Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 229

A-CAST: Entwicklung eines Plugin-basierten Tools zur Unfallanalyse mit CAST (Causal Accident Analysis)

Martin Root

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer/in:	M. Sc. Asim Abdulkhaleq
Beginn am:	5. Mai 2015
Beendet am:	5. November 2015
CR-Nummer:	I.6.1, I.5.5

Kurzfassung

Unfälle sind auch in der heutigen Zeit keine Seltenheit. Bei sicherheitskritischen Systemen kann ein Unfall viele Menschenleben kosten. Deshalb sollte aus bereits geschehenen Unfällen gelernt werden. Durch eine Analyse solcher lassen sich mögliche zukünftige Unfälle vermeiden. Eine neue Methode für die Unfallanalyse ist der CAST-Ansatz, welcher auf dem STAMP-Modell basiert. Im Zuge dieser Arbeit wird ein Tool entwickelt, dass die Funktionen der CAST Analyse implementiert und so eine Unterstützung für die Analyse von Unfällen bieten soll.

Abkürzungsverzeichnis

XSTAMPP An eXtensible STAMP Platform As Tool Support for Safety Engineering

CAST Causal Accident Analysis

STAMP Systems-Theoretic Accident Model and Processes

STPA System-Theoretic Process Analysis

RCP Rich Client Platform

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

1 Einleitung

1.1 Motivation

Unfälle sind heutzutage keine Seltenheit. Trotz immer besser werdenden Sicherheitsmaßnahmen, lassen sich mögliche Unfälle nicht immer vorhersehen und verhindern. Die sicherheitskritischen Systeme werden immer komplexer und den Überblick über das Gesamtsystem zu erhalten, wird deutlich schwerer. Anwendungsbeispiele für solche Systeme sind Flugzeuge, Züge, Chemielabore oder auch Atomkraftwerke. Zusätzlich sind die potentiellen Gefahren und Risiken, die ein Unfall verursachen kann, viel größer geworden.

Ein Beispiel hierfür sind Flugzeuge. Sie können jetzt deutlich mehr Menschen transportieren als noch vor 10 Jahren, weshalb die mögliche Opferzahl bei Unfällen noch höher ist. Die Vermeidung eines Unfalls kann viele Menschenleben retten. Deshalb sollte das Thema Sicherheit nicht vernachlässigt beziehungsweise verharmlost werden.

Da es nicht möglich ist, Unfälle komplett zu verhindern, sollte es zum Ziel werden aus den Unfällen zu lernen. Das ist heutzutage aber meist nicht der Fall; oft wird nur ein Verantwortlicher gesucht und ist einer gefunden wird nicht nach den wirklichen Gründen geforscht, welche den Unfall verursacht haben könnten. Es werden nur die Symptome bekämpft und nicht die wirkliche Ursache des Unfalls. Auf diese Weise werden auch weiterhin viele Unfälle auftreten und es werden keine Fortschritte auf dem Gebiet Sicherheit gemacht.

Genau diese Problematik ist Thema dieser Bachelorarbeit. Ziel ist es, die Unfälle zu analysieren, Schwachstellen im System aufzudecken und aus den Fehlern zu lernen. Dadurch wird es möglich, das bestehende System sicherer zu machen und mögliche zukünftige Unfälle zu verhindern.

1.2 Problemstellung und Ziel

Um nach der Ursache für einen Unfall zu suchen, wird STAMP (Systems-Theoretic Accident Model and Processes) herangezogen, ein Unfallmodell das von Nancy Leveson entwickelt wurde [PSA15b]. STAMP beinhaltet die Gefahrenanalyse STPA (Systems Theoretic Process Analysis) und die Unfallanalyse CAST (Casual Accident Analysis). Bisher gibt es aber

noch keine Anwendung, die CAST unterstützt. Teil dieser Bachelorarbeit ist es daher, eine Applikation zu entwickeln, welche die Anwendung von CAST ermöglicht.

Es existiert bereits ein Tool namens XSTAMPP [Abd15], welches an der Universität Stuttgart entwickelt wurde und das STAMP Model implementiert. Dieses Tool unterstützte aber bisher nur STPA (Systems Theoretic Process Analysis), die Gefahrenanalyse. Ziel dieser Bachelorarbeit ist es, A-CAST, ein Plugin für STAMPP, zu entwickeln, dass die Funktionen von CAST in STAMP einbindet. Das Tool STAMPP basiert auf der Rich Client Plattform (RCP), weshalb A-CAST ebenfalls als RCP-Anwendung entwickelt wird.

Mithilfe von A-CAST soll eine Möglichkeit geboten werden Unfälle mit dem CAST-Ansatz möglichst einfach und schnell analysieren zu können. Da es bisher noch kein Tool gibt, welches die Funktionen von CAST implementiert, ist die Analyse oft sehr aufwendig. Da A-CAST Open-Source ist, also jedem zur Verfügung steht, sollte auf diese Weise jeder die Möglichkeit besitzen, den CAST-Ansatz für die Analyse eines Unfalls zu verwenden.

2 Grundlagen

In diesem Kapitel werden die nötigen Grundlagen für diese Bachelorarbeit vorgestellt.

2.1 Begrifflichkeiten

Im Folgenden werden einige Begrifflichkeiten geklärt. Im Englischen unterscheidet man zwischen den Begriffen “Safety“ und “Security“. Safety bezieht sich darauf, dass das System jemandem gefährlich werden kann (zum Beispiel Unfälle), während Security bedeutet, dass jemand dem System gefährlich werden kann. Im Deutschen gibt es diese begriffliche Unterscheidung leider nicht. Wenn in den folgenden Kapiteln vom Thema Sicherheit gesprochen wird, ist hiermit der englische Begriff “Safety“ gemeint.

Mit sicherheitskritischen Systemen sind Systeme gemeint, bei denen Sicherheit eine sehr große Rolle spielt, da mögliche Unfälle sehr viele Menschenleben kosten oder große Schäden verursachen können. Ein solches sicherheitskritischem System wäre zum Beispiel ein Atomkraftwerk. Wenn in dieser Arbeit von einem „System“ gesprochen wird, bezieht sich das auf ein solches komplexes sicherheitskritisches System.

Thema dieser Arbeit ist die Unfallanalyse, deshalb sollte die genaue Definition eines Unfalls geklärt werden. Ein Unfall ist: „ein Ereignis, das unbeabsichtigt geschieht und Schaden anrichtet und bei dem auch Menschen verletzt oder getötet werden können“ [Wö15]. Wichtig hierbei ist, dass ein Unfall ein unbeabsichtigtes Ereignis ist, sollte jemand ein System manipulieren, handelt es sich hierbei nicht um einen Unfall. Deshalb wird dieser Punkt in der Analyse für die Sicherheit auch nicht betrachtet.

2.2 Unfallmodelle

Bisherige Unfallmodelle basieren meist auf einem Kettenprinzip; das heißt ein Vorfall führt zum nächsten, der wieder zum nächsten Vorfall führt. Dieses Vorgehen wird weitergeführt bis der Unfall erreicht wurde. Beispiele für solche Modelle wären das Domino-Unfallmodell [LRH09]. Ein weiteres Modell ist die Fault Tree Analyse [Cle02], die auf einem ähnlichen

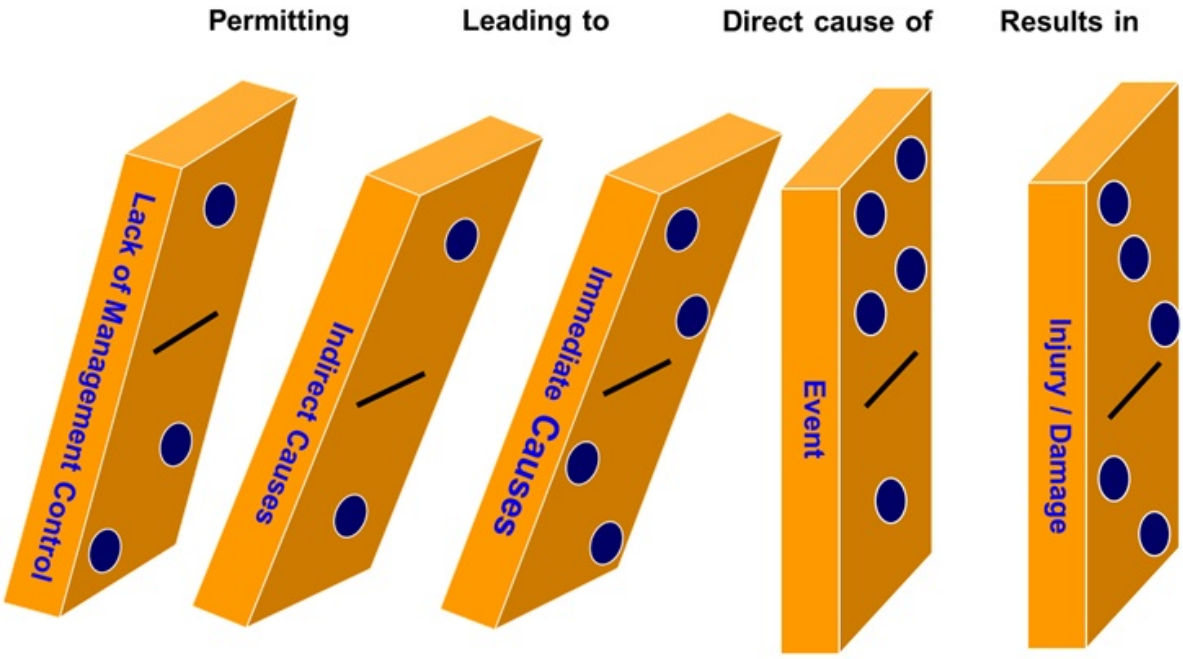


Abbildung 2.1: Domino Unfallmodell von Heinrich [Jed11]

Prinzip aufbaut. Hier wird für die Modellierung ein Baum benutzt. In Abbildung 2.1 ist ein Beispiel für das Domino Model zu sehen.

In den letzten Jahren sind sicherheitskritische Systeme viel größer und komplexer geworden, sie habe sich stark entwickelt. Ebenso ist die Technik weit vorangeschritten. Diese schnelle Entwicklung stellt durchaus ein Problem dar. Die Modelle, wie die oben genannten, sind nicht für diese fortgeschritten System entwickelt worden und wurden in der Zwischenzeit auch nicht erneuert. Früher waren die Systeme nicht so komplex wie heute, weshalb die gängigen Unfallmodelle überholt sind und sich eher nur noch auf simple kleine Systeme erfolgreich anwenden lassen. Weiterhin sind die gängigen Unfallmodelle darauf ausgelegt, dass ein Unfall durch das Versagen oder Ausfallen einer Komponente auftritt. Dies ist bei den heutigen komplexen Systemen oft nicht die Ursache für den Unfall, weshalb sich diese Unfallmodelle auf große und komplexe Anwendungssysteme meist nicht mehr anwenden lassen [Lev11].

Ein weiterer Punkt ist, dass die meisten Modelle auf einer Folge von Ereignissen basieren. Problem hierbei ist, dass die Ereignisse, welche in Betracht gezogen werden, subjektiv gewählt werden. Das heißt, wenn ein System komplexer und größer ist, kommt es oft zu unterschiedlichen Ergebnissen in der Analyse, falls diese von verschiedenen Personen durchgeführt wird. Weiterhin ist es nicht einfach den Überblick über die geschehenen Ereignisse zu behalten. Es besteht auch immer die Möglichkeit Ereignisse vorne oder hinten an die Kette anzuhängen, was wieder zu unterschiedlichen Resultaten führen kann. Ein allgemeines Problem dieser Modelle liegt darin, dass oft nur die naheliegenden Ereignisse in

der Kette in Betracht gezogen werden und als Ursache für den Unfall festgehalten werden. So wird oft einzelnen Personen die Schuld für den Unfall gegeben, obwohl der wirkliche Grund vielleicht weit hinten in der Ereigniskette lag [Bau06].

2.3 STAMP

STAMP steht für *Systems-Theoretic Accident Model and Processes* und wurde von Nancy Leveson entwickelt. Es ist ein Unfallmodell, welches versucht die Unfälle auf andere Weise abzubilden als es bisherige Modelle (siehe Kapitel Unfallmodelle) tun. STAMP basiert auf der System-Theorie, welche im Zeitraum von 1930-1940 entstanden ist. Sie war eine erste Antwort auf die Analyse von den immer komplexer werdenden Systemen [Ulr88]. Grundsätzlich betrachtet sie ein System als Ganzes und versucht nicht es in einzelne Teile aufzuteilen. Es wird davon ausgegangen, dass Eigenschaften des System nur in seiner Gesamtheit bewertet werden können. Um dies zu erreichen, werden zusätzlich die sozialen und technischen Aspekte in Betracht gezogen. Die Eigenschaften eines Systems werden auch aus den Beziehungen der einzelnen Teile abgeleitet. Allgemein gilt, dass STAMP Unfälle als ein Kontrollproblem beschreibt und nicht auf Zuverlässigkeit der einzelnen Komponenten im System zurückführt [Lev11].

STAMP lässt sich in zwei Komponenten aufteilen: STPA und CAST. STPA steht für die Gefahrenanalyse, sie wird angewendet um mögliche Gefahren in einem System aufzudecken um so die Sicherheit eines Systems gewährleisten zu können. CAST hingegen steht für die Unfallanalyse. Sie wird angewendet, wenn ein Unfall bereits aufgetreten ist und dieser wird mithilfe von CAST analysiert um die wirklichen Gründe und Umstände, welche zum Unfall geführt haben, herauszufinden und so auch zukünftige Unfälle verhindern zu können. In späteren Abschnitten werden diese beiden Teile von STAMP genauer beschrieben.

STAMP besteht neben der System-Theorie aus 3 Hauptteilen:

- Sicherheitsauflagen
- Hierarchische Sicherheits-Kontrollstruktur
- Prozess Modelle

Diese werden in den folgenden Unterabschnitten näher erläutert.

2.3.1 Sicherheitsauflagen

Sicherheitsauflagen spielen eine sehr wichtige Rolle in STAMP. Unfälle oder Ausfälle im System sind dadurch definiert, dass Sicherheitsauflagen verletzt wurden. Sie legen fest was gelten muss, damit Sicherheit gewährleistet werden kann. Die Schwierigkeit liegt darin, diese

Sicherheitsauflagen für Komponenten im System festzulegen. Je komplexer und größer ein System, desto aufwändiger wird es diese zu finden. Eine Sicherheitsauflagen könnte beispielsweise so aussehen: “Die Temperatur des Motors sollte nicht höher als 120 Grad Celsius sein“. Um den Grad an Sicherheit eines Systems zu erhöhen sollten die Sicherheitsauflagen vor dem Entwurf und Design der entsprechenden Komponenten im System festgelegt werden.

