Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Implementierung des Systems XSTAMPP 4 als Einzelplatzanwendung

Maurice Greiner

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Wolfgang Fechner
Beginn am:	08. Januar 2020
Beendet am:	02. September 2020

Kurzfassung

Mit dem fortschreitenden technologischen Wandel nehmen zunehmend komplexe soziotechnische Systeme eine wichtige Rolle des modernen Lebens ein. Zur Analyse des Faktors Safety stellt Nancy Leveson [Lev16] dazu Systems-Theoretic Accident Model and Processes (STAMP) und die daraus resultierende Gefahrenanalysemethode Systems-Theoretic Process Analysis (STPA) [LT19] vor. Die neue Webanwendung eXtensible STAMP Platform (XSTAMPP) 4, zur Umsetzung von STPA, wurde in dieser Arbeit zum breiten Einsatz in Industrie und Forschung als Einzelplatzanwendung umgesetzt. Dazu wird in dieser Arbeit der Entwurf und die Umsetzung der Einzelplatzanwendung vorgestellt. Auch werden Schritte zu Verbesserungen der Anwenderfreundlichkeit und Erweiterbarkeit eingebracht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

A-CAST Automated CAST. 25

A-STPA Automated STPA. 25

BSCU Brake System Control Unit. 41

CAST Causal Analysis using System Theory. 13

DI Dependency Injection. 33

DRY Don't Repeat Yourself. 33

IoC Inversion of Control. 33

RxDB Reactive Database. 37

STAMP Systems-Theoretic Accident Model and Processes. 3

STPA Systems-Theoretic Process Analysis. 3

UCA Unsafe Control Action. 21

XSTAMPP eXtensible STAMP Platform. 3

XSTPA Extended Approach to STPA. 25

1 Einleitung

1.1 Ausgangssituation & Themendarstellung

In der Wissenschaftsphilosophie existiert der Begriff der Technikphilosophie. Wikipedia versteht darunter sowohl die philosophische Untersuchung der Bedeutung der Technik als auch die Auseinandersetzung mit dem Verhältnis von Mensch, Umwelt und Technik zueinander. Besonders der letzte Teil dieses Satzes nimmt in dieser Arbeit eine wichtige Rolle ein. Bilden Mensch, Umwelt und Technik eine Einheit, so entsteht ein soziotechnisches System. Banse et al. [BMW02] schreiben, dass Technik nicht als etwas Statisches angesehen werden kann, sondern als Bereich mit Genese, Dynamik und Wandel. Jedoch bieten sich mit immer neuen Technologien in soziotechnischen Systemen auch immer neue Gefahren für Mensch und Umwelt.

Zur Analyse von Safety in soziotechnischen Systemen, Safety fortlaufend als Begriff der Unfallvermeidung (Accidents Prevention), wird in dieser Arbeit der etablierte systemdenkende Ansatz STAMP verwendet. STAMP wurde von Nancy Leveson in ihrem Buch *Engineering a Safer World: Systems Thinking Applied to Safety* [Lev16] vorgestellt. STAMP ermöglicht ein besseres Verständnis warum Unfälle entstehen und wie sich diese in der Zukunft vermeiden lassen (vgl. [Lev04]).

Diese Arbeit ist Teil des fortlaufenden Entwicklungsprozesses der Softwarereihe XSTAMPP, die zur Durchführung von STAMP Methoden dient. Fokus ist dabei die bestehende Webanwendung XSTAMPP 4 als offlinefähige Einzelplatzanwendung umzusetzen.

XSTAMPP 4 bietet derzeit nur die, im späteren vorgestellte, STAMP Methode STPA als Projekttyp an. Diese dient zur Durchführung einer Gefahrenanalyse (Hazard Anaysis). In zwei weiteren parallelen Arbeiten wird jedoch sowohl die STAMP Methode Causal Analysis using System Theory (CAST), von Zimmermann [Zim20] auf Basis von Levesons *CAST Handbooks* [Lev19] zur Unfall-/Zwischenfall Analyse (Accident and Incident Analysis), als auch eine Erweiterung der STPA Analyse um ein Model Checking Verfahren, von Held [Hel20], in die Einzelplatzanwendung integriert.

1.2 Relevanz des Themas & Motivation

Es gibt zahlreiche Techniken der Gefahrenanalysen. Dabei ist stets das Kernziel Gefahren vorbeugend zu identifizieren, ihr Gefahrenpotenzial einzuordnen und diese Gefahren gänzlich zu verhindern oder sie, bei einem späteren Auftreten, entsprechend zu kontrollieren. Dadurch soll die

Eigenschaft der System Safety erreicht werden. Ericson liefert in seinem Buch *Hazard Analysis Techniques for System Safety* [Eri+15], zu traditionellen Gefahrenanalysetechniken, eine mehr als detaillierte Aufführung und Einteilung.

Die Notwendigkeit zur System Safety wächst mit dem immer schnelleren Aufkommen neuer und zunehmend komplexer Technologien. Gefahren für Mensch und Umwelt sollten nicht mit ihrem Auftreten, sondern schon proaktiv während des eigentlichen Entwicklungsprozesses erkannt werden.

STAMP kann mit der Gefahrenanalysemethode STPA, nach Leveson [Lev16] gegenüber traditioneller Ansätze, auch bei komplexen Systemen Anwendung finden. Mit der Entwicklung einer Software zur Anwendung von STAMP Methoden bietet sich damit, auch im stetigen Wandel der Technik, ein robustes Werkzeug System Safety bereits zur Konzeption und Entwicklung neuer Systeme auszubauen.

1.3 Zielsetzung

Ziel dieser Arbeit ist die Umsetzung von XSTAMPP 4, und der eingesetzten softwaregestützten STPA Analysemethode, als offlinefähige Einzelplatzanwendung. Darüber hinaus sollen auch, wie in vorheriger Arbeiten zu XSTAMPP, in einem fortlaufenden Entwicklungsprozess die Anwenderfreundlichkeit ausgebaut und Funktionen verbessert und erweitert werden.

Auch sollen, zum Zweck der Erweiterung der Einzelplatzanwendung durch parallelen Arbeiten, geteilt nutzbare Komponenten und Funktionen bereitgestellt werden.

1.4 Aufbau der Arbeit

Folgende Auflistung liefert einen Gesamtüberblick der nächsten Kapitel und soll dem Leser Aufbau und Struktur dieser Arbeit zeigen.

Kapitel 2 - Grundlagen Zu Beginn werden alle notwendig Grundlagen erläutert. Im Besonderen wird STAMP und die Gefahrenanalysemethode STPA präzisiert. Anschließend wird die Webanwendung XSTAMPP 4 und deren softwaregestützte STPA Analysemethode vorgestellt, welche als Basis für die zu entwickelnde Einzelplatzanwendung dient.

Kapitel 3 - Verwandte Arbeiten Zur Einordnung dieser Arbeit in den Entwicklungsprozess von XSTAMPP werden einige verwandte Arbeiten zu diesem Thema vorgestellt. Dazu wird auch auf die Notwendigkeit eingegangen und damit welche bestehende Lücke mit dieser Arbeit geschlossen werden soll.

Kapitel 4 - Analyse und Entwurf In diesem Kapitel werden die wichtigsten Anforderungen an die zu entwickelnde Software aufgelistet. Anschließend wird der Entwurf zur Umsetzung der architektonischen Kernanforderungen vorgestellt.

Kapitel 5 - Implementierung Hier werden die Umsetzung des Entwurfs und im Entwicklungsprozess zusätzlich implementierte Funktionsmerkmale vorgestellt. Ebenso werden verwendete Prinzipien und Guidelines aufgeführt, die bei der Implementierung eingesetzt wurden.

Kapitel 6 - Anwendungsbeispiel In einem Anwendungsbeispiel wird die Durchführung einer STPA Analyse in der entwickelten Einzelplatzanwendung demonstriert.

Kapitel 7 - Diskussion In diesem Kapitel werden einige der umgesetzten Funktionsmerkmale diskutiert und bewertet. Insbesondere sollen hier auch Änderungen an existierenden Elementen von XSTAMPP 4 begründet werden.

Kapitel 8 - Zusammenfassung und Ausblick Abschließend werden zentrale Punkte der Arbeit noch einmal zusammengefasst. Auch wird hier ein Ausblick mit weiteren Ideen für zukünftige Arbeiten gegeben.

2 Grundlagen

2.1 STAMP

Von Leveson entwickelt und in ihrem Buch *Engineering a Safer World: Systems Thinking Applied to Safety* [Lev16] vorgestellt, nutzt STAMP einen systemtheoretischen Ansatz zur Betrachtung von *Safety* innerhalb komplexer soziotechnischer Systeme bestehend aus Mensch, Umwelt und Technik. *Safety* ist dabei der Schutz vor aller Art möglicher *Accidents* (Unfälle), die innerhalb eines solchen Systems auftreten können. Leveson nutzt zur Konkretisierung von *Accidents* hierzu den zentralen Begriff eines *Loss* (Verlusts). Jeder *Accident* hat zwangsläufig einen *Loss* zur Folge.

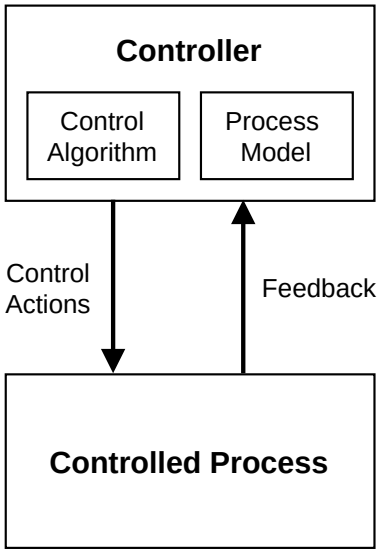

Abbildung 2.1: Generische Kontrollschleife aus dem STPA Handbuch [LT19]

In STAMP, mit seinem ganzheitlichen systemtheoretischen Ansatz, entstehen *Accidents*, und damit *Losses*, durch externe Störeinflüsse, Systemkomponentenausfälle oder durch eine dysfunktionale Kommunikation zwischen Systemkomponenten bzw. dysfunktionale Verhalten individueller Systemkomponenten, welche das System in einen gefährlichen Systemzustand bringen. Zur Vermeidung von *Losses* wird *Safety* in STAMP dazu als Kontrollproblem innerhalb eines dynamischen Systems betrachtet. STAMP fußt dazu auf den folgenden drei Konzepten:

Safety Constraints sollen garantieren, dass ein System nicht in einen gefährlichen Systemzustand kommen kann. Folglich entstehen *Losses* nur durch mangelhafte Umsetzung oder Abwesenheit von *Safety Constraints*. *Safety Constraints* können dabei durch automatische Systemvorgänge erzwungen werden, oder aber durch den Menschen selbst.

Hierarchische Safety Control Structure Innerhalb einer hierarchischen Kontrollstruktur garantiert, in einem top-down Ansatz, jeder sogenannte *Controller*, Mensch oder Maschine, einer in der Hierarchie übergeordneten Ebene, dass alle notwendigen *Safety Constraints* in der folgenden Ebene umgesetzt werden. Die Kommunikation erfolgt mittels gegebenen *Control Actions* und einem resultierenden *Feedback* zwischen einem übergeordneten *Controller* und einem *Controlled Process* (vgl. Abbildung 2.1).

Process Models bilden hierbei die notwendige Grundlage eines jeden *Controllers*. Jeder *Controller* braucht ein Prozessmodell des *Controlled Processes*, um dessen Verhalten entsprechend bewerten und damit *Safety Constraints* garantieren zu können (vgl. Abbildung 2.1). Diesen können, z. B. mittels Variablen, basierend auf Regelmechaniken bewerten, welche Werte zulässig sind, bzw. wie auf Systemzustände zu reagieren ist. In diesem Kontext wird der Entscheidungsprozess, der auf Basis des *Process Models* agiert, auch als *Control Algorithm* bezeichnet.

STAMP selbst bietet keine Analysemethode, gibt aber den entsprechenden Rahmen für diese vor. Im Fokus einer solchen Analyse steht dabei die Frage, warum die existierenden Kontrollen, innerhalb der *Control Structure*, ein *Accident* nicht verhindern. Im folgenden Abschnitt wird dazu STPA als eine solche Gefahrenanalysemethode vorgestellt.

2.2 STPA

In diesem Abschnitt werden für diese Arbeit einige wesentliche Punkte des *STPA Handbook* [LT19] von Leveson und Thomas zusammengefasst.

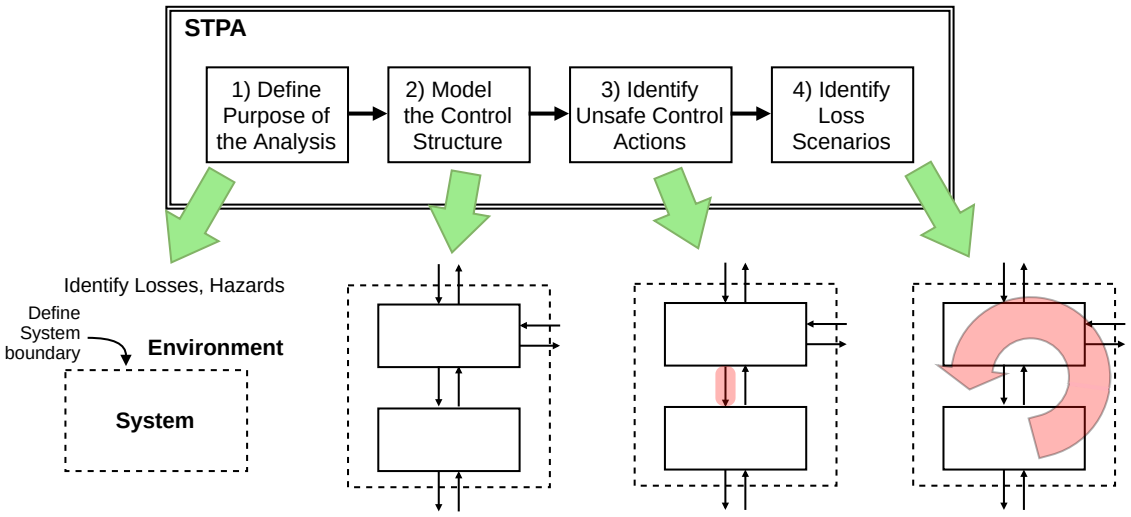

Abbildung 2.2: STPA Prozess aus dem STPA Handbuch [LT19]

Abbildung 2.2 aus dem *STPA Handbook* [LT19] von Leveson und Thomas zeigt die vier Schritte zur Durchführung einer STPA Gefahrenanalyse. Die Motivation und wichtige Definitionen der einzelnen Schritte werden hier näher beschrieben. Diese können auch im Kapitel 6 zum Anwendungsbeispiel der Einzelplatzanwendung mit dem Beispiel des *STPA Handbooks* [LT19] von Leveson und Thomas noch einmal nachvollzogen werden.

