Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Entwicklung eines Konzeptes zur Implementierung von Regelalgorithmen im Rahmen einer STPA und dessen Umsetzung in XSTAMPP 4.0

Tobias Heck

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Wolfgang Fechner
Beginn am:	8. Mai 2019
Beendet am:	8. November 2019

Kurzfassung

Die starke Zunahme von komplexer Software in modernen Systemen hat die Entwicklung neuer Analysetechniken für Unfallkausalität notwendig gemacht. Eine dieser Techniken ist *STPA*, welche Systeme anhand von Regelkreisen darstellt. Innerhalb eines Regelkreises bestimmt der Regelalgorithmus, unter welchen Bedingungen das Senden eines Regelsignals stattfindet. Für die Analyse des Regelalgorithmus kann es hilfreich sein, diesen explizit zu modellieren. Hierfür existieren zahlreiche Modellierungskonzepte, von denen einige in speziell für STPA entwickelten Programmen implementiert sind. Diese Arbeit präsentiert das Ergebnis einer umfangreichen Recherche zu diesen Konzepten. In diesem Zusammenhang wird auch ein Vergleich anhand der vier Kriterien *Verständlichkeit*, *Flexibilität*, *Skalierbarkeit* und *Erweiterbarkeit* durchgeführt. Basierend auf dem Ergebnis des Vergleichs und unter Berücksichtigung der Anwendbarkeit bei Menschen wird ein Konzept vorgestellt, das sich an Formeln orientiert. Dieses ist so konzipiert, dass eine automatische Generierung von Funktionsgraphen möglich ist, welche die Inhalte kontinuierlicher Regelsignale in Abhängigkeit der relevanten Prozessvariablen darstellen. Zusätzlich legt die Arbeit einen Vorschlag dar, wie das Modellierungskonzept auf Aktoren übertragen werden kann. Beide Konzepte werden zusammen mit einer Generierungsfunktion für Graphen in Version 4.1 der STPA-Software *XSTAMPP* integriert und dokumentiert. Das Eingabefeld der Formeln besitzt dabei eine automatische Vervollständigung für Prozessvariablennamen und eine Syntaxprüfung. Bei den Graphen ist auch eine partielle Darstellung der Formeln möglich. Abschließend findet eine STPA-Analyse zu einem Anlegemanöver einer Fähre statt, in deren Rahmen die Implementierung des neuen Konzepts erprobt wird. Die Analyse des Regelalgorithmus wird zunächst ohne explizite Modellierung durchgeführt, und anschließend unter Zuhilfenahme einer solchen wiederholt. Dabei ist eine signifikante Steigerung der identifizierten Verlustszenarien zu beobachten, wodurch gezeigt ist, dass zumindest Fälle existieren, in denen eine Modellierung des Regelalgorithmus mit dem hier vorgestellten Konzept einen positiven Effekt auf die Identifikation der Verlustszenarien hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

Abkürzungen

ACC Adaptive Cruise Control. 40

AIS Automatic Identification System. 84

API Application Programming Interface. 29

A-STPA Automated tool support for STPA. 15, 33

ATP Automatic Train Protection. 41

AVH Automatic Vehicle Hold. 40

C Constraint. 89, 90, 91

DNF Disjunktive Normalform. 44, 55

ESS Engine Stop-Start. 40

FMEA Failure Mode and Effect Analysis. 19

FTA Fault-Tree Analysis. 19

GPS Global Positioning System. 63, 84

H Hazard. 25, 27, 62, 64, 82, 87

HAZOP Harzard and Operability Analysis. 19

HTML HyperText Markup Language. 30

HTTP Hypertext Transfer Protocol. 30, 72

IOC Inversion of Control. 30

IPA Information-technology Promotion Agentur Japan. 35

ISO International Organization for Standardization. 53

L Loss. 62

R Responsibility. 62

RCP Rich Client Platform. 29

REST Representational State Transfer. 30, 72

RM Studio Risk Management Studio. 35

SafetyHAT Safety Hazard Analysis Tool. 15, 36

SAHRA STPA based Hazard and Risk Analysis. 34

SC Safety Constraint. 87

SHERPA Systematic Human Error Reduction and Prediction Approach. 48

SpecTRM Specification Tools and Requirements Methodology. 33

SQL Structured Query Language. 31

STAMP Systems Theoretic Accident Model and Processes. 15, 18, 19, 20, 21, 23, 25, 29, 31, 38, 46, 47

STPA System-Theoretic Process Analyis. 15, 16, 17, 18, 19, 23, 24, 25, 29, 30, 31, 33, 34, 35, 36, 40, 46, 47, 48, 49, 50, 53, 58, 59, 61, 62, 65, 66, 68, 71, 76, 79, 81, 86, 98, 99

STPA-RC System-Theoretic Process Analyis Refined Controller-analysis. 49

THERP Technique for Human Error Rate Prediction. 48

TRACEr Technique for the Retrospective and Predictive Analysis of Cognitive Errors. 48

UCA Unsafe Control Action. 25, 27, 64, 87, 91

UML Unified Modeling Language. 43

XSTAMPP eXtensible STAMP Platform. 7, 15, 17, 18, 19, 29, 30, 31, 33, 53, 54, 71, 75, 76, 77, 78, 79, 81, 83, 94, 98, 99, 100

1 Einleitung

Der schnelle technische Fortschritt in den vergangenen Jahren hat dazu geführt, dass der Komplexitätsgrad von Autos, Flugzeugen, Maschinen und anderen Systemen, mit denen wir auf täglicher Basis interagieren, stark gestiegen ist. Bei der Verwendung dieser Systeme ist essentiell, dass zu keinem Zeitpunkt das Risiko einer körperlichen Verletzung oder anderweitigen Schädigung besteht: Ein autonomes Fahrzeug darf nicht ungebremst mit dem Vordermann kollidieren, genauso wenig wie ein Kernreaktor bei Überhitzung radioaktive Strahlung freigeben darf. Das Vorhersehen potentieller Unfallursachen und das Entwerfen von entsprechenden Gegenmaßnahmen ist ein unverzichtbarer Schritt bei Design und Bau sicherheitskritischer Systeme. Die Wahrscheinlichkeit, Risiken zu übersehen, steigt jedoch mit zunehmender Komplexität des Systems an.

Leveson erkannte, dass traditionelle Analysetechniken den Herausforderungen moderne Systeme nicht gewachsen sind und stellte 2004 das auf Systemtheorie basierende Unfallkausalitätsmodell *STAMP* [Lev04] und 2011 eine darauf aufbauende Analysemethode namens *STPA* vor [Lev11]. In STAMP wird eine Sichtweise vorgestellt, die sich weniger auf das Verhindern von Unfällen selbst konzentriert, sondern den Fokus stattdessen auf die Erfüllung von Randbedingungen richtet. Dadurch können nicht nur Ausfälle einzelner Komponenten, sondern auch unsichere Interaktionen zwischen Komponenten zur Gefährdung werden. Sicherheit wird in STAMP zu einem Problem ausreichender Kontrolle. STPA ist eine Analysemethode, die durch ihren Top-Down-Ansatz auch bei großen und komplexen Systemen problemlos angewendet werden kann. Sie unterscheidet den *Regler* von dem geregelten *Prozess*. Der Regler nimmt durch Aktoren auf den Prozess Einfluss und durch Sensoren Informationen zu dem Prozess auf. Sicherheitsrisiken entstehen dort, wo Signale zwischen diesen vier Komponententypen fehlen, irrtümlich gesendet werden oder verspätet/verfrüht sind. Eine schematische Darstellung von STPA ist in Abb. 1.1 zu sehen. Ausführlichere Erklärungen zu STAMP und STPA folgen in Kapitel 2.

2018 wurde ein Handbuch zum Gebrauch von STPA in realen Systemen veröffentlicht. Dieses rückte die *Regelalgorithmen* des Reglers noch weiter in den Vordergrund. Zuvor lag der Schwerpunkt mehr auf deren Prozessmodellen, welche eine interne Repräsentation des tatsächlichen Prozesses sind von Sensoren aktualisiert werden. Der Regler steuert die Aktoren ausgehend von diesem Modell, wobei durch eine Abweichung der internen Repräsentation von der Wirklichkeit ein Sicherheitsrisiko entsteht. Die Regelalgorithmen stellen das Element zwischen Prozessmodell und Aktoren dar. Sie sind zuständig für die Auswertung des Prozessmodells und die darauf basierende Ableitung von Signalen für die Aktoren. Hierdurch ist es möglich, Szenarien zu modellieren, in denen die Informationen des Reglers zu dem Prozess zwar korrekt sind, die an die Aktoren weitergegebenen Signale aber dennoch falsch. Für den Fall dass der Regler ein Softwaresystem ist, würde das Prozessmodell den Speicher, und die Regelalgorithmen den Programmcode darstellen, wodurch falscher Code und falsche Daten getrennt behandelt werden können.

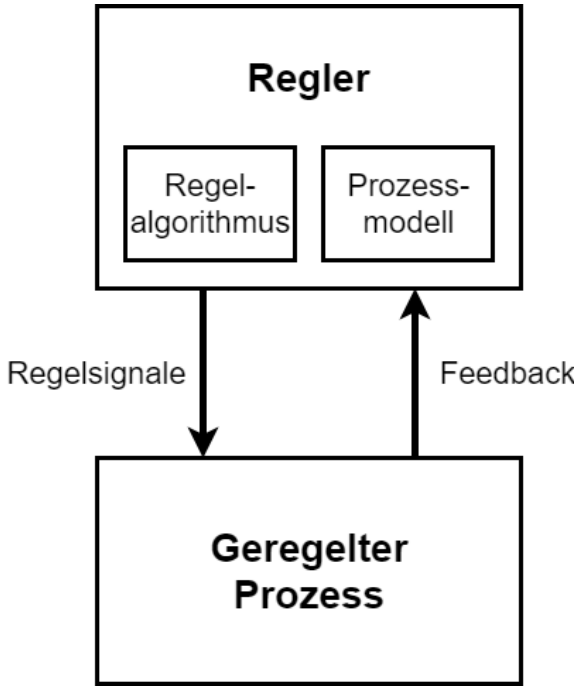


Abbildung 1.1: Regler und geregelter Prozess

Obgleich es kein Problem darstellt, die Risiko-Analyse mit STPA manuell durchzuführen und die resultierenden Tabellen und Diagramme mit Stift und Papier oder allgemein gebräuchlicher Software zu erstellen, hat die Verwendung von speziell auf STPA zugeschnittener Software unzweifelhaft viele Vorteile. So kann etwa die Ableitung der Randbedingungen aus den vorher definierten möglichen Signalfehlern automatisch erfolgen. Es gibt eine Reihe von Programmen, die für STAMP und STPA entwickelt wurden. Allein auf der dritten STAMP-Konferenz 2014 am Masachusetts Institute of Technology wurden drei solcher Programme vorgestellt: *An STPA Tool* [ST14], *SafetyHAT* [Hom14] und *A-STPA* [AW14]. Ein recht gebräuchliches Programm mit über 7.500 Downloads aus 50 Ländern (Stand: Juni 2017) ist *XSTAMPP* [Abd17]. Es wurde im Rahmen der STAMP-Konferenz 2015 von Abdulkhaleq und Wagner als eine auf A-STPA basierende kostenlose Open-Source-Software vorgestellt und wird seitdem beständig weiterentwickelt [AW15a]. Bei der Entwicklung lag der Hauptaugenmerk auf Erweiterbarkeit, um bei Bedarf zusätzliche Funktionalität schnell hinzufügen zu können. Seit Version 4.0 ist XSTAMPP eine Webanwendung, die das parallele Arbeiten mehrerer Sicherheitsingenieure am selben Projekt ermöglicht.

1.1 Problemstellung

Die in den vergangenen Jahren stark an Popularität gewonnene Risikoanalysemethode STPA hat 2018 ein Handbuch erhalten, welches die Bedeutung der Regelalgorithmen unterstreicht. Durch die Aufspaltung des Reglers in Prozessmodell und Algorithmen ist die getrennte Modellierung falscher Entscheidungen aufgrund inkorrekter Informationen und falscher Entscheidungen trotz korrekter Informationen möglich. Es stellt sich jedoch die Frage, wie die Regelalgorithmen im konkreten Anwendungsfall modelliert werden.

1.1.1 Modellierung der Regelalgorithmen

Eine leicht umsetzbare Möglichkeit ist die *Entscheidungstabelle*, da sich hier die Randbedingungen direkt ablesen lassen [JD70]. Allerdings eignen sich Entscheidungstabellen im Allgemeinen schlecht für kontinuierliche Variablen, wie sie z. B. bei der Geschwindigkeitskontrolle eines autonomen Fahrzeugs nötig ist. Eine kontinuierliche Variable kann theoretisch unendlich viele verschiedene Werte annehmen, was die Zahl der Tabelleneinträge unmöglich zu bewältigen macht. Um derartige Variablen auf angemessene Weise zu verarbeiten, ist der Gebrauch von *Formeln* eine naheliegende Alternative. Hier gestaltet sich aber die Ableitung der Randbedingungen schwierig. Es ist klar, dass der Regler keinen von der Formel abweichenden Wert an einen Aktor senden darf, aber unter welchen Umständen das dennoch passieren kann, ist hier weniger intuitiv erkennbar als bei der Entscheidungstabelle. Für die Modellierung der Regelalgorithmen gibt es eine Vielzahl von Möglichkeiten – vorstellbar wäre u. a. auch ein Zustandsautomat. Des Weiteren ist ein Ansatz vorstellbar, der das Modell von dem Anwendungsfall des Reglers oder sogar vom Regler selbst abhängig macht: Da ein Regler nur eine Abstraktion von einer Einheit ist, die einen Prozess kontrolliert, kann er in der Realität sowohl einen Computer als auch einen Menschen oder eine Menschengruppe repräsentieren. Es ist die Möglichkeit in Betracht zu ziehen, dass die Entscheidungsprozesse dieser zwei Reglertypen von unterschiedlichen Modellen profitieren.

1.1.2 Derzeitige Limitierungen von XSTAMPP

XSTAMPP hat sich als wichtiges Werkzeug im Bereich der Risikoanalyse etabliert. In der im April 2019 präsentierten Version 4.0 ist jedoch die Modellierung der Regelalgorithmen noch nicht möglich. Dies ist hinsichtlich der Vorteile einer expliziten Analyse des Entscheidungsverfahrens der Regler als eine Limitierung zu bewerten, die den Benutzer in bestimmten Analyse-Szenarien einschränken kann.

1.2 Beiträge

Diese Forschungsarbeit leistet die folgenden Beiträge:

Review existierender Modellierungskonzepte: Es werden mehrere für STPA vorgesehene Programme sowie eine große Zahl manuell durchgeführter Analysen in Bezug auf die Modellierung von Regelalgorithmen untersucht und strukturiert zusammengefasst. In diesem Zusammenhang werden Potenziale und Defizite aufgezählt. Zusätzlich wird ein Überblick über die in der Literatur dargelegten Vorschläge zur Anpassung der Modelle an menschliche Regler gegeben.

Entwicklung eines Konzepts zur Modellierung von Regelalgorithmen: Die Arbeit stellt eine Modellierungsmethode vor, die auf der sorgfältigen Abwägung mehrerer Qualitätskriterien basiert. Diese wird mithilfe einer formalen Definition präzisiert. Vor- und Nachteile der Methode werden anhand einer teilweisen und einer vollständigen STPA-Analyse aufgezeigt und detailliert erörtert.

Vorschlag zur Übertragung des Modells auf Aktoren: Es wird eine Idee präsentiert, wie das für Regelalgorithmen entwickelte Konzept auf die eine ähnliche Funktion besitzenden Aktoren übertragen werden kann. Dazu werden direkt übertragbare Aspekte sowie notwendige Anpassungen beschrieben.

XSTAMPP Version 4.1: Das fertig ausgearbeitete Modellierungskonzept für Regelalgorithmen wird als Version 4.1 in XSTAMPP implementiert. Dabei werden mehrere Prinzipien befolgt, die sich als positiver Einfluss auf die Wartbarkeit und Erweiterbarkeit von Software bewährt haben. Die Implementierung wird nicht nur in dieser Arbeit, sondern auch in Codekommentaren und der Erklärung des Programms dienenden Textdateien dokumentiert. Das Konzept wird weiterhin auf ein zweites Modul übertragen, mit dem die Aufgaben der Aktoren auf die selbe Weise modelliert werden können.

1.3 Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Hintergrund erklärt dem Leser die Grundlagen von Systemtheorie, dem Unfallkausalitätsmodell STAMP, der Risikoanalysetechnik STPA und des Programms XSTAMPP und bringt diese in Zusammenhang.

Kapitel 3 – State of the Art gibt einen Überblick über aktuelle Vorgehen bei der Modellierung von Regelalgorithmen und fasst aktuelle Erkenntnisse in Bezug auf menschliche Regler zusammen.

Kapitel 4 – Modellierungskonzept für Regelalgorithmen sammelt Kriterien zur Evaluation der Güte von Modellen und wendet diese auf existierende Modellierungsstrategien an. Anschließend stellt das Kapitel ein eigenes Konzept vor und demonstriert dieses anhand eines kurzen Beispiels.

Kapitel 5 – Implementierung in XSTAMPP dokumentiert die Implementierung des Modellierungskonzepts in XSTAMPP 4.1 und erklärt dessen Funktionsweise.

Kapitel 6 – Eine STPA-Analyse mit XSTAMPP 4.1 führt beispielhaft eine umfangreiche STPA-Analyse mit der neuen Version von XSTAMPP durch und beurteilt anhand dieser die Eignung der neu eingeführten Module.

Kapitel 7 – Schlusswort fasst die Arbeit zusammen und gibt Anhaltspunkte für weitergehende Forschung.

2 Hintergrund

Der Fokus dieser Arbeit liegt auf der Entwicklung eines Konzepts zur Modellierung einer Komponente der Risikoanalysetechnik STPA: den Regelalgorithmen. Hierfür soll auch eine Auswertung mehrerer Dokumente zu existierenden Analysen mit STPA durchgeführt werden. Um dieser sachgemäß folgen zu können, ist ein zumindest oberflächliches Verständnis von den zugrunde liegenden Methoden unabdinglich. Folglich wird in diesem Kapitel ein grober Überblick über Systemtheorie, STAMP, und STPA gegeben. Im zweiten Teil dieser Arbeit wird das gefundene Konzept in die Software XSTAMPP integriert. Auch zu dieser Software soll eine grundlegende Einführung gegeben werden.

Das Ziel dieses Kapitels ist dabei nicht, eine vollständige Beschreibung zu bieten – dies würde den Rahmen dieser Arbeit sprengen. Für detailliertere Erläuterungen sei der Leser auf auf diese Themen spezialisierte Literatur verwiesen.

2.1 Systemtheorie

Systemtheorie wurde 1946 von dem Biologen Bertalanffy als eine Reaktion auf den Reduktionismus entwickelt. In seinen Publikationen betont er, dass reale Systeme ihrer Umwelt gegenüber offen seien und ständig mit ihr interagieren. Außerdem sei es möglich, dass sich die Eigenschaften eines Systems durch das Zusammenspiel ihrer Komponenten verändern. Durch diese sog. *Emergenz* sei eine kontinuierliche Evolution zu beobachten. Bertalanffys Theorie verschiebt hierbei den Schwerpunkt weg von den Merkmalen der einzelnen Systemteile hin zu deren Interaktionen und Anordnung im Ganzen. Hierdurch verliert der physikalische Aufbau der Komponenten an Bedeutung, wodurch organische Körper, soziologische Körper und technische Körper vereinheitlicht werden.

In Bertalanffys Theorie werden eine Reihe von Konzepten eingeführt, die sich auch in STAMP und STPA wiederfinden lassen: Ein System wird von seiner Umwelt abgeteilt durch die *System-Umwelt-Grenze*. Informationen, die von der Umwelt in das System gelangen, heißen *Eingabe*, Informationen, die aus dem System in die Umwelt übergehen, *Ausgabe*. Beide Informationsströme beeinflussen den internen *Zustand*. Das System hat ein *Ziel*, das den angestrebten Zustand vorgibt, und durch Steuerung der Ein- und Ausgabe erreicht wird. Hierdurch entsteht eine fundamentale *Hierarchie*. Auch der Begriff *Prozess* ist bereits in der Systemtheorie enthalten [Ber68].

Abb. 2.1 veranschaulicht ein System, wie es konzeptionell der Theorie von Bertalanffy entspricht. Das Ziel wird mit dem Prozesszustand verglichen und je nach Resultat findet eine Anpassung der Eingabe und/oder Ausgabe statt. Beispielsweise könnte das System ein Stausee sein: Bei Überfüllung muss die Schleuse des Damms weiter geöffnet werden um mehr Wasser abfließen zu lassen. Das Ziel entspricht einem idealen Wasserpegel [Mea08].

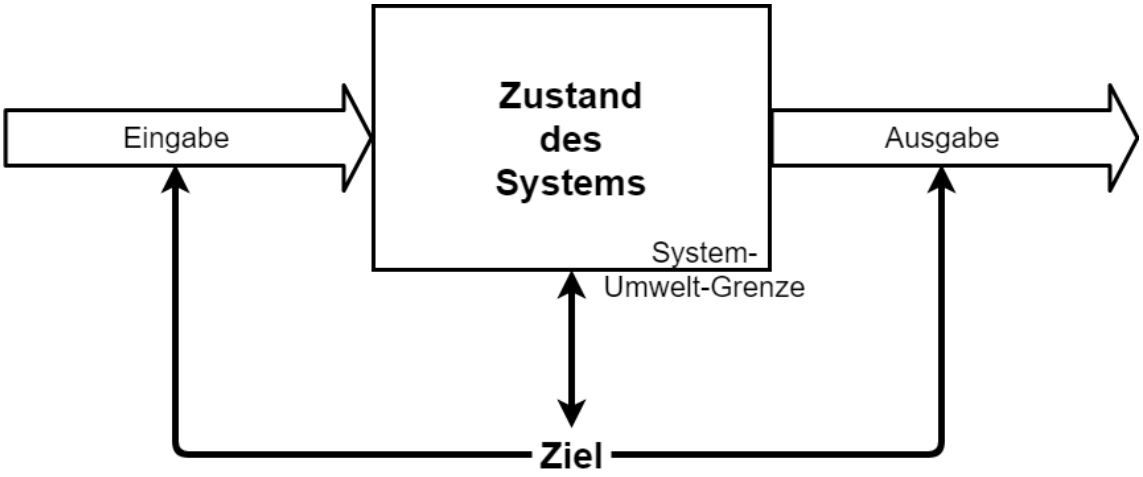


Abbildung 2.1: Modell eines Systems

2.2 STAMP

Zur Zeit der Entstehung der Systemtheorie waren vom Menschen produzierte Systeme noch weitaus weniger komplex als heute. Gegen Ende des 20. Jhd. nahm deren Komplexität jedoch rasant zu. Ehemals rein analoge Systeme wurden zusehends durch digitale Komponenten erweitert. Darüber hinaus stieg der Umfang an organisatorischen und sozialen Kontrollen, sowie die Interaktionen zwischen den Komponenten soziotechnischer Systeme. Zu den populärsten Techniken zur Risikoanalyse zählten zu dieser Zeit immer noch Methoden aus den 60er Jahren. Drei der am weitesten verbreiteten Techniken waren Fault-Tree Analysis (FTA), die 1961 von Watson entwickelt wurde [II99], Failure Mode and Effect Analysis (FMEA), die 1949 entstand [Def49], und Harzard and Operability Analysis (HAZOP), welche im Jahre 1964 entworfen wurde [Kle92]. Der Versuch vieler Ingenieure, die für die analoge Welt entwickelten Techniken zu verwenden, um die Risiken moderner, hochkomplexer und mit Computern ausgestatteter Systeme zu analysieren, blieb jedoch oftmals erfolglos [Lev11].

Dieser Tatbestand veranlasste Leveson 2004 dazu, ein neues Modell auszuarbeiten, welches an die Entwicklungen moderner Systeme angepasst ist, und welches zudem soziale und politische Komponenten verarbeiten kann. Dieses Modell heißt *STAMP* (System-Theoretic Accident Model and Processes). In STAMP werden Systeme als zusammenhängende Komponenten betrachtet, die durch Feedback-Regelkreise in einem Equilibrium gehalten werden. Systeme sind keine statischen, sondern dynamische Prozesse, die sich ständig an ihre Umgebung anpassen, um ihre Ziele zu erreichen. Sicherheit wird automatisch dadurch erzielt, dass angemessene Randbedingungen zwischen den Komponenten erfüllt sind, was bei dem Entwurf des Systems sichergestellt werden muss. Ein Unfall kann entstehen, wenn die Interaktionen zwischen menschlichen, organisatorischen und technischen Reglern zu einer Verletzung der Randbedingungen führt. Dies wird als *fehlerhafter Prozess* bezeichnet. Ist ein Unfall eingetreten, kann mit STAMP ermittelt werden, welche der Randbedingungen verletzt wurde und warum die Regler nicht in der Lage waren, diese zu garantieren. Durch diese Vorgehensweise können nicht nur einfache Komponentenausfälle, sondern weitaus komplexere Ursachen analysiert werden. Dazu gehören z. B. unzureichende Produktionsprozesse, ein unzureichender Systementwurf oder ein fehlender Informationsaustausch zwischen menschlichen Reglern. Dies geht über traditionelle Techniken hinaus, die den Unfall schlicht mit dem Ausfall

einer Komponente begründen, ohne klarzustellen, wie dies in Zukunft vermieden werden kann. Ob der Verbau robusterer Komponenten, das Einbringen von Redundanz durch mehrere gleiche Komponenten, oder die Anpassung des Produktionsverfahrens die richtige Wahl ist, kann mit STAMP problemlos analysiert werden.

Das Vorgehen bei einer Unfallanalyse mit STAMP hat den folgenden Ablauf: Zunächst ist ein Modell des Systems notwendig. Dieses besteht aus einem hierarchischen Aufbau der Regler und geregelten Prozesse, der sich in einer Menge miteinander vernetzter Regelkreise manifestiert. Auch Regler, die auf den Entwurf oder Bau des Systems Einfluss hatten, sind in dieses Modell einzubringen. Die Sicherheit des Systems wird garantiert durch diverse Randbedingungen, welche durch die Regelkreise erzwungen werden. Tritt ein Unfall ein, so muss einer der folgenden zwei Fälle vorliegen:

1. Die Randbedingungen wurden nicht durch die Regler erzwungen: Mindestens ein Regler hat ein nötig gewesenes Regelsignal nicht gesendet, zu einem falschen Zeitpunkt gesendet, oder ein falsches Regelsignal gesendet.

2. Angemessene Regelsignale wurden gesendet, aber nicht befolgt: An mindestens einer Stelle in der Hierarchie wurde ein für die Sicherheit nötig gewesenes Regelsignal von einem Regler gesendet, jedoch von dem Empfänger ignoriert.

Es ist zu beachten, dass diese Ursachen je nach Art des Reglers unterschiedlich interpretiert werden müssen. Eine beispielhafte Darstellung eines Regelkreises, bestehend aus einem Regler, einem geregelten Prozess, einem Aktor und einem Sensor, ist in Abb. 2.2 zu sehen. Die Art der Unfallursache kann grob in drei Kategorien eingeteilt werden: der *Betrieb des Reglers*, das *Verhalten der Aktoren und des geregelten Prozesses*, und die *Kommunikation und Koordination zwischen Reglern*. Im Falle von menschlichen Reglern spielen außerdem Kontext und Verhaltensformung eine entscheidende Rolle.

2.2.1 Betrieb des Reglers

Der Betrieb des Reglers hat drei primäre Bestandteile: die *Eingabe*, die *Regelalgorithmen* und das *Prozessmodell*. Falsche, fehlende oder unzureichende Regelsignale, die für die Erfüllung der Randbedingungen notwendig sind, stammen von Fehlern in diesen drei Bestandteilen. In einer hierarchischen Kontrollstruktur unterstehen viele Regler wiederum der Kontrolle eines weiteren Reglers und erhalten von diesem eine *Eingabe*. Die Eingabe enthält u. a. Steueranweisungen und Informationen, die für die Sicherheit des Systems benötigt werden. Fehlt diese Eingabe oder ist sie fehlerhaft, kann dies zu einem falschen Betrieb des Reglers führen.

Die *Regelalgorithmen* umfassen sowohl die Prozeduren, die von Ingenieuren für die technischen Regler entworfen wurden, als auch die Prozeduren, denen menschliche Regler folgen. Regelalgorithmen können unfähig sein, Randbedingungen zu erfüllen, weil sie von Anfang an schlecht konzipiert waren, weil sich der Prozess ändert, oder weil sie im Rahmen einer Änderung unsicher werden. Bei dem Entwurf von Regelalgorithmen ist zu beachten, dass es immer eine Verzögerungszeit zwischen den einzelnen Teilen eines Regelkreises gibt, z. B. zwischen Messung und Erhalt der Daten oder zwischen Senden und Ausführen eines Regelsignals. Je nach Art der Verzögerung müssen unterschiedliche Komponenten für einen Ausgleich sorgen [Bre92]. So muss u. U. ein

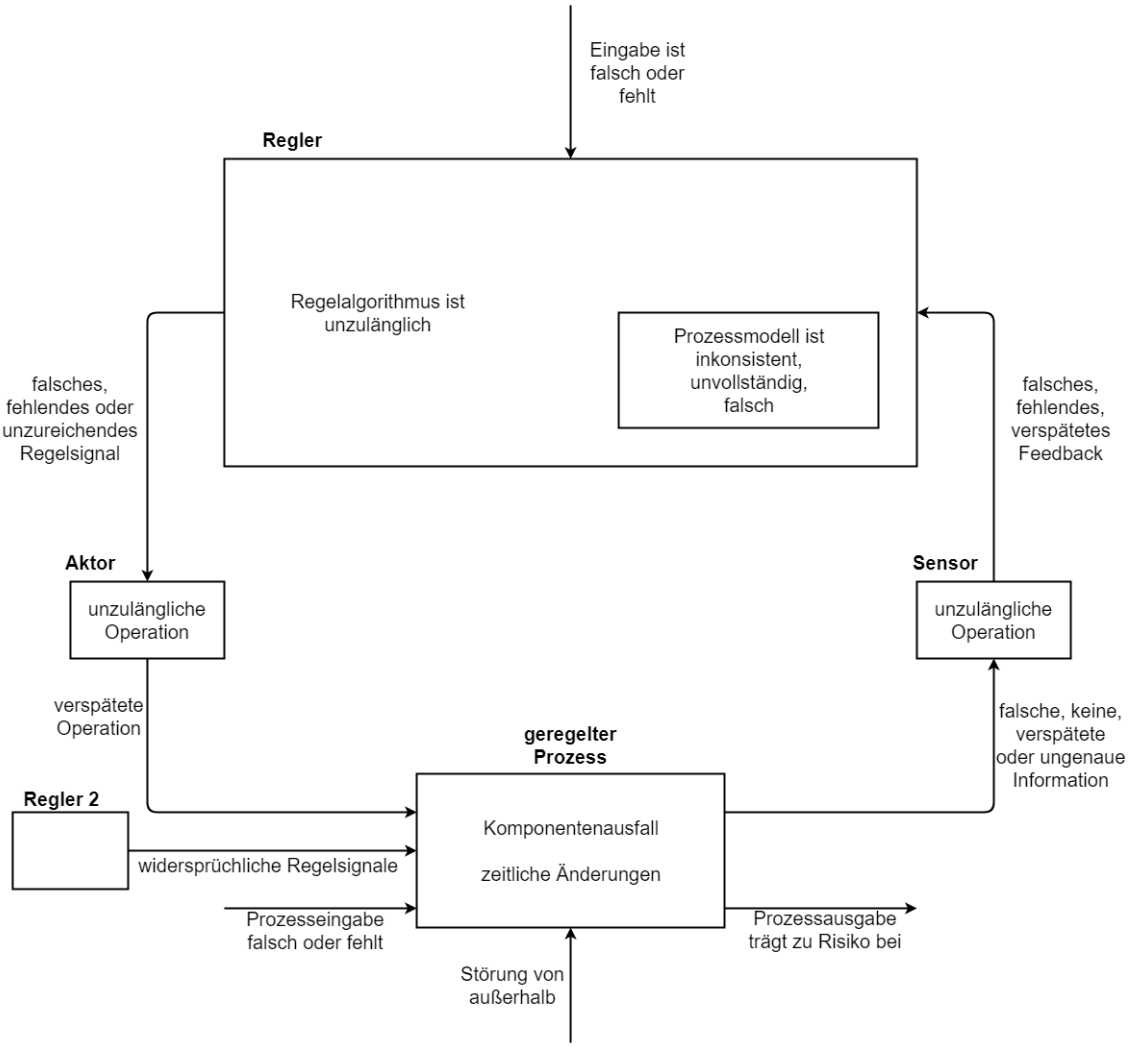


Abbildung 2.2: Regelkreis im Detail

Regelalgorithmus entworfen werden, der dazu imstande ist, nötige Regelsignale vorherzusagen. Auch Vorsteuerungsstrategien sind anwendbar. Werden Verzögerungen nicht in angemessener Weise bedacht, kann es zu Unfällen kommen.

Leplat merkte an, dass viele Unfälle ihre Ursachen in der *asynchronen Evolution* eines Systems haben, d. h. dass eine Komponente geändert wird, ohne die anderen Komponenten entsprechend anzupassen [Lep84]. Auch die Abnutzung einer einzelnen Komponente fällt unter diesen Begriff. In beiden Fällen sind es die irrtümlichen Annahmen der unveränderten Komponenten zu dem Verhalten des geänderten Teilsystems, die zu dem Unfall führen. Die Vermeidung asynchroner Evolution erfordert Überwachung von Änderungen und zureichende Kommunikation zwischen Änderungen feststellenden und hierarchisch höhergestellten Regelstrukturen. Bei der Vereinbarung von Randbedingungen werden immer Annahmen zu dem System gemacht. Sobald diese Annahmen nicht mehr gelten, können die Regler die Randbedingungen nicht mehr garantieren.

Die Regelsignale eines Reglers basieren auf seinem Prozessmodell. Dies ist eine interne Repräsentation des geregelten Prozesses und kann sowohl den Speicher eines Computers, als auch das mentale Modell eines Menschen darstellen. Existiert eine Diskrepanz zwischen dem internen Modell und realen Prozess, können die Signale des Reglers ungenau werden, da sie von einem inkorrekten Prozess ausgehen. Beispielsweise könnte die Software eines gelandeten Flugzeugs fälschlicherweise annehmen, dass das Flugzeug noch in der Luft ist, und das Fahrwerk einfahren. Auch das mentale Modell der Systementwickler ist von Bedeutung. So könnte z. B. das Entwicklungsteam einer Software von anderer Hardware ausgehen, als die im Nachhinein tatsächlich verwendete Hardware. Die Situation wird noch dadurch verkompliziert, dass oft mehrere Regler existieren, deren Prozessmodelle konsistent gehalten werden müssen. Die häufigste Ursache einer Inkonsistenz ist die Unvollständigkeit des Prozessmodells eines Reglers, d. h. einige mögliche Szenarien wie etwa der Ausfall bestimmter Komponenten sind nicht definiert. Es sei anzumerken, dass absolute Vollständigkeit nicht zu erreichen ist, sondern ein ausreichend hoher Grad an Vollständigkeit das Ziel ist. Zu den Ursachen eines falschen Prozessmodells gehören, dass das Modell von Anfang an falsch war, dass der Regler kein oder falsches Feedback von einem Sensor bekommt, oder dass das Modell trotz richtigem Feedback falsch aktualisiert wird. Unzureichendes Feedback kann von einem inkorrekt funktionierendem Sensor, einem Fehler in der Übertragung, oder daher kommen, dass die Weitergabe bestimmter Informationen nicht im Entwurf des Systems vorgesehen war. Ist das Feedback verspätet, so ist das Prozessmodell nur für eine kleine Zeitspanne fehlerhaft.

2.2.2 Aktoren und der geregelte Prozess

Neben einem falschen Betrieb des Reglers besteht die Möglichkeit, dass der Regler die richtigen Befehle sendet, jedoch der geregelte Prozess diese Befehle nicht umsetzt. Ein möglicher Grund dafür ist, dass das Signal aufgrund eines defekten Informationskanals während der Übertragung verfälscht wird oder verloren geht. Ein weiterer Grund könnte der Ausfall eines Aktors oder einer anderen geregelten Komponente sein. Externe Störungen, die vom Regler nicht beeinflusst werden können, und dessen Wirken auf den Prozess zu Unfällen beiträgt, stellen einen dritten möglichen Grund dar. Sinngemäße Faktoren gelten auch im Fall, dass das Regelsignal an einen hierarchisch niederen Regler gesendet wird. Ein verbreiteter Mangel ist z. B., dass während dem Entwurf oder Bau eines Systems sicherheitsrelevante Informationen nicht adäquat weitergeleitet werden.

2.2.3 Kommunikation und Koordination zwischen Reglern

Wenn mehrere Regler parallel arbeiten, kann es zu schlecht koordinierten Regelsignalen kommen. Häufig sind Kommunikationsdefizite die Ursache. Leplat fand heraus, dass mehr Unfälle in Systemteilen entstehen, in denen sich die Zuständigkeitsbereiche zwei oder mehr Regler überlappen – auch wenn es sich um menschliche Regler handelt [Lep84]. Oftmals sind gerade in solchen Bereichen die Zuordnung der Verantwortlichkeiten ungenügend, wodurch es zu Konflikten zwischen mehreren unabhängigen Signalen kommt [Lev11].

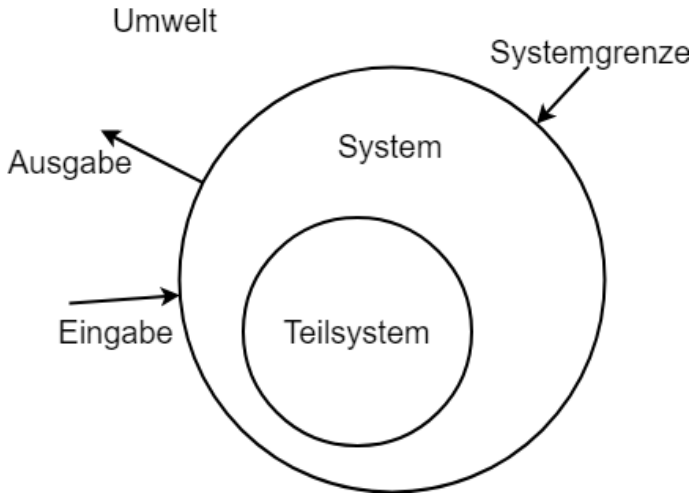


Abbildung 2.3: System und Umwelt

2.3 STPA

STPA steht für System-Theoretic Process Analysis und wurde von Leveson entwickelt, um die in STAMP eingeführten Unfallursachen erstmals in einer Risikoanalysetechnik unterzubringen. Ein weiteres Ziel war eine umfassende Unterstützung der Ingenieure bei der Erzielung guter Ergebnisse, wie sie z. B. bei Fault-Tree Analysis fehlt. Ohne diese Unterstützung sind die Resultate stark von der fachlichen Kompetenz der Analytiker abhängig und unterliegen dem Risiko, unvollständig zu sein. Ein drittes Ziel war die Entwicklung einer Technik, die schon vor dem Systementwurf angewendet werden kann und diesen leitet. Dies ist eine Erleichterung insbesondere in Bezug auf die Kosten. STPA kann aber auch auf bereits existierende Entwürfe angewendet werden.

Ähnlich wie bei anderen Unfallanalysetechniken liegt der Hauptaugenmerk einer Analyse mit STPA auf dem Sammeln von Informationen darüber, wie die abgeleiteten Randbedingungen verletzt werden könnten. Je nach Zeitpunkt der Analyse kann mithilfe dieser Informationen sichergestellt werden, dass die Randbedingungen während Entwurf, Konstruktion, Test und Betrieb erfüllt sind [Lev11]. Der grundlegende Ablauf von STPA umfasst vier Phasen: die Definition von Verlusten, die Modellierung des Systems, die Identifikation von unsicheren Regelsignalen und die Identifikation von Verlustszenarien. Diese Phasen werden in den folgenden Abschnitten erläutert.

2.3.1 Definition von Verlusten

Ein *Verlust* betrifft immer etwas, das für Stakeholder einen Wert besitzt. Beispiele für Verluste sind der Verlust von Menschenleben, Verletzungen, Sachschäden, Umweltverschmutzung, Imageschäden, oder Datenleaks. Um die Verluste zu definieren, werden zunächst alle Stakeholder identifiziert, dann deren ”Stakes” im System und zuletzt die damit zusammenhängenden Verluste. Nach der Bestimmung der Verluste geht es im nächsten Schritt um die *Gefährdungen*. Diese unterscheiden sich von den Verlusten dadurch, dass sie vollständig innerhalb der Systemgrenzen definiert sind. Gelangt das System in einen Zustand, der eine Gefährdung darstellt, ist unter gewissen Umweltbedingungen ein Verlust möglich. Abb. 2.3 veranschaulicht ein solches System in seiner Umwelt. Da die

Systementwickler keine Kontrolle über die Umwelt haben, und mit dem Eintreten von Worst-Case-Bedingungen zu rechnen ist, besteht deren Aufgabe in der Vermeidung von Gefährdungen. Einige mögliche Gefährdungen sind z. B. die Verletzung des Minimalabstands zwischen zwei Flugzeugen oder das Austreten giftiger Chemikalien. Beide Szenarien führen nicht zwangsläufig zu einem Verlust, aber unter ungünstigen Umweltbedingungen ist der Verlust unausweichlich. Anzumerken ist, dass Gefährdungen immer Systemzustände sind, die verhindert werden müssen: Ein Flugzeug in der Luft kann zwar zu mehr Verlusten führen als am Boden, dies wird aber im Rahmen einer STPA-Analyse nicht als Gefährdung angesehen, da es ein zur Erfüllung der Ziele notwendiger Zustand ist. Aus der letztendlich zusammengestellten Liste von Gefährdungen werden schließlich die Randbedingungen auf Systemebene abgeleitet. So wird aus ”Flugzeug hält Minimalabstand nicht ein” ganz trivial ”Flugzeug muss Minimalabstand einhalten”. Jede Randbedingung lässt sich zu einer oder mehreren Gefährdungen zurückführen und jede Gefährdung zu einem oder mehreren Verlusten. Eine eins-zu-eins-Verknüpfung wird nicht erzwungen. Die gefundenen Verluste, Gefährdungen und Randbedingungen können hiernach noch verfeinert oder mit Prioritäten versehen werden.

