Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Adaption des Systems XSTAMPP 4 an die Analysemethode STAMP/CAST in der Einzelplatzanwendung

Eva Zimmermann

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Wolfgang Fechner
Beginn am:	8. Januar 2020
Beendet am:	2. September 2020

Kurzfassung

Täglich geschehen Unfälle, die analysiert werden müssen und für die Erklärungen gefunden werden sollten. Dazu gibt es einen Analyseprozess CAST, der auf STAMPP aufbauend, existierende Unfälle betrachtet und durch dessen Erkenntnisse weitere Unfälle verhindert werden sollen. Um diesen Prozess zu unterstützen, wird in dieser Bachelorarbeit eine Einzelplatzanwendung umgesetzt, die den Anwender bei der Analyse von Unfällen unterstützt. Dafür wurde aufbauend auf der Theorie und den existierenden Arbeiten, eine Anforderungsanalyse durchgeführt, auf dessen Grundlage dann die Einzelplatzanwendung implementiert wurde. Als Ergebnis der Arbeit wurde eine Software fertiggestellt, die es dem Analyst ermöglicht, alle Schritte von CAST durchzuführen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

CAST Causal Analysis Based on System Theory. 13

FMECA Failure Modes and Effects Criticality Analysis. 18

FTA Fault Tree Analysis. 18

SIS Sicherheitsinformationssystem. 22

STAMP System Theoretic Accident Model and Process. 13

STPA System-Theoretic Process Analysis. 14

XSTAMPP eXtensible STAMP Platform. 13

1 Einleitung

1.1 Motivation

Täglich geschehen Unfälle. Diese zu vermeiden ist eine wichtige und herausfordernde Aufgabe. Die erste Frage, die sich nach dem Auftreten eines Unfalles stellt, ist meist die Frage nach einem Schuldigen. Dabei wird außer Acht gelassen, dass dieser, im Regelfall, versucht hat, bestmöglich nach seinem Kenntnisstand zu handeln [BSS17].

Doch in der Analyse von Unfällen sollten andere Faktoren, als das Finden eines Schuldigen, im Vordergrund stehen. Denn *"An accident where innocent people are killed is tragic, but not nearly as tragic as not learning from it."* [Lev19, Seite2].

Um aus den Unfällen lernen zu können, soll hier Causal Analysis Based on System Theory (CAST) verwendet werden. Die Analysemethode basiert auf System Theoretic Accident Model and Process (STAMP) und soll kausale Faktoren herausstellen, die zum Unfall beigetragen haben. Das Ziel ist es, die Zusammenhänge zu analysieren, die zum Unfall geführt haben, denn die Verurteilung eines Schuldigen beseitigt nicht die Ursachen der Unfälle. Wenn ein Schuldiger nach bestem Gewissen handelte, wird eine Neubesetzung der Stelle das Unfallrisiko nicht vermindern können [Lev19].

In der Durchführung von CAST werden sowohl Faktoren innerhalb des Systems, wie Arbeitsabläufe betrachtet, aber auch das System als Ganzes, wie beispielsweise die firmeninternen Philosophien [Lev19].

Mit der Implementierung der Einzelplatzanwendung soll dieser Prozess unterstützt werden, um somit hilfreiche Analyseergebnisse zu erstellen. Durch die Anwendung soll eine einheitliche Darstellung von Analysen erreicht werden. Außerdem soll hierdurch sichergestellt werden, dass alle Teile der Analyse durchgeführt werden, um eine vollständige Analyse zu erreichen. Die Implementierung der Software stellt somit einen wichtigen Schritt dar, um die Theorie [Lev19] des Analysesprozesses zu unterstützen und so die Anwendung einfacher zu gestalten.

1.2 Aufgabenstellung

Um die Aufgabe der Umsetzung der Einzelplatzanwendung erfüllen zu können, wurde zunächst im ersten Schritt die Anforderungsanalyse durchgeführt. Außerdem soll sich das System in die Einzelplatzanwendung eingliedern, die von der eXtensible STAMP Platform (XSTAMPP) Version 4.1 entwickelt wird.

Bei der XSTAMPP4.1[1] Anwendung handelt es sich um eine Webanwendung zum Analyseprozess System-Theoretic Process Analysis (STPA). Dies wird im Rahmen einer Bachelorarbeit [Gre20] zu einer Einzelplatzanwendung transferiert. Gemeinsam ist das Ziel, eine Anwendung zu schaffen, die sowohl CAST-Analysen, wie auch die STPA-Analysen verbindet. Die Implementierung der CAST-Analyse wird in dieser Bachelorarbeit umgesetzt.

1.3 Struktur der Arbeit

Kapitel 2 beschreibt zunächst die Grundlagen, auf die sich die Anwendung stützt.

Kapitel 3 fasst kurz zusammen, welche Arbeiten mit dieser Bachelorarbeit im Zusammenhang stehen.

Kapitel 4 analysiert im folgenden die Anforderungen, die an das System gestellt werden.

Kapitel 5 dokumentiert die wichtigsten Konzepte, die bei der Implementierung benutzt wurden, sowie kurz die genutzten Technologien.

Kapitel 6 beschreibt die implementierte Software anhand eines durchgängigen Beispiels.

Kapitel 7 gibt nun zum Abschluss einen kurzen Überblick über die Arbeit und zeigt im Ausblick das weitere Potential der Software auf.

[1] Zugang zur Webanwendung: https://web.xstampp.de/

2 Grundlagen

2.1 System Theoretic Accident Model and Process (STAMP)

STAMP ist der zugrunde liegende Prozess, auf welchem der CAST-Analyseprozess basiert. Dabei handelt es sich um ein Modell, dass auf Kausalitäten beruht [Lev19].

Der Zeitpunkt der Entwicklung der Systemtheorie liegt nach dem 2. Weltkrieg. Denn durch die komplexer werdenden Systeme wurde auch der Sicherheitsaspekt wichtiger. In der Systemtheorie wird das Gesamtsystem betrachtet, um auch Verbindungen zwischen den einzelnen Komponenten, die nicht offensichtlich gekoppelt sind, identifizieren zu können. Zuerst fand sie zwischen den 1950ern und 1960ern Anwendung bei Flugabwehrraketensystemen [Lev19].

Nicht die Verbindung der Komponenten ist entscheidend für das Verhalten, sondern die über diese Verbindung ausgetauschten Interaktionen. Hierbei werden sowohl soziale Aspekte, als auch technische berücksichtigt, denn es gibt Eigenschaften, die nur bei Betrachtung des Gesamtsystems erkannt werden können [Lev19].

STAMP besteht aus drei Hauptbestandteilen, die im Folgenden beschrieben werden. Diese beinhalten: *"safety constraints, hierarchical control structures, and process models"* [Lev11, Seite 76].

2.1.1 Safety Constraints

Safety Constraints bilden die Grundlage des STPA-Prozesses. Dieser wird dafür genutzt, Ereignisse zu verhindern, die zu einem Loss führen [Lev11].

Es wird dabei zwischen *aktiven und passiven Steuerungen* unterschieden [Lev11].

Eine *passive Steuerung* führt ein System auch im Fall eines Fehlers in einen Zustand, der als sicher angesehen wird. Dies geschieht bereits allein durch die Gegenwart der Steuerung. Ein weiterer Ansatz ist hierbei die Einführung von Interaktionssperren. Diese führen dazu, dass die Kommunikation unterbrochen wird, sodass der sichere Zustand erhalten bleibt oder wiederhergestellt wird [Lev11].

Bei *aktiven Steuerungen* reicht die Präsenz der Steuerung alleine noch nicht aus, um die Sicherheit zu gewährleisten. Dafür sind die folgenden Maßnahmen zur Umsetzung notwendig. Zum einen müssen Variablen oder Zustände überwacht werden. Dadurch ist sichergestellt, dass das System Gegenmaßnahmen einleitet, wenn ein gefährlicher Wert erreicht wird. Dieser Wert wird als solcher durch eine Interpretation der gemessenen Variable definiert. Durch diese Einstufung muss

eine Gegenmaßnahme getroffen werden, die dafür Sorge trägt, einen Loss zu verhindern. In der Umsetzung wird das System auf einen Zustand zurückgesetzt, oder in einen Zustand transferiert, der als sicher eingestuft wird und somit einen Unfall verhindert [Lev11].

Die Constraints werden anschließend einem Controller zugeteilt, der für die korrekte Umsetzung Sorge trägt. Diese ist durch eine festgehaltene Verantwortung definiert [Lev11].

2.1.2 Hierarchische Kontrollstruktur

Ein System wird in STAMP als eine Hierarchie angesehen, das Strukturen auf verschiedenen Ebenen besitzt. Diese interagieren miteinander. Dabei gibt die höherliegende Ebene vor, wie sich die niedrigeren Abstraktionsebenen zu verhalten haben. In der Umsetzung geschieht dies mittels den zuvor festgelegten Safety Constraints. Im Umkehrschluss bedeutet dies ebenfalls, dass nicht existierende Safety Constraints dazu führen können, dass Beeinflussungen bezüglich des Verhaltens von einer höheren nicht an eine niedrigere Ebene weiter gegeben werden können [Lev11].

Daraus resultierend können Unfälle entstehen. Denn nicht weitergegebene oder inkorrekte Safety Constraints stellen ein Sicherheitsrisiko dar. Um zu analysieren, welche Ursachen den Unfällen zu Grunde liegen, wird die *“hierarchial control structure”* genutzt [Lev11].

Die Abbildung der Kontrollstruktur kann bei komplexen Systemen sehr unübersichtlich sein. Um dies zu verhindern, ist es meist hilfreich, verschiedene Abstraktionslevel der Kontrollstruktur einzeln abzubilden. Ebenfalls kann es sein, dass nur Teile dieser Struktur betrachtet werden sollen. Hierzu werden die nicht zu betrachtenden Teile als äußere Einflüsse wahrgenommen, die in den zu betrachtenden Teil hinein- und herausgegeben werden können. Somit werden relevante Teile für die Analyse gesondert betrachtet [Lev11].

Zum Ablauf der Analyse gilt es zu beachten, zunächst die Hazards zu identifizieren. Dies geschieht auf Systemebene. Um nun anschließend die Safety Constraints aufzustellen, sollte die Kontrollstruktur von oben nach unten aufgebaut werden. Dadurch können für einzelne Teile die Constraints festgelegt werden [Lev11].

Dabei gilt es immer zu beachten, dass in den heutigen Zeiten Wettbewerb, sich ändernde Technologien oder auch Gesetzesänderungen dazu führen, dass Safety Constraints neu bewertet werden müssen [Lev11].

2.1.3 Prozessmodell

Das Prozessmodell sorgt dafür, dass die zuvor beschriebenen Safety Constraints umgesetzt werden. Dieses setzt der Controller innerhalb der Kontrollstruktur um. Benötigt wird dieses um einen Prozess zu steuern. Im Allgemeinen gibt es zwei Arten von Controllern. Zum einen Menschen und zum anderen Maschinen. Diese handeln nach einem zuvor definierten Modell. Dabei kann es beliebig viele Variablen innerhalb dieses Modells geben [Lev11].

Eine Variable besitzt zu jedem Zeitpunkt einen Wert, der für den Controller steuerbar ist. Um die Prozesse zu steuern, braucht ein Controller diverse Modelle. Zum einen können dies mentale Modelle sein, zum anderen Prozessmodelle. In diesen Modellen werden ebenfalls die Zusammenhänge und Zustände der Prozessvariablen, mit denen interagiert wird, festgehalten[Lev11].

Die Ursache für Unfälle, bei denen Komponenten untereinander interagieren, sind häufig Fehler im Prozessmodell. Der Fehler entsteht dann, wenn das erstellte Prozessmodell nicht mit dem ablaufenden Prozess übereinstimmt. Dabei können nach N. Levenson [Lev11, Seite 88] folgende Probleme auftreten:

- "Die gegebenen Steuerbefehle sind entweder falsch oder unsicher."
- "Maßnahmen, um die Kontrolle zu erhalten, damit die Sicherheit gegeben ist, wurden nicht vorgesehen."
- "Korrekte Steuerbefehle werden zum falschen Zeitpunkt gegeben oder waren zum falschen Zeitpunkt bereit. Diese können sowohl zu früh als auch zu spät gegeben werden."
- "Die Steuerbefehle werden zu früh gestoppt oder zu lange gegeben."

Prozessmodelle sind also von besonderer Wichtigkeit für STAMP, um ein Verständnis zu erlangen, wie Unfälle zu Stande gekommen sein können. Sowie Erkenntnisse darüber zu gewinnen, weshalb es eine unzureichende Kontrolle über den zu kontrollierenden Prozess zum Zeitpunkt des Unfalls gab. Somit werden sie also gebraucht, um ein sicheres System zu entwerfen [Lev11].

Aufgeteilt wird der STAMP-Analyseprozess in zwei Teile. Zum einen in den CAST-Prozess und zum anderen in den STPA-Prozess. Diese beiden Modelle werden im Folgenden erklärt.

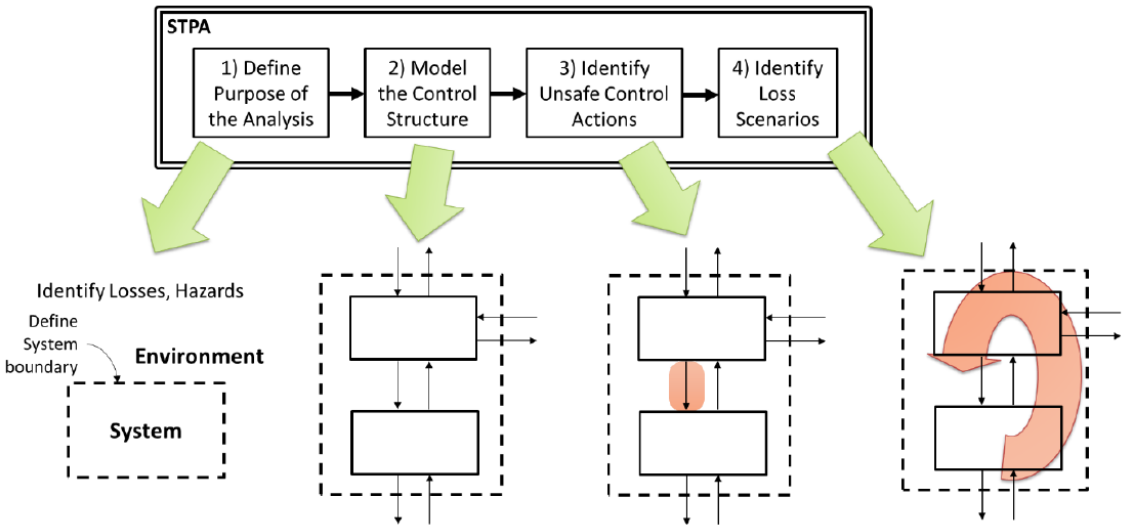

Abbildung 2.1: Ablauf von STPA [LT18].

2.2 System-Theoretic Process Analysis(STPA)

Bei STPA handelt es sich um eine neuere Hazardanalyse. Um zu evaluieren, wie effektiv STPA gegenüber den bereits existierenden Hazardanalysen (wie Fault Tree Analysis (FTA), Failure Modes and Effects Criticality Analysis (FMECA),...) ist, wurde eine Studie durchgeführt. Dabei ergaben die Ergebnisse, dass STPA meist mehr Probleme identifiziert, wie die bereits etablierten Hazardanalysen [LT18]. Dies resultiert daraus, dass STPA für komplexere Systeme genutzt werden kann, bei denen existierende Hazardanalysen an ihre Grenzen stoßen. Sie werden heute in der Industrie in diversen Bereichen ausführlich genutzt [AW14]. Außerdem stellte sich STPA als weniger zeitaufwendig und ressourcenschonender heraus. [LT18]

STPA wird aufgrund der Vollständigkeit erwähnt, da es ebenfalls in der Einzelplatzanwendung vorhanden ist. STPA besteht aus vier Schritten, welche in Abbildung 2.1 zu sehen sind.