2.2.1 Schritt 1 - Definiere den Zweck der Analyse

Am Anfang jeder STPA Analyse steht die Festlegung des Zwecks der Analyse. Dies geschieht mit der Identifikation aller *Losses*, die mit der Gefahrenanalyse verhindert werden sollen. Leveson und Thomas [LT19] definieren *Losses* dazu wie folgt:

„A loss involves something of value to stakeholders. Losses may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the stakeholders.“ (Leveson und Thomas [LT19])

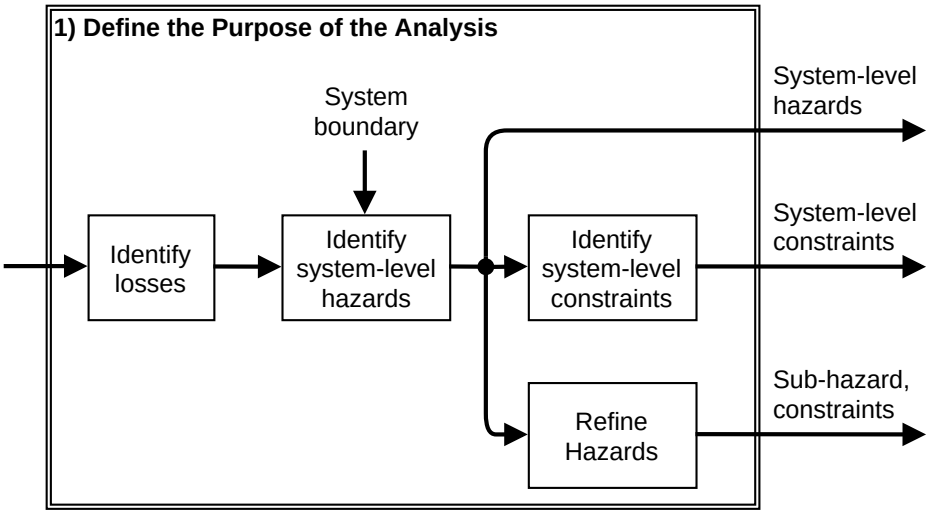

Abbildung 2.3: Definiere den Zweck der Analyse aus dem STPA Handbuch [LT19]

Anschließend werden *Hazards* identifiziert; konkreter *System-Level Hazards*. Für den Analysten sind nur solch *Hazards* relevant, wenn diese auch aktiv kontrolliert werden können. Dazu werden in diesem Schritt auch die Systemgrenzen festgelegt.

„A hazard is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss.“ (Leveson und Thomas [LT19])

„A system is a set of components that act together as a whole to achieve some common goal, objective, or end. A system may contain subsystems and may also be part of a larger system.“ (Leveson und Thomas [LT19])

Aus den *System-Level Hazards* lassen sich direkt die *System-Level (Safety) Constraints* ableiten. Diese enthalten die notwendigen Bedingungen und Verhaltensweisen, um einen *Hazard* zu verhindern. Abbildung 2.3 zeigt neben einer Zusammenfassung diesen Abschnitts auch die optionalen Schritte, um zusätzlich *Hazards* und *Safety Constraints* weiter zu verfeinern.

2.2.2 Schritt 2 - Modellierung der Control Structure

Im zweiten Schritt der Analyse wird die *Control Structure* des Systems modelliert. Dabei wird mit einer abstrakten Betrachtungsweise begonnen und sukzessive die *Control Structure* weiter ausgearbeitet. Ergänzend zu der Beschreibung in Abschnitt 2.1 modelliert eine *Control Structure* kein ausführbares Modell eines physischen Systems, weder spiegeln die Kontrollschleifen zwangsläufig die tatsächliche Kommunikation wider. Die *Control Structure* bildet hingegen ein funktionales Modell der notwendigen Kontrollschleifen, um *Safety Constraints* innerhalb des Systems umzusetzen.

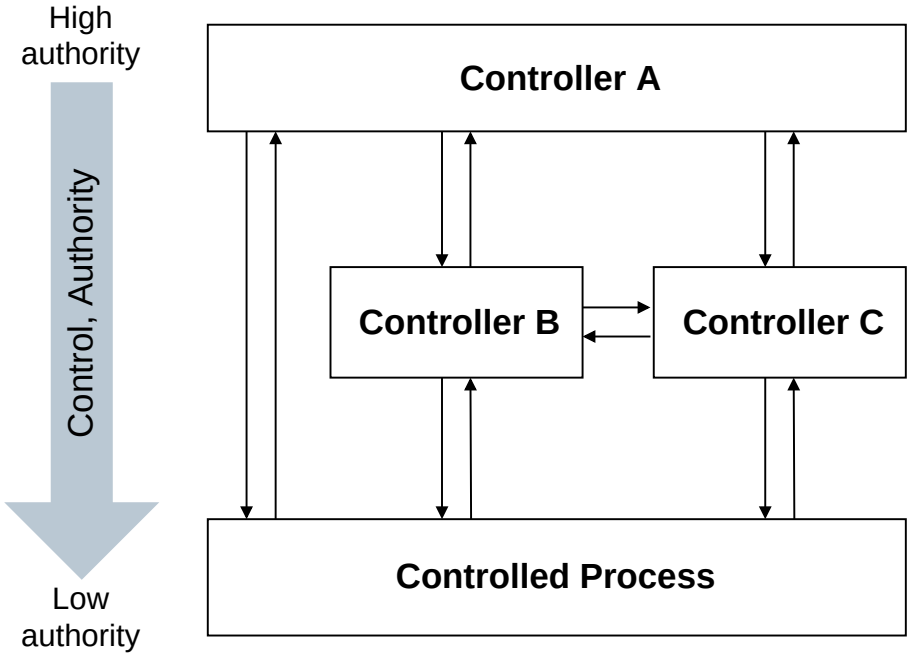

Abbildung 2.4: Generische Control Structure aus dem STPA Handbuch [LT19]

Die *Control Structure* wird iterativ immer weiter ausgearbeitet. In diesem Prozess werden neu identifizierten *Controllern* auch fortlaufend *Responsibilities* zugewiesen. *Responsibilites* verbinden einen *Controller* mit einer oder mehreren entsprechenden *Safety Constraints*, die dieser umzusetzen hat.

2.2.3 Schritt 3 - Unsafe Control Actions identifizieren

„An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard.“ (Leveson und Thomas [LT19])

Leveson und Thomas [LT19] führen vier Möglichkeiten an, mit denen *Control Actions* unsicher werden können.

- Das nicht Bereitstellen einer *Control Action* führt zu einem *Hazard*.
- Das Bereitstellen einer *Control Action* führt zu einem *Hazard*.
- Eine potenziell sichere *Control Action* wird gegeben, aber zu früh, zu spät oder in der falschen Reihenfolge.
- Die *Control Action* dauert zu lange an oder stoppt zu früh (im Falle von *Control Actions* mit stetigen Signalen).

Diese vier Möglichkeiten sind laut Leveson und Thomas [LT19] beweisbar vollständig und decken alle Szenarien zur Entstehung von Unsafe Control Action (UCA)s ab.

Sind UCAs identifiziert können *Controller Constraints* ausgearbeitet werden, die das konkrete Verhalten eines *Controllers* spezifizieren, um eine UCA zu verhindern (siehe Abbildung 2.5).

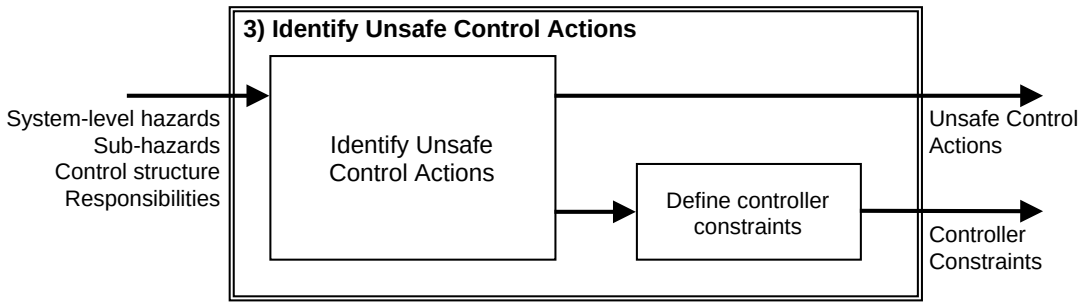

Abbildung 2.5: Control Action Analyse aus dem STPA Handbuch [LT19]

2.2.4 Schritt 4 - Loss Scenarios identifizieren

Im finalen Schritt 4 werden *Loss Scenarios* erarbeitet. Diese beschreiben die Kausalfaktoren, welche zu einer UCA führen. Leveson und Thomas [LT19] beschreiben dazu zwei Fragen die erwogen werden müssen. Warum sollte eine UCA auftreten? Und warum sollte eine *Control Action* unangemessen ausgeführt oder nicht ausgeführt werden? Dazu werden auch folgenden übliche Szenarien angegeben:

- Ausfälle bezogen auf den *Controller* (für physische *Controller*)
  - Physischer Ausfall des *Controllers* selbst

  - Stromausfall
  - ...
- Inadäquater *Control Algorithm*
  - Mangelhafte Implementierung des spezifischen *Control Algorithm*
  - Der *Control Algorithm* selbst ist fehlerhaft
  - Der *Control Algorithm* wird durch Änderungen oder Degeneration über die Zeit fehlerhaft
- Unsicherer *Control Input*
  - Es wird eine UCA von einem anderen *Controller* entgegengenommen
- Unpassendes *Process Model*
  - Der *Controller* erhält ein inkorrektes *Feedback*/Information
  - Der *Controller* erhält ein korrektes *Feedback*/Information, aber interpretiert dieses inkorrekt oder ignoriert es
  - Der *Controller* erhält kein *Feedback*/Information, wenn notwendig (verzögert oder niemals)
  - Notwendiges *Controller Feedback*/Information existiert nicht

Die *Control Structure* kann in diesem Schritt auch um *Actuators* und *Sensors* erweitert werden. *Actuators* geben den konkreten Prozess der Umsetzung einer *Control Action* wieder, mit der ein *Controller* auf einen *Controlled Process* Einfluss nimmt. Analog geben *Sensors* den konkreten Prozess vor, mit dem *Feedback* gemessen oder erkannt wird. *Actuators* und *Sensors* werden dazu in der *Control Structure* entsprechend ergänzt. Diese beiden Konzepte sind hilfreich, um in diesem Schritt den genauen Grund für das Auftreten eines *Loss Scenarios* zu identifizieren.

2.3 Webanwendung XSTAMPP 4

XSTAMPP 4 leitet und unterstützt einen Analysten bei der Durchführung einer STPA Analyse. Als Webanwendung mit einer Nutzerverwaltung bietet XSTAMPP 4 Analysten auch die Möglichkeit zum kooperative Arbeiten an gemeinsamen Projekten. Auch gibt es Funktionen, in denen sich Analysten zu Gruppen zusammenzuschließen um mehrere Projekte einer bestimmten Domäne einsehen oder verwalten zu können. Innerhalb solcher Gruppen können mittels Rechteverwaltung Zugriffsrechte vergeben werden. XSTAMPP 4 bietet momentan nur STPA als Analysemethode an. In der Analyse selbst wird der Analyst dabei durch alle Schritte von STPA geleitet und kann die notwendigen Einträge in Tabellen anlegen und verwalten. Hierzu wird der Analyst Schritt für Schritt durch den Analyseprozess geleitet.