2.3.2 Hierarchische Sicherheits-Kontrollstruktur

Um den Überblick über das System und die Prozesse welche darin ablaufen zu bekommen, wird eine hierarchische Sicherheits-Kontrollstruktur angelegt. Diese setzt sich aus mehreren Kontrollschleifen im System zusammen. In Abbildung 2 ist ein Beispiel einer solchen Kontrollschleife zu sehen.

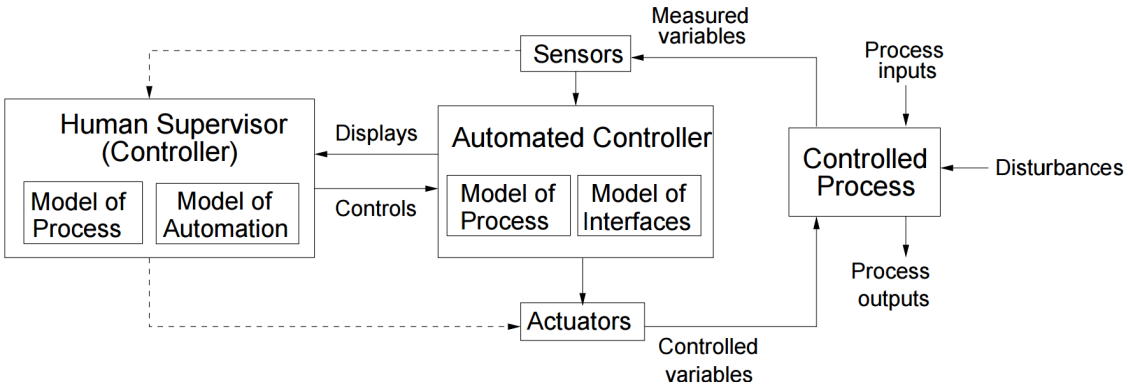


Abbildung 2.2: Aufbau einer Kontrollschleife [LDDM03]

Solche Kontrollschleifen helfen dabei zu verstehen, weshalb oder wie ein Unfall auftreten kann. Wie in Abbildung 2 zu sehen ist, wird dafür ein Controller angelegt der einen kontrollierten Prozess überwacht. Hierfür kann es Aktuatoren geben die beispielsweise Aktionen im Prozess einleiten können. Sensoren messen die Daten, welche währenddessen anfallen. Ein Unfall tritt nur auf, wenn ein Kontrollproblem vorherrscht. Dies könnte auftreten, wenn beispielsweise Sicherheitsauflagen fehlen, oder falsches Feedback im Prozess zurückgegeben wird. Die hierarchische Kontrollstruktur kann bei einem großem System ziemlich komplex werden, deshalb wird sie oft in mehrere Teile aufgeteilt um so den Überblick behalten zu können. In Abbildung 4 ist ein Beispiel zu sehen, wie eine solche Kontrollstruktur aussehen kann.

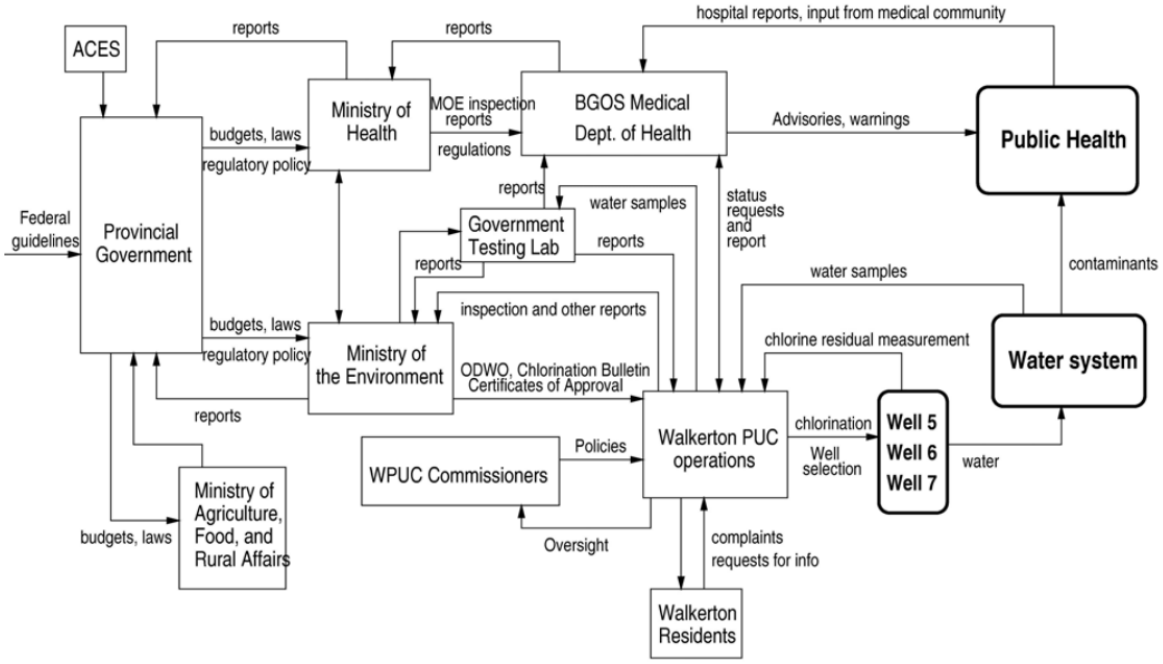


Abbildung 2.3: Kontrollstruktur eines Wasserwerks in Ontario. Linien die von links in eine Box verlaufen stellen Kontrollinien dar. Die anderen Linien repräsentieren Feedback. Die eckigen Boxen stehen für Controller. [LDDM03]

2.3.3 Prozess Modelle

Das Prozess Modell spielt eine wichtige Rolle in STAMP. Hier wird vorgegeben was gelten muss, sodass ein Prozess kontrolliert ablaufen kann. Jeder Controller sollte ein Modell des zu kontrollierenden Prozess enthalten. Unfälle können nur auftreten, falls dieses Modell das entsprechende System fehlerhaft, beziehungsweise nicht vollständig abbildet oder der Controller unsichere Befehle weitergibt. Sollte ein fehlerhaftes Prozess Modell existieren kann das zu Problemen führen, welche wiederum für einen Unfall verantwortlich sein könnten. Im folgenden werden mögliche Beispiele für solche Probleme beschrieben [LDDM03]:

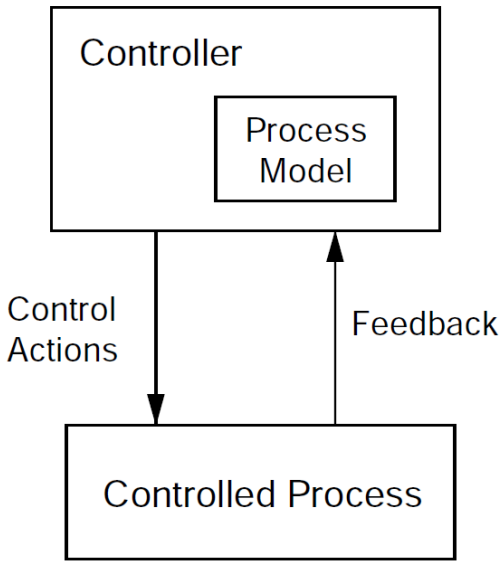


Abbildung 2.4: Prozess Modell. [LDDM03]

1. Falsche oder unsichere Befehle wurden vom Controller erteilt.
2. Benötigte Aktionen zur Kontrolle (für die Sicherheit) sind nicht vorhanden
3. Potentiell korrekte Befehle werden zum falschen Zeitpunkt (zu früh oder zu spät) erteilt.
4. Die Kontrolle wird zu früh gestoppt oder dauert zu lange an.

2.4 CAST

CAST steht für *Causal Accident Analysis*. Es basiert auf dem STAMP Modell und ist speziell für die Analyse von Unfällen entworfen worden. Es verwendet die Kontrollstrukturen und Sicherheitsauflagen, welche bereits in den vorherigen Kapiteln genauer vorgestellt wurden. Mithilfe von CAST soll das Ziel erreicht werden, dass man einen bereits geschehenen Unfall analysiert und versucht die wirklichen Gründe für das Auftreten dieses zu finden. Dabei soll das komplette System mit einbezogen werden. Die Beziehungen und Abhängigkeiten werden mithilfe der Kontrollstruktur modelliert, um so einen Überblick über das Gesamtsystem zu erhalten. So soll ermöglicht werden, nicht nur Symptome zu bekämpfen, sondern die wirklichen Ursachen eines Unfalls zu finden. Gelingt dies, können noch vorhandene Schwächen und Lücken in der Sicherheit des Systems entdeckt und behoben werden, was wiederum mögliche zukünftige Unfälle verhindern kann. Grundsätzlich wird bei CAST eine Reihenfolge an Schritten angewendet, wobei diese nicht explizit eingehalten werden muss. Es besteht durchaus die Möglichkeit einen Schritt schon durchzuführen bevor die vorigen fertiggestellt sind [Lev11].

2.4.1 Rückschaufehler

Ein weiterer wichtiger Punkt in CAST ist der Rückschaufehler. Damit sind Fehler gemeint, welche nach dem Unfall betrachtet und als leicht zu sehen erachtet werden. Also Fehler die eigentlich offensichtlich sind, aber trotzdem geschehen sind. Oft werden Personen, die solche Fehler begangen haben, nur beschuldigt und es wird nicht geschaut, warum sie den Fehler gegangen haben. Es sollte davon ausgegangen werden, dass die Person den Fehler nicht mit Absicht begangen hat. Daher sollte analysiert werden, warum die betreffende Person die Schritte ausgeführt und wie es dazu gekommen ist. So lassen sich zukünftige Fehler dieser Art verhindern [PSA15a].

2.4.2 Schritte von CAST

Teil dieser Bachelorarbeit ist es, eine Anwendung zu entwickeln, welche die Analyse mithilfe von CAST ermöglicht. Im Folgenden werden daher die Schritte von CAST in der Reihenfolge aufgeführt, in welcher sie auch im Tool A-CAST implementiert wurden.

1. Grundlagen der Analyse
2. Konstruieren der Kontrollstruktur und deren Abhängigkeiten
3. Unfall Analyse

In den folgenden Abschnitten werden die einzelnen Schritte genauer erläutert.

2.4.3 Grundlagen der Analyse

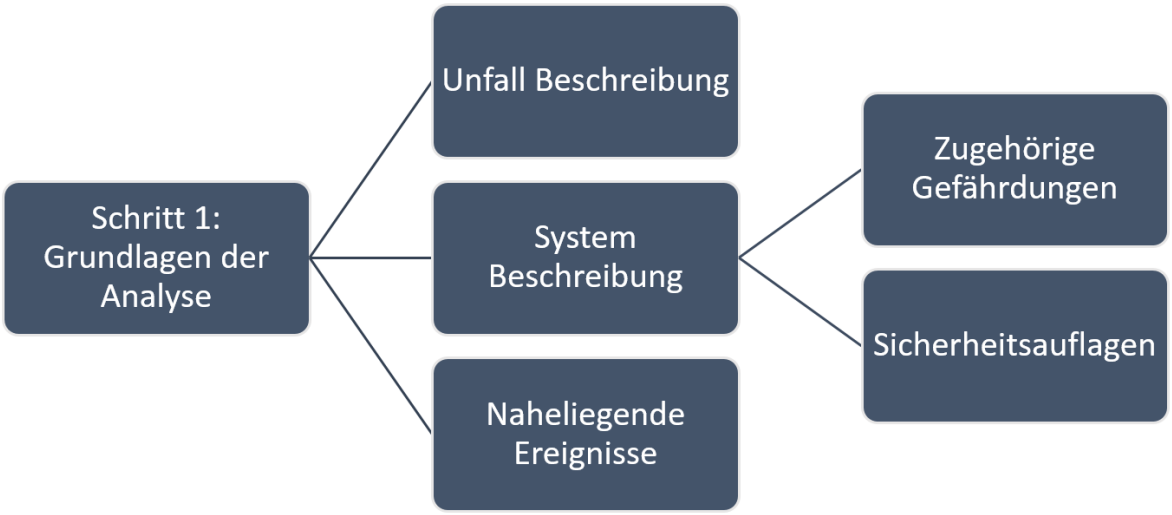


Abbildung 2.5: Darstellung des 1. Schrittes und seinen Unterpunkten.

Wie in Abbildung 2.5 zu sehen ist, wird zuerst der Unfall genau beschrieben, dazu gehören beispielsweise eine Auflistung der Geschehnisse, Zeitpunkt und der Ort des Unfalls. Diese Maßnahme soll dazu dienen, dass stets ein Überblick über den Unfall vorhanden ist und sollte ein Detail des Unfalls vergessen werden, kann hier stets das fehlende Wissen aufgefrischt werden.

Der nächste Schritt ist die System-Beschreibung. Hier wird auf STAMP zurückgegriffen und die Gefährdungen, sowie die Sicherheitsauflagen festgehalten. Da bei der Analyse mit CAST der Unfall bereits geschehen ist, müssen die Sicherheitsauflagen verletzt worden sein oder erst gar nicht vorhanden gewesen sein und somit neu angelegt werden.

Zum Schluss werden naheliegende Ereignisse festgehalten. Es wird eine Art Protokoll angelegt, in welchem die letzten Aktivitäten, im Zeitraum kurz vor dem Unfall, protokolliert werden. Dazu wird ein Zeitstempel und das zugehörige Ereignis notiert. Bei einem Flugzeugabsturz könnte ein Ereignis dieses Protokolls beispielsweise folgendermaßen aussehen:

Beschreibung: Pilot hat die Instrumente des Cockpits überprüft.

Zeitpunkt: 16:15 Uhr 1.1.2015.