2.3.2 Modellierung des Systems

In der zweiten Phase von STPA ist das Ziel die Anfertigung eines Modells für das System. Dieses besteht aus hierarchisch aufgebauten Regelkreisen, wie sie in STAMP eingeführt wurden: Regler senden Signale an den geregelten Prozess. Diese werden von den Regelalgorithmen bestimmt, welche wiederum von dem internen Modell über den Prozess ausgehen, dem Prozessmodell. Das Prozessmodell wird aktualisiert von Feedbacksignalen von dem geregelten Prozess. Probleme können an jedem dieser Punkte auftreten, und müssen systematisch identifiziert werden. Abb. 2.4 demonstriert die Verschachtelung multipler Regelkreise am Beispiel eines vereinfachten Luftfahrtmodells. Typischerweise befinden sich Regler, die über andere Regler eine gewisse Autorität verfügen, im Modell über diesen. So hat etwa die Flugverkehrskontrolle direkte Kontrolle über den Piloten, während der Bordcomputer der Kontrolle des Piloten untersteht. Daraus folgt, dass nach unten gerichtete Pfeile Regelsignale repräsentieren, wohingegen nach oben gerichtete Pfeile für Feedback stehen. Auch bei der Erstellung des Modells wird zunächst mit einem groben Entwurf gestartet, der dann iterativ verfeinert wird. Dieser Top-Down-Ansatz ermöglicht auch die Modellierung von komplexen Systemen. Dabei können die vorher identifizierten Randbedingung benutzt werden, um nötige Komponenten zu identifizieren. Für die Geschwindigkeitsverminderung eines Flugzeuges werden z. B. Bremsen und Umkehrschub benötigt. Nachdem auf diese Weise alle Elemente definiert wurden, muss im nächsten Schritt die Frage nach der Art der Kontrolle beantwortet werden: Ist die Betätigung der Bremsen durch den Bordcomputer ausreichend, oder muss dem Piloten die Möglichkeit gegeben werden, den Computer im Notfall zu umgehen und manuell zu bremsen? Hierbei ist auch die Vergabe von Verantwortlichkeiten wichtig, etwa dass der Pilot den Bremsvorgang initiiert, jedoch der Bordcomputer die Option hat, die Bremsen bei Blockieren kurzzeitig zu lösen. Erst nachdem diese Verantwortlichkeiten zugeteilt wurden, kann auf den genauen Inhalt der Regel- und Feedbacksignale eingegangen werden. Hierbei ist zu beachten, dass in dem gegebenen Feedback alle zur Steuerung der Regelsignale benötigten Informationen enthalten sind.

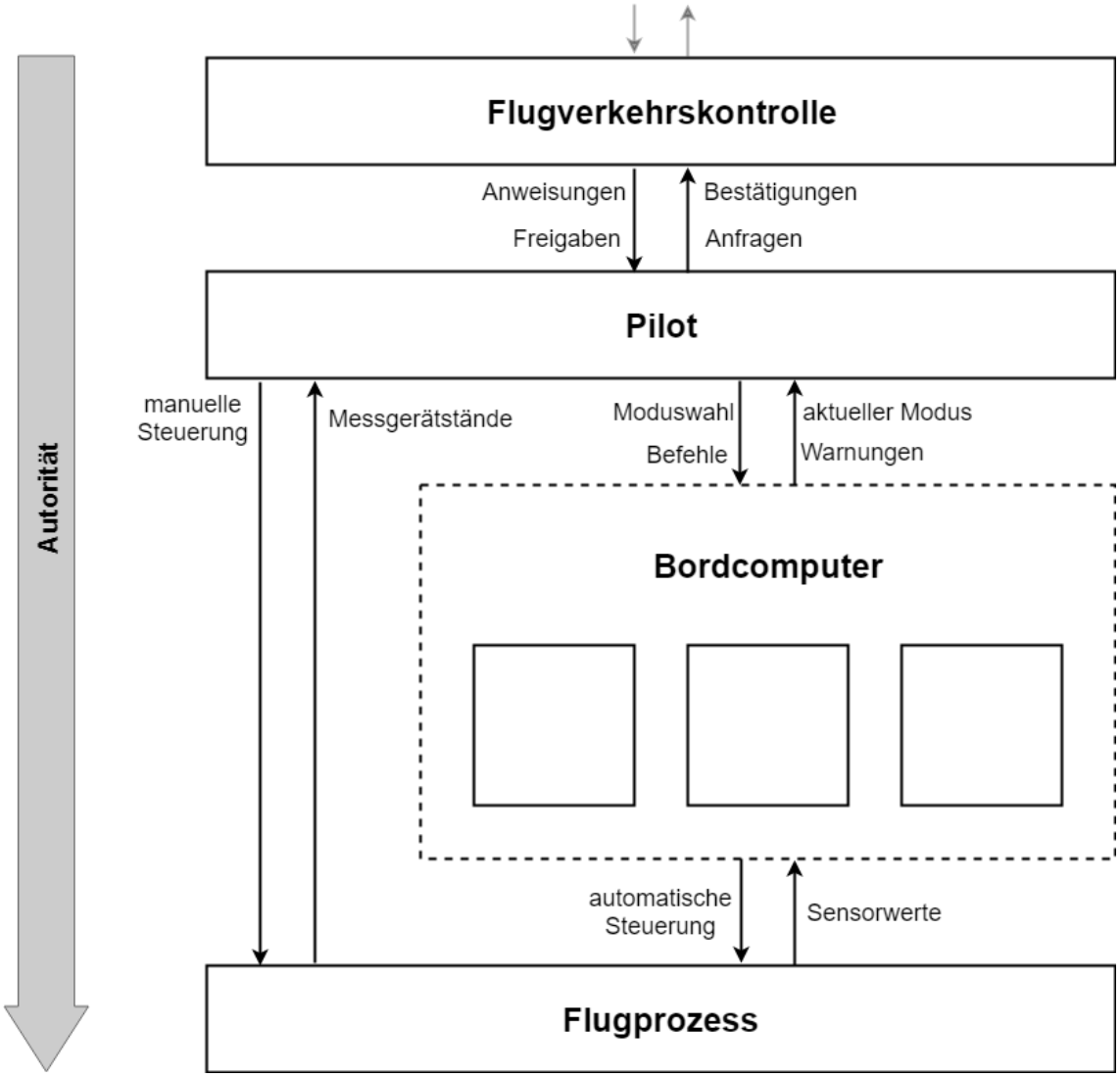


Abbildung 2.4: Regelkreisstruktur in einem Luftfahrtszenario (das englische Original ist in *STPA Handbook*, S. 24 zu finden)

2.3.3 Identifikation von unsicheren Regelsignalen

Ein unsicheres Regelsignal ist ein Regelsignal, welches unter Umständen zu einer Gefährdung führt. Tabelle 2.1 zeigt eine Auswahl unsicherer Regelsignale im Zusammenhang mit einer Flugzeugbremse, die vom Bordcomputer gesteuert wird. Insgesamt gibt es vier verschiedene Arten, auf die ein Regelsignal unsicher sein kann:

1. Das Nichtsenden eines Regelsignals führt zu einer Gefährdung.

2. Das Senden eines Regelsignals führt zu einer Gefährdung.

3. Das Senden eines potentiell sicheren Regelsignals, jedoch zu früh, zu spät, oder in der falschen Reihenfolge, führt zu einer Gefährdung.

Regelsignal	**Nichtsenden verursacht Gefährdung**	**Senden verursacht Gefährdung**	**Zu früh, zu spät, oder falsche Reihenfolge**	**Zu früh gestoppt, zu lange gesendet**
Bremse	UCA-1: Bordcomputer sendet kein Bremssignal nach Aufsetzen auf Landebahn [H-1]	UCA-2: Bordcomputer sendet Bremssignal während Startvorgang [H-2] UCA-3: Bordcomputer sendet Bremssignal unzureichender Stärke nach Aufsetzen auf Landebahn [H-1] UCA-4: Bordcomputer sendet ungleichmäßiges Bremssignal an rechte und linke Seite nach Aufsetzen auf Landebahn [H-1]	UCA-5: Bordcomputer sendet Bremssignal zu spät (x Sekunden nach Aufsetzen auf Landebahn) [H-1]	UCA-6: Bordcomputer stoppt Bremssignal zu früh (bevor Anrollgeschwindigkeit erreicht ist) [H-1]

Tabelle 2.1: Identifikation von unsicheren Regelsignalen

4. Das Senden eines Regelsignals dauert zu lange an oder wird zu früh gestoppt (dieser Fall ist nur bei andauernden Signalen anwendbar).

Die unsicheren Regelsignale werden üblicherweise mit UCA abgekürzt (aus dem Englischen: Unsafe Control Action) und durchnummeriert. Außerdem wird empfohlen, alle Gefährdungen zu referenzieren, zu denen das jeweilige unsichere Regelsignal führen kann. Diese werden mit H abgekürzt (englisch: Hazard). Die Definition eines unsicheren Regelsignals besteht aus fünf Teilen:

UCA-1: Bordcomputer sendet kein Bremssignal nach Aufsetzen auf Landebahn [H-1]
<Quelle> <Typ> <Regelsignal> <Kontext> <Referenz>

Der erste Teil spezifiziert den Namen des Reglers, der das Signal sendet. Der zweite Teil teilt das Signal in eine der oben angegeben Klassen ein (sendet, sendet nicht, sendet zu früh/spät und sendet zu kurz/lange). Der dritte Teil gibt den Namen des Signals an. Der vierte Teil bestimmt den Kontext, unter welchem das Signal unsicher ist und der fünfte Teil referenziert die dazugehörige Gefährdung. Die Bestimmung des Kontextes ist wichtig, da es kein Regelsignal gibt, das immer unsicher ist – ansonsten würde es nicht in das System aufgenommen werden. Die Identifikation von unsicheren Regelsignalen bei menschlichen Reglern funktioniert analog und kann somit in die Gesamtanalyse integriert werden.

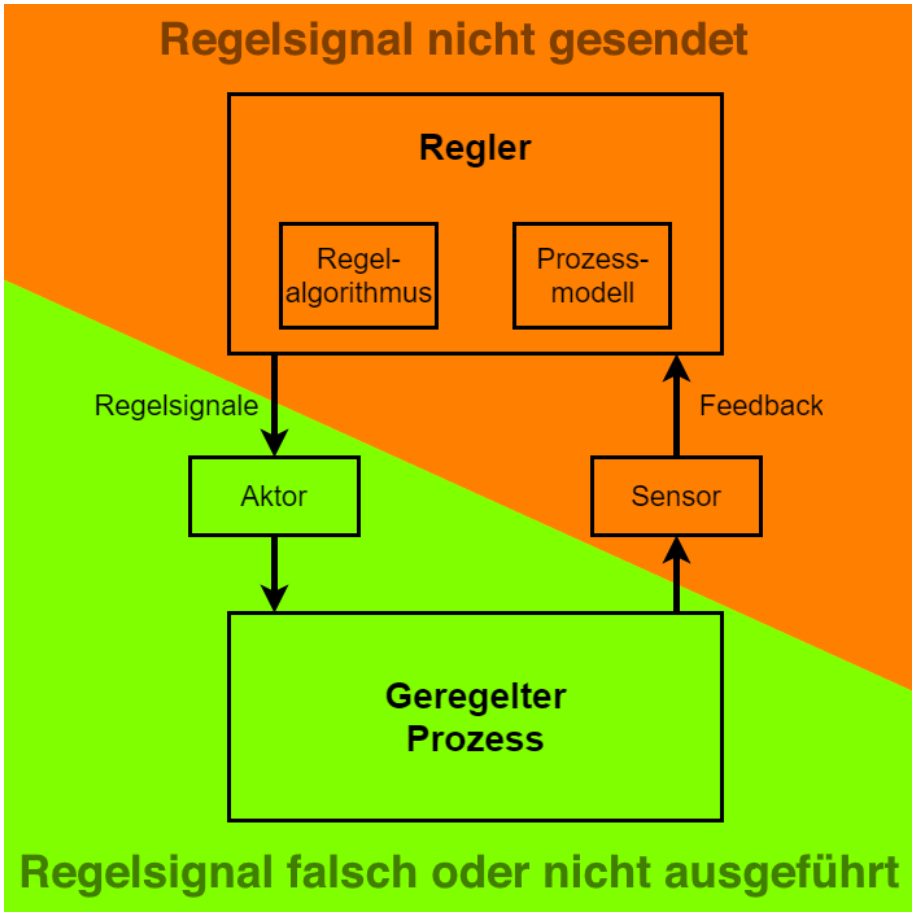


Abbildung 2.5: mögliche Verlustszenarien

Nachdem alle unsicheren Regelsignale identifiziert wurden, werden im nächsten Schritt die Randbedingungen auf Reglerebene abgeleitet. Wie schon auf Systemebene ist dies eine triviale Translation von ”Bordcomputer sendet kein Bremssignal” nach ”Bordcomputer muss Bremssignal senden”.

2.3.4 Identifikation von Verlustszenarien

Ein Verlustszenario ist ein Szenario, in dem ein unsicheres Regelsignal gesendet wird, was zu einer Gefährdung und schließlich zu einem Verlust führt. Zwei Arten von Verlustszenarien werden unterschieden: Im ersten Fall entsteht das Verlustszenario durch das Senden eines unsicheren Regelsignals. Im zweiten Fall entsteht das Verlustszenario dadurch, dass ein korrekt gesendetes Regelsignal falsch oder nicht ausgeführt wird. Abb. 2.5 veranschaulicht diese Einteilung. Die Abbildung enthält außerdem Aktoren und Sensoren, da diese bei der Suche nach Ursachen für unsichere Regelsignale und Feedback hilfreich sind. Zu Beginn der Suche wird im Modell eines der Signale betrachtet und dann von dort aus im Regelkreis rückwärts gelaufen, um mögliche Ursachen für ein unsicheres Regelsignal zu finden. Dies wird für alle übrigen Signale wiederholt. Im Allgemeinen gibt es vier Gründe, warum ein Regler ein unsicheres Regelsignal sendet: ein vollständiges Aussetzen des Reglers (etwa ausgelöst durch einen Defekt oder Stromausfall), ein unzureichender Regelalgorithmus, eine unsichere Eingabe von einem hierarchisch höherem Regler,

oder ein inkorrektes Prozessmodell. Für ein inkorrektes Prozessmodell gibt es eine Vielzahl von Ursachen. Dazu gehören fehlendes oder falsches Feedback von einem Sensor, ein verspätetes Eintreffen von Feedback oder sogar das Nichtvorhandensein notwendiger Informationskanäle. Wird während der Analyse festgestellt, dass ein möglicher Grund für ein unsicheres Regelsignal ein inkorrektes Prozessmodell sein kann, dann muss geklärt werden, wo die benötigten Informationen herkommen und auf welche Weise Probleme entstehen können. Auch hier sind viele Ursachen denkbar. Ein Sensor könnte defekt sein oder im Laufe der Zeit ungenauer werden, im Prozess könnten unvorhergesehene Bedingungen auftreten, oder die Informationsübertragung könnte fehlerhaft sein.

Für Verlustszenarien, die durch falsche oder fehlende Ausführung eines Regelsignals zustande kommen, müssen sowohl Faktoren in Betracht gezogen werden, die die Aktoren, als auch solche, die den geregelten Prozess beeinflussen. Zu ersteren gehören Defekte des Aktors, nicht erhaltene Regelsignale und irrtümlich ausführende Aktoren. Zu letzteren gehören externe Störungen und widersprüchliche Einflüsse von anderen Aktoren [LT18].

2.4 XSTAMPP

XSTAMPP steht für *eXtensible STAMP Platform* und wurde 2015 von Abdulkhaleq und Wagner vorgestellt [AW15a]. Ziel des Programms ist die Unterstützung bei der Arbeit mit STAMP und STPA. Bis einschließlich Version 3.1.2 baute XSTAMPP auf die Eclipse Rich Client Platform (RCP) auf [McA+10]. Der Code dieser Version ist Open Source und kann auf der offiziellen GitHub-Seite heruntergeladen werden.[1] Enthalten ist dort neben dem Quellcode auch eine vorkompilierte Version und sieben Plugins. Die sich aktuell in Entwicklung befindliche Version 4.0 setzt die Funktionalität erstmals in einer Webanwendung um. Dies erlaubt die Verwendung der reichhaltigen Funktionalität ohne jede Installation eines Programms. Außerdem ermöglicht es das parallele Arbeiten an dem selben Projekt auf mehreren Computern. Der im Back-End laufende Server synchronisiert die Analyseergebnisse und verhindert gegenseitiges Überschreiben mithilfe von Datensatzsperren. Der Code dieser Version ist nicht öffentlich. Eine der Hauptideen von XSTAMPP war seit jeher Erweiterbarkeit. Die Webanwendung besitzt daher ein öffentliches Application Programming Interface (API), mit welchem Drittsoftware an die Plattform angeschlossen werden kann.

Abb. 2.6 zeigt die Benutzeroberfläche von XSTAMPP 4.0. Über die Navigationsleiste an der linken Seite können die einzelnen Schritte der Analyse aufgerufen werden. Viele der Analyseergebnisse wie etwa Verluste und Gefährdungen werden in sortier- und filterbaren Tabellen angezeigt und lassen sich verknüpfen, um gegenseitige Rückführbarkeit zu erleichtern. Bei der Konstruktion der Regelkreise lassen sich die einzelnen Komponenten von der Werkzeugleiste rechts auf die Oberfläche ziehen, verkleinern oder vergrößern, sowie benennen. Pfeile dienen dazu, Informationsflüsse anzuzeigen und können gleichermaßen benannt werden. Die hier definierten Systemteile werden automatisch in entsprechende Tabellen übertragen und können dort um weitere Details ergänzt werden. Das Prozessmodell findet aus Platzgründen nicht direkt im Regelkreis, sondern in einem separaten Bereich erstellt. Im vierten Schritt ist außerdem eine Verfeinerung des Regelkreisdiagramms möglich. XSTAMPP beinhaltet eine rudimentäre Benutzerverwaltung, mithilfe welcher verschiedenen Nutzern unterschiedliche Privilegien zugeteilt werden können. Dazu gehören das reine Einsehen

[1] https://github.com/SE-Stuttgart/XSTAMPP

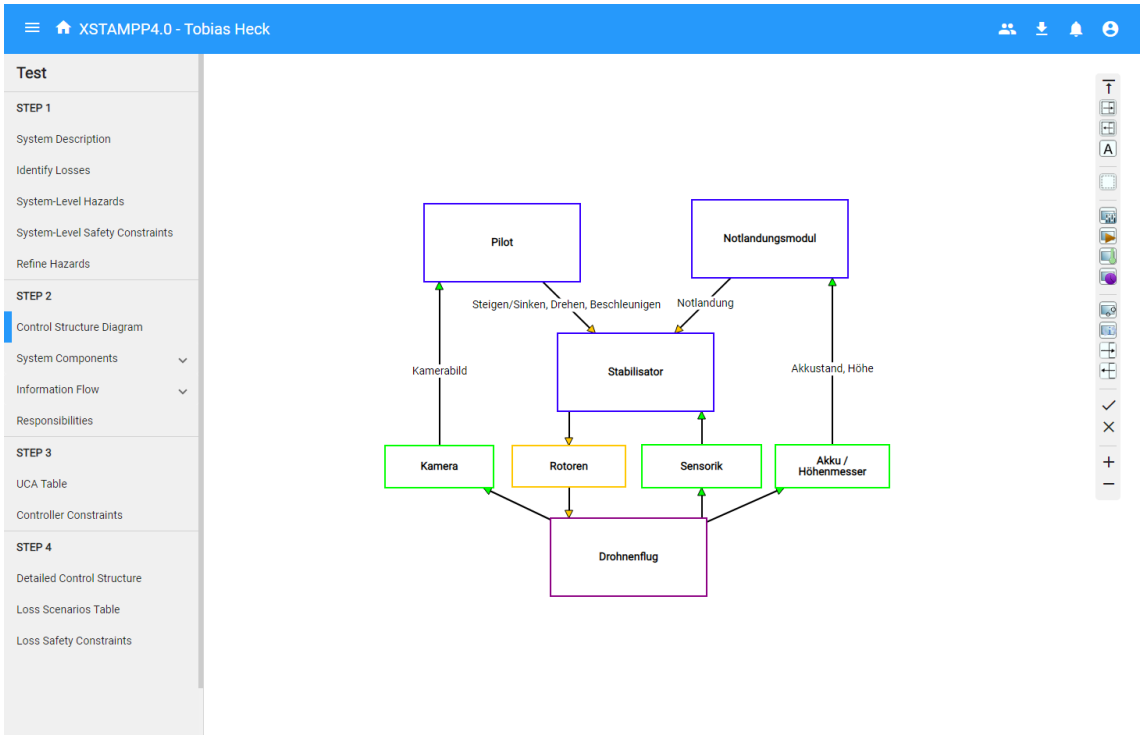


Abbildung 2.6: Benutzeroberfläche von XSTAMPP

eines Projekts, das Bearbeiten dieses oder das Erstellen von neuen Projekten. Die Exportfunktion soll nach Entwicklungsende einerseits die Speicherung in einem maschinell lesbaren Binärformat, als auch die Ausgabe eines inhaltlich konfigurierbaren Berichts unterstützen.

2.4.1 Technologien in XSTAMPP

XSTAMPP 4.0 baut auf mehrere für die Webentwicklung vorgesehene Technologien auf. Diese werden im Folgenden kurz vorgestellt.

1. Docker ist ein Service zur Virtualisierung unabhängiger Softwarekomponenten innerhalb sogenannter Container, auf welchen eine leichtgewichtige Variante von Linux läuft. Das gesamte Back-End von XSTAMPP läuft auf der von Docker bereitgestellten virtuellen Umgebung und profitiert so von deren Standardisierung der Entwicklungs- und Laufzeitumgebung.

2. Spring ist ein umfangreiches Framework für Java-Programme. XSTAMPP baut im Back-End auf diesem auf und macht Gebrauch von der angebotenen Inversion of Control (IOC). Dies verbessert die Erweiterbarkeit der STPA-Analysesoftware.

3. Angular ist ein Open-Source-Framework, das auf TypeScript basiert. In XSTAMPP übernimmt es die die Verwaltung des kompletten Front-Ends. Angular stellt zahlreiche Designwerkzeuge zu Verfügung und erlaubt dessen direkte Einbindung in den HTML-Code. Das Framework übernimmt die Aktualisierung der STPA-Tabellen bei Datenänderungen, die Kommunikation via asynchroner HTTP-Anfragen, und einige weitere Dinge.

4. NGINX ist eine Software zur Verarbeitung von Webanfragen. Sie zeichnet sich durch eine Abwicklung der eingehenden Verbindungen in einem einzelnen Thread aus, wodurch sie performant und ressourcensparend ist [Kin19]. In XSTAMPP ist NGINX außerdem für die RESTful Schnittstelle zuständig, über die Drittsoftware auf das System zugreifen kann.

5. PostgreSQL ist ein objektrelationales Datenbankverwaltungssystem, das sich mit SQL und einigen speziellen Befehlen ansteuern lässt. Es wird in XSTAMPP verwendet, um alle persistenten Daten effizient zu speichern und zu laden. Aus Sicherheitsgründen existieren zwei getrennte Datenbanken für die STPA-Analyseprojekte und die Benutzerverwaltung.

6. Hibernate bildet die Schnittstelle zwischen der objektorientierten Arbeitsweise von XSTAMPP und der relationalen Datenhaltung auf der Datenbank. Sie abstrahiert die Verwaltung der persistenten Daten, indem sie sie bei Speicher- und Ladevorgängen in eine Form konvertiert, mit der das jeweils andere Ende umgehen kann.

2.4.2 Erweiterung um Regelalgorithmen

Diese Arbeit hat das Ziel, XSTAMPP um Funktionen zu erweitern, die die Modellierung von Regelalgorithmen erlauben. Diese sollen sich nahtlos in die vorhandene Benutzeroberfläche einfügen, und über die selben Sicherheitsvorkehrungen verfügen, die existierende Funktionen aufweisen. Dazu gehören einerseits die Verwendung von Datensatzsperren zur Vorbeugung von unabsichtlichen Überschreiben bei der gleichzeitigen Bearbeitung eines Eintrags. Andererseits muss auch die Überprüfung der Zugriffsrechte gewährleistet werden. Ebenso wie die aktuell mit XSTAMPP verwalteten Analyseergebnisse werden auch die Regelalgorithmen via Hibernate in die Datenbank übertragen, die für die Speicherung der Projektdaten zuständig ist. Die Implementierung der neuen Funktionen ist in Kapitel 5 dokumentiert.

2.5 Zusammenfassung

In diesem Kapitel wurden die Grundlagen aller Modelle und Techniken erklärt, die für das Verständnis der restlichen Arbeit hilfreich sind. Systemtheorie ist eine Ansatz, bei dem Systeme als dynamische Körper betrachtet werden, die auf ihr jeweiliges Ziel hinarbeiten, indem sie Einfluss auf ihre Eingaben und Ausgaben nehmen. STAMP ist ein Unfallkausalitätsmodell, das Systeme als Regelkreise darstellt. Ein Regler steuert einen Prozess, indem er Regelsignale an Aktoren sendet. Information zu dem Prozess erhält er über Sensoren. Im Inneren des Reglers befindet sich ein Modell des Prozesses sowie ein Regelalgorithmus, mit dem die zu sendenden Regelsignale bestimmt werden. Mögliche Ursachen für einen Unfall werden mit der Analysetechnik STPA identifiziert. Dabei werden zunächst unsichere Regelsignale gesammelt und anschließend Szenarien ermittelt, in denen diese auftreten können. Ein Programm, das für die STPA-Analyse dienliche Funktionen bietet, ist XSTAMPP. Teil dieser Arbeit ist die Implementierung der Regelalgorithmen in dieses Programm. Dazu werden im nächsten Kapitel mehrere Dokumentationen existierender STPA-Analysen untersucht.

3 State of the Art

STPA ist eine weit verbreitete Analysetechnik, und so existieren zahlreiche Programme, Dokumente erfolgter Analysen und Forschungsarbeiten. Inhalt dieses Kapitels sind die Ergebnisse einer ausführlichen Recherche zu dem bisherigen Umgang mit Regelalgorithmen. Dabei werden zunächst Programme betrachtet, zu dessen Funktionsumfang die Bearbeitung der STPA-Schritte gehört. Da solche Programme zum Teil rein konzeptionell sind und nicht für die vielfältigen Anforderungen der Industrie entwickelt wurden [ST14], werden im Nachfolgenden zusätzlich eine Auswahl von STPA-Analyseergebnissen untersucht. Hierbei ist die Auswertung möglichst verschiedenartiger Regler nicht zu vernachlässigen, da das zu entwickelnde Modell in möglichst vielen Fällen anwendbar sein soll. Insbesondere die Diskrepanz zwischen den Denkweisen technischer und menschlicher Regler darf nicht vernachlässigt werden. Deshalb wird diese in einem eigenen Abschnitt gesondert behandelt. Die Bewertung der gesammelten Konzepte erfolgt in Kapitel 4.

3.1 Weitere STPA Programme

Nach ausführlicher Recherche wurden acht Programme gefunden, die speziell auf Analysen mit STPA zugeschnitten sind. Zwei dieser Programme werden hier nicht behandelt. Dies sind XSTAMPP und A-STPA. Beide Programme wurden bereits in Kapitel 2 besprochen. Zwei weitere wurden nicht bzw. noch nicht veröffentlicht, und können daher nicht direkt untersucht werden. Es existieren jedoch Berichte zu dessen Funktionen, auf welche kurz eingegangen wird.

3.1.1 SpecTRM

Specification Tools and Requirements Methodology (SpecTRM) wurde bereits 1998 von Leveson et al. vorgestellt und ist damit das älteste Programm für Sicherheitsanalysen nach STPA-Muster [Lev+98]. Obgleich STPA selbst zum Veröffentlichungszeitpunkt des Programms noch nicht existierte, lassen die Ideen in SpecTRM bereits dessen Grundstrukturen erkennen. So ist z. B. das Sammeln von Gefährdungen und das Ableiten von Randbedingungen Teil der Prozedur. Abb. 3.1 präsentiert ein Analyseprojekt eines Thermostats in SpecTRM. *Outline* zeigt die sechs Phasen, die während einer Analyse durchlaufen werden, sowie deren Erzeugnisse. In der dritten Phase werden das Verhalten des Systems und das des Nutzers modelliert, welche in SpecTRM getrennte Elemente sind. Ebenso wie in STPA können Systemkomponenten oder Nutzer Regelsignale senden und Feedback erhalten, hier schlicht Input und Output genannt. Für diese Signale können Vorbedingungen definiert werden. Beispielsweise sendet in der Abbildung das Thermostat ein Einschaltsignal an den Ofen, wenn es kalt ist, d. h. wenn sich die Variable `Temperatur` im Zustand `kalt` befindet. Die Variablen können hier sowohl diskrete, als auch kontinuierliche Form annehmen, wobei kontinuierliche Variablen auch mit > und < verglichen werden können. Damit das Signal nur eine Zeiteinheit lang gesendet wird, ist die Bedingung ab der zweiten Überprüfung falsch. Zusätzlich zu den Vorbedingungen kann auch der

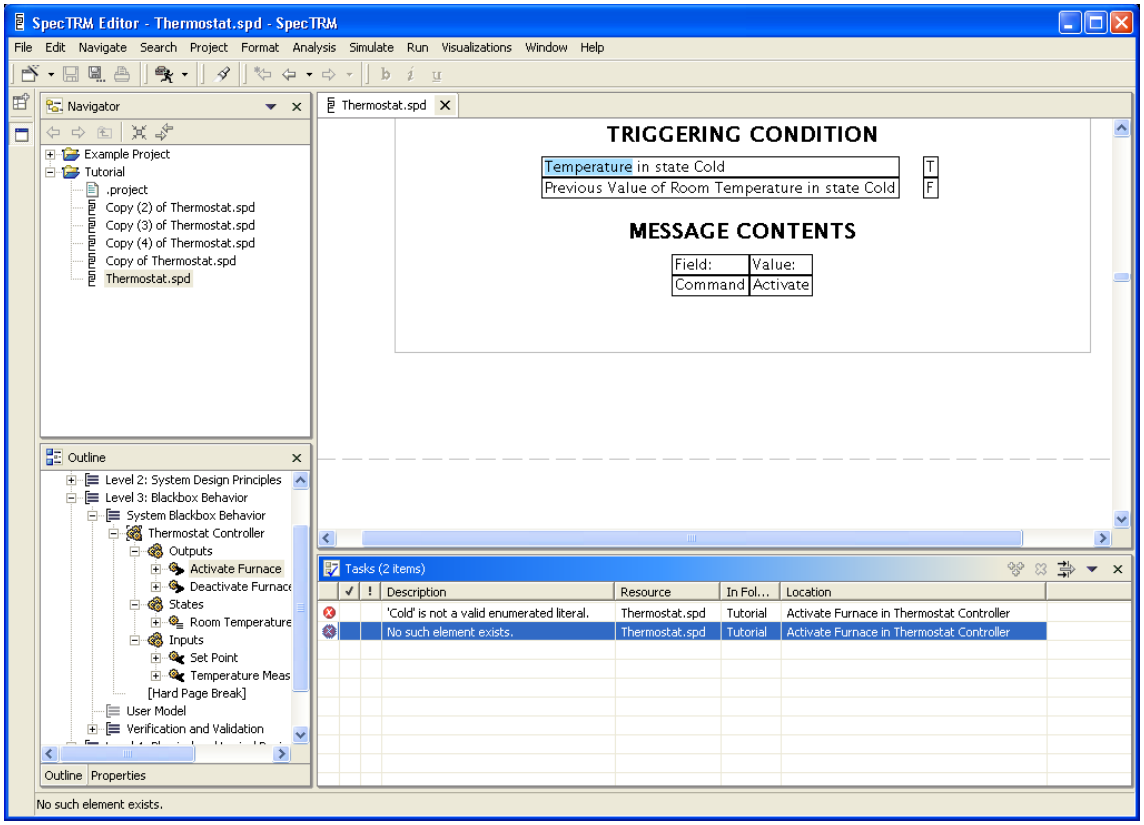


Abbildung 3.1: Benutzeroberfläche von SpecTRM

Inhalt eines Regel- oder Feedbacksignals bestimmt werden. Genau wie bei den Vorbedingungen sind diskrete und kontinuierliche Variablen möglich. Das System- und Nutzerverhalten kann außerdem durch das Anlegen von Modi weiter spezialisiert werden. Dabei entspricht ein Moduswechsel einer Transition in einem endlichen Automaten, d. h. von jedem Modus kann nur in eine fest vorgegebene Menge von Modi gewechselt werden. Jeder Modus besitzt seine eigenen Vorbedingungen für die Regelsignale. Mithilfe dieses reichhaltigen Funktionsumfangs ist der Aufbau eines sehr flexiblen Regelalgorithmus möglich. In Kapitel 4.1 werden die in SpecTRM dargebrachten Konzepte näher untersucht.

3.1.2 SAHRA

STPA based Hazard and Risk Analysis (SAHRA) befindet sich derzeit noch in Entwicklung und konnte deshalb nicht direkt begutachtet werden [Kra+15]. Das Programm entsteht an der Universität Zürich in Zusammenarbeit mit einem Partner. Um den Anforderungen der Partnerschaft gerecht zu werden, wurde SAHRA als eine Erweiterung der kommerziellen UML/SysML Modellierungssoftware Enterprise Architect konzipiert. Ziel ist, die neuen STPA-Daten eng mit den vorhandenen Anforderungs- und Entwurfs-Daten zu koppeln, um bei Modifikationen Konsistenz zu gewährleisten. Abb. 3.2 zeigt einen in einer frühen Entwicklerversion von SAHRA erstellten Regelkreis [Kra+16]. Für den Regler ”Door Controller” wurde ein Prozessmodell mit drei Prozessmodellvariablen modelliert, die jeweils drei Werte annehmen können. Ein Regelalgorithmus ist leider nicht zu sehen und

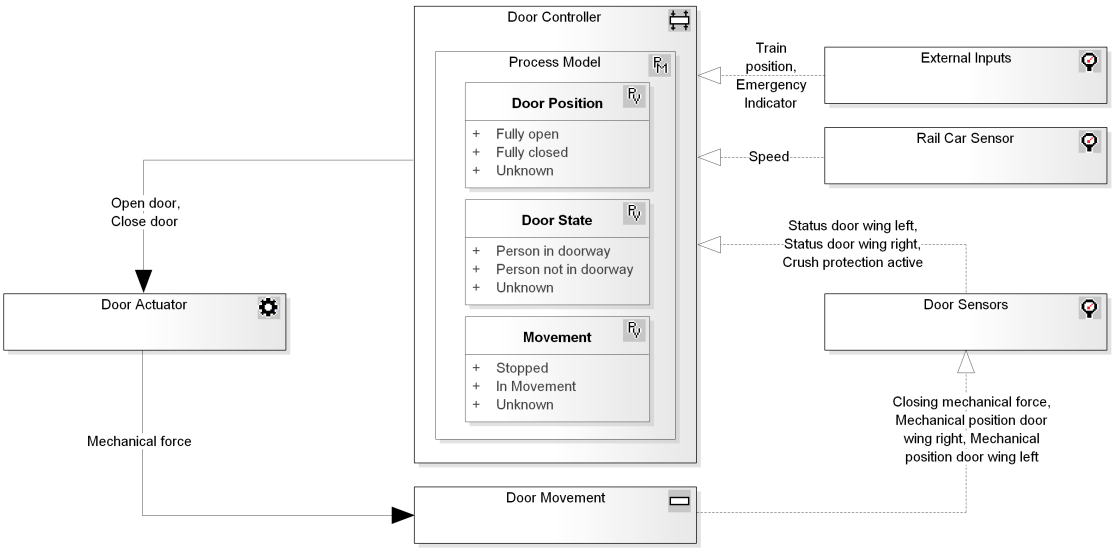


Abbildung 3.2: Regelkreis in SAHRA

auch kontinuierliche Variablen werden nicht demonstriert. Tatsächlich sind in keinem der bisher zu SAHRA veröffentlichten Dokumenten nähere Informationen bezüglich Regelalgorithmen zu finden. Auch wenn die Möglichkeit besteht, dass diese dennoch implementiert sind oder noch implementiert werden, sind die dort verarbeiteten Ideen momentan nicht zugänglich.

3.1.3 RM Studio

Risk Management Studio (RM Studio) ist eine kommerzielle Software, die seit 2008 von Stiki entwickelt wird [Bro+17]. Der Schwerpunkt liegt auf der Dokumentation von Risiken und deren Auswirkungen auf die interne Firmenstruktur. STPA wird als zusätzliche Funktionalität in Form einer allein lauffähigen Erweiterung implementiert. Diese erlaubt die Erstellung der Systemstruktur als Regelkreisdiagramm, die Erfassung von Verlusten, Gefährdungen, unsicheren Regelsignalen und Randbedingungen, sowie deren Verbindung und Kategorisierung. Abb. 3.3 zeigt einen mit RM Studio erstellten Regelkreis. Die Regler wurden hier blau, die Prozesse gelb gefärbt. Die Baumstruktur links ermöglicht den Wechsel zu anderen Sichten. Die Modellierung der Regler ist in RM Studio stark vereinfacht. Während Regelsignale und Feedback mithilfe der Verbindungspfeile eingetragen werden können, ist das Hinzufügen von Regelalgorithmen oder Prozessmodellen nicht direkt möglich. Neben Systemkomponenten und Verbindungen erlaubt RM Studio auch das Hinzufügen von Annotationen. Diese können im Bedarfsfall grundlegende Informationen zu Regelalgorithmus und Prozessmodell in Fließtextform enthalten. Ein explizit dafür vorgesehenes, angepasstes Eingabefeld existiert jedoch nicht. Auch im weiteren Verlauf der Analyse mit RM Studio ist keine Eingabemöglichkeit für Regelalgorithmen oder Prozessmodelle vorhanden. Demnach können auch hier keine Modellierungskonzepte entnommen werden.

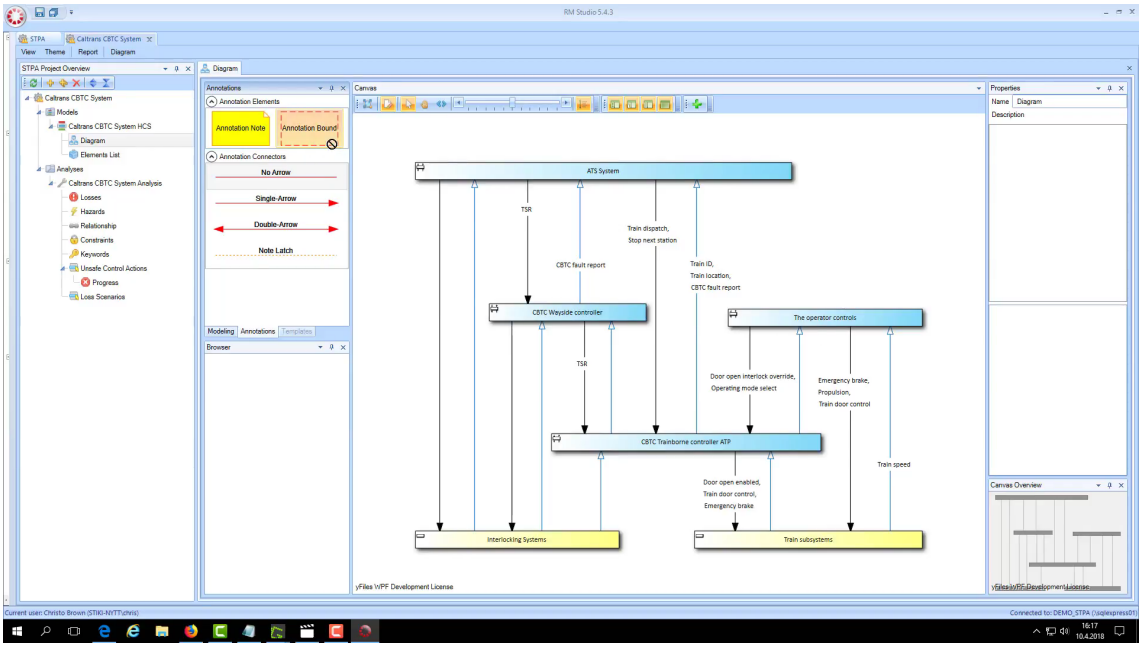


Abbildung 3.3: Benutzeroberfläche von RM Studio

3.1.4 STAMP Workbench

STAMP Workbench wurde im Jahr 2018 von der Information-technology Promotion Agentur Japan (IPA) veröffentlicht und ist kostenlos [I18]. Das Programm erlaubt die strukturierte Bearbeitung der in STPA definierten Schritte, und bietet verschiedene Exportmöglichkeiten an. In Abb. 3.4 ist ein Ausschnitt der in Abb. 2.4 dargestellten Luftfahrtregelstruktur zu sehen, der mit STAMP Workbench angefertigt wurde. Zusätzlich wurde ein Teil des Prozessmodells der Flugverkehrskontrolle erstellt. Dieses besteht in STAMP Workbench aus einer Menge Prozessvariablen, die jeweils eine vordefinierte Zahl konkreter Werte annehmen kann. Die Eingabe eines Regelalgorithmus ist in dieser Sicht nicht direkt möglich. Zwar hat jeder Regler Felder für Verantwortlichkeiten und Beschreibung, die man prinzipiell auch für den Regelalgorithmus verwenden kann, jedoch ist die Form auf einen Fließtext beschränkt. Zudem werden die Felder nicht im Schaubild selbst angezeigt.

STAMP Workbench erlaubt die Deklaration von unsicheren Regelsignalen in einer Tabelle. Abb. 3.5 verdeutlicht dies am Beispiel des Regelsignals ”Starterlaubnis”. Eine Besonderheit der Tabelle ist die Spalte ”CA提供条件”(dt. ”Vorraussetzungen für das Senden des Regelsignals”). Über diese Spalte kann der Nutzer einen grundlegenden Regelalgorithmus definieren. Im Beispiel wurde eine informelle Beschreibung der Zustände einiger Prozessmodellvariablen gegeben. Auch dieses Feld ist nicht mehr als ein Bereich für einen Fließtext. Die Namen der Prozessmodellvariablen und dessen Werte müssen aus dem Regelkreis abgeschrieben und händisch zu einer booleschen Formel verknüpft werden. Überdies ist das Feld nur im Zusammenhang mit den unsicheren Regelsignalen sichtbar. Ein Einsehen ist weder in einer eigenen, abgetrennten Ansicht, noch während der Identifikation der Verlustszenarien möglich. Abschließend muss noch angemerkt werden, dass die Regelsignale in STAMP Workbench binär sind: Das Signal wird gesendet oder nicht gesendet. Fälle, in denen das Signal mehr als zwei Zustände annehmen oder kontinuierlich sein muss, sind somit nicht modellierbar.