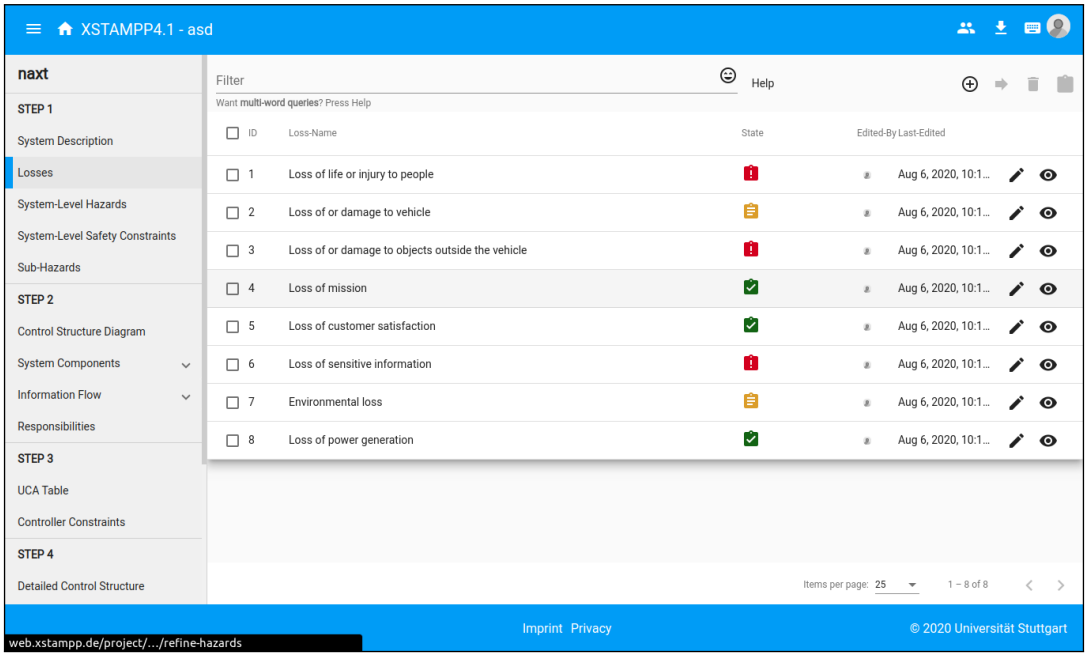

Abbildung 2.6: XSTAMPP 4 Loss Tabelle

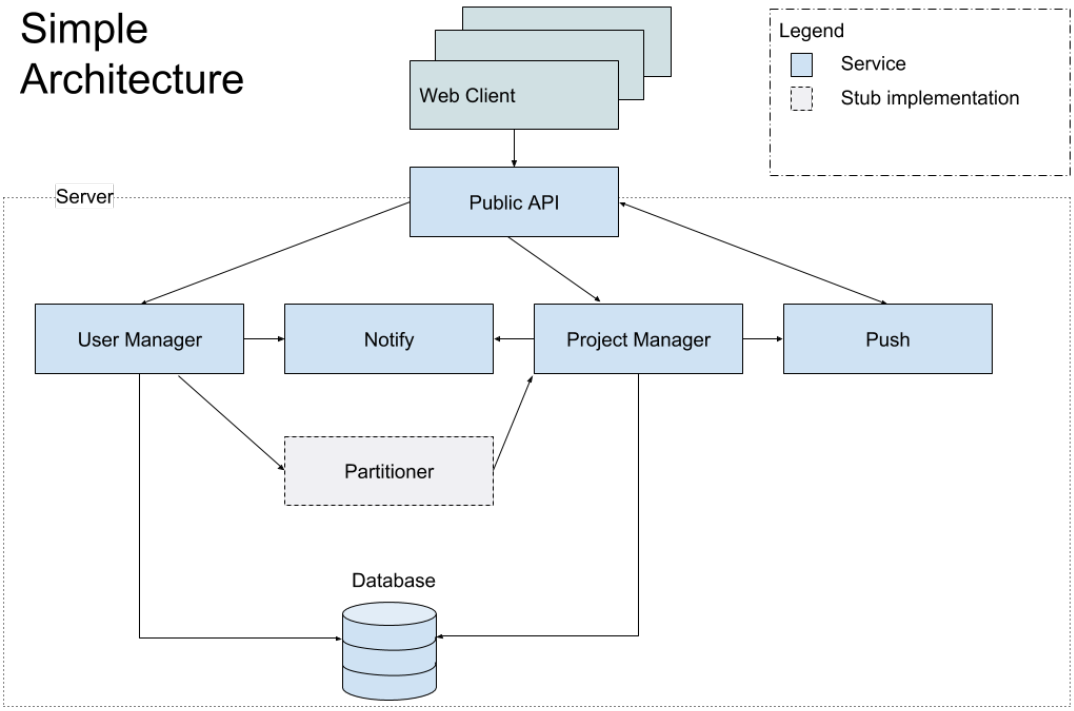

Abbildung 2.7: XSTAMPP 4 Architektur

Abbildung 2.6 zeigt beispielhaft, wie ein STPA Projekt mittels der Navigationsleiste (links im Bild) mit allen notwendigen Schritten und entsprechenden Unterschritten navigiert werden kann. Geöffnet ist die Tabelle zur Identifikation von *Losses* (die Einträge entsprechen dem späteren Anwendungsbeispiel). Eine weitere Kernfunktion von XSTAMPP 4 ist der grafische Editor zur Erstellung der C*ontrol Structure*. Komponenten und jeglicher Informationsfluss wird dabei ebenfalls in Tabellen abgelegt.

Die Architektur von XSTAMPP 4 (siehe Abbildung 2.7) nutzt das bekannte Webapplikationsframework Angluar[1] zur Entwicklung des XSTAMPP 4 Web-Frontends. Der Web-Client spricht via Public API (HTTP) mehrere Spring Boot[2] Web-Services an. Zur Persistierung wird eine PostgreSQL[3] Datenbank genutzt.

[1] https://angular.io/ (Typescript)
[2] https://spring.io/ (Java)
[3] https://www.postgresql.org/

3 Verwandte Arbeiten

In diesem Kapitel wird die vorliegende Arbeit in die Historie an vorangegangenen Arbeiten zu XSTAMPP eingeordnet. Dazu wird auch die Notwendigkeit dieser Arbeit herausgestellt und welche Lücke mit ihr gefühlt wird.

Im Jahr 2014 wurde das erste, an dem Institut für Software Engineering der Universität Stuttgart entwickelte, Werkzeug zur softwaregestützten Durchführung von STPA von Abdulkhaleq und Wagner [AW14] unter dem Namen Automated STPA (A-STPA) vorgestellt. Mit der Anwendung von A-STPA in verschiedenen Domänen der Industrie, und der Weiterentwicklung von STAMP, entstanden fortlaufend neue Anforderungen an das Werkzeug. Um diesen und auch zukünftigen Anforderungen mit einem beständigen Entwicklungsprozess begegnen zu können wurde auf Basis von A-STPA das erste XSTAMPP, von Abdulkhaleq und Wagner [AW15] als Open Source Werkzeug 2015 auf der Eclipse Plattform vorgestellt. Mit nachfolgenden Versionen, darunter XSTAMPP 2, vorgestellt von Abdulkhaleq und Wagner [AW16], wurden sowohl weitere Plugins, wie Extended Approach to STPA (XSTPA), als auch die jüngere Analysemethode Automated CAST (A-CAST) integriert. Auch wurden mit zahlreichen folgenden Arbeiten XSTAMPP bis zur Version 3[1] um neue Funktionen erweitert.

Mit XSTAMPP 4[2] wurde 2019 erstmals nicht die bestehende Eclipse Version erweitert, sondern eine Neuauflage von XSTAMPP als Webanwendungen umgesetzt. Mit diesem Entwicklungsschritt bietet XSTAMPP heute mit der neusten STPA Version eine moderne Arbeitsplattform zum kooperativen Arbeiten an STPA Projekten. Für den breiten Einsatz in der Industrie und Forschung ist in einigen Anwendungsfällen eine rein webbasierte Nutzung von XSTAMPP 4 hinderlich. Auch ist, trotz der Möglichkeit XSTAMPP 4 auf betriebsinterner Infrastruktur anzubieten, ein eigenes Bereitstellen der XSTAMPP 4 Umgebung für kleinere Unternehmen ein möglicher abschreckender Faktor. Hier soll eine offlinefähige Einzelplatzanwendung von XSTAMPP 4, welche in dieser Arbeit umgesetzt wird, diese elementare Lücke in der Anwenderfreundlichkeit von XSTAMPP 4 schließen.

[1] https://github.com/SE-Stuttgart/STPA-Safety-based-Testing-Plugin
[2] https://web.xstampp.de/

4 Analyse und Entwurf

Die Anforderungsanalyse findet auf Basis von XSTAMPP 4 statt. In diesem Abschnitt werden dazu die Funktionsmerkmale, im weiteren Features genannt, in formloser Textform aufgeführt und beschrieben. Die Features sind dabei eine Sammlung mehrerer thematisch zusammenhängender funktionaler und nicht funktionaler Anforderungen, die zu Beginn der Entwicklungsarbeit an die Einzelplatzanwendung gestellt wurden. Anschließend wird der entwickelte Architekturentwurf vorgestellt.

4.1 Anforderungen auf Grundlage von XSTAMPP 4

Nicht alle Features von XSTAMPP 4 sind relevant für eine Umsetzung als Einzelplatzanwendung. Features zur Zugriffsrechte und Nutzerverwaltung sind in einem Offlinekontext nicht notwendig. Auch die Persistierung in einer externen Datenbank ist nur bedingt sinnvoll. Aus diesen Gründen beschränken sich die folgenden Features auf Funktionen zur Projektverwaltung und zum STPA Analyseprozess von XSTAMPP 4.

[F-1] STPA Analyseprozess

Alle Schritte zur Durchführung des Analyseprozesses von XSTAMPP 4 müssen auch in der Einzelplatzanwendung umsetzbar sein. Das umfasst sowohl die vier STPA Schritte als auch alle Erweiterungen dieser Schritte. Form, Gestaltung und Umfang sollen dabei erhalten bleiben. Anpassungen zur Anwenderfreundlichkeit sind möglich, sofern sie sich nicht zu weit von der Webanwendung entfernen. Der Analyst muss alle bekannten Funktionen wiedererkennen; konkreter darf dem Analysten kein größerer Umlernprozess zugemutet werden.

[F-2] Import und Export von XSTAMPP 4 Projektdateien

XSTAMPP 4 bietet neben der Persistierung von Projektdateien, innerhalb einer Datenbank, auch die Möglichkeit zum Import und Export von Projektdateien im JSON Format[1]. Die Einzelplatzanwendung muss ebenfalls über einen Import und Export des XSTAMPP 4 Formats für Projektdateien verfügen. Änderungen an der Projektstruktur, durch Erweiterungen der Einzelplatzanwendung, dürfen die Abwärtskompatibilität zu diesem Format nicht verletzen.

[1] https://www.json.org/json-en.html JavaScript Object Notation

[F-3] PDF Export von Projekten

Zur Dokumentation und Präsentation können in XSTAMPP 4 STPA Projekte als PDF-Datei exportiert werden. Die Einzelplatzanwendung soll ebenfalls über eine solche Funktion verfügen. Diese soll ggf. auch zusätzliche Erweiterungen der Einzelplatzanwendung beinhalten.

4.2 Zusätzliche Anforderungen

[F-4] Portabilität der Einzelplatzanwendung

Die Einzelplatzanwendung soll auf den Betriebssystem Microsoft Windows, macOS und Linux einsetzbar sein.

[F-5] Bereitstellung bzw. Ergänzung der User-Dokumentation

Sollten Änderungen oder Ergänzungen der Einzelplatzanwendungen eine User-Dokumentation benötigen, muss diese von der Einzelplatzanwendung in angemessener Form angeboten werden.

[F-6] Bereitstellung bzw. Ergänzung der Dokumentation zur Wartung

In einem fortlaufenden Entwicklungsprozess, mit wechselnden Mitwirkenden, muss die Einstiegshürde möglichst gering sein. Dazu soll das Projekt mit Dokumentation zur Wartung und Erweiterung der Einzelplatzanwendung versehen werden.

[F-7] Bereitstellung eines Installationsprozesses aus der Webumgebung XSTAMPP 4

Die Einzelplatzanwendung soll auf der XSTAMPP 4 Webseite angeboten werden und, falls notwendig, mit entsprechender Dokumentation zur Installation versehen werden. Der Installationsprozess selbst sollte dabei möglichst unkompliziert sein.

[F-8] Funktionen zur Erweiterung um CAST und ein Model Checking Verfahren

Zur Unterstützung zweier paralleler Arbeiten zur Erweiterung der Einzelplatzanwendung (siehe [Zim20] und [Hel20]) sollen entsprechende Schnittstellen und Funktionsklassen zur gemeinsamen und einheitlichen Nutzung bereitgestellt werden.

4.3 Architektur

In diesem Abschnitt werden neben dem Entwurf selbst auch der Entscheidungsprozess zur Lösungsfindung beleuchtet. Auch werden alternative Ansätze besprochen, die im Entwurfsprozess nicht eingeschlagen wurden.

4.3.1 Grundlage

Mit der Entwicklung der Einzelplatzanwendung soll keine vollständige Neuentwicklung der Software stattfinden. Daher ist es sinnvoll alle Möglichkeiten der Wiederverwendung der bereits existierenden Software aus XSTAMPP 4 gegeneinander abzuwägen. Wie im Grundlagenkapitel bereits erwähnt besteht die Architektur von XSTAMPP 4 aus drei Kernkomponenten: einem Angular Web-Frontend, mehreren Spring Boot Web-Services zur Bereitstellung der Businesslogik und einer PostgreSQL Datenbank zur Persistierung der Daten.

4.3.2 Mögliche Ansätze

Wenn dies auch durchaus möglich wäre, so ist es eigentlich nicht zumutbar die einzelnen Web-Services von XSTAMPP 4 auf einem Anwendercomputer einzusetzen (siehe F-7). Weiterhin sind, wie im vorangehenden Kapitel 4 zur Anforderungsanalyse erwähnt, nicht alle diese Funktionen relevant. Auch ist die persistente Speicherung von Projekten innerhalb einer Datenbank im Kontext einer Einzelplatzanwendung nicht zwangsläufig sinnvoll, da Projekte auch im Dateisystem des Betriebssystems abgelegt werden können. Es sollten daher Möglichkeiten erwogen werden, mit denen sich die ersten beiden Komponenten innerhalb einer Software zusammenlegen lassen.

Die größte Wiederverwendung von bestehender Software würde sich erreichen lassen, wenn man in einer Java Applikation den bestehenden Web-Client innerhalb eines Web-Containers etablieren würde. Dann könnte, nach einmaliger Anpassung, die Businesslogik zur Projektverwaltung sowohl in der Einzelplatz- als auch in der Webanwendung eingesetzt werden. Wenn auch denkbar ist diese Vorgehensweise eher unüblich und folglich wenig dokumentiert und unterstützt. Auch ist der Aufwand dieser einmalig notwendigen Anpassung schwer abschätzbar.