Im ersten Schritt werden zunächst die grundlegenden Fragen beantwortet. Dabei geht es vor allem darum, die Losses zu identifizieren. Im Gegensatz zu den existierenden Hazardanalysen wird hier mehr betrachtet, als der Verlust von Menschenleben. Im zweiten Schritt wird die Kontrollstruktur modelliert. Dabei werden die Zusammenhänge zwischen den im System bestehenden Komponenten abgebildet. Hierbei handelt es sich um einen iterativen Prozess, in dem zunächst die Kontrollstruktur abstrakt gebaut und dann verfeinert wird. Im dritten Schritt werden unsichere Kontrollaktionen festgehalten. Dabei wird betrachtet, wie die in Schritt zwei definierten Komponenten der Kontrollstruktur dazu führen können, dass der Loss (definiert in Schritt eins) auftreten konnte. Anschließend werden sie verwendet, um Einschränkungen und auch Anforderungen auf funktionaler Ebene zu definieren. Im vierten und letzten Schritt werden Szenarien erstellt. Dabei werden die unsicheren Kontrollaktionen analysiert, um die Ursachen zu finden, in denen sie auftreten können. Die erstellten Szenarien erklären, weshalb Verluste aufgetreten sind. Dabei wird betrachtet, was unsichere

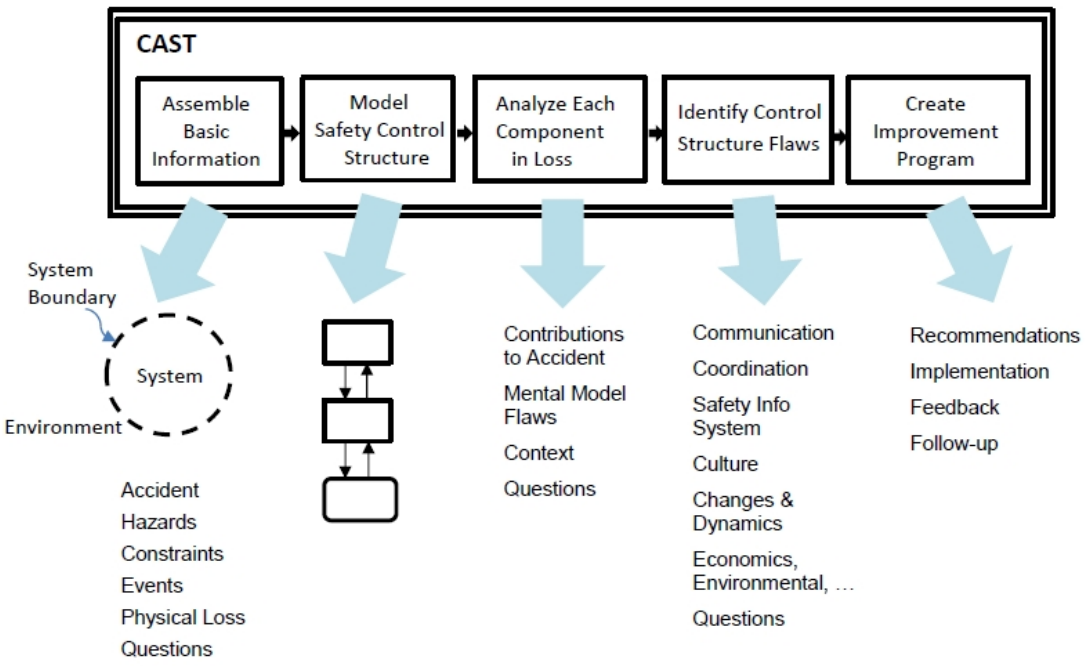

Abbildung 2.2: Ablauf von CAST [Lev19].

Kontrollaktionen verursacht haben könnte oder weshalb sichere Kontrollaktionen nicht korrekt ausgeführt oder befolgt wurden. Durch die Szenarien können neue Maßnahmen abgeleitet werden, um mit diesen die Sicherheit im System zu erhöhen [LT18].

2.3 Causal Analysis Based on System Theory (CAST)

In CAST geht es, wie in STPA, darum, die Ursache für das unerwünschte Ereignis zu finden. Das Ziel hierbei ist es, solche Ereignisse in Zukunft zu vermeiden und somit aus den Unfällen zu lernen. Dabei geht es im Speziellen nicht darum, nur die Unfallursache zu finden, sondern im Ablauf des Prozesses so viele Systemrisiken wie möglich zu erkennen, um aus diesen zu lernen. Am einfachsten ist es, einen Schuldigen für den Unfall zu finden, um bis zum nächsten Unfall dem Alltagsgeschäft wieder nachgehen zu können. Um dies zu vermeiden, sollte der CAST-Prozess genutzt werden [Lev19]. Dieser wird nun im Folgenden genauer erläutert.

Wie in Abbildung 2.2 zu sehen ist wird CAST, laut N. Levenson, in die folgenden fünf Schritte gegliedert [Lev19, Seite 34]:

1. “Das Sammeln von Basisinformationen.”
2. “Modellierung der Sicherheitskontrollstruktur.”
3. “Analysieren jeder Komponente in Bezug auf den Verlust.”
4. “Identifizierung der Mängel innerhalb der Kontrollstruktur.”
5. “Erstellung eines Planes zur Verbesserung des Programms.”

2.3.1 Schritt 1 - Basisinformationen

Zunächst werden alle Informationen darüber gesammelt, um welche Art von System es sich handelt und wie dieses konkret aussieht. Die System-Hazards halten fest, wie es zu den Losses kam. Anhand der identifizierten System-Hazards werden die System-Constraints identifiziert und festgelegt. Anschließend sollen die Ereignisse sachlich festgehalten werden. Hierbei ist darauf zu achten, dass keine Schuldigen gesucht oder Rückschlüsse gezogen werden. Zudem sollen alle Fragen festgehalten werden, die im Laufe der Analyse aufkommen. In Bezug auf diese Fragen sollte ebenfalls beachtet werden, dass diese am Anfang der Analyse zunächst sehr allgemein formuliert sind und im Laufe der Zeit verfeinert werden können [Lev19].

2.3.2 Schritt 2 - Sicherheitskontrollstruktur

Ziel ist es, herauszufinden, weshalb die Kontrollstruktur, die die Hazards verhindern sollte, dies nicht konnte und wie sie verbessert werden kann, um eine sichere Kontrollstruktur zu erreichen. Diese bildet die Grundlage für die kommenden Schritte. Um eine möglichst gute Kontrollstruktur zu erreichen, lohnt es sich zunächst eine möglichst abstrakte Kontrollstruktur zu zeichnen und zu einem späteren Zeitpunkt Verfeinerungen vorzunehmen. Ein anderer Ansatz ist es, bei den existierenden Hazards zu beginnen. Durch die Betrachtung dieser lassen sich Controller ableiten, die zur Verhinderung der Hazards notwendig sind. Man kann ebenfalls eine Kombination dieser beiden Ansätze verfolgen. Es ist möglich, dass es bereits ähnliche Vorfälle in der Vergangenheit gab. Bei diesen wurden eventuell Änderungen in der Kontrollstruktur vorgenommen, die zum Zeitpunkt des Unfalls aber nicht wirksam waren. Bei diesen Änderungen sollte nun analysiert werden, weshalb diese nicht ausreichend waren und wie sie verbessert werden müssen, um eine sichere Kontrollstruktur zu erreichen [Lev19]. Im Laufe der Analyse stellt sich der Analyst Fragen. Resultierend aus diesen ergeben sich im Laufe der Analyse weitere *“constraints and controlls”* [Lev19, Seite 47], die notwendig sind, um die Sicherheit herzustellen. Dabei sollte beachtet werden, dass man die Kontrollstruktur nicht zu detailliert beginnen sollte, um keine wichtigen Teile zu übersehen [Lev19].

Wenn die Liste der Controller begonnen wurde, wird jedem eine Responsibility zugeteilt. Bei den Controllern kann es sich sowohl um einzelne Personen, als auch um Organisationen handeln. Die Analyse über die Wirksamkeit der Responsibility ist ein wichtiger Bestandteil des CAST-Prozesses. Dabei wird betrachtet, ob diese ausreichend ist und welche Änderungen ansonsten notwendig sind. Dabei ist ebenfalls die Schlussfolgerung zulässig, dass eine Responsibility nicht zu dem ihr zugewiesenen Controller gehört und einem anderen zugeteilt werden muss. Somit müssen dann ebenfalls Teile der Kontrollstruktur überprüft werden. Im Regelfall wird es in einer Analyse sowohl unwirksame, als auch falsch zugewiesene Responsibilities geben [Lev19].

Die Responsiblities werden durch den Analysten mittels Dokumentation der Firmen oder der zuständigen Behörde identifiziert. Außerdem sollten die menschlichen Controller über ihre Responsibilities, sowie die der Anderen befragt werden. Die erfragten Responsibilities sollten hierbei mit jenen übereinstimmen, die in der Theorie festgehalten wurden. Zur Verhinderung von Unfällen ist es also notwendig, das Wissen über das System und somit auch über die Responsibilities in regelmäßigen Abständen aufzufrischen [Lev19].

2.3.3 Schritt 3 - Komponentenanalyse

In der Komponentenanalyse wird jede Komponente und somit jeder Controller, der in Abschnitt 2.3.2 aufgestellten Kontrollstruktur im Detail analysiert. Dabei geht es zunächst darum, welche Rolle dieser gespielt hat. Dann wird eine Erklärung für das jeweilige Verhalten gesucht. Dies beinhaltet auch die Analyse, weshalb der Controller sein Verhalten für das Richtige hielt. Genauso wie in anderen Schritten, können auch in diesem Schritt noch weitere Fragen entstehen, die festgehalten werden. Außerdem geht es auch hier nicht um Schuldzuweisungen, sondern um die Identifizierung und Analyse des Verhaltens einzelner Elemente der Kontrollstruktur. Dabei müssen diverse Schritte für jeden einzelnen Controller nach N. Levenson [Lev19] betrachtet werden: Zunächst muss geklärt werden, welche Responsibility der Controller im Unfall hatte. Anschließend, welche Rolle er im Unfall spielte. Dabei wird betrachtet, ob der Controller für den aufgetreten Loss relevant war. Falls dies der Fall ist, wird ebenfalls dokumentiert wie gehandelt wurde oder aus welchen Gründen sich der Controller für keine Handlung entschieden hat. Dabei kann das Mental- wie auch das Prozessmodell bemängelt, sowie anschließend erklärt werden [Lev19].

Außerdem wird analysiert, weshalb der Controller sein Verhalten für das Richtige hielt. Dabei soll der Fokus auf dem Verhalten und dessen Ursache liegen. Beispiele für fehlerhaftes Verhalten können unter anderem mangelnde Anleitungen für die getätigte Arbeit oder Gründe wie Druck sein. Es ist ebenfalls möglich, dass ein Controller seine Aufgabe missverstanden hat oder die Funktion der kontrollierten Komponente nicht korrekt verstanden wurde. Hierbei entstehen meist weitere Fragen, die wieder festgehalten werden müssen. Wichtig ist in diesem Zusammenhang, sich besonders an der Frage *"Warum?"* festzuhalten und die Schuldfrage unberücksichtigt zu lassen. Es ist ebenfalls möglich, dass Personen, die als menschliche Controller agieren, nicht nach bestem Gewissen gehandelt haben. Dies muss, sofern es der Fall ist, selbstverständlich erkannt werden, tritt aber nach N. Levenson [Lev19] eher selten auf. Es ist logisch, dass das Verhalten des Controllers im Nachhinein inkorrekt erscheint. Wichtig ist aber, herauszufinden, weshalb er es zu diesem Zeitpunkt für die richtige Wahl hielt. Im Regelfall gilt, je komplexer die Systeme sind, desto schwieriger ist das Verhalten im Notfall. Zur Hilfe werden deshalb meist Checklisten genutzt. Für einen Software-Controller werden, wie für einen menschlichen Controller, zunächst Anforderungen erfasst. Wenn Software Controller bei einem Unfall fehlerhaft sind, handelt es sich hierbei meist um ein Missverständnis bezüglich der Anforderung. Die Gründe, die identifiziert wurden, sind ein wichtiger Schritt, um Verbesserungen in Abschnitt 2.3.5 generieren zu können. Um diese möglichst hilfreich zu gestalten, also künftige Unfälle gut zu vermeiden, muss die Rolle der einzelnen Komponenten möglichst ausführlich betrachtet worden sein [Lev19].

2.3.4 Schritt 4 - Mängelidentitfizierung

Als nächstes wird das Gesamtsystem, dass durch die Kontrollstruktur abgebildet wird, analysiert. Dabei werden Faktoren betrachtet, die zum Unfall beigetragen haben. Die zu betrachtenden Themen sind laut N. Levenson [Lev19]:

"Kommunikation und Koordination"[Lev19, Seite 77]: In diesem Abschnitt werden die Kommunikationskanäle zwischen den einzelnen Elementen der Kontrollstruktur analysiert, um herauszufinden, ob die Kommunikationskanäle ausreichend sind. Dabei wird zunächst betrachtet, welche Kommunikationskanäle im Notfall genutzt werden, um dann die Funktionsweise dieser zu überprüfen. Es kann auch sein, dass Kommunikationskanäle wegen einer Wartung außer Betrieb sind. Wichtig

ist, dass der Controller den Notfallplan kennt und weiß, was der korrekte Ablauf ist, also welche Aktionen er ausführen muss. Dabei kann es vorkommen, dass der Controller seine Verantwortung missverstanden hat oder eine ungenügende Rückmeldung der Kontrollstruktur ein Problem darstellt. Ein regelmäßiges Überprüfen und funktionsfähig halten der Kommunikationskanäle kann hierbei Fehler und somit Unfälle vermeiden [Lev19].

"Sicherheitsinformationssystem (SIS)"[Lev19, Seite 77]: In einem SIS werden Informationen zu Gefahren festgehalten. So können diese gespeichert und möglichst einfach weitergegeben werden. Ebenfalls soll eine Dokumentation angefertigt werden. Diese ermöglicht eine Erkennung von Änderungen bereits beim ersten Auftreten. So können Unfälle frühzeitig erkannt werden. Dadurch sollte ebenso sichergestellt sein, dass erstellte Standards auch bezüglich der Sicherheit umgesetzt werden. Auch sollten Unterschiede zwischen den in der Theorie existierenden Modellen und der Praxis im Bezug auf die Abschätzung der Risiken dokumentiert werden. Diese Details sollten alle in der SIS des Unternehmens enthalten sein. CAST hilft dabei, Faktoren mit einzubeziehen, die auf systematischer Ebene zum SIS beitragen können. Auch dies ist ein ständiger Prozess, der zur Verbesserung dient. So wird aus den aufgetretenen Ereignissen gelernt und das SIS stetig verbessert. Ein Unfallbericht ist für diesen Abschnitt ein wichtiges Dokument, um Informationen zu erhalten. Denn das Vorhandensein eines SIS reicht noch nicht zwangsläufig aus. Ein gut konzipiertes SIS, kann Unfälle verhindern. Die Probleme des SIS liegt oft in den großen Mengen an Daten, die analysiert werden müssen, um die Trends erkennen zu können. Ein weiteres Problem stellen oft die verwendeten *"retrieval and dissemination mechanisms"* [Lev19, Seite 82] dar. Bei der Analyse sollten sich die Fragen gestellt werden, ob die Informationen, die für die Erkennung des Unfalls notwendig waren, nicht erfasst wurden, verloren gegangen sind, oder nicht direkt verwendet werden konnten. Das Sicherheitsmanagment profitiert von den im SIS festgehaltenen Informationen und kann diese nutzen [Lev19].

"Sicherheitsmanagement"[Lev19, Seite 77]: In der Theorie sollten sich das Sicherheitsmanagement und die in Abschnitt 2.3.2 aufgestellte Kontrollstruktur entsprechen. Allerdings ist es möglich, dass in der Industrie wichtige Controller, die für den Unfall von Bedeutung waren, im Sicherheitsmanagement nicht enthalten sind. Bei der Analyse des Sicherheitsmanagements sollte überprüft werden, ob die Kontrollstruktur wirksam ist. Es muss berücksichtigt werden, dass die Safety Constraints, die für das Verhalten der Organisation notwendig sind, in der Kontrollstruktur bedacht werden. Es ist wichtig, dass die Controller klare Aufgaben zugeteilt bekommen, für welche sie verantwortlich sind. Dabei muss ebenfalls die Zuständigkeit explizit benannt werden. Denn die klare Verteilung der Aufgaben geschieht, um Gefahren und auch Unfällen vorzubeugen. Ebenfalls sollten die Unterschiede zwischen der dokumentierten Kontrollstruktur und der zum Zeitpunkt des Unfalls genutzten verglichen werden. Dabei wird betrachtet, ob es Zwischenfälle gab, die darauf hingewiesen haben, dass Fehler vorlagen. Ebenfalls wird analysiert, wie mit diesen Zwischenfällen umgegangen und wie und wohin diese gemeldet wurden. Es ist wichtig, ob nur Schuld zugewiesen, oder eine Analyse der Kontrollstruktur durchgeführt wurde, um weitere Zwischenfälle effektiv zu verhindern. Es sollte ebenfalls sicher gestellt sein, dass die Änderungen hierbei beaufsichtigt werden. Das Sicherheitsmanagment einer Firma wird von außerhalb überprüft. Meist werden hier staatliche Aufsichtsbehörden oder auch gewerbliche Behörden/Organisationen eingesetzt [Lev19].

