs erstellt. Die Projekte referenzieren die Quelldateien, die innerhalb des ModelSim Projekt liegen. Ebenfalls sind die vollständigen UCF-Dateien enthalten. Diese können innerhalb des ModelSim Projekts gefunden werden.

A.2. Verwendete Programme

In diesem Abschnitt werden alle verwendeten Programme in der Tabelle A.1 erfasst. Für jedes Programm ist der genaue Name, die Version sowie eine kurze Beschreibung angegeben, wozu das Programm genau verwendet wurde.

Programm	Version	Beschreibung
MiKTeX	2.9	Enthält und verwaltet die TeXPakete. Beinhaltet die verschiedenen benötigten Interpreter.
TeXstudio	2.6.2	Editor für TeXDateien.
Microsoft Visio Professional 2010	14.0	Erstellung von verschiedensten Diagrammen.
ModelSim PE Student Edition	10.3a	Hardwaresimulation.
ISE Design Suite	14.6	Hardwaresynthese für Xilinx Produkte, FPGAs und CPLDs.
Enterprise Architect	10.0	Unter anderem zum Erstellen von UML- und SysML-Diagrammen.

Tabelle A.1.: Tabellarische Auflistung der Programme, die zum Erstellen der Arbeit benötigt wurden.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 23. Mai 2014	
Ort, Datum	Unterschrift