Der hier gewählte Ansatz ist die Zusammenlegung innerhalb eines Web-Container-Frameworks, bestehend aus einer Browser-Rendering-Engine und einer NodeJS Laufzeitumgebung. Dieser Ansatz ist auch in der Industrie sehr populär. Hier lassen sich bewährte Webtechnologien auch in der Entwicklung von Einzelplatzanwendungen einsetzen. Nachteil ist, dass die Businesslogik in JavaScript oder einer NodeJS kompatiblen Programmiersprache neu geschrieben werden muss. Hierbei handelt sich bei der benötigten Businesslogik aber um einen eher überschaubaren Aufwand. Auch bietet dieser Ansatz größere Freiheiten. Einige Entwicklungsentscheidungen aus XSTAMPP 4 können neu getroffen werden. Dadurch werden Anpassungen an Anforderungen einer Einzelplatzanwendung möglich (siehe F-8).

4.3.3 Architekturentwurf

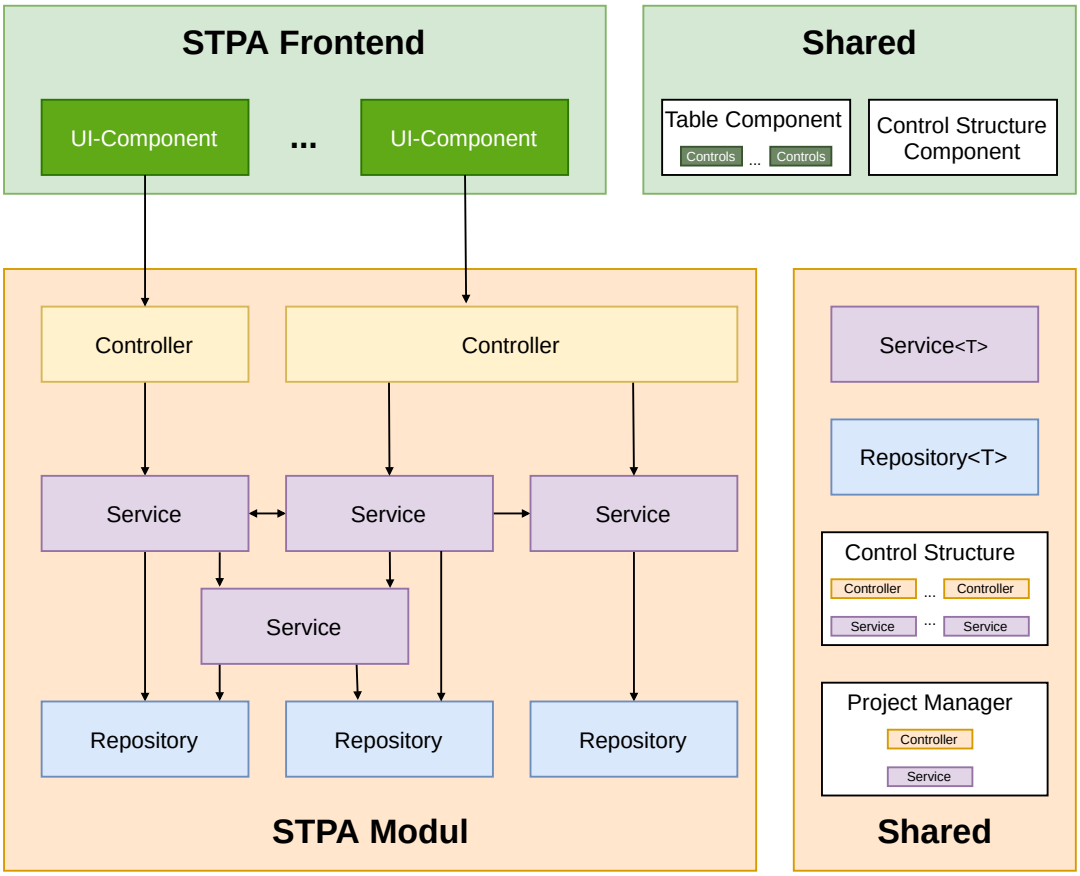

Abbildung 4.1: Architekturentwurf der Einzelplatzanwendung

Die Architektur des Frontends ist durch das Angularframework[2] in weiten Teilen bereits vorgegeben. Bereits bestehende XSTAMPP 4 UI-Komponenten (User Interface) können bei Bedarf angepasst oder direkt wiederverwendet werden (siehe Abbildung 4.1). Auch der Entwurf des neu zu entwickelnden STPA Moduls orientiert sich an der 3-Schichtenarchitektur der XSTAMPP 4 Spring Boot Web-Services. Der Entwurf des STPA Moduls in Abbildung 4.1 besteht aus den folgenden Schichten:

Präsentationsschicht Diese sehr dünne Schicht bestehen aus mehreren Controllern und bündelt einen oder mehrere Services der Businesslogikschicht für die Nutzung durch eine UI-Komponente. Auch wird durch diese Schicht ein direkter Zugriff auf Services reguliert.

Businesslogikschicht Die Logikschicht besteht aus einer Reihe von Services für alle benötigten Verarbeitungsmechanismen. Auch bildet diese Schicht, falls notwendig, Datenbankobjekte auf Transferobjekte ab, die vom Frontend benötigten werden.

[2] https://angular.io/guide/architecture

Persistenzschicht Services beziehen mittels Repositories Datenbankobjekte aus der Persistenzschicht. Für jedes Datenbankobjekt existiert ein entsprechendes Repository mit Funktionen zum neu Erstellen, Schreiben, Lesen und Entfernen von Datenbankobjekten aus der Datenbank.

Um eine Erweiterung der Einzelplatzanwendung zu ermöglichen, im Speziellen durch einen zusätzlichen CAST Analyseprozess, sind geteilte Komponenten, sowohl für das STPA Frontend als auch für das STPA Modul, in eigene Shared Module ausgelagert. In Abbildung 4.1 sind dazu beispielhaft Interfaces für Services und Repositories aufgeführt. Auch wurden Komponenten eingeführt zur übergreifenden STAMP Projektverwaltung und die Nutzung der *Control Structure* innerhalb eines STAMP Moduls oder des Frontends, sowie eine generische Tabelle mit allen notwendigen Eigenschaften zur übergreifenden Nutzung.

5 Implementierung

In diesem Kapitel werden allgemeine Entwicklungsprinzipien und -entscheidungen zu umgesetzten Funktionalitäten der Einzelplatzanwendung, aber auch Änderungen und Erweiterungen beschrieben, die bei der Entwicklung zusätzlich aufgekommen sind. Auch werden Details zur Implementierung und verwendete Technologien näher erläutert.

5.1 Allgemein angewendete Prinzipien und Guidelines

Inversion of Control (IoC) oder auch Dependency Injection (DI) ist eines der stärksten Umsetzungsparadigma der objektorientierten Programmierung und findet auch bei der Entwicklung der XSTAMPP 4 Einzelplatzanwendung Anwendung. Angular nutzt DI fest integriert in seinem Framework, aber auch bei der Entwicklung des STPA Moduls wird dieses Prinzip eingesetzt. Hierzu wird die Bibliothek InversifyJS[1] verwendet. Mittels Annotationen lassen sich Abhängigkeiten zwischen Klassen zur Laufzeit automatisch auflösen. Wird im Listing 5.1 die Klasse *LossService* zur Laufzeit initialisiert, werden die notwendigen Repositories *ProjectRepo*, *LastIdRepo* und *LossRepo* mit der Annotation *@inject* automatisch rekursiv mit deren Abhängigkeiten erstellt. Auch ist der Service selbst durch *@injectable* unter seinem Klassennamen als injizierbar markiert.

Listing 5.1 Beispiel zur Dependency Injection

```
@injectable()
export class LossService extends Service<Loss> {
  constructor(
    @inject(ProjectRepo) projectRepo: ProjectRepo,
    @inject(LastIdRepo) lastIdRepo: LastIdRepo,
    @inject(LossRepo) private readonly lossRepo: LossRepo
  ) {
    super(Loss, projectRepo, lossRepo, lastIdRepo);
  }
}
```

Don't Repeat Yourself (DRY) Für einfache Entwicklungsarbeit und zur einfachen Erweiterung durch andere Mitwirkende ist eine einheitliche Programmstruktur elementar. Daher sind, wo möglich und sinnvoll, abstrakte Klassen oder Interfaces in einem Shared Modul angelegt (siehe Abbildung 4.1).

[1] http://inversify.io/

Listing 5.2 Beispiel zur Nutzung von Interfaces und abstrakten Klassen

```
@injectable()
export class ControllerConstraintRepo extends Repository<ControllerConstraint> {
  constructor(@inject(StpaDBConnector) dbConnector: StpaDBConnector) {
    super(ControllerConstraint, StpaBaseQuery.entityDependent, dbConnector);
  }

  public async removeAllByUnsafeControlActionId(
    projectId: string,
    controlActionId: number,
    unsafeControlActionId: number
  ): Promise<ControllerConstraint[]> {
    const collection = await this.getCollection();
    return collection
      .find()
      .where('projectId')
      .eq(projectId)
      .where('parentId')
      .eq(controlActionId)
      .where('id')
      .eq(unsafeControlActionId)
      .remove();
  }
}
```

Listing 5.2 veranschaulicht wie eine Reihe einheitlicher Methoden (*find(), update(), remove(),* . . .) der generischen abstrakten Klasse Repository geerbt werden können, aber auch ein konkretes Repository zusätzlich um eigene kontextbezogene Funktionen erweitert werden kann.

Diese Shared Modules für Frontend und STAMP Module bieten damit fundamentale STAMP Funktionen und einfache und einheitlich zu nutzende Entwicklungshilfen (siehe F-8), wie sie besonders für die Erweiterung der Einzelplatzanwendung um einen CAST Prozess durch Zimmermann [Zim20], notwendig sind.

5.2 Basis Framework

Für die Umsetzung der Einzelplatzanwendung wird das Open-Source Framework Electron[2] verwendet. Mit Electron lassen sich plattformübergreifend Einzelplatzanwendungen entwickeln, welche ohne Installationsprozess aufseiten des Nutzers in direkt ausführbare Programme kompiliert werden können. Erfüllt werden damit insbesondere F-4, zur Portabilität, und ein Teil von F-7, zum Installationsprozess. Bekannte Electronanwendungen sind Microsoft Teams, der Facebook

[2] https://www.electronjs.org/

Messenger und Slack. Electron nutzt Chromium[3] mit einem Browserprozess zur Ausführung der Anwendungslogik und mehreren Renderingprozessen, für jedes angezeigte Fenster. Damit lassen sich übliche Webtechnologien, wie HTML, CSS und JavaScript, verwenden und damit auch die bereits bestehenden UI-Komponenten des XSTAMPP 4 Web-Clients. Nachteil von Electron ist die reduzierte Performanz im Vergleich zu nativen Applikationen. Auch ist die initiale Paketgröße von Electron mit über 100 MB störend. Nichtsdestotrotz hat sich Electron in der Industrie bewährt und stellt eine detaillierte Dokumentation zur Verfügung.

Entwickelt wird in der Programmiersprache TypeScript[4]. TypeScript lässt sich direkt in JavaScript kompilieren, bietet aber, anders als JavaScript, auch bei wachsenden Projekten, mit mehreren wechselnden Entwicklern, die Möglichkeit, durch unter anderem Typisierung und statische Codeprüfungen, wartbaren und weniger fehleranfälligen Programmcode zu erstellen.

5.3 STPA Analyseprozess

In der Einzelplatzanwendung können gleichzeitige mehrere STAMP Projekte (STPA/CAST) geöffnet werden (siehe Abschnitt 5.4 zur Anpassung der Benutzeroberfläche). Wird ein STPA Projekt neu erstellt, oder ein existierendes Projekt geöffnet, erreicht man die gewohnte Ansicht wie man sie in XSTAMPP 4 vorfindet (siehe F-1). Im Kapitel 6 zum Anwendungsbeispiel wird exemplarisch eine STPA Analyse demonstriert.

Im folgenden Abschnitt werden, zum Zweck der geteilten Nutzung (siehe F-8) durch CAST, und zur Verbesserung der Anwenderfreundlichkeit (siehe F-1) einige Änderungen an den Komponenten der STPA Anaylse behandelt.

5.4 Anpassung der Benutzeroberfläche

Die Tabelle in XSTAMPP 4 (siehe Abbildung 5.1) wurde häufig für den entsprechenden Anwendungszweck maßgearbeitet. Das macht eine generische Nutzung der Tabelle, besonders im Kontext von der Erweiterung der Einzelplatzanwendung um CAST, unnötig kompliziert und zeitaufwendig. Auch staucht das *Detail Sheet*, mit dem Tabelleneinträge bearbeitet werden (siehe Abbildung 5.1), die Tabelle, was bei weiteren Elementen innerhalb der Tabelle visuell irritiert. Deshalb wurde eine generische Tabelle mit allen notwendigen Funktionen erstellt, um alle STPA und CAST Anwendungszwecke abzudecken (siehe F-8). Abbildung 5.2 zeigt die neue übersichtlichere Tabelle, welche beliebig um eine Reihe von erstellten Controls, z. B. zur Texteingabe, erweitert werden kann, wie diese in Kapitel 6 zum Anwendungsbeispiel ausgiebiger zu sehen sind. Änderungen müssen in der Einzelplatzanwendung auch nicht mehr zusätzlich bestätigt werden.

[3] https://www.chromium.org/
[4] https://www.typescriptlang.org/

Geöffnete Projekte werden in der Einzelplatzanwendung nicht durch eine Projektverwaltung, wie in XSTAMPP 4, mit je einem gleichzeitig geöffneten Projekt navigiert, sondern durch Projekt-Tabs am oberen Fensterrand mit beliebig vielen geöffneten STPA oder CAST Projekten (siehe Abbildung 5.2).