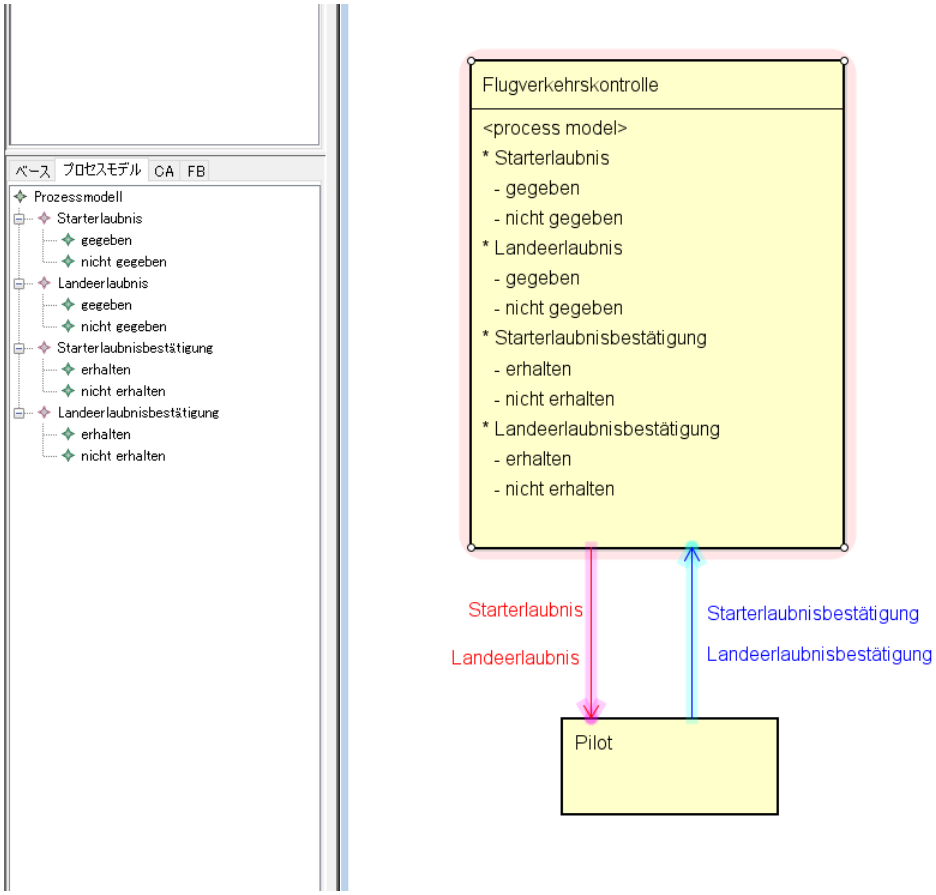
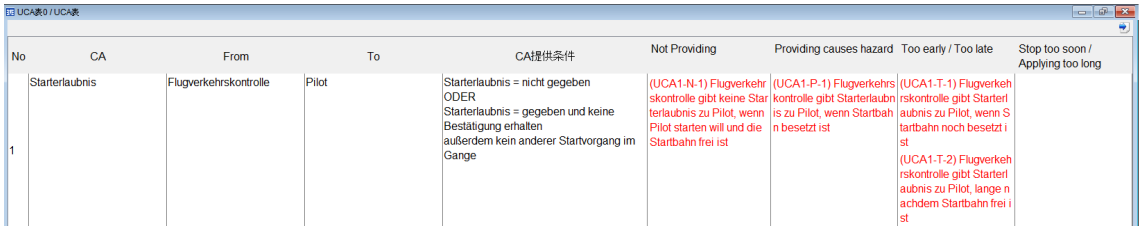


Abbildung 3.4: Regelkreis in STAMP Workbench



No	CA	From	To	CA提供条件	Not Providing	Providing causes hazard	Too early / Too late	Stop too soon / Applying too long
1	Starterlaubnis	Flugverkehrskontrolle	Pilot	Starterlaubnis = nicht gegeben ODER Starterlaubnis = gegeben und keine Bestätigung erhalten außerdem kein anderer Startvorgang im Gange	(UCA1-N-1) Flugverkehrskontrolle gibt keine Starterlaubnis zu Pilot, wenn Pilot starten will und die Startbahn frei ist	(UCA1-P-1) Flugverkehrskontrolle gibt Starterlaubnis zu Pilot, wenn Startbahn besetzt ist	(UCA1-T-1) Flugverkehrskontrolle gibt Starterlaubnis zu Pilot, wenn Startbahn noch besetzt ist (UCA1-T-2) Flugverkehrskontrolle gibt Starterlaubnis zu Pilot, lange nachdem Startbahn frei ist	

Abbildung 3.5: Unsichere Regelsignale in einer Tabelle in STAMP Workbench

3.1.5 SafetyHAT

Safety Hazard Analysis Tool (SafetyHAT) wurde 2014 von Becker und Hommes vorgestellt [Hom14][BEH14]. Das Programm legt Wert auf schnelle Datenverarbeitung, umfassende Konsistenz beim Erstellen und Löschen von Daten, sowie durchgehende Verknüpfung der einzelnen Produkte. Der Prozess in SafetyHAT unterscheidet sich von STPA in einigen Belangen. Zunächst durchläuft der Sicherheitsingenieur acht Phasen, die die vier in STPA üblichen Phasen weiter aufgliedern. Darüber hinaus besitzt das Programm zwei zusätzliche Kategorien für unsichere Regelsignale, die außerdem noch ergänzt werden können. Abb. 3.6 zeigt die dritte Phase ”Control Action Input Form”. Hier werden alle Regelsignale definiert. Anders als in STAMP Workbench funktioniert dies nicht in

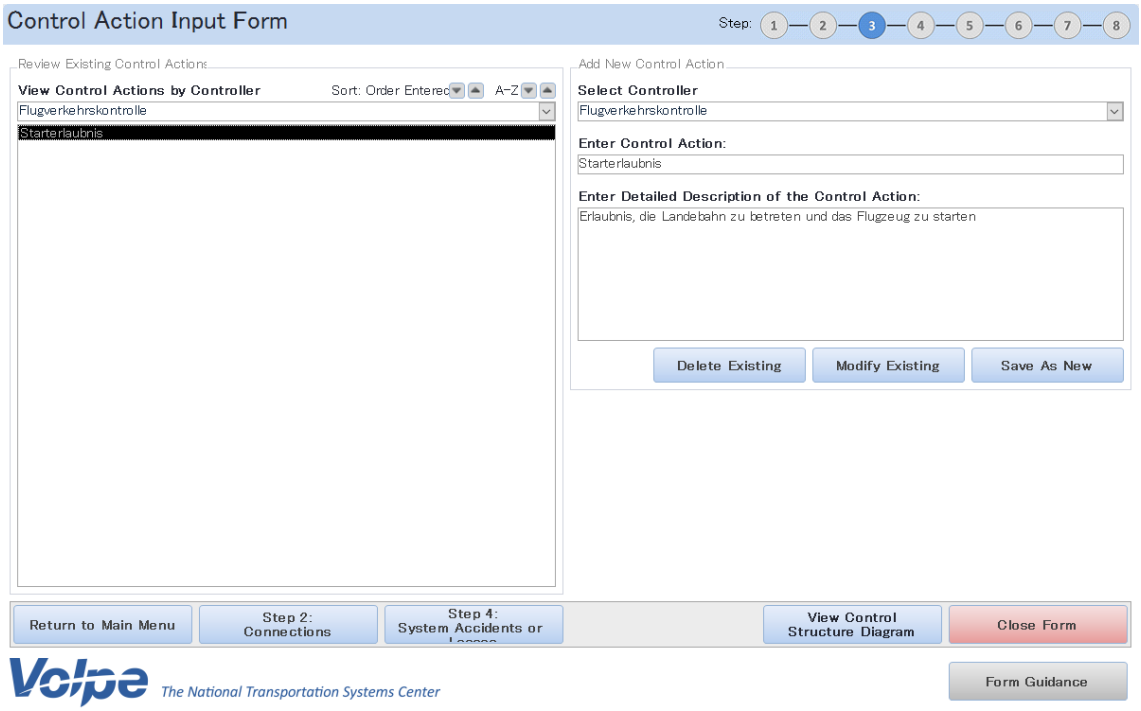


Abbildung 3.6: Benutzeroberfläche von SafetyHAT

Form eines Diagramms, sondern einer Liste. Das Festlegen des Signalempfängers ist nicht möglich. SafetyHAT besitzt keinerlei Funktionalität hinsichtlich Prozessmodell oder Regelalgorithmus. Folglich können auch diesem Programm keine Konzepte entnommen werden.

3.1.6 An STPA Tool

An STPA Tool wurde an der dritten STAMP-Konferenz 2014 an der Massachusetts Institute of Technology von Suo und Thomas präsentiert. Da das Programm nie veröffentlicht wurde, beruht diese Analyse auf den verfügbaren Dokumenten. Ebenso wie die meisten anderen Programme enthält An STPA Tool Funktionen zur Auflistung von Gefährdungen, Erstellung der Regelkreisstruktur, und Identifikation von unsicheren Regelsignalen. Die potentiell unsicheren Signale werden dabei automatisch aus der Regelkreisstruktur generiert. Der Nutzer muss nur angeben, welche davon tatsächlich zu Gefährdungen führen können. Diese Angaben können auf Konflikte überprüft werden, z. B. wenn unter den gleichen Ausgangsbedingungen sowohl das Senden als auch Nichtsenden zu einer Gefährdung führt. Als Exportformat wird `.xml` verwendet. In Abb. 3.7 ist eine mit An STPA Tool konstruierte Regelkreisstruktur zu sehen, die eine Türsteuerung darstellt. Darüber befinden sich einige Gefährdungen. Ähnlich wie in STAMP Workbench kann dem Regler ein Prozessmodell hinzugefügt werden, welches aus einer Menge Prozessmodellvariablen besteht. Auch hier besitzen diese eine Liste konkreter Werte, die sie annehmen können. Die Eingabe eines Regelalgorithmus ist in dieser Sicht nicht möglich.

Während der in Abb. 3.8 gezeigten Phase werden die unsicheren Regelsignale bestimmt. Hierbei werden alle in der Regelkreisstruktur möglichen Signale angezeigt und können mit zutreffenden Gefährdungen verbunden werden. Daraufhin wird im unteren Teil des Fensters ein Modell des

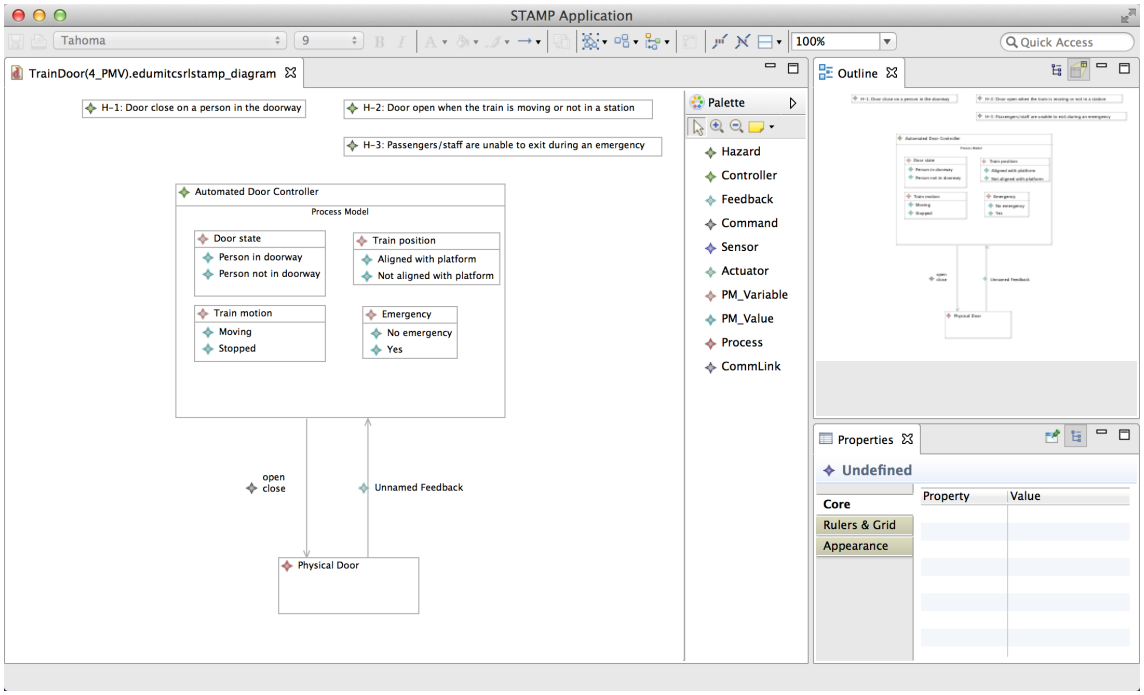


Abbildung 3.7: Regelkreiserstellung in An STPA Tool

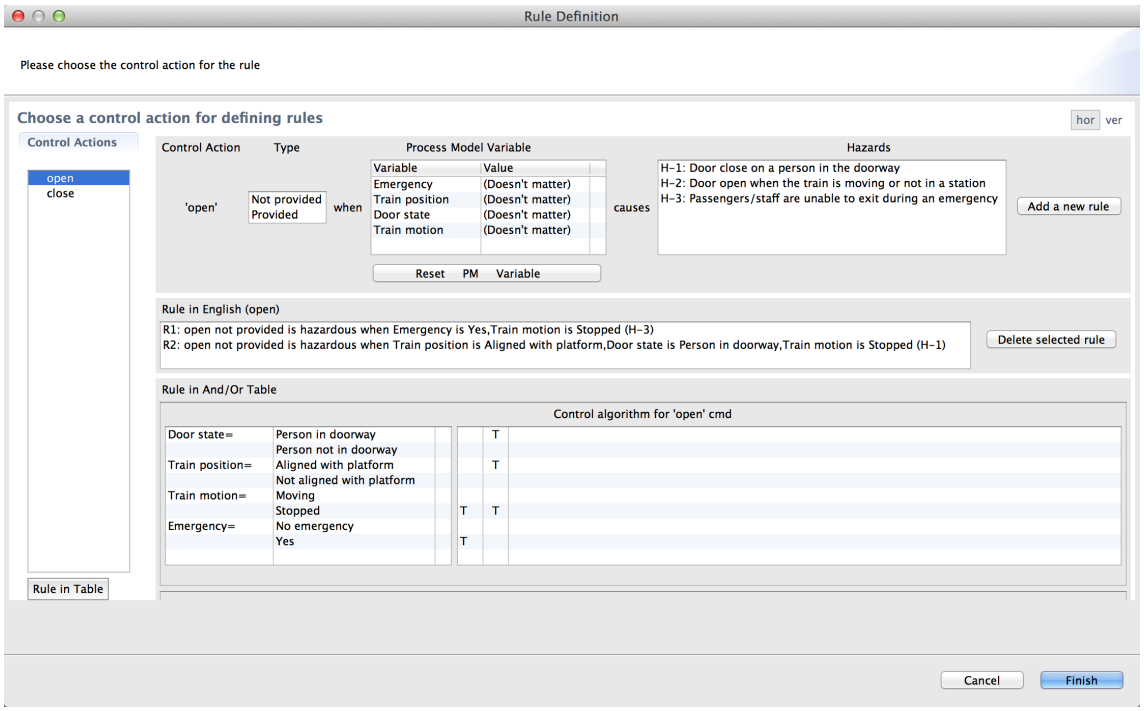


Abbildung 3.8: Kontrollalgorithmen in An STPA Tool

notwendigen Regelalgorithmus automatisch generiert. T steht für *true* und zeigt eine notwendig wahre Bedingung an. Die Zeilen einer Spalte sind mit UND, die Spalten selbst mit ODER verknüpft. In der Abbildung zu sehen ist demnach die Anweisung:

```
IF (trainMotion=stopped AND emergency=yes)
OR (doorState=personInDoorway AND trainPosition=alignedWithPlatform
AND trainMotion=stopped)
THEN open()
```

Ein manuelles Nachkonfigurieren des Algorithmus ist nicht möglich. Ebenso wenig können kontinuierliche Werte verarbeitet werden [ST14]. Für eine ausführlichere Erläuterung zu Entscheidungstabellen sei der Leser auf Abschnitt 3.2.2 verwiesen.

3.1.7 Fazit

In diesem Abschnitt wurden sechs STPA-Programme untersucht. Drei davon besitzen keine (veröffentlichte) Funktion zur Modellierung von Regelalgorithmen. Die anderen drei Programme unterscheiden sich in ihren Vorgehensweisen. STAMP Workbench hat für eine als Fließtext verfasste Beschreibung der Regelsignalvorbedingungen eine Tabellenspalte vorgesehen. An STPA Tool generiert einen an Randbedingungen erinnernden Regelalgorithmus automatisch aus den unsicheren Regelsignalen. SpecTRM bietet die Eingabe von Vorbedingungen und Regelsignalinhalten für sowohl diskrete, als auch für kontinuierliche Variablen an. Zusätzlich können Modi definiert werden. Im nächsten Abschnitt werden weitere Modellierungsmethoden aus existierenden STPA-Analysen herausgearbeitet.

3.2 Regelalgorithmen in existierenden STPA-Analysen

In den letzten Jahren ist der Bekanntheitsgrad von STPA stark gestiegen. Damit einher nahm auch die Zahl der mit STPA durchgeführten Analysen zu [LM16][LT13]. Viele der Analysen sind in Form von Artikeln, Berichten und Abschlussarbeiten dokumentiert. Umfangreiche Dokumentationen enthalten auch Details zur Vorgehensweise bei der Modellierung der Regelalgorithmen. Im Allgemeinen sind die Modelle so gewählt, dass sie zum Reglertypen passen und auch von außenstehenden Personen nachvollzogen werden können. Bei der Entwicklung eines Konzeptes, das die Konstruktion ebenjener Modelle zum Ziel hat, sind die bereits existierenden Vorgehensweisen in hohem Maße zu berücksichtigen. Demgemäß werden in diesem Abschnitt eine Auswahl STPA-Analysen untersucht. Besonderer Augenmerk wurde darauf gelegt, eine möglichst große Vielfalt in Bezug auf die analysierten Systeme zu erzielen.

Rejzek und Hilbes [RH14] und Thomas et al. [Tho+12] wandten STPA an jeweils einem Kernkraftwerk mit Druckwasserreaktor an. In ersterer Analyse wurde der Prozess dabei minimal abgewandelt, um bereits mit anderen Methoden identifizierte Gefährdungen zu integrieren. Rejzek et al. [Rej+12] und Antoine [Ant13] benutzten STPA, um Protonentherapiesysteme zu untersuchen. Hier wurden sowohl der Teilchenbeschleuniger selbst, als auch das medizinische Personal in das Modell aufgenommen. Bei Sulaman et al. [Sul+14] und Sulaman et al. [Sul+19] wurde eine STPA-Analyse bei einem Notbremsassistenten eines Personenkraftwagens durchgeführt und deren Ergebnisse mit FMEA (s. Abschnitt 2.2) verglichen. Martínez [Mar15] testeten STPA an einer Servolenkung. Im

Rahmen dieses Tests wurden auch das Management und die Entwicklung betreffende Komponenten modelliert. Placke [Pla14] nutzten STPA zur Analyse drei verschiedener Techniken im Automobilbereich: Automatic Vehicle Hold (AVH), Engine Stop-Start (ESS) and Adaptive Cruise Control (ACC). Auch bei Abdulkhaleq und Wagner [AW13], Oscarsson et al. [Osc+16] und Hommes [Hom12] lag der Fokus auf ACC. Die Ergebnisse wurden teilweise an speziell für die Studie zusammengebauten Konzeptfahrzeugen verifiziert. Bagschik et al. [Bag+17] wandten STPA an einem fahrerlosen Baustellenfahrzeug an, das bei mobilen Baustellen auf der Autobahn als Schlusslicht dem Konvoi folgt und anderen Fahrzeugen den Beginn der Baustelle signalisiert. Merladet et al. [Mer+19] nutzten STPA, um potentielle Risiken bei Raketenstarts zu identifizieren. Dabei wurden auch Regulierungen und Auftraggeber in das Systemdiagramm eingearbeitet. Ein ähnliches Thema findet sich in Dunn [Dun13], wo die automatische Inbetriebnahme eines Satelliten nach Erreichen des Orbits mit STPA analysiert wurde. Bei Dong [Don13] und Takano et al. [Tak+17] wurden STPA-Analysen an Schienenverkehrssystemen durchgeführt. Letztere Studie fokussierte sich auf Signalanlagen, während erstere das komplette soziotechnische System in Betracht zog. Chatzimichailidou et al. [Cha+17] und Pappot und de Boer [PB15] testeten STPA an ferngesteuerten Drohnen, wobei externe Faktoren jedoch nicht in Betracht gezogen wurden. Plioutsias und Karanikas [PK15] nutzten STPA zur Analyse des Kampfpilotentrainingsprogramms der südeuropäischen Air Force in Bezug auf das F-16 Fighting Falcon Kampfflugzeug. Bei Schmid und Stanton [SS18] lag der Hauptaugenmerk auf den seit der Jahrhundertwende an Häufigkeit zugenommenen Laserangriffe auf Passagierflugzeuge während Start- und Landevorgang. Das Ziel war insbesondere die Ermittlung geeigneter Gegenmaßnahmen zu derartigen Angriffen. Revell et al. [Rev+19] wandten STPA an den Bestimmungen zur Einteilung der Flugbesatzung in einem Druckverlustszenario an. Und schließlich benutzten Alemzadeh et al. [Ale+14] STPA in der roboterassistierten Chirurgie und Adesina et al. [Ade+17] zur Analyse des Überwachungsprozesses von Arzneimittelsicherheit.

Bei vielen der oben aufgeführten Analysen sind die Regelalgorithmen nur unzureichend oder nicht dokumentiert. Im folgenden werden daher nur auf jene Schriftstücke eingegangen, die hinsichtlich dessen umfassende Angaben vorweisen.

3.2.1 Funktionsgraphen

In mehreren Analysen wurden Szenarien betrachtet, bei denen es erforderlich war, sowohl einige Prozessmodellvariablen als auch einige Regelsignale als kontinuierliche Variablen zu modellieren. Solche Szenarien enthalten oft Fälle, in denen der Regelalgorithmus eine der kontinuierlichen Prozessmodellvariablen auf eine der kontinuierlichen Regelsignale abbilden muss. Des Weiteren existieren Fälle, in denen der Regelalgorithmus ein binäres Signal sendet, dessen Wert von einer Ungleichung mit ein oder zwei kontinuierlichen Prozessmodellvariablen abhängt. Hier gilt ohne Beschränkung der Allgemeinheit, dass der Regler das Signal 1 sendet, genau dann, wenn eine Ungleichung der Form $f(x) < y$ (x, y: Prozessmodellvariablen) zu *wahr* ausgewertet wird. Ansonsten wird 0 bzw. nichts gesendet.

Ein Beispiel für ein Szenario, in dem derartige Regelalgorithmen modelliert werden müssen, ist der Bremsvorgang eines Zuges vor einer roten Signalanlage. Zu einem Zusammenstoß und damit zu einem Verlust kann es kommen, wenn die Anlage bei rot überfahren wird. Für die Vermeidung dieser Gefährdung ist zunächst der Zugführer zuständig. Seine Aufgabe ist es, den Zug frühzeitig abzubremsen, sodass er vor der Signalanlage zum Stehen kommt. Im Falle, dass der Zug zu schnell ist, muss die Notbremse aktiviert werden. Hierfür verantwortlich ist nicht der Zugführer, sondern das

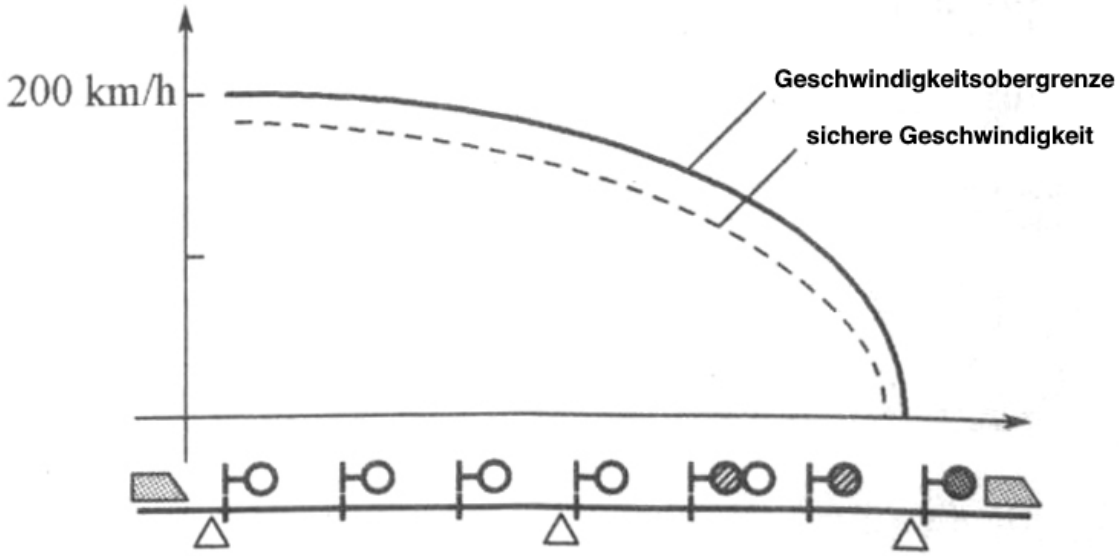


Abbildung 3.9: Beispielfunktionsgraph (Englisches Original in Dong [Don13])

Automatic Train Protection (ATP) System. Dieses hat zwei hier relevante Variablen in seinem Prozessmodell: die Geschwindigkeit des Zuges und die noch verbleibende Distanz zur Signalanlage. Je näher der Zug der Signalanlage kommt, desto geringer wird die erlaubte Geschwindigkeit. Das ATP sendet ein Bremssignal an die Notbremse, wenn diese Geschwindigkeit überschritten wird. Intern wertet der Regelalgorithmus des ATP eine Ungleichung mit den zwei Variablen `geschwindigkeit` und `distanz` aus.

In Dong [Don13] wurde ein solcher Regelalgorithmus mit einem Funktionsgraphen modelliert. Dieser ist in Abb. 3.9 zu sehen. Die x-Achse zeigt die Distanz, die y-Achse die Geschwindigkeit an. Im Graphen sind zwei Linien zu erkennen. Die gestrichelte Linie zeigt den Geschwindigkeitsverlauf des Zuges bei einem regulären Bremsmanöver. Die durchgezogene Linie ist die Geschwindigkeit, dessen Überschreitung zu einer Betätigung der Notbremse durch das ATP führt.

Es ist offensichtlich, dass ein Funktionsgraph, wie er hier demonstriert wurde, ebenso benutzt werden kann, um die Beziehung zwischen einer kontinuierlichen Prozessmodellvariablen und einem kontinuierlichem Regelsignal darzustellen. Gleichfalls sind Ungleichung der Form $f(x) < 0$, d. h. Ungleichungen mit nur einer Variablen möglich. Es gibt mehrere Möglichkeiten, das Konzept für noch speziellere Anwendungsfälle zu erweitern. Beispielsweise können Ungleichungen der Form $f(x) < y < g(x)$ durch Zeichnen zweier Kurven dargestellt werden, deren Zwischenraum die Lösungsmenge enthält. Außerdem ist es möglich, drei kontinuierliche Werte in einem dreidimensionalen Graphen anzuzeigen.

3.2.2 Entscheidungstabellen

Eine Entscheidungstabelle ist eine Tabelle, in deren Vorspalte die Namen von Bedingungen eingetragen sind. Jede Spalte repräsentiert eine Regel, die die einzelnen Bedingungen logisch verknüpft, indem in die jeweilige Zeile eine der drei Bezeichnungen `WAHR`, `FALSCH` oder $*$ (egal) eingetragen

Abschluss =	Abitur	W	W	W
	mittlere Reife			
beherrschte Sprachen =	Javascript	W		W
	PHP		W	W
Erfahrung =	viel	W	W	
	wenig			W
	keine			

Tabelle 3.1: Entscheidungstabelle für eine Stellenbewerbung

wird. Die Bedingungen sind zeilenweise `UND`-verknüpft, während die Ergebnisse der einzelnen Regeln spaltenweise `ODER`-verknüpft sind. Das Gesamtergebnis bestimmt den Ausgang der JA-NEIN-Entscheidung, für die die Tabelle steht. Beispielsweise kann anhand dem Vorhandensein mehrerer Qualifikationen entschieden werden, ob ein Bewerber eingestellt wird oder nicht. Die Entscheidungstabelle in solch einem Szenario könnte so aussehen wie in Tabelle 3.1. Anhand des Felds ”Erfahrung” wird hier die Modellierung einer nicht-binären Bedingung demonstriert: Die möglichen Werte bzw. Zustände werden in der nächsten Spalte aufgelistet. Erst danach folgen die Regeln, wobei `WAHR` mit W abgekürzt ist. Sind alle Werte der Bedingungen aufgelistet, entfällt auch die Notwendigkeit, Bedingungen mit `FALSCH` zu markieren. In An STPA Tool sind Regelalgorithmen als Entscheidungstabellen implementiert (s. Abschnitt 3.1.6). Viele STPA-Analysen machen Gebrauch der Entscheidungstabelle, um Regelalgorithmen, Zustandsübergänge und Ähnliches darzustellen [Fle+12][Pla14][Fle+13].

Ein Nachteil der Entscheidungstabellen besteht darin, dass kontinuierliche Variablen nicht ohne weiteres integriert werden können. Leveson und Thomas [LT13] legen diesbezüglich einen Lösungsansatz am Beispiel eines Reisezugs dar: Die Geschwindigkeit des Zugs ist offensichtlich eine kontinuierliche Variable, da die sie unendlich viele unterschiedliche Werte annehmen kann. Gleichwohl ist es nicht nötig, sich mit all diesen Werten – nicht einmal vielen dieser Werte – zu befassen. Für die Risikoanalyse der Zugtüren z. B. reicht das Wissen darüber aus, ob der Zug hält (`geschwindigkeit = 0`) oder sich bewegt (`geschwindigkeit > 0`). Durch sorgfältige Diskretisierung der Prozessmodellvariablen kann auf diese Weise die Komplexität der Entscheidungstabellen und der darauf aufbauenden Analyse signifikant reduziert werden. Zusätzlich ermöglicht die Diskretisierung das nachträgliche Aufspalten oder Zusammenführen von Zuständen innerhalb automatisierter Prozesse (z. B. ”Zug in Bewegung” ⇒ ”Zug fährt langsam” und ”Zug fährt schnell”). Ebenso ist die Eliminierung überflüssiger Tabelleneinträge möglich. Während Leveson und Thomas [LT13] den Detailgrad der diskreten Zustände als weniger wichtig erachten, messen sie der Vollständigkeit derer eine hohe Bedeutung bei: Die Zustände einer Zugtür mit `offen` und `geschlossen` zu beschreiben, scheint zunächst vollständig zu sein. Bei näherer Betrachtung fällt jedoch auf, dass es noch einen weiteren, potentiell sicherheitskritischen Zustand `teilweise offen` gibt.

Tatsächlich ist die Diskretisierung der Prozessmodellvariablen und Regelsignale eine häufig angewandte Technik. Bagschik et al. [Bag+17] benutzen zwischen drei und vier Zustände für Variablen wie `Betriebsmodus`, `Umgebung` und `Hindernis präsent`. H17 definieren Modi für mehrere Regler und Umgebungsvariablen. Fleming et al. [Fle+13] diskretisieren u. a. Wetter-, Geschwindigkeits- und Distanzwerte.

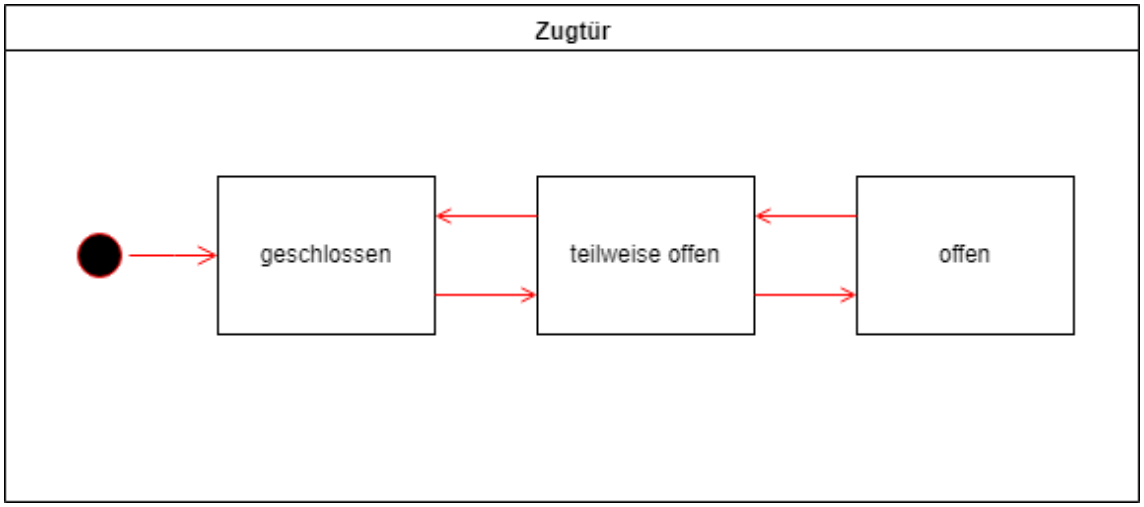


Abbildung 3.10: UML-Zustandsmaschinendiagramm

Variablen, für die zwei unterschiedliche Zustände definiert sind, haben bei einem Zustandswechsel immer nur eine Möglichkeit, d. i. in den anderen Zustand überzugehen. Bei mehr als zwei möglichen Zuständen sind den Übergängen manchmal Beschränkungen auferlegt. So muss etwa eine Zugtür von dem geschlossenen Zustand erst in den halb offenen Zustand gehen, bevor sie vollständig offen sein kann. Hashimoto [H17] modelliert dieses Verhalten mit UML-Zustandsmaschinendiagrammen. In dieser Art von Diagramm sind alle Zustände eines (Unter-)Systems als Kästen dargestellt. Erlaubte Zustandsübergänge werden mit Pfeilen angezeigt. Ein ausgefüllter Kreis spezifiziert den Startzustand. Abb. 3.10 demonstriert das Zustandsmaschinendiagramm an der zuvor behandelten Zugtür. Man merke, dass keine Pfeile zwischen den Zuständen `geschlossen` und `offen` gezeichnet sind. Eine formale Spezifikation von Zustandsmaschinendiagrammen ist in *OMG Unified Modeling Language (OMG UML)* zu finden. Dort sind weitere Elemente definiert, mit denen u. a. auch die Modellierung von Übergangsbedingungen möglich ist. Zustandsmaschinendiagramme können Entscheidungstabellen vollständig ersetzen, wenn die Zustände direkt mit den Regelsignalen korrelieren und alle Übergangsbedingungen angegeben sind. In [H17] ist dies in Teilen des Systems zu beobachten.

3.2.3 Boolesche Schaltkreise

Entscheidungstabellen bilden eine Menge binärer Eingabesignale auf ein binäres Ausgabesignal ab. Dabei werden immer zuerst Teilmengen der Eingabesignale mit `UND`, und anschließend diese Teilmengen mit `ODER` verknüpft. Im Kern sind sie damit aussagenlogische Formeln in *disjunktiver Normalform* (DNF). Die Beschränkung auf diese Form kann in gewissen Fällen zu einer Explosion der benötigten Spalten führen: Man betrachte den Fall, bei dem sich ein Druckventil nur öffnet, wenn mindestens einer von drei *Temperatursensoren*, einer von drei *Drucksensoren* und einer von drei *Wasserstoffsensoren* ein entsprechendes Signal sendet. Dies kann mit folgender Formel übersichtlich ausgedrückt werden:

$$(T_1 \vee T_2 \vee T_3) \wedge (D_1 \vee D_2 \vee D_3) \wedge (W_1 \vee W_2 \vee W_3)$$

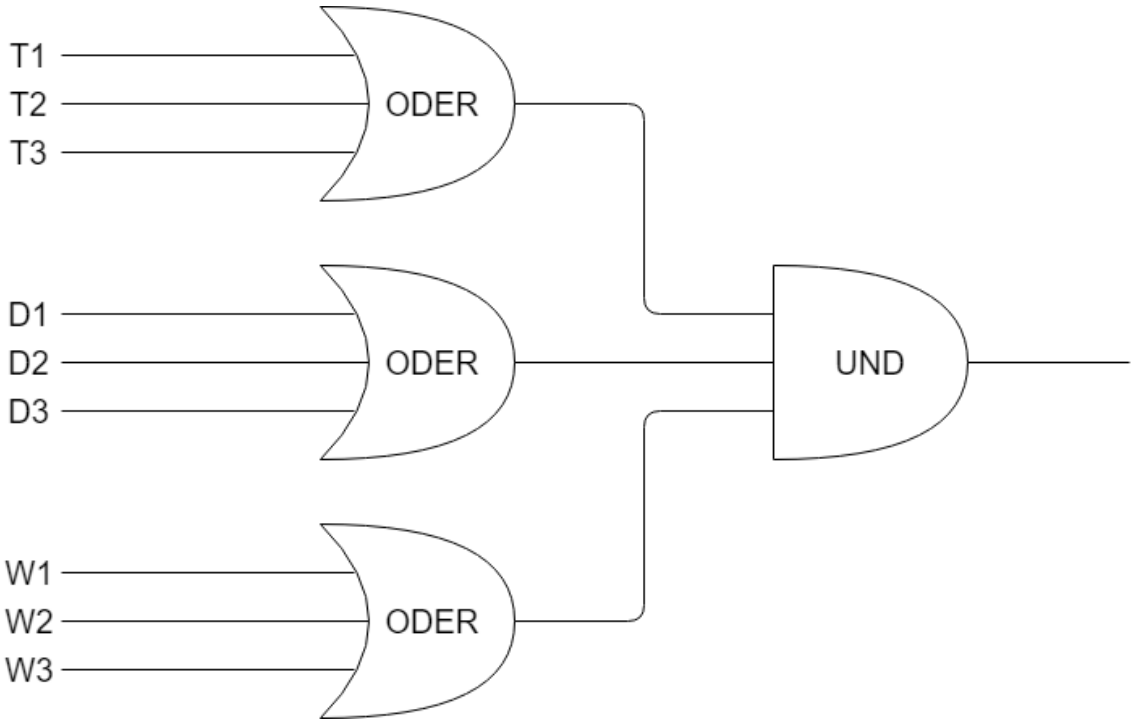


Abbildung 3.11: Boolescher Schaltkreis

Eine äquivalente Entscheidungstabelle würde für die selbe aussagenlogische Formel 27 Spalten benötigen:

$$(T_1 \wedge D_1 \wedge W_1) \vee (T_1 \wedge D_1 \wedge W_2) \vee (T_1 \wedge D_1 \wedge W_3) \vee (T_1 \wedge D_2 \wedge W_1) \vee \ldots$$

Ein weiterer Nachteil von Entscheidungstabellen besteht darin, dass nur ein Ausgabesignal pro Tabelle dargestellt werden kann. Dies kann beim Vergleich mehrerer Signale der Übersicht abträglich sein.

Leveson et al. [Lev+14] umgehen diese Nachteile durch Verwendung boolescher Schaltkreise. Ebenso wie Entscheidungstabellen werden boolesche Schaltkreise benutzt, um aussagenlogische Formeln zu repräsentieren. Eingabesignale, Ausgabesignale und Zwischenergebnisse werden dabei als Linien dargestellt, Verknüpfungen sind durch charakteristische Formen ausgedrückt. In welcher Reihenfolge und auf welche Weise die Eingabesignale verknüpft werden, unterliegt keiner Beschränkung. Bei den Verknüpfungen sind neben `UND`, `ODER` und `NICHT` u. a. auch `XOR`, `NICHT UND` und `NICHT ODER` möglich, wodurch selbst komplexe Formeln kompakt dargestellt werden können. Des Weiteren können in ein und den selben Schaltkreis mehrere Ausgabesignale eingefügt werden. Dabei ist auch die Wiederverwendung von Zwischenergebnissen kein Problem. Abb. 3.11 zeigt einen booleschen Schaltkreis für das zuvor beschriebene Druckventil. Die drei `ODER`-Verknüpfung sammeln zunächst die Signale jeweils dreier gleichartiger Sensoren. Ist jeweils eines oder mehr davon `WAHR`, wird auch die nachfolgende `UND`-Verknüpfung `WAHR`.

3.2.4 Formeln

Um dem Sicherheitsingenieur zusätzlich die Möglichkeit zu geben, die Bedingungen der einzelnen Regelsignale in Isolation betrachten zu können, haben Leveson et al. [Lev+14] diese auch als boolesche Formeln aufgeschrieben. Während komplexe Formeln schnell an Lesbarkeit verlieren, können insbesondere simple Formeln deutlich platzsparender als Entscheidungstabellen oder Schaltkreise sein. Eine weitere positive Eigenschaft liegt darin, dass die Variablen nicht zwangsweise diskretisiert werden müssen: Es ist durchaus möglich, die kontinuierlichen Eingabewerte mit arithmetischen Operatoren zu verknüpfen oder in Funktionen zu verarbeiten.

3.2.5 Fließtextbeschreibungen

Zu guter Letzt muss auch die Option, Regelalgorithmen innerhalb eines Fließtextes zu beschreiben, Erwähnung finden. Diese Option haben u. a. Sulaman et al. [Sul+19] gewählt. Der Vorteil eines Textes liegt in seiner Informalität. Fokus und Detailgrad der Beschreibung können den Bedürfnissen entsprechend angepasst werden, und der Leser wird nicht aus dem Lesefluss gebracht. Für die Erläuterung allzu komplexer Algorithmen sind Fließtexte nicht geeignet.

3.2.6 Fazit

Die Recherche hat gezeigt, das Regelalgorithmen auf vielerlei Weise modelliert werden können. Jeder der Ansätze hat unterschiedliche Ausprägungen und dadurch das Potential, sich in bestimmten Situationen als der am besten geeignetste herauszustellen. Funktionsgraphen sind eine sehr anschauliche Art, die Beziehungen zwischen zwei oder drei kontinuierlichen Variablen zu zeigen. Mehr als drei Variablen sind jedoch problematisch. Bisher erlaubt keines der STPA-Programme das Zeichnen von Graphen. Entscheidungstabellen haben keine solche Beschränkung und können automatisch reduziert werden. Sie erfordern aber die Diskretisierung der Variablen. An STPA Tool und SpecTRM besitzen Funktionen zur Erstellung von Entscheidungstabellen. In SpecTRM können zusätzlich Modi definiert werden, die mögliche Übergänge der diskretisierten Zustände anzeigen. Boolesche Schaltkreise können mehrere Regelsignale im selben Diagramm darstellen. Sie sind bisher in keinem Programm implementiert. Formeln und Fließtexte sind zwei weitere Möglichkeiten zur Modellierung der Regelalgorithmen, die im Beschreibungsfeld des Reglers unterkommen können, was in jedem der betrachteten Programme möglich ist. STAMP Workbench bietet sogar ein eigenes Feld dafür an. Hilfestellung bei der Erstellung, Minimierung und Darstellung von Formeln wird nicht geboten.

Ein ausführlicher Vergleich der verschiedenen Modellierungsmethoden wird in Kapitel 4 durchgeführt. Zuvor ist jedoch die Frage zu klären, inwiefern sich die zu modellierenden Regelalgorithmen unterscheiden, und wie diese Unterschiede in das Modellierungskonzept eingearbeitet werden können. Insbesondere die Algorithmen zweier Reglerklassen sollen hierbei analysiert werden: menschliche und technische Regler. Ein wichtiges Ziel von STAMP war es, Systeme auf eine Weise zu betrachten, die keiner Unterscheidung von technischen, menschlichen und soziologischen Körpern bedarf [Lev04]. Nichtsdestoweniger muss die Möglichkeit in Betracht gezogen werden, dass Unterschiede existieren, die die Modellierung der Regelalgorithmen betreffen. Zu diesem Thema liegen bereits zahlreiche Forschungsartikel vor. Diese werden im folgenden Abschnitt zusammengefasst.