Die *"Sicherheitskultur"*[Lev19, Seite 77] ist ebenfalls ein wichtiger Bestandteil, durch den das SIS und das Sicherheitsmanagement beeinflusst werden. Eine Sicherheitskultur ist immer vorhanden. Deshalb sollte analysiert werden, welche Auswirkungen sie auf den Unfall hat. Es gibt diverse Kulturen die in N. Levensons Handbuch [Lev19] beschrieben werden. Die Sicherheitskultur wird von

der Leitung des Unternehmens festgelegt und sollte in schriftlicher Form als Sicherheitsphilosophie festgehalten werden. Geschieht dies nicht, ist bereits eine große Sicherheitslücke ausfindig gemacht. Die schriftliche Festhaltung der Richtlinien ist für die Sicherheit aber noch nicht ausreichend. Es muss überprüft werden, wie die Sicherheitskultur bei den Mitarbeitern und beim Management ankommt, ob sie auch umgesetzt wird und wie die Richtlinien an die Mitarbeiter vermittelt werden. Es ist möglich, dass das Unternehmen die Richtlinien niedergeschrieben hat aber das tatsächliche Verhalten davon stark abweicht. Um dies zu überprüfen, können Mitarbeiterbefragungen verwendet werden. Das Wichtigste im Bezug auf die Umsetzung der Sicherheitskultur ist das Management, denn die Mitarbeiter müssen die Sicherheit vor andere Faktoren stellen können, wie zum Beispiel Kosten und sich dabei der Unterstützung des Managements und Unternehmens sicher sein [Lev19].

"Veränderungen und Dynamiken" können im *"Laufe der Zeit im System und in der Umgebung"*[Lev19, Seite 77] auftreten. Hierbei können geplante aber auch ungeplante Änderungen auftreten, die Unfälle verursachen können. Es gibt diverse Stellen, in denen Änderungen durchgeführt werden. Zum einen am System selbst, zum anderen kann es aber auch in der Interaktion mit der Systemumgebung zu Änderungen kommen. Wenn die Änderungen, die durchgeführt werden, geplant sind, sollte durch das Management genau überprüft werden, wie sich die Änderungen auswirken. Dann muss angemessen darauf reagiert werden, denn nur so lassen sich Unfälle vermeiden. Änderungen eines Arbeitsablaufs führen beispielsweise dazu, dass entsprechende Arbeitsanweisungen angepasst werden müssen. Sollte dies nicht geschehen und die veraltete Anweisung sogar verworfen werden, stellt dies ein großes Unfallrisiko dar. Es kann aber ebenfalls Änderungen geben, die nicht geplant sind. Diese Änderungen müssen möglichst früh erkannt werden. Dies geschieht durch gewisse Indikatoren, die durch die Annahmen mittels Analysen regelmäßig zu überprüfen sind, um so weiterhin die Sicherheit zu gewährleisten. Änderungen können beispielsweise bei Produkten von externen Herstellern auftreten, die in das System integriert sind. Treten diese Änderungen ohne Information des Nutzers auf, muss dieses im System frühzeitig erkannt werden [Lev19].

Als nächstes werden *"interne und externe wirtschaftliche und verwandte Faktoren in der Systemumgebung"*[Lev19, Seite 77] betrachtet. Bei diesen Faktoren sollte besonderer Wert darauf gelegt werden, die Problemerkennung frühzeitig zu beginnen. Denn die Vernachlässigung oder Verringerung der Sicherheitsvorkehrungen kann zu einem Unfall beitragen. Dies sollte trotz wirtschaftlicher Veränderungen berücksichtigt werden. Ein weiteres Problem kann die Verschmelzung zwischen Wohnsiedlungen und Firmengeländen sein. Denn mit vielen Arbeitsplätzen geht die Verstädterung in den Gebieten um die Firma einher. Eine besondere Gefahr existiert, wenn die bestehenden Sicherheitsrisiken nicht beachtet werden, um die Attraktivität des Wohnraums in der Nähe des Arbeitsplatzes zu erhöhen. Dies kann bei einem Unfall tragisch sein. Dabei handelt es sich meist um Änderungen, die im Laufe der Zeit geschehen sind und die meist noch keine Berücksichtigung finden. Die Wettbewerbsfähigkeit ist für eine Firma ein wichtiges Ziel für das Sicherheitsanstrengungen eventuell verringert werden. Weitere Gründe können Verhandlungen sein, durch die eine Ablenkung vorhanden ist. Zu Lasten der Sicherheit gehen beispielsweise ebenfalls Anstrengungen des Managements zur Verbesserung der Arbeitsmoral ihrer Mitarbeiter. Ein großes Problem, wenn Software involviert ist, ist dass Automatisierungen als unfehlbar angesehen werden oder zuverlässiger erscheinen, als wenn diese Aufgabe durch einen Menschen übernommen wird. Es sollte hierbei überprüft werden, ob die Software keine Wechselwirkungen oder andere Änderungen im Kontext des Gesamtsystems mit sich bringt, die so vorher noch nicht existiert hatten. Diese können zu neuen Problemen und somit zu neuen Ursachen für Unfälle führen [Lev19].

2.3.5 Schritt 5 - Verbesserungen

Als letztes werden nun Verbesserungen aus den Erkenntnissen der letzten Schritte herausgearbeitet und dokumentiert. Um ähnliche Unfälle in Zukunft verhindern zu können, sollten diese Empfehlungen umgesetzt werden. Diesen Prozess unterstützend, kann es ebenfalls hilfreich sein, Verbesserungen mittels Programmen umzusetzen. Die Empfehlungen werden hierbei aus den Erkenntnissen der vorherigen Schritte erarbeitet. Meist ist die Anzahl der gesammelten Empfehlungen in der CAST Analyse sehr hoch. Daher empfiehlt es sich, die Empfehlungen in verschiedene Zeiträume aufzuteilen. Es gibt Empfehlungen, die sich direkt umsetzen lassen, andere hingegen erst später und im letzten Zeitraum der sich festlegen lässt, braucht es behördliche Änderungen oder neue Gesetze. Bei der Aufstellung von Empfehlungen kann die Einteilung in Zeiträume schwierig sein. Wenn die Einteilung nicht eindeutig zuzuordnen ist, wird die Empfehlung auf langfristig gesetzt, anstatt sie nicht zu dokumentieren [Lev19].

Um sicherstellen zu können, dass die Empfehlungen umgesetzt werden, muss die Verantwortung für diese klar zugewiesen und deren Umsetzung überprüft werden. Wenn ein Controller verändert wurde, sollte ihm, mittels eines Programms, die Möglichkeit gegeben werden, eine Rückmeldung zu geben, um zu ermitteln, ob die Änderungen hilfreich waren[Lev19].

Bei weiteren Unfällen, zu denen bereits eine CAST-Analyse existiert, sollte zuerst überprüft werden, ob die damals erstellten Empfehlungen umgesetzt wurden und ob diese wirksam sind. Im Allgemeinen ist die Verbesserung eines System ein kontinuierlicher Prozess [Lev19].

2.4 Wieso STPA und CAST in einer gemeinsamen Anwendung verwendet werden sollten

STPA und CAST basieren beide auf STAMP. STPA ist eine Hazard Analyse und wird somit genutzt um ein System zu analysieren, bevor ein Unfall aufgetreten ist. Dabei werden alle theoretisch möglichen Unfallszenarien analysiert und so versucht diese zu vermeiden. Dies steht im Gegensatz zu CAST in der ein konkret aufgetretenes Unfallszenario analysiert wird. STPA kann bereits zur Analyse genutzt werden, wenn man sich erst in der Konzeptentwicklung befindet. Auf Grundlage des STPA-Prozesses wird versucht, das System, von Anfang an, möglichst sicher zu gestalten. Die CAST-Analysen von bereits vergangenen Unfällen können zur Unterstützung des Prozesses besonders hilfreich sein. Dabei wird das aufgetretene Unfallszenario in STPA berücksichtigt. Dadurch sollen, wenn möglich, weitere Verluste verhindert werden [Lev19]. Somit ist es in jedem Fall sinnvoll, eine Anwendung zu schaffen, die CAST und STPA verbindet und die Möglichkeit schafft, beide Analysen im selben Programm öffnen und bearbeiten zu können.

Wie in STPA bereits in Abschnitt 2.2 beschrieben, werden in Schritt 4 Szenarien dokumentiert und analysiert. Diese enthalten im Idealfall alle möglichen Unfallszenarien. Wenn nun in einem System ein Unfall aufgetreten ist und dieser mittels des CAST-Prozess analysiert wird, sollte zunächst überprüft werden, ob zu diesem bereits eine STPA-Analyse existiert. Falls dies der Fall ist, sollten die dort festgehaltenen Szenarien betrachtet werden. Wenn dieses Szenario dokumentiert wurde, stellt sich die Frage, weshalb es keine oder eine nicht wirksame Gegenmaßnahme gab. Es ist aber auch möglich, dass es zwischen der Theorie und dem laufenden Betrieb eine Diskrepanz gibt. Dann ist das Szenario nicht in STPA zu finden. Als letztes gibt es außerdem die Möglichkeit, dass das

Unfallszenario nie in Betracht gezogen wurde. Abschließend lässt sich hiermit aber sagen, dass es für STPA, wie auch für CAST hilfreich ist, die Projekte der anderen Analyse in der selben Anwendung öffnen zu können [Lev19].

2.4.1 bereits existierendes Tool

Es wurde eine Suche durchgeführt, um die bereits existierenden CAST-Anwendungen zu finden. Dabei wurde lediglich eine Software gefunden. Hierbei handelt es sich um die an der Universität Stuttgart entwickelte Software XSTAMPP. XSTAMPP ist eine Software, die mit verschiedenen Plugins erweitert werden kann [AW16]. Eines dieser Plugins ist A-CAST, welches die Funktionalitäten des CAST-Prozesses beinhaltet. Der abgebildete CAST-Prozess in dieser Software besteht aus drei Schritten [Roo15] und wurde vor der Veröffentlichung des CAST Handbuches [Lev19] von N. Levenson implementiert.

3 Verwandte Arbeiten

"Engineering a Safer World - Systems Thinking Applied to Safety" [Lev11] wurde im Jahr 2011 von N. Levenson veröffentlicht und bildet die Grundlage für alle folgenden Arbeiten, die zur CAST-Analyse geschrieben wurden. In der Veröffentlichung wird der CAST-Prozess, sowie der STPA-Prozess und STAMP im Allgemeinen beschrieben.

Auf Grundlage dieses Buches entstand im Folgenden die Umsetzung des Eclipse Plugin XSTAMPP an der Universität Stuttgart. In einer Veröffentlichung [AW15] hierzu wurde der Ausblick gegeben, CAST in das Tool zu integrieren. Auf Grundlage dieses Ausblickes wurde eine Bachelorarbeit [Roo15] zur Implementierung des A-CAST-Plugins für das XSTAMPP Eclipse Plugin erstellt.

Anschließend wurde das A-CAST-Plugin in der Veröffentlichung von Wagner und Abdulkhaleq [AW16] vorgestellt.

Im Weiteren wurde 2016 eine Fachstudie [BSS17] veröffentlicht, die den Zugunfall von Bad Aibling betrachtet. Dabei wurde mittels CAST der Unfall analysiert.

2019 wurde, mittels eines Handbuches, eine Erweiterung des CAST-Prozesses vorgenommen. In diesem wurde die Theorie des gesamten CAST-Prozess ausführlich beschrieben [Lev19]. Dabei werden alle Schritte umfassend dargestellt, sowie an einem Beispiel erklärt.

Anschließend wurde das Handbuch noch durch eine Veröffentlichung ergänzt [LSM19].

Im Handbuch wird der Shell Moerdijik Unfall genutzt, um die einzelnen Schritte im Detail zu erklären. Da hierbei zu ausführliche Erklärungen nicht förderlich für das Verständnis des CAST-Prozesses sind, wurde der gesamte Analyseprozess anhand dieses Beispiels in einer eigenen Veröffentlichung festgehalten [Lev17].

XSTAMPP wurde anschließend um eine Webanwendung erweitert (XSTAMPP4.1). Nun wird diese Anwendung um eine Einzelplatzanwendung erweitert. Die Umsetzung ist auf drei Bachelorarbeiten verteilt worden. In dieser Bachelorarbeit wird das CAST-Modul implementiert. Die Implementierung und Umsetzung des STPA-Moduls[Gre20] und das Modell-Checking, sowie die Verfeinerung der UCA's [Hel20] wurden in anderen Bachelorarbeiten umgesetzt.

4 Anforderungsanalyse

Aufbauend auf den Grundlagen wurde im Folgenden eine Anforderungsanalyse durchgeführt, um die relevanten Systemteile zu erkennen, zu konzipieren und umzusetzen. Deshalb wird der CAST-Prozess in der Einzelplatzanwendung auf den neusten Erkenntnissen des Handbuches [Lev19] aufgebaut, sowie durch die Bachelorarbeit der Transferierung der Webanwendung XSTAMPP4.1 (STPA)[Gre20] in der gemeinsamen Einzelplatzanwendung beeinflusst. Die Anforderungsanalyse bildet die Grundlage für die Implementierung des Systems.

4.1 Ableitungen aus dem Handbuch - funktionale Anforderungen

Zuerst werden die Anforderungen festgehalten, die sich aus dem CAST-Handbuch [Lev19], sowie den Shell Moerdijik Unfall [Lev17] ableiten lassen.

4.1.1 Schritt 1

In Schritt 1 werden zunächst die Basisinformationen gesammelt, die für den Unfall von besonderer Relevanz sind. Diese sind in Abbildung 4.1 zu sehen. Zuerst beschreibt der Analyst das System in aller Ausführlichkeit. Dabei wird in einem Text festgehalten, um welche Art von System es sich handelt, sowie eine allgemeine Beschreibung des Systems gegeben. Im nächsten Schritt wird die Unfallbeschreibung festgehalten. Dabei wird der Unfall im Detail beschrieben, sowie die Dokumente verlinkt, die die Analyse unterstützen, wie beispielsweise der Unfallbericht. Danach werden die Hazards festgehalten. Es handelt sich dabei um die Gefährdungen, die zu Verlusten führen. Dazu wird der Name und die Beschreibung festgehalten. Anschließend werden die Constraints betrachtet. Sie werden genutzt, um die Gefahren, die auftreten können zu vermeiden. Dabei wird der Name angegeben, sowie die Beschreibung und ein Eingabefeld, in das der zu vermeidende Hazard eingetragen wird. Als letztes werden in Schritt 1 alle Ereignisse des Unfallherganges, beginnend bei einem sinnvollen frühen Zeitpunkt bis zum Unfallzeitpunkt, festgehalten [Lev19].

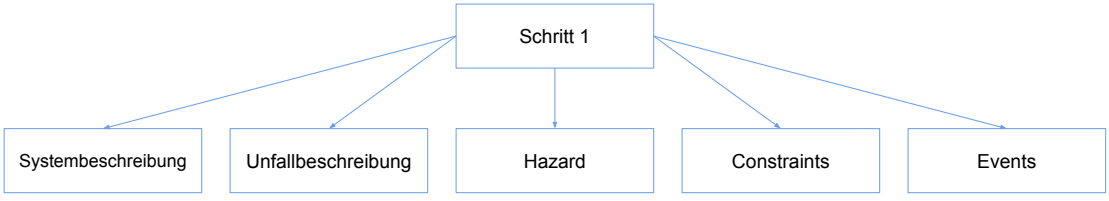

Abbildung 4.1: Anforderungsanalyse von Schritt 1.

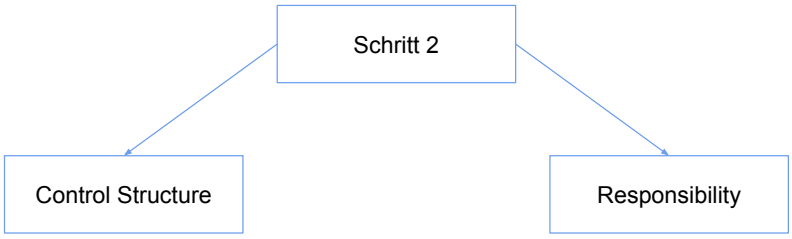

Abbildung 4.2: Anforderungsanalyse von Schritt 2.

4.1.2 Schritt 2

In Schritt 2 wird die Kontrollstruktur erstellt, sowie die Responsibility der einzelnen Elemente dokumentiert(Abbildung 4.2). Bei der erstellten Kontrollstruktur handelt es sich um die zum Zeitpunkt des Unfalls aktuell existierende Struktur. Diese kann von der in der Theorie abweichen. Zur Modellierung werden Boxen und Pfeile genutzt, die die Verbindungen und Prozesse darstellen, welche im System stattfinden. Anschließend werden die Responsibilities festgehalten. Dazu wird der Name, sowie eine Beschreibung erfasst. Ebenfalls wird dokumentiert, welcher Controller die Verantwortung trägt, sowie die Safety Constraints die durchgesetzt werden [Lev19].

4.1.3 Schritt 3

Als nächste wird in Schritt 3 die Rolle im Unfall festgehalten (Abbildung 4.3) Hier hält der Analyst fest, welche Rolle der Controller im Unfall gespielt hat. Dazu wird erfasst, um welchen Controller es sich handelt und ein Name vergeben. Danach sollte dokumentiert werden, ob der Controller am Unfall beteiligt war und wie er gehandelt hat. Außerdem wie das Verhalten des Controllers erklärt werden kann. Abschließend wird konkretisiert, ob und welche Fehler im Mentalmodell und/oder Prozessmodell vorlagen [Lev19].