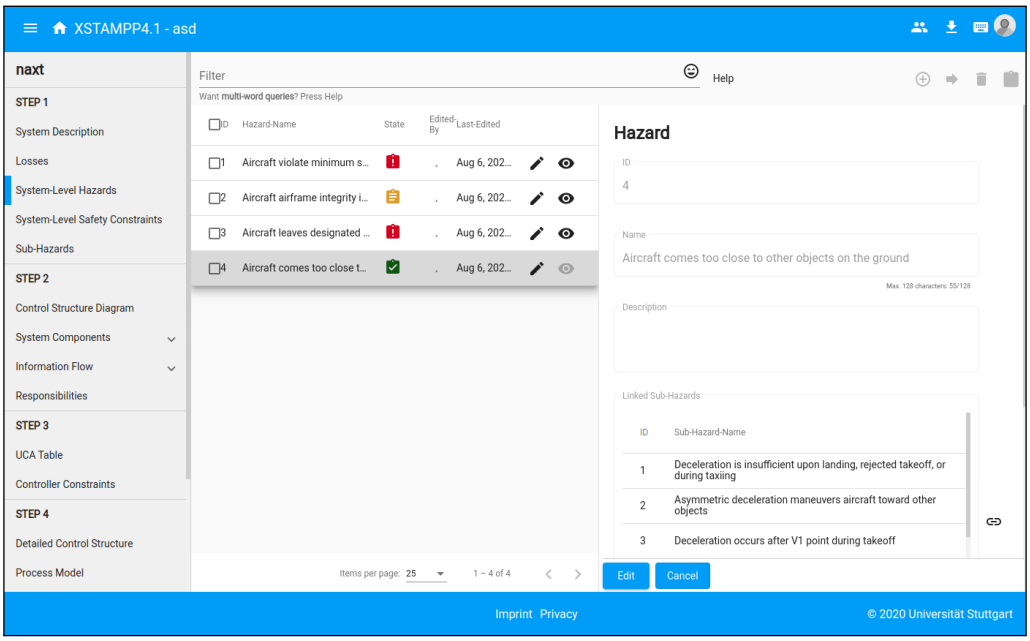

Abbildung 5.1: XSTAMPP 4: Detail Sheet

Die Erstellung von *Loss Scenarios* wurde von einem großen Dialogfenster in drei einzelne Schritte innerhalb einer Tabellenspalte aufgeteilt. Diese können nacheinander durchlaufen werden und sollen den Analysten bei diesem Schritt besser unterstützen. Eine Demonstration dieser Funktionalität wird im Abschnitt 6.4 des Anwendungsbeispiels gegeben.

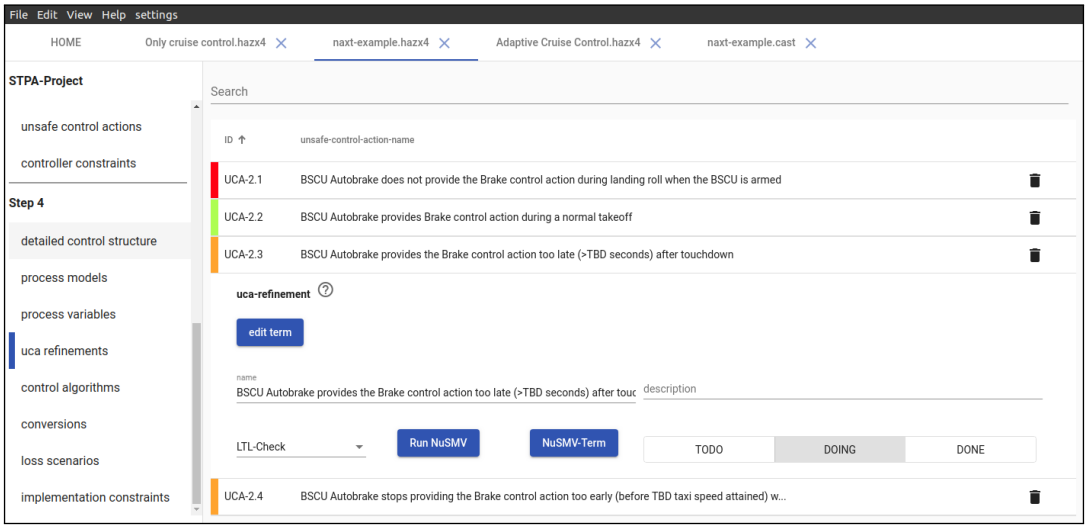

Abbildung 5.2: Einzelplatzanwendung: Integriertes Detail Sheet

5.5 Speicherung von Projekten

Die Anforderungen durch F-2, zum Import und Export von STAMP Projekten, werden in der Einzelplatzanwendung abweichend von XSTAMPP 4 gelöst. Geöffnete Projekte werden in der Einzelplatzanwendung in einer Reactive Database (RxDB)[5] gehalten (via IndexedDB Adapter). Das ermöglicht, dass beim erwünschten oder unerwünschten Schließen der Einzelplatzanwendung keine Projektdaten verloren gehen und beim nächsten Start der Anwendung wieder erscheinen. Bei der Speicherung von Projekten via *File* → *Save/Save As...*, oder durch einen Dialog beim Schließen von Projekt-Tabs, können Projekte im jeweiligen Format als Datei im System abgelegt werden. Die verwendete Dateiendung für STPA Projekte ist *.hazx4*. Damit können Projekte aus der Webanwendung XSTAMPP 4 auch voll kompatibel in der Einzelplatzanwendung geöffnet werden. Die Einzelplatzanwendung speichert zusätzliche Informationen aus der Erweiterung des STPA Prozesses, um z. B. das Model Checking Verfahren, auch in diesem Format. Beim Import in XSTAMPP 4 werden diese ignoriert und gehen verloren.

5.6 Zusätzliche Funktionen

Sub Safety Constraints

Die Verfeinerung der *Safety Constraints* in *Sub Safety Constraints* wurden in einen separaten Unterschritt innerhalb von Schritt 1 verlagert. Zuvor wurden *Sub Safety Constraints* direkt in der *Sub Hazard* Ansicht angelegt und verwaltet. Damit soll dieser Schritt für den Analysten besser sichtbar und verwaltbar sein.

Zusammenlegung von System- und Informationsflusskomponenten

Für System- und Informationsflusskomponenten der *Control Structure* gibt es in XSTAMPP 4 für jeden Typen jeweils eine Tabelle. Diese wurden in zwei Tabellen für System- und Informationsflusskomponenten zusammengelegt. Damit wird dem Analysten eine bessere Übersicht zum Anlegen und Verwalten aller *Control Structure* Komponenten gegeben.

Erweiterung der Regelalgorithmen zu STPA

Die im Rahmen der Masterarbeit von Heck [Hec19] implementierten Funktionen für Regelalgorithmen wurden unter dem Schritt zu *Conversions* für *Actuators* auch um *Conversions* für *Sensors* erweitert.

[5] https://rxdb.info/

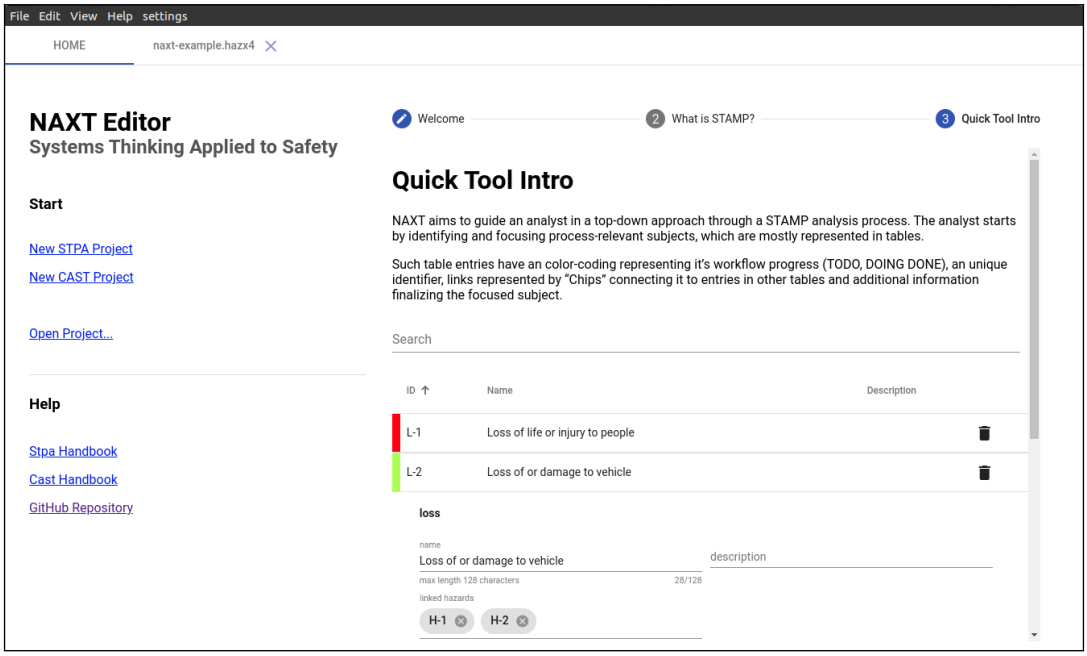

Abbildung 5.3: Nutzerdokumentation

5.7 Dokumentation

Zur Bereitstellung von Nutzerdokumentation (siehe F-7), im Rahmen von Änderungen der Benutzeroberfläche, werden im Startfenster der Einzelplatzanwendung (siehe Abbildung 5.3) zugrundeliegende Funktionen der Tabelle und von Tabellenelementen erklärt. Hierzu werden auch nutzbare Beispielkomponenten eingesetzt, wie z. B. eine funktionsfähige Tabelle in Abbildung 5.3.

Neben Programmkommentaren, wo notwendig, liegen generischen Komponenten oder Programmmodulen zusätzlich zur Entwicklerdokumentation (siehe F-6) Markdown[6]-Dateien bei. Diese sind in den meisten Entwicklungsumgebungen und grafischen Versionsverwaltungssystemen direkt einsehbar und sollen dem Entwickler grundlegende Konzepte begreifbar machen und Referenzen sowie Nutzungsbeispiele zur Verfügung stellen. In Abbildung 5.4 wird die in Abbildung 4.1 und Listing 5.2 verwendete abstrakte Klasse zur Erstellung von *Repositories* mit Referenzen zu verwendeten Technologien und an Nutzungsbeispielen erklärt.

[6] https://markdown.de/

RxDB Repository

RxDB is a real time Database for JavaScript Applications. You can read more about it here.
The abstract `Repository` class offers basic functionalities to **insert**, **update**, **get** and **remove** into the RxDB.

How to use

The `Repository` uses the takes a `Model`, a `BaseQuery` and a `dbConnector`. The lowercase **model name** is the corresponding **collection name** in the database provided by the dbConnector.

The `BaseQuery` is a function which creates a pouchdb mango query object with the **key** attributes to uniquely identify an entity in the database.

Example with custom function

```
@injectable()
export class SubRecommendationRepo extends Repository<SubRecommendation> {
  constructor(@inject(CastDBConnector) dbConnector: DBConnector) {
    super(SubRecommendation, CastBaseQuery.parentId, dbConnector);
  }

  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);
  }
}

public static parentId<T extends ProjectId & ParentIdString & IdString>
  ({ projectId, parentId, id }: T): EqSelector {
    return { selector: {
      projectId: { $eq: projectId },
      parentId: { $eq: parentId },
      id: { $eq: id }
    }};
}
```

Abbildung 5.4: Markdown Dokumentation

6 Anwendungsbeispiel

In diesem Kapitel wird als Anwendungsbeispiel das im *STPA Handbook* [LT19] von Leveson und Thomas verwendeten Beispiel in der Einzelplatzanwendung nachgestellt. Damit soll exemplarisch eine STPA Analyse mit den wichtigsten Kernelementen der entwickelten Einzelplatzanwendung demonstriert werden. Zur besseren Übersicht verfolgt das Beispiel nur einen Analysepfad entlang einer Brake System Control Unit (BSCU) als Bestandteil eines Flugzeugs der kommerzielle Luftfahrt. Es werden auch nicht alle Schritte einer STPA Analyse gezeigt oder besprochen, dennoch vermittelt das Beispiel auch die Kernpunkte einer STPA Analyse und kann ergänzend zu den Beschreibungen in Kapitel 2 gelesen werden.

6.1 Schritt 1 - Definiere den Zweck der Analyse

Identifiziere Losses Zu Beginn der Gefahrenanalyse werden *Losses* identifiziert. *Losses* werden dazu in einer Tabelle angelegt und können mittels des *description*-Felds weiter beschrieben werden. Ebenso werden später verlinkte *Hazards* angezeigt. In Abbildung 6.1 sind typische *Losses*, die ein Nutzer bei einer Analyse dabei in aller Regel vermeiden will.