2.4.4 Konstruieren der Kontrollstruktur

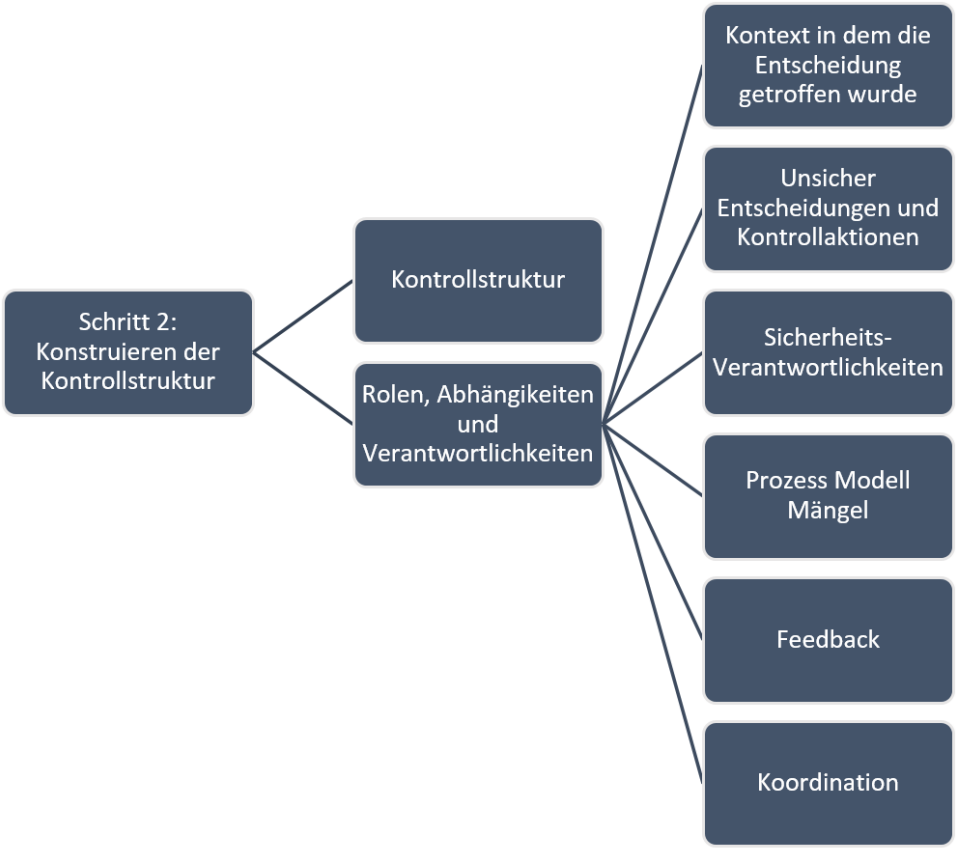


Abbildung 2.6: Darstellung des 2. Schrittes und seinen Unterpunkten.

Im 2. Schritt wird die Kontrollstruktur angelegt. Hierbei muss das komplette System modelliert werden, das heißt jede Rolle und ihre Beziehung sollte in der Kontrollstruktur angelegt werden. Wie die Kontrollstruktur erstellt wird, ist bereits in Kapitel 2.3.2 beschrieben worden. Ein Beispiel hierfür ist in Abbildung 2.3 zu sehen. Nach dem Anlegen der Kontrollstruktur, werden nun wie in Abbildung 2.6 zu sehen ist, Verantwortlichkeiten der einzelnen Rollen im System definiert. Um noch mal auf das Beispiel eines Flugzeugabsturzes zurückzukommen, wären hier mögliche Rollen beispielsweise der Pilot, die Flugzeugcrew und auch der Flughafen selbst. Für solche Rollen im System muss nun festgehalten werden, wie es zu einer Entscheidung oder der ausgeführten Aktion gekommen ist. Hierbei werden folgende Punkte betrachtet:

- Sicherheitsbezogene Verantwortlichkeiten
- Unsichere Entscheidungen und Kontrollaktionen
- Mängel im Prozess Modell

- Kontext, in welchem die Entscheidungen getroffen wurden
- Feedback
- Koordination

Um das noch an einem Beispiel zu erläutern, ist in Abbildung 2.7 aufgeführt, wie die eben genannten Punkte festgehalten werden können. Hierfür wird wieder ein Flugzeugabsturz herangezogen und die Verantwortlichkeiten der Rolle „Flugzeugcrew“ gezeigt. Es wird in dieser Abbildung jeweils nur ein Beispiel für jede der oben genannten Punkte präsentiert, in der damaligen Analyse wurden natürlich noch weitere gefunden.

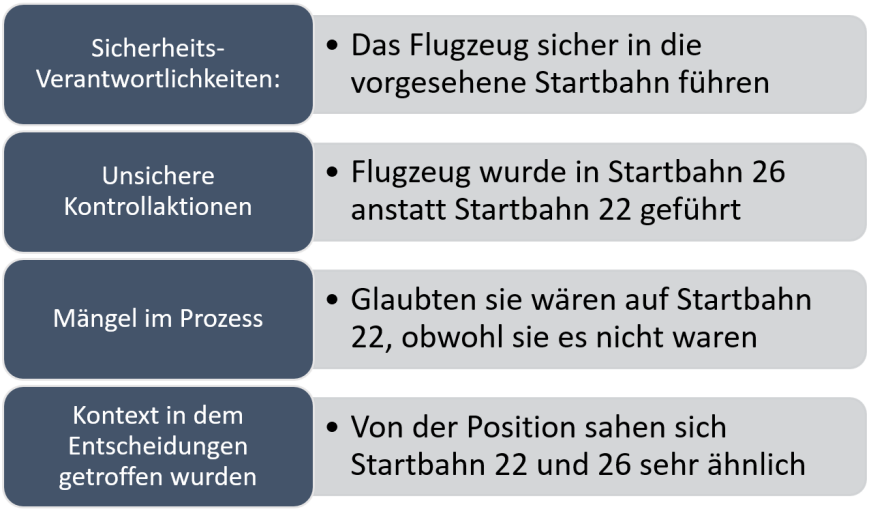


Abbildung 2.7: Eine Rolle im System und ihre Verantwortlichkeiten [PSA15a].

Mit *sicherheitsbezogenen Verantwortlichkeiten* werden die Sicherheitsauflagen für eine Rolle im System angelegt. Im Beispiel war eine solche Sicherheitsauflage, die Wahl der richtigen Startbahn für das Flugzeug. Die *unsicheren Entscheidungen und Kontrollaktionen* beschreiben die Aktionen welche mitverantwortlich für den Unfall waren. Im Beispiel war das die Wahl der falschen Startbahn für das Flugzeug. Mit *Mängel im Prozess Model* sind Fehler im Modell selbst gemeint, das heißt die Rolle hat ein falsches Bild vom System selbst oder auch einer Komponente im System, mit welcher sie interagiert. Im Beispiel war ein Mangel im Prozess Modell, dass die Flugzeugcrew dachte, sie wäre auf der richtigen Startbahn, obwohl sie es nicht waren. Der *Kontext in dem die Entscheidungen getroffen wurden*, beschreibt den Zustand in der die Entscheidung getroffen wurde. Im Beispiel war der Kontext, dass vom Blickwinkel der Flugzeugcrew, die verwechselte Startbahn der richtigen Startbahn sehr ähnlich sah.

2.4.5 Unfall Analyse

Bei der Unfall Analyse werden die Funde und Ergebnisse, also die Gründe die zum Unfall geführt haben, festgehalten. Dazu gehören die verletzten Sicherheitsauflagen sowie Empfehlungen, welche Mängel in der Sicherheit des Systems beheben sollen. Wenn möglich werden für jede Rolle im System die Funde und Empfehlungen einzeln angelegt, meistens sind mehrere Rollen im System am Verschulden des Unfalls verantwortlich.

2.5 STPA

STPA steht für *System-Theoretic Process Analysis*. Es basiert auf STAMP und ist für die Gefährdungsanalyse im System zuständig. STPA sollte so früh wie möglich in der Entwicklung des Systems angewendet werden. So können Schwachstellen bereits beim Entwurf festgestellt und spätere zusätzliche Kosten vermieden werden. Es ist aber auch möglich es bei bereits bestehenden Systemen zu verwenden um Schwachstellen, Gefährdungen oder auch Softwarefehler und menschliche Fehler im System festzustellen. Oft werden zusätzlich Mängel in den Anforderungen an das System entdeckt. STPA ist vor allem für große komplexe Systeme gedacht, eben dort wo andere Modelle (siehe Kapitel 2.2) an ihre Grenzen stoßen. [AW14]

Bei STPA wird eine bestimmte Reihenfolgen an Schritten durchgeführt. Im Folgenden werden die einzelnen Schritte aufgeführt, es wird aber nur eine kurze Erklärung dazu geben, da STPA nicht Teil des Themas dieser Bachelorarbeit war. Die Schritte für die Analyse sollten in folgender Reihenfolge durchgeführt werden:

1. Mögliche Unfälle und Gefährdungen identifizieren
2. Zeichnen der Kontrollstruktur
3. Identifizieren von unsicheren Kontroll-Aktionen
4. Identifizieren von kausalen Faktoren und das Kreieren von möglichen Szenarios

In den ersten zwei Schritten wird das System auf mögliche Unfälle und Gefährdungen geprüft. Hierbei wird auf das STAMP Modell zurückgegriffen. Beim dritten Schritt werden unsichere Kontroll-Aktionen gesucht, also Aktionen die zu einer Gefährdung führen könnten. Im letzten Schritt werden kausale Faktoren gesucht, welche zum Fehlverhalten führten und ebenso Szenarien für das System erstellt, welche die bestehende Sicherheitsauflagen testen um so weitere Gefährdungen entdecken zu können. [AW14]

2.6 XSTAMPP

XSTAMPP steht für *"An eXtensible STAMP Platform As Tool Support for Safety Engineering"* und wurde an der Universität Stuttgart entwickelt. Es ist eine Plattform welche das STAMP-Modell und dessen Funktionen implementiert. Zuerst gab es das Tool A-STPA, welches durch ein Entwicklungsprojekt an der Universität Stuttgart entstanden ist. A-STPA wurde entwickelt um die Verwendung der STPA-Methodik zu ermöglichen und wurde im Februar 2014 als Open-Source-Software veröffentlicht. STPA war zu der Zeit schon oft erfolgreich im Einsatz, jedoch gab es kein Tool, das die Verwendung von STPA ermöglichte. Aus diesem Grund ist A-STPA entwickelt und dessen Funktionen bereits erfolgreich in XSTAMPP integriert worden. [AW]

XSTAMPP wurde mit dem Hintergrund entwickelt eine Plattform zu bilden, welche leicht um neue Funktionalitäten auf Basis des STAMP-Modells erweitert werden kann. Ein Beispiel hierfür ist die Erweiterung der XSTAMPP-Plattform um A-CAST, ein Tool das die Funktionalität von CAST implementiert und das Thema dieser Bachelorarbeit darstellt. In Abbildung 2.8 ist ein Überblick über die Architektur von XSTAMPP zu sehen.

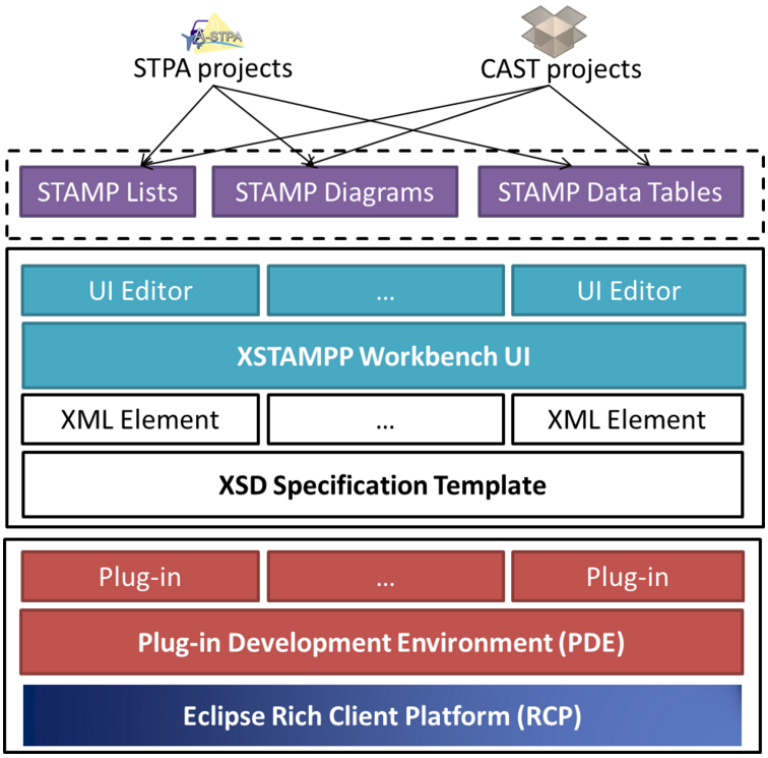


Abbildung 2.8: Die Architektur von XSTAMPP im Überblick. [AW]

XSTAMPP baut auf der *Eclipse Rich Client Platform (RCP)* auf. Dadurch lässt es sich leicht um Plug-ins erweitern, wie beispielsweise um A-CAST. XSTAMPP selbst implementiert die

Funktionalitäten aus STAMP, worauf wiederum Plug-ins zugreifen können (siehe Abbildung 2.8). Des Weiteren bietet XSTAMPP eine Benutzeroberfläche an, auf welcher Projekte gespeichert, geladen, gelöscht und bearbeitet werden können. Um dies zu ermöglichen, muss das entsprechende Plug-in seine Daten im XML-Format speichern und ein entsprechendes XSD-Schema für das Einlesen der Daten bereitstellen. Im Folgenden wird die XSTAMPP Benutzeroberfläche vorgestellt. [AW]

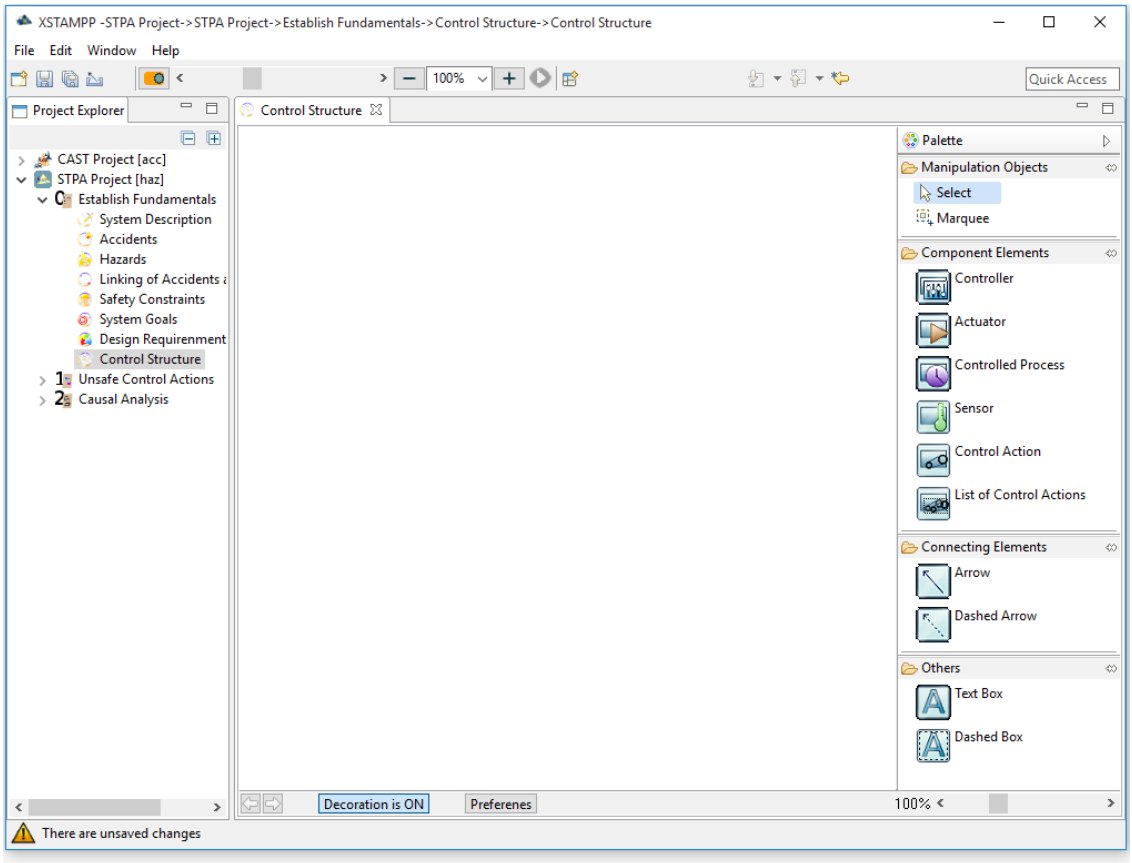


Abbildung 2.9: XSTAMPP Workbench UI mit einem CAST und STPA Project. [AW]

Die Benutzeroberfläche in XSTAMPP hat das Design einer normalen Eclipse-RCP Anwendung. Das Layout der Views kann beliebig verändert werden so wie es in einer RCP-Anwendung üblich ist. In Abbildung 2.9 ist die standardmäßige Benutzeroberfläche von XSTAMPP zu sehen. Auf der linken Seite ist ein Projekt-Explorer, hier werden die gespeicherten Projekte angezeigt und verwaltet. Auf der rechten Seite werden die Editoren der entsprechenden Projekte angezeigt, beispielsweise die Kontrollstruktur.