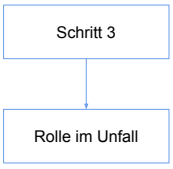

Abbildung 4.3: Anforderungsanalyse von Schritt 3.

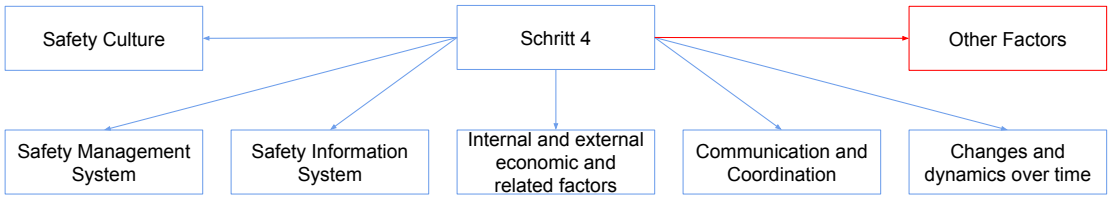

Abbildung 4.4: Anforderungsanalyse von Schritt 4.

4.1.4 Schritt 4

Anschließend wird in Schritt 4 die Kontrollstruktur als Ganzes betrachtet. Dies steht im Gegensatz zu dem in Abschnitt 4.1.3 beschriebenen Konzept. Der Fokus sollte hierbei darauf liegen, die Zusammenhänge zu verstehen und herauszufinden, wie das Zusammenspiel der einzelnen Komponenten die Kontrollstruktur beeinflusst [Lev19]. Nach N. Levenson müssen hierbei 6 Kategorien betrachtet werden [Lev19, Seite 77]:

- “Sicherheitskultur”
- “Design des Sicherheitsmanagementsystems”
- “Sicherheitsinformationsystem”
- “Interne und externe wirtschaftliche und verwandte Faktoren”
- “Kommunikation und Koordination”
- “Änderungen und Dynamiken im Laufe der Zeit”

Außerdem wurde, wie in Abbildung 4.4 rot gekennzeichnet, eine zusätzliche Kategorie “Andere Faktoren” eingefügt. Dort können weitere Erkenntnisse eingetragen werden, die keiner anderen Kategorie zugeordnet werden können.

Die Informationen aller sieben Kategorien werden in Textform festgehalten. Zudem können relevante Dokumente verlinkt werden.

Ergänzt werden diese um die Möglichkeit, einzelne Controller genauer zu analysieren. Dadurch können Informationen, die spezifisch für einen Controller aufgefallen sind, dokumentiert werden. Umgesetzt wird dies in allen 6 von N. Levenson [Lev19] festgehaltenen Kategorien. Bei der Kommunikation und Koordination werden aber immer Verbindungen zwischen 2 Controllern betrachtet, zu denen spezifische Information dokumentiert werden.

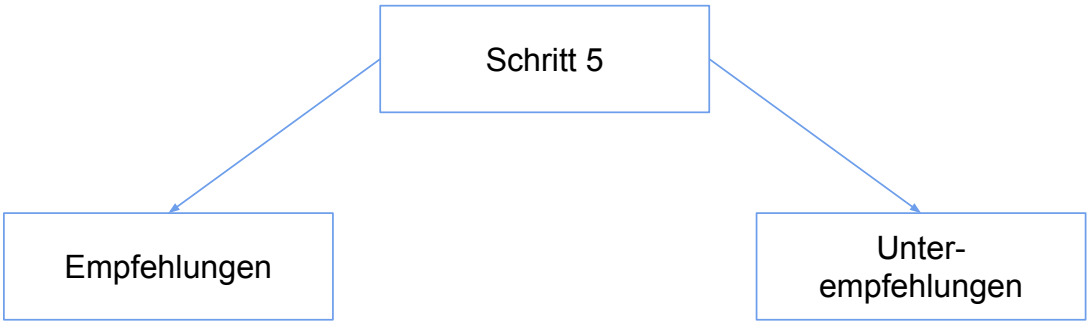

Abbildung 4.5: Anforderungsanalyse von Schritt 5.

4.1.5 Schritt 5

In Schritt 5 werden nun Empfehlungen gesammelt (Abbildung 4.5). Hierbei werden jeweils Name und Beschreibung der Empfehlung festgehalten. Zusätzlich wird über verschiedene Zustände die Umsetzungsdauer erfasst. Dabei gibt es die Möglichkeit, eine Empfehlung in mehrere Unterempfehlungen aufzuteilen. Außerdem wird festgehalten, welche Person oder Firma für die Umsetzung der Empfehlung verantwortlich ist [Lev19].

4.2 Fragen

Fallen in den ersten vier Schritten Fragen an, so können diese in Schritt 5 genutzt werden, um Empfehlungen zu formulieren. Diese Fragen müssen hierzu erfasst und zusammen mit ihren Antworten dokumentiert werden [Lev19]. Daraus resultiert, dass hierfür ebenfalls eine Ansicht notwendig ist. Außerdem wurde hierbei entschieden, dass es sinnvoll ist, bei der Anlegung eines Elements in den ersten vier Schritten die Möglichkeit zu bieten, zu jeder Zeit Fragen erstellen zu können.

4.3 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen ergeben sich aus der Aufgabenstellung der Bachelorarbeit. Das System soll als Einzelplatzanwendung implementiert werden und muss somit auf den gängigen Betriebssystemen: Windows, MacOs und Linux funktionsfähig sein.

Außerdem muss das System in der Lage sein, sowohl CAST als auch STPA zu unterstützen. Da STPA in der Bachelorarbeit [Gre20] die XSTAMPP 4.0 Webanwendung in eine Einzelplatzanwendung transferiert, besteht eine Anforderung darin, die CAST-Oberfläche an ein gemeinsames Design

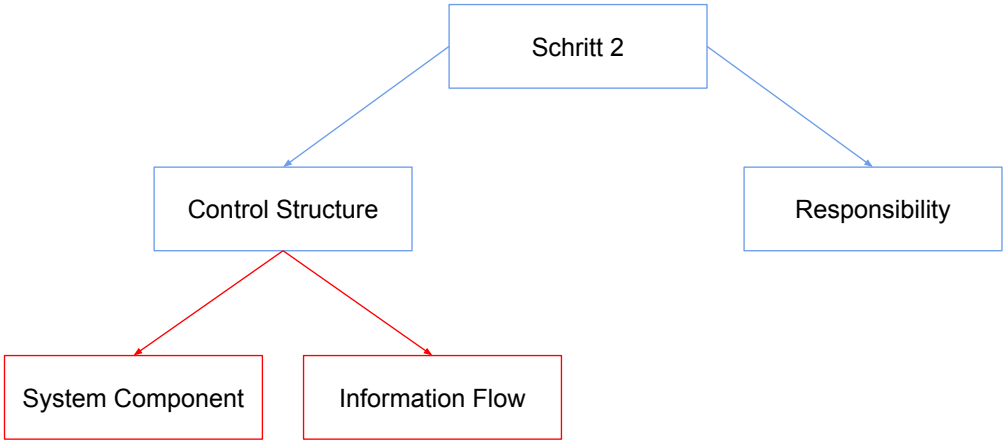

Abbildung 4.6: Erweiterung von Schritt 2.

anzupassen. Dies geschieht, damit der Analyst sowohl STPA, wie auch CAST nach den gleichen Grundprinzipien verwenden kann. Dazu gibt es ebenfalls eine Erweiterung der Funktionalität der Kontrollstruktur, die in Abschnitt 4.4 genauer beschrieben wird.

Eine weitere Anforderung an das System ist, dass Dokumente verlinkt und geöffnet werden können. Dazu müssen Schnittstellen vorhanden sein, die das Öffnen von externen Anwendungen, zum Beispiel mit einem PDF-Reader ermöglichen.

4.4 Ergänzungen durch STPA

Da die Anwendung STPA und CAST vereint und die Anforderungen in Schritt 2 allgemein, im Bezug auf das Design die selben sind, wird für beide die gleiche Kontrollstruktur genutzt. Um dies optimal umzusetzen, wird die Kontrollstruktur, wie in Abbildung 4.6 zu sehen ist, ergänzt.

Die Kontrollstruktur ist wie in STPA aufgebaut, um dem Analysten eine vergleichbare Ansicht zu geben. Somit muss nicht zwischen STPA und CAST Projekten umgedacht werden und das Nutzererlebnis bleibt möglichst einheitlich. Dazu werden die Elemente der Kontrollstruktur in Systemkomponenten und Information-Flow aufgeteilt. Die Systemkomponente beinhaltet die Controller, Aktuatoren, Sensoren, und Controlled-Processes. Der Information-Flow beinhaltet die Controll-Aktionen, das Feedback, den Input und den Output. Hierbei können im STPA-Prozess ebenfalls Elemente hinzugefügt werden, die nicht zwangsläufig in der graphischen Repräsentation vorhanden sind. [1]

Es wurde sich hierbei ebenfalls Gedanken darüber gemacht, ob es sinnvoll ist, Teile der Kontrollstruktur von einer STPA-Analyse in eine CAST-Analyse übernehmen zu können. Es wurde sich dagegen entschieden, da die Kontrollstruktur nicht so ausführlich sein muss, wie in STPA.

[1] siehe Webanwendung: https://web.xstampp.de

Außerdem werden sich weitere Teile im Laufe der Analyse ergeben, da hierbei ebenfalls Fragen generiert werden sollen. Daher ist es sinnvoll sich ausführlich mit jedem Element zu beschäftigen [Lev19].

4.5 Erweiterung des Systems um die Prozessvariablen

Ein Änderung gibt es zu der Oberfläche von STPA innerhalb der Tabelle in Schritt 2. In der System-Komponente, wenn es sich um einen Controller handelt, wird ein weiteres Feld ergänzt. Dabei handelt es sich um die Anlagemöglichkeit der Prozessvariablen mit einem Namensfeld [Lev19].

In Schritt 3 wird dann die Möglichkeit gegeben, jede in Schritt 2 angelegte Prozessvariable zu analysieren. Hierzu wird festgehalten, welchen Wert die Prozessvariable während des Unfalles besitzt. Dabei wird der ebenfalls, falls notwendig, eine Erklärung für dieses Verhalten dokumentiert [Lev19].

5 Implementierung

In diesem Kapitel soll genauer darauf eingegangen werden, welche Technologien verwendet wurden. Außerdem werden die grundlegenden Konzepte, sowie die Architektur der Einzelplatzanwendung beschrieben.

5.1 Verwendete Technologien

Elektron wird als Framework genutzt, in dem die Anwendung laufen soll und besitzt eine MIT-Lizenz. Es wurde gewählt, da es auf allen gängigen Betriebssystemen funktionsfähig ist. Electron basiert auf Chromium und Node.js. Aufgrund dessen können hier alle gängigen Programmiersprachen verwendet werden, die für Webentwicklungen genutzt werden. Daher bietet sich Angular als Programmiersprache an, sowie TypeScript für die Logik [OE20].

Im Frontend wird das Framework Angular verwendet, welches von Google LCC entwickelt und von der Online-Community weiterentwickelt wird und ist mit einer MIT-Lizenz versehen. Somit werden im Frontend die Programmiersprachen TypeScript, sowie HTML genutzt [Goo20].

RxDB (Reactive Database) ist eine Datenbank für JavaScript Anwendungen, die für Electron oder Node.js Anwendungen genutzt werden kann und besitzt eine Apache-2.0 Lizenz. Reactive bedeutet, dass hier sowohl der State abgefragt werden kann, es aber auch möglich ist, auf Änderungen zu subscriben. Somit ist sie gut geeignet, um in Echtzeit mit dem Angular Frontend zusammenzuarbeiten [Mey20]. Die darauf aufbauenden logischen Anwendungen werden in TypeScript geschrieben.

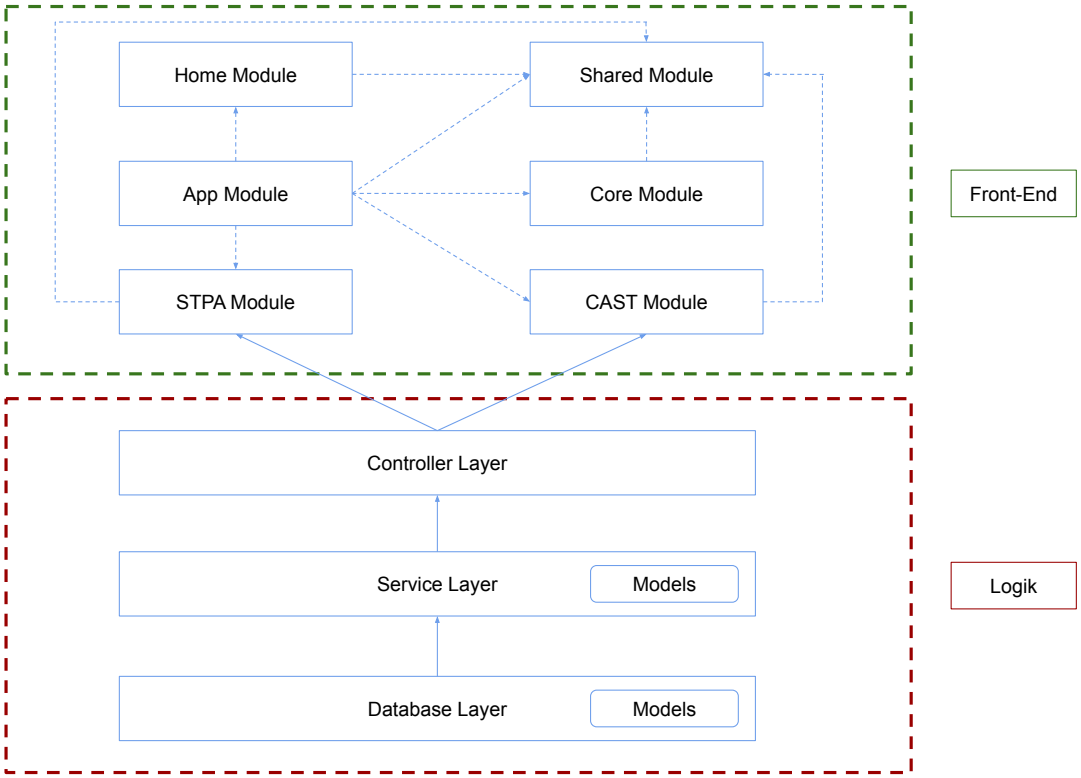

Abbildung 5.1: Architektur der Einzelplatzanwendung (angelehnt an [Pie19]).

5.2 Architektur

In diesem Abschnitt wird die Architektur der Einzelplatzanwendung beschrieben. Diese ist aufgebaut, wie in Abbildung 5.1 zu sehen ist. Inhaltlich wird das System in eine Frontend-Schicht und eine Backend-Schicht geteilt. Die Frontend-Schicht ist in einzelne Module aufgeteilt, das CAST-Module und das STPA-Module, sowie allgemeine Module, die diese erweitern oder die Grundlage bilden. Die Logik-Schicht wird inhaltlich in 3 Teile aufgeteilt. Die Datenbankschemata, die Services, sowie die Controller-Schicht, die die Funktionalitäten für die Frontend Zugriffe bereit stellt. Erklärt wird die Architektur nun beispielhaft anhand eines Hazards.

Listing 5.1 Hazard Controller.

```
@injectable()
export class HazardController {
  constructor(
    @inject(HazardService) private readonly hazardService: HazardService,
    @inject(ConstraintHazardLinkService) private readonly constraintHazardLinkService:
ConstraintHazardLinkService
  ) {}

  public async create(projectId: string): Promise<Hazard> {
    return this.hazardService.create({ ...new Hazard(), projectId });
  }

  public getAll$(projectId: string): Observable<HazardTableModel[]> {
    return this.hazardService.getAllTableModels$(projectId);
  }

  public async update(hazard: Hazard): Promise<Hazard> {
    return this.hazardService.update(hazard);
  }

  public async remove(hazard: Hazard): Promise<boolean> {
    return this.hazardService.remove(hazard);
  }

  public async createConstraintLink(link: ConstraintHazardLink): Promise<ConstraintHazardLink>
 {
    return this.constraintHazardLinkService.create(link, false);
  }

  public async removeConstraintLink(link: ConstraintHazardLink): Promise<boolean> {
    return this.constraintHazardLinkService.remove(link);
  }
}
```

5.2.1 Controller Layer

Der Hazard-Controller ist implementiert, wie in Listing 5.1 zu sehen ist. Dabei werden die Funktionen bereit gestellt, die anschließend im Abschnitt 5.2.9 genutzt werden. Es handelt sich hiermit also um die Verbindung zum Frontend. Der Hazard Controller stellt die Funktionen: *anlegen*, *updaten*, *löschen* bereit. Außerdem gibt die Funktion *getAll$* einen HazardTableModel Array zurück. Dieser erhält alle gespeicherten Tabelleneinträge, die durch das Frontend in der Tabelle dem Anwender angezeigt werden. Falls Verlinkungen zwischen einzelnen Komponenten möglich sind, werden diese mittels Links in dieser Schicht erstellt. Zum Anlegen eines Links werden die ProjektId, sowie die Ids der beteiligten Komponenten verwendet. Im Beispiel aus Listing 5.1 kann eine Verlinkung zwischen einem Hazard und einem Constraint erzeugt werden.