3.3 Vergleich menschlicher und technischer Regler

Unter den Begriff ”Technischer Regler” fallen eine große Zahl sehr unterschiedlicher Konstrukte. Sowohl primitive Thermostate als auch hochkomplexe Autopiloten zählen dazu. Trotzdem gibt es viele gemeinsame Eigenschaften. Technische Regler haben eine feste Schnittstelle für Eingaben, die sie nicht selbstständig erweitern können. Somit können sie z. B. kein Feedback von Sensoren bekommen, mit denen sie nicht verbunden sind. Die an der Schnittstelle ankommenden Daten werden unverändert in den Speicher geschrieben oder über eine fest vorgegebene Abbildung konvertiert. Der Speicher ist zu Beginn des Betriebs leer. Die enthaltenen Datenstrukturen sind von Menschen entworfen worden und bleiben in ihrem Grundprinzip unverändert. Der Regelalgorithmus trifft Entscheidungen ausschließlich anhand der im Speicher befindlichen Prozessvariablen und Informationen anderer Regler. Er ist statisch und kann sich nicht selbstständig anpassen. Unter den gleichen Vorbedingungen trifft er immer die gleiche Entscheidung, d. h. das Regelsignal hat exakt den selben Inhalt. Technische Regler zögern nicht.

In STAMP, wie es von Leveson vorgestellt wurde, können Regler, über die diese Annahmen gemacht werden können, mit Leichtigkeit modelliert werden [Mon16]. Die so entstandenen Modelle haben sich bereits in der Praxis bewährt [LT13][Sul+19][SS18] und gezeigt, dass sich STAMP gut für technische Regler eignet. Bei der Darstellung der Regelalgorithmen kam eine Vielfalt von Konzepten zum Einsatz. Demgegenüber werden bei nicht-technischen Reglern die Regelalgorithmen oft nicht explizit modelliert. Eine Ausnahme ist Thornberry [Tho14]), der Entscheidungstabellen verwendet. Dennoch ist klar, dass menschliche Regler eine andere Sichtweise erfordern.

Menschen unterscheiden sich von Maschinen stark [Fit51]. Von den oben aufgeführten Annahmen über technische Regler gelten bei Menschen keine. Sowohl ihre Prozessmodelle, als auch ihre Regelalgorithmen übertreffen Maschinen an Komplexität bei weitem. Darüber hinaus können sie sich ihrer Umwelt dynamisch anpassen. Die Prozessmodelle menschlicher Regler werden in der Literatur oft ”Mentale Modelle” genannt, während Regelalgorithmen mit ”Entscheidungsprozesse” umschrieben werden. Diese Arbeit verwendet jedoch im Sinne der Konsistenz auch hiernach die ursprünglichen Begriffe.

Bereits in Leveson [Lev11] wird auf die komplexe Natur menschlicher Regler hingewiesen. Leveson erkennt, dass Menschen nicht immer den vorgesehenen Prozeduren folgen, und das sogar einer ihrer Vorteile ist, da sie aufgrund ihrer Flexibilität und Fähigkeit, sich an wandelnde Umstände anzupassen, eingesetzt werden. Dadurch gleichen sie falsche Annahmen der Ingenieure aus. Menschliches Versagen ist eine natürliche Folge davon. In Leveson und Thomas [LT13] wird außerdem darauf hingedeutet, dass in STPA Fehler von Mensch und Maschine ähnlich behandelt werden, weil Maschinen dazu gemacht sind, Menschen zu ersetzen. Dennoch gibt es wichtige Unterschiede, an dessen Umsetzung in STPA noch gearbeitet wird.

Ein Beispiel in Leveson et al. [Lev+14] beschreibt die Anforderung an einen Piloten, die Bremse manuell zu steuern, falls die Automatik versagt. Eine solche Anforderung lässt sich offensichtlich nicht auf die selbe Weise erzwingen, wie es bei einer physischen Komponente möglich wäre. Sie kann aber in Prozeduren, Trainings und Kontrollen eingearbeitet werden und bei Offenbarwerden von Schwierigkeiten eine Designänderung der Bremse bewirken.

Pawlicki et al. [Paw+16] geben zu Bedenken, dass die Entscheidungen menschlicher Regler stark von den benutzten Gerätschaften und der Umwelt abhängt. Beispielsweise erstellt ein Medizinphysiker einen Behandlungsplan basierend auf seinem Training, seiner Erfahrung, Informationen über den

Patienten und den verfügbaren medizinischen Geräten. Feedback wird während oder nach der Erstellung des Plans bereitgestellt, und aktualisiert das Prozessmodell des Medizinphysikers, um den aktuell geregelten Prozess widerzuspiegeln. Außerdem erlernen und verbessern menschliche Regler ihren Regelalgorithmus mit der Zeit: Das Verhalten des Medizinphysikers ändert sich mit jedem erstellten Behandlungsplan und er fängt an, Teile des Prozesses abzukürzen.

Montes [Mon16] definiert den Unterschied zwischen menschlichen und technischen Reglern wie folgt: Technische Regler reagieren schnell, akkurat und stabil (und übertreffen Menschen meist bei einer spezifischen Aufgabe). Eine Voraussetzung hierfür ist jedoch, dass alle Annahmen über ihre Umwelt erfüllt sind. Ändern sich diese Annahmen und der Regler besitzt keine passenden Steuermöglichkeiten oder Einstellungen, die Änderung zu kompensieren, wird er instabil. In einer sich häufig ändernden Umwelt sind Regler nötig, die ein gewisses Maß an Kreativität besitzen. Menschen können ihre Regelalgorithmen dynamisch anpassen, wodurch sie in der Lage sind, den Mangel an einer Maschine zu erkennen, die Aufgabe für sie zu übernehmen und durch kreative Maßnahmen Stabilität zu gewährleisten. Dieses Konzept ist jedoch oft schwierig, in den Systementwurf einzuarbeiten. Anders als bei technischen Reglern, kann bei menschlichen Reglern eine Informationsflut (etwa durch hunderte von Displays) von Nachteil sein, wenn der Regler die Übersicht verliert oder den Operationsmodus verwechselt. Auf der anderen Seite besagt das Yerkes-Dodson-Gesetz, dass auch ein Reizmangel zu einer verminderten Leistungsfähigkeit führt, dass also bei monotonen Aufgaben die Fehleranfälligkeit steigt.

Montes [Mon16] beschreibt einige Nachteile von STPA bei der Analyse menschlicher Regler. Dazu gehört, dass sich das Prozessmodell nicht für gewisse Arten von Systemmodellen eigne, die menschliche Regler benötigen. Außerdem fehlen einige fundamentale Überlegungen in der Analyse (z. B. das Arbeitsumfeld oder individuelle Verhaltensmuster). Weiterhin merkt Montes [Mon16] an, dass menschliche Regler nicht nur ihren Regelalgorithmus dynamisch anpassen, sondern darüber hinaus auch die Quellen, aus denen sie Feedback erhalten. Gleichzeitig passen sie die Struktur ihres Prozessmodells an, um neue Arten von Informationen verarbeiten zu können. Der aktuelle STPA-Prozess sieht bei menschlichen Reglern vor, nach Unstimmigkeiten in Feedback, Prozessmodell und Regelalgorithmus zu suchen. Obwohl dieses Vorgehen effektiver als traditionelle Verfahren ist, vereinfacht es die Rolle des Menschen in komplexen System immer noch zu stark, da es sie wie einen technischen Regler betrachtet. Das Prozessmodell eines menschlichen Reglers enthält mehr Informationen über das System in seiner Gesamtheit als das eines technischen Reglers, und es entwickelt sich weiter. Genauso wichtig bei der Analyse ist die Miteinbeziehung der Leistungsbeschränkungen und der Individualität von Menschen. Zu guter Letzt bemerkt Montes [Mon16], dass das Prozessmodell menschlicher Regler im Gegensatz zu Maschinen zu Beginn des Betriebs nicht leer ist, sondern bereits Informationen und Erfahrungen aus der Vergangenheit enthält.

Es existiert eine große Zahl von Techniken, die im Rahmen einer Riskikoanalyse den Menschen deutlich komplexer modellieren als STPA. Beispiele für solche Techniken sind Systematic Human Error Reduction and Prediction Approach (SHERPA) [Emb86], Technique for Human Error Rate Prediction (THERP) [ADS83] und Technique for the Retrospective and Predictive Analysis of Cognitive Errors (TRACEr) [SK12]. Anders als bei STPA ist jedoch die gleichzeitige Betrachtung der technischen Regler nicht möglich. Außerdem betont Thornberry [Tho14], dass zu detaillierte Modelle schwer zu analysieren sind. Der unkomplizierte Ansatz von STPA sei besser. Stattdessen schlägt er eine Erweiterung für STPA vor, mit der einige der Nachteile behoben werden. Die erste Änderung betrifft das Prozessmodell. Hier haben menschliche Regler stattdessen zwei Modelle: Ein Modell des geregelten Prozesses und ein Modell, das die Automatisierung betrifft. Eine weitere

Änderung passt das Feedback an. Zusätzlich zu dem im Systemdesign vorgesehenen Feedback von Sensoren und Displays haben menschliche Regler eine sensorische Wahrnehmung, mit der sie zusätzliche Informationen aufnehmen. Dazu gehört z. B. das Geräusch des Motors bei der Steuerung von Fahrzeugen mit Verbrennungsmotor. Vorsicht ist geboten, wenn durch technische Neuheiten (wie elektrische Motoren) sensorisches Feedback verschwindet und Sicherheitsrisiken entstehen. Das erweiterte Modell besitzt außerdem Möglichkeiten zur Darstellung von nicht wahrgenommenem Feedback oder falsch interpretiertem Feedback. Die nächste Änderung betrifft die Eingabe durch hierarchisch übergeordnete Strukturen. Hier spezifiziert Thornberry [Tho14] niedergeschriebene und antrainierte Prozeduren, externe Umwelteinflüsse, die über sensorische Wahrnehmung hinaus gehen, die Arbeitskultur, den sozialen Kontext, und physiologische Faktoren wie Stress, Müdigkeit und Auslastung. All diese Dinge nehmen Einfluss darauf, wie Menschen Feedback in ihre Prozessmodelle aufnehmen und wie sie Entscheidungen treffen. Eine letzte Änderung fügt dem Reglermodell Affordanz hinzu. Diese beschreibt u. a. Situationen, in denen Menschen den erforderten Knopf oder Schalter nicht finden, oder aus Versehen einen falschen Knopf drücken.

Montes [Mon16] baut auf dieser Erweiterung auf und fügt weitere Elemente hinzu. Das Ergebnis heißt System-Theoretic Process Analyis Refined Controller-analysis (STPA-RC) und fügt dem System zunächst Phasen hinzu, die die Zeitspanne, in der das System existiert, sinnvoll unterteilt. Mögliche Phasen sind z. B. Entwicklung, Bau, Normalbetrieb, Reparatur etc. Phasen können wiederum in Unterphasen gegliedert werden (Parken, Halten, Fahren…). In jeder Phase werden nicht nur gegenwärtige Einflüsse betrachtet, sondern auch mögliche Einflüsse von vergangenen Phasen. Ein weiteres Element, das erweitert wurde, ist die Affordanz. Hier schlägt Montes [Mon16] vor, spezielles Affordanz-Feedback zu verwenden, um dem menschlichen Regler ein ordnungsgemäßes Absenden seines Regelsignals zu signalisieren. Im Gegensatz zu normalem Feedback ist dieses unabhängig von dem Prozess. So könnte etwa ein Knopf bei Aktivierung vibrieren oder aufleuchten. Montes [Mon16] erweitert auch das Prozessmodell um zusätzliche Strukturen. In STPA-RC verfügen menschliche Regler über Informationen zu der gesamten Kontrollhierarchie. Dazu gehört, welche Komponenten Priorität haben, und in welcher Phase bzw. Unterphase sich das System befindet. Außerdem besitzen menschliche Regler Motive, die ihre Regelalgorithmen beeinflussen. Ein Beispiel hierfür ist ein Pilot, der schnell nach Hause möchte und sich bei der Landung des Flugzeugs beeilt. Die von Thornberry [Tho14] eingeführten externen Umwelteinflüsse werden in STPA-RC weiter verfeinert. Hier gibt es den Arbeitsplatz (Lichtverhältnisse, Lärm, Temperatur, physische Ergonomie und Arbeitsauslastung), die menschliche Individualität (physische Merkmale, Risikotoleranz, Vertrauen, körperliches und seelisches Wohlbefinden, Alter und Müdigkeit), und andere Einflüsse (Erfahrung, Training, Kultur). Weitere Leitwörter und -sätze für die Analyse von Risiken bei der Einwirkung von organisatorischem Einfluss auf Menschen finden sich in Stringfellow [Str10]. Dort werden als potentielle Unfallursachen z. B. ”Ziele falsch priorisiert” oder ”Missverstehen von Prozessgrenzen” vorgeschlagen. Abb. 3.12 verdeutlicht das aktualisierte Reglermodell in STPA-RC. Komponenten, die menschlichen Reglern hinzugefügt werden müssen, sind grün markiert.

Auch France [Fra17] schlägt eine Erweiterung von STPA, die jedoch etwas andere Schwerpunkte setzt. Bei der Risikoanalyse sind auf drei Aspekte zu achten: der Regelsignalentscheidungsprozess (wie hat der Mensch die Entscheidung getroffen?), das Prozessmodell (welche Dinge weiß er über den Prozess, das Prozessverhalten und die Umwelt?), und Prozessmodellupdates (wie ist er zu dem Wissen gekommen?). France [Fra17] greift die Idee auf, dass Menschen unterschiedlich priorisierte Ziele haben. Beispielsweise wäre das Hauptziel eines Autofahrers, am Ziel anzukommen, während ein Nebenziel Sicherheit ist. Es wird angemerkt, dass die Ursache für fehlende Updates oftmals ein Stimulus ist, dem es an Salienz fehlt, etwa ein zu leises akustisches Signal. Weitere Ursachen

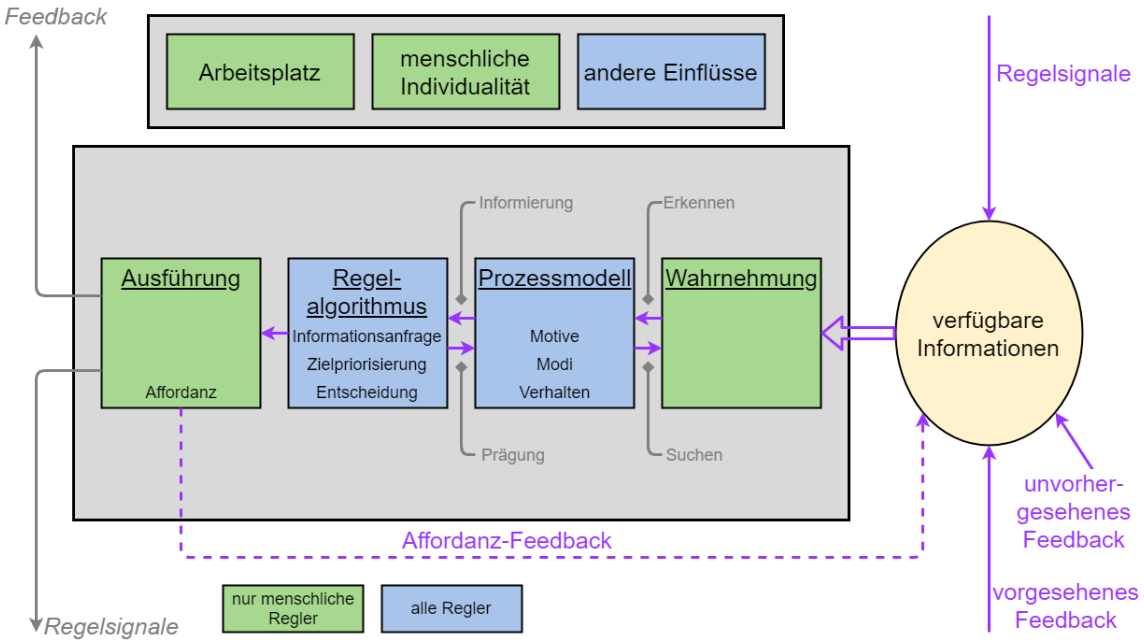


Abbildung 3.12: STPA-RC

sind zu hoher Aufwand, sich das Update zu beschaffen (z. B. Anweisungen in einem dicken Handbuch), Überraschung (seltene Fehlfunktionen), und Zweifel an der Richtigkeit (Warnungen, die oft fälschlicherweise auslösen). In solchen Situationen müssen Menschen provisorische Einschätzungen machen, basierend auf unsicheren, unvollständigen und zum Teil widersprüchlichen Informationen [Dek06]. Sie sind gleichermaßen konfrontiert mit *bekannten Unbekannten* – Informationen, dessen Fehlen ihnen bewusst ist – und *unbekannten Unbekannten* – Informationen, von denen sie nicht wissen, dass sie sie benötigen [Fra17].

Zusammenfassend lässt sich sagen, dass es eine Vielzahl von Ideen gibt, die Komplexität des menschlichen Verstands in einem Modell akkurat abzubilden. Weiterhin gibt es mehrere Ansätze diese Modelle in STPA zu integrieren. In diesem Abschnitt wurden nur einige wenige behandelt. Für eine umfassendere Auseinandersetzung mit diesem Thema sei der Leser auf entsprechende Literatur verwiesen. Bei der Ausarbeitung eines Konzepts für die Modellierung von Regelalgorithmen, das sich sowohl auf technische als auch auf menschliche Regler anwenden lässt, sollen auch die in diesem Abschnitt gesammelten Überlegungen berücksichtigt werden.

3.4 Zusammenfassung

In diesem Kapitel wurde der aktuelle Stand der Forschung in Bezug auf die Modellierung von Regelalgorithmen vorgestellt. Zunächst lag der Fokus auf Programmen, die Ingenieure bei der Analyse mit STPA unterstützen. Insgesamt wurden sechs Programme betrachtet. SAHRA, RM Studio und SafetyHAT besitzen keine (veröffentlichte) Funktion für die Modellierung von Regelalgorithmen. SpecTRM erlaubt die Deklaration von Vorbedingungen und Inhalte von Regelsignalen, integriert kontinuierliche Variablen und modelliert Modiwechsel mithilfe endlicher Automaten. STAMP Workbench sieht für die Vorbedingungen eine Spalte in der Tabelle für unsichere Regelsignale

vor, während An STPA Tool eine Entscheidungstabelle aus den unsicheren Signalen automatisch generiert. Zusätzlich zu den Programmen wurden manuell durchgeführte STPA-Analysen betrachtet und Möglichkeiten zur Modellierung von Regelalgorithmen extrahiert. Zu den fünf gefundenen Methoden gehören Funktionsgraphen, Entscheidungstabellen, boolesche Schaltkreise, Formeln und Fließtexte. Zuletzt wurden menschliche und technische Regler miteinander verglichen.

Basierend auf diesen Befunden ist das Ziel im nächsten Kapitel, ein Konzept zu entwickeln, welches dynamisch genug ist, die Algorithmen einer Vielfalt von Reglern zu modellieren.

4 Modellierungskonzept für Regelalgorithmen

Gegenstand dieses Kapitels ist die Entwicklung eines Konzepts zur Modellierung der Regelalgorithmen in STPA. Dieses muss so konzipiert sein, dass es auch in der Praxis einsatzfähig ist. Um dies zu garantieren, wurden in Kapitel 3 bereits existierende Konzepte betrachtet. Bei dem Entwurf einer eigenen Idee ist eine Anlehnung an diese empfehlenswert. Zur Beurteilung der Vor- und Nachteile der gesammelten Konzepte sind sinnvolle Kriterien nötig, welche zunächst erarbeitet werden müssen. Anschließend kann ein objektiver Vergleich stattfinden. Bei der Ausarbeitung eines Konzeptes ist das Ziel, möglichst viele der Kriterien zu erfüllen, um so zu einem qualitativ hochwertigen Ergebnis zu gelangen. Das Konzept wird zunächst formal definiert und später anhand eines kurzen Beispiels demonstriert. Dies dient einem tieferen Verständnis und beugt Missverständnisse vor. Des Weiteren erlaubt die Anwendung des Konzeptes an einem fiktiven Szenario, Vorteile zu erproben, sowie Defizite aufzudecken. Eine Bewertung anhand des Beispiels wird nachfolgend dargelegt.

Ein weiteres Ziel dieses Kapitels ist, Parallelen zwischen Regelalgorithmen und Aktoren zu erforschen. Beide Komponenten sind im Kern Abbildungen. Bei Regelalgorithmen wird das aktuelle Prozessmodell auf ein passendes Regelsignal abgebildet, während Aktoren die eingehenden Regelsignale wieder zu einer Änderung der Prozessvariablen konvertieren. Es soll erörtert werden, inwiefern diese Parallelen genutzt werden können, das hier entwickelte Konzept auf Aktoren zu übertragen. Dieses Thema wurde in der Literatur bisher nicht behandelt.

4.1 Bewertung der aktuellen Modellierungskonzepte

Bei der Aufstellung geeigneter Kriterien ist es wichtig, alle Aspekte, die zu der Güte eines Modells beitragen, vollständig abzudecken. Zu diesem Zwecke ist es ratsam, bereits existierende Kriterienlisten zu Rate zu ziehen. Passende Qualitätseigenschaften für Modelle im Allgemeinen zu definieren, ist schwierig, da für unterschiedliche Themenbereiche sehr unterschiedliche Arten von Modellen existieren. Die Suche muss weiter eingeschränkt werden, indem auch der Themenbereich des Modells berücksichtigt wird. Das hier zu entwerfende Konzept soll eine angemessene Darstellung von Regelalgorithmen ermöglichen. Für Algorithmen bzw. generell Software existieren zahlreiche Listen und Standards, die deren Qualität an fest definierten Eigenschaften messen. Ein häufig verwendeter und von vielen als tauglich eingeschätzter Standard ist *ISO/IEC 25010:2011*. Die dort aufgeführten Qualitätseigenschaften eignen sich nicht direkt für die Konzipierung eines Modells. Sie bieten aber einen guten Ansatzpunkt bei der Zusammenstellung einer vollständigen Liste von Kriterien. Im hiernach folgenden Vergleich sollen daher Kriterien verwendet werden, die aus den Qualitätseigenschaften des ISO-Standards inspiriert und abgeleitet sind. Die resultierenden Kriterien für ein hochwertiges Modellierungskonzept für Regelalgorithmen sind *Verständlichkeit*, *Flexibilität*, *Skalierbarkeit* und *Erweiterbarkeit*. Bei der Gegenüberstellung existierender Konzepte und der Ausarbeitung eines eigenen Konzepts wird deren Güte an genau diesen vier Kriterien beurteilt. Eine Erläuterung der Kriterien findet sich im folgenden Abschnitt.

4.1.1 Kriterien

Verständlichkeit bezeichnet die Eigenschaft eines Modells, von einer großen Zahl an Menschen in relativ kurzer Zeit verstanden zu werden. Die korrekte Konstruktion eines verständlichen Modells ist schnell erlernbar. Verständlichkeit kann auf vielerlei Weise erreicht werden, z. B. durch eine klare, übersichtliche Struktur und der Benutzung von ähnlich Farben und Formen für semantisch zusammengehörende Komponenten. Nicht verständliche Modelle bergen die Gefahr, von unterschiedlichen Betrachtern unterschiedlich aufgefasst zu werden. Des Weiteren kann es potentielle Betrachter davon abschrecken, sich mit dem Modell zu beschäftigen. Der mit unverständlichen Modellen verbundene Zeitaufwand beschränkt sich nicht nur auf das Erlernen, sondern schließt auch die Konstruktion und die Erfassung existierender Modelle ein. Ein verständlichen Modellierungsverfahren kann außerdem mit wenig Aufwand in XSTAMPP oder anderen Programmen implementiert und gewartet werden, wodurch eine weitere Verbreitung möglich ist.

Flexibilität ist eine Qualität, die die Anwendung des Konzepts in einem breiteren Spektrum an Szenarien erlaubt. Eine flexible Modellierungsstrategie ist dynamisch genug, die Regelalgorithmen technischer, menschlicher und soziologischer Regler adäquat darzustellen. Es existieren Beschränkungen weder in Bezug auf die Form der verwendbaren Prozessvariablen, noch in Bezug auf die Form der Regelsignalinhalte. Die Modellierung der Algorithmen ist möglich, unabhängig davon, ob das Ziel des Regelsignals ein Aktor oder ein hierarchisch niederer Regler ist. Flexibilität kann entweder dadurch erreicht werden, dass ein einziges Konzept in allen denkbaren Szenarien angewendet werden kann, oder dass mehrere Konzepte miteinander verbunden und dynamisch ausgetauscht werden können.

Skalierbarkeit bezeichnet die Eigenschaft, bei steigendem Umfang des dargestellten Regelalgorithmus mit einer akzeptablen Geschwindigkeit an Komplexität zuzunehmen. In anderen Worten, ein Modellierungskonzept skaliert gut, wenn damit auch ausführliche Algorithmen mit dementsprechendem Aufwand und in dementsprechendem Platz dargestellt werden können. Hierzu gehört insbesondere die Möglichkeit, komplexe Regelalgorithmen in einer angemessenen Zeitspanne zu modellieren und dessen Modell verstehen und damit arbeiten zu können. Für eine Implementierung bedeutet das, dass das Programm auch bei umfangreichen Algorithmen einen annehmbaren Speicherverbrauch hat und flüssig kontrollierbar bleibt, bzw. dass durch die Allokation der Modellgröße entsprechenden Ressourcen Performanzprobleme behoben werden können. Es ist zu bedenken, dass die Größe des Regelalgorithmus nicht nur von der Zahl der Regeln, sondern auch von der Zahl der durchschnittlichen verwendeten Prozessvariablen abhängt. Ein gut skalierbares Modell steigt in seinem Umfang proportional zu diesen Werten an.

Erweiterbarkeit bedeutet, dass das Konzept in einer Weise strukturiert ist, die bei Bedarf die Ergänzung zusätzlicher Inhalte erlaubt. Ein besonders hoher Stellenwert hat hier Automatisierung. Modelle sollen maschinenlesbar sein, um so maschinengestützte Optimierungen sowie Konvertierungen in andere Modelltypen vornehmen zu können. Erweiterbarkeit war eines der Urkonzepte von XSTAMPP, und soll auch mit der Implementierung einer Modellierungsfunktion für Regelalgorithmen beibehalten werden.

Ein perfektes Konzept würde jedes der vier oben vorgestellten Kriterien zur Gänze erfüllen. Es muss allerdings die Möglichkeit in Betracht gezogen werden werden, dass einige der Anforderungen nicht miteinander vereinbar sind. In diesem Fall existiert das perfekte Konzept nicht und es müssen die Kriterien priorisiert und Kompromisse gemacht werden, um ein Konzept zu finden, welches zumindest wichtige Anforderungen erfüllt. Zu guter Letzt muss angemerkt werden, dass

die im folgenden durchgeführte Bewertung keineswegs ein auf Tests basierendes und anhand von Messwerten evaluiertes Ergebnis repräsentiert, sondern als reines Theoretisieren über den Entwurf eines Konzepts interpretiert werden muss, das das Potenzial hat, in realen Anwendungen brauchbar zu sein.

4.1.2 Bewertung

Zunächst wird der Modellierungsaufwand betrachtet. Dieser ist für das Kriterium *Verständnis* relevant. Formeln tun sich hier hervor, da sie zur Erstellung einer Regel nur die Eingabe einiger Zeichen erfordern. Auch Entscheidungstabellen lassen sich in sehr wenig Zeit konstruieren, da die Prozessvariablen inklusive ihrer Zustände vorgeneriert werden können, und nur noch die Eintragung der Bedingungen erforderlich ist. Der Vorteil von booleschen Schaltkreisen besteht darin, dass Zwischenergebnisse, die aus einer Regel gewonnen wurden, in anderen Regeln wiederverwendet werden können. Dies wird jedoch dadurch ausgeglichen, dass zusätzliche Ressourcen in die Gestaltung eines übersichtlichen Schaltplans fließen müssen. Einzig Funktionsgraphen eignen sich nicht dazu, von Hand modelliert zu werden. Hier wäre es erforderlich, die Regeln mithilfe von Formeln zu definieren und die Graphen aus diesen zu generieren.

Ein weiterer für das *Verständnis* wichtiger Punkt ist die Übersichtlichkeit des Modells. Die Vorbedingungen und der Inhalt der Signale müssen für den Sicherheitsingenieur auf einen Blick erkennbar sein. Auch hierfür eignen sich Formeln durch ihre Kompaktheit gut. Selbst lange Formeln bleiben lesbar, sofern an angebrachten Stellen Zeilenumbrüche eingefügt werden. Funktionsgraphen sind hier jedoch deutlich überlegen, da hier der Zusammenhang zwischen Prozessvariablen oder Regelsignalinhaltsvariablen sofort sichtbar ist. Boolesche Schaltkreise sind visuell sperrig, aber sie können in einem Diagramm mehrere Regeln darstellen. Bei komplexen Regeln nimmt die Übersicht zusehends ab. Ähnliche Nachteile zeigen Entscheidungstabellen, die oft trotz automatischer Minimierung verhältnismäßig groß bleiben. Durch ihre Erzwingung von DNF können sie in seltenen Fällen exponentiell viele Spalten erfordern. Mit zunehmender Größe steigt ferner die Gefahr, in der Spalte oder Zeile zu verrutschen.

Verständnis beinhaltet auch die Zugänglichkeit und den Schwierigkeitsgrad des initialen Lernprozesses. In dieser Hinsicht ist die intuitive Natur der Graphen und Tabellen ein Vorteil. Diese sind auch in den Medien oft genutzte Darstellungsformen, um Informationen zu übermitteln, und somit den meisten Menschen vertraut. Demgegenüber ist die Bereitschaft, sich mit Formeln und insbesondere Schaltkreisen zu beschäftigen, bei damit unvertrauten Menschen im Allgemeinen geringer.

Für eine hohe *Flexibilität* ist notwendig, mit allen verwendbaren Variablentypen umgeben zu können. Insbesondere stehen hier kontinuierliche Variablen im Mittelpunkt. Funktionsgraphen eignen sich gut für diese, da sie an jeder Achse ein kontinuierliches Werteintervall anzeigen können. Ein ebenso brauchbares Ergebnis liefern Formeln. Hier können kontinuierliche Werte durch beliebiges Verknüpfen arithmetischer Operatoren berechnet und verglichen werden. In Entscheidungstabellen können die für das Auslösen des Signals geforderten Werte zwar in die entsprechenden Zellen eingetragen werden (z. B. "> 100.0" oder "$= carAheadSpeed \pm 0.7$"), aufgrund des DNF-Formats können jedoch mehrere Anforderungen an dieselbe Variable nicht aufgeteilt werden, wodurch unmäßig große Zellen entstehen können. Des Weiteren ist uneindeutig, welche Zeile benutzt wird, wenn Beziehungen zwischen mehreren Variablen auszudrücken sind. In Formeln ist die Nutzung von diskreten und kontinuierlichen Prozessvariablen gleichermaßen möglich. Auch die parallele

Nutzung in Formeln wie $(x = 5) \wedge y$ ist denkbar. Während boolesche Schaltkreise auf diskrete Werte beschränkt sind, können fuzzy-logische Schaltkreise auch mit kontinuierliche Variablen rechnen. Diese sind jedoch üblicherweise auf das Intervall $[0, 1]$ beschränkt, was für reale Szenarien unpraktisch ist.

Genauso wichtig für die *Flexibilität* einer Modellierungsstrategie ist die Möglichkeit, Regelsignalinhalte zu definieren. Dies ist mit Entscheidungstabellen, die nur darauf ausgelegt sind, eine Ja-Nein-Entscheidung zu treffen, nicht möglich. In Funktionsgraphen kann eine der Achsen dazu verwendet werden, den Wert der ausgehenden Variable anzuzeigen. Dadurch bleibt jedoch in zweidimensionalen Graphen nur noch eine Dimension für eingehende Werte übrig. Bei Schaltkreisen zeigen sich ähnliche Schwächen wie bei den Entscheidungstabellen: Boolesche Schaltkreise haben eine binäre Ausgabe und fuzzy-logische Schaltkreise sind auf ein kleines Intervall beschränkt. Einzig Formeln erlauben es, sowohl Regelsignale mit diskreten Inhalten (durch boolesche Operatoren), als auch mit kontinuierlichen Inhalten (durch arithmetische Operatoren) zu erzeugen, ohne dem Sicherheitsingenieur Einschränkungen aufzuerlegen.

Das Modellierungskonzept muss auch bei einer großen Zahl Regeln anwendbar sein, um dem Kriterium *Skalierbarkeit* genüge zu tun. Sowohl Entscheidungstabellen, als auch Funktionsgraphen, können maximal eine Regel pro Diagramm darstellen. Die Zahl der Diagramme ist somit gleich der Zahl der Regeln. Entscheidungstabellen benötigen zur Speicherung nur eine Liste der relevanten Variablen, deren Werte und deren Beziehung zu anderen Variablen. Funktionsgraphen benötigen weitaus mehr Speicher, es sei denn es werden nur aktuell gebrauchte Graphen gespeichert und alle anderen bei Bedarf neu generiert. Formeln sind kompakter als Entscheidungstabellen und Funktionsgraphen, weswegen mehre davon zeitgleich betrachtet werden können. Vom Speicherbedarf her sind sie gleichauf mit Entscheidungstabellen. Schaltkreise können als einziges Modellierungskonzept mehrere Regeln in einem Diagramm zeigen, wodurch ihre Zahl auch bei wachsendem Regelsatz konstant bleibt. Leider steigt stattdessen die Zahl der Gatter, was zu einem Verlust der Übersicht führt. Die Regeln zu gruppieren und auf mehrere Schaltkreise aufzuteilen vermag das Problem zu lösen. Der Speicherverbrauch ist geringfügig höher als bei Formeln und Entscheidungstabellen, da zusätzlich die Koordinaten der Gatter und Verbindungswegpunkte gespeichert werden müssen.

Skalierbarkeit erfordert auch, dass große Regeln mit vielen Variablen modelliert werden können. Hier zeigt sich der Hauptnachteil von Funktionsgraphen. Bei zwei Dimensionen kann ein binäres Regelsignal von nicht mehr als zwei Prozessvariablen abhängen. während es bei einem kontinuierlichen Signal nur eine Prozessvariable ist. Die anderen drei Methoden haben keine Einschränkung in Bezug auf die Zahl der Variablen, sie verlieren aber bei großen Regeln an Lesbarkeit. Insbesondere die Risikoanalyse mit Schaltkreisen wird bei zunehmender Komplexität aufwändig, bedingt durch die steigende Schwierigkeit bei der Zuordnung der Gatterverbindungen.

Ein wichtiger Teil von *Erweiterbarkeit* ist die Maschinenlesbarkeit eines Modells. Formeln, Entscheidungstabellen und Schaltkreise lassen sich einwandfrei einlesen und verarbeiten, da sie alle im Kern Verknüpfungen von Bauteilen mit eindeutigen Definition sind. Bei Funktionsgraphen ist die Maschinenlesbarkeit von der Speichermethode abhängig. Ist die dem Graphen zugrunde liegende Formel hinterlegt, gleicht das Verfahren der Formel. Sind jedoch nur die Koordinaten der Punkte gespeichert, kann die ursprüngliche Funktion nicht rekonstruiert werden.

Ein weiterer Teil der *Erweiterbarkeit* ist die Möglichkeit, die Modellierungsmethode konzeptuell erweitern zu können. Funktionsgraphen besitzen in der Regel zwei Dimensionen. In speziellen Szenarien kann es jedoch dienlich sein, dreidimensionale Graphen zu verwenden. Auch das Überlagern

mehrerer Kurven in demselben Koordinatensystem ist möglich. Für Schaltkreisen können designtechnisch viele Hilfestellungen ergänzt werden. Denkbar wären die Beifügung zusätzlicher Kommentare, die Verwendung unterschiedlicher Farben für Signale und Gatter, sowie die Einführung von Unterschaltkreisen. In Formeln können Möglichkeiten gesucht werden, die Übersichtlichkeit der Struktur zu verbessern. Einige Ideen sind unterschiedlich große Klammern, abgekürzte Variablennamen, und Zeilenumbrüche an passenden Stellen. Bei Entscheidungstabellen gibt es z. B. die Option, mehrere übereinanderstehende Zellen mit einem logischen `ODER` zu verknüpfen, etwa durch Ersetzen der Trennlinien durch ein $\vee$-Zeichen. Dies kann in einigen Fällen die Größe der Tabelle drastisch reduzieren.

4.1.3 Automatische Generierung des Regelalgorithmus

Suo und Thomas [ST14] haben ein Konzept vorgestellt, in dem der Regelalgorithmus in Form einer Entscheidungstabelle automatisch aus den zuvor vom Sicherheitsingenieur in einer Kontexttabelle als unsicher markierten Regelsignalen generiert wird. Hierzu einige Kommentare:

(1) Der Hauptvorteil der Methode besteht darin, dass alle denkbaren Szenarien aufgelistet und einzeln abgearbeitet werden. Dadurch wird verhindert, dass Szenarien übersehen werden. Die Auflistung ist jedoch nur dann möglich, wenn alle Prozessvariablen diskretisiert werden. Eine kontinuierliche Variable hat unendlich viele Zustände, weswegen ein Eintragen in die Tabelle nicht funktioniert. Alternativ können Variablen temporär diskretisiert und bearbeitet werden. Nach der Generierung des Algorithmus werden die Variablen wieder in ihre kontinuierliche Form gebracht, und die entsprechenden Regeln angepasst. Während diesem Vorgang wird der Algorithmus kompakter, da mehrere Zustandskombinationen in einer Regel zusammengefasst werden können.

(2) Auch die Regelsignale selbst sind binär. Genau wie bei den Prozessvariablen müssten kontinuierliche Inhalte (wie z. B. eine Geschwindigkeitsvorgabe) zunächst diskretisiert, und die resultierenden Zustände als getrennte Signale bearbeitet werden (z. B. ”niedrige Geschwindigkeitsvorgabe”, ”hohe Geschwindigkeitsvorgabe”, …). Auch hier werden die Regeln anschließend vereinigt.

(3) Eine weitere Option ist, diskrete und kontinuierliche Variablen in zwei getrennten Phasen zu bearbeiten. Hierbei fällt der Vorteil der Kontexttabellen für kontinuierliche Variablen weg. Dafür nimmt auch der Overhead ab, der bei der Mehrfachbearbeitung entsteht.

(4) In welcher Form der Regelalgorithmus generiert wird, ist von der zur Identifikation der unsicheren Regelsignale benutzten Methode unabhängig. Das heißt, dass es kein Problem darstellt, statt den Entscheidungstabellen einen Liste von Formeln oder einen Schaltkreis zu generieren.

(5) Die Länge der Kontexttabelle steigt exponentiell zu der Zahl der Variablen an. Um auch die Analyse größerer Systeme möglich zu machen, müssen die Prozessmodelle der Regler möglichst klein gehalten, bzw. große Regler mit mehreren Verantwortlichkeiten in mehrere kleine Regler unterteilt werden. Außerdem müssen bestimmte Wertekombinationen von vorn herein als sicher oder unsicher markierbar sein, um die Tabelle auf eine überschaubare Größe zu reduzieren.

Unter den entsprechenden Bedingungen kann das Konzept dem Sicherheitsingenieur zahlreiche Vorteile bieten, darunter etwa die Sicherheit, alle Szenarien betrachtet zu haben. Eine solche Hilfestellung darf jedoch nicht zur Einschränkung werden. Zumindest die nachträgliche Bearbeitung des generierten Regelalgorithmus ist unerlässlich. Ebenso muss die Möglichkeit, bestimmte Prozessvariablen von der Inklusion in der Kontexttabelle auszuschließen, in Betracht gezogen werden.

4.1.4 Eignung für menschliche Regler

In Kapitel 3 wurde die Wichtigkeit einer angemessenen Behandlung menschlicher Regler besprochen. Eine solche ist für die *Flexibilität* der Modellierungsstrategie von hoher Wichtigkeit. Um STPA noch besser an die Komplexität der menschlichen Regelprozesse anzupassen, wurden bereits einige Erweiterungen vorgeschlagen [Tho14][Mon16][Fra17]. Die meisten davon erweitern den Regler um zusätzliche Komponenten, die den Regelalgorithmus selbst oft nur indirekt betreffen. Ein Vorschlag, der direkt mit den Algorithmen in Verbindung steht, ist die Integration mehrerer Prozessmodelle. Diese stellt jedoch für keines der hier betrachteten Modellierungskonzepte ein Hindernis dar, weil Variablen auch aus unterschiedlichen Prozessmodellen problemlos kombiniert werden können. Auf der anderen Seite der Regelalgorithmen wurde die Ergänzung von Affordanz vorgeschlagen. Dort wird u. a. die Analyse von Fehlgriffen durchgeführt. Dabei ist die Annahme, dass das Regelsignal bereits korrekt entschieden wurde. Wie der Regelalgorithmus modelliert ist, spielt hier keine Rolle.

Castilho et al. [Cas+18] demonstrieren die Anwendung von STPA an einem Piloten während einem Start bei Seitenwind. Sie analysieren hier auch die kontinuierlichen Signale, die der Pilot über diverse Ruder an das Flugzeug gibt. Eine Besonderheit dieser ist, dass selbst starke Ausschläge nicht zwangsweise ein unsicheres Regelsignal darstellen, solange das Ruder nur kurzzeitig in der Extremposition verbleibt. Tatsächlich sind derartige Korrekturen bei schwierigen Startmanövern normal. Regelsignale, die extreme Werte über einen längeren Zeitraum beibehalten, werden der Kategorie ”zu lange gesendet” zugeordnet. In der Kategorie ”Nichtsenden verursacht Gefährdung” stehen unsichere Signale, die das Ruder in die Ausgangsposition bringen, d. h. Signale, die einen Wert nahe 0 enthalten. Die Kategorie ”zu spät gesendet” enthält dagegen Korrektursignale, die der Pilot nicht rechtzeitig sendet. Castilho et al. [Cas+18] kommentieren, dass bei vielen der kontinuierlichen Signale nicht klar ist, in welche Kategorie sie fallen. Als Leitwörter eignen sich die Kategorienamen damit für kontinuierliche Werte schlecht. Die Ausarbeitung eines alternativen Vorgehens ist jedoch nicht Teil dieser Arbeit. Für die Bewertung der Modellierungskonzepte wird angenommen, dass alle unsicheren Regelsignale identifiziert wurden und nur noch die Suche nach möglichen Ursachen aussteht.