3 A-CAST Tool

In diesem Kapitel wird das Tool A-CAST vorgestellt. Dabei wird auf die Architektur, das Design und die Implementierung eingegangen.

3.1 Architektur

A-CAST wurde als Plug-in für XSTAMPP entwickelt und basiert daher ebenfalls auf der Rich-Client-Platform. Für die Entwicklung wurden Eclipse und die Programmiersprache Java verwendet.

3.1.1 Rich Client Plattform

Die Rich Client Plattform (kurz RCP) Architektur bietet leichte Erweiterbarkeit und Flexibilität. Bis auf den Kernel selbst sind fast alle anderen Komponenten in RCP Erweiterungen und können daher nach Belieben entfernt oder hinzugefügt werden. Das heißt die entsprechende Applikation ist komplett modular aufgebaut und lässt sich auf diese Weise einfach an die gewünschte Funktionalität anpassen. So lassen sich beliebige Applikationen entwickeln. Dabei kann ein Plug-in ein Web Browser, Dialog, Explorer und Weiteres sein. Es gibt bereits vorgefertigte Plug-ins wie beispielsweise einen Export-Wizard oder auch ein Hilfe-Fenster. RCP benutzt SWT (Standard Widget Toolkit) für die graphischen Oberflächen, was für gute Performance auch auf schwächeren Systemen führt. Ein weiterer Vorteil von SWT ist, dass beispielsweise Dialoge und Fenster das gleiche Aussehen haben wie die des nativen Betriebssystems. [Sil09]

3.2 Design und Implementierung

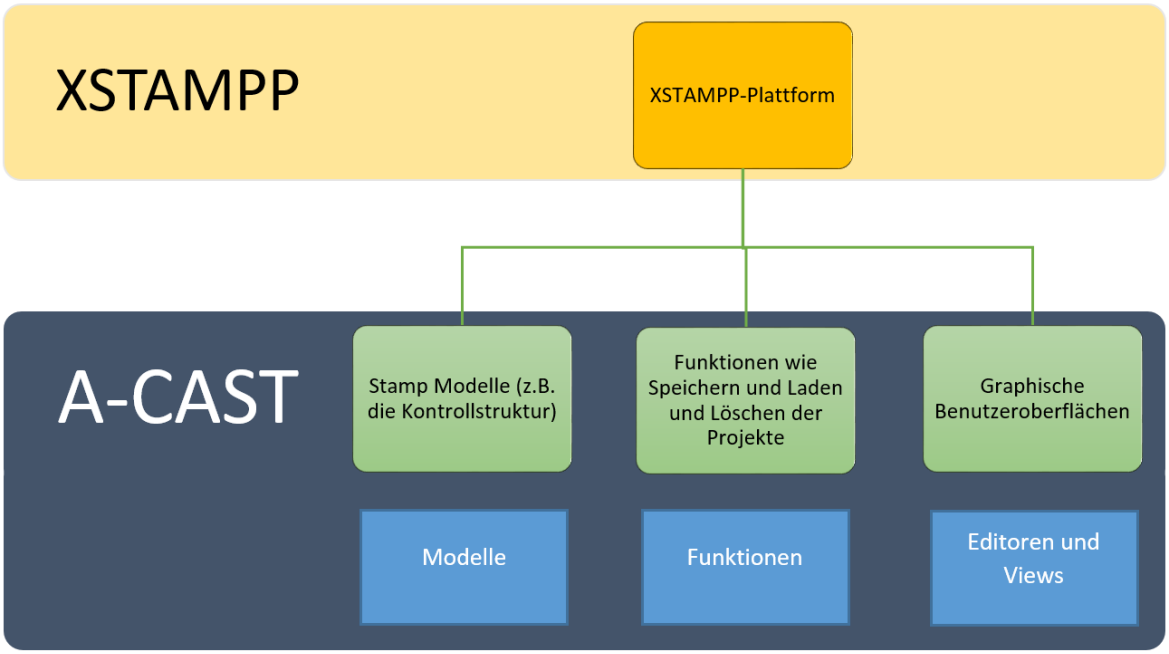


Abbildung 3.1: Design von A-CAST. Die blauen Rechtecke stehen für Parts welche nur aus A-CAST stammen. Die grünen Rechtecke stehen für Teile die aus der XSTAMPP-Plattform übernommen wurden.

In Abbildung 3.1 wird das Design der Implementierung von A-CAST gezeigt. Die Implementierung lässt sich in drei Teile aufsplitten.

Der erste Part sind die Modelle, hiermit sind die Klassen gemeint, welche bestimmte Kernkomponenten von CAST modellieren, beispielsweise die Sicherheitsauflagen, Gefährdungen oder auch die Kontrollstruktur (mehr dazu im Kapitel 2.4 CAST). A-CAST bezieht einen Teil dieser Modelle aus XSTAMPP, so sind beispielsweise die Gefährdungen und Sicherheitsauflagen aus der XSTAMPP-Plattform.

Neben den STAMP-Modellen aus XSTAMPP besitzt A-CAST auch eigene Modelle, welche nur für die CAST-Analyse benötigt werden. Dazu gehören beispielsweise die naheliegenden Events oder auch die Verantwortlichkeiten der einzelnen Rollen im System. Diese Modelle werden benötigt um die Daten später speichern zu können. Im Folgenden wird dies anhand des Modells für naheliegende Events näher beschrieben.

Listing 3.1 Klasse welche ein naheliegendes Event modelliert

```
@XmlRootElement(name = "event")
public class ProximalEvent {


   // Attribute der Events
   private String description;
   private String date;
   private String time;
   private int ID;

   // Konstruktor für die Erstellung eines naheliegenden Events
   public ProximalEvent(String description, String time, String date, int id) {
      this.description = description;
      this.date = date;
      this.time = time;
      this.ID = id;
   }

   // Getter und Setter für die einzelnen Attribute
   ...

}
```

Im obigen Code-Ausschnitt ist die Klasse aufgeführt, welche ein naheliegendes Event beschreibt. Sie besitzt vier Attribute. Eine Beschreibung des Events, ein Datum, eine Uhrzeit und eine ID um die einzelnen Events voneinander unterscheiden zu können. Werden nun im Verlauf der Anwendung von A-CAST Events erstellt und gespeichert, so werden diese in einer XML-Datei abgelegt. Um dies zu ermöglichen muss die Klasse als XML-Element gekennzeichnet werden. Dafür wird über die Klasse eine Annotation in folgender Form angelegt: `@XmlRootElement(name = “event“)`. Das Attribut *name* beschreibt welchen Namen das entsprechende XML-Element hat. Im folgenden Listing ist zu sehen wie Events in der XML-Datei gespeichert werden.

Listing 3.2 Auschnitt der XML-Datei eine gespeicherten Projekt in A-CAST. Abbgebildet wird das XML-Element für ein Event.

```
<event>
  <date>20.10.2015</date>
  <description>Pilot entered Plain</description>
  <ID>1</ID>
  <time>21:38:02</time>
</event>
```

Den nächsten Teil in A-CAST bilden Funktionen. Hiermit ist die Logik gemeint, welche die Abläufe in A-CAST abbilden. Für ein Event gibt es beispielsweise die Funktionen: Hinzufügen,

Löschen, Bearbeiten. Diese Funktionen werden in einer separaten Klasse implementiert. Dort werden die Events abgelegt und wenn gewünscht auch gespeichert.

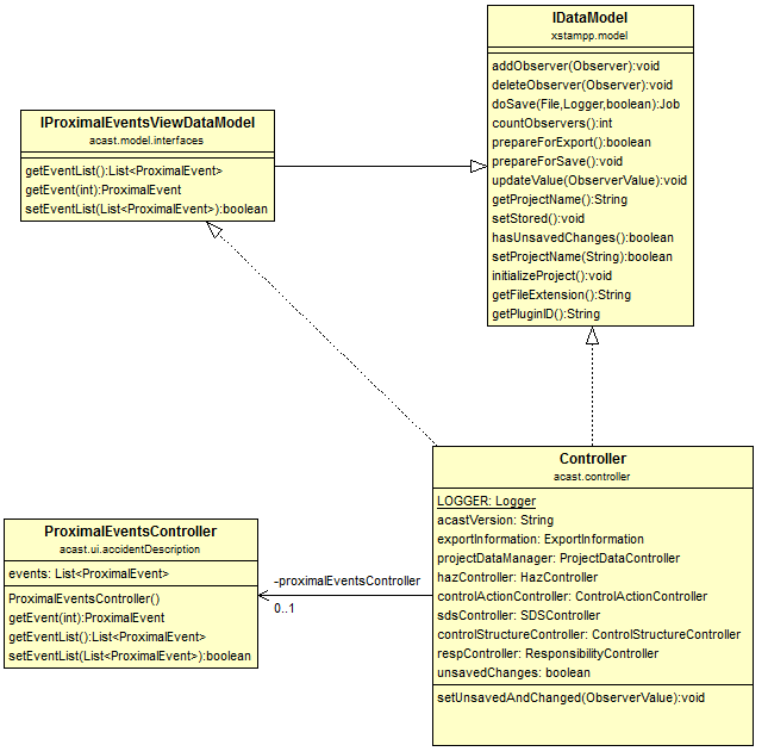


Abbildung 3.2: Zusammenhang der Klassen, welche für die Events in A-CAST verantwortlich sind.

In Abbildung 3.2 ist ein Beispiel zu sehen, welches den Zusammenhang der Klassen, welche für die Events verantwortlich sind, modelliert. Das Interface *IProximalEventsViewDataModel* stellt die Funktionen bereit, welche für die Bearbeitung von Events nötig sind und erbt wie der *Controller* von dem Interface *IDataModel*. Dieses Interface stammt aus XSTAMPP und bietet beispielsweise die Methoden Speichern und Aktualisieren an. Der *Controller* implementiert diese Funktionen, allerdings befindet sich die Logik nicht in dieser Klasse. Für jede Komponente in A-CAST, existiert ein separater Controller der die Logik dieser einzelnen Komponente implementiert. In Abbildung 3.2 ist dies anhand der Komponente der naheliegenden Events dargestellt. In der Klasse *ProximalEventsController* wird die Logik der Funktionen implementiert, beispielsweise das Setzen der Event-Liste oder auch die Methode *getEvent(int)*, welche ein Event anhand der ID zurückliefert. Im *Controller* wird lediglich der *ProximalEventsController* instanziiert und die entsprechende Methoden aufgerufen. Der eben dargestellte Zusammenhang bezieht sich nicht nur auf die Events sondern jede Komponente in A-CAST ist auf die Weise implementiert. So kann leicht ein Überblick über die Funktionalität einer einzelnen Komponente gewonnen werden und jede Komponente besitzt

ihren eigenen Controller. Dies bietet Modularität, wodurch gewünschte Komponenten leicht entfernt, hinzugefügt oder angepasst werden können ohne andere Komponenten in A-CAST zu beeinflussen.

Den letzten Teil in A-CAST bilden die Editoren. Hiermit sind Benutzeroberflächen gemeint, welche die einzelnen Schritte (siehe Kapitel 2.4 CAST) von CAST darstellen sollen. Ein Editor lässt sich durch Doppelklick auf den gewünschten Schritt im Projekt-Explorer öffnen (siehe Abbildung 2.9). In diesen ist es möglich den gewünschten Schritt auszuführen, beispielsweise das Anlegen der Kontrollstruktur. Die Editoren können nebeneinander geöffnet werden, minimiert werden oder auf die gewünschte Größe angepasst werden; sie besitzen die gleichen Funktionen wie beispielsweise ein Editor in Eclipse. Wenn in einem Editor etwas bearbeitet und noch nicht gespeichert wurde, wird links unten im Fenster eine Nachricht angezeigt (siehe Abbildung 2.9). Man kann entweder Änderungen im jeweiligen Editor speichern oder jedes Editors auf einmal speichern.

3.3 Funktionen

A-CAST bietet neben den Funktionen zur Unterstützung von CAST (diese werden im Kapitel 4 Anwendungsbeispiel vorgestellt) auch eine Export-Funktion an. Diese wird in den folgenden Abschnitten näher erläutert.

3.3.1 Export

Mithilfen von A-CAST ist es möglich die erstellten Projekte und deren Daten als PDF, CSV und PNG-Datei zu exportieren. Für die Implementierung des Export wurde XLST (Extensible Stylesheet Language Transformation) verwendet. Damit lassen sich XML-Dateien in andere Formate wie PDF oder CSV konvertieren. Für die Transformation wird eine XSL-Datei angelegt. Diese beschreibt den Aufbau der PDF und welche Elemente aus der XML-Datei für die PDF verwendet werden.

Beim Export ist es möglich die einzelne Schritte und deren Unterpunkte von A-CAST (siehe Kapitel 2.4 CAST) jeweils getrennt voneinander zu exportieren. So kann beispielsweise nur die Unfallbeschreibung oder nur die Kontrollstruktur exportiert werden. Weiterhin besteht auch die Möglichkeit die komplette Analyse zu exportieren. Für den Export stehen zwei unterschiedliche Fenster zu Verfügung (siehe Abbildung 3.2).

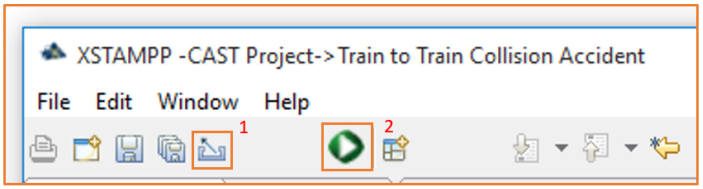


Abbildung 3.3: Auschnitt der Toolbar von A-CAST. Hier befinden sich zwei Buttons für den Export. Button Nr.1 öffnet das Exportfenster wo das gewünschte Exportformat ausgewählt werden kann. Button Nr.2 öffnet ein Export-Fenster, in welchem der Export von allen Formaten (csv, pdf, png) auf einmal möglich ist.