Listing 5.2 Hazard Datenbankschema.

```
export const hazardSchema: RxJsonSchema<Hazard> = {
  title: 'cast: hazard schema',
  description: '',
  version: 0,
  keyCompression: false,
  type: 'object',
  properties: {
    projectId: { type: 'string' },
    id: { type: 'string' },
    label: { type: 'string' },
    description: { type: 'string' },
    name: { type: 'string' },
    state: { type: 'string' },
  },
  indexes: ['projectId', ['projectId', 'id']],
  required: ['projectId', 'id', 'label', 'name', 'description', 'state'],
};
```

5.2.2 Database Layer

Der Datenbankenlayer legt fest, wie die Datenbankeinträge aufgebaut sind. Der Aufbau der Hazardeinträge wird in Listing 5.2 gezeigt. Hierbei werden in den Properties die einzelnen vorhandenen Felder angegeben, mit den verwendeten Typen, sowie die *"indexes"* die genutzt werden, um die Einträge zu durchsuchen. Des Weiteren werden die erforderlichen (*"required"*) Felder benannt. Hier wird ebenfalls, das in Abbildung 5.1 vermerkte Modell (*"Hazard"*) genutzt, dass vorgibt, welche Einträge ein Hazard besitzt.

Listing 5.3 Hazard Service.

```
@injectable()
export class HazardService extends Service<Hazard> {
  constructor(
    @inject(ProjectRepo) projectRepo: ProjectRepo,
    @inject(LastIdRepo) lastIdRepo: LastIdRepo,
    @inject(HazardRepo) private readonly hazardRepo: HazardRepo,
    @inject(ConstraintHazardLinkService) private readonly constraintHazardLinkService:
ConstraintHazardLinkService
  ) {
    super(Hazard, projectRepo, hazardRepo, lastIdRepo);
  }

  public getAllTableModels$(projectId: string): Observable<HazardTableModel[]> {
    return combineLatest([
      this.getAll$(projectId),
      this.constraintHazardLinkService.getConstraintChipMapByHazardIds$(projectId),
    ]).pipe(
      map(([hazards, constraintChipMap]) => {
        return hazards.map(hazard => new HazardTableModel(hazard, constraintChipMap.get(hazard
.id), []));
      })
    );
  }
}
```

5.2.3 Service Layer

Die Controller rufen die Services auf. Diese werden durch eine generische Service Klasse erweitert, die das Hazard Model beinhaltet. Über die generische Service Klasse wurden die *create, update, remove* und *getAll* Funktionen ausgelagert, um so Code-Duplikate bestmöglich vermeiden zu können. Hierzu werden ebenfalls die dazugehörigen Repositories verwendet. Es wird in jedem Service nun nur noch die *getAllTableModels$* Methode implementiert. Diese baut aus den Tabelleneinträgen die sogenannten TableModels. Diese werden im Frontend in der Tabelle verwendet, um die Daten aus dem Backend direkt im Frontend im HTML verwenden zu können. Dadurch können sie direkt anzeigt werden. Im Service läuft somit die gesamte Logik des Programmes.

5.2.4 Core Module

Das Core Module enthält die Frontend-Services. Dabei gibt es einen Service mit dem beispielsweise die aktuelle ProjektId ermittelt werden kann. Dies ist im Listing 5.4 zu sehen. Des Weiteren gibt es einen Electronservice mittels welchem unter anderem ein neues Electronfenster, zur Verlinkung von Dokumenten, geöffnet werden kann. Außerdem gibt es einen Messageservice, um Informationen, wie eine Fehlermeldung, an den Benutzer weitergeben zu können, sowie den Projektservice mit dem neue Projekte erstellt werden.

5.2.5 App Module

Das App Module ist das Root-Module, dass jede Angular-Applikation besitzt. Denn jede Angular App besitzt mindestens eine NgModule Klasse [Goo20]. Hier werden die einzelnen Module die genutzt werden importiert. In dieser Anwendung werden somit die Module CAST, STPA, Home, Shared and Core importiert.

5.2.6 Shared Module

Im Shared Module werden die Front-End Komponenten, die in den Ansichten zu sehen sind, generisch implementiert. Somit können mehrere Module, wie die CAST und STPA Module auf die gleichen Views zugreifen und diese nutzen. Hier werden in den Controlls spezielle Views festgehalten, die innerhalb einer Tabellenzeile angezeigt werden können. Dabei wird beispielsweise ein Textfeld implementiert. Dieses wird dann in einem anderen Modul verwendet und dort mit einem expliziten Label und den aktuell existierenden Daten gefüllt. Außerdem werden hier auch Komponenten wie die Tabelle, der Filter und ebenfalls die Kontrollstruktur und der Texteditor implementiert. Auch diese können dann anschließend in den benötigten Modulen verwendet werden.

5.2.7 Home Module

Im Home-Module befindet sich die Implementierung der Home-Komponente. In dieser können Funktionen aufgerufen werden, um die STPA und CAST Module zu starten. Dabei handelt es sich beispielsweise um das Erstellen eines neuen CAST Projektes, womit das CAST-Module gestartet wird, nachdem ein neues Projekt mittels dem Projektservice (Core-Module) erzeugt wird. Analog hierzu können STPA Projekte erstellt werden, die dann das STPA-Module aufrufen.

5.2.8 STPA Module

Das STPA Module wurde in den Bachelorarbeiten [Gre20] und [Hel20] implementiert und wird in diesen beschrieben.

Listing 5.4 CreateHazard Methode.

```
 hazardController.create(this.navigationService.currentPoint.projectId).catch((err: Error) =>
{
      this.msg.info(err.message);
      this.shouldFocus = false;
    });
```

5.2.9 CAST Module

Im CAST-Module werden die expliziten Ansichten und der Aufruf der Logik umgesetzt. Beispielhaft wird hier die Anlage eines Hazards beschrieben. Der Programmcode hierzu ist in Listing 5.4 zu sehen. Beim Anlegen eines Hazards wird auf den Controller des Hazards aus der Logik-Schicht und auf dessen create-Methode zugegriffen. Beim Erstellen eines Hazards wird die ProjektId mitgegeben. Diese wird durch den Aufruf des NavigationServices ermittelt. Wenn das Anlegen des Hazards fehlschlagen sollte, wird der MessageService aufgerufen und eine Fehlermeldung gegeben. Zusätzlich wird die boolean Variable shouldFocus auf false gesetzt, da keine Tabellenspalte fokussiert werden kann, die aufgrund des Fehlers nicht erstellt wurde. Außerdem werden die anderen Funktionen aufgerufen, die in den Services implementiert wurden. In einer Hazardtabellenzeile kann ebenfalls ein Constraint verlinkt werden. Beim Hinzufügen eines Chips wird der createConstraintLink aufgerufen und beim Löschen wird dieser wieder mittels removeConstraintLink entfernt.

5.3 Angewendete Prinzipien und Guidelines

Es wurde versucht, das System so zu implementieren, dass es gut erweiterbar ist. Hierzu können, sowohl im Frontend, wie auch in der Logik Schicht, neue Module eingefügt werden. Dabei werden ebenfalls einige Guidelines und Prinzipien angewendet, die nun im Folgenden kurz erklärt werden. Um die Verständlichkeit des Codes zu erhöhen, wurde zunächst alles im Rahmen der Möglichkeiten von TypeScript typisiert.

Des Weiteren wurde *"inversion of control"* verwendet. Dabei wird Inversify genutzt [Jan17]. Die Anwendung ist in Listing 5.3 zusehen. Hierbei wird mittels @inject direkt auf das Repository zugegriffen ("*Dependency Injection*").

Außerdem wurde das *Respository Pattern* verwendet. Dabei geht es darum, die Bussinesslogik von der Schicht zu trennen, in der explizit auf die Daten zugegriffen wird. Das bedeutet, dass die Repositories eine Zwischenschicht zwischen der Datenbank und der Service Schicht bilden. Somit können ebenfalls größere Code Duplikate vermieden werden und die Wartung wird einfacher [Ede20]. Im Listing 5.5 ist der Lösch-Aufruf zu sehen, der ausgeführt wird, wenn eine Empfehlung gelöscht wird. Im Lösch-Dialog wird hierbei angezeigt, dass die dazugehörigen Unterempfehlungen ebenfalls gelöscht werden. Die Frontend Komponente ruft dabei den Controller und dieser den Service auf. Von hier wird nun die zuvor beschriebene *"Dependency Injection"* durchgeführt, die dann die Repository-Methode ausführt. In der abgebildeten Methode werden nun zunächst die Projektids verglichen und dann alle Unterempfehlungen gelöscht, in denen die ParentId der Id der Empfehlung entspricht.

Listing 5.5 Methode zur Löschung von Unterempfehlungen, wenn die Empfehlungen gelöscht wurde.

```
  public async removeAllForRecommendationId(projectId: string, recommendationId: string):
Promise<boolean> {
    const collection = await this.getCollection();
    return collection
      .find()
      .where('projectId')
      .eq(projectId)
      .where('parentId')
      .eq(recommendationId)
      .remove()
      .then(() => true);
  }
}
```

Als Ganzes wurde versucht, die Anwendung nach dem *KISS* (“Keep it simple, stupid”) und *DRY*(“Don’t repeat yourself”) Prinzip, also möglichst einfach und ohne Wiederholungen zu halten [Luk18].

5.4 Funktionen

Um dem Anwender ein besseres Nutzererlebnis zu bieten und so das bestmögliche Analyseergebnis zu erhalten, wurden weitere Funktionen implementiert.

5.4.1 Tabelle

Die am meisten verwendete Funktion ist die Tabelle. Dabei gibt es einige grundlegende Features, die im Folgenden kurz beschrieben werden. Angelegt wird eine neue Tabellenzeile über den Plus-Button. Dabei gibt es zwei Arten. Der blaue Plus-Button legt eine neue Tabellenzeile an. Außerdem wird in mehreren Ansichten ein pinker Plus-Button verwendet, wenn eine Vorauswahl getroffen werden muss. In jeder Tabelle, außer in der *“Rolle im Unfall“*, gibt es die Möglichkeit eine Tabellenspalte auch wieder zu löschen. Dabei wird im Löschdialog angezeigt, welcher Tabelleneintrag gelöscht wird und welcher zugehörige Eintrag mit gelöscht wird, falls Abhängigkeiten bestehen. Ein Beispiel hierfür ist, dass bei der Löschung einer Empfehlung die dazugehörigen Unterempfehlungen ebenfalls gelöscht werden. Bei jeder Tabelle gibt es ebenfalls eine Filterungsmöglichkeit. Diese wird nun im folgenden Abschnitt 5.4.2 beschrieben.

5.4.2 Filterung

Eine weitere Funktion, die sich in jeder Tabelle bietet, ist die Filterungen von Elementen. Dabei können bestimmte Felder der Tabellenspalten ausgewählt werden. Diese werden mittels eines Chips angezeigt. In diesen kann nun mittels eines Suchbegriffes nach bestimmten Tabelleneinträgen gesucht werden. Außerdem kann die Suche durch den Klick auf das *NOT* vor dem Chip verneint werden.

Abbildung 5.2: Ansicht eines verlinkten Dokuments in der Einzelplatzanwendung.

Auch können mehrere Suchbegriffe, mit *AND* oder *OR* verknüpft, sowie ebenfalls *verneint* werden. Außerdem gibt es hier die Möglichkeit explizit nach Zuständen zu suchen. Die durchsuchbaren Felder können relativ einfach angepasst werden. Zusätzlich ist es möglich, Filter vorausgewählt anzeigen zu lassen. Dies ist in der *“Rolle im Unfall“*-Ansicht zu sehen ist. Hier wird als Komponenten-Typ der Controller vorausgewählt, um nur Controller in der Tabelle anzeigen zu lassen.

5.4.3 Verlinkte Dokumente

In den Abschnitten in denen Freitextfelder vorhanden sind, gibt es ebenfalls immer die Möglichkeit Dokumente zu verlinken. Dabei wurde hier die Umsetzung von zwei Arten der Verlinkung ermöglicht. Diese ist aber jederzeit um weitere Arten erweiterbar. Zum einen gibt es die Möglichkeit URLs zu verlinken, hierzu wird ein Name und eine Webseite angeben. Dabei wird beispielsweise eine Firmen-Webseite oder ein wichtiges Onlinedokument verlinkt. Bei einem Klick auf *“Öffnen”* (Abbildung 5.2) wird die URL in einem eigenen Electron-Fenster geöffnet. Dadurch ist es möglich das verlinkte Dokument parallel zu der CAST-Analyse zu betrachten und Informationen parallel abzurufen.

Zum anderen gibt es die Möglichkeit PDF-Dokumente zu verlinken. Dabei kann über *“durchsuchen”* der Pfad einer PDF angeben werden. Wenn das Namensfeld leer ist, wird dies automatisch mit dem PDF Namen gefüllt, anderenfalls kann hier ebenfalls ein Name vergeben werden. Auf dem “Öffnen”-Button wird die PDF im Standard PDF-Viewer des PCs geöffnet. Hier wird beispielsweise häufig der offizielle Unfallbericht verlinkt.

In beiden Fällen gibt es außerdem die Möglichkeit die verlinkten Dokumente zu löschen oder zu bearbeiten.

5.4.4 Fragen

In jeder Ansicht, beginnend mit den *Hazards*, bis einschließlich den *internen und externen wirtschaftlichen und verwandten Faktoren*, können Fragen angelegt werden. Dabei können Fragen im Textfeld benannt und dann mittels des Plus-Buttons hinzugefügt werden. Diese, sowie die bereits erstellten Fragen, werden darunter in einer Liste angezeigt. Fragen, die mittels des Plus-Buttons angelegt wurden, werden mit einem grünen Verlinkungssymbol gekennzeichnet. Außerdem wird hinter der Frage mittels eines Chipfeldes die verlinkte Entität angezeigt. Es wurde ebenfalls die Möglichkeit hinzugefügt, zu einer Entität eine bereits bestehende Frage zu ergänzen. Dazu wird auf die Frage in der Liste geklickt und das Verlinkungssymbol, sowie der Chip erscheinen. Auf dem selben Weg können die Verlinkungen von Fragen ebenfalls wieder aufgehoben werden. Alle Fragen werden in der Fragen & Antwort Ansicht gesammelt. Dort werden alle Fragen sowie ihre verlinkten Komponenten angezeigt. Außerdem können hier Antworten eingetragen werden, die sich im Laufe der Analyse ergeben haben. Es gibt hier nicht die Möglichkeit, weitere Verlinkungen zu erzeugen, da dies nur in einer Tabellenzeile der Schritte 1 - 4 direkt geschehen soll. Der Analyst hat nur hier den gesamten Überblick über das betrachtete Objekt und kann sich somit noch einmal ein Bild darüber machen, ob die Verlinkung sinnvoll ist. Es können aber auch in dieser Ansicht weitere Fragen erstellt werden, denn bei der Durchsicht oder auch der Beantwortung einer Frage können neue Fragen auftreten. Ziel ist es, möglichst alle Fragen zu sammeln und Antworten auf diese zu finden, um am Ende daraus bestmögliche Empfehlungen ableiten zu können.

5.4.5 Zustände

Der Zustand einer Tabellenzeile wird an der linken Seite der Zeile angezeigt (Abbildung 5.3). In rot wird der Zustand angezeigt, wenn die Komponente auf *TODO* gesetzt ist. Es wird ebenfalls markiert, wenn die Komponente auf *DOING* gesetzt wurde (mittels der Farbe gelb). Die Farbe grün steht für *DONE*. Im einzelnen bedeuten die Zustände folgendes: Auf *TODO* wird die Komponente standardmäßig dann gesetzt, wenn eine neue Tabellenzeile durch das Klicken auf den Plus Button

Abbildung 5.3: Ansicht eines Hazards mit der Markierung über die States: *TODO* (Daten aus [BSS17, Seite 20]).

erzeugt wird. Der Zustand sollte verwendet werden, wenn die Komponente noch nicht bearbeitet wurde. Wenn die Komponente bearbeitet wurde aber die Bearbeitung noch nicht abgeschlossen ist, wird die Komponente auf *DOING* gesetzt. Wenn die Komponente auf *DONE* gesetzt wird, ist die Bearbeitung dieser vollständig abgeschlossen.