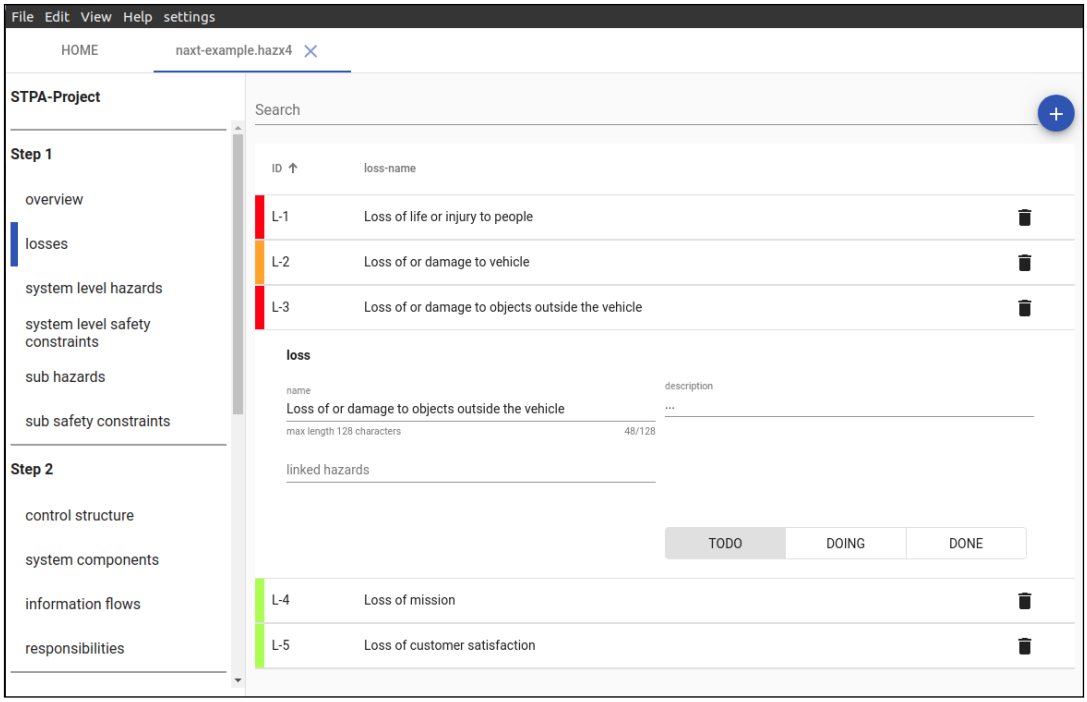

Abbildung 6.1: Anwendungsbeispiel: Identifiziere Losses

Definiere System-Level Hazards Sind System und Systemgrenzen festgelegt, können *System-Level Hazards* erstellt werden. Dazu werden wieder in einer Tabelle Einträge angelegt und je nach Bedarf detaillierter beschrieben. *System-Level Hazards* entstehen aus Systemzuständen und Bedingungen, die zu *Losses* führen. Daher werden hier zusätzlich auch die entsprechenden *Losses* als sogenannte Chips verlinkt. Diese Chips, mit der entsprechenden eindeutigen Identifikationsetikette (Label) bestehend aus Typkürzel und Informationsnummer (*L-1*), geben auch den Namen des hier verlinkten *Losses* wieder, wenn man sie mit dem Mauszeiger ansteuert. Auch können Verlinkungen wieder entfernt werden. Der Benutzer kann hier also zu jeder Zeit Änderungen vornehmen und muss sich nicht zu Beginn festlegen. Abbildung 6.2 demonstriert all diese Funktionen und darüber hinaus auch bereits Verlinkungen aus den nächsten Schritten. Diese werden erst in den folgenden Schritten erklärt, sind aber schon enthalten, um dem Beispiel eine anschließende einfachere Rückverfolgung der Verlinkungen zu ermöglichen.

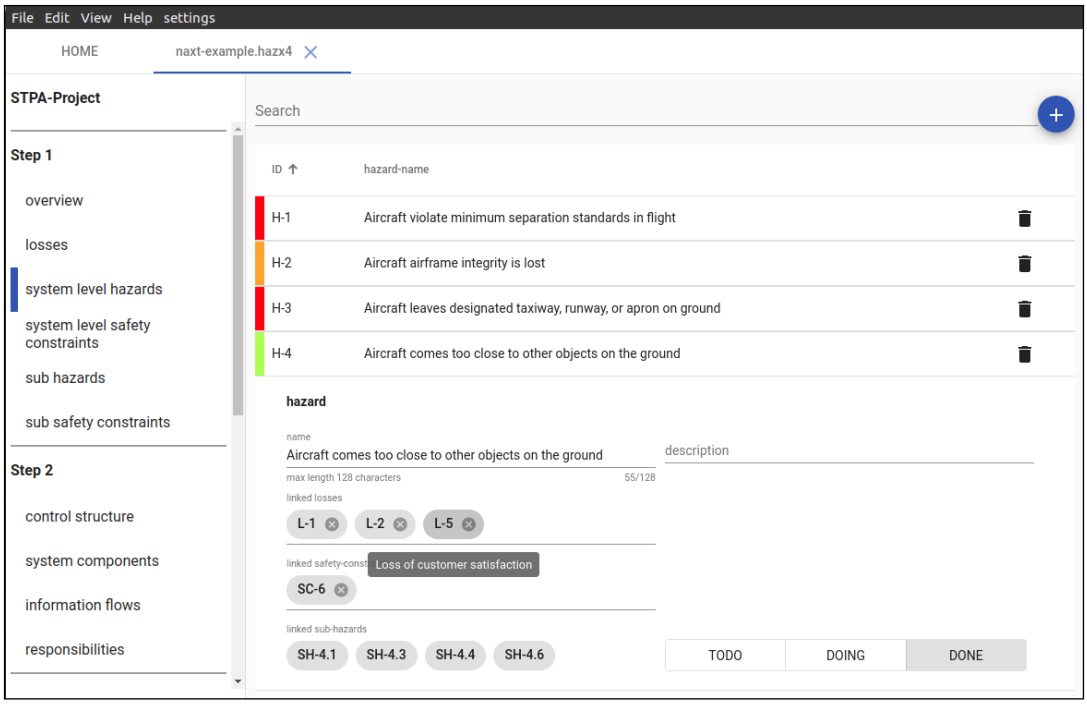

Abbildung 6.2: Anwendungsbeispiel: Definiere System-Level Hazards

Definiere System-Level Safety Constraints Zu jedem *System-Level Hazard* werden in diesem Schritt *System-Level Safety Constraints* erstellt, um diesen zu verhindern. In aller Regel genügt es die den *System-Level Hazard* zu negieren. In Abbildung 6.3 ist für *H-4* eine entsprechende *System-Level Safety Constraint* angelegt. *System-Level Safety Constraints* können, wie fast alle Tabelleneinträge auch, hier näher beschrieben werden.

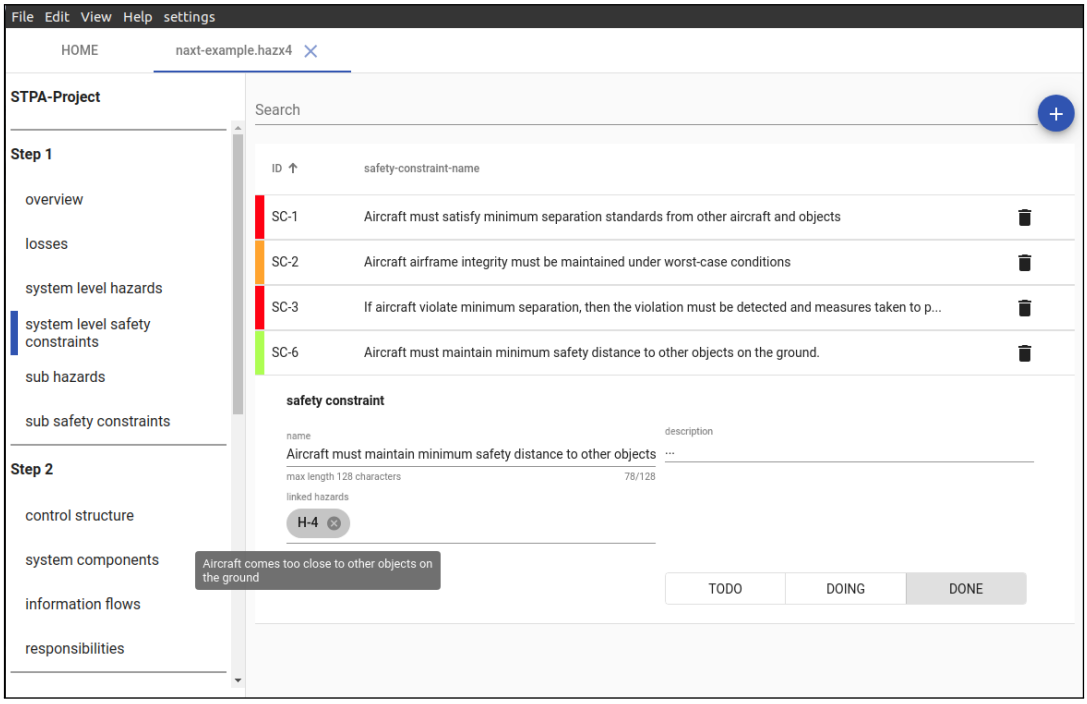

Abbildung 6.3: Anwendungsbeispiel: Definiere System-Level Safety Constraints

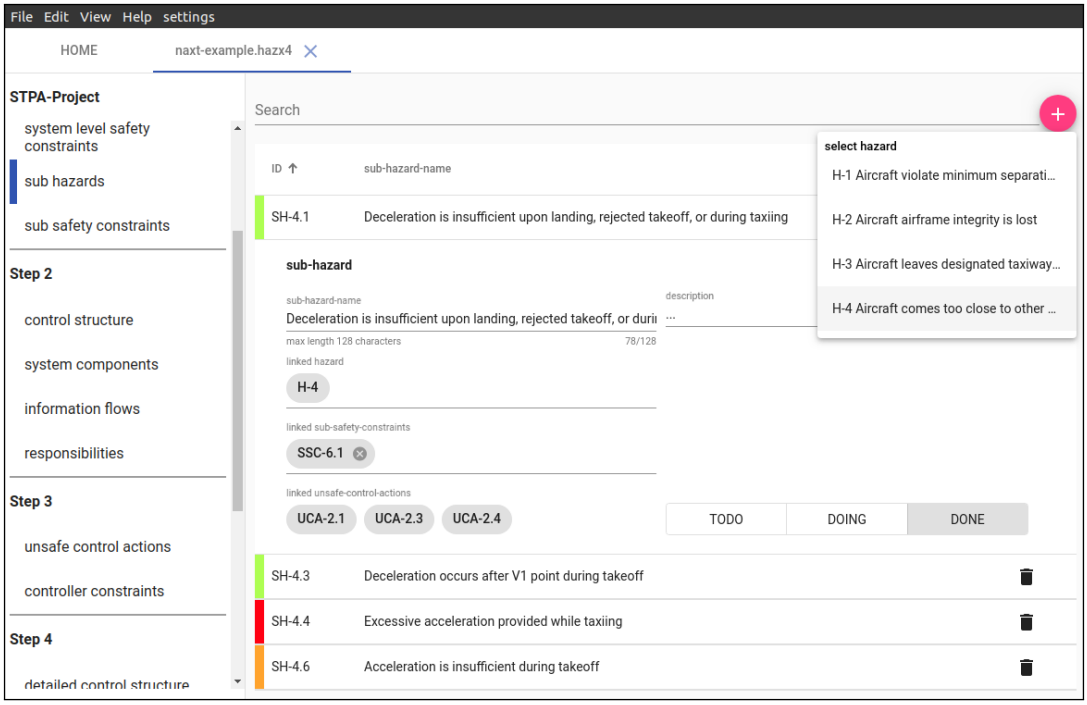

Abbildung 6.4: Anwendungsbeispiel: Verfeinerung in Sub Hazards

Verfeinerung in Sub Hazards *System-Level Hazard*, wie auch *System-Level Safety Constraints* im nächsten Schritt, können optional noch weiter verfeinert werden. Damit lassen sich auch in komplexeren Analysen *System-Level Hazard* in einzelne konkretere *Sub Hazards* aufteilen. In Abbildung 6.4 wird dazu *H-4* in kleinere *Sub Hazards* zerlegt. Hier zeigt sich auch, dass beim Anlegen von sogenannten abhängigen Tabelleneinträgen, angezeigt durch einen roten +-Button (zuvor blau), zuerst ein Elterneintrag ausgewählt werden muss. Dazu werden in diesem Schritt alle *System-Level Hazards* angezeigt aus denen ausgewählt werden kann. Diese Verlinkung ist dauerhaft und wird durch einen Chip ohne x-Button zum Entfernen verdeutlicht.

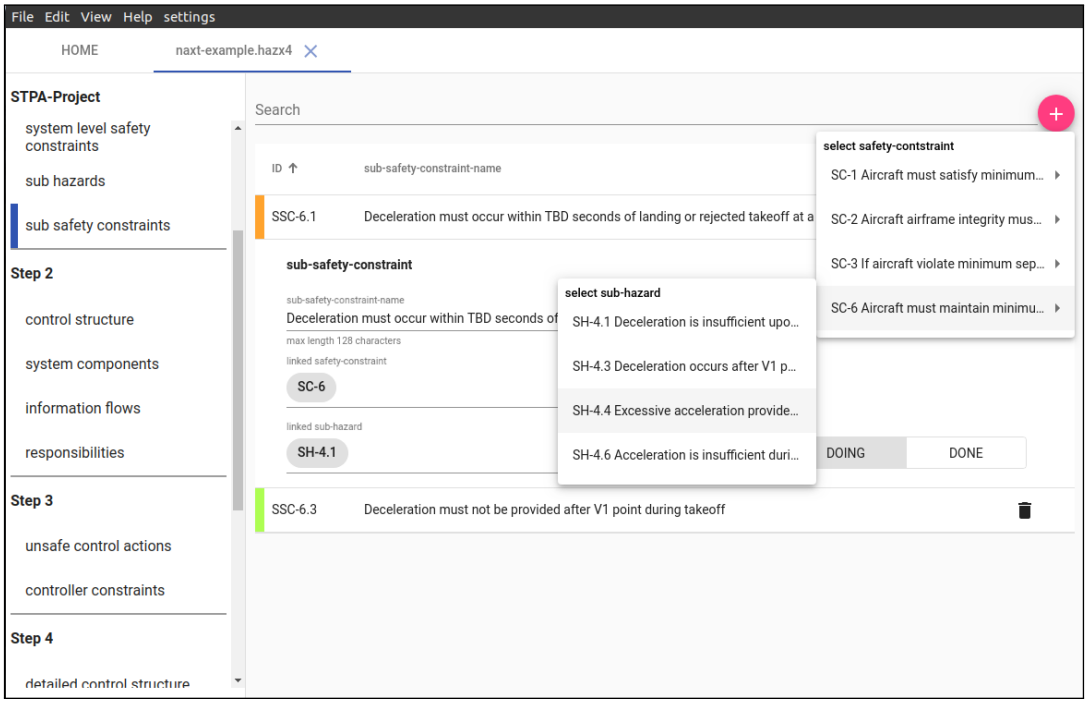

Abbildung 6.5: Anwendungsbeispiel: Verfeinerung in Sub Safefty Contraints

Verfeinerung in Sub Safefty Contraints Mit der Verfeinerung in *Sub Hazards* können ebenfalls *System-Level Safety Constraints* weiter verfeinert werden. Dazu werden zu jedem *Sub Hazard* ein *Sub Safety Constraint* angelegt, welches diesen verhindert. Hierzu werden sogar zwei notwendige Elterneinträge festgelegt und durch entsprechende Chips angezeigt (siehe Abbildung 6.5).