Der Export kann über zwei verschiedene Wizards durchgeführt werden. Durch Klicken des Buttons Nr.2 (siehe Abbildung 3.2) wird ein Export-Fenster geöffnet, in welchem sich alle drei Export-Arten auf einmal durchführen lassen (siehe Abbildung 3.3). Hier kann mithilfe von Checkboxen die gewünschte Exportart gewählt werden. Auf diese Weise muss der Export nicht so oft hintereinander gestartet werden. In der Combo-Box lässt sich das zu exportierende Projekt wählen. Weiterhin kann ein Logo, sowie die Farbe für die Schrift und den Hintergrund festgelegt werden. Für den Export der Kontrollstruktur kann noch zusätzlich gewählt werden, ob die Dekoration der Komponenten mit beziehungsweise nicht mit exportiert werden soll. Nach Betätigen des *Finish-Button* öffnet sich automatisch ein Ordner im gewählten Pfad. in diesem befinden sich die exportierten Dateien.

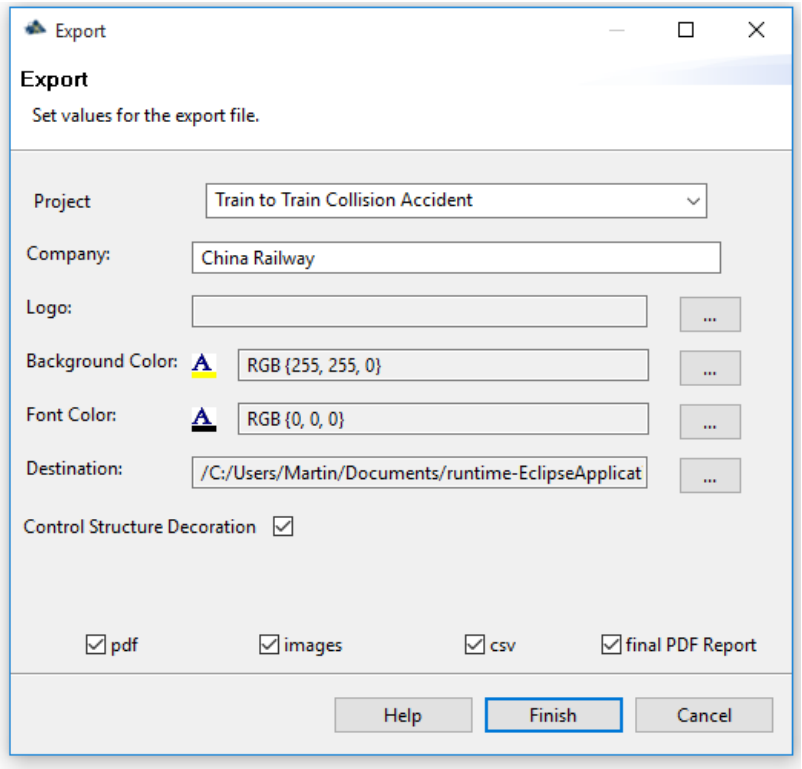


Abbildung 3.4: Export-Fenster in A-CAST. Hier können alle Formate auf einmal exportiert werden.

Durch Klicken des Buttons 1 (siehe Abbildung 3.3) wird das Fenster in Abbildung 3.5 geöffnet. Es ist ebenso möglich das Fenster über das Kontext-Menü im Projekt-Explorer zu öffnen. Hier kann zwischen den einzelnen Export-Formaten in einer Baumstruktur ausgewählt werden. Diese werden in den folgenden Abschnitten näher erläutert.

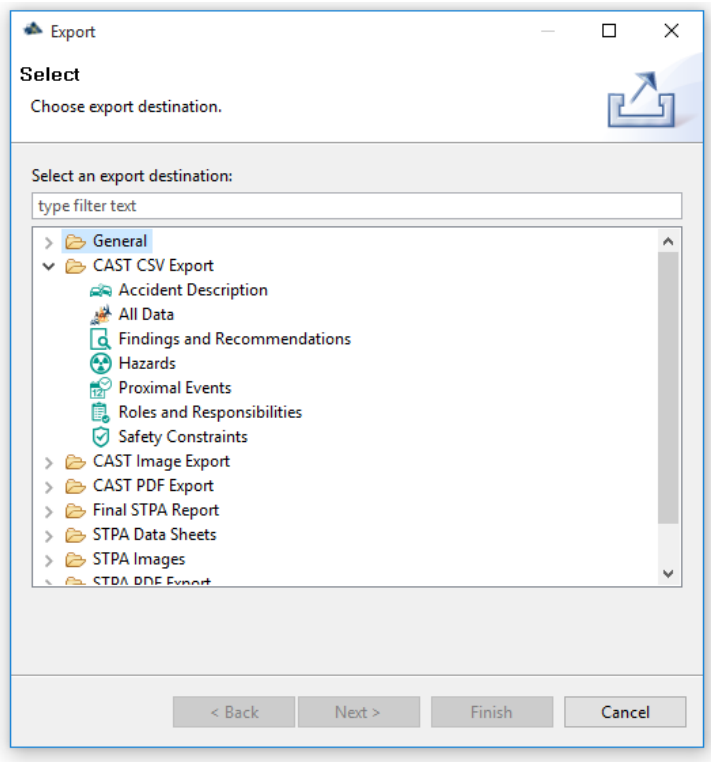


Abbildung 3.5: Export-Fenster in A-CAST. Hier kann zwischen den einzelnen Exportformaten gewählt werden.

Export PDF

Beim PDF-Export ist es möglich jeden der Schritte in A-CAST einzeln oder die gesamte Analyse in einer PDF zu exportieren. Nach Wahl des zu exportierenden Schrittes, kann noch die Farbe für Schrift und Hintergrund ausgewählt werden. Hierfür ist zusätzlich eine Vorschau der gewählten Farben zu sehen (siehe Abbildung 3.6). Danach muss noch der Speicherort gewählt werden und der *Finish-Button* betätigt werden. Der Fortschritt des Exports wird unten in der Leiste von A-CAST angezeigt. Verlief der PDF-Export erfolgreich öffnet sich automatisch die eben exportiere PDF-Datei. In Abbildung 3.7 ist ein Beispiel für die PDF-Datei der naheliegenden Schritte zu sehen.

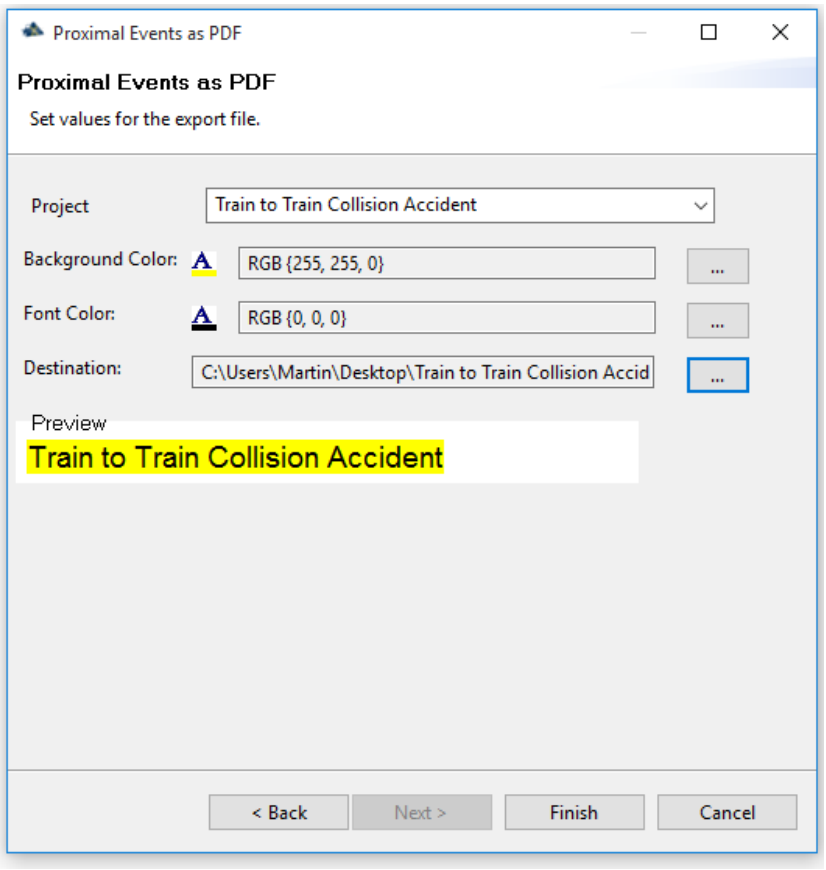
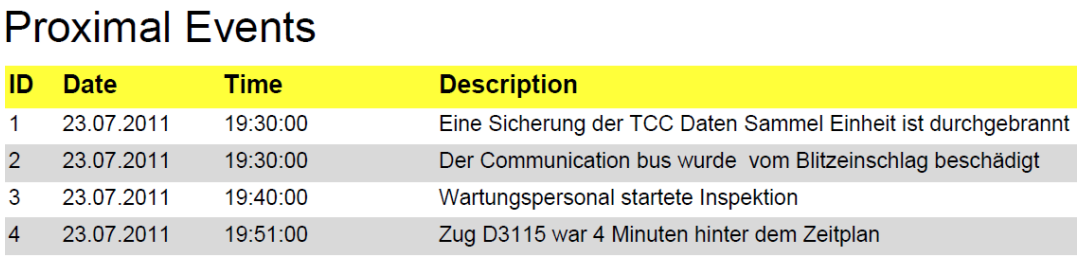


Abbildung 3.6: PDF-Export A-CAST.



Proximal Events

ID	Date	Time	Description
1	23.07.2011	19:30:00	Eine Sicherung der TCC Daten Sammel Einheit ist durchgebrannt
2	23.07.2011	19:30:00	Der Communication bus wurde vom Blitzeinschlag beschädigt
3	23.07.2011	19:40:00	Wartungspersonal startete Inspektion
4	23.07.2011	19:51:00	Zug D3115 war 4 Minuten hinter dem Zeitplan

Abbildung 3.7: Auschnitt der PDF-Datei der naheliegenden Schritte von A-CAST.

Export CSV

Der CSV-Export verläuft ähnlich wie der PDF-Export. Nach Auswahl des zu exportierenden Schrittes ist es noch möglich den Separator auszuwählen (siehe Abbildung 3.8). Die weiteren Schritte verlaufen genauso wie beim PDF-Export. In Abbildung 3.9 ist ein Beispiel einer solchen CSV-Datei zu sehen.

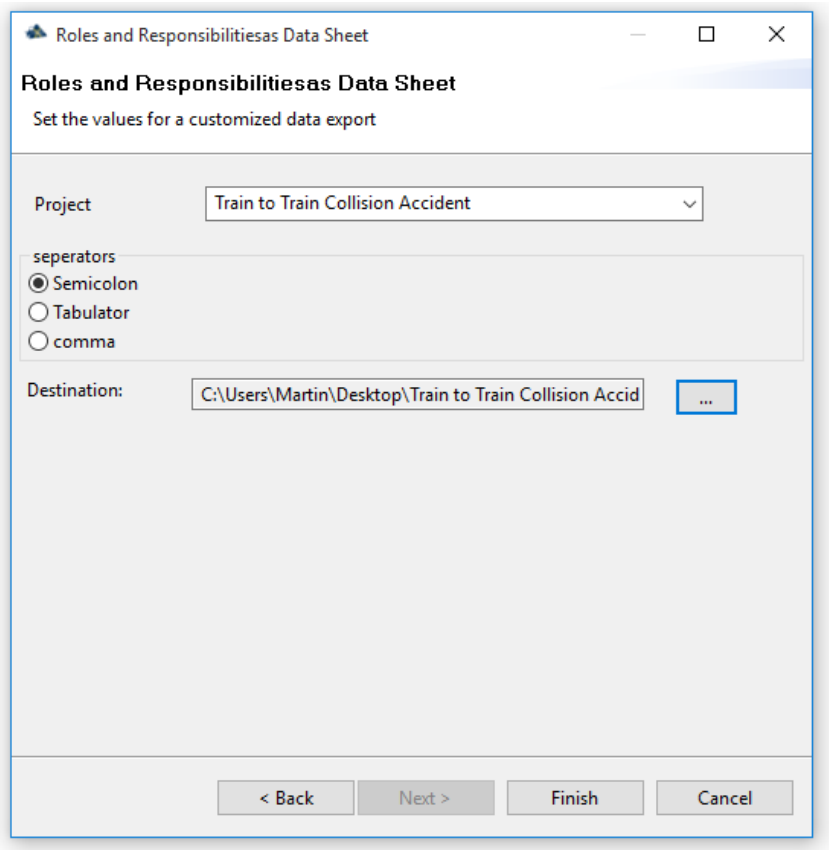


Abbildung 3.8: CSV-Export A-CAST.

Proximal Events

ID	Date	Time	Description
1	23.07.2011	19:30:00	Eine Sicherung der TCC Daten Sammel Einheit ist durchgebrannt
2	23.07.2011	19:30:00	Der Communication bus wurde vom Blitzeinschlag beschädigt
3	23.07.2011	19:40:00	Wartungspersonal startete Inspektion
4	23.07.2011	19:51:00	Zug D3115 war 4 Minuten hinter dem Zeitplan

Abbildung 3.9: Auschnitt der CSV-Datei der naheliegenden Schritte in A-CAST.

Export PNG

Der Export von PNG-Dateien verläuft komplett identisch zum PDF-Export. Die Fenster und Schritte für den Export entsprechen genau denen vom PDF-Export (siehe Abbildung 3.6).

3.4 Erweiterungen

Da A-CAST in Zukunft um weitere Funktionalitäten erweitert werden könnte, wird in diesem Abschnitt erläutert wie ein neuer Editor in A-CAST angelegt werden kann. Dafür wird im folgenden eine kurze Anleitung beschrieben, welche in einem simplem Beispiel darstellt, wie ein neuer Editor angelegt werden kann. Wichtig ist, dass hierfür eine **Eclipse RCP-Version** verwendet werden sollte.

3.4.1 Schritt 1

Zunächst sollte eine neue Klasse angelegt werden, die von *StandartEditorPart* ableitet. *StandartEditorPart* ist eine abstrakte Klasse aus XSTAMPP und bietet bereits die Funktion für das Speichern und Updaten. Im Listing 3.3 ist ein Beispiel für eine neu erstellte Editor-Klasse zu sehen.