Ursachen, die mit der Ungenauigkeit oder Fehlerhaftigkeit von Prozessvariablen zusammenhängen, werden im Prozessmodell bearbeitet, während Ursachen, die mit dem Verlorengehen von Regelsignalen zusammenhängen, an der Grenze zwischen Regler und Aktor behandelt werden. Die Analyse des Regelalgorithmus beschränkt sich auf Szenarien, in denen trotz korrekter Variablen das falsche Regelsignal gewählt wurde. Bei menschlichen Reglern stellen Regelalgorithmen im Allgemeinen eher Vorschriften als tatsächliche Gedankengänge dar, da diese komplex und dynamisch sind. Die Fähigkeit von Menschen, unvorhergesehene Problemstellungen auf kreative Weise zu lösen, ist ein immenser Vorteil gegenüber Maschinen. Gleichzeitig macht sie die Aufnahme sämtlicher für die Entscheidung relevanter Faktoren unmöglich. Abweichende Ziele, Kulturen und Persönlichkeiten, sowie nachlassende Leistungsfähigkeit ausgelöst durch Müdigkeit, hohe Belastung, Krankheit, Alter oder Langeweile können nicht ohne Weiteres in den Regelalgorithmus integriert werden. Trotzdem

kann der Algorithmus je nach Darstellungsweise bei der Analyse behilflich sein. Beispielsweise kann insbesondere an Formeln gut erkannt werden, welche Stellen anfällig für Rechenfehler sind. Bei Graphen sind Fehlerursachen möglicherweise schwieriger zu sehen. In Bezug auf diskrete Variablen eignen sich auch Entscheidungstabellen und Schaltkreise für die Kausalanalyse.

4.1.5 Fazit

Der Vergleich zeigt, dass keine Methode perfekt ist, sondern jede Vor- und Nachteile besitzt. Der Hauptvorteil von Schaltkreisen besteht darin, mehrere Regeln parallel darstellen zu können. Jedoch verliert das Modell mit jeder weiteren Regel an Übersichtlichkeit, wodurch sich dieser Vorteil nicht vollständig auszahlt. Darüber hinaus ist für die Erstellung von Schaltungen zusätzliche Zeit erforderlich. Funktionsgraphen zeichnen sich vor allem durch ihre Anschaulichkeit aus. Allerdings können sie schon ab drei Variablen nicht mehr verwendet werden. Die Erzeugung der Graphen selbst wird normalerweise mithilfe von Formeln vorgenommen, da ein manuelles Zeichnen ungenau ist. Keine Beschränkung bei der Anzahl der Variablen und wenig Aufwand bei der Erstellung bieten Entscheidungstabellen. Diesen haben aber den Nachteil, bei einer großen Zahl möglicher Variablenzustände übermäßig lang, und bei ungünstigen logischen Verknüpfungen exponentiell breit zu werden. Im Gegensatz zu Schaltkreisen können Entscheidungstabellen auch mit kontinuierlichen Prozessvariablen umgehen. Die Inhalte der Regelsignale sind jedoch binär. Formeln sind kompakter als Entscheidungstabellen und nehmen kontinuierliche Variablen sowohl als Eingabe, als auch als Ausgabe an. Dies ist vor allem bei der Modellierung menschlicher Regler wichtig, kann aber auch bei technischen Reglern gebraucht werden.

4.2 Vorschlag für ein Modellierungskonzept

Auf Basis des vorangegangenen Vergleichs wird vorgeschlagen, Regelalgorithmen mit einem Konzept zu modellieren, das sich an Formeln orientiert. Dies steht im Gegensatz zu den in An STPA Tool verwendeten Entscheidungstabellen. Thomas [Tho13] vertritt die Ansicht, dass die Betrachtung kontinuierlicher Variablen als diskrete Einheiten für eine Riskikoanalyse mit STPA ausreichend ist. In der Tat ist in vielen Fällen der genaue Wert einer Prozessvariablen weniger wichtig als sein ungefährer Wertebereich. Dennoch hat die Literaturrecherche gezeigt, dass in bestimmten Fällen auch die Betrachtung kontinuierlicher Variablen notwendig ist. Das Konzept soll auch in solchen Fällen anwendbar sein, wodurch sich die Wahl auf Formeln und Funktionsgraphen einschränkt. Auch eine Limitierung der darstellbaren Variablenzahl ist jedoch zu vermeiden, weswegen die Entscheidung trotz einer etwas geringeren Zugänglichkeit auf Formeln fällt. Die Übersichtlichkeit von Graphen kann dennoch genutzt werden, indem zu allen geeigneten Formeln eine Kurve generiert wird, die die Beziehung zweier kontinuierlicher Variablen demonstriert.

Die Darstellung des Regelalgorithmus als Formelmenge verhindert selbstverständlich nicht die Anwendung von Kontexttabellen zur Bestimmung der unsicheren Regelsignale. Ebenso ist die automatische Generierung der Regeln uneingeschränkt möglich. Anders als in An STPA Tool müssen diese aber anschließend editierbar sein, damit die in den Kontexttabellen fehlenden kontinuierlichen Regelsignale nachgetragen werden können. Dabei ist darauf zu achten, dass auch manuell erzeugte Formeln maschinell lesbar sind. Für diesen Zweck muss deren Konstruktion feste Regeln auferlegt werden. Diese werden im folgenden Abschnitt erläutert.

4.2.1 Formale Definition

Ein Regelalgorithmus besteht aus einer Menge von *Regeln*. Eine Regel ist ein Wort der Sprache, die von der folgenden Grammatik beschrieben wird:

```
regel := ausgabeVariable '=' (term | formel)

term := kontVar | '-' term | konstante | funktion | term arithOp term
        | '(' term ')'

funktion := funktionsname '(' argumente ')'

argumente := term | term ',' argumente

arithOp := '+' | '-' | '*' | '/' | '^'

formel := boolVar | '¬' formel | '[' gleichung ']' | formel boolOp formel
          | '(' formel ')'

gleichung := term relation term | term relation gleichung

relation := '=' | '<' | '>' | '≤' | '≥' | '≠'

boolOp := '∧' | '∨' | '⊕'
```

Die Nichtterminalsymbole `ausgabeVariable`, `kontVar`, `konstante`, `funktionsname` und `boolVar` sind dabei beliebige Kombinationen aus Buchstaben, Ziffern und Unterstrichen, wobei mindestens ein Buchstabe vorhanden sein muss. Nur `konstante` darf eine reine Zahl sein, die bei Bedarf auch einen Punkt enthalten kann. Es folgen einige Beispiele für Regeln:

(1) $feueralarm = (feuerSichtbar \vee rauchSichtbar) \wedge \neg feueralarmAusgelöst$

(2) $sicherheitsAbstand = max(15, geschwindigkeit/2)$

(3) $luftTemperatur = 1.5 * wunschTemperatur - 0.5 * aktuelleTemperator$

(4) $bremsen = [geschwindigkeit > limit] \vee hindernisVoraus$

(5) $geschwHalten = [sicherheitsAbstand < distanz < 1.25 * sicherheitsAbstand]$
$\wedge [0.9 * geschwVordermann \leq geschw \leq 1.1 * geschwVordermann]$

Die Semantik der *Regeln* entspricht den üblichen Gesetzen der Mathematik und Logik. Kontinuierliche Variablen (`kontVar`) und Konstanten werden zunächst über arithmetische Operatoren miteinander verrechnet. Die Reihenfolge der Auswertung ist bekanntermaßen ( ) ⇒ ^ ⇒ *, / ⇒ +, -. Bei Gleichwertigkeit haben linke Operatoren Vorrang vor rechten Operatoren, wobei im Sinne der Übersichtlichkeit trotzdem Klammern empfohlen werden. Ebenso werden Funktionen ausgewertet, deren Parameter mit Kommas getrennt sind. Wird das Ergebnis direkt an die ausgegebene Variable weitergegeben, so ist sie kontinuierlich (s. Beispiele 2 und 3). Andernfalls werden mindestens zwei Ergebnisse über Relationen verglichen und mit eckigen Klammern umschlossen, um eine booleschen Variable (boolVar) zu repräsentieren (s. Beispiele 4 und 5). Mehrere boolesche Variablen werden mit booleschen Operatoren (boolOp) verglichen und das Ergebnis an die ausgegeben Variable gegeben, die in diesem Fall binär ist (`WAHR` oder `FALSCH`).

Die obigen Beispielregeln sind bei weitem nicht so komplex, wie sie in realen Fällen sein können. Trotzdem erstrecken sie sich zum Teil über mehrere Zeilen. Der Grund dafür ist die Länge der Variablennamen. Durch Ersetzen aller Namen durch passende Abkürzungen kann die Größe der Regeln drastisch reduziert werden. Da jedoch jede Variable eine Prozessvariable oder ein Regelsignal referenziert, sollten die Abkürzung so gewählt sein, dass die jeweilige Zugehörigkeit sofort ersichtlich ist. Beispielsweise ist das Abkürzen von `geschwindigkeit` und `geschwindigkeitVordermann` mit `g1` und `g2` nicht zu empfehlen. Eine bessere Alternative wäre `gschw` und `gschwVor`.

Einer der Vorteile technischer Regler gegenüber Menschen ist ihre Fähigkeit, auch schwierige Berechnungen in Sekundenbruchteilen zu vollbringen. Auch solche Berechnungen lassen sich mit *Regeln* modellieren, doch deren Analyse kann für Menschen schwierig sein. Für den Fall, dass die Regel genau zwei kontinuierliche Variablen enthält, kann dem Sicherheitsingenieur geholfen werden, indem deren Beziehung in einem Funktionsgraphen verdeutlicht wird. Ist das Ausgabesignal kontinuierlich und befindet sich genau eine Variable auf der rechten Seite der Gleichung, so ist die Darstellung trivial: Das Ausgabesignal wird auf die y-Achse und die Prozessvariable auf die x-Achse abgebildet. Ist das Ausgabesignal binär und in der booleschen Formel befinden sich Vergleiche von Termen mit zwei kontinuierlichen Variablen, so kann jeder dieser Vergleiche in einem eigenen Graphen verdeutlicht werden. Bei mehr als einer Relation sind mehrere Kurven möglich. Auf die y-Achse abgebildet wird dabei die alleinstehende Variable, d. h. `geschwindigkeit` im vierten Beispiel, und `distanz` und `geschw` im fünften Beispiel. Eine passende Skalierung kann aus der Gleichung oft nicht herausgelesen werden und wird vom Sicherheitsingenieur manuell festgelegt. Bei Regeln mit mehr als zwei kontinuierlichen Variablen lässt sich zumindest ein partieller Graph zeichnen. Hierfür werden schrittweise Variablen auf einen konstanten Wert fixiert, bis nur noch zwei Variablen übrig bleiben, welche wie oben beschrieben dargestellt werden. Die Auswahl der Werte trifft der Sicherheitsingenieur.

Während der Identifikationsphase der unsicheren Regelsignale wird der Regelalgorithmus noch nicht benötigt. Erst bei der Kausalanalyse muss untersucht werden, inwiefern der Regelalgorithmus Ursache des unsicheren Signals sein kann. Ebenso wie beim Entwurf des Prozessmodells kann auch bei der Konstruktion des Algorithmus fehlendes Feedback erkannt und nachträglich eingetragen werden. Da STPA ein iteratives Verfahren ist [Tho13], kann die Suche nach zusätzlichen Feedbackkanälen auch erst nach der Identifikation der unsicheren Regelsignale erfolgen. Dies ermöglicht es, Teile des Regelalgorithmus aus den gesammelten unsicheren Signalen automatisch zu generieren. Eine Voraussetzung hierfür ist, dass die Regelsignale in eine Form gebracht werden, die maschinell verarbeitet werden kann. Insbesondere muss formal spezifiziert werden, unter welchen Bedingungen ein Regelsignal bzw. das Nichtsenden eines Regelsignals unsicher ist. Kontexttabellen sind ideal für diesen Zweck, da sie bereits zu Beginn der Analyse alle Variablenkombinationen enthalten, und der Sicherheitsingenieur lediglich markiert, welche davon unsicher sind. Dieses Konzept wurde bereits erfolgreich angewendet [ST14]. Leider beschränkte sich die Anwendung dabei auf diskrete Variablen. Bei der Übertragung auf kontinuierliche Variablen treten neue Schwierigkeiten auf: Prozessvariablen, die eine unendliche Zahl von Zuständen besitzen, können nicht ohne Weiteres in die Kontexttabelle integriert werden, und die Sicherheit kontinuierlicher Regelsignale hängt nicht nur vom Kontext, sondern auch vom Inhalt des Signals ab.

Es gibt mehrere Möglichkeiten, diese Probleme anzugehen. Ein vergleichsweise unkomplizierter Vorschlag ist, die unsicheren Regelsignale in zwei Phasen zu identifizieren. Bei Signalen, die nur von diskreten Variablen abhängen, kann wie gehabt auf die Kontexttabelle zurückgegriffen werden. Alle anderen Signale werden ohne Hilfestellung untersucht. Automatisch generiert wird nur der Teil

des Regelalgorithmus, der aus der Kontexttabelle hervorgeht. Verbliebene Regeln müssen manuell implementiert werden. Der Nachteil bei diesem Vorgehen ist, dass die Gefahr besteht, unsichere kontinuierliche Signale zu übersehen. Dies wird noch dadurch begünstigt, dass sich die bisherigen Leitwörter für solcherlei Signale schlecht eignen [Cas+18].

Eine zweite Idee ist, alle diskreten Signale mit der Kontexttabelle zu bearbeiten. Hierbei werden bei Bedarf kontinuierliche Prozessvariablen manuell in Wertebereiche unterteilt, welche in die entsprechenden Zellen eingetragen werden. Die Wertebereiche können auch von anderen Variablen abhängen, z B. `temperatur < wunschTemperatur`. Insbesondere diese Eigenschaft trennt das Prinzip von der bisher üblichen Art der Diskretisierung ab. Die Praxistauglichkeit solcher Kontexttabellen haben bereits Abdulkhaleq und Wagner [AW15b] demonstriert. Auch dort werden Formeln verwendet, um zu modellieren, unter welchen Bedingungen das Senden und Nichtsenden von Regelsignalen unsicher ist. Kontinuierliche Prozessvariablen und deren Wertebereiche können bei der Generierung des Regelalgorithmus unverändert übernommen werden, wodurch Regeln wie in Beispiel 4 und 5 entstehen. Nachteile dieser Methode sind, dass die Wertebereiche der Variablen manuell festgelegt werden müssen und dass die Analyse der kontinuierlichen Regelsignale noch immer getrennt stattfindet.

Werden auch kontinuierliche Signale mithilfe der Tabelle untersucht, kann der Wertebereich dieser entweder an einem Stück analysiert oder aufgeteilt und getrennt bearbeitet werden. Eine Eigenschaft von Regelsignalen mit mehr als zwei möglichen Zuständen ist, dass nicht pauschal festgelegt werden kann, ob diese sicher oder unsicher sind, da das oft auch vom Inhalt des Signals abhängt. Beispielsweise erscheint `Schub` im Parkzustand zunächst unsicher zu sein. Jedoch ist auch 0 Teil des Wertebereichs und führt hier zu keiner Gefährdung. Aus diesem Grund muss zusätzlich zu der Entscheidung ”unsicher” eintragen werden, welchen Wertebereich dies betrifft. Trotz dieser Information ist es nicht möglich, zu kontinuierlichen Signalen Regeln zu generieren.

Eine genauere Erforschung der Identifikationsmethoden von unsicheren kontinuierlichen Regelsignalen und der Möglichkeit, daraus Regelalgorithmen zu generieren, ist nicht Teil dieser Arbeit. Es scheint jedoch, dass in Bezug auf die Reihenfolge eine Analyse der unsicheren Signale der Konstruktion des Algorithmus vorzuziehen ist. Die Ermittlung fehlender Sensoren und die Kausalanalyse erfolgen im Anschluss daran. Ein mögliches Vorgehen wird im nächsten Abschnitt anhand eines Beispiels demonstriert.

4.2.2 Demonstration des Konzepts

Das in Abschnitt 4.2.1 vorgestellte Modellierungskonzept soll nun an dem Regelalgorithmus einer Drohne veranschaulicht werden. Viele moderne Drohnen besitzen die Funktion, bei drohender Akkuerschöpfung automatisch eine Notlandung durchzuführen. Die hier gezeigte STPA-Analyse wird sich auf das für diese Funktion verantwortliche Modul beschränken. Dabei werden außerdem Teile des Prozesses abgekürzt, die für die Demonstration des Konzepts irrelevant sind. Für eine detailliertere Analyse sei der Leser auf Kapitel 6 verwiesen.

Die hier betrachteten Verluste sind ”L-1: Die Drohne nimmt Schaden” und ”L-2: Die Drohne bricht ihren Einsatz ab”. Die dazu gehörigen Gefährdungen sind ”H-1: Die Drohne befindet sich im freien Fall [L-1]” und ”H-2: Die Drohne reagiert während des Einsatzes nicht auf die Steuerung des Piloten [L-2]”. Hieraus lassen sich die Sicherheitsanforderungen ”S-1: Die Drohne darf nicht in den freien Fall kommen [H-1]” und ”S-2: Die Drohne muss während des Einsatzes auf die Steuerung des

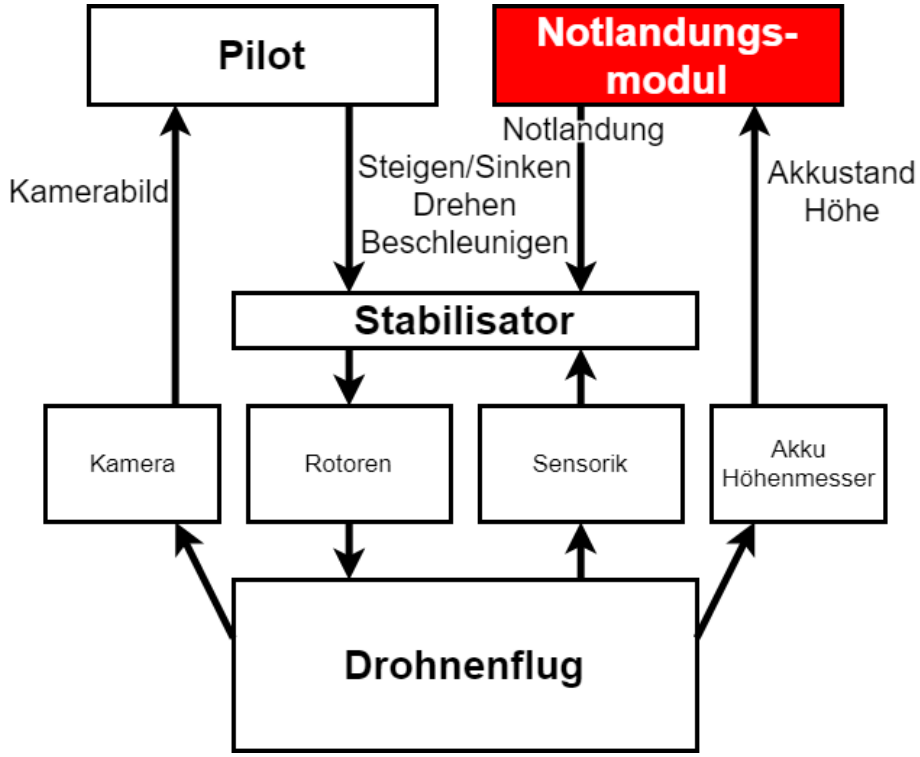


Abbildung 4.1: Regelkreishierarchie der Drohne

Piloten reagieren [H-2]" ableiten. Somit ergeben sich die folgenden Verantwortlichkeiten für das Notlandungsmodul: "R-1: Die Drohne zur Landung bringen, bevor der Akku erschöpft ist [S-1]" und "R-2: Dem Piloten während des Einsatzes die Steuerung erlauben [S-2]. Es ist wichtig, in allen Definitionen auf die entsprechenden Referenzen zu verweisen, damit auch bei komplexen Systemen immer klar ist, zu welchem Zweck ein Eintrag da ist. Hierfür werden die Definitionen mit Buchstaben kategorisiert, und mit Zahlen nummeriert. Die Buchstaben *L*, *H*, *S* und *R* stehen für die englischen Begriffe *Loss*, *Hazard*, *Safety Constraint* und *Responsibility*.

Abb. 4.1 zeigt eine vereinfachte Regelkreishierarchie der Drohne. Im Normalbetrieb sendet eine an der Drohne befestigte Kamera Bildmaterial an den Piloten, der anhand diesem die Drohne über die Regelsignale *Steigen/Sinken*, *Drehen* und *Beschleunigen* kontrolliert. Der Stabilisator steuert anhand dieser Signale die vier Rotoren der Drohne, aber achtet gleichzeitig darauf, dass sich die Drohne nie zu stark neigt, nicht durch Wind vom Kurs abgebracht wird, und allgemein alle Dreh- und Antriebsmomente zurücksetzt, sobald das entsprechende Regelsignal 0 wird. Zu diesem Zweck bekommt der Stabilisator Feedback von Gyroskopen, Beschleunigungsmessern, Drehratensensoren, GPS-Empfängern, Neigungssensoren, und Kompassen [Win16]. Das hier analysierte Notlandungsmodul ist in der Abbildung rot markiert. Es entscheidet anhand des Akkustands, ob eine Notlandung durchgeführt werden muss. In einem solchen Fall wird diese Information über ein entsprechendes Signal an den Stabilisator weitergegeben, der daraufhin sämtliche Eingaben des Piloten ignoriert, und stattdessen auf die Signale des Notlandungsmoduls reagiert.

Die für die Sicherheit relevanten Prozessvariablen sind u. a. *Akkustand*, der die verbleibende Akkuleistung in Prozent enthält, und die *Flughöhe* der Drohne in Metern. Da beide Variablen kontinuierlich sind, muss für die Kontexttabellen ein Wertebereich festgelegt werden. Tabelle 4.1 enthält

Akkustand	**Flughöhe**	**Nichtsenden verursacht Gefährdung**	**Senden verursacht Gefährdung**	**Zu früh, zu spät, oder falsche Reihenfolge**	**Zu früh gestoppt, zu lange gesendet**
≤10%	0m				
≤10%	> 0m	UCA-1: kein Notlandungssignal bei niedrigem Akkustand [H-1]		UCA-2: Notlandungssignal erst, nachdem Akku leer ist [H-1]	UCA-3: Notlandungssignal gestoppt, bevor Drohne gelandet ist [H-1]
>10%	egal		UCA-4: Notlandungssignal, obwohl Drohne noch im Einsatz ist [H-2]		UCA-5: Notlandungssignal nicht gestoppt, nachdem Akku aufgeladen ist [H-2]

Tabelle 4.1: Kontexttabelle für `Notlandung`

Algorithmus 4.1 Regelalgorithmus des Notlandungsmoduls

$$notlandung = [berechneAkkustand(volt) \leq 10]$$
$$höhe = aktuelleHöhe - startHöhe$$
$$steigen/sinken = max(-100, -8 * höhe - 20)$$
$$drehen = 0$$
$$querBeschleunigen = 0$$
$$längsBeschleunigen = 0$$

alle identifizierten unsicheren Regelsignale zum *Notlandungssignal*, während Tabelle 4.2 unsichere Regelsignale bezüglich des *Steigen/Sinkensignals* anzeigt. *Notlandung* ist binär, wohingegen *Steigen/Sinken* einen Wert zwischen -100 und 100 enthält, der die gewünschte in Änderung der Flughöhe in prozentualer Schubkraft repräsentiert. Der unsichere Wertebereich ist in der Tabelle in eckige Klammern gesetzt. Insgesamt wurden zwölf unsichere Regelsignale gefunden, davon fünf Notlandungssignale und sieben Steigen-/Sinkensignale.

Die Regel des Notlandungssignals kann anhand der Kontexttabelle automatisch generiert werden. Händisch erstellt werden die Regeln der verbliebenen Signale *Steigen/Sinken*, *Drehen*, *Quer Beschleunigen* und *Längs Beschleunigen*, wobei die letzteren drei für die Landung nicht benötigt werden und auf 0 gesetzt sind. Algorithmus 4.1 zeigt den fertigen Regelalgorithmus des Notlandungsmoduls. Der Akkustand kann aus der gegenwärtig anliegenden Spannung berechnet werden, weil diese bei sinkender Kapazität leicht abfällt. Da es bei der Höhe des Bodens regionale Unterschiede gibt, wird die Flughöhe aus der Differenz der Starthöhe und der aktuellen Höhe berechnet. Die Sinkrate wurde so konfiguriert, dass sie während der letzten 10 Meter linear abfällt und bei 0 Metern -20% erreicht, um ein zu schnelles Sinken vor der Landung (UCA-8) zu verhindern.

Eine Kausalanalyse des Steigen-/Sinkensignals ergibt, dass die Berechnung der Flughöhe nur auf ebenen Flächen funktioniert. Bei abschüssigem Gelände oder bei Gelände mit erhöhten oder vertieften Plattformen führt diese Art der Berechnung zu den unsicheren Signale UCA-7, UCA-8, UCA-9 und UCA-12. Die Analyse des Notlandungssignal zeigt Schwächen bei der Konvertierung

Akkustand	**Flughöhe**	**Nichtsenden verursacht Gefährdung**	**Senden verursacht Gefährdung**	**Zu früh, zu spät, oder falsche Reihenfolge**	**Zu früh gestoppt, zu lange gesendet**
≤10%	0m		UCA-6: [Steigen-/Sinken > 0] Erneutes Steigen nach erfolgter Notlandung [H-1]		
≤10%	>0m, ≤5m	UCA-7: [Steigen-/Sinken ≥0] Kein Sinken bzw. Steigen kurz vor der Landung [H-1]	UCA-8: [Steigen-/Sinken < -50] Zu schnelles Sinken kurz vor der Landung [H-1]		UCA-9: [Steigen-/Sinken < -50] Zu späte Verringerung der Sinkrate [H-1]
≤10%	> 5m		UCA-10: [-50 < Steigen/Sinken < 0] Sinkrate zu langsam [H-1]	UCA-11: [Steigen/Sinken < 0] Sinksignal zu spät nach Notlandungssignal [H-1]	UCA-12: [-50 < Steigen/Sinken < 0] Sinkrate zu früh verlangsamt [H-1]
>10%	egal				

Tabelle 4.2: Kontexttabelle für `Steigen/Sinken`

von der Spannung zur verbliebenen Akkuleistung. Da der Spannungsverlauf nicht nur von der Akkuleistung, sondern auch vom Alter des Akkus abhängig ist, ist es nicht unwahrscheinlich, dass die Notlandung eines Tages erst aktiviert wird, wenn die Akkuleistung nicht mehr für eine sichere Landung ausreicht. Hierdurch käme es zu dem unsicheren Regelsignal UCA-2.

Abb. 4.2 zeigt den zum Steigen-/Sinkensignal gehörigen Funktionsgraphen. Im Gegensatz zur Regel sieht man hier sofort, an welcher Stelle die Drohne ihre Sinkrate verringert. Für die automatische Generierung sind Informationen über das Verhalten der Funktion `max` nötig. In diesem Fall können diese anhand des konventionellen Namens bezogen werden. Allgemein ist dies jedoch nicht möglich. Beispielsweise gibt es keinen Weg, ohne menschliche Hilfe die Ausgabewerte der Funktion `berechneAkkustand` zu ermitteln.

4.2.3 Bewertung

Die oben durchgeführte STPA-Analyse des Notlandungssmoduls hat als abschließendes Ergebnis innerhalb des Regelalgorithmus zwei verschiedene Typen von Ursachen für unsichere Regelsignale identifiziert. Der erste Typ beschreibt Makel am Algorithmus, die dadurch entstehen, dass Informationen nicht miteinkalkuliert werden, die in bestimmten Fällen gebraucht werden, um einen Fehler in dem berechneten Ergebnis zu vermeiden. *STPA Handbook* bezeichnet dies als ”Fehlerhafte Spezifikation des Regelalgorithmus”. Der zweite Typ beschreibt Makel am Algorithmus, die dadurch entstehen, dass die Berechnung in bestimmten Fällen ungenau ist. *STPA Handbook* nennt das ”Regelalgorithmus wird mit der Zeit ungeeignet aufgrund von Änderungen oder Abbau”. Dies

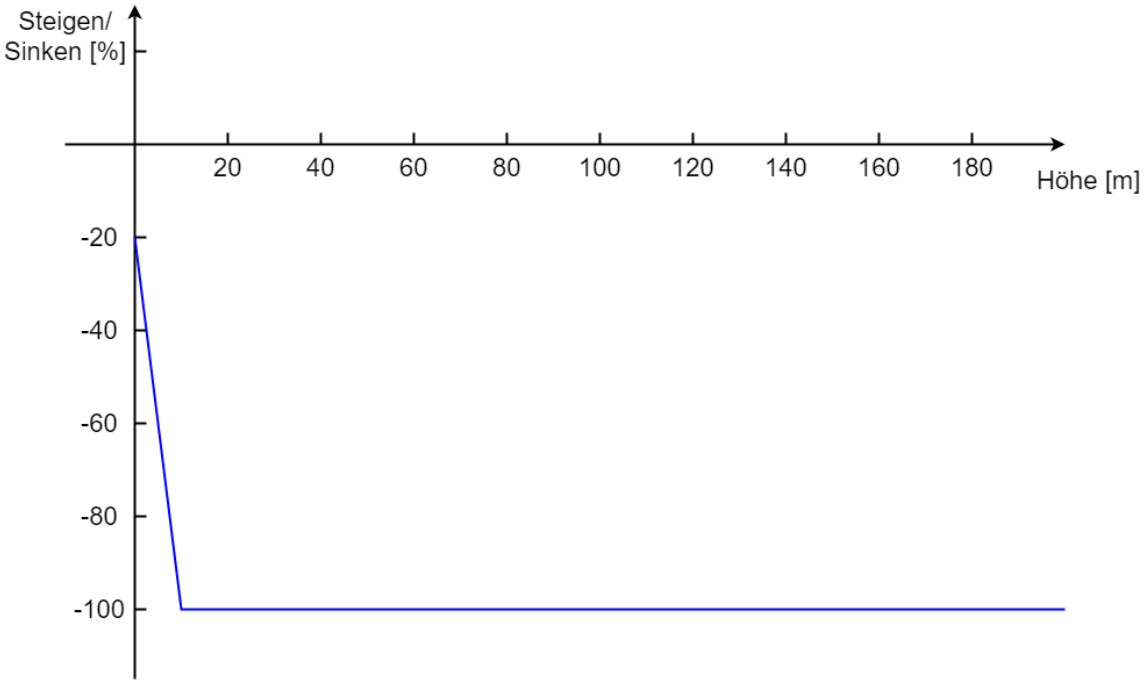


Abbildung 4.2: Funktionsgraph des Steigen-/Sinkensignals

sind jedoch nicht alle Arten von Ursachen, die durch das Konzept gefunden werden können. Bei der Analyse komplexerer Systeme ist die Identifikation weiterer Typen zu erwarten. Ein Beispiel hierfür sind Systeme mit menschlichen Reglern, bei denen außerdem zu schwierige Berechnungen als Ursache für unsichere Regelsignale bestimmt werden könnten.

Das Modellierungskonzept hat zweifellos das Potential, Ursachen zu ergründen, die mit herkömmlichen Methoden schwierig zu finden sind. In vielen Fällen können einige der Ursachen jedoch auch bei eingehender Untersuchung des Prozessmodells gefunden werden. So lässt sich etwa die von dem Startpunkt ausgehende Berechnung der Flughöhe dem existierenden Leitsatz ”Regler bekommt korrektes Feedback, aber interpretiert es falsch” zuordnen. Die Integration des Konzepts in eine STPA-Analyse ermöglicht eine genauere Untersuchung des Regelalgorithmus, jedoch verkompliziert sie den Prozess zusätzlich. Der Entwurf der kontinuierlichen Kontexttabellen und des Regelalgorithmus kostet insbesondere bei komplexen Systemen viel Zeit, die stattdessen für andere Analysephasen genutzt werden könnte. Dabei garantiert die Modellierung des Algorithmus nicht, dass alle Ursachen für unsichere Signale gefunden werden. Beispielsweise ist das Modell zu abstrakt, um ein Nichtsenden eines wichtigen Regelsignals aufgrund von schlechtem Prozess-Scheduling zu identifizieren. Ein solches Szenario trat z. B. während der Mission des Mars Pathfinders auf, als prioritäre Prozesse nicht weiter ausgeführt wurden, weil sie auf Ressourcen von nachrangigen Prozessen warteten [Jon97]. Solche und ähnliche Probleme gehen durch die Abstrahierung in dem Modell verloren. Alles in allem soll das Konzept mehr ein optionales als ein obligatorisches Element darstellen, dessen Durchführungsdetails an das System angepasst werden können.

In diesem Abschnitt wurde gezeigt, dass sich der Regelalgorithmus als eine Menge von Formeln modellieren lässt, die die Abhängigkeiten der Regelsignale von den Prozessvariablen verdeutlichen. Im Folgenden soll besprochen werden, inwiefern sich dieses Konzept auf Aktoren übertragen lässt.

4.3 Übertragung des Konzepts auf Aktoren

In Bezug auf ihre Aufgabe im Gesamtbild der STPA-Analyse existieren einige Parallelen zwischen Regelalgorithmen und Aktoren. Beide Komponenten haben die Funktion, eine Menge von Eingabewerten auf einen Ausgabewert abzubilden. Daher muss in Erwägung gezogen werden, ob das für Regelalgorithmen entwickelte Konzept mit oder ohne kleinere Anpassungen auf die Modellierung von Aktoren übertragen werden kann. Ist dies möglich, können bei Bedarf auch Aktoren zu einem höheren Detailgrad analysiert werden, was zu einer Steigerung der gefundenen Ursachen für unsichere Signale führt.

Aktoren befinden sich im Regelkreis zwischen Reglern und Prozessen. Eingaben erhalten sie ausschließlich vom Regler in Form von Regelsignalen, während Ausgaben ausschließlich in Richtung Prozess gehen. Die eingehenden Regelsignale können sowohl diskreter als auch kontinuierlicher Natur sein und beeinflussen die Ausgabe direkt. Die Ausgabe wiederum manipuliert die Variablen des Prozesses. Einige dieser Variablen werden von Sensoren eingelesen und im Prozessmodell des Reglers abgespeichert, andere bleiben unbekannt. In manchen Fällen ist es möglich, ohne spezielle Messung auf indirekt veränderte Variablen zu schließen. Beispielsweise bewirkt das Lenken eines Fahrzeugs nicht nur eine (per GPS-Empfänger messbare) Änderung der Fahrtrichtung, sondern auch eine Seitenkraft, auf die über Lenkwinkel, Geschwindigkeit und Fahrzeuggewicht geschlossen werden kann, um bei Bedarf abzubremsen. Die meisten indirekt beeinflussten Variablen sind für die Risikoanalyse nicht von Bedeutung (z. B. Reibungswärme oder Magnetfelder elektrischer Bauteile). Obwohl die physische Welt von Grund auf kontinuierlich ist, gibt es auch bei den Ausgaben des Aktors diskrete Fälle: Die Lichtstärke einer Kontrollleuchte anzugeben, ist eher eine Verkomplizierung als eine tatsächliche Hilfe. Ebenso unnötig ist die kontinuierliche Bezifferung eines Mechanismus, der das Drücken eines Knopfes verhindert. Sinnvoller ist, die Regelsignale hier eins-zu-eins auf die physische Welt abzubilden.

Mithilfe von Regelalgorithmen werden häufig komplexe und von vielen Variablen abhängige Entscheidungen getroffen. Bei menschlichen Reglern kommen darüber hinaus Einflüsse von Arbeitsplatz, Persönlichkeit und Erfahrungen dazu. Demgegenüber sind Aktoren eher primitive Komponenten, deren simple Konvertierungsaufgabe sich schwerlich als Entscheidung bezeichnen lässt. Auf Softwarefehler zurückzuführende Aussetzer des Regelalgorithmus, wie sie beim Mars Pathfinder aufgetreten sind, sind bei reinen Aktoren nicht möglich (bei Aktoren mit eigenem Puffer oder ähnlichen Steuerelementen werden diese als selbstständige Regler betrachtet). Ein weiterer Unterschied liegt darin, dass Aktoren oft diskrete in kontinuierliche Variablen konvertieren. Ein Beispiel hierfür ist ein Tür-Auf-Signal an den Motor einer Zugtür, der die Tür daraufhin temporär mit einer bestimmten Geschwindigkeit verschiebt. Anders als bei Lampen kann hier die Modellierung mit kontinuierlichen Werten sinnvoll sein, da zu schnelle Geschwindigkeiten ein Sicherheitsrisiko darstellen. Das Modellierungskonzept muss um diese Funktionalität erweitert werden, um auch in solchen Fällen anwendbar zu sein. Bei Reglern unterscheidet man zwischen Menschen und Maschinen. Im Gegensatz dazu sind Aktoren ausnahmslos Maschinen und ihre Konvertierungen weisen im Normalfall eine hohe Präzision auf. Es gibt jedoch ebenso Fälle, in denen die Ausgaben umweltbedingt an Genauigkeit einbüßen (z. B. ist die Verlangsamung beim Bremsen stark vom Untergrund abhängig). Des Weiteren verlieren Aktoren durch Abnutzung und Witterungserscheinungen an Zuverlässigkeit, was sie eher mit menschlichen Reglern vergleichbar macht. Trotz einiger Unterschiede gibt es aber letztendlich keine Eigenschaft, die eine Übertragung des Modellierungskonzepts auf Aktoren verhindern würde.

Ein Aktor hat im Allgemeinen genau eine Zuständigkeit. Auch komplexe Aktoren, wie etwa Roboterarme, können im Modell in einzelne Komponenten untergliedert werden. Der Roboterarm z. B. besteht aus zahlreichen synthetischen Gelenken, die unabhängig voneinander angesteuert werden können. Ein einzelnes Gelenk kann sich nur in eine Richtung bewegen, weswegen ein Ausgabewert genügt. Dadurch weisen die Regeln der Aktoren eine vergleichsweise niedrige Komplexität auf, und sind oft nur von ein oder zwei Eingabewerten abhängig. Trotz der Möglichkeit dazu wird das Konzept jedoch vorerst nicht eingeschränkt, d. h. alle in Regelalgorithmen mögliche Regeln sind auch bei Aktoren möglich. Dies dient nicht nur dazu, auf Ausnahmefälle vorbereitet zu sein, sondern erleichtert auch das Erlernen des Modellierungskonzepts. Auf der anderen Seite ist aber eine unwesentliche Erweiterung notwendig, die die Konvertierung von diskreten zu kontinuierlichen Variablen betrifft. Hierfür wird zunächst die Syntax angepasst, indem folgende Regel geändert wird:

```
term := boolVar | kontVar | '-' term | konstante | funktion | term arithOp term
        | '(' term ')'
```

Somit können Terme neben kontinuierlichen Variablen auch zu booleschen Variablen abgeleitet werden. Semantisch gesehen wird ein wahrer Wert als 1 und ein falscher Wert als 0 betrachtet. Dies ermöglicht das Negieren und Aktivieren von bestimmten Teilen einer Formel. Um die Regeln von Reglern und Aktoren konsistent zu halten, kann die Verwendung von booleschen Variablen in Termen auch bei Reglern erlaubt werden. Aufgrund der voraussichtlich sehr geringen Zahl von Regeln pro Aktor wird vorgeschlagen, Aktoren zu gruppieren und deren Regeln in einem Algorithmus zusammenzufassen. Um die Regeln zuordnen zu können, werden sie mit dem Namen des Aktors eingeleitet. Weiterhin wird die Gesamtheit der Regeln hier Konvertertabelle genannt, um sie begrifflich vom Regelalgorithmus abzugrenzen.

Algorithmus 4.2 zeigt eine Konvertertabelle am Beispiel der Drohne. Sie enthält die vier Rotoren, sowie den Motor, der zum Kippen der Kamera zuständig ist. Die Namen der Aktoren befinden sich links der Regeln und sind fett geschrieben. Bei der Konversion werden Zwischenergebnisse übersprungen und nur der für die Analyse relevante Wert angegeben. Tatsächlich würde das Schubsignal zunächst einen elektrischen Widerstand bestimmen, der den Stromfluss des Rotormotors regelt. Basierend auf dem Stromfluss ist die Stärke des Magnetfeldes im Motor, welcher sodann eine Kraft auf den Rotor ausübt, die diesen beschleunigt. Die Beschleunigung endet, sobald Luft- und Reibungswiderstand zu groß werden. In der Konvertertabelle angegeben ist der Auftrieb an diesem Punkt in Newton. Sie ergibt sich aus dem zwischen 0 und 1 liegenden Schubsignal und der Luftdichte in Kilogramm pro Kubikmeter. Dabei ist die Luftdichte eine Variable, die von der Drohne nicht gemessen wird, aber trotzdem ihre Einflussnahme auf den Prozess verändert, und daher miteinbezogen werden muss. Bei der Kamera wird angegeben, wie schnell sie ihre Ausrichtung ändert. Die Änderung steht in Radianten pro Sekunde. Zu Demonstrationszwecken werden hier zwei diskrete Regelsignale `kameraHoch` und `kameraRunter` verwendet. Ist eines dieser Signale 1, dreht sich die Kamera in die entsprechende Richtung.

Die Konvertertabelle zeigt, dass sich das in Abschnitt 4.2 entwickelte Konzept nicht nur für Regelalgorithmen eignet, sondern problemlos auf Aktoren übertragen werden kann. Dabei können kleinere Anpassungen, wie z. B. die Integration von booleschen Variablen in kontinuierliche Terme, den Anwendungsbereich erweitern.

Algorithmus 4.2 Konvertertabelle der Drohnenaktoren

Rotor 1:	$auftrieb1 = 0.345 * schub1 * luftDichte$
Rotor 2:	$auftrieb2 = 0.345 * schub2 * luftDichte$
Rotor 3:	$auftrieb3 = 0.345 * schub3 * luftDichte$
Rotor 4:	$auftrieb4 = 0.345 * schub4 * luftDichte$
Kameramotor:	$winkelÄnderung = 0.35 * kameraHoch - 0.35 * kameraRunter$

4.4 Zusammenfassung

In diesem Kapitel wurde ein ein Modellierungskonzept für Regelalgorithmen dargelegt, das vier Kriterien erfüllt: *Verständlichkeit*, *Flexibilität*, *Skalierbarkeit* und *Erweiterbarkeit*. Das Konzept basiert auf mehreren bereits existierenden Ansätzen, die in einem umfassenden Vergleich gegenübergestellt wurden. Ein auf diesem Konzept aufbauendes Modell lässt sich mithilfe einer formalen Grammatik erstellen und repräsentiert eine Menge von Formeln. Enthalten diese genau zwei kontinuierliche Variablen, kann daraus ein Funktionsgraph generiert werden. Es wurde eine kurze STPA-Analyse an dem Notlandungsmodul einer Drohne durchgeführt, die gezeigt hat, dass mit dem Modellierungskonzept Ursachen für unsichere Regelsignale gefunden werden können, die sonst schwierig zu finden sind. Außerdem wurde demonstriert, wie sich das Konzept auf Aktoren übertragen lässt.

5 Implementierung in XSTAMPP

Dieses Kapitel dokumentiert die Implementierung mehrerer der in Kapitel 4 vorgeschlagener Funktionen. Zu den implementierten Funktionen gehören das Erstellen, Einsehen, Suchen, Bearbeiten und Löschen von Regeln, sowie die syntaktische Analyse des Regelinhalts. Außerdem wurde eine Autovervollständigungsfunktion für die Namen der Prozessvariablen, und eine Möglichkeit zur Anzeige der Regeln in einem Graphen integriert. Für die Integration von Aktoren wurden darüber hinaus dieselben Funktionen auch in Form einer Konvertertabelle umgesetzt, mit der die Beziehungen zwischen ein- und ausgehenden Regelsignalen von Aktoren modelliert werden können. Nicht implementiert wurde die automatische Generierung von Regeln anhand der unsicheren Regelsignale, da die entsprechende Tabelle bisher keine Kontexte aufnehmen kann. Diese Funktion kann jedoch zu einem späteren Zeitpunkt problemlos ergänzt werden. Der folgende Bericht enthält zum Teil technische Beschreibungen der Funktionsweise und setzt entsprechende Fachkenntnisse voraus. Für eine beispielhafte Verwendung der neuen Funktionen sei der Leser auf Kapitel 6 verwiesen.