6.2 Schritt 2 - Modellierung der Control Structure

In diesem Schritt wird die *Control Structure* modelliert. Dem Analysten steht dazu der in XSTAMPP 4 entwickelten *Control Structure* Editor zur Verfügung (siehe Abbildung 6.6). Mit diesem lassen sich alle System- und Kontrollflusskomponenten einer *Control Structure* aus einer Auswahl anlegen (rechts im Bild).

System- und Kontrollflusskomponenten werden parallel in zwei Tabellen angelegt und können hier verwaltet und geändert werden.

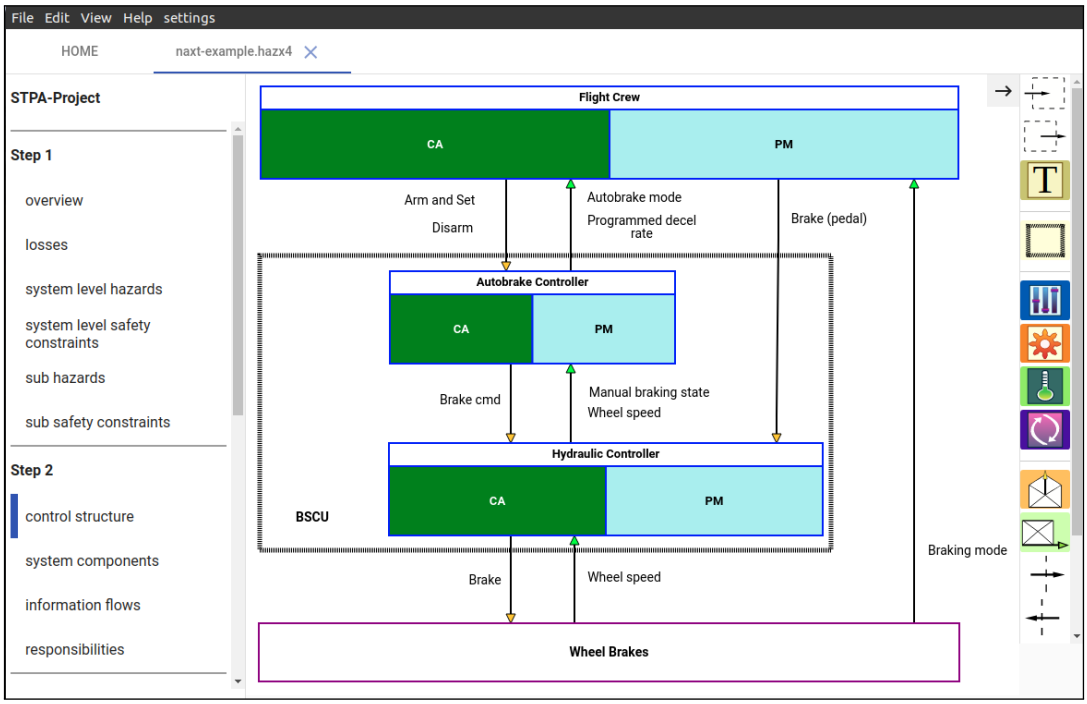

Abbildung 6.6: Anwendungsbeispiel: Control Structure

In einem iterativen Modellierungsprozess, und damit der fortlaufenden Identifikation neuer *Controller* innerhalb der *Control Structure*, werden diesen auch fortlaufende *Responsibilities* zugewiesen. *Responsibilities* werden dazu in einer separaten Tabelle (siehe Abbildung 6.7) angelegt und mit entsprechenden *System-Level Safety Constraints* bzw. *Sub Safety Constraints* verlinkt.

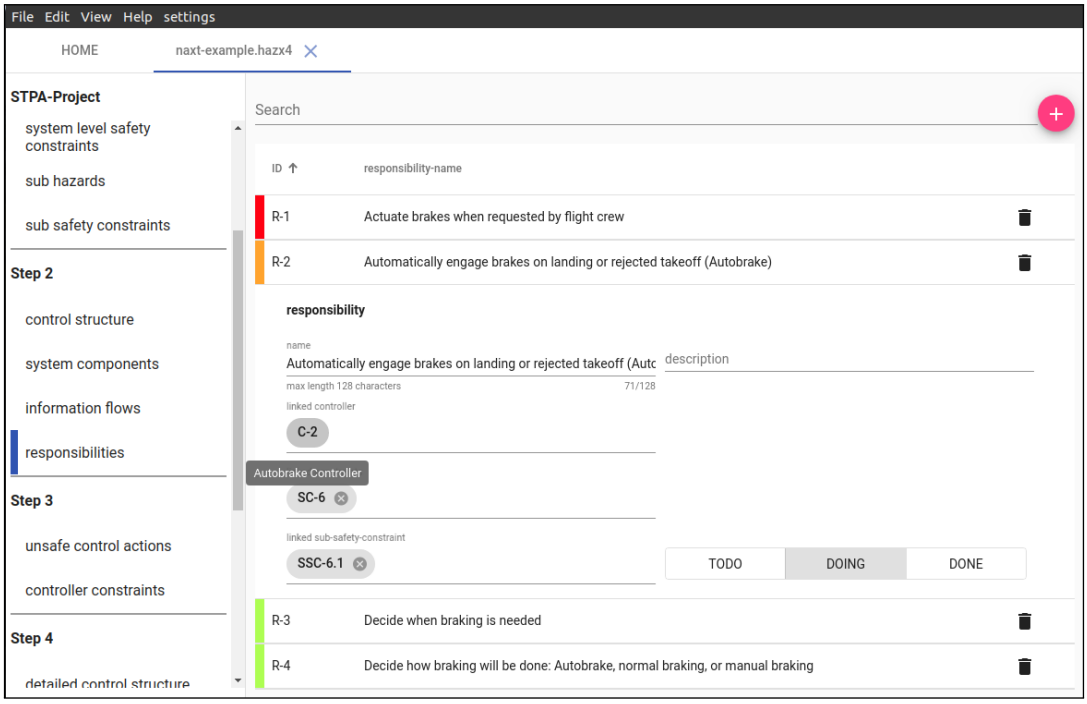

Abbildung 6.7: Anwendungsbeispiel: Responsibilities

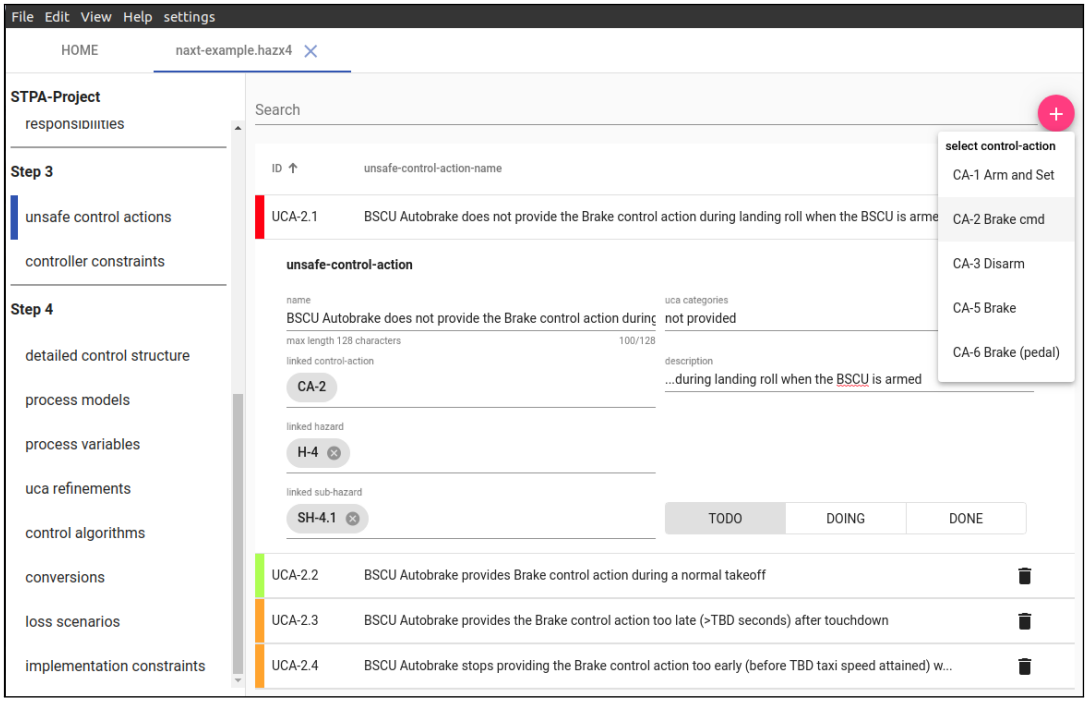

Abbildung 6.8: Anwendungsbeispiel: Unsafe Control Actions identifizieren

6.3 Schritt 3 - Unsafe Control Actions identifizieren

Anhand der Control Structure lassen sich sukzessive UCAs identifizieren. Dazu werden in Schritt 3 (siehe Abbildung 6.8) für unsichere *Control Actions* UCAs entsprechend der in Kapitel 2 vorgestellten Entstehungsmöglichkeiten angelegt. UCAs haben dazu neben Feldern für Verlinkungen und Beschreibungen auch ein Feld für die Kategorien *Not Provided*, *Provided*, *Stopped Too Soon or Applied Too Long* und *Too Early or Too Late*.

Sind UCAs identifiziert lassen sich auch *Controller Constraints* für diese in der ebenfalls in Schritt 3 enthaltenen Tabelle anlegen.

6.4 Schritt 4 - Loss Scenarios identifizieren

Zur Erstellung von *Loss Scenarios* beschäftigt sich der Analyst mit der Frage, wie es zu den identifizierten UCAs gekommen ist. Dazu erstellt der Analyst für eine konkrete UCA in der *Loss Scenarios* Tabellenansicht einen Eintrag zu diesem. Abbildung 6.9 zeigt die *General Information* den ersten Schritt in der Erstellung eines *Loss Scenarios*. Hier kann neben der untersuchten *Control Action* und der UCA, mit deren Kategorie, auch ein Name für das *Loss Scenario* vergeben werden. Im nächsten Schritt *Category* steht dem Analysten eine Auswahl aus den in Abschnitt 2.2.4 vorgestellten *Loss Scenario* Kategorien zur Verfügung (siehe Abbildung 6.10). Der Analyst kann hier die jeweilige Ober- und Unterkategorie auswählen und erhält dynamisch weitere Felder zur Eingabe weiterer, für diese Kategorie notwendige, Informationen.

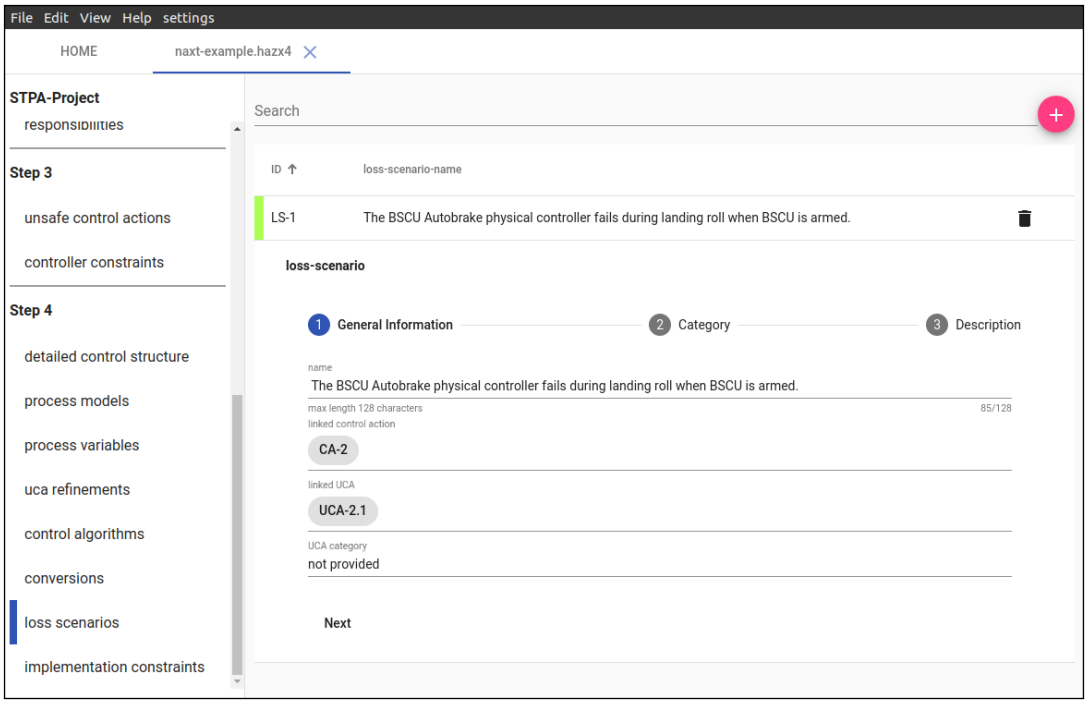

Abbildung 6.9: Anwendungsbeispiel: Loss Scenario (Schritt 1)