Listing 3.3 Beispiel für eine neu erstellte Editor-Klasse in A-CAST

```
public class MyNewEditor extends StandardEditorPart{

   @Override
   public String getId() {
      return null;
   }

   @Override
   public void createPartControl(Composite parent) {
      // Hier kann das Layout und Design des Editor angelegt werden
   }
}
```

3.4.2 Schritt 2

Jetzt sollte ein neuer Editor in der *xstampp.acastr plugin.xml* angelegt werden. Hierfür wird im Tab *Extensions* die Erweiterung **org.eclipse.ui.editors** ausgewählt. Durch einen Rechtsklick öffnet sich ein Kontextmenü in dem unter dem Reiter *new* der Punkt *editor* ausgewählt werden sollte. Nun muss noch die eben erstellte Klasse unter dem Punkt *class* ausgewählt werden und eine ID für den Editor vergeben werden. In Abbildung 3.3 ist ein Beispiel hierfür zu sehen.

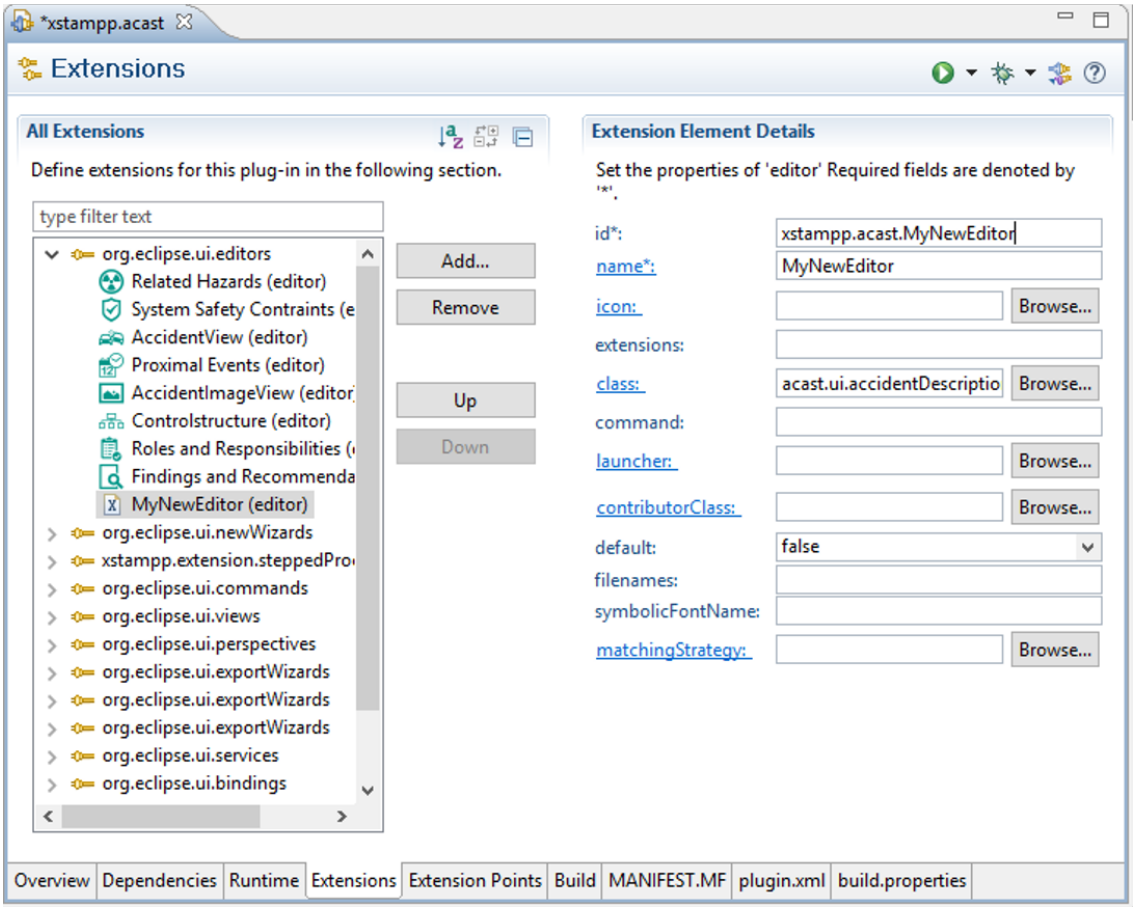


Abbildung 3.10: Erstellung eines neuen Editors in A-CAST.

3.4.3 Schritt 3

Jetzt muss der Editor nur noch dem Projekt-Explorer hinzugefügt werden. Dafür muss wieder der *Extensions-Tab* in der *xstampp.acast plugin.xml* geöffnet werden. Diesmal wird die Erweiterung **xstampp.extension.steppedProcess** ausgewählt. Durch einen Rechtsklick öffnet sich ein Kontextmenü in dem unter der Reiter *new* der Punkt *step* ausgewählt werden sollte. Jetzt muss die EditorId, welche vorhin erstellt wurde unter dem Punkt *editorId*:* eingetragen werden und eine ID für den eben erstellen Schritt selbst vergeben werden. In Abbildung 3.4 wird dieser Ablauf nochmal dargestellt.

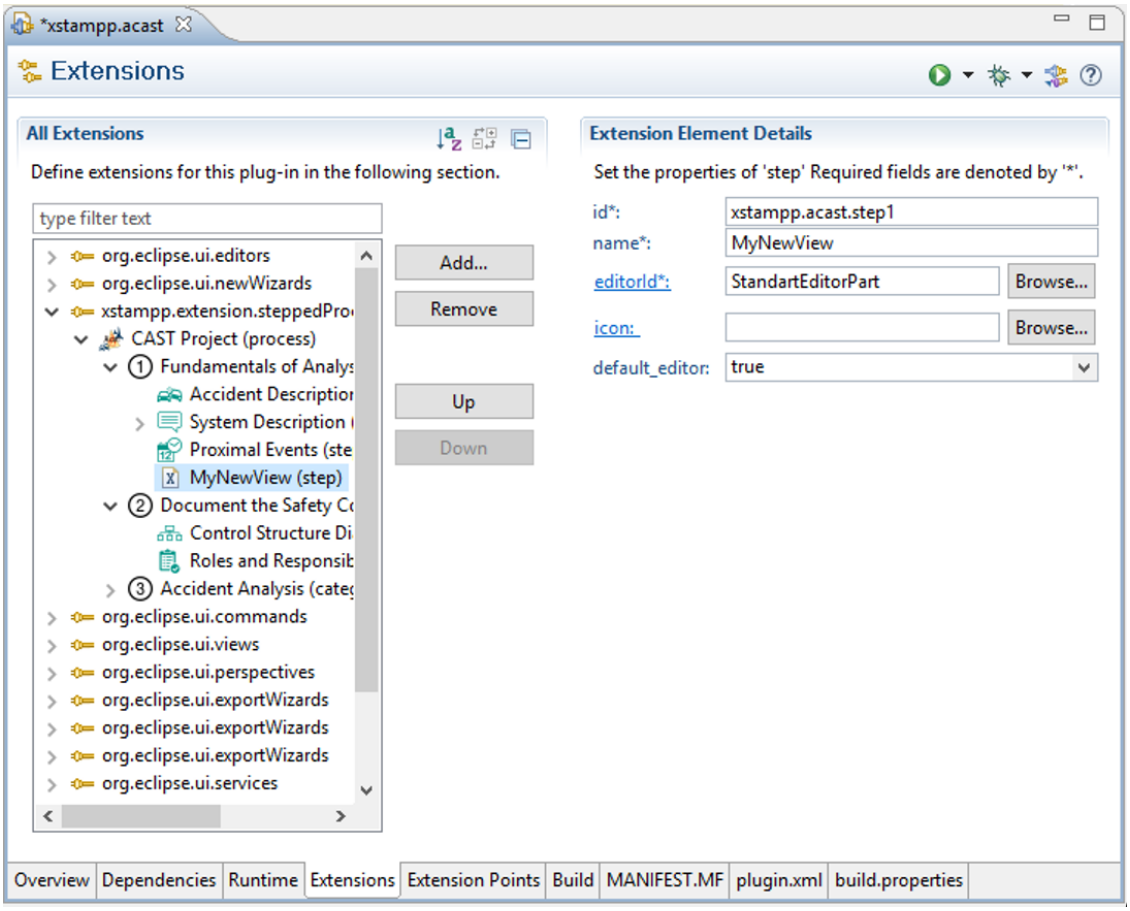


Abbildung 3.11: Erstellung eines neuen Schritts im Projekt-Explorer von A-CAST.

4 Anwendungsbeispiel

In diesem Kapitel wird das Tool A-CAST anhand einer bereits durchgeführten Analyse mit CAST vorgestellt. Dabei werden die einzelnen Schritte der Analyse mithilfe von A-CAST erstellt. Als Beispiel dient ein Zugunfall in China den Airong Dong mithilfe von CAST analysiert hat [D$^+$12]. Die Daten der Analyse stammen komplett aus seiner Ausarbeitung und werden in diesem Kapitel lediglich mit A-CAST modelliert.

Dabei soll der Fokus aber nicht auf dem Beispielunfall liegen, sondern auf dem Tool A-CAST. Es wird darauf eingegangen, auf welche Weise die einzelnen Schritte durchgeführt werden und welche Aktionen dafür nötig sind. Ebenso wird für jeden Schritt auch die entsprechende graphische Oberfläche gezeigt und beschrieben, auf welche Art mit ihr interagiert werden kann.

4.1 Informationen zum Unfall

Zunächst ein paar Information zu dem Unfall. Es handelt sich um ein Zusammenprall zweier Züge in China auf der Yong-Wen High Speed Linie in Wenzhou City am 23. Juli 2011 um 20:30:05 Uhr. Der Zug D301, mit einer Geschwindigkeit von 99 km/h, stieß mit dem Zug D3115 zusammen, welcher in dieselbe Richtung fuhr aber nur eine Geschwindigkeit von 16 km/h besaß. Der Zusammenstoß hatte die Entgleisung von den letzten 2 Wagons des Zuges D3115 und den 5 ersten Wagons der Zuges D301 zufolge. Bei diesem Unfall sind 40 Menschen verunglückt und es gab 120 Verletzte. Der sachliche Schaden wurde auf 193,7 Millionen Yuan geschätzt, was bei heutigem Wechselkurs circa 26,8 Millionen Euro entspricht [D$^+$12].

4.2 Durchführung der Schritte von CAST

4.2.1 Schritt 1: Unfall Beschreibung

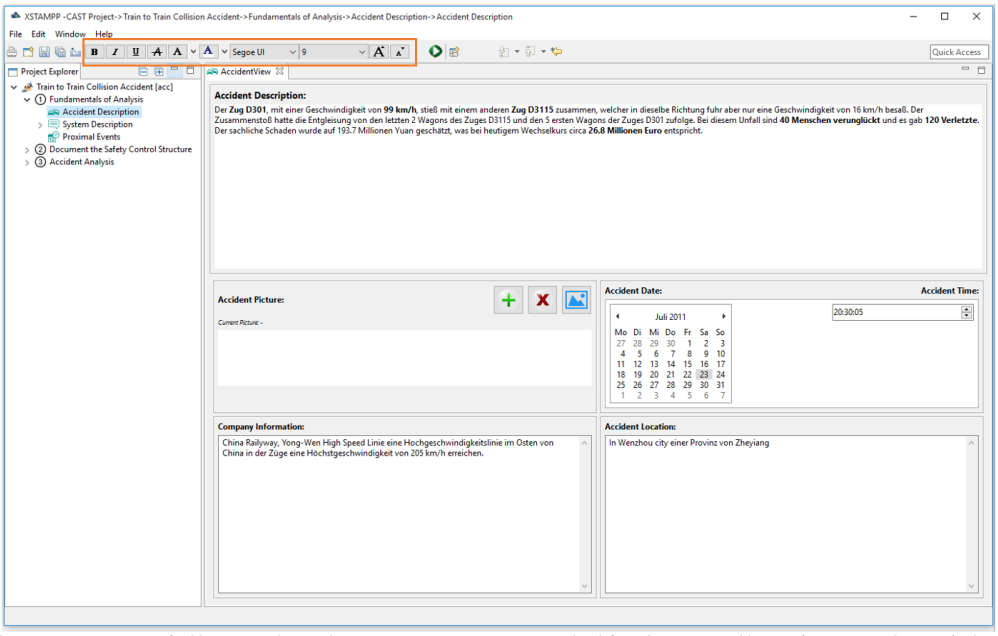


Abbildung 4.1: Unfall-Beschreibung A-CAST. Mithilfe der Toolbar (rot makiert) lässt sich der Text an die gewünschte Form anpassen.

In Abbildung 4.1 ist der Editor für die Unfall-Beschreibung zu sehen. Allgemein soll mithilfe dieses Editor ein kurzer Überblick über den Unfall erstellt werden. Dies soll dabei helfen schnell die nötigen Information des Unfalls zu finden, falls diese benötigt werden. Um einen guten Überblick zu bieten, sind die Informationen des Unfalls in fünf Teile aufgeteilt:

- Unfall-Beschreibung
- Unfall-Bilder
- Unfall-Datum und Zeitpunkt
- Unfall-Ort
- Firmeninformation (Firmen, welche am Unfall beteiligt waren)

In der Unfall-Beschreibung werden die grundlegenden Informationen des Unfalls festgehalten. Zusätzlich können weitere Details des Unfalls festgehalten werden. Im Beispiel des Zugunfalls wurden die Geschwindigkeit der beiden Züge sowie die Opfer, Verletzten und der sachliche Schaden festgehalten. Es gibt zusätzlich eine Funktion um den Text weiter zu bearbeiten. So kann mithilfe des Text-Editors in der Toolbar (siehe Abbildung 4.1) das Layout des Textes

angepasst werden. Hierbei werden die Standardfunktionen unterstützt, wie beispielsweise Ändern der Schriftart, fett, kursiv, unterstrichen und auch die Schriftgröße kann angepasst werden. Auf diese Weise können wichtige Informationen hervorgehoben werden.

In dem Feld für die *Unfall-Bilder* lassen sich beliebig viele Bilder des Unfalls hinzufügen. Hierfür muss auf den *Hinzufügen-Button* (siehe Abbildung 4.2) gedrückt werden, daraufhin öffnet sich ein File-Chooser, in welchem ein Bild ausgewählt werden kann. In der Liste wird dann der Pfad des Bildes angezeigt. Wird ein Element in der Liste markiert, kann es durch Drücken des *Löschen-Button* (siehe Abbildung 4.2) entfernt werden. Um das entsprechende Bild anzuzeigen muss es in der Liste markiert werden und dann der Button *Bild-Anzeigen* (siehe Abbildung 4.2) gedrückt werden. Daraufhin öffnet sich ein neuer Editor, in dem das ausgewählte Bild angezeigt wird.