5.1 Regeltabelle

Im Rahmen einer vollständigen STPA-Analyse durchläuft der Sicherheitsingenieur vier Phasen. Die Navigationsleiste von XSTAMPP ist dieser Phasen entsprechend gegliedert. Da die Definition der Regelalgorithmen zu der vierten Analysephase gehört, ist es sinnvoll, die dazugehörige Funktionalität als einen Unterpunkt in diesen Teil der Navigationsleiste zu integrieren. In bestehenden STPA-Analyseprogrammen ist es teilweise möglich, das eng mit den Regelalgorithmen verwandte Prozessmodell direkt in den Regelkreisen anzuzeigen. Ein Nachteil dieser Darstellungsweise ist jedoch der Mangel des zur Verfügung stehenden Platzes, wodurch der Regelkreis schon bei simplen Prozessmodellen an Übersichtlichkeit einbüßt. XSTAMPP wird daher eine eigene Seite zur Bearbeitung der Prozessmodelle besitzen. Die Implementierung der Regelalgorithmen soll zu diesem Schema konsistent sein und die Funktionalität in eine eigene Seite integrieren, die sich in der Navigationsleiste unter Phase 4 erreichen lässt.

5.1.1 Implementierung

Das Front-End von XSTAMPP ist mit dem Open-Source-Framework Angular umgesetzt. In diesem ist die Struktur der Webseite in sogenannte *Module* unterteilt, die sich hierarchisch schachteln lassen. Die tatsächlich sichtbaren Elemente befinden sich in *Komponenten*. In XSTAMPP existiert eine Komponente für jede über der Navigationsleiste erreichbare Unterseite. Die Navigationsleiste wiederum ist in einer Komponente eine Ebene darüber definiert. Für die Erstellung des auf die Regelalgorithmen verweisenden Eintrags in der Navigationsleiste wurden hier kleinere Änderungen an der Darstellung und am Routing vorgenommen. Der Regelalgorithmus selbst wird in Form einer Tabelle angezeigt, um so Platz für zusätzliche Informationen zu den einzelnen Regeln zu

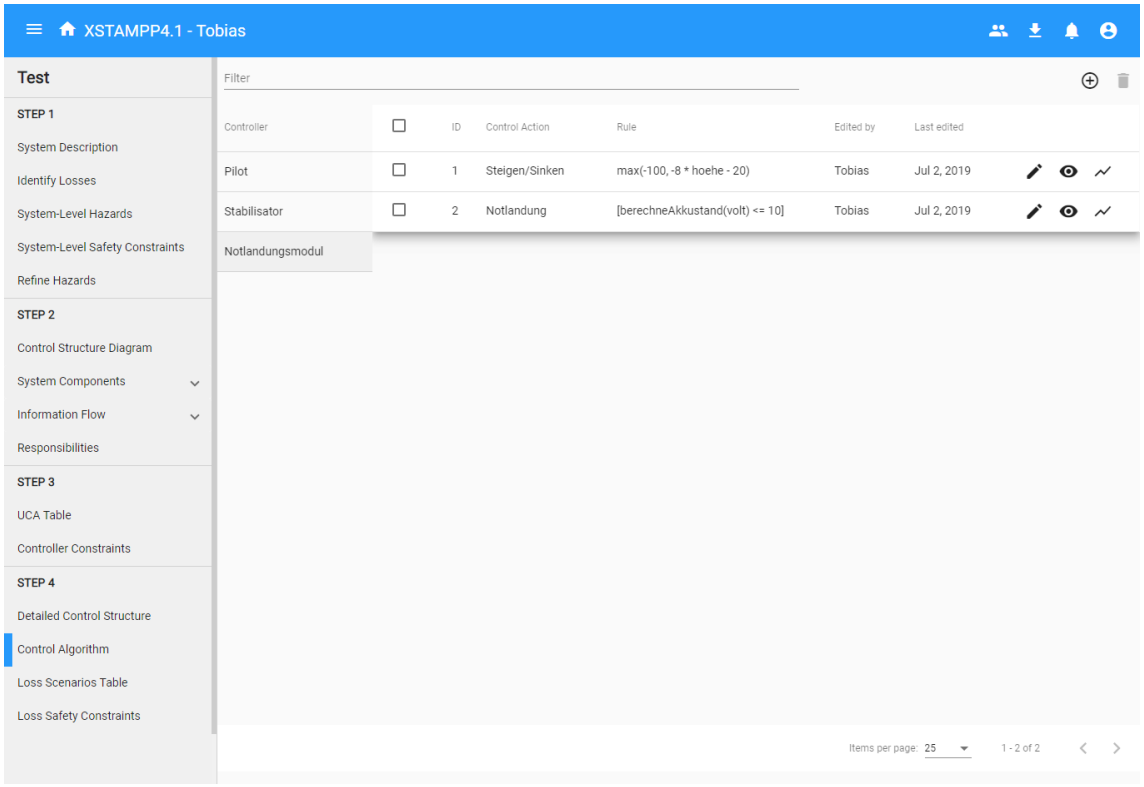


Abbildung 5.1: Tabelle für die Regeln des Regelalgorithmus

bieten. Abb. 5.1 veranschaulicht dieses Konzept. Die Bearbeitung der Einträge findet in einem ausklappbaren Fenster statt. Auch dies ist konsistent zu der aktuellen Implementierung der anderen Analyseergebnisse. Die Umsetzung erfolgt in einer `.html`-Datei, in der auf die abstrakte Komponente *mat-drawer* verwiesen wird, die mit Angular mitgeliefert wird und den Ausklappvorgang implementiert. Innen befinden sich Verweise auf eine speziell für XSTAMPP entwickelte Tabellenkomponente. Diese Verweise nennen auch die Namen der Prozeduren, die bei den verschiedenen Nutzereingaben aufgerufen werden, und in einer TypeScript-Datei deklariert sind. Beim Aufruf des Fensters zur Regelbearbeitung werden diesem die Namen und Datentypen der benötigten Einträge übergeben. Ist die Bearbeitung abgeschlossen, wird die in der Regelalgorithmuskomponente deklarierte Speicherprozedur aufgerufen. Auf dem Weg zum Back-End werden mehrere Schichten durchlaufen, die im Folgenden kurz vorgestellt sind:

(1) Die **Regelalgorithmuskomponente** bringt die Tabelle in den Ausgangszustand, benachrichtigt den Nutzer über den Erfolg oder Misserfolg der Bearbeitung, und entscheidet, ob eine alte Regel editiert oder eine neue Regel erstellt wurde.

(2) Der **Regelalgorithmen-Service** bereitet die Adresse vor, an die die Daten gesendet werden, und spezifiziert den genauen Inhalt der Fehlermeldung, die bei Misserfolg angezeigt wird.

(3) Der **Anfrage-Service** bereitet mithilfe des erstellten Pfads und einer für den Zugang benötigten Authentifizierungstokens die HTTP-Anfrage vor, und sendet diese an das Back-End.

(4) Die **REST-Schnittstelle** im Back-End empfängt die Anfrage und wandelt sie zurück in ein Java-Objekt um. Ist die Anfrage letztendlich erfolgreich, wird von hier eine Benachrichtigung an alle Clients gesendet, die sich als Beobachter registriert haben.

(5) Der **Daten-Service** fügt dem Regelobjekt Metadaten wie den Namen des Bearbeiters und den Zeitpunkt der Bearbeitung an.

(6) Die **Hibernate-Klasse** bereitet schließlich die Datenbankanfrage vor und sendet diese an die Datenbank. Nach Erhalt der Antwort werden die Schichten ein weiteres Mal rückwärts durchlaufen.

Bei einer Änderung der Daten werden diese mithilfe einer Beobachterfunktion in die Tabelle geladen und eine Benachrichtigung gesendet, die den Benutzernamen des Bearbeiters enthält. Ebenso wie der Anfrage-Service benötigt auch die Beobachterfunktion ein Authentifizierungstoken, um Zugriff auf die Daten zu erhalten. Die für die Löschung von Regeln zuständige Prozedur macht Gebrauch von dem Service, der die Datensatzsperren verwaltet. In Regeln ist nicht der Name des berechneten Signals, sondern dessen ID gespeichert, anhand welcher der Name beim Laden der Tabelle miteingelesen wird. So bleiben die Regeln auch bei Namensänderungen repräsentativ. Für die Eingabe des rechten Teils wurde ein Freitextfeld platziert, das Zeilenumbrüche unterstützt und seine Größe dynamisch an die Formel anpasst. In dieses Feld wurde mithilfe der Programmbibliothek *Angular Mentions* [Mac18] eine Funktion eingebunden, mit der eine automatische Vervollständigung eingegebener Prozessvariablen möglich ist.

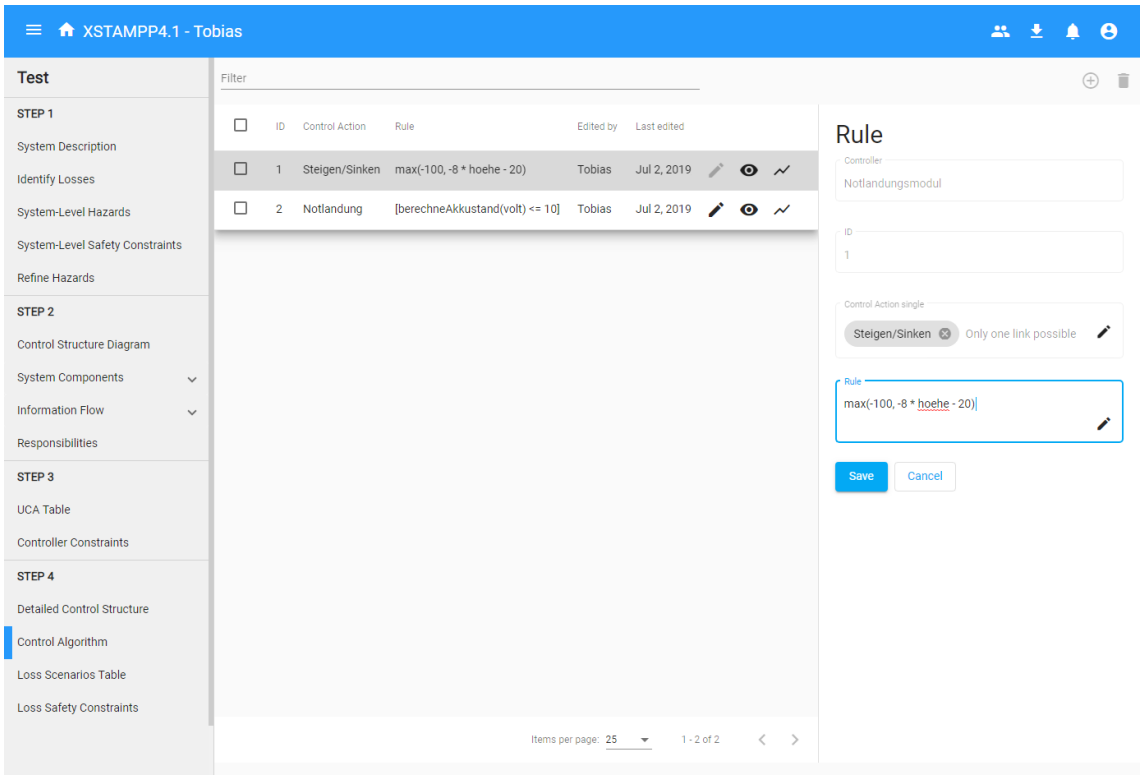


Abbildung 5.2: Bearbeitung einer Regel in einem ausklappbaren Fenster

5.1.2 Benutzeroberfläche

Die Regelalgorithmenverwaltung lässt sich wahlweise über den entsprechenden Menüpunkt in der Navigationsleiste, oder durch Anfügen von ”control-algorithm” an die Projekt-ID in der Adresszeile des Browsers erreichen. In der Tabelle befinden sich die Regeln jeweils eines Reglers, welcher über die Schaltflächen auf der linken Seite ausgewählt wird. Durch Anklicken der Spaltenüberschriften lassen sich Regeln nach ID, Regelsignal, Regelinhalt, Bearbeiter oder Bearbeitungszeitpunkt sortieren. Zudem steht in der linken oberen Ecke eine Suchleiste zur Verfügung. Es existieren Schaltflächen zum Erstellen, Bearbeiten und Einsehen von Regeln, deren Betätigung ein Fenster auf der rechten Seite öffnet, das eine Eingabemaske für die Regel selbst und den Namen des Signals enthält. Diese ist in Abb. 5.2 zu sehen. Das Regelsignal wird aus einer Drop-Down-Liste ausgewählt, die alle Signale des aktuell ausgewählten Reglers enthält, während die Eingabe des Regelinhalts in einem Freitextfeld erfolgt. In diesem kann durch Eingabe von `@` die Autovervollständigung aufgerufen werden, die eine Liste aller Prozessvariablen anzeigt, dessen Anfangsbuchstaben der bisher eingegeben Zeichenfolge entsprechen. Wird die Regel nur eingesehen, ist die Maske ausgegraut und lässt keine Bearbeitung zu. Dies ist auch der Fall, wenn eine Bearbeitung angefordert wurde, aber aufgrund eines von einem anderen Nutzer initiierten Bearbeitungsvorgangs nicht zugelassen wurde. Das Löschen von Regeln erfolgt über eine Schaltfläche rechts neben der Schaltfläche, die für die Erstellung zuständig ist.

5.2 Funktionsgraph

Zur Visualisierung der Regeln wurde eine Funktion implementiert, mit der aus jeder vom Nutzer definierten Regel ein Graph generiert werden kann, welche den Anforderungen genügt. Diese Funktion stellt eine Erweiterung der Regelalgorithmenverwaltung dar, und sollte somit direkt von dort erreichbar sein. Auch während der Anzeige des Graphen muss die vollständige Regel sichtbar sein.

5.2.1 Implementierung

Für die Implementierung des Funktionsgraphen wurde die Programmbibliothek *Function Plot* [Pop15] verwendet. Diese interpretiert eine als Zeichenkette übergebene Formel und konstruiert daraus die passende Kurve. Es werden eine große Zahl Funktionen erkannt und korrekt ausgewertet, darunter z. B. auch die Gamma-Funktion und die Bellschen Zahlen. Zusätzlich werden Parameter übergeben, die die Beschriftungen der Achsen, die Überschrift und das Renderverfahren bestimmen. Obwohl *Function Plot* den Hauptteil der Interpretation übernimmt, müssen die Formeln von einem vorausgehenden Parsing-Algorithmus analysiert und angepasst werden. Dieser durchläuft mehrere Phasen, die im Folgenden beschrieben sind:

(1) Zunächst wird die Regel auf ihre syntaktische Korrektheit überprüft. Für diesen Zweck wurde die in Abschnitt 4.2.1 angegebene Grammatik mit dem Grammatikparser *nearley* [Cha14] kompiliert und als `javascript`-Datei eingebunden. Ist die Syntax der Regel inkorrekt, wird dies in einem Informationsfenster angezeigt. Zusätzlich wird die Stelle markiert, die eine Korrektur erfordert.

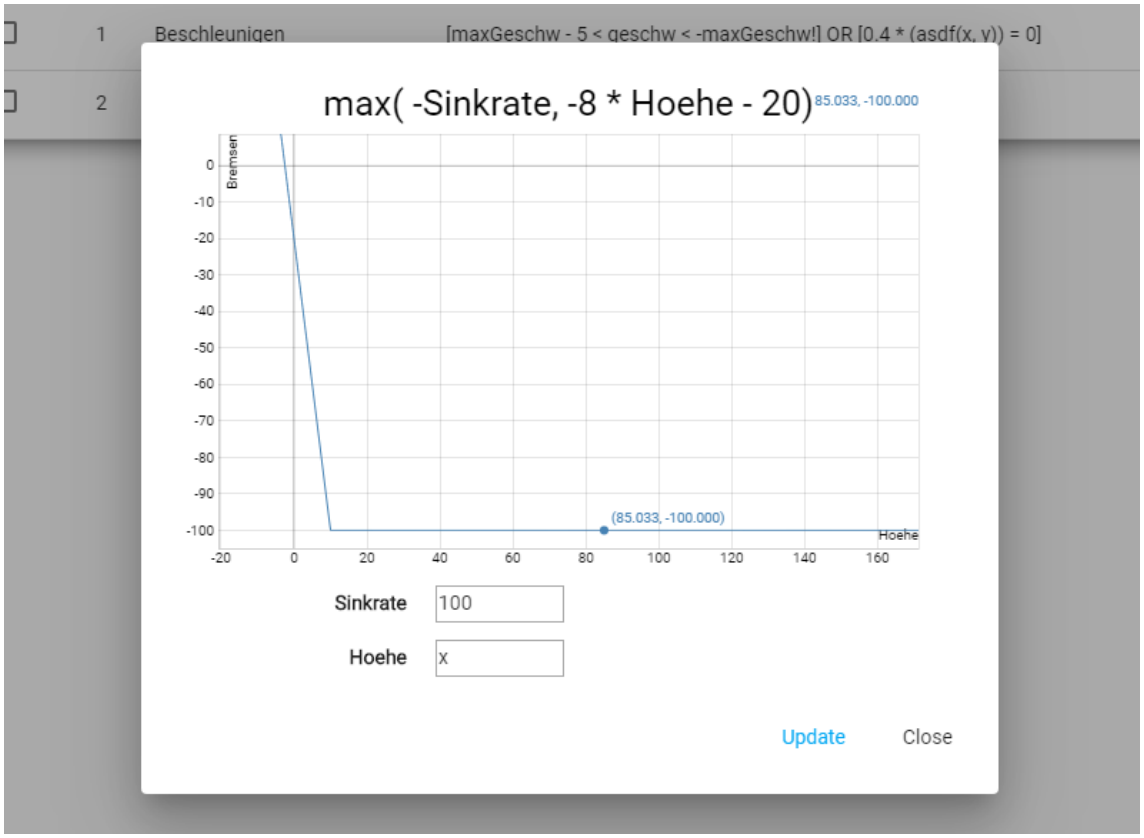


Abbildung 5.3: Ein von XSTAMPP Version 4.1 generierter Funktionsgraph

(2) Ist die Syntax der Regel korrekt, und die Regel enthält keine eckigen Klammern, wird die Beschriftung der y-Achse auf den Namen des Regelsignals festgelegt und direkt zu Phase 3 übergegangen. Hat die Regel eckige Klammern, wird der Inhalt des ersten Klammerpaars extrahiert und an ihren Relationen (=, <, etc.) gegliedert. In diesem Fall wird für die y-Achse eine Variable verwendet, die in ihrem Glied allein steht (z. B. wird bei $wunschTemp - 3 < temperatur < wunschTemp + 3$ die y-Achse auf `temperatur` gesetzt). Alle anderen Glieder werden einzeln in der nächsten Phase bearbeitet.

(3) In einem dritten Schritt werden alle verbliebenen Variablen bestimmt. Hierbei werden Funktionsnamen und Konstanten (`pi`, `e`, etc.) nicht mitgezählt. Jede Variable besitzt ein Eingabefeld, über welches der Nutzer Werte zuweisen kann. Dabei signalisiert das Zuweisen von ”x”, dass sich der Wert der Variable entlang der x-Achse ändert.

(4) Zuletzt werden in den Regeln alle Variablen mit ihrem zugewiesenen Wert ersetzt und der Graph gezeichnet. Ein nachträgliches Bearbeiten der Werte ist möglich.

Die oben beschriebene Implementierung genügt einer Vielzahl vorstellbarer Anwendungsfälle. Es gibt einige Szenarien, in denen sich Defizite offenbaren.

- Während der Syntaxanalyse wird nicht überprüft, ob die verwendeten Funktionen von *Function Plot* erkannt werden. Eine syntaktisch richtige Regel, die eine unbekannte Funktion enthält, löst nicht die Fehlermeldung von XSTAMPP 4.1, sondern die der Programmbibliothek aus.

- Der Graph einer Regel, die mehr als ein Paar eckige Klammern enthält, repräsentiert immer den Inhalt des am weitesten links stehenden Paars. Um andere Inhalte anzuzeigen, muss die Formel entsprechend umgestellt werden.

5.2.2 Benutzeroberfläche

Der Aufruf des Graphen geschieht über eine Schaltfläche ganz rechts in der Tabelle. Diese ist sowohl bei ein- als auch ausgeklapptem Eingabefenster sichtbar. Die Anzeige erfolgt in einem eigenen Fenster, das sich in der Mitte der Benutzeroberfläche von XSTAMPP öffnet und das Eingabefenster nicht verdeckt. Die Wertebereiche der Achsen lassen sich durch Zoomen und Verschieben des Graphen beliebig anpassen. Zudem wird die aktuelle x-Koordinate der Maus sowie die daraus berechnete y-Koordinate eingeblendet. Wurde der Graph aus einer Gleichung oder Ungleichung mit mehreren Gliedern erstellt, wird jedes davon mit einer eigenen Kurve dargestellt. Dabei werden zur Verbesserung der Übersichtlichkeit unterschiedliche Farben verwendet. Das Fenster kann entweder durch Betätigung einer Schaltfläche am unteren Rand, oder durch einen Klick auf den ausgegrauten Bereich geschlossen werden. Unter dem Graphen befindet sich eine Liste der in der Regel vorkommenden Variablen, sowie Eingabefelder, über die die Zuweisung von Werten möglich ist. Standardmäßig hat die erste Variable den Wert ”x”, während alle restlichen Variablen auf 0 gesetzt werden. Der Nutzer kann diese Werte manipulieren und den Graphen durch Betätigung einer Schaltfläche neu zeichnen. Abb. 5.3 demonstriert das Fenster anhand eines Beispielgraphen.

Es gibt Fälle, in denen die Konstruktion eines Graphen teilweise oder vollständig unmöglich ist. Hierfür gibt es mehrere Ursachen. Schlägt die syntaktische Auswertung der Regel fehl, wird der Nutzer über diesen Tatbestand in einem kleineren Fenster informiert. Dabei werden auch Informationen zur Fehlerart und -stelle eingeblendet. Auch der Versuch, eine nicht vereinfachte Gleichung oder Ungleichung (z. B. $2y = 4x$) darzustellen, führt zu einer spezifischen Fehlermeldung. Alle weiteren Sonderfälle lösen die interne Fehlerbehandlung der Programmbibliothek des Graphen aus.

5.3 Konvertertabelle

In STPA werden Systeme in unterschiedliche Komponenten unterteilt, von denen jede die Ursache eines unsicheren Regelsignals sein kann, welches letztlich zu einem Verlust führt. Um eine möglichst große Zahl von Verlustszenarien bestimmen zu können, muss jede dieser Komponenten risikoanalytisch untersucht werden. Der Schwerpunkt bisher lag auf der Entwicklung und Implementierung eines Modellierungskonzepts für Regelalgorithmen. Eine weitere in *STPA Handbook* beschriebene Komponente sind die Aktoren. Insbesondere wird hier der Fall beschrieben, dass ein Aktor das Signal des Reglers korrekt erhält, aber falsch umsetzt [LT18, S. 49]. Für die Bearbeitung dieses Falls kann eine Modellierung der Funktionsweise des Aktors hilfreich sein.

Betrachtet man ein System als Regelkreis, bilden Aktoren das Verbindungsstück zwischen Regler und geregelter Prozess. Dabei ist die vom Regler kommende Eingabe ein Regelsignal, d. i. eine Information, welche sich als eine Menge von Werten darstellen lässt. Die Ausgabe wiederum ist eine Manipulation der physischen Welt. Auch diese lässt sich als eine der Natur des Aktors entsprechende

physikalische Größe beschreiben. Die Aktoren selbst fungieren damit als eine Art Konverter zwischen Ein- und Ausgabewerten. Damit weisen sie einige Parallelen zu Regelalgorithmen auf, die in gleicher Weise eingehende Prozessvariablen in ausgehende Regelsignale umwandeln. Die Existenz dieser Parallelen lässt die Vermutung zu, dass das für Regelalgorithmen entwickelte Konzept auch für Aktoren eine angemessene Modellierungsweise ist.

Als ein weiterer Beitrag dieser Arbeit wurde deshalb ein Modul für XSTAMPP entwickelt, in dem für jeden Aktor eine separate Konvertertabelle verwaltet werden kann. Das Modul lässt sich über eine Schaltfläche erreichen, die direkt unter der für die Regelalgorithmen zuständigen Schaltfläche positioniert ist. Hier lassen sich auf die selbe Weise Konvertierungen erstellen, einsehen, suchen, bearbeiten und löschen. Auch die Darstellung fertiger Konvertierungen in einem Graphen ist möglich. Unterschiede zu der Regeltabelle existieren zweierlei: Erstens wurde die Autovervollständigung so angepasst, dass anstelle der Prozessvariablen die von Reglern ausgehenden Signale angezeigt werden. Zweitens werden die Konvertierungen nicht nach Regler, sondern nach Aktor gruppiert.

5.4 Ausbaupotential

XSTAMPP wurde über die vergangenen Jahre kontinuierlich weiterentwickelt. Es ist vorherzusehen, dass die Entwicklung auch nach Abschluss von Version 4.1 fortgesetzt wird. Unter anderem ist die Implementierung der bereits in Version 3 enthaltenen temporalen logischen Formeln zur Ableitung von Softwareverifizierungstests vorstellbar. Um auch für zukünftige Entwickler eine reibungslose Weiterarbeit zu gewährleisten, wurden bei der Entwicklung des Regelalgorithmenmoduls mehrere Konzepte eingearbeitet, die sich in Bezug auf die Erweiterbarkeit als positiver Einfluss bewährt haben. Eines dieser Konzepte ist die Verwendung *loser Kopplung* und *starker Kohäsion*. Ein Softwaremodul ist lose gekoppelt, wenn es weitgehend eigenständig arbeiten kann, und nur wenig auf Datenaustausch mit anderen Modulen angewiesen ist. Ein Softwaremodul besitzt eine starke Kohäsion, wenn sein Zuständigkeitsbereich eine logisch abgeschlossene Einheit darstellt, d. h. wenn alle übernommenen Aufgaben eng miteinander verwandt sind. Es ist erwiesen, dass Programme, in denen diese Konzepte umgesetzt sind, einen höheren Grad an Wartbarkeit und Erweiterbarkeit besitzen [HM95][GS08]. Weiterhin wurde bei der Implementierung der Regelalgorithmen darauf geachtet, sowohl die Einbettung des Moduls im System, als auch die interne Struktur an die bestehenden Module anzupassen. Dadurch bleibt der Code im Gesamtbild konsistent, was Wartung und Ausbau des Programms zusätzlich begünstigt. Ein dritter Faktor sind die verschiedenen Frameworks, in die XSTAMPP eingebunden ist. Sowohl Spring als auch Angular wurden speziell dafür konzipiert, die nachträgliche Ergänzung von Funktionen zu erleichtern [Spr].

5.5 Dokumentation

Zur Gewährleistung einer problemfreien Weiterentwicklung des Projekts sind neben einer übersichtlichen Codestruktur auch natürlichsprachige Informationen zu diesem wichtig. Diese werden in Form von Dokumentationen bereitgestellt, von denen in XSTAMPP mehrere Arten existieren, die sich in ihren Schwerpunkten unterscheiden. Im Folgenden wird ein kurzer Überblick über die bestehenden Dokumentationstypen, sowie die im Rahmen dieser Arbeit ergänzten Informationen gegeben.

Eine in Java und Javascript gebräuchliche Art der Dokumentation sind Javadoc- bzw. JSDoc-Kommentare. Diese werden im Code über wichtigen Methoden platziert und enthalten diverse Informationen über dessen Funktion, die durch spezielle Tags kategorisiert sind. Solcherlei Kommentare können anschließend maschinell verarbeitet und in einem gut lesbaren Format wie etwa HTML strukturiert zusammengefasst werden. Auch in XSTAMPP wird von Javadoc- und JSDoc-Kommentaren Gebrauch gemacht. Demgemäß sind auch alle hier entwickelten relevanten Methoden mit solchen Kommentaren versehen.

Parallel dazu finden auch normale Codekommentare Verwendung. Diese befinden sich zum Großteil im Inneren von Methoden und erklären den genauen Ablauf der dort stattfindenden Berechnung. Bei der Implementierung der Regelalgorithmen wiesen insbesondere die Methoden, die für das vor der Erstellung des Funktionsgraphen nötige Parsing zuständig sind, einen Grad an Komplexität auf, der die Ergänzung von Codekommentaren unabdinglich macht. Darüber hinaus wurden alle die Verwaltung der Regeltabelle betreffenden Funktionen, sowie Codeabschnitte im Back-End kommentiert.

Neben der Dokumentation im Quellcode besitzt XSTAMPP außerdem einige Markdown-Dateien, die die verwendeten Konzepte und Architekturen ähnlich einem Handbuch auf eine abstraktere Weise beschreiben. Es sind u. a. Informationen über die Funktionsweise der Datensatzsperren, die verschiedenen Nutzerprivilegien, und andere wiederverwendbare Komponenten vorhanden. In diesem Bereich wurden mehrere Architekturdiagramme und die dazugehörigen Erklärungen angepasst. Zusätzlich wurden Einträge über den Graphengenerator sowie dessen Anbindung an die externen Bibliotheken verfasst und die Implementierung der Autovervollständigungsfunktion für die Prozessvariablen beschrieben.

Das Repository von XSTAMPP 4.1 wird mit dem Versionsverwaltungswerkzeug *GitLab* unterhalten. Dort befindet sich auch eine speziell für XSTAMPP erstellte Wiki. Anders als die bisher besprochenen Dokumentationsteile sind hier keine Informationen zur Softwarearchitektur selbst, sondern nur Angaben zur Steuerung der aktuellen Testserver und der Datenbank enthalten. Die hier geleisteten Beiträge haben derlei Strukturen unangetastet gelassen, weshalb von einer Erweiterung dieser Dokumentation abgesehen wurde.

Zu guter Letzt ist auch diese Arbeit als Teil der für die Implementierung des entwickelten Konzepts entstandenen Dokumentation anzusehen. Dabei stehen hier weniger die Implementierungsdetails und technischen Aspekte der Umsetzung, sondern mehr die Planung und die der gewählten Methoden zugrunde liegenden Ideen im Vordergrund. Außerdem werden Nachteile beschrieben, die als Ansatzpunkte für Erweiterungen des Konzepts dienen können.

5.6 Zusammenfassung

In diesem Kapitel wurde die Implementierung des in Kapitel 4 entworfenen Modellierungskonzepts in XSTAMPP beschrieben. Das implementierte Modul erlaubt die Verwaltung von Regeln in einer Tabelle, die an die vorhandene Datenbank angebunden ist und versehentliches Überschreiben simultan durchgeführter Schreibzugriffe mit Datensatzsperren vorbeugt. In den Regeln lassen sich die Namen der Prozessvariablen automatisch vervollständigen. Außerdem können fertige Regeln als Graph dargestellt werden. Aufgrund der Parallelen in den Funktionsweisen von Regelalgorithmen und Aktoren wurde das selbe Konzept verwendet, um eine zweite Tabelle für die Definition von

Konvertierungen zu integrieren. Bei der Implementierung wurden verschiedene Prinzipien befolgt, die ein Warten und Erweitern der Module erleichtert. Für den selben Zweck wurden darüber hinaus mehrere Teile der Dokumentation erweitert.

Im nächsten Kapitel ist eine vollständige mithilfe von XSTAMPP durchgeführte STPA-Analyse dokumentiert.

6 Eine STPA-Analyse mit XSTAMPP 4.1

Bei dem Entwurf des Konzepts für Regelalgorithmen wurden einige Gütemerkmale in Erwägung gezogen. Die Beurteilung basierte jedoch auf rein theoretischen Überlegungen. Zwar fand auch eine Demonstration am Beispiel eines Notlandungsmoduls einer Drohne statt, diese wurde jedoch in vielen Teilen unvollständig gelassen. Um die Potenziale und Defizite der neuen Modellierungsmethode in einem größeren Spektrum ergründen zu können, ist die Durchführung einer umfassenderen STPA-Analyse erforderlich. Beispielsweise muss erforscht werden, inwieweit die durch Betrachtung des Regelalgorithmenmodells identifizierbaren Verlustszenarien schon im Voraus identifiziert werden können. Zu diesem Zweck wird im Folgenden eine beispielhafte STPA-Analyse vorgeführt. Im Rahmen dieser wird erstmals die in XSTAMPP 4.0 entwickelte Weboberfläche und die in Version 4.1 eingeführte Regelalgorithmentabelle verwendet. Obgleich die Analyse die Eignung des neuen Konzepts zeigen kann, ist ein direkter Vergleich zu anderen Modellierungskonzepten nicht möglich. Hierzu müsste die selbe Untersuchung mit unterschiedlichen Programmen von unterschiedlichen Teams durchgeführt werden. Dies ist im Rahmen dieser Arbeit leider nicht möglich. Des Weiteren wird die Identifikation der Verlustszenarien aufgrund der hohen Detailgenauigkeit von STPA auf eine Auswahl unsicherer Regelsignale beschränkt.

Bei der Wahl des Analyseszenarios wurden verschiedene Anforderungen in Erwägung gezogen. Ein wichtiges Ziel bei der Ausarbeitung des Modellierungskonzepts war die Anwendbarkeit bei menschlichen Reglern. Das Szenario wurde deshalb so gewählt, dass mindestens ein menschlicher Regler existiert. Von den in Abschnitt 3.3 besprochenen Erweiterungen zu STPA wird jedoch keinen Gebrauch gemacht, da diese noch nicht in XSTAMPP implementiert sind. Um ein ergiebiges Bild der Regelalgorithmen zu bekommen, ist zudem wichtig, ein Szenario aufzustellen, in dem ein umfangreicher Algorithmus vorhersehbar ist.

Der folgende Bericht ist in fünf Abschnitte unterteilt, von denen die ersten vier den vier Phasen einer STPA-Analyse entsprechen. Im fünften Teil befindet sich eine auf den Ergebnissen der Analyse basierende Beurteilung des Modellierungskonzepts.

6.1 Grundlegende Angaben zum analysierten System

Die folgende Analyse betrachtet ein Anlegemanöver einer Binnenfähre an einem Kai, Dalben, Hafenmolden oder Pier. Dabei wird von einem Docking aus eigener Kraft ohne Einsatz von Schlepper-, Lotsen oder Festmacherhilfen ausgegangen. Während des Anlegemanövers hat der Kapitän die Aufgabe, durch Bedienung mehrerer Propeller und Ruder die Fähre längs der Anlegestelle auszurichten und dieser langsam anzunähern. Gleichzeitig muss er Wasserströmung und Wind ausgleichen. Für diese Analyse wird das Manöver als abgeschlossen betrachtet, sobald das Schiff sicher am Reibholz oder Fender anliegt. Die Vertäuung mit Festmacherleinen wird vernachlässigt.

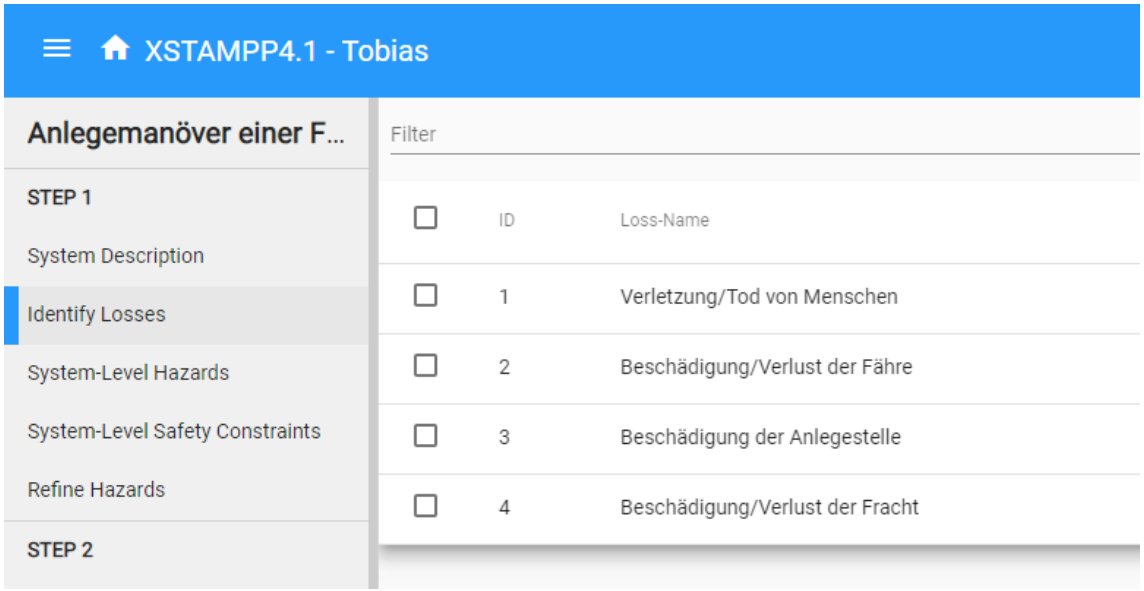


Abbildung 6.1: Identifikation der potentiellen Verluste

Für das System wurden vier Arten von Verlusten identifiziert und gemäß Abb. 6.1 in XSTAMPP erfasst. In dieser Analyse liegt ein Verlust vor, wenn ein oder mehrere Menschen körperlich zu Schaden kommen oder ihr Leben verlieren, die Fähre wesentliche Schäden davonträgt, kentert oder sinkt, die Anlegestelle beschädigt wird, oder die Fracht verloren geht oder unbrauchbar wird. Unter den Verlust von Fracht fällt auch das Austreten von Öl oder giftigen Substanzen in das Hafenbecken. Ein Abbruch des Anlegemanövers aus diversen Gründen kann als Fehlschlag der Mission angesehen werden, ist hier aber nicht Gegenstand der Betrachtung, da in diesem Fall kein permanenter Schaden entsteht.

Zu einem Verlust kann es kommen, wenn das System in einen Zustand gelangt, der eine Gefährdung darstellt. Für die vorliegenden Verluste wurden insgesamt vier Gefährdungen festgestellt, die im Folgenden beschrieben sind.

[H-1] Krängung der Fähre wird zu stark: Wird die Krängung der Fähre zu stark, kann Ladung oder Mannschaft über Bord gehen. Außerdem besteht die Gefahr des Kenterns. Dies kann zu den Verlusten [L-1], [L-2] und [L-4] führen.

[H-2] Fähre nimmt zu viel Geschwindigkeit auf: Nimmt die Fähre zu viel Geschwindigkeit auf, kann das Schiff unter Umständen nicht rechtzeitig gebremst werden, wodurch es zu einer Kollision mit der Anlegestelle, einem Wasserfahrzeug oder einem anderen im Wasser befindlichen Objekt kommt. Dies kann zu den Verlusten [L-1], [L-2], [L-3] und [L-4] führen.

[H-3] Kapitän verliert vollständig oder teilweise die Kontrolle über die Fähre: Ein Kontrollverlust kann sowohl durch einen Ausfall eines Bauteils des Schiffs, als auch durch eine plötzliche Arbeitsunfähigkeit des Kapitäns verursacht werden. In beiden Fällen ist ein Auftreten der Verluste [L-1], [L-2], [L-3] und [L-4] möglich.

[H-4] Sicherheitsabstand zu anderen Wasserfahrzeugen wird unterschritten: Kommt die Fähre einem anderen Wasserfahrzeug zu nahe, besteht die Gefahr eines Zusammenstoßes. Dies kann zu den Verlusten [L-1], [L-2] und [L-4] führen.

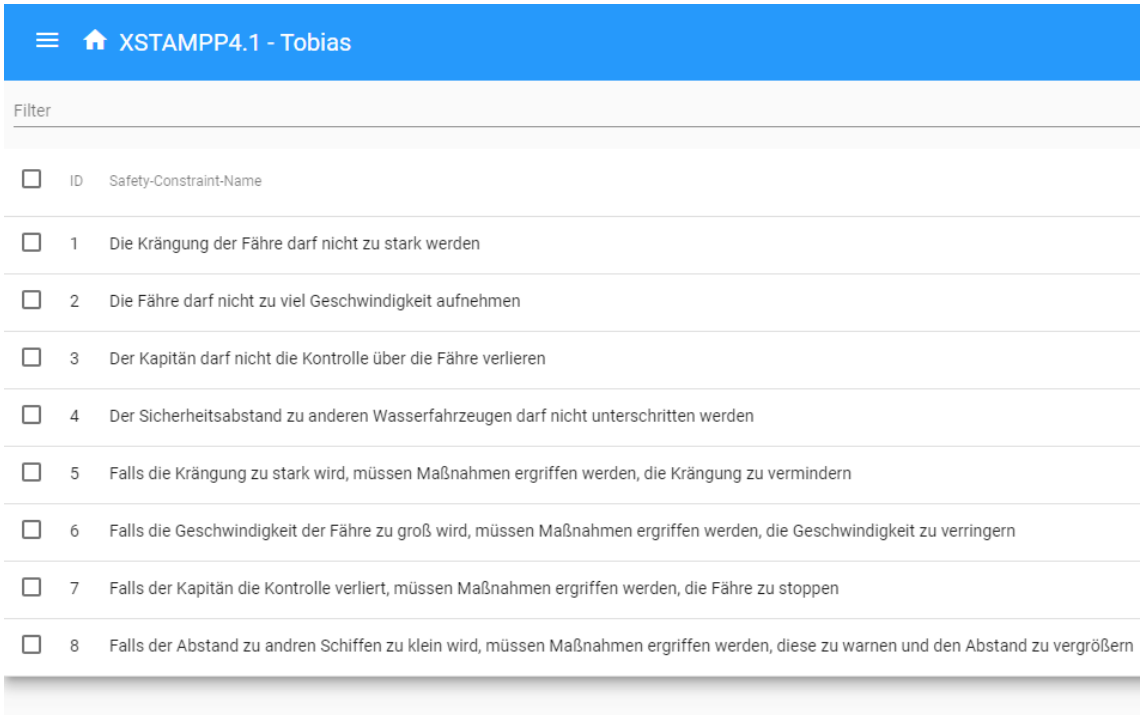


Abbildung 6.2: Die aus den Gefährdungen abgeleiteten Randbedingungen

Das Aufgrundlaufen des Schiffs wurde nicht in die Liste der Gefährdungen aufgenommen, da dieses bedingt durch die zumeist sehr regelmäßige topographische Gestalt des Gewässergrunds innerhalb des Manövrierraums äußerst unwahrscheinlich ist. Des Weiteren besteht selbst im Falle eines Grundkontakts ein eher geringes Verlustrisiko. Auch Brände und vergleichbare Gefährdungen wurden nicht aufgenommen, da diese nicht speziell das Anlegemanöver, sondern den Betrieb der Fähre im Allgemeinen betreffen, und daher im Rahmen einer separaten Analyse zu behandeln sind.