5.4.6 Chip

Chips werden im ganzen System benutzt um Verlinkungen zwischen Komponenten zu kennzeichnen. Dabei gibt es verschiedenen Arten. Zum einen Chips die nur angezeigt werden, aber nicht entfernt werden können. Diese werden beispielsweise bei den Verlinkungen zwischen den Empfehlungen und den Unterempfehlungen genutzt. Zum anderen gibt es Chips mit der Möglichkeit zur Bearbeitung, Löschung und auch Chips bei denen mehrere Objekte verlinkt werden können. Ein Beispiel sind die Verlinkungen zwischen Hazards und Constraints. Wenn man über den Chip *“hovered”* wird der Name des Chips angezeigt.

6 Anwendungsbeispiel

In diesem Kapitel wird das System anhand eines durchgängigen Beispiels beschrieben. Als Beispiel dient hierbei die Fachstudie [BSS17] des Zugunglücks von Bad Aibling. Diese wurde vervollständigt, da es sich um eine CAST-Analyse nach einer älteren Veröffentlichung handelt [Lev11], während das System auf dem CAST-Handbuch [Lev19] basiert. Zur Vervollständigung wurde als Grundlage der offizielle Unfallbericht der deutschen Bahn genutzt [Bun18].

6.1 Home Komponente

Zunächst folgt eine kurze Beschreibung der Home-Ansicht der Anwendung, die in Abbildung 6.1 dargestellt ist. In der Home-Komponente kann ein neues CAST-Projekt angelegt werden. Dabei wird ein neuer Tab erzeugt und dort hin navigiert. Neue Projekte werden immer mit *"untitledCastProjekt"* benannt.

Gespeicherte Projekte können hier ebenso geöffnet werden. Die zugehörigen Dateien besitzen hierbei die Endung *".cast"*. Um eine Datei auszuwählen wird hierfür das Verzeichnis der Dateien geöffnet. Nach dem Öffnen wird zu dem zugehörigen Tab navigiert und das Projekt geöffnet.

Wenn man ein neues Projekt anlegt und keine Änderungen darin vornimmt, kann ein Projekt über das Kreuz auf dem Tab geschlossen werden. Wenn Änderungen vorgenommen wurden, öffnet sich ein Speicher-Dialog. In diesem können die Änderungen entweder verworfen oder gespeichert werden. Wenn das Projekt gespeichert werden soll und bereits ein Pfad existiert, wird das bestehende Projekt überschrieben. Anderenfalls wird das Dateiverzeichnis angezeigt und das Projekt muss

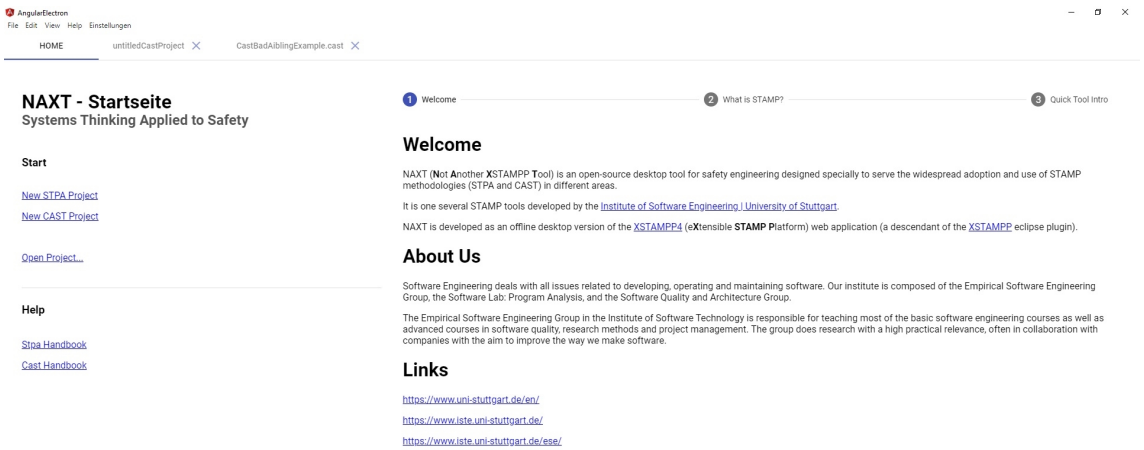

Abbildung 6.1: Ansicht der Home-Komponente.

benannt werden. Zur Änderung von Name und Speicherort des Projektes kann im Menü *"Save as... "* ausgewählt werden. Es öffnet sich hierbei wieder das Dateiverzeichnis. Wird die Anwendung geschlossen ohne die Projekte zu schließen, so werden diese beim Neustart der Anwendung automatisch wieder geöffnet.

In der Home Ansicht ist zudem das CAST-Handbuch [Lev19] eingefügt. Dieses öffnet die PDF des Handbuches in einem eigenen Electron-Fenster. Durch das eigenständige Fenster wird es ermöglicht, das CAST-Handbuch während einer Analyse parallel zu nutzen, um Details nachschlagen zu können.

In den ersten beiden Ansichten wird zunächst eine kleine Willkommensansicht mit den Links zur Universität Stuttgart, dem ISTE und dem ESE abgebildet. Dabei wird mit einem Klick auf einen der Links ein neues Electron-Fenster mit der jeweiligen Seite geöffnet. In der zweiten Ansicht gibt es eine kurze Erklärung über STAMP im Allgemeinen und wofür es genutzt wird. Außerdem ist in der Home Komponente ein Intro des System gegeben. Hier existiert eine voll funktionsfähige Tabelle, um die Funktionen dieser zu demonstrieren. Außerdem wird hierbei die Funktionalität des blauen, sowie des pinken Plus-Buttons und die Chipverlinkung erklärt.

6.2 Fragen und Antworten

In der Fragen und Antworten Ansicht sind alle Fragen aufgelistet, die in den einzelnen Komponenten festgehalten wurden. Außerdem können neue Tabelleneinträge angelegt werden. Neue Fragen können bei der Durchsicht der existierenden Fragen, sowie bei der Beantwortung dieser entstehen [Lev19]. Die festgehaltene Frage die in Abbildung 6.2 zu sehen ist, wurde in Abschnitt 6.11 (*Reiter "Kommunikation und Koordination"*) erstellt. Dabei handelt es sich um eine entstandene Frage zur Kommunikation zwischen dem Zugführer und dem Fahrdienstleiter, in Bezug auf die Notfallbenachrichtigung. Dazu wurde ebenfalls eine Antwort gefunden, die bereits dokumentiert wurde. Der Chip zeigt eine Verlinkung zu den jeweiligen Komponenten an, in denen die Frage als benötigt angesehen wird. In diesem Fall ist hierbei nur die Komponente angegeben in der die Frage erstellt wurde, also bei der Analyse der Kommunikation zwischen den beiden bereits genannten Parteien, zum Zeitpunkt des Unfalls [BSS17; Bun18].

Abbildung 6.2: Ansicht der Fragen und Antworten (Daten abgeleitet aus [BSS17; Bun18]).

Abbildung 6.3: Ansicht der Unfallbeschreibung in der Einzelplatzanwendung (Daten aus [BSS17, Seite 20]).

6.3 Systembeschreibung

Zunächst werden in der Systembeschreibung alle grundsätzlich wichtigen Informationen gesammelt und die Grundzüge beschrieben. In diesem Beispiel handelt es sich um eine eingleisige Zugstrecke, auf der ein Fahrdienstleiter steuert, welche Züge sich auf dem Gleis befinden dürfen [BSS17]. Außerdem kann es hier beispielsweise sinnvoll sein die Strecke bildlich festzuhalten. Die Ansicht ist dabei die selbe, wie in der Unfallbeschreibung (Abbildung 6.3).

6.4 Unfallbeschreibung

In der Ansicht gibt es ein Freitextfeld, das über diverse Möglichkeiten verfügt, Texte hervorzuheben. In diesem sollte zur Unfallbeschreibung unter anderem der Zeitpunkt des Unfalls, sowie das Datum und ein grober Verlauf des Unfallhergangs festgehalten werden. In Bezug auf die CAST-Analyse des Zugunglückes in Bad Aibling, wird hier festgehalten, dass der Unfall am 09.02.2016 um 06:46 Uhr aufgetreten ist. Ebenfalls wird beschrieben, wie viele Leute verletzt wurden, beziehungsweise tödlich verunglückten. Außerdem, wer am Unfall beteiligt war und mit welchen Geschwindigkeiten die Züge kollidierten [BSS17; Bun18]. Zur Unterstützung der Analyse gibt es hier die Möglichkeit, Dokumente zu verlinken. Diese Funktion wurde bereits in Abschnitt 5.4.3 erklärt. Bei der Unfallbeschreibung wird meist der offizielle Unfallbericht [Bun18] verlinkt.

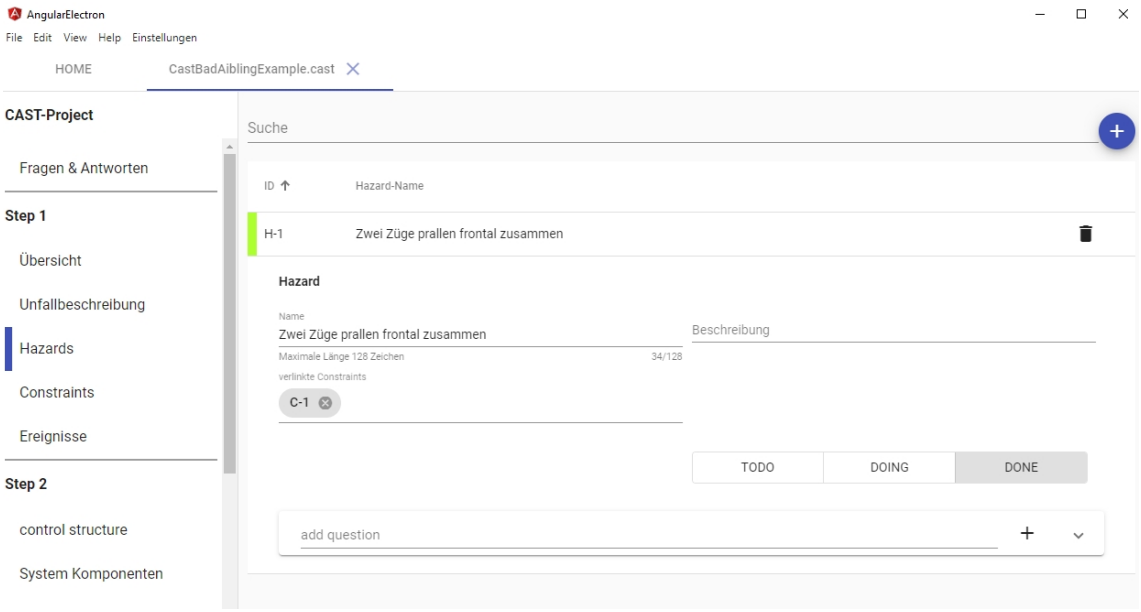

Abbildung 6.4: Ansicht eines Hazards in der Einzelplatzanwendung (Daten aus [BSS17, Seite 20]).

6.5 Hazard

Im nächsten Unterpunkt werden die Hazards festgehalten (Abbildung 6.4). So ist der offensichtliche Hazard in dem Beispiel Bad Aibling, dass zwei Züge frontal auf einer eingleisigen Strecke zusammengestoßen sind [BSS17]. Außerdem kann hier eine Beschreibung eingetragen, sowie eine Verlinkung zu beliebig vielen Constraints hinzugefügt werden. Der Bearbeitungszustand der Komponente kann hier ebenso zugeordnet werden. Näheres zu den Zuständen findet sich in Abschnitt 5.4.5. Außerdem können in Schritt 1 von den Hazards bis einschließlich Schritt 4, Fragen angelegt werden, wenn sich bei der Analyse und Anlage des Hazards (andere Elemente) eine Frage ergibt (Abschnitt 5.4.4). Neue Hazards werden über den blauen Plus-Button rechts oben angelegt.

6.6 Constraints

Ein Constraint in diesem Beispiel ist, dass keine zwei Züge auf einer Strecke frontal aufeinander zu fahren dürfen. Die daraus resultierende Kollision führt sonst zu einem Unfall [BSS17]. Hier gibt es die gleiche Ansicht wie im Reiter “Hazards”. Der einzige Unterschied ist, dass ein Constraint nur einen Hazard verlinken kann [Lev19].

6.7 Ereignisse

Als letztes werden in Schritt 1 die Ereignisse festgehalten (Abbildung 6.5). Dabei soll sachlich nacheinander festgehalten werden was vor dem Unfall, bis zum Zeitpunkt des Unfalls geschehen ist. Beginnen sollte man mit einem Ereignis, das zeitlich weit genug vor dem Unfall liegt, sodass alle

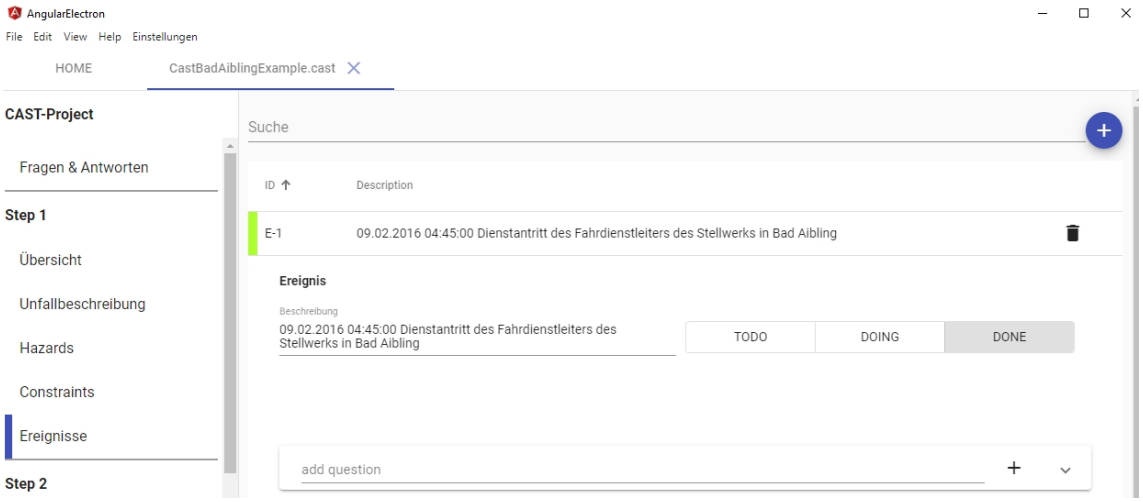

Abbildung 6.5: Ansicht eines Ereignisses in der Einzelplatzanwendung (Daten aus [BSS17, Seite 21]).

relevanten Informationen berücksichtigt werden. Dabei sollte alles Geschehene festgehalten werden, auch wenn es nicht relevant für den Unfall erscheint. Denn auch nebensächliche Ereignisse können im Nachhinein wichtig für den Unfall sein [Lev19]. Ein sinnvoller Zeitpunkt für die Analyse der Ereignisse ist, im Bezug auf das Zugunglück, der Beginn der Schicht des Fahrdienstleiters. Dann wird der genau Ablauf bis zum Zeitpunkt des Unfall aufgeführt. [BSS17].

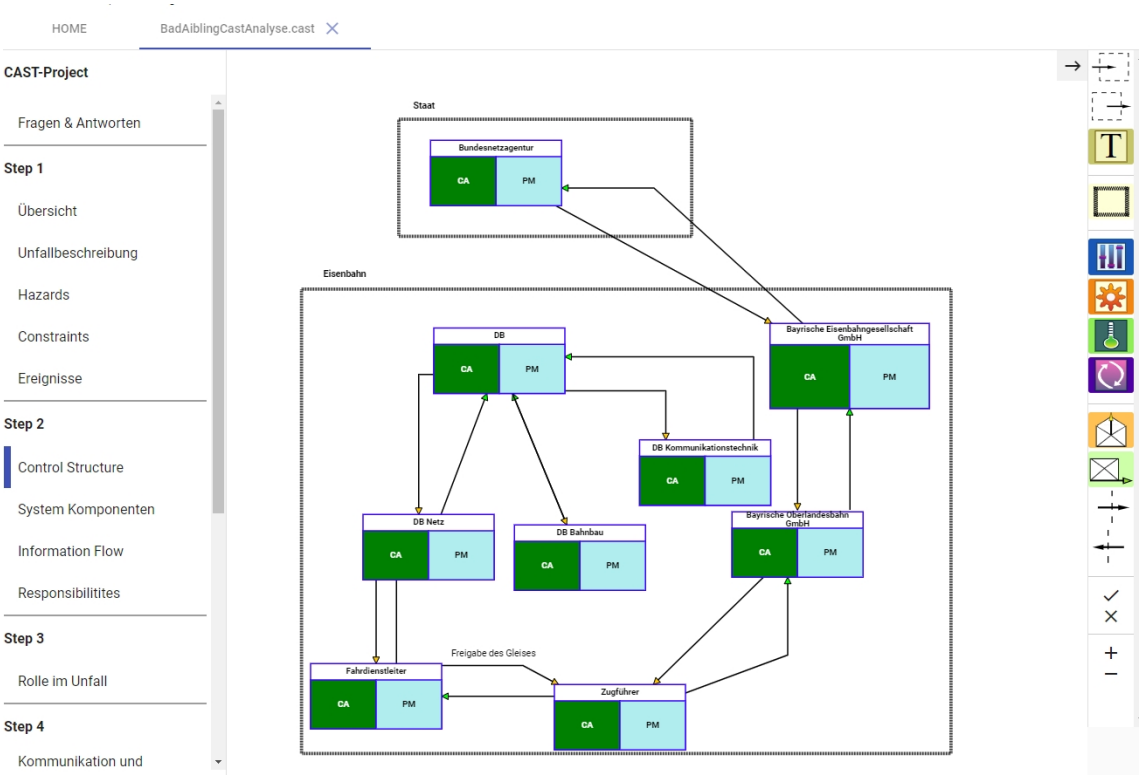

Abbildung 6.6: Ansicht der Kontrollstruktur in der Einzelplatzanwendung (Daten aus [BSS17, Seite 28]).