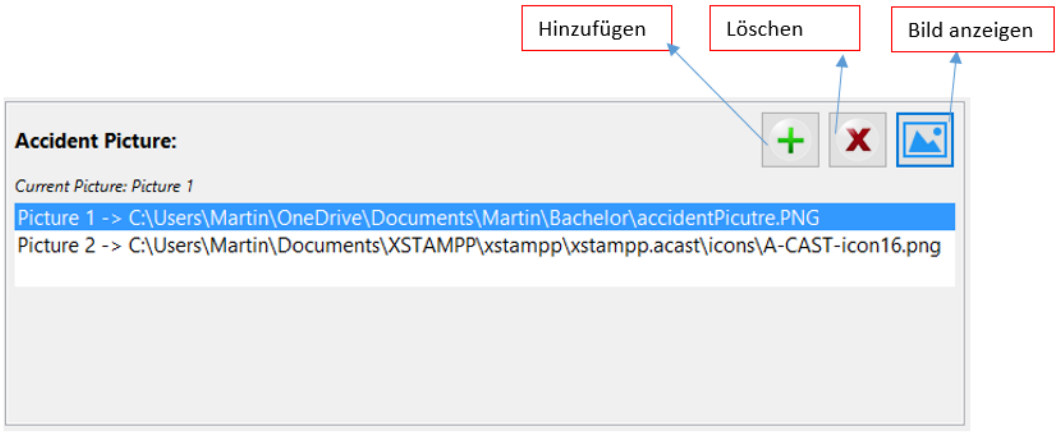


Abbildung 4.2: Hier können Bilder des Unfalls hinzugefügt, gelöscht und angezeigt werden. Zusätlich wird der Pfad des Bildes in der Liste angezeigt.

In dem Feld *Unfall-Datum* kann das Datum und der Zeitpunkt des Unfalls gewählt werden. Für das Auswählen des Datums wird das SWT Date Time Objekt verwendet. Zu sehen ist der Date-Chooser in Abbildung 4.1. Beim *Unfall-Ort* werden detaillierte Informationen über den Unfall angegeben. Hierzu steht ein simples Textfeld zu Verfügung, in dem die Informationen festgehalten werden können. Im Feld *Firmeninformation* werden Informationen über die Firma aufgeführt, welche primär am Unfall beteiligt ist. Hierzu steht wieder ein simples Textfeld zur Verfügung in welchem die nötigen Details über die Firma notiert werden können.

Der nächste Editor im 1. Schritt ist der Hazard-Editor. Hier werden die Gefährdungen, welche am Unfall vorhanden waren, aufgelistet. Durch einen Klick auf den Hinzufüge-Button wird ein neues Element in der Liste angelegt. Hier wird die Gefährdung beschrieben. Falls die Beschreibung etwas länger sein sollte, gibt es auf der rechten Seite eine Textbox, in der eine längere Beschreibung Platz findet (siehe Abbildung 4.3). Durch einen Klick auf eine Gefährdung in der Liste wird die entsprechende Beschreibung angezeigt.

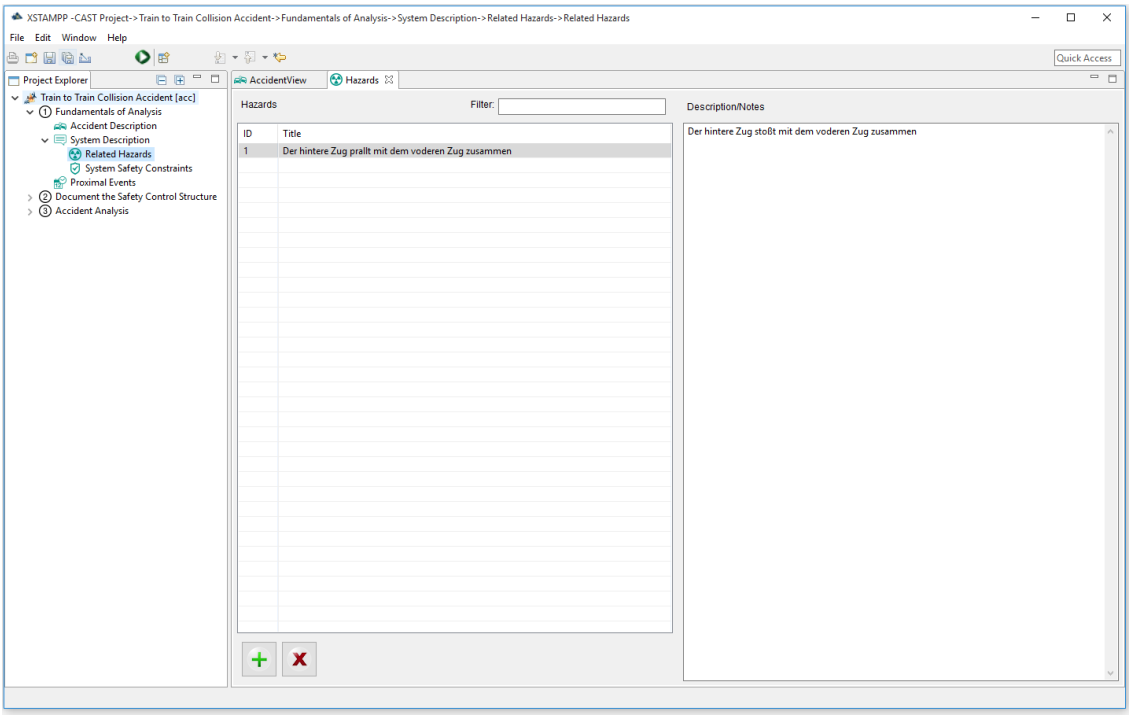


Abbildung 4.3: Editor in dem Hazards (Gefährdungen) hinzugefügt und beschrieben werden können.

Im nächsten Editor (siehe Abbildung 4.4) werden die Sicherheitsauflagen angelegt. Hier werden nur Sicherheitsauflagen aufgelistet, welche für das gesamte System gelten und beim Unfall nicht eingehalten wurden. Sollten die entsprechenden Sicherheitsauflagen fehlen, werden sie hier ergänzt. Durch Klicken des Hinzufüge-Buttons wird eine neue Sicherheitsauflage erstellt, in welcher wiederum eine kurze und längere Beschreibung hinzugefügt werden kann. Die graphische Oberfläche ist die gleiche wie bei den Gefährdungen.

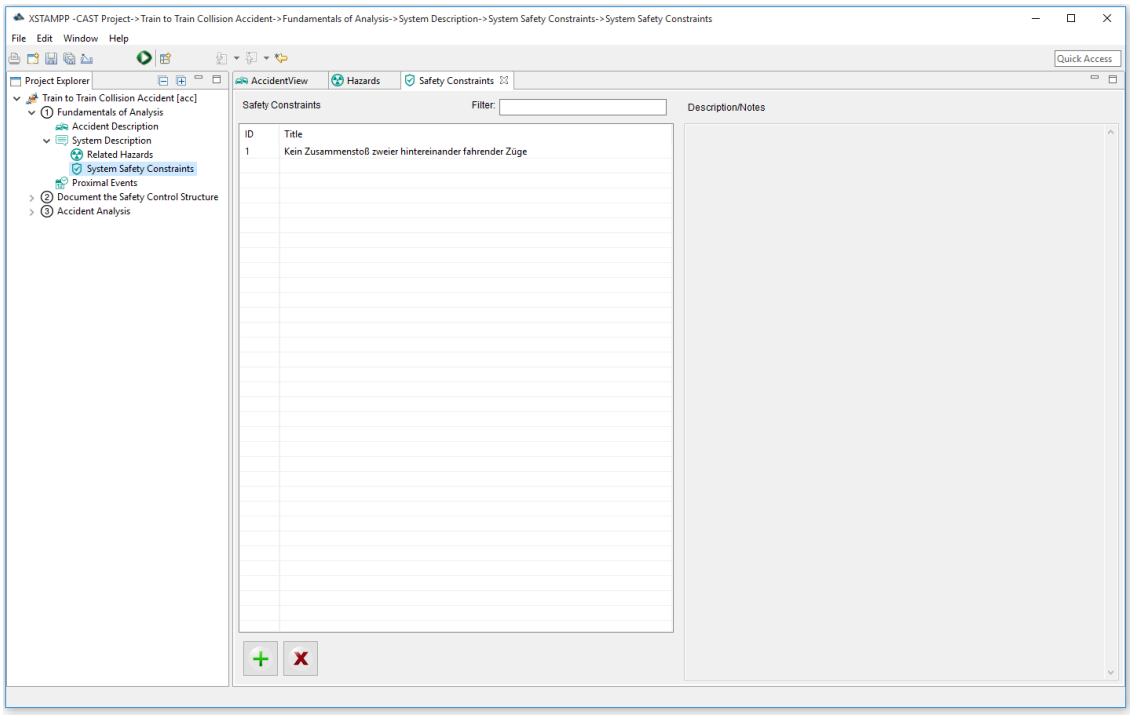


Abbildung 4.4: Editor, in welchem Sicherheitauflagen hinzugefügt und beschrieben werden können.

Der Editor für die naheliegenden Events (siehe Abbildung 4.5) ermöglicht das Hinzufügen und Löschen von Events. Diese besitzen eine ID für die Identifizierung, ein Datum und eine Uhrzeit, sowie eine Beschreibung. Der Editor stellt die Informationen in einer Tabelle da, durch Doppel-Klick auf das entsprechende Feld kann der Wert bearbeitet werden. Um ein neues Event hinzuzufügen muss der Hinzufüge-Button gedrückt werden, darauf hin werden die ID und das heutige Datum und Uhrzeit bereits als Standard-Werte eingetragen. Löschen ist nur möglich, wenn das entsprechende Event markiert wurde und dann der Löschen-Button gedrückt wird. Für das Editieren des Datums und der Uhrzeit wird ein simpler Datums- beziehungsweise Zeit-Auswahldialog eingeblendet. Auf diese Weise ist es nicht möglich, dass falsche Uhrzeiten oder Daten eingetragen werden können. In Abbildung 4.6 ist der Auswahldialog zu sehen.

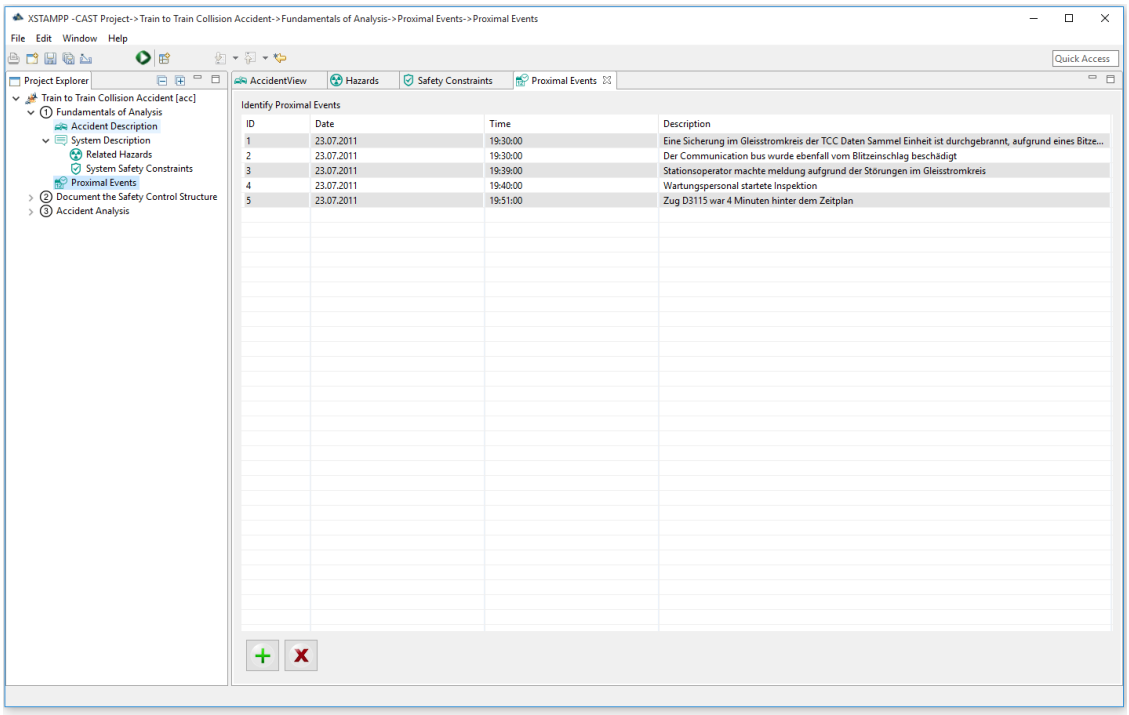


Abbildung 4.5: Editor in welchem naheliegende Events, welche kurz vor dem Unfall passiert sind hinzugefügt werden können.

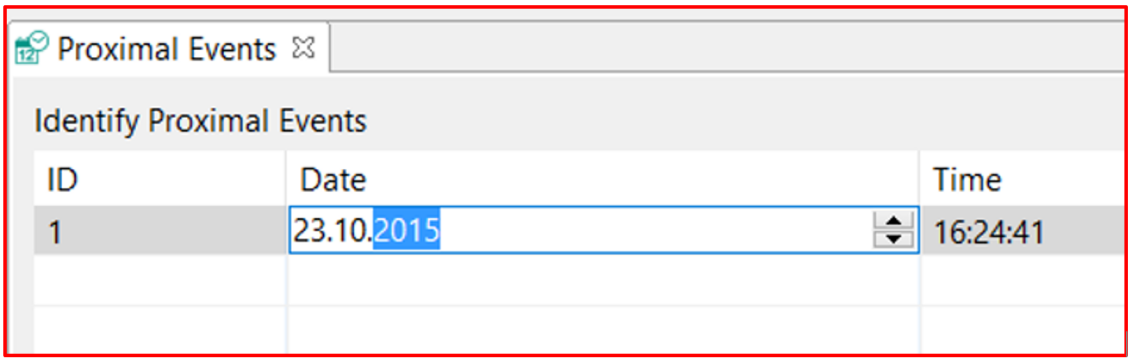


Abbildung 4.6: Auswahlsdialog für das Setzen des Datums und der Uhrzeit eines naheliegenden Events.

4.2.2 Schritt 2: Anlegen der Kontrollstruktur

Die Kontrollstruktur wurde komplett aus XSTAMPP übernommen, sie ist genau dieselbe wie bei A-STPA. Sie bietet die Komponenten und Funktionen an welche bereits in Kapitel 2.3 beschrieben wurden. Ein Beispiel für eine Kontrollstruktur ist in Abbildung 4.7 zu sehen. Die einzelnen Komponenten lassen sich per Drag-and-Drop aus der Komponenten-Leiste auf

der rechten Seite (siehe Abbildung 4.7) anlegen. Dabei kann für jede Komponente ein Name vergeben werden. Wichtig ist dass jede Komponenten einen unterschiedlichen Namen besitzen muss. Vergibt der Benutzer einen Namen für eine Komponente der bereits existiert wird an den Namen die Endung „(2)“ angehängt. Existiert also beispielsweise eine Komponente mit dem Namen „Controller“ und ich vergebe eine anderen Komponenten diesen Namen, so wird dieser Name automatisch zu „Controller (2)“ umbenannt. Der Grund dafür ist, dass im Verlauf der Analyse mit CAST für Rollen im System Verantwortlichkeiten und Funde festgehalten werden. Damit später dabei keine Verwechslungen entstehen, sollte jede Rolle im System einen unterschiedlichen Namen besitzen.