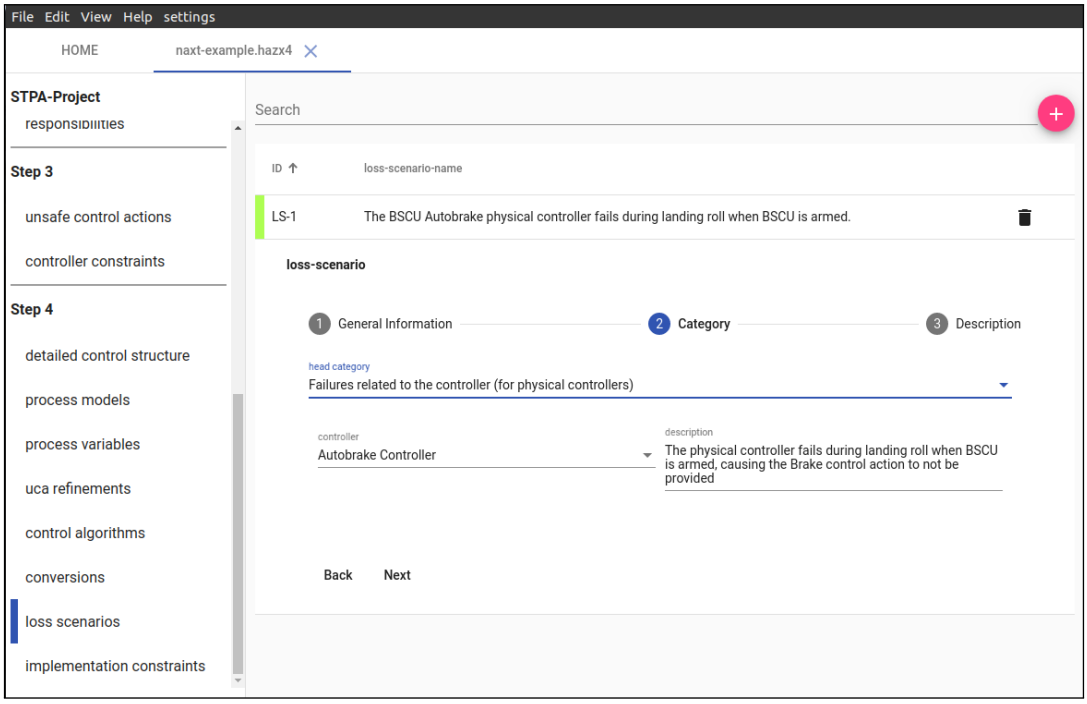

Abbildung 6.10: Anwendungsbeispiel: Loss Scenario (Schritt 2)

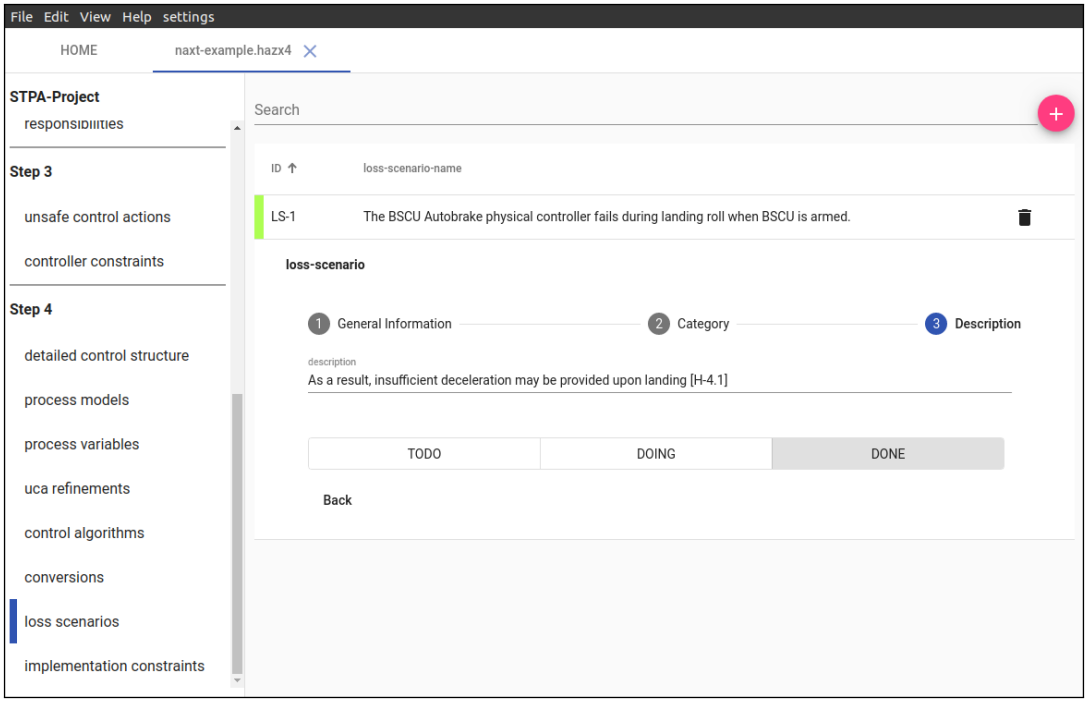

Abbildung 6.11: Anwendungsbeispiel: Loss Scenario (Schritt 3)

Im letzten Schritt *Description* (siehe Abbildung 6.11) kann dem *Loss Scenario* eine abschließende Beschreibung gegeben werden. Schlussendlich kann für jedes *Loss Scenario* noch eine empfohlene *Implementation Constraint* in einer separaten Tabelle erstellt werden. In diesem Schritt wurden einige Elemente zur Identifikation von *Loss Scenarios* übersprungen (siehe Abbildung 6.11). Dazu gehören die Erweiterung der *Control Structure* um *Actuators* und *Sensors*, welche zur Aufklärung der Kausalfaktoren in diesem Schritt notwendig sein können, wie auch Funktionen zu *Process Models* und Regelalgorithmen (siehe [Hec19]).

7 Diskussion

In diesem Kapitel soll sowohl das Ergebnis der entwickelten Einzelplatzanwendung als auch der Entwicklungsprozess diskutiert werden.

Der in XSTAMPP 4 verwendete STPA Analyseprozess wurde erfolgreich in der Einzelplatzanwendung umgesetzt. Darüber hinaus wurde in einigen Komponenten der STPA Analyse die Anwenderfreundlichkeit verbessert und Elemente eingefügt, um den Analysten in der Anwendung zu unterstützen. Darunter ist sowohl die Neuimplementierung der Tabelle, zur generischen Nutzung innerhalb der Einzelplatzanwendung, als auch die Verbesserung der Übersichtlichkeit bei der Erstellung von *Loss Scenarios*.

Der STPA Analyseprozess wurde während der Entwicklung fortlaufend um gewünschte Funktionen erweitert oder angepasst. Im Entwurf und der Umsetzung wurden Kernkomponenten von XSTAMPP 4 in generische STAMP Komponenten extrahiert sowie abstrakte Klassen und Interfaces angelegt und in Shared Modules gesammelt. Diese Module ermöglichen damit die einfache und unkomplizierte Erweiterung der Einzelplatzanwendung um die weitere Analysemethode CAST durch eine parallele Bachelorarbeit (siehe [Zim20]). Erweiterungen des STPA Moduls wurden so angelegt, dass diese trotz neuer Projektdaten voll abwärtskompatibel zur Webanwendung XSTAMPP 4 bleiben.

Zur Nutzung der neuen Tabelle wurde ebenfalls eine neue Filterkomponente erstellt. Diese Filterkomponente hat sich in der Nutzung der Einzelplatzanwendung als unzureichend herausgestellt. Der Analyst sollte in einigen Schritten besser durch die Analyse geleitet werden. Ein konkretes Beispiel ist die Nutzung der Tabelle zur Identifikation von UCAs. Die in Kapitel 2 vorgestellten Möglichkeiten zur Entstehung von UCAs sollten in dieser Ansicht, ähnlich wie in XSTAMPP 4, im Fokus stehen. Jedoch ohne große Teile der Ansicht einzunehmen.

Über den Analyseprozess hinaus wurde eine plattformübergreifende Einzelplatzanwendung erstellt, welche ohne Installation auf Microsoft Windows, macOS und Linux ausgeführt werden kann. Damit wird die Nutzung der Einzelplatzanwendung in einem Offlinekontext so einfach wie möglich gestaltet. Auch wurde sich an bekannten Praktiken ähnlicher Anwendungen orientiert, um dem Nutzer keine zusätzlichen Nutzungshürden aufzubürden.

Nicht umgesetzt wurde, aus zeitlichen Gründen, die Funktionen zum Export von PDF-Dateien (siehe F-2). Auch wurde vorerst keine Möglichkeit zum Download der Einzelplatzanwendung über die Internetumgebung XSTAMPP 4 eingerichtet (siehe F-7). Hierzu hat sich kein sinnvoller

Integrationspunkt ergeben, um Änderungen an der Einzelplatzanwendung zu publizieren ohne fortlaufende Änderungen an XSTAMPP 4.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In dieser Arbeit wurde, zur Nutzung der STAMP Analysemethode STPA, die Webanwendung XSTAMPP 4 als eine plattformübergreifende Einzelplatzanwendung entwickelt. Gleichzeitig wurde in zwei parallelen Arbeiten die Einzelplatzanwendung sowohl um den Analyseprozess CAST (siehe [Zim20]) als auch, zur Ergänzung von STPA, um ein Model Checking Verfahren (siehe [Hel20]) erweitert. Dazu wurden in der Umsetzung der Einzelplatzanwendung grundlegende STAMP Komponenten extrahiert und übergreifend verfügbar gemacht. Auch wurde eine gemeinsame Projektverwaltung für STPA und CAST implementiert.

Zusätzlich wurde der STPA Analyseprozess um weitere gewünschte Funktionen erweitert und bestehende Funktionen umstrukturiert. Auch wurden Verbesserungen der Anwenderfreundlichkeit durchgeführt und eine Nutzerdokumentation zur Nutzung der Einzelplatzanwendung integriert.

8.2 Ausblick

Wie auch vorherige XSTAMPP Arbeiten soll auch diese Anwendung fortlaufend verbessert und an Nutzerwünsche angepasst werden. Daher soll ein Ausblick an empfohlenen nächsten Entwicklungsschritten gegeben werden.

Der Analyst sollte beim STPA Analyseprozess, wie im Kapitel 7 bereits diskutiert, besser durch einige Schritte geleitet werden. Dazu wird empfohlen die Filterkomponente der Tabelle zu optimieren. In der Webanwendung von XSTAMPP 4 ist die Tabelle vorkonfigurierbar. In dieser Arbeit wurde ein anderer Ansatz gewählt, durch welchen sich keine ersichtliche Verbesserung ergab.

Wie in XSTAMPP 4 hätte im Rahmen dieser Arbeit ein Export eines Projekts als PDF-Datei implementiert werden sollen. Dies wurde in Ermangelung von Entwicklungszeit final nicht umgesetzt. Zu Dokumentations- und Präsentationszwecken sollte diese Funktionen daher in zukünftigen Arbeiten integriert werden.

Literaturverzeichnis

[AW14] A. Abdulkhaleq, S. Wagner. „Open Tool Support for System-Theoretic Process Analysis“. In: *2014 STAMP Conference at MIT, Boston, USA* (2014) (zitiert auf S. 25).

[AW15] A. Abdulkhaleq, S. Wagner. „XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering“. In: *2015 STAMP Conference at MIT, Boston, USA* (2015) (zitiert auf S. 25).

[AW16] A. Abdulkhaleq, S. Wagner. „XSTAMPP 2.0: new improvements to XSTAMPP Including CAST accident analysis and an extended approach to STPA“. In: *2016 STAMP Conference at MIT, Boston, USA* (2016) (zitiert auf S. 25).

[BMW02] G. Banse, B. Meier, H. Wolffgramm. *Technikbilder und Technikkonzepte im Wandel: eine technikphilosophische und allgemeintechnische Analyse*. Forschungszentrum Karlsruhe, 2002 (zitiert auf S. 13).

[Eri+15] C. A. Ericson et al. *Hazard analysis techniques for system safety*. John Wiley & Sons, 2015 (zitiert auf S. 14).

[Hec19] T. Heck. „Entwicklung eines Konzeptes zur Implementierung von Regelalgorithmen im Rahmen einer STPA und dessen Umsetzung in XSTAMPP 4.0“. 2019 (zitiert auf S. 37, 49).

[Hel20] F. Held. „Erweiterung der Einzelplatzanwendung des Systems XSTAMPP 4 um die Verfeinerung der Unsafe Control Actions für ein Model Checking“. 2020 (zitiert auf S. 13, 28, 53).

[Lev04] N. Leveson. „A new accident model for engineering safer systems“. In: *Safety science* 42.4 (2004), S. 237–270 (zitiert auf S. 13).

[Lev16] N. G. Leveson. *Engineering a Safer World : Systems Thinking Applied to Safety*. The MIT Press, 2016. ISBN: 9780262533690 (zitiert auf S. 3, 13, 14, 17).

[Lev19] N. G. Leveson. „CAST Handbook“. In: http://psas.scripts.mit.edu/home/get_file4.php?name=CAST_handbook.pdf (2019) (zitiert auf S. 13).

[LT19] N. G. Leveson, J. P. Thomas. „STPA Handbook“. In: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf (2019) (zitiert auf S. 3, 17–21, 41).

[Zim20] E. Zimmermann. „Adaption des Systems XSTAMPP 4 an die Analysemethode STAMP/CAST in der Einzelplatzanwendung“. 2020 (zitiert auf S. 13, 28, 34, 51, 53).

Alle URLs wurden zuletzt am 31.08.2020 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.

Horb am Neckar, 01.09.2020, 

Ort, Datum, Unterschrift