6.8 Kontrollstruktur

In diesem Reiter wird die Kontrollstruktur modelliert (Abbildung 6.6). Dabei gibt es Boxen und Pfeile. Eine Input- und Output Box kann immer nur mit dem korrespondierenden Input-, Output-Pfeil verbunden werden und mit dem anderen Ende jeder Box verbunden sein, außer mit sich selbst. Die anderen Boxen sind Aktuatoren, Controlled-Processes, Controller und Sensor. Zudem gibt es die Pfeile Control-Action und Feedbacks. Eine Kontrollaktion verbindet einen Controller mit einem Aktuator/Controlled-Process und einen Aktuator mit einem Controlled-Process. Ein Feedbackpfeil verbindet einen Sensor mit einem Controller und einen Controlled-Process mit einem Sensor, sowie einen Controlled-Process mit einem Controller. Boxen kann ein Name zugeteilt werden und Pfeilen ein Label. Außerdem gibt es eine Dashbox um Einheiten zu kennzeichnen und Textfelder, um Text in die Zeichnung einzutragen. Die Funktionalitäten der Kontrollstruktur wurden aus dem XSTAMPP 4.1 [1] übernommen.

Im Reiter Systemkomponente sind die Elemente Controller, Sensoren, Aktuatoren und Controlled-Processes ebenfalls tabellarisch festgehalten. Dabei wird immer der Typ und der Name in der Tabelle übernommen, wenn das Element in der Kontrollstruktur angelegt wurde. Es können ebenfalls Elemente in der Tabelle angelegt werden. Diese sind in der Kontrollstruktur verlinkbar. In der

[1] siehe Webanwendung: https://web.xstampp.de

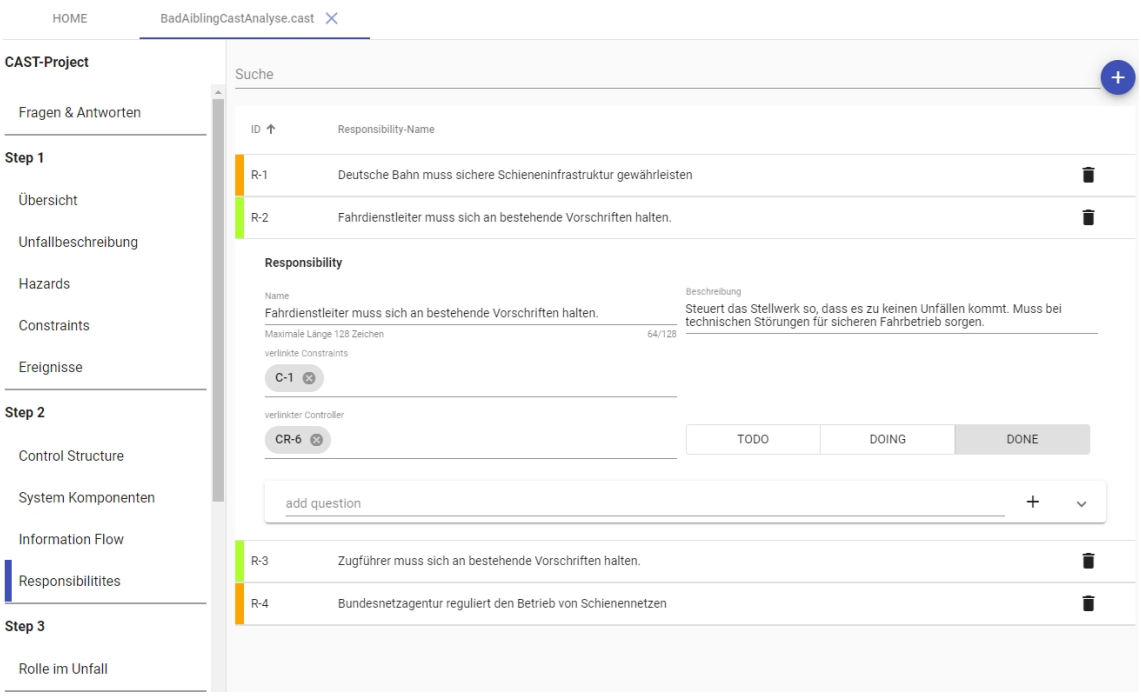

Abbildung 6.7: Ansicht einer Responsibility in der Einzelplatzanwendung (Daten aus [BSS17, Seite 25]).

Tabelle können außerdem noch Beschreibung und Zustand zu jedem Element der Kontrollstruktur hinzugefügt werden. Analog werden im Information-Flow Input, Output, Feedback und Control-Action angelegt. Hier gibt es die gleichen Möglichkeiten wie in der System Komponente.

Die in Abbildung 6.6 zu sehende Kontrollstruktur bildet die wichtigsten Controller, die am Unfall beteiligt waren, ab. Hier werden beispielsweise der Zugführer und der Fahrdienstleiter dargestellt. Außerdem wird hierbei durch die Dashboxen abgebildet, welche Controller der Eisenbahn und welche dem Staat zugeordnet werden. Der Controller *Fahrdienstleiter* wird hierbei ebenfalls in der System Komponente um eine Prozessvariable *“Streckenbelegung”* ergänzt [BSS17].

6.9 Responsibilities

In Schritt 2 werden auch die Responsibilities festgehalten (Abbildung 6.7). Zunächst kann ein Name für die Responsibility vergeben werden. Es folgt eine kurze Beschreibung, die Zustände und das Fragenfeld. Zusätzlich gibt es zwei Felder, um Verlinkungen einzufügen. In einem Feld kann ein Controller hinzugefügt werden, der die Responsibility umsetzen soll. Im anderen Feld wird ein Constraint verlinkt. Dieser soll durch die Responsibility überwacht werden.

Ein Beispiel für eine Responsibility ist die, sich an die Vorschriften zu halten. Dies betrifft den Fahrdienstleiter. Der Controller, der für die Umsetzung zuständig ist, wird im Chip angezeigt und hier vom Benutzer zusätzlich bereits im Namensfeld festgehalten. Welche Aufgaben dies genau

beinhaltet, ist im Beschreibungsfeld dokumentiert. Außerdem kann zugewiesen werden, welcher Constraint hierdurch umgesetzt wird. In diesem Beispiel ist der Constraint zur Überwachung der Streckenbelegung verlinkt [BSS17].

6.10 Rolle im Unfall

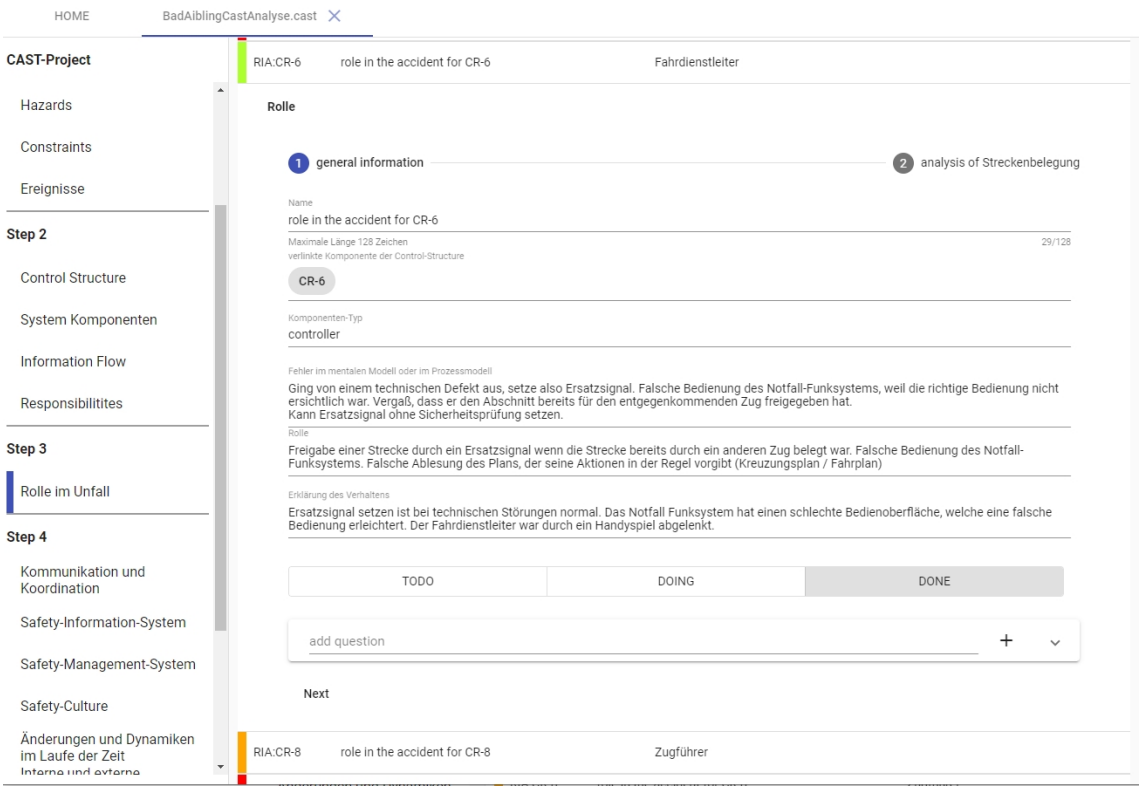

Abbildung 6.8: Tabelleneintrag eines Controllers im Rolle des Unfall Reiters der Einzelplatzanwendung (Daten aus [BSS17, Seite 25]).

In der Rolle im Unfall (Abbildung 6.8) wird zunächst für jedes Element der Kontrollstruktur eine Tabellenzeile erzeugt. Standardmäßig wird hier nach Controllern gefiltert. Diese werden mit “role in the accident CR-Id“ benannt. Außerdem wird in der hinteren Tabellenzeile der Name des Controllers angezeigt. Der Name der Komponente kann beim Öffnen der Tabellenspalte umbenannt werden. Außerdem wird der Typ der Komponente angegeben, wie beispielsweise *Controller*, sowie ein Chip der verlinkten Komponente angezeigt. Als nächstes soll, jeweils in einem Textfeld, die Rolle im Unfall, das Prozess- und Mentalmodell, sowie eine Erklärung zum Verhalten des Controllers beschrieben werden. Außerdem wird hier jeweils ein Zustand festgelegt, sowie Fragen erstellt falls diese auftreten[Lev19].

Wenn zu dem Controller ebenfalls Prozessvariablen angelegt wurden, so wird für jede Prozessvariable ein eigener Tab erzeugt. Dabei wird der Name der Variable angezeigt. Dann wird der aktuelle Wert während des Unfalls festgehalten, sowie versucht, eine Erklärung für die falsche Interpretation des Wertes zu finden, falls der Wert vom Controller beispielsweise falsch interpretiert wurde. Dies stellt ein Extra dar, dass nicht explizit im Handbuch [Lev19] erwähnt wurde.

Im Abschnitt Rolle im Unfall, können keine Tabellenspalten durch einen Plus-Button erzeugt, oder Zeilen gelöscht werden. Sie werden bei der Löschung des Elements der Kontrollstuktur automatisch mitgelöscht.

Abbildung 6.9: Ansicht einer Prozessvariable die analysiert wird, in der Rolle im Unfall (Daten angelehnt an[BSS17; Bun18]).

Beispielhaft bedeutet, dies nun von oben nach unten in der Ansicht der Analyse des Fahrdienstleiters. Zuerst wird der Controller als read-only Link angezeigt. Anschließend wird dann der Fehler im Modell des Controllers angegeben. Dabei wird festgehalten, dass er nachdem er den ersten Zug freigegeben hatte, den zweiten Zug mittels eines extra gesetzten Signals frei gab. Da er davon ausging, dass es sich bei der Sicherheitsvorkehrung, die ihm anzeigte dass das Gleis belegt ist, um ein technisches Problem handelte. Anschließend betätigte er einen Knopf um ein Notfallsignal auszulösen, allerdings geschah hierbei ein Bedienungsfehler. Des Weiteren geschah ein Ablesefehler. Die Rolle im Unfall war die falsche Bedienung, sowie die Freigabe eines belegten Gleises. Die Erklärung für das Verhalten des Controllers ist zum einen, dass der Fahrdienstleiter ein Handyspiel spielte, zum anderen wirkte sich die Gestaltung des Notfallsystems auf die Bedienung aus [BSS17].

Außerdem werden die Prozessvariablen analysiert, wie in Abbildung 6.9 zu sehen ist. Eine beispielhafte Variable könnte die Streckenbelegung sein. Dabei wird anschließend der Wert angegeben, der in diesem Fall *belegt* oder *frei* sein kann. Da der Fahrdienstleiter das Gleis als frei betrachtete, als er es trotz Warnung des Systems für den zweiten Zug frei gab, handelt es sich hierbei um eine Missinterpretation der Prozessvariable, da der Wert der Prozessvariable in der Realität belegt war [BSS17].

6.11 Mängelidentifizierung

Als nächstes wird das Gesamtsystem, dass durch die Kontrollstruktur abgebildet wird, betrachtet und die dort bestehenden Mängel analysiert [Lev19].

In der *Kommunikation und Koordination* gibt es ein Freitextfeld und die Möglichkeit, Dokumente zu verlinken, wie bereits in Abbildung 6.3 zu sehen ist. Außerdem gibt es die Möglichkeit, in der Tabelle über den Plus-Button Tabellenzeilen zu erzeugen. Dadurch ist es möglich, einzelne Verbindungen explizit zu betrachten. Dazu werden die zwei Controller in der Tabellenzeile direkt verlinkt. Dabei kann ebenfalls eine Beschreibung und die Zustände *TODO*, *DOING* oder *DONE* angeben werden. Eine Verbindung, die man hier beispielsweise analysieren sollte, ist die Kommunikation zwischen dem Zugführer und dem Fahrdienstleiter. Diese werden mittels Chips als Verlinkungen festgehalten. In der Beschreibung wird nun dokumentiert, dass keine Kommunikation zu Stande kam, da der zur Kontaktaufnahme benötigte Knopf nicht betätigt wurde. Daraus ableitend ergibt sich selbstverständlich die Frage, weshalb der richtige Notfallknopf nicht gefunden wurde und wie dies verbessert werden muss. Resultierend aus dieser Frage kann dann eine Empfehlung erstellt werden (die Überarbeitung des Notfalldesigns der Knöpfe)[BSS17]. In den Abschnitten *Sicherheitsinformationssystem*, *Sicherheitsmanagement*, *Sicherheitskultur*, *Änderungen und Dynamiken im Laufe der Zeit* und *interne und externe wirtschaftliche und verwandte Faktoren* gibt es ebenfalls, wie in Abbildung 6.10 ein Freitextfeld und die Möglichkeit Dokumente zu verlinken. Außerdem gibt es hier ebenfalls eine Tabelle, die aufgebaut ist wie die Vorherige. Hier lassen sich statt zwei Controllern aber nur ein Controller verlinken, da in diesen Ansichten immer ein spezifischer Controller betrachtet wird.

In der letzten Ansicht *"andere Faktoren"* können in einem Freitextfeld ebenfalls weitere Faktoren festgehalten werden, die in den anderen Ansichten nicht dazugehören, aber ebenfalls wichtig für die Analyse sind. Außerdem können, falls notwendig, Dokumente verlinkt werden.