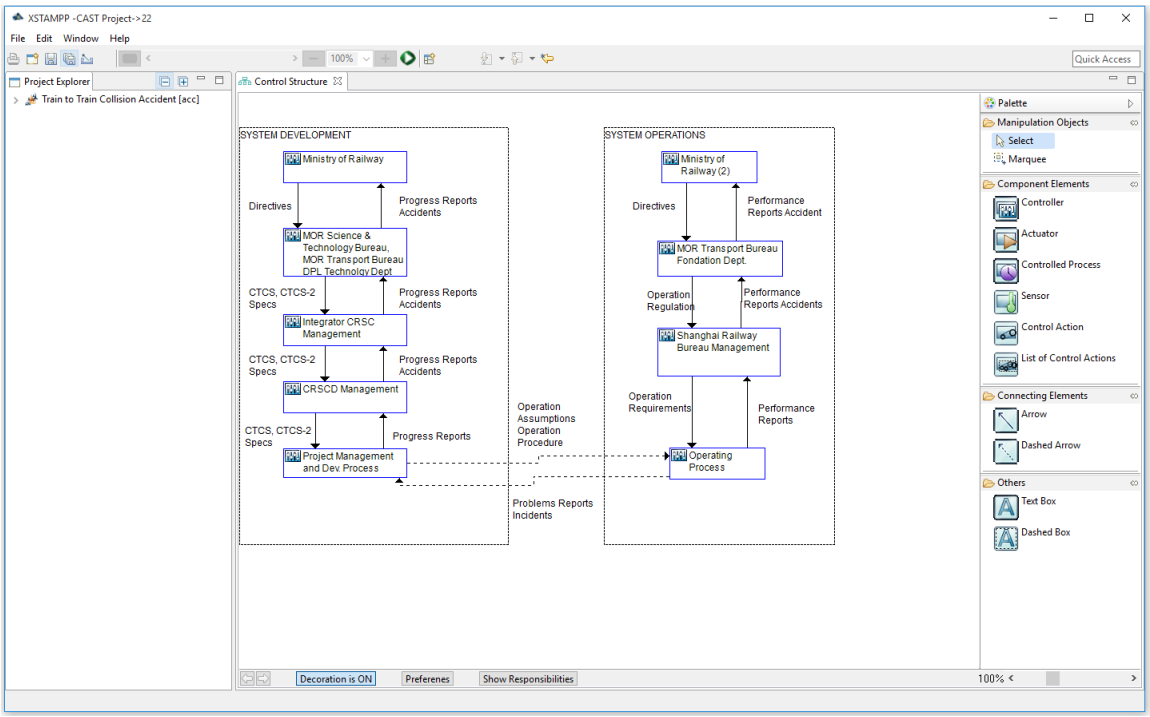


Abbildung 4.7: Editor für die Kontrollstruktur, hier kann diese entworfen und beschrieben werden.

Im Editor für die Kontrollstruktur ist es möglich Verantwortlichkeiten für die einzelnen Komponenten in der Kontrollstruktur festzulegen (siehe Kapitel 2.4.2). Um die Verantwortlichkeiten festzulegen, lässt sich ein zusätzliches Fenster aus dem Editor der Kontrollstruktur heraus öffnen. Durch Klicken des *Show Responsibilities-Button* unten in der Leiste des Kontrollstruktur-Editors oder durch Doppelklick auf eine der Komponenten in der Kontrollstruktur wird das „Verantwortlichkeiten-Fenster“ geöffnet (siehe Abbildung 4.8). Hier lassen sich Verantwortlichkeiten durch Klicken des Hinzufüge-Buttons anlegen und wenn gewünscht auch wieder löschen. Um zwischen den einzelnen Verantwortlichkeiten hin- und herzuwechseln können die Buttons auf der linken Seite des Fensters (siehe Abbildung 4.8) betätigt werden. Die gerade ausgewählte Verantwortlichkeit wird durch einen roten Rahmen

um den entsprechenden Button ausgedrückt. Mithilfe der Combo-Box kann zwischen den einzelnen Rollen im System gewechselt werden.

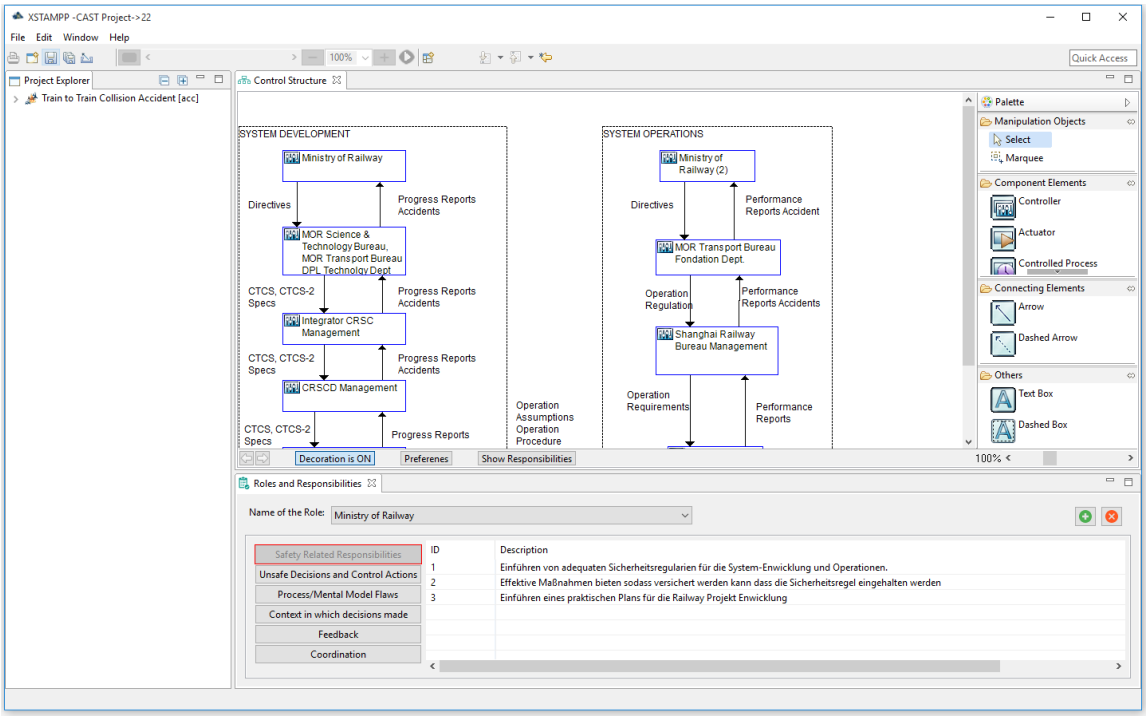


Abbildung 4.8: Editor um Verantwortlichkeiten für einzelne Rollen im System festzulegen. Die einzelnen Rollen können über ein Combo-Box ausgewählt werden.

4.2.3 Schritt 3: Unfall Analyse

Im letzten Schritt werden die Ergebnisse und Funde der Analyse, sowie Empfehlungen für die einzelnen Rollen im System festgehalten. Durch Drücken des Hinzufüge-Buttons lassen sich beliebig viele Funde hinzufügen. Diese werden im Editor in Tabellenform festgehalten. Wählt man einen bereits hinzugefügten Fund aus, so kann dieser über den Löschen-Button entfernt werden. Die Ergebnisse lassen sich wieder für jede Komponente im System separat festhalten. Durch die Combo-Box können die einzelnen Komponenten auswählt werden (siehe Abbildung 4.8).

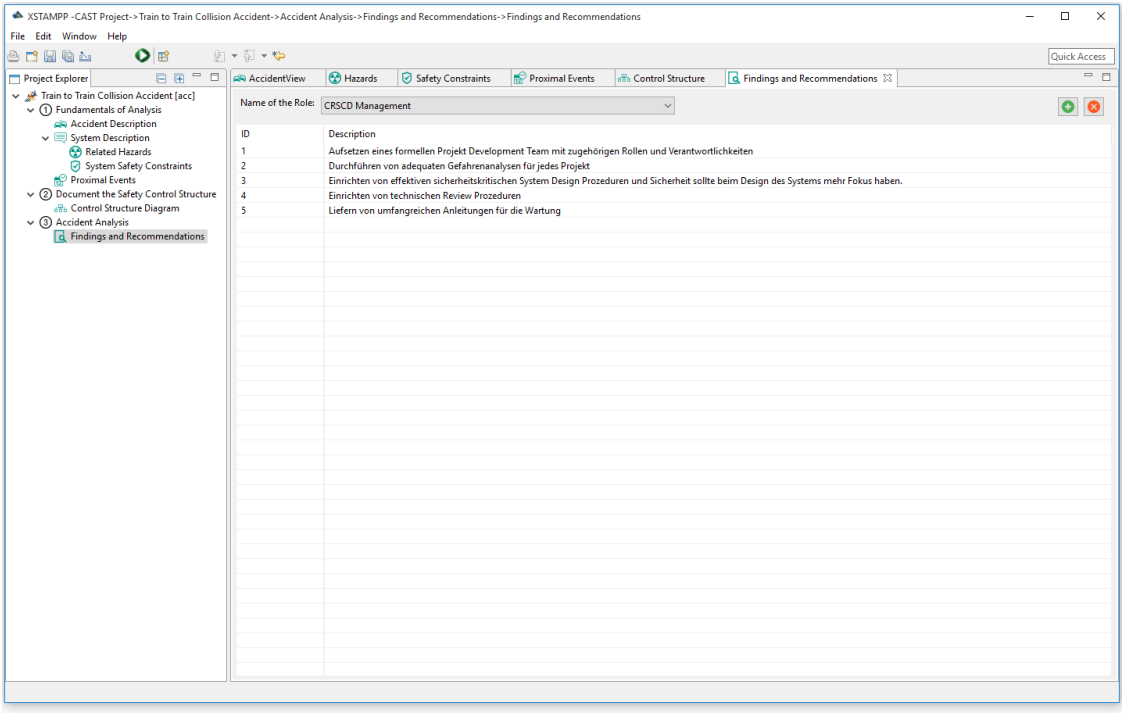


Abbildung 4.9: Editor, in welchem Empfehlungen und Funde der Analyse hinzugefügt und beschrieben werden können.

5 Evaluation

In diesem Kapitel wird eine Auswertung des Tools A-CAST gezeigt. Dazu wurde eine kurze Teststudie durchgeführt, welche hier vorgestellt wird. Um die Ergebnisse festzuhalten wurde ein Fragebogen erstellt und ausgewertet.

5.1 Test-Studie

Für die Evaluation von A-CAST wurde eine Test-Studie durchgeführt. Hierfür wurden fünf Studenten der Softwaretechnik gebeten, A-CAST zu testen und anschließend einen Fragebogen auszufüllen. Für den Test wurden die Daten des Zugunfalls hinzugezogen, welcher bereits in Kapitel 4.1 vorgestellt wurden.

5.1.1 Aufbau der Test-Studie

Zunächst wurde den Testern eine kurze Beschreibung von CAST und STAMP vorgelegt, da diese mit der XSTAMPP-Plattform noch nicht vertraut waren. Zusätzlich bekamen sie Beispiel-Daten eines Zugunfalls, welche sie mithilfe von A-CAST modellieren sollten. Um Informationen über die einzelnen Schritte von A-CAST zu erhalten, stand ihnen ein Hilfe-Fenster (siehe Abbildung 5.1) zu Verfügung. Dort konnten sie Informationen finden, wie die einzelnen Schritte der Analyse in A-CAST durchgeführt werden können. Die Aufgabe der Tester war es, die Informationen aus den Beispiel-Daten in A-CAST einzutragen und anschließend einen Fragebogen auszufüllen.

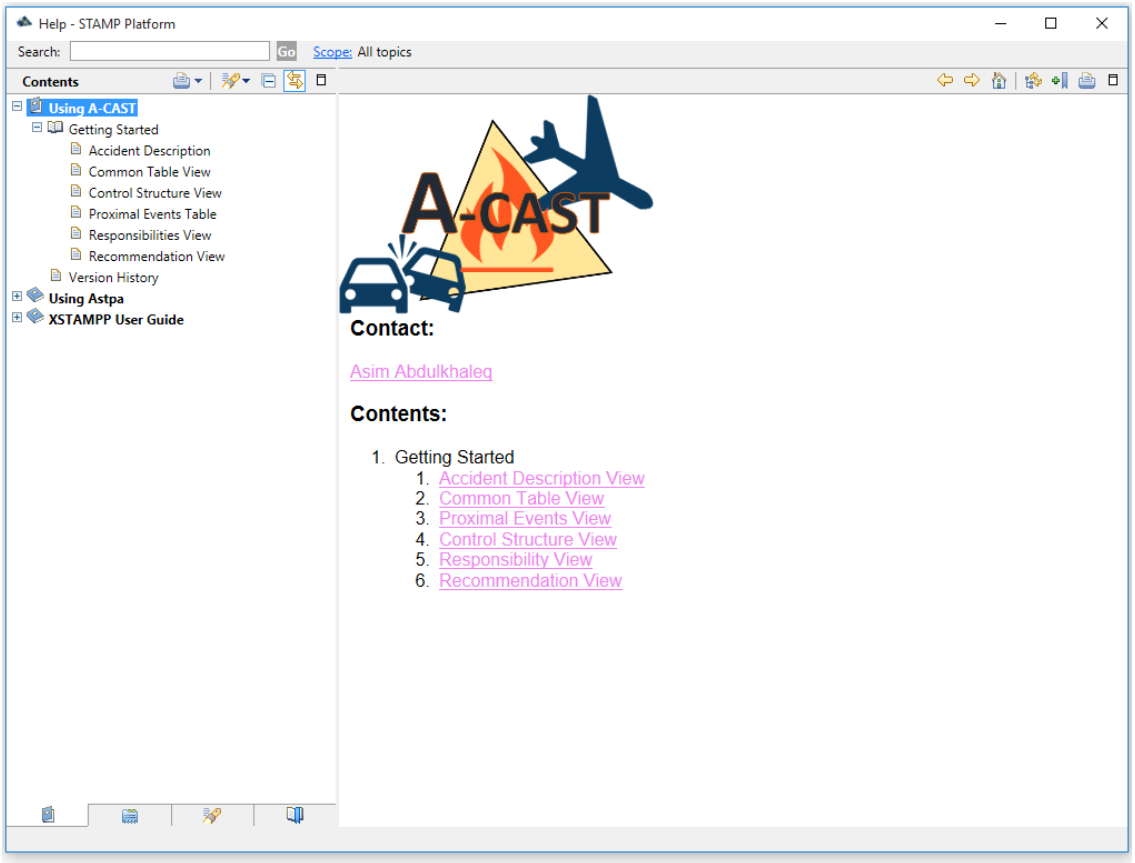