Aus den gesammelten Gefährdungen lassen sich durch Invertierung der Aussagen die dazugehörigen Randbedingungen ableiten. Zusätzlich werden Randbedingungen definiert, die dem System vorschreiben, wie es im Falle einer Gefährdung Verluste reduzieren muss. Diese Art von Randbedingungen schreiben grundsätzlich keine genaue Lösung oder Implementation vor, da eine verfrühte Fixierung das Risiko birgt, potentiell bessere Lösungen zu übersehen [LT18, S. 20-21]. Insgesamt wurden acht Randbedingungen bestimmt, die in Abb. 6.2 zu sehen sind.

6.2 Modellierung des Regelkreises

In der zweiten Analysephase wird zunächst ein Modell des Systems konstruiert, welches die Einflussnahme des Reglers auf den Prozess über Aktoren sowie die Informationserfassung über Sensoren spezifiziert und in einem Regelkreis veranschaulicht (Abb. 6.3). Gegenstand dieser Analyse ist das Anlegemanöver einer Fähre, weshalb für den geregelten Prozess die *Bewegung der Fähre* gewählt wurde. Die Steuerung des Schiffs wird von einem *Kapitän* übernommen, der im Regelkreis die Rolle des Reglers einnimmt. Während des Anlegemanövers befindet sich der Kapitän auf der

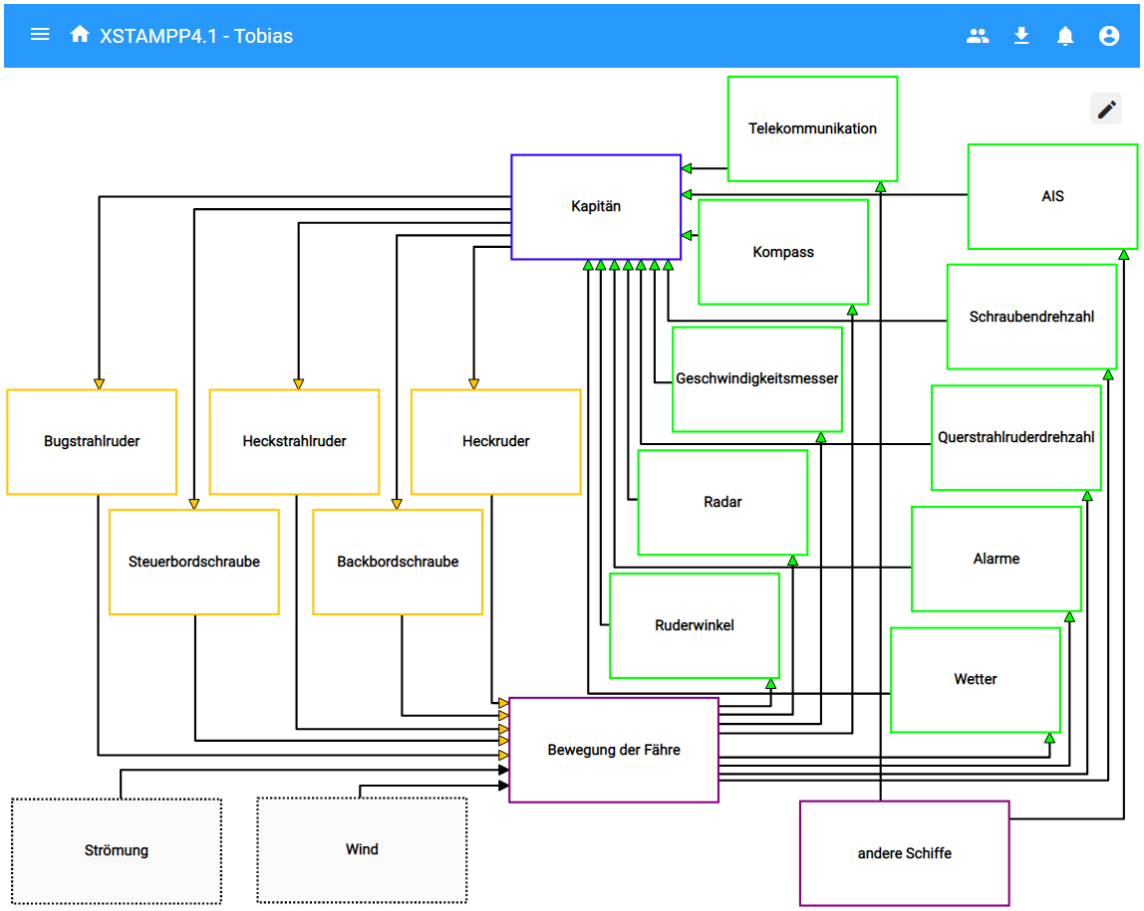


Abbildung 6.3: In XSTAMPP modelliertes Regelkreisdiagramm der Fähre

Kommandobrücke. von wo er Steuerungsbefehle an verschiedene am Schiffsrumpf angebrachte Propeller und bewegliche Ruder sendet. Hierbei gibt es schiffsabhängige Unterschiede in Anzahl und Typ der verbauten Teile. Diese Analyse geht von dem allgemeinen Fall aus, bei dem zwei Schiffsschrauben auf gegenüberliegenden Seiten des Rumpfs angebracht sind, die sich in entgegengesetzte Richtungen drehen, um den Radeffekt auszugleichen. Im Diagramm sind diese als *Steuerbord-* und *Backbordschraube* bezeichnet. Weiterhin wird die Existenz eines (oder mehrerer uniformer) *Heckruder(s)* angenommen. Zum Zwecke einer besseren Manövrierfähigkeit besitzen viele Fähren außerdem Querstrahlruder, welche entgegen des Namens keine Ruder sondern seitlich angebrachte Propeller sind. Diese sind als *Bug-* und *Heckstrahlruder* bezeichnet. Einige Fähren haben anstelle der Querstrahlruder rotierbare Schiffsschrauben [Mew01]. Die Analyseergebnisse lassen sich jedoch weitestgehend auf diese übertragen. Neben den direkt für die Bewegung des Schiffs zuständigen Propellern und Rudern besitzen Fähren eine Vielzahl von Aktoren, die für andere Zwecke gebraucht werden. Hierzu gehören z. B. der über größere Distanzen einsetzbare Autopilot und die Systeme zur Aufzeichnung von Position, Geschwindigkeit und anderen Daten. Für das Anlegemanöver haben solcherlei Aktoren keine Bedeutung, weshalb sie in der Analyse nicht miteinbezogen wurden. Die *Bewegung der Fähre* wird neben den Aktoren zusätzlich von einigen Umwelteinwirkungen beeinflusst. Insbesondere *Strömung* und *Wind* sind für die Untersuchung wichtige Faktoren.

Um durchgängig alle für das Manöver benötigten Informationen zu erhalten, macht der Kapitän Gebrauch von diversen Sensoren. Die Ausrichtung des Schiffs liest er an einem *Kompass*, die Geschwindigkeit und den Schub an *Geschwindigkeits-* und *Drehzahlmessern* ab. Zusätzlich stehen Angaben zum aktuellen *Winkel des Heckruders* zur Verfügung. Die Position anderer Schiffe und der Verlauf des Hafenbeckens werden auf einem *Radarmonitor* angezeigt. Ergänzend dazu existieren unterschiedliche *Alarme*, die den Kapitän auf ein Unterschreiten des Sicherheitsabstands, Systemausfälle und weitere Notsituationen aufmerksam macht. Die Windstärke wird mit *Wettersensoren* erfasst. Neben Informationen zum eigenen Schiff erhält der Kapitän außerdem Angaben zu anderen Schiffen via *Telekommunikation* und das *Automatic Identification System (AIS)*. Zu guter Letzt ist auch die direkte audiovisuelle Wahrnehmung der Hafenumgebung ein wichtiger Teil des benötigten Feedbacks. Ebenso wie bei den Aktoren wurden auch mehrere Sensoren nicht in die Analyse mit aufgenommen. Dies beinhaltet Global Positioning System (GPS), Navigationssystem und Sextant, die im Normalfall eher bei der Routenplanung im größeren Maßstab als bei Anlegemanövern gebraucht werden. Im Zusammenhang mit der Vernachlässigung der Gefahr eines Aufgrundlaufens wurde zudem das Echolot nicht aufgenommen.

Nach der Modellierung der Systemkomponenten und deren Verbindungen werden im nächsten Schritt allen existierenden Reglern Verantwortungsbereiche zugeteilt. Diese sind im Allgemeinen Verfeinerungen der Randbedingungen und spezifizieren, was jeder Regler im Einzelnen tun muss, damit aus der kollektiven Zusammenarbeit die Erfüllung der Randbedingungen gewährleistet ist. Für das Anlegemanöver einer Fähre ist der einzige direkt in Verbindung stehende Regler der Kapitän. Diesem wurden insgesamt elf Verantwortlichkeiten zugeteilt, welche in Abb. 6.4 zu sehen sind.

Filter

Controller	ID	Responsibility-Name
Kapitän	1	Geschwindigkeit erhöhen, wenn Krängung zu stark wird
	2	Geschwindigkeit vermindern, wenn Fähre zu schnell wird
	3	Geschwindigkeit anpassen, um einen umweltbedingten Drift der Fähre auszugleichen
	4	Geschwindigkeit des Bugstrahlruders anpassen, um einen umweltbedingten Drift der Fähre auszugleichen
	5	Geschwindigkeit des Heckstrahlruders anpassen, um einen umweltbedingten Drift der Fähre auszugleichen
	6	Geschwindigkeit des Bugstrahlruders anpassen, um das Schiff korrekt auszurichten
	7	Geschwindigkeit des Heckstrahlruders anpassen, um das Schiff korrekt auszurichten
	8	Ruderwinkel anpassen, um das Schiff korrekt auszurichten
	9	Ruderwinkel vermindern, wenn Krängung zu stark wird
	10	hohe Geschwindigkeiten über einen längeren Zeitraum vermeiden, um einen Kontrollverlust vorzubeugen
	11	Warnung über Funk geben, wenn ein Kontrollverlust stattfindet

Abbildung 6.4: Die Verantwortlichkeiten des Kapitäns

In einem letzten Schritt wird die zur Wahrnehmung der oben definierten Verantwortungen erforderlichen Kommunikation zwischen den einzelnen Systemkomponenten ermittelt. Dazu wird Gebrauch von den bereits im Regelkreisdiagramm (Abb. 6.3) vorhandenen Verbindungen gemacht. In dieser Analyse wurden auf Seiten der Sensoren 24 *Feedbacksignale*, auf Seiten der Aktoren 10 *Regelsignale* definiert. Diese sind in Tabelle 6.1 und 6.2 aufgelistet. Dabei ist neben Quelle und Ziel des Signals auch ein Wertebereich angegeben, welcher bei kontinuierlichen Signalen den minimalen und maximalen Wert, und bei diskreten Signalen die möglichen Zustände angibt. Beispielsweise ist der Anstellwinkel des Heckruders üblicherweise auf einen Wert zwischen -35 und 35 Grad beschränkt, da es bei größeren Winkeln zu einem Strömungsabriss kommt, was zu einer signifikanten Abbremsung des Schiffs führt [LH17]. Einige Werte sind willkürlich gewählt, aber entsprechen ungefähr der Spezifikation einer Binnenfähre.

Feedbacksignal	**Quelle**	**Ziel**	**Wertebereich**
Magnetfeld	Bewegung der Fähre	Kompass	0° bis 360°
Wasserstrom	Bewegung der Fähre	Geschwindigkeitsmesser	-∞ bis ∞ m/s
Wellenreflektion	Bewegung der Fähre	Radar	0% bis 100%
Auslenkung	Bewegung der Fähre	Ruderwinkel	-35° bis 35°
Schraubenrotation	Bewegung der Fähre	Schraubendrehzahl	-250 bis 250
Querstrahlruderrotation	Bewegung der Fähre	Querstrahlruderdrehzahl	-700 bis 700
Abstand	Bewegung der Fähre	Alarme	0 m bis ∞ m
Ausgangsleistung	Bewegung der Fähre	Alarme	0 bis 210 kW
Luftstrom	Bewegung der Fähre	Wetter	0 bis ∞ m/s
Luftstromwinkel	Bewegung der Fähre	Wetter	0° bis 360°
Radiowellen	andere Schiffe	Telekommunikation	–
Datenübertragung	andere Schiffe	AIS	[-100, 100]²
Ausrichtung	Kompass	Kapitän	0° bis 360°
Geschwindigkeit	Geschwindigkeitsmesser	Kapitän	-20 bis 20 kn
Hinderniskoordinaten	Radar	Kapitän	[-100, 100]²
Winkel	Ruderwinkel	Kapitän	-35° bis 35°
Schraubendrehzahl	Schraubendrehzahl	Kapitän	-250 bis 250
Querstrahlruderdrehzahl	Querstrahlruderdrehzahl	Kapitän	-700 bis 700
Annäherungsalarm	Alarme	Kapitän	wahr/falsch
Motorproblem	Alarme	Kapitän	wahr/falsch
Windgeschwindigkeit	Wetter	Kapitän	0 bis ∞ m/s
Windrichtung	Wetter	Kapitän	0° bis 360°
Sprachnachrichten	Telekommunikation	Kapitän	–
Schiffskoordinaten	AIS	Kapitän	[-100, 100]²

Tabelle 6.1: Feedbacksignale für ein Anlegemanöver einer Fähre

Regelsignal	Quelle	Ziel	Wertebereich
Bugstrahlschub	Kapitän	Bugstrahlruder	-100% bis 100%
Heckstrahlschub	Kapitän	Heckstrahlruder	-100% bis 100%
Ruderausschlag	Kapitän	Heckruder	-100% bis 100%
Steuerbordschub	Kapitän	Steuerbordschraube	-100% bis 100%
Backbordschub	Kapitän	Backbordschraube	-100% bis 100%
Bugstrahlruderdrehzahl	Bugstrahlruder	Bewegung der Fähre	-700 bis 700
Heckstrahlruderdrehzahl	Heckstrahlruder	Bewegung der Fähre	-700 bis 700
Anstellwinkel	Heckruder	Bewegung der Fähre	-35° bis 35°
Steuerbordschraubendrehzahl	Steuerbordschraube	Bewegung der Fähre	-250 bis 250
Backbordschraubendrehzahl	Backbordschraube	Bewegung der Fähre	-250 bis 250

Tabelle 6.2: Regelsignale für ein Anlegemanöver einer Fähre

6.3 Bestimmung der unsicheren Regelsignale

Die dritte Phase einer STPA-Analyse umfasst die Identifikation unsicherer Regelsignale und die daraus folgende Ableitung der Randbedingungen auf Reglerebene. Für das Anlegemanöver einer Fähre wurden insgesamt 82 unsichere Regelsignale identifiziert, von denen einige in den Abbildungen 6.5, 6.6 und 6.7 aufgeführt sind. Die unsicheren Regelsignale des Heckstrahlschub- und des Backbordschub-Signals sind nicht dargestellt, da sie vollständig analog zu dem Bugstrahlschub- und Steuerbordschub-Signal sind. Gleichermaßen wurden die von den Aktoren ausgehenden Regelsignale weggelassen, da sie mit dem entsprechenden eingehenden Signal übereinstimmen. Die während des Anlegemanövers zugelassene Geschwindigkeit wird auf 2 Knoten beschränkt, da höhere Geschwindigkeiten als unsicher gelten [Cha19]. Der Sicherheitsabstand zu anderen Schiffen beträgt 10 Meter bei normaler Ladung und 50 Meter bei Ladung von Gütern höherer Gefahrklassen [BS77]. Als kritische Krängung sind 10° bei Geradeausfahrt und 12° bei Fahrt im Drehkreis festgelegt [EU98]. Die präzise Einteilung aller relevanten Werte in sichere und unsichere Bereiche ist hilfreich, kann aber bei Bedarf erst zu einem späteren Zeitpunkt erfolgen. In diesem Fall wird die Zahl durch ”TBD” ersetzt.

Während der Identifikationsphase der unsicheren Regelsignale wurden einige reale Unfälle betrachtet, um sicherzustellen, dass eine möglichst große Zahl von Signalen erfasst wird. Am 20. September 2018 kenterte die tansanische Fähre MV Nyerere während eines Anlegemanövers in Bwisya, was zum Tod von mindestens 229 Menschen führte [IFR19]. Die unmittelbare Ursache für das Unglück war die plötzliche Initiierung einer starken Kurve bei vorhandener Krängung. Diese wird durch die unsicheren Regelsignale des Ruderausschlags ”[UCA-2] Senden, wenn die Krängung bereits stark ist (= 10/12°)” und ”[UCA-7] zu langes Senden, wenn die Krängung bereits stark ist (= 10/12°)” repräsentiert. Weitere Ursachen waren eine Beladung des Schiffs weit über dem spezifizierten Limit, fehlende Fachkenntnisse des Kapitäns und Ablenkung durch ein Telefongespräch. Das Überladen der Fähre und das Einsetzen eines untrainierten Steuermanns haben keine entsprechenden unsicheren Regelsignale, da diese nicht das Anlegemanöver selbst, sondern den Betrieb des Schiffs als Ganzes betreffen. Das Telefongespräch ist gleichzusetzen mit einem partiellen Kontrollverlust, der in der Verantwortlichkeit ”[H-3] Kapitän verliert vollständig oder teilweise die Kontrolle über die Fähre” festgehalten ist.

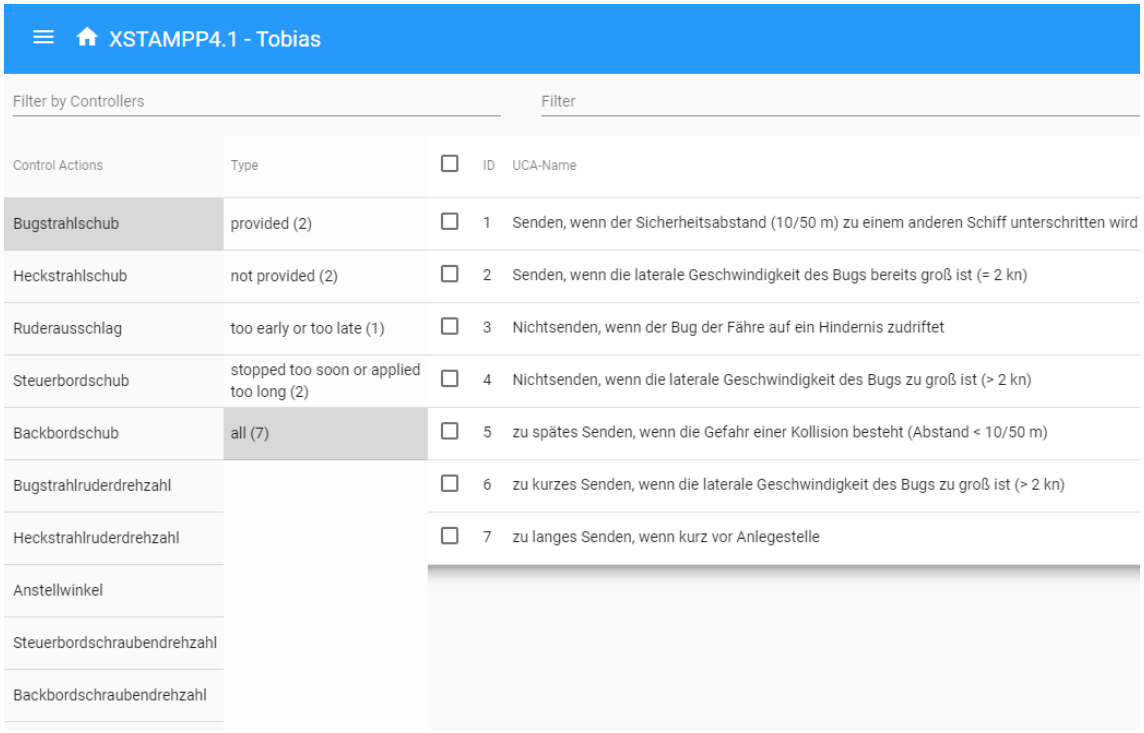


Abbildung 6.5: Unsichere Regelsignale für das Bugstrahlschub-Signal

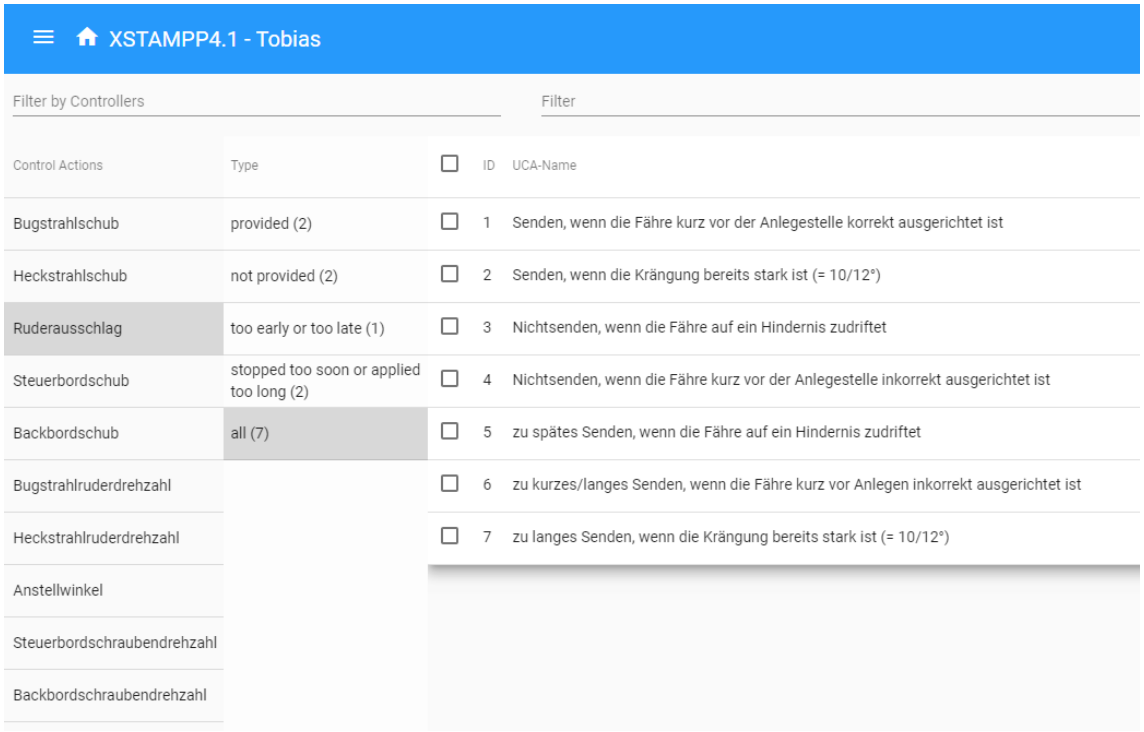


Abbildung 6.6: Unsichere Regelsignale für das Ruderausschlag-Signal

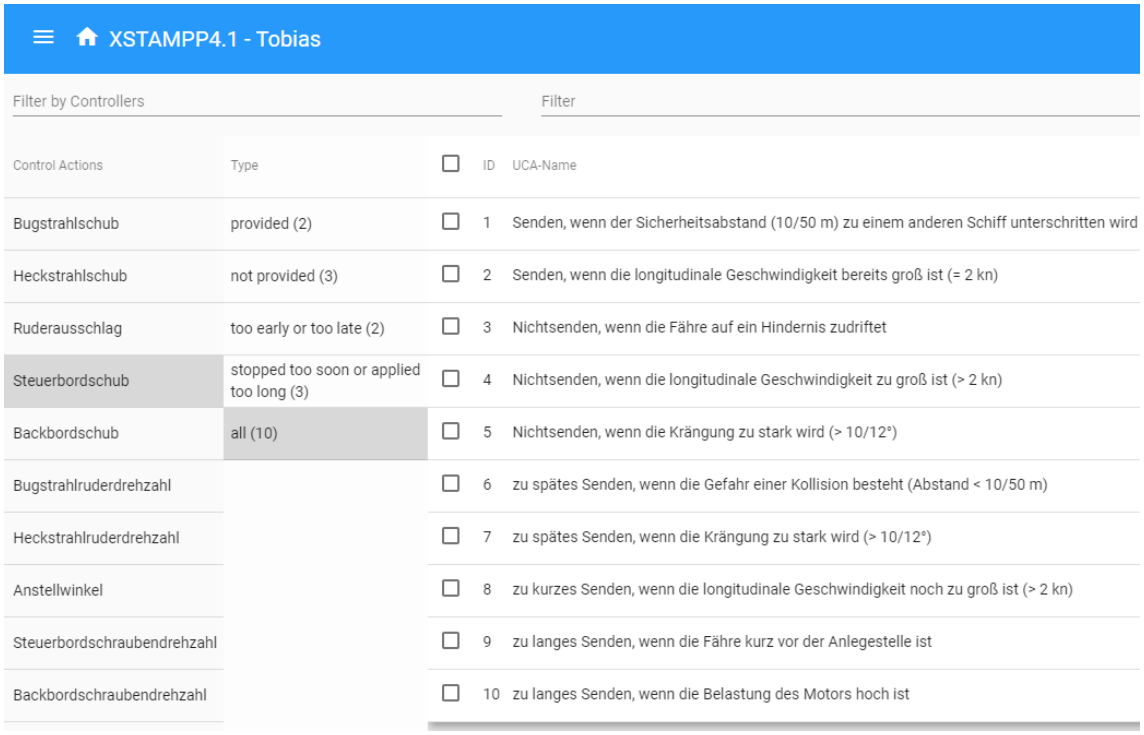


Abbildung 6.7: Unsichere Regelsignale für das Steuerbordschub-Signal

Am 12. Januar 2018 kollidierte die italienische Fähre *Fantastic* mit dem norwegischen Kreuzfahrtschiff *Viking Star* im Hafen von Barcelona [Cou18]. Ursache war ein Motordefekt, infolge dem die Fähre unkontrolliert auf die Anlegestelle des Kreuzfahrtschiffs zutrieb. Dies korreliert mit dem unsicheren Regelsignal des Steuerbord- und Backbordschubs ”[UCA-3] Nichtsenden, wenn die Fähre auf ein Hindernis zudriftet”. Der Kapitän der *Fantastic* hat durch Auswerfen des Ankers versucht, die Geschwindigkeit des Schiffs zu reduzieren, und zusätzlich nahe Schiffe gewarnt. Er hat damit die im Zuge dieser Analyse herausgearbeiteten Randbedingungen ”[SC-7] Falls der Kapitän die Kontrolle verliert, müssen Maßnahmen ergriffen werden, die Fähre zu stoppen” und ”[SC-8] Falls der Abstand zu anderen Schiffen zu klein wird, müssen Maßnahmen ergriffen werden, diese zu warnen und den Abstand zu vergrößern” erfüllt.

Zusätzlich zu den Randbedingungen auf Systemebene werden nach erfolgter Identifikation der unsicheren Regelsignale die Randbedingungen auf Reglerebene abgeleitet. Auch diese entstehen durch eine simple Invertierung. Im Folgenden sind die den 24 oben aufgeführten unsicheren Signalen entsprechenden Randbedingungen aufgelistet. Alle restlichen Randbedingungen ergeben sich analog.

Bugstrahlschub

[C-1] Kapitän darf Bugstrahlschub nicht senden, wenn der Sicherheitsabstand (10/50 m) zu einem anderen Schiff unterschritten wird

[C-2] Kapitän darf Bugstrahlschub nicht senden, wenn die laterale Geschwindigkeit des Bugs bereits groß ist (= 2 kn)

[C-3] Kapitän muss Bugstrahlschub senden, wenn der Bug der Fähre auf ein Hindernis zudriftet

[C-4] Kapitän muss Bugstrahlschub senden, wenn die laterale Geschwindigkeit des Bugs zu groß ist (> 2 kn)

[C-5] Kapitän darf Bugstrahlschub nicht zu spät senden, wenn die Gefahr einer Kollision besteht (Abstand < 10/50 m)

[C-6] Kapitän darf Bugstrahlschub nicht zu kurz senden, wenn die laterale Geschwindigkeit des Bugs zu groß ist (> 2 kn)

[C-7] Kapitän darf Bugstrahlschub nicht zu lang senden, wenn die Fähre kurz vor der Anlegestelle ist

Ruderausschlag

[C-1] Kapitän darf Ruderausschlag nicht senden, wenn die Fähre kurz vor der Anlegestelle korrekt ausgerichtet ist

[C-2] Kapitän darf Ruderausschlag nicht senden, wenn die Krängung bereits stark ist (= 10/12°)

[C-3] Kapitän muss Ruderausschlag senden, wenn die Fähre auf ein Hindernis zudriftet

[C-4] Kapitän muss Ruderausschlag senden, wenn die Fähre kurz vor der Anlegestelle inkorrekt ausgerichtet ist

[C-5] Kapitän darf Ruderausschlag nicht zu spät senden, wenn die Fähre auf ein Hindernis zudriftet

[C-6] Kapitän darf Ruderausschlag nicht zu kurz oder lang senden, wenn die Fähre kurz vor Anlegen inkorrekt ausgerichtet ist

[C-7] Kapitän darf Ruderausschlag nicht zu lang senden, wenn die Krängung bereits stark ist (= 10/12°)

Steuerbordschub

[C-1] Kapitän darf Steuerbordschub nicht senden, wenn der Sicherheitsabstand (10/50 m) zu einem anderen Schiff unterschritten wird

[C-2] Kapitän darf Steuerbordschub nicht senden, wenn die longitudinale Geschwindigkeit bereits groß ist (= 2 kn)

[C-3] Kapitän muss Steuerbordschub senden, wenn die Fähre auf ein Hindernis zudriftet

[C-4] Kapitän muss Steuerbordschub senden, wenn die longitudinale Geschwindigkeit zu groß ist (> 2 kn)

[C-5] Kapitän muss Steuerbordschub senden, wenn die Krängung zu stark wird (> 10/12°)

[C-6] Kapitän darf Steuerbordschub nicht zu spät senden, wenn die Gefahr einer Kollision besteht (Abstand < 10/50 m)

[C-7] Kapitän darf Steuerbordschub nicht zu spät senden, wenn die Krängung zu stark wird (> 10/12°)

[C-8] Kapitän darf Steuerbordschub nicht zu kurz senden, wenn die longitudinale Geschwindigkeit noch zu groß ist (> 2 kn)

[C-9] Kapitän darf Steuerbordschub nicht zu lang senden, wenn die Fähre kurz vor der Anlegestelle ist

[C-10] Kapitän darf Steuerbordschub nicht zu lang senden, wenn die Belastung des Motors hoch ist

6.4 Verlustszenarien

In der letzten Analysephase werden eine möglichst große Zahl Szenarien gesammelt, bei denen es zu einem unsicheren Regelsignal kommen kann. Zu diesem Zweck müssen sämtliche Komponenten des Regelkreises betrachtet werden. Das in dieser Arbeit entwickelte Modellierungskonzept kann möglicherweise eine Steigerung der bei der Betrachtung der Regelalgorithmen und Aktoren gefundenen Szenarien bewirken. Um dies zu prüfen, werden die Verlustszenarien zunächst auf herkömmliche Weise identifiziert. Anschließend folgt die Modellierung des Regelalgorithmus des Kapitäns und der Konvertertabellen der Propeller und Ruder. Zuletzt wird versucht, anhand der entworfenen Modelle weitere Szenarien zu finden. Gelingt dies, so kann das als Beleg für die Effektivität des Modells gewertet werden. Aufgrund der großen Zahl unsicherer Regelsignale wird im Folgenden eine willkürlich gewählte Auswahl betrachtet.

6.4.1 Identifikation ohne Regelalgorithmus

Es wurden die unsicheren Regelsignale Bugstrahl-UCA-7, Ruderausschlag-UCA-2 und Steuerbord-UCA-5 untersucht, wodurch sich insgesamt 24 Verlustszenarien ergeben haben.

Zu langes Senden des Bugstrahlschub-Signals, wenn die Fähre kurz vor der Anlegestelle ist

(1) Der Kapitän sendet ein Schubsignal an das Bugstrahlruder in sicherer Distanz, wird jedoch abgelenkt und beendet das Senden zu spät. Gründe hierfür können eine hohe Arbeitslast (z. B. durch schwierige Wetterbedingungen, gleichzeitige Kommunikation mit anderen Menschen oder Reagieren auf unerwartete Ereignisse wie Fehlalarme), Konzentrationsschwierigkeiten (z. B. durch Müdigkeit oder eine zu niedrige Arbeitslast), oder eine Priorisierung privater Belange sein.

(2) Der Kapitän beendet das Senden des Signals zu spät, weil er von einer zu niedrigen Geschwindigkeit des Schiffs ausgeht. Gründe hierfür können eine falsche Anzeige der Geschwindigkeit (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Geschwindigkeitssensor oder eine ungewöhnliche Wasserströmung genau am Sensor), eine fehlende Anzeige der Geschwindigkeit (z. B. durch einen defekten Display oder einen defekten Geschwindigkeitssensor), oder eine Fehlinterpretation der angezeigten Geschwindigkeit sein.

(3) Der Kapitän beendet das Senden des Signals zu spät, weil er von einer zu hohen Distanz zur Anlegestelle ausgeht. Gründe hierfür können ein zu unauffälliger Annäherungsalarm (z. B. durch einen Lautsprecherdefekt, einen unregelmäßig auslösenden Annäherungssensor oder

eine ungewöhnlich gebaute Anlegestelle), ein ausbleibender Annäherungsalarm (z. B. durch einen Lautsprecherdefekt, einen Annäherungssensordefekt oder eine ungewöhnlich gebaute Anlegestelle) oder eine Fehlinterpretation des Annäherungsalarms sein.

(4) Der Kapitän beendet das Senden des Signals zu spät, weil er von einer zu niedrigen Drehzahl der Schraube ausgeht. Gründe hierfür können eine falsche Anzeige der Drehzahl (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Drehzahlsensor oder eine ungewöhnliche Wasserströmung genau am Sensor), eine fehlende Anzeige der Drehzahl (z. B. durch einen defekten Display oder einen defekten Drehzahlsensor), oder eine Fehlinterpretation der angezeigten Drehzahl sein.

(5) Der Kapitän beendet das Senden des Signals zu spät, weil er von einer falschen Windstärke oder -richtung ausgeht. Gründe hierfür können eine falsche Anzeige der Windstärke oder -richtung (z. B. durch einen Fehler im Display, ungenau funktionierende Wettersensoren oder einer Verwirbelung im Sensorbereich), eine fehlende Anzeige der Windstärke oder -richtung (z. B. durch einen defekten Display oder defekte Wettersensoren), oder eine Fehlinterpretation der angezeigten Windstärke oder -richtung sein.

(6) Der Kapitän beendet das Senden des Signals zu spät, weil er sich bei der Abschätzung des Bremswegs verrechnet hat, obwohl er korrekte Informationen über Schiff und Umwelt besitzt. Grund hierfür kann Unerfahrenheit in Bezug auf das Schiff oder den angesteuerten Hafen sein.

(7) Der Kapitän beendet das Senden des Signals zu spät, weil er in seinem Training unzureichend über das Verhalten des Schiffs oder die Gegebenheiten des angesteuerten Hafens informiert wurde. Gründe hierfür können zwischenmenschliche Missverständnisse, zu kurzes Training, schlechte Methodik oder falsche bzw. fehlende Inhalte im Trainingsprogramm sein.

(8) Der Kapitän beendet das Senden des Signals zu spät, weil er einen oder mehrere das Verhalten des Schiffs beeinflussende Faktoren vergisst, in die Kalkulation des Bremswegs miteinzubeziehen. Grund hierfür können von der Norm abweichende Umstände in Bezug auf Schiff (z. B. starke oder ungleichmäßige Beladung) oder Hafen (z. B. Wassertiefe) sein.

Senden des Ruderausschlag-Signals, wenn die Krängung bereits stark ist (= 10/12°)

(9) Der Kapitän ist abgelenkt und sendet irrtümlich ein zu starkes Signal an das Heckruder. Gründe hierfür können eine hohe Arbeitslast (z. B. durch schwierige Wetterbedingungen, gleichzeitige Kommunikation mit anderen Menschen oder Reagieren auf unerwartete Ereignisse wie Fehlalarme), Konzentrationsschwierigkeiten (z. B. durch Müdigkeit oder eine zu niedrige Arbeitslast), oder eine Priorisierung privater Belange sein.

(10) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er von einer zu niedrigen Krängung des Schiffs ausgeht. Gründe hierfür können eine falsche Anzeige der Krängung (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Neigungsmesser oder extreme Wellenbewegungen), eine fehlende Anzeige der Krängung (z. B. durch einen defekten Display oder einen defekten Neigungsmesser), oder eine Fehlinterpretation der angezeigten Krängung sein.

(11) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er von einem falschen Seitenwind ausgeht. Gründe hierfür können eine falsche Anzeige der Windstärke oder -richtung (z. B. durch einen Fehler im Display, ungenau funktionierende Wettersensoren oder einer

Verwirbelung im Sensorbereich), eine fehlende Anzeige der Windstärke oder -richtung (z. B. durch einen defekten Display oder defekte Wettersensoren), oder eine Fehlinterpretation der angezeigten Windstärke oder -richtung sein.

(12) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er von einer zu hohen Geschwindigkeit des Schiffs ausgeht. Gründe hierfür können eine falsche Anzeige der Geschwindigkeit (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Geschwindigkeitssensor oder eine ungewöhnliche Wasserströmung genau am Sensor), eine fehlende Anzeige der Geschwindigkeit (z. B. durch einen defekten Display oder einen defekten Geschwindigkeitssensor), oder eine Fehlinterpretation der angezeigten Geschwindigkeit sein.

(13) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er das Risiko einer Überschreitung des kritischen Punkts unterschätzt. Gründe einer hohen Risikobereitschaft können Zeitknappheit oder die Kultur oder Persönlichkeit des Kapitäns sein.

(14) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er in seinem Training unzureichend über die Bedeutung oder die möglichen Folgen einer Überschreitung des kritischen Punkts informiert wurde. Gründe hierfür können zwischenmenschliche Missverständnisse, zu kurzes Training, schlechte Methodik oder falsche bzw. fehlende Inhalte im Trainingsprogramm sein.

(15) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er ein abnormales Verhalten des Schiffs nicht ausreichend berücksichtigt. Gründe für das abnormale Verhalten können eine Beschädigung oder eine starke oder ungleichmäßige Beladung sein.

Nichtsenden des Steuerbordschub-Signals, wenn die Krängung zu stark wird (> 10/12°)

(16) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er unvermittelt arbeitsunfähig geworden ist. Gründe hierfür können Schlaf, körperliche Beschwerden (z. B. Schlaganfälle oder Ohnmacht), oder Verhinderung durch externe Faktoren (z. B. extreme Wetterbedingungen) sein.

(17) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er abgelenkt ist. Gründe hierfür können eine hohe Arbeitslast (z. B. durch schwierige Wetterbedingungen, gleichzeitige Kommunikation mit anderen Menschen oder Reagieren auf unerwartete Ereignisse wie Fehlalarme), Konzentrationsschwierigkeiten (z. B. durch Müdigkeit oder eine zu niedrige Arbeitslast), oder eine Priorisierung privater Belange sein.

(18) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er von einer zu niedrigen Krängung des Schiffs ausgeht. Gründe hierfür können eine falsche Anzeige der Krängung (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Neigungsmesser oder extreme Wellenbewegungen), eine fehlende Anzeige der Krängung (z. B. durch einen defekten Display oder einen defekten Neigungsmesser), oder eine Fehlinterpretation der angezeigten Krängung sein.

(19) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er von einer zu hohen Geschwindigkeit des Schiffs ausgeht. Gründe hierfür können eine falsche Anzeige der Geschwindigkeit (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Geschwindigkeitssensor oder eine ungewöhnliche Wasserströmung genau am Sensor), eine

fehlende Anzeige der Geschwindigkeit (z. B. durch einen defekten Display oder einen defekten Geschwindigkeitssensor), oder eine Fehlinterpretation der angezeigten Geschwindigkeit sein.

(20) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er von einer zu hohen Drehzahl der Schraube ausgeht. Gründe hierfür können eine falsche Anzeige der Drehzahl (z. B. durch einen Fehler im Display, einen ungenau funktionierenden Drehzahlsensor oder eine ungewöhnliche Wasserströmung genau am Sensor), eine fehlende Anzeige der Drehzahl (z. B. durch einen defekten Display oder einen defekten Drehzahlsensor), oder eine Fehlinterpretation der angezeigten Drehzahl sein.

(21) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er von einem falschen Seitenwind ausgeht. Gründe hierfür können eine falsche Anzeige der Windstärke oder -richtung (z. B. durch einen Fehler im Display, ungenau funktionierende Wettersensoren oder einer Verwirbelung im Sensorbereich), eine fehlende Anzeige der Windstärke oder -richtung (z. B. durch einen defekten Display oder defekte Wettersensoren), oder eine Fehlinterpretation der angezeigten Windstärke oder -richtung sein.

(22) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er das Risiko einer Überschreitung des kritischen Punkts unterschätzt. Gründe einer hohen Risikobereitschaft können Zeitknappheit oder die Kultur oder Persönlichkeit des Kapitäns sein.

(23) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er in seinem Training unzureichend über die Bedeutung oder die möglichen Folgen einer Überschreitung des kritischen Punkts informiert wurde. Gründe hierfür können zwischenmenschliche Missverständnisse, zu kurzes Training, schlechte Methodik oder falsche bzw. fehlende Inhalte im Trainingsprogramm sein.

(24) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, weil er ein abnormales Verhalten des Schiffs nicht ausreichend berücksichtigt. Gründe für das abnormale Verhalten können eine Beschädigung oder eine starke oder ungleichmäßige Beladung sein.

6.4.2 Erstellung des Modells

Nach der erschöpfenden Suche nach Verlustszenarien für die drei oben behandelten unsicheren Regelsignale ohne Betrachtung des Regelalgorithmus folgt hiernach die Durchführung einer zweiten Identifikationsphase, die zusätzlich Gebrauch von dem in Kapitel 4 entwickelten Modellierungskonzept und der in Kapitel 5 vorgestellten Implementierung macht. Damit diese möglich ist, muss der modellierte Regelalgorithmus mindestens die Regelsignale beinhalten, die zu den drei analysierten unsicheren Signalen gehören. Alle anderen Regelsignale können weggelassen werden, da es bei diesen keinen Versuch gab, Verlustszenarien ohne explizite Modellierung des Algorithmus zu sammeln, weshalb ein Vergleich nicht stattfinden kann. Diesen Überlegungen entsprechend zeigt Abb. 6.8 den in XSTAMPP modellierten Regelalgorithmus für *Bugstrahlschub*, *Ruderausschlag* und *Steuerbordschub*. Bei einem regulären Anlegemanöver dienen die von den Regeln errechneten kontinuierlichen Werte als ungefähre Richtlinien, und können nach Ermessen des Kapitäns um kleine Beträge geändert werden. In Notsituationen hingegen werden häufig extreme Werte (z. B. 100%, 0%, ...) erfordert, dessen strikte Befolgung kritisch ist.