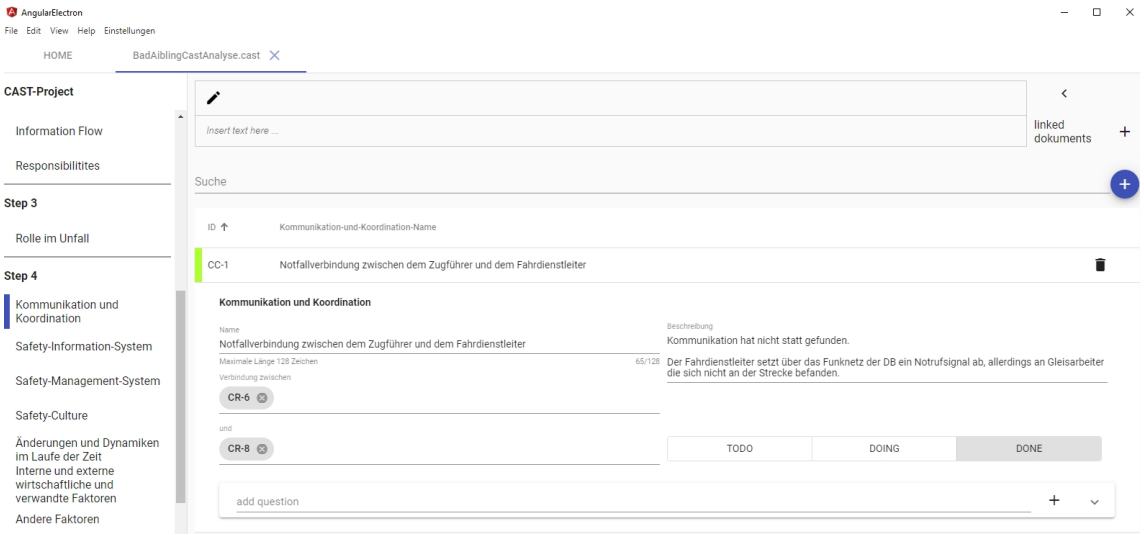

Abbildung 6.10: Ansicht der Kommunikation und Koordination in der Einzelplatzanwendung (Daten angelehnt an [BSS17; Bun18]).

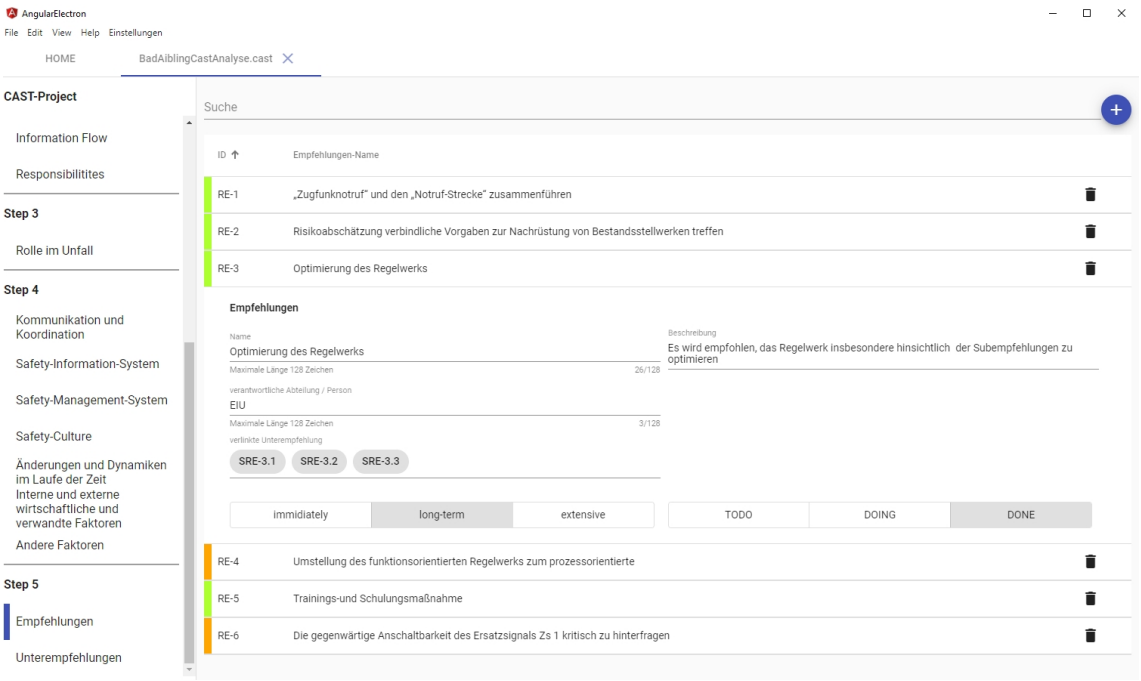

Abbildung 6.11: Ansicht einer Empfehlung in der Einzelplatzanwendung (Daten aus [Bun18, Seite 100]).

6.12 Empfehlungen

In Schritt 5 werden die Empfehlungen erstellt. Die Ansicht ist in Abbildung 6.11 zu sehen. Dabei wird zuerst der Name der Empfehlung festgehalten. Wenn notwendig, kann diese in einer Beschreibung genauer definiert werden. Außerdem wird sie, um die Verantwortlichkeit der Umsetzung festzuhalten, entweder einer Person oder einer Abteilung zugewiesen. Außerdem wird in der Button-Gruppe festgehalten, wie lange die Umsetzung der Empfehlung dauern wird. Sie kann direkt umgesetzt werden und wird somit in der Button-Gruppe mit *"immidately"* festgehalten. Des Weiteren kann sie auf längerfristig mittels *"long-term"* und, falls eine behördliche Änderung oder ein neues Gesetz umgesetzt werden muss, auf *"extensive"* gesetzt werden [Lev19]. Außerdem können zu einer Empfehlung beliebig viele Unterempfehlungen verlinkt werden. Dies geschieht über den Reiter *"Unterempfehlungen"*. Bei dem Klick des pinken Plusbuttons wird eine Auswahl der existierenden Empfehlungen geben, wie in Abbildung 6.12 zu sehen ist. Dabei wird beim Klicken auf eine Empfehlung eine neue Unterempfehlung erzeugt und hierbei im Chip-Feld ein readonly-Chip mit der ausgewählten Empfehlung angezeigt. Außerdem kann die Umsetzung einer Unterempfehlung nie länger dauern, als die dazugehörige Empfehlung. Daher können die längeren Zustände der Button-Gruppe nicht mehr angeklickt werden. Wenn man eine Empfehlung löscht, wird im Lösch-Dialog angezeigt, dass die zugehörigen Unterempfehlungen ebenfalls gelöscht werden. Das Löschen kann in diesem Dialog abgebrochen werden.

Aus den entstandenen Fragen, sowie ihren Antworten, können Empfehlungen abgeleitet werden. Ein Beispiel hierfür ist, die Knöpfe für die Notfallbenachrichtungen zusammenzuführen, da der betätigte Knopf im Unfall statt den Zugführer, Gleisarbeiter die nicht vor Ort waren benachrichtigte [BSS17; Bun18]. Eine weitere Empfehlung ist die *"Optimierung des Regelwerks"*[Bun18, Seite 100]. Da

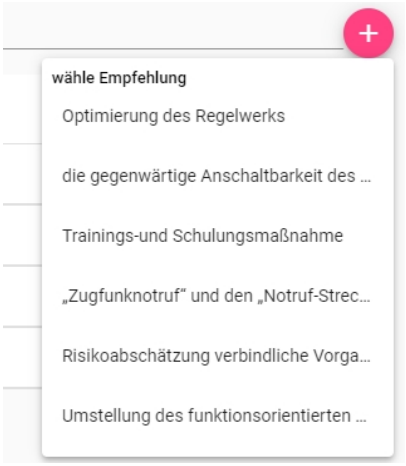

Abbildung 6.12: Auswahlmenü des Plus-Buttons im Reiter Unterempfehlungen (Daten aus [Bun18, Seite 100]).

es sich hier um eine größere Empfehlung handelt wurde diese in einzelne Unterempfehlungen aufgeteilt, die wie in Abbildung 6.11 zu sehen ist, angezeigt werden [Bun18]. Außerdem wird das *“Eisenbahninfrastrukturunternehmen (EIU)”* als verantwortliches Unternehmen zur Umsetzung dieser Regeln festgelegt [Bun18, Seite 100]. Um nun die Unterempfehlungen anzulegen, wird in deren Reiter, das über den Klick des pinken Buttons (Abbildung 6.12) erreichte Menü, verwendet. Durch dieses wird die Empfehlung *“Optimierung des Regelwerks”*[Bun18, Seite 100] ausgewählt. Dann wird eine Unterempfehlung erstellt, sowie die Empfehlung verlinkt und wie in Abbildung 6.11 zu sehen ist, dort ebenfalls die Verlinkung angezeigt.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein System implementiert, dass den CAST Prozess abbildet, um den Analysten bei seiner Analyse zu unterstützen. Dafür wurde zunächst die Theorie dargelegt, auf die sich die Anforderungsanalyse bezieht. In der Anforderungsanalyse werden sowohl die einzelnen Schritte, wie auch die Anpassungen durch XSTAMPP 4.1 und STPA dargelegt. Auf Grundlage der Anforderungsanalyse wurde die Einzelplatzanwendung implementiert. Dazu wurden zunächst die Technologien festgelegt. Darauf aufbauend die Architektur, sowie die Erklärungen der einzelnen Komponenten und Funktionen. Dabei werden ebenfalls kurz die Prinzipien und Guidelines erklärt. Im Anschluss wird die Software anhand eines Anwendungsbeispiels vorgestellt. Dabei werden alle Schritte des CAST-Prozesses, sowie die Home-Komponente erklärt.

Ausblick

Der nächste logischer Schritt sollte sein, Feedback über die implementierte Software zu erhalten und die erlangte Resonanz auszuwerten.

Im Weiteren werden Features vorgestellt, die aufgrund des Zeitaufwandes in dieser Bachelorarbeit nicht mehr umsetzbar waren.

Es könnte darüber nachgedacht werden, die Kommunikation und Koordination noch weiter zu analysieren. Dabei ist denkbar, jede vom Controller ausgehende und eingehende Kante in einer Art Matrix darzustellen. Ein Konzept hierfür ist bereits entstanden und wurde in Abbildung 7.1 festgehalten. Hierbei werden kritische Kanten in Orange, sowie mittels eines Ausrufezeichens

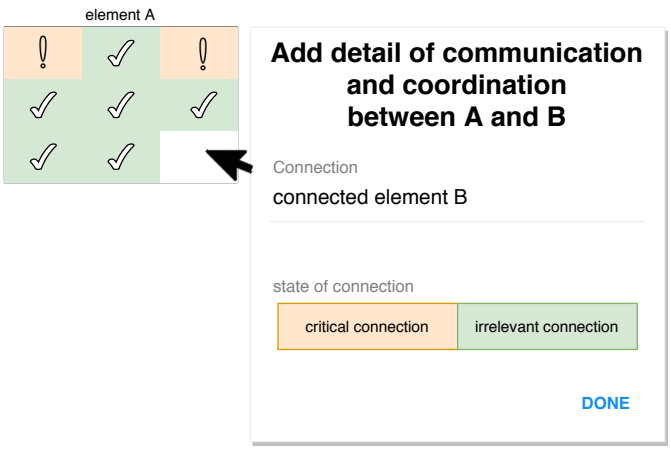

Abbildung 7.1: Konzept zur Erweiterung der derzeitigen Kommunikation und Koordination - Schritt 4.

gekennzeichnet. Während unauffällige Kanten mittels Haken und in grün gekennzeichnet werden. Dabei wird als Element A der Controller genannt, dessen Kommunikationskanäle überprüft werden sollen. In jedem Feld wird nun die Kommunikation und Koordination mit einem anderen Controller analysiert und bewertet. Es könnte hier ebenfalls noch überlegt werden, ein Beschreibungsfeld einzublenden, wenn die Verbindung als kritisch betrachtet wird. Der Vorteil bei dieser Art der Überprüfung wäre, dass auch überprüft wird, ob Kommunikationsverbindungen fehlen. Eventuell können hierbei auch Kommunikationskanäle auffallen, bei denen Informationen über Umwege zum anderen Controller gelangen, aber eine direkt Verbindung bestehen sollte.

Auch die anderen Kategorien von Schritt 4 können voraussichtlich noch ausgebaut werden, wenn notwendig oder gewünscht.

Außerdem ist es möglich die Fragen-Ansicht zu erweitern. Dabei könnten beispielsweise Filteroptionen einfügt werden, um nach Verlinkungen und expliziten Fragen zu suchen.

Abschließend wäre es sinnvoll einen PDF-Report anzubieten. Dabei sollte es möglich sein, einzelne Teile von CAST als PDF zu generieren. Ein Beispiel hierfür könnten der Export der Empfehlungen sein. Diese werden von diversen Organisation genutzt. Somit ist eine PDF zur Verteilung hierbei sinnvoll. Dabei sollte es ebenfalls die Option geben, die ganze Analyse zu exportieren.

Literaturverzeichnis

[AW14] A. Abdulkhaleq, S. Wagner. „A-STPA: Open Tool Support for System- Theoretic Process Analysis.“ In: 2014 STAMPP Conference at MIT, 2014 (zitiert auf S. 18).

[AW15] A. Abdulkhaleq, S. Wagner. „XSTAMPP: An eXtensible STAMP Platform As Tool Support for Safety Engineering“. In: 2015 STAMPP Conference at MIT, 2015 (zitiert auf S. 27).

[AW16] A. Abdulkhaleq, S. Wagner. „XSTAMPP 2.0: New Improvements to XSTAMPP Including CAST Accident Analysis and an Extended Approach to STPA“. In: 2016 STAMPP Conference at MIT, 2016 (zitiert auf S. 25, 27).

[BSS17] H. M. Berner, M. Schäfer, R. Stercken. „Fachstudie - Analyse des Zugunfalls in Bad Aibling, Deutschland mit STAMP/CAST Ansatz“. Fachstudie. Universität Stuttgart, 2016-2017 (zitiert auf S. 13, 27, 44, 47–58).

[Bun18] Bundesstelle für Eisenbahnunfalluntersuchung. *Unfallbericht Aktenzeichen:60uu2016-02/005-3323*. Version 1.0. 29.10.2018. URL: https://www.eisenbahn-unfalluntersuchung.de/SharedDocs/Downloads/EUB/Untersuchungsberichte/2016/114_Bad_Aibling_-_Kolbermoor.pdf?__blob=publicationFile&v=1 (zitiert auf S. 47–49, 56–59).

[Ede20] N. Eder. *Repository Pattern*. 2001 - 2020. URL: https://www.norberteder.com/das-repository-pattern-anhand-eines-beispiels-inkl-tests/ (zitiert auf S. 41).

[Goo20] Google. *Angular Website*. 2010-2020. URL: https://angular.io/guide/architecture-modules (zitiert auf S. 35, 40).

[Gre20] M. Greiner. „Implementierung des Systems XSTAMPP 4 als Einzelplatzanwendung“. Bachelorthesis. Universität Stuttgart, 2020 (zitiert auf S. 14, 27, 29, 32, 40).

[Hel20] F. Held. „Erweiterung der Einzelplatzanwendung des Systems XSTAMPP 4 um die Verfeinerung der Unsafe Control Actions für ein Model Checking“. Bachelorthesis. Universität Stuttgart, 2020 (zitiert auf S. 27, 40).

[Jan17] R. H. Jansen. *Infersify Website*. 2015 - 2017. URL: http://inversify.io/ (zitiert auf S. 41).

[Lev11] N. G. Leveson. *Engineering a Safer World: Systems thinking applied to safety*. MIT Press, Cambridge, 2011 (zitiert auf S. 15–17, 27, 47).

[Lev17] N. G. Leveson. „CAST Analysis of the Shell Moerdijk Accident“. In: MIT. Cambridge: EU Major Accident Hazards Bureau(MAHB), 2016-2017 (zitiert auf S. 27, 29).

[Lev19] N. G. Leveson. *CAST HANDBOOK: How to Learn More from Incidents and Accidents*. 2019. URL: http://sunnyday.mit.edu/CAST-Handbook.pdf (zitiert auf S. 13, 15, 19–25, 27, 29–32, 34, 47, 48, 50, 51, 55, 57, 58).

[LSM19] N. Levenson, D. Straker, S. Malmquist. „Updating the Concept of Cause in Accident Investigation“. In: ISASI (International Society of Air Safety Investigators), the Hague, Netherlands, Sep. 2019, S. 14 (zitiert auf S. 27).

[LT18] N. G. Leveson, J. P. Thomas. *STPA Handbook*. März 2018. URL: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf (zitiert auf S. 18, 19).

[Luk18] Lukas von generic.de AG. *Clean Code*. 25.05.2018. URL: https://generic.de/2018/05/25/dry-vs-kiss-clean-code-prinzipien/ (zitiert auf S. 42).

[Mey20] D. Meyer. *RxDB Website*. 2020. URL: https://rxdb.info/ (zitiert auf S. 35).

[OE20] OpenJS-Foundation, Electron-Contributors. *Electron Website*. 2013-2020. URL: https://www.electronjs.org/ (zitiert auf S. 35).

[Pie19] B. Pietrucha. *Angular Layers*. 2 Jul 2019. URL: https://angular-academy.com/angular-architecture-best-practices/ (zitiert auf S. 36).

[Roo15] M. Root. „A-CAST: Entwicklung eines Plugin-basierten Tools zur Unfallanalyse mit CAST (Causal Accident Analysis)“. Bachelorthesis. Universität Stuttgart, 2015 (zitiert auf S. 25, 27).

Alle URLs wurden zuletzt am 25. 08. 2020 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.

Schwieberdingen, 28.08.2020, E. Zimmer

Ort, Datum, Unterschrift