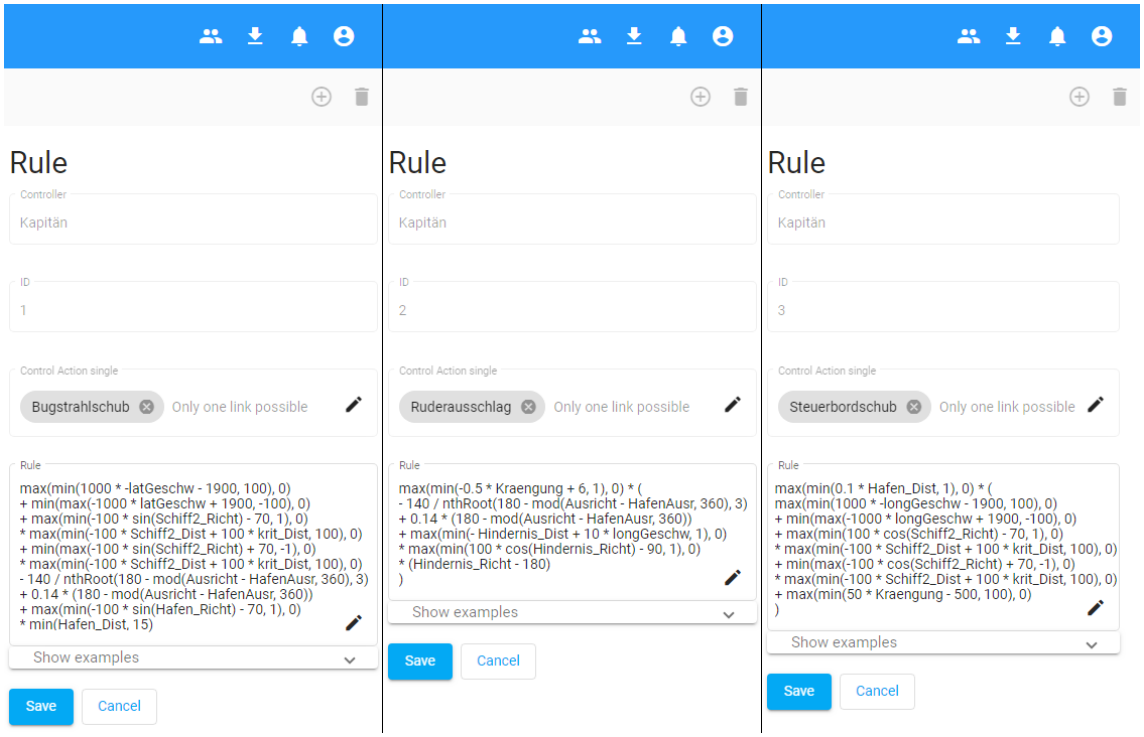


Abbildung 6.8: Drei Regeln im Regelalgorithmus des Kapitäns

`latGeschw` und `longGeschw` bezeichnen die laterale und longitudinale Geschwindigkeit der Fähre in Knoten, wobei ein positiver Wert eine Bewegung nach vorn bzw. Steuerbord und ein negativer Wert eine Bewegung nach hinten bzw. Backbord ausdrückt. Windgeschwindigkeit und Wasserströmung werden als bereits in diese Geschwindigkeiten mit einkalkuliert angenommen. `Ausricht` ist die Ausrichtung des Schiffs in Grad, wobei 0° nach Norden, 90° nach Osten, 180° nach Süden und 270° nach Westen zeigt. Gleichermaßen ist `HafenAusr` die Ausrichtung der Anlegestelle und damit der Sollwert der Ausrichtung der Fähre nach Beendigung des Manövers. Auch die Richtung, in der sich die Anlegestelle befindet (`Hafen_Richt`), verwendet dieses Schema. Darüber hinaus gibt es `Hafen_Dist`, was die Distanz zur Anlegestelle in Metern anzeigt. In gleicher Weise werden die Richtungen und Distanzen zu dem nächsten Schiff (`Schiff2_Richt`, `Schiff2_Dist`) und dem nächsten Hindernis (`Hindernis_Richt` und `Hindernis_Dist`) gemessen. `krit_Dist` gibt außerdem den nötigen Sicherheitsabstand zu dem Schiff an (10 oder 50 Meter). Zu guter Letzt bezeichnet `Kraengung` die aktuelle Krängung der Fähre in Grad.

In Abb. 6.9 ist ein Graph der Regel zu ”Ruderausschlag” illustriert. Dieser ist so konfiguriert, dass er den empfohlenen Ausschlag in Abhängigkeit der Ausrichtung der Fähre angibt. Alle anderen Variablen wurden auf konstante Werte gesetzt, die im Gesamtbild ein reguläres Anlegemanöver repräsentieren, d. h. eine leichte Krängung, eine Geschwindigkeit von einem Knoten und kein Hindernis in unmittelbarer Nähe. Der Graph demonstriert, dass die Fähre bis 180° Abweichung nach links, bei mehr als 180° Abweichung nach rechts lenkt. Dabei wird der Ruderausschlag kleiner, je näher die Fähre dem Sollwert ihrer Ausrichtung kommt.

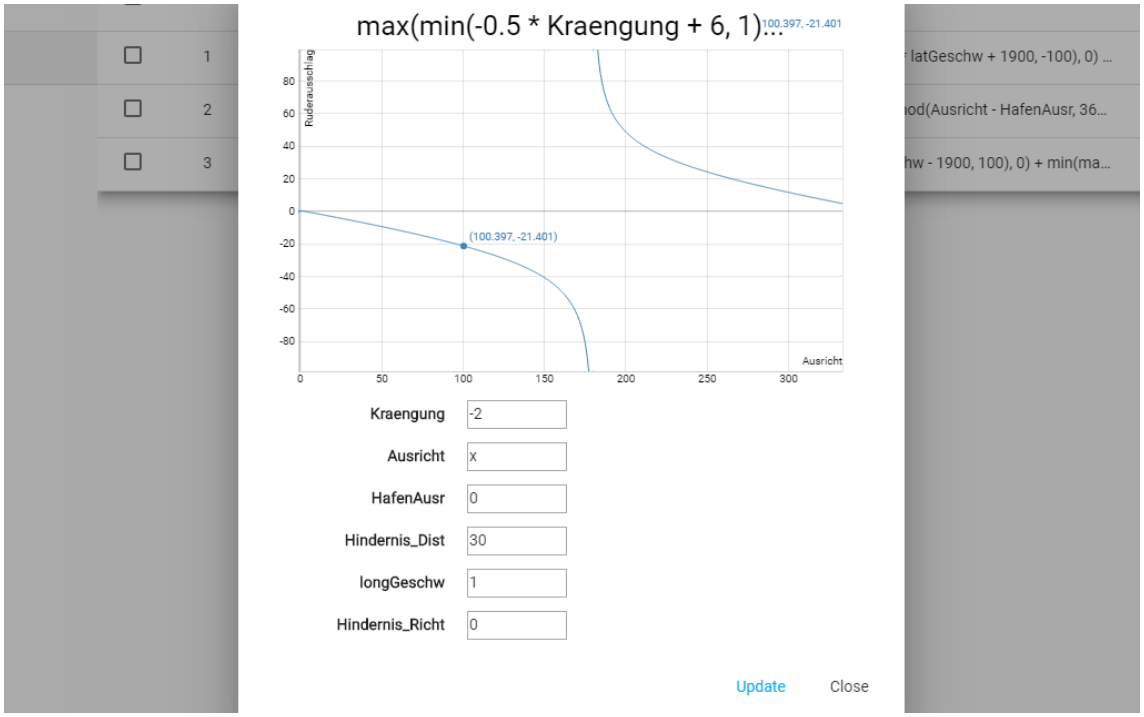


Abbildung 6.9: Graph des Ruderausschlag-Signals

6.4.3 Identifikation mit Regelalgorithmus

Durch Zuhilfenahme des Regelalgorithmus wurden für die drei untersuchten Regelsignale sieben zusätzliche Verlustszenarien identifiziert.

Zu langes Senden des Bugstrahlschub-Signals, wenn die Fähre kurz vor der Anlegestelle ist

(25) Der Kapitän beendet das Senden des Signals zu spät, weil die Fähre noch nicht korrekt ausgerichtet ist, und er durch Senden des Signals die Ausrichtung abschließen will. Gründe hierfür können die Ausrichtung erschwerende Wetterverhältnisse, ein ungewöhnliches Verhalten der Fähre (z. B. durch einen Defekt oder hohe bzw. ungleichmäßige Beladung), oder eine hohe Arbeitslast sein.

(26) Der Kapitän beendet das Senden des Signals rechtzeitig, um vor der Anlegestelle abzubremsen, jedoch nicht rechtzeitig, um vor einem anderen, hervorstehenden Hafenstück zum Stillstand zu kommen. Gründe hierfür können Unvertrautheit mit dem Hafen, Unvertrautheit mit den Ausmaßen des Schiffs, oder schwierige Sichtverhältnisse (z. B. durch Nacht, Nebel oder ein ungünstiges Sichtfeld von der Brücke) sein.

Senden des Ruderausschlag-Signals, wenn die Krängung bereits stark ist (= 10/12°)

(27) Der Kapitän sendet ein zu starkes Signal an das Heckruder, während die Krängung der Fähre zwischen 10° und 12° liegt, weil er eine leichte Kurve fährt, die jedoch tatsächlich noch keine Krängung über 10° rechtfertigt. Grund hierfür kann unzureichendes Training sein.

(28) Der Kapitän sendet ein zu starkes Signal an das Heckruder, weil er einem Hindernis ausweichen will. Gründe hierfür können ein zu spätes Bemerken des Hindernisses (z. B. aufgrund von schlechten Sichtverhältnissen, zu kleinen Ausmaßen des Hindernisses oder hoher Arbeitslast) oder ein plötzlich auftauchendes Hindernis (z. B. verirrte Flusstiere) sein.

Nichtsenden des Steuerbordschub-Signals, wenn die Krängung zu stark wird (> 10/12°)

(29) Der Kapitän sendet kein Schubsignal an die Steuerbordschraube, während die Krängung der Fähre zwischen 10° und 12° liegt, weil er eine leichte Kurve fährt, die jedoch tatsächlich noch keine Krängung über 10° rechtfertigt. Grund hierfür kann unzureichendes Training sein.

(30) Der Kapitän sendet trotz gefährlicher Krängung kein Schubsignal an die Steuerbordschraube, weil er bereits zu nah an der Anlegestelle ist und eine Kollision vermeiden möchte. Gründe hierfür können schwierige Wetterverhältnisse oder ein ungewöhnliches Verhalten der Fähre (z. B. durch einen Defekt oder hohe bzw. ungleichmäßige Beladung) sein.

(31) Der Kapitän sendet trotz gefährlicher Krängung kein Schubsignal an die Steuerbordschraube, weil sich vor ihm ein anderes Schiff oder Hindernis befindet. Gründe für eine Blockierung des Kurses durch ein anderes Schiff können mangelnde Kommunikation (z. B. durch defekte Geräte, hohe Arbeitslast oder Sprachbarrieren) in Verbindung mit schwierigen Wetterverhältnissen oder einem ungewöhnlichen Verhalten der Fähre (z. B. durch einen Defekt oder hohe bzw. ungleichmäßige Beladung) sein. Gründe für eine Blockierung durch ein Hindernis können ein zu spätes Bemerken dieses (z. B. aufgrund von schlechten Sichtverhältnissen, zu kleinen Ausmaßen des Hindernisses oder hoher Arbeitslast) oder ein plötzlich auftauchendes Hindernis (z. B. verirrte Flusstiere) sein.

6.5 Fazit

Durch Zuhilfenahme des Regelalgorithmus wurden sieben weitere Verlustszenarien identifiziert, was hier einem Anstieg von 78% der zu einem unzureichenden Algorithmus gehörenden Szenarien entspricht. Die Identifikationsphase selbst hat dabei weitaus weniger Zeit in Anspruch genommen, als die Identifikationsphase mit herkömmlichen Mitteln. Der Großteil der gefundenen Szenarien behandelt ein ungünstiges Zusammenspiel mehrerer für das Senden des Regelsignals relevanter Faktoren. Jedoch wurden auch Szenarien gefunden, die die falsche Verarbeitung eines einzelnen Faktors behandeln.

Das Ergebnis zeigt, dass das Einbeziehen eines Regelalgorithmenmodells, das nach dem hier vorgestellten Konzept entworfen wurde, eine signifikante Hilfe bei der Sammlung von Verlustszenarien sein kann. Auch wenn das Ergebnis nicht zeigt, dass die Anwendung des Konzepts immer sinnvoll ist, wurde zumindest die Existenz von Fällen, in denen ein klarer positiver Effekt erzielt werden kann, nachgewiesen.

Um von dem Modell des Regelalgorithmus Gebrauch machen zu können, ist zunächst die Erstellung dieses erforderlich. Hierbei muss je nach Zahl und Komplexität der Regelsignale mit einem erheblichen Zeitaufwand gerechnet werden. Ob die zusätzlich gefundenen Szenarien diese Ressourcen wert ist, bedarf einer sorgfältigen Abwägung. Des Weiteren ist nicht klar, ob das selbe Ergebnis nicht auch mit anderen Modellierungskonzepten oder speziellen Analysemethoden erreicht werden kann. Um dies zu untersuchen, ist eine direkte Gegenüberstellung notwendig. Außerdem macht der

obige Test keine Aussage dazu, ob die Modellierung der Aktoren mit Konvertertabellen ebenso gute Resultate erzielen kann, da diese im Normalfall weitaus weniger komplex als der korrespondierende Regelalgorithmus sind.

6.6 Zusammenfassung

In diesem Kapitel wurde beispielhaft eine umfassende STPA-Analyse mithilfe der neu implementierten XSTAMPP Version 4.1 durchgeführt. Der Fokus der Analyse lag auf dem Anlegemanöver einer Binnenfähre an einem Kai, Dalben, Hafenmolden oder Pier – einem Szenario, das sowohl die Betrachtung menschlicher Regler, als auch die Modellierung umfangreicher Regeln erlaubte.

Zu Beginn der Analyse wurden vier Verluste identifiziert, aus denen die Ableitung von acht Randbedingungen möglich war. Diese machten u. a. Angaben zu Krängung, Geschwindigkeit und Sicherheitsabstand. Das zum Anlegemanöver erstellte Regelkreisdiagramm zeigt die Einflussnahme des Kapitäns auf die Bewegung der Fähre. Es enthält neben mehreren Aktoren zur Steuerung des Schiffs und Sensoren für Geschwindigkeits-, Ausrichtungs- und Umgebungsdaten verschiedene Wettereinflüsse sowie Kommunikation vonseiten anderer Schiffe. Hieraus wurden die Verantwortlichkeiten des Kapitäns abgeleitet. Zu den zahlreichen Regel- und Feedbacksignalen erfolgte zudem die Definition von Wertebereichen. Die Identifikationsphase der unsicheren Regelsignale erbrachte insgesamt 82 Ergebnisse, von denen sich jedoch viele ähnelten. Aus den Regelsignalen ergaben sich 82 Randbedingungen auf Reglerebene. Es fanden zwei separate Analysephasen für Verlustszenarien statt. Während der ersten Phase wurden ohne explizite Modellierung des Regelalgorithmus für drei Regelsignale 24 Szenarien gefunden, von denen neun zu einem unzureichenden Algorithmus gehörten. Bei der zweiten Phase war unter Zuhilfenahme eines Regelalgorithmus, der mit dem in Kapitel 4 vorgestellten Konzept modelliert wurde, die Identifikation weiterer sieben Szenarien möglich.

Die Analyseergebnisse zeigten, dass die Nutzung des neuen Konzepts zur Erstellung des Regelalgorithmus die Zahl der gefundenen Verlustszenarien signifikant vergrößern kann. Ein Vergleich mit anderen Modellierungskonzepten hat nicht stattgefunden.

7 Schlusswort

Abschließend sollen die Ergebnisse der gesamten Arbeit in einer übersichtlichen Zusammenfassung festgehalten werden. Danach folgen einige Vorschläge, an welchen Stellen weitere Forschung sinnvoll ist. Dabei werden auf einen Vergleich mit anderen Modellierungskonzepten, XSTAMPP, und Konvertertabellen eingegangen.

7.1 Zusammenfassung der Arbeit

In STPA werden Systeme als Regelkreise betrachtet, wobei ein oder mehrere Regler Signale an Aktoren senden, um Prozesse zu manipulieren. Welche Regelsignale zu welchem Zeitpunkt gesendet werden, entscheidet der Regelalgorithmus anhand der Variablen im Prozessmodell. Unsichere Regelsignale, die zu Verlusten führen, können an jeder Stelle des Regelkreises ihre Ursache haben, inklusive des Regelalgorithmus. Um die Suche nach Verlustszenarien zu vereinfachen, kann es hilfreich sein, den Regelalgorithmus explizit zu modellieren. Hierfür existieren mehrere Konzepte. Eines davon ist eine Modellierung mithilfe von Entscheidungstabellen, deren Vorspalte alle relevanten Variablenkonfigurationen enthält. Die Zellen enthalten boolesche Werte, die zunächst zeilenweise mit dem logischen UND, dann spaltenweise mit dem logischen ODER ausgewertet werden. Neben Entscheidungstabellen werden auch Formeln, Graphen, Schaltkreise und Fließtexte zur Modellierung von Regelalgorithmen verwendet. Die existierenden Konzepte wurden anhand der Kriterien *Verständlichkeit*, *Flexibilität*, *Skalierbarkeit* und *Erweiterbarkeit* beurteilt, wobei als Teil der *Flexibilität* auch die Möglichkeit Beachtung fand, das Konzept in eine an menschliche Regler angepasste STPA-Analyse zu integrieren. Auf Basis des Vergleichs wurde ein Konzept vorgeschlagen, welches sich stark an Formeln orientiert. Es definiert eine *Regel* als eine Abbildung von einer Menge Prozessvariablen zu einem Regelsignal. Sowohl boolesche als auch kontinuierliche Werte können verarbeitet werden. Sind die unsicheren Regelsignale in eine Kontexttabelle eingetragen, ist es möglich, aus der *Nichtsenden*-Spalte der Tabelle alle Regeln mit booleschen Signalen zu generieren. Regeln mit kontinuierlichen Signalen sind aufgrund der Einschränkungen der Kontexttabelle nicht generierbar. Um eine zusätzliche visuelle Repräsentation des Regelalgorithmus zu gewähren, hat diese Arbeit einen Vorschlag dargelegt, wie aus einer Regel ein vollständiger oder partieller Graph erzeugt werden kann. Ferner wurde Anregung gegeben, auf welche Weise ein Übertragen des Modellierungskonzept auf Aktoren möglich ist.

Es gibt mehrere Programme, die das Durchführen einer STPA-Analyse auf dem Computer erlauben. Viele der Programme besitzen keine Funktion zur Modellierung des Algorithmus; einige bauen auf die Entscheidungstabelle. Im Rahmen dieser Arbeit wurde das entwickelte Konzept in die Software XSTAMPP integriert. Dies beinhaltet neben der Verwaltung von Regeln die automatische Vervollständigung von Prozessvariablennamen, die Syntaxanalyse der Regeln, und die Anzeige und Konfiguration der entsprechenden Funktionsgraphen. Auch wurde die Übertragung der Modellierungsmethode auf Aktoren in Form von Konvertertabellen durchgeführt.

Zuletzt fand eine Analyse eines Anlegemanövers statt, um die Effektivität des Konzepts und dessen Implementierung zu erproben. Das Ergebnis zeigte, dass das Konzept das Potential hat, viele zuvor nicht gefundene Verlustszenarien zu identifizieren.

7.2 Ansätze für weiterführende Forschung

In dieser Arbeit wurden einige Themen angesprochen, die einer näheren Erforschung bedürfen. Dazu gehören die Effektivität von Formeln im direkten Vergleich zu anderen Modellierungskonzepten, sinnvolle Erweiterungen für XSTAMPP und die Brauchbarkeit von Konvertertabellen. Diese werden im Folgenden erörtert.

7.2.1 Vergleich mit anderen Modellierungskonzepten

Die Arbeit hat gezeigt, dass ein Modell des Regelalgorithmus, welches im Kern einer Menge von Formeln entspricht, einen positiven Einfluss auf die Zahl der gefundenen Verlustszenarien haben kann. Es ist sehr wahrscheinlich, dass auch mit anderen Modellierungsstrategien ein solcher Effekt erreichbar ist. Eine wichtige Frage ist daher, welches Konzept sich am besten für die Analyse eignet. In diesem Zusammenhang muss auch untersucht werden, inwiefern die Konzepte Unterschiede bei der Analyse von menschlichen, technischen und soziologischen Reglern aufweisen. Obwohl die Modellierung mithilfe von Regelalgorithmen durch mehrere Programme unterstützt wird, gibt es klare Abweichungen bei der Implementierung. Auch hier können Unterschiede erforscht werden.

Neben der Zahl der zusätzlich identifizierten Szenarien wurde hier auch auf Faktoren wie Verständlichkeit, Flexibilität, Skalierbarkeit und Erweiterbarkeit eingegangen. Dabei waren jedoch keine über theoretische Überlegungen hinausgehende Experimente möglich. Solcherlei Experimente können ebenso Teil weiterführender Forschung sein.

7.2.2 XSTAMPP

XSTAMPP wurde um mehrere Funktionen erweitert, die die Arbeit mit den hier vorgestellten Konzepten ermöglichen. Nichtsdestoweniger gibt es noch immer zahlreiche Stellen, an denen eine Ergänzung weiterer Funktionen sinnvoll ist. Ein wichtiger Punkt ist die automatische Generierung des Regelalgorithmus aus den unsicheren Regelsignalen. Hierzu fehlt bisher die Möglichkeit, unterschiedliche Konfigurationen der Prozessvariablen zu erstellen und mit den gewünschten Signalen zu verbinden. Weiterhin ist zu untersuchen, auf welche Weise kontinuierliche Variablen und kontinuierliche Regelsignale verarbeitet werden können. Ein weiterer Punkt ist die Verbesserung des Programms in Bezug auf die Analyse menschlicher Regler. Für diesen Zweck wurden bereits mehrere Ansätze vorgeschlagen [Tho14][Fra17][Mon16]. Bisher ist jedoch keiner dieser Ansätze in einer Software umgesetzt worden. Ein dritter Punkt ist die Implementierung temporaler logischer Formeln zur Ableitung von Softwareverifizierungstests. Diese existierte bereits in alten Versionen von XSTAMPP Abdulkhaleq und Wagner [AW15b], ist jedoch noch nicht in die Weboberfläche von Version 4.0 integriert.

7.2.3 Konvertertabellen

Ein Teil der Arbeit behandelte die Möglichkeit, das Modellierungskonzept für Regelalgorithmen auf Aktoren zu übertragen. In XSTAMPP fand eine solche Übertragung in Form der Konvertertabellen statt. Zu der Effektivität dieser Tabellen fanden keine Untersuchungen statt. Weiterführende Forschung könnte an diesem Punkt ansetzen oder ermitteln, inwiefern weitere Komponenten des Regelkreises (z. B. Sensoren) von einer expliziten Modellierung mit Formeln profitieren. Zusätzlich kann geprüft werden, ob sich andere Konzepte bei diesen weniger komplexen Abbildungen besser eignen.

Literaturverzeichnis

[ADS83] H. E. G. A. D. Swain. *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications*. Techn. Ber. Sandia National Labs, 1983. DOI: 10.2172/5752058 (zitiert auf S. 48).

[AW13] A. Abdulkhaleq, S. Wagner. „Experiences with Applying STPA to Software-Intensive Systems in the Automotive Domain“. In: *2013 STAMP Workshop*. März 2013. DOI: 10.18419/opus-3531 (zitiert auf S. 41).

[AW14] A. Abdulkhaleq, S. Wagner. *A-STPA: An Open Tool Support for System-Theoretic Process Analysis*. STAMP 2014 Conference at MIT. 2014 (zitiert auf S. 16).

[AW15a] A. Abdulkhaleq, S. Wagner. „XSTAMPP: An eXtensible STAMP platform as tool support for safety engineering“. In: *2015 STAMP Conference at MIT*. 2015. DOI: 10.13140/2.1.3862.0486 (zitiert auf S. 16, 29).

[AW15b] A. Abdulkhaleq, S. Wagner. „Integrated Safety Analysis Using Systems-Theoretic Process Analysis and Software Model Checking“. In: *Computer Safety, Reliability, and Security: 34th International Conference, SAFECOMP 2015 Delft, The Netherlands, September 23 – 25, 2015 Proceedings*. Springer, 2015. DOI: 10.1007/978-3-319-24255-2_10 (zitiert auf S. 62, 100).

[Abd17] A. Abdulkhaleq. *XSTAMPP Foundations and Examples: eXtensible STAMP Platform for Safety Engineering*. 2017. URL: https://www.slideshare.net/AsimAbdulkhaleq/xstampp-foundations (zitiert auf S. 16).

[Ade+17] A. A. Adesina, Q. Hussain, S. Pandit, M. Rejzek, A. M. Hochberg. „Assessing the Value of System Theoretic Process Analysis in a Pharmacovigilance Process: An Example Using Signal Management“. In: *Pharmaceutical Medicine* 31.4 (Aug. 2017), 267—278. DOI: https://doi.org/10.1007/s40290-017-0195-5 (zitiert auf S. 41).

[Ale+14] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson, R. Iyer. „Systems-Theoretic Safety Assessment of Robotic Telesurgical Systems“. In: *Computer Safety, Reliability, and Security* (2014), S. 213–227. DOI: https://doi.org/10.1007/978-3-319-24255-2_16 (zitiert auf S. 41).

[Ant13] B. Antoine. „Systems Theoretic Hazard Analysis (STPA) Applied to The Risk Review of Complex Systems: An Example From The Medical Device Industry“. Diss. Massachusetts Institute of Technology, 2013 (zitiert auf S. 40).

[BEH14] C. Becker, Q. van Eikema Hommes. *Transportation systems safety hazard analysis tool (SafetyHAT) user guide*. Techn. Ber. John A. Volpe National Transportation Systems Center, 2014 (zitiert auf S. 37).

[BS77] K. Basel-Stadt. *Hafenordnung für die Rheinhäfen beider Basel*. § 23.4, 23.5. März 1977. URL: https://www.gesetzessammlung.bs.ch/app/de/texts_of_law/955.460 (zitiert auf S. 87).

[Bag+17] G. Bagschik, T. Stolte, M. Maurer. „Safety Analysis Based on Systems Theory Applied to an Unmanned Protective Vehicle“. In: *Procedia Engineering* 179 (2017), S. 61–71. DOI: https://doi.org/10.1016/j.proeng.2017.03.096 (zitiert auf S. 41, 43).

[Ber68] L. von Bertalanffy. *General System Theory: Foundations, Development, Applications*. George Braziller, 1968 (zitiert auf S. 19).

[Bre92] B. Brehmer. „Dynamic decision making: Human control of complex systems“. In: *Acta Psychologica* 81.3 (1992), S. 211–241. DOI: https://doi.org/10.1016/0001-6918(92)90019-A (zitiert auf S. 21).

[Bro+17] C. Brown, J. Zheng, S. H. Björnsdóttir, M. Rejzek. „STPA software module: a Eurostars funded software project“. In: *5th European STAMP/STPA Workshop and Conference*. Sep. 2017 (zitiert auf S. 35).

[Cas+18] D. S. Castilho, L. M.S.Urbina, D. Andrade. „STPA for continuous controls: A flight testing study of aircraft crosswind takeoffs“. In: *Safety Science* 108 (Okt. 2018). DOI: https://doi.org/10.1016/j.ssci.2018.04.013 (zitiert auf S. 58, 62).

[Cha+17] M. M. Chatzimichailidou, N. Karanikas, A. Plioutsias. „Application of STPA on Small Drone Operations: A Benchmarking Approach“. In: *Procedia Engineering* 179 (2017), S. 13–22. DOI: https://doi.org/10.1016/j.proeng.2017.03.091 (zitiert auf S. 41).

[Cha14] K. Chandra. *nearly*. 2014. URL: https://nearley.js.org/ (zitiert auf S. 74).

[Cha19] A. Chakrabarty. *How Bow Thruster is Used for Maneuvering a Ship?* März 2019. URL: https://www.marineinsight.com/marine-navigation/how-bow-thruster-is-used-for-maneuvering-a-ship/ (zitiert auf S. 87).

[Cou18] G. Couzens. *'This could have ended in tragedy' Shocking moment ferry HITS cruise liner*. Jan. 2018. URL: https://www.express.co.uk/news/world/904140/ferry-cruise-ship-collision-barcelona-port-spain-engine-failure (zitiert auf S. 89).

[Def49] U. S. D. of Defense. *Procedures for performing a failure mode effect and critical analysis*. MIL-P-1629. Nov. 1949. URL: https://www.slideshare.net/AsimAbdulkhaleq/xstampp-foundations (zitiert auf S. 20).

[Dek06] S. Dekker. *The Field Guide to Understanding Human Error*. Ashgate Publishing Company, Juni 2006 (zitiert auf S. 50).

[Don13] A. Dong. „Application of CAST and STPA to railroad safety in China“. Magisterarb. Massachusetts Institute of Technology, Jan. 2013 (zitiert auf S. 41, 42).

[Dun13] N. C. Dunn. „Satellite System Safety Analysis Using STPA“. Magisterarb. Massachusetts Institute of Technology, 2013. URL: http://hdl.handle.net/1721.1/85777 (zitiert auf S. 41).

[EU98] R. der Europäischen Union. *SchSV - Schiffssicherheitsverordnung*. Anhang zu Anhang I, Kapitel II-1, Teil B, Abschnitt 1, Nummer 1, f und g. Sep. 1998. URL: https://www.umwelt-online.de/regelwerk/cgi-bin/suchausgabe.cgi?pfad=/gefahr.gut/see/schishv1.htm&such=Schiff (zitiert auf S. 87).

[Emb86] D. E. Embrey. „SHERPA: A systematic human error reduction and prediction approach“. In: *International Topical Meeting on Advances in Human Factors in Nuclear Power Systems*. 1986 (zitiert auf S. 48).

[Fit51] P. M. Fitts. *Human engineering for an effective air-navigation and traffic-control system*. National Research Council, 1951 (zitiert auf S. 47).

[Fle+12] C. Fleming, T. Ishimatsu, Y. Miyamoto, H. Nakao, M. Katahira, N. Hoshino, J. Thomas, N. Leveson. „Safety Guided Spacecraft Design Using Model-Based Specifications“. In: *Proceedings of the 5th IAASS Conference: A Safer Space for Safer World*. 2012 (zitiert auf S. 43).

[Fle+13] C. H. Fleming, M. S. Placke, N. Leveson. *STPA Analysis of NextGen Interval Management Components: Ground Interval Management (GIM) and Flight Deck Interval Management (FIM)*. Techn. Ber. Federal Aviation Administration und Lincoln Lab, Sep. 2013 (zitiert auf S. 43).

[Fra17] M. E. France. „Engineering for Humans: A New Extension to STPA“. Magisterarb. Massachusetts Institute of Technology, Juni 2017 (zitiert auf S. 49, 50, 58, 100).

[GS08] G. Gui, P. D. Scott. „New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability“. In: *2008 The 9th International Conference for Young Computer Scientists* (Nov. 2008). DOI: 10.1109/ICYCS.2008.270 (zitiert auf S. 77).

[H17] 橋本岳男, システムモデルを用いたSTAMP/STPA試行の事例紹介 In: *2nd Japanese STAMP Workshop*. 2017 (zitiert auf S. 43, 44).

[HM95] M. Hitz, B. Montazeri. „Measuring Coupling and Cohesion In Object-Oriented Systems“. In: *Proceedings of International Symposium on Applied Corporate Computing*. Okt. 1995, S. 25–27 (zitiert auf S. 77).

[Hom12] Q. D.V. E. Hommes. „Applying System Theoretical Hazard Analysis Method to Complex Automotive Cyber Physical Systems“. In: *9th International Conference on Design Education and 24th International Conference on Design Theory and Methodology*. Bd. 7. Aug. 2012, S. 705–717. DOI: doi:10.1115/DETC2012-70527 (zitiert auf S. 41).

[Hom14] Q. Hommes. *SafetyHAT: A Transportation Systems Safety Hazards Analysis Tool*. STAMP 2014 Conference at MIT. 2014 (zitiert auf S. 16, 37).

[I18] 石井正悟 et. al., システム理論に基づく安全性解析手法STAMP/STPAの普及促進 In: *SEC journal* 1 (Aug. 2018), S. 73–75 (zitiert auf S. 36).

[IFR19] IFRC. *Emergency Plan of Action Operation Final Report – Tanzania: Ferry Accident*. Juli 2019. URL: http://adore.ifrc.org/Download.aspx?FileId=243079 (zitiert auf S. 87).

[II99] C. A. E. II. „Fault Tree Analysis - A History“. In: *17th International System Safety Conference*. System Safety Society, Aug. 1999 (zitiert auf S. 20).

[Iso] *ISO/IEC 25010:2011*. März 2011. URL: https://www.iso.org/standard/35733.html (zitiert auf S. 53).

[JD70] F. J. J. Johnston, J. C. Davis. *Decision tables in data processing: a report*. National Computing Centre, 1970 (zitiert auf S. 17).

[Jon97] M. Jones. *What really happened on Mars Rover Pathfinder*. Dez. 1997. URL: https://cse.buffalo.edu/~bina/cse321/fall2015/MarsRover.pdf (zitiert auf S. 66).

[Kin19] Kinsta. *What Is Nginx? A Basic Look at What It Is and How It Works*. Juni 2019. URL: https://kinsta.com/knowledgebase/what-is-nginx/ (zitiert auf S. 31).

[Kle92] T. A. Kletz. *Hazop and Hazan: Identifying and Assessing Process Industry Hazards*. Inst of Chemical Engineers UK, 1992 (zitiert auf S. 20).

[Kra+15] S. S. Krauss, M. Rejzek, C. Hilbes. „Tool Qualification Considerations for Tools Supporting STPA“. In: *Procedia Engineering* 128 (2015), S. 15–24. DOI: https://doi.org/10.1016/j.proeng.2015.11.500 (zitiert auf S. 34).

[Kra+16] S. S. Krauss, M. Rejzek, C. Senn, C. Hilbes. „SAHRA - an integrated software tool for STPA“. In: *4th European STAMP Workshop*. ZHAW Zürcher Hochschule für Angewandte Wissenschaften, Sep. 2016. DOI: 10.21256/zhaw-4926 (zitiert auf S. 34).

[LH17] J. Liu, R. Hekkenberg. „Sixty years of research on ship rudders: effects of design choices on rudder performance“. In: *Ships and Offshore Structures* 12.4 (2017), S. 495–512. DOI: 10.1080/17445302.2016.1178205 (zitiert auf S. 86).

[LM16] C. H. N. Lahoz, S. R. G. Medeiros. *Systematic review on STPA: Final Results*. 2016 STAMP Workshop. 2016 (zitiert auf S. 40).

[LT13] N. G. Leveson, J. P. Thomas. *An STPA Primer*. Partnership for Systems Approaches to Safety und Security, 2013 (zitiert auf S. 40, 43, 47).

[LT18] N. G. Leveson, J. P. Thomas. *STPA Handbook*. Partnership for Systems Approaches to Safety und Security, März 2018 (zitiert auf S. 26, 29, 65, 76, 83).

[Lep84] J. Leplat. „Occupational accident research and systems approach“. In: *Journal of Occupational Accidents* 6.1–3 (Sep. 1984), S. 211–241. DOI: https://doi.org/10.1016/0376-6349(84)90036-1 (zitiert auf S. 22, 23).

[Lev+14] N. Leveson, C. Wilkinson, C. Fleming, J. Thomas, I. Tracy. *A Comparison of STPA and the ARP 4761 Safety Assessment Process*. Techn. Ber. Okt. 2014 (zitiert auf S. 45–47).

[Lev+98] N. G. Leveson, J. D. Reese, M. P. E. Heimdahl. „SpecTRM: a CAD system for digital automation“. In: *17th Digital Avionics Systems Conference*. IEEE, Nov. 1998. DOI: 10.1109/DASC.1998.741474 (zitiert auf S. 33).

[Lev04] N. G. Leveson. „A new accident model for engineering safer systems“. In: *Safety Science* 42.4 (Apr. 2004), S. 126–141. DOI: https://doi.org/10.1016/S0925-7535(03)00047-X (zitiert auf S. 15, 46).

[Lev11] N. G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 2011 (zitiert auf S. 15, 20, 23, 24, 47).

[Mac18] D. MacFarlane. *Angular Mentions*. Juli 2018. URL: https://www.npmjs.com/package/angular-mentions (zitiert auf S. 73).

[Mar15] R. S. Martínez. „System theoretic process analysis of electric power steering for automotive applications“. Magisterarb. Massachusetts Institute of Technology, 2015 (zitiert auf S. 40).

[McA+10] J. McAffer, J.-M. Lemieux, C. Aniszczyk. *Eclipse Rich Client Platform*. 2. Aufl. Addison-Wesley Professional, Mai 2010 (zitiert auf S. 29).

[Mea08] D. H. Meadows. *Thinking in Systems: A Primer*. Chelsea Green Publishing, 2008 (zitiert auf S. 19).

[Mer+19] A. D. Merladet, C. Lahoz, A. Silveira, S. Fugivara. „STPA Applied to Launch Operations Management“. In: *2019 STAMP Workshop*. 2019 (zitiert auf S. 41).

[Mew01] F. Mewis. „Pod Drives – Pros and Cons“. In: *HANSA-Schiffahrt-Schiffbau-Hafen* 138 (2001), S. 25–30 (zitiert auf S. 84).

[Mon16] D. R. Montes. „Using STPA to Inform Developmental Product Testing“. Diss. Massachusetts Institute of Technology, Feb. 2016 (zitiert auf S. 47–49, 58, 100).

[Osc+16] J. Oscarsson, M. Stolz-Sundnes, N. Mohan, V. Izosimov. „Applying systems-theoretic process analysis in the context of cooperative driving“. In: *11th IEEE Symposium on Industrial Embedded Systems*. IEEE, Mai 2016. DOI: 10.1109/SIES.2016.7509433 (zitiert auf S. 41).

[PB15] M. Pappot, R. J. de Boer. „The Integration of Drones in Today's Society“. In: *Procedia Engineering* 128 (2015), S. 54–63. DOI: https://doi.org/10.1016/j.proeng.2015.11.504 (zitiert auf S. 41).

[PK15] A. Plioutsias, N. Karanikas. „Using STPA in the Evaluation of Fighter Pilots Training Programs“. In: *Procedia Engineering* 128 (2015), S. 25–34. DOI: https://doi.org/10.1016/j.proeng.2015.11.501 (zitiert auf S. 41).

[Paw+16] T. Pawlicki, A. Samost, D. W. Brown, R. P. Manger, G.-Y. Kim, N. G. Leveson. „Application of systems and control theory-based hazard analysis to radiation oncology“. In: *Medical Physics* 43.3 (März 2016). DOI: 10.1118/1.4942384 (zitiert auf S. 47).

[Pla14] M. S. Placke. „Application of STPA to the integration of multiple control systems: a case study and new approach“. Magisterarb. Massachusetts Institute of Technology, 2014 (zitiert auf S. 41, 43).

[Pop15] M. Poppe. *Function Plot*. Apr. 2015. URL: https://mauriciopoppe.github.io/function-plot/ (zitiert auf S. 74).

[RH14] M. Rejzek, C. Hilbes. „Use of STPA in digital instrumentation and control systems of nuclear power plants“. In: *Informatik 2014*. Gesellschaft für Informatik e.V., 2014, S. 649–650 (zitiert auf S. 40).

[Rej+12] M. Rejzek, N. Leveson, B. Antoine, C. Hilbes, M. Grossmann, D. Meer. „Evaluation of STPA in the Safety Analysis of the Gantry 2 Proton Radiation Therapy System“. In: *1st MIT STAMP Workshop*. Apr. 2012 (zitiert auf S. 40).

[Rev+19] K. M. A. Revell, C. Allison, R. Sears, N. A. Stanton. „Modelling distributed crewing in commercial aircraft with STAMP for a rapid decompression hazard“. In: *Ergonomics* 62.2 (Feb. 2019), S. 156–170. DOI: 10.1080/00140139.2018.1514467 (zitiert auf S. 41).

[SK12] S. T. Shorrock, B. Kirwan. „Development and application of a human error identification tool for air traffic control“. In: *Applied Ergonomics* 33.4 (Juli 2012). DOI: https://doi.org/10.1016/S0003-6870(02)00010-8 (zitiert auf S. 48).

[SS18] D. Schmid, N. A. Stanton. „How are laser attacks encountered in commercial aviation? A hazard analysis based on systems theory“. In: *Safety Science* 110 (Dez. 2018), S. 178–191. DOI: https://doi.org/10.1016/j.ssci.2018.08.012 (zitiert auf S. 41, 47).

[ST14] D. Suo, J. Thomas. *An STPA Tool*. 2014 STAMP Conference at MIT. 2014 (zitiert auf S. 16, 33, 40, 57, 61).

[Spr] *Reference Documentation - Spring*. Version 4.3.9. 2016. URL: https://docs.spring.io/spring/docs/4.3.9.RELEASE/spring-framework-reference/htmlsingle/#overview-dependency-injection (zitiert auf S. 77).

[Str10] M. V. Stringfellow. „Accident analysis and hazard analysis for human and organizational factors“. Diss. Massachusetts Institute of Technology, Okt. 2010 (zitiert auf S. 49).

[Sul+14] S. M. Sulaman, T. Abbas, K. Wnuk, M. Höst. „Hazard Analysis of Collision Avoidance System using STPA“. In: *Proceedings of the 11th International ISCRAM Conference*. Mai 2014 (zitiert auf S. 40).

[Sul+19] S. M. Sulaman, A. Beer, M. Felderer, M. Höst. „Comparison of the FMEA and STPA safety analysis methods — a case study“. In: *Software Quality Journal*. Bd. 27. 1. Springer, 2019, 349—387. DOI: https://doi.org/10.1007/s11219-017-9396-0 (zitiert auf S. 40, 46, 47).

[Tak+17] Y. Takano, H. Sasaki, Y. Morita, K. Sugiura, T. Kawano. „Application and extension of STAMP/STPA to Railway Signalling System“. In: *2nd Japanese STAMP Workshop*. Nov. 2017 (zitiert auf S. 41).

[Tho+12] J. Thomas, F. L. de Lemos, N. Leveson. *Evaluating the Safety of Digital Instrumentation and Control Systems in Nuclear Power Plants*. Techn. Ber. Massachusetts Institute of Technology, Nov. 2012 (zitiert auf S. 40).

[Tho13] J. P. Thomas. „Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis“. Diss. Massachusetts Institute of Technology, 2013. DOI: 10.2172/1044959 (zitiert auf S. 59, 61).

[Tho14] C. L. Thornberry. „Extending the Human-Controller Methodology in SystemsTheoretic Process Analysis (STPA)“. Magisterarb. Massachusetts Institute of Technology, Juni 2014 (zitiert auf S. 47–49, 58, 100).

[Uml] *OMG Unified Modeling Language (OMG UML)*. Version 2.5.1. Dez. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (zitiert auf S. 44).

[Win16] C. Winkler. *How Many Sensors are in a Drone, And What do they Do?* Juli 2016. URL: https://www.sensorsmag.com/components/how-many-sensors-are-a-drone-and-what-do-they-do (zitiert auf S. 63).

Alle URLs wurden zuletzt am 1. 08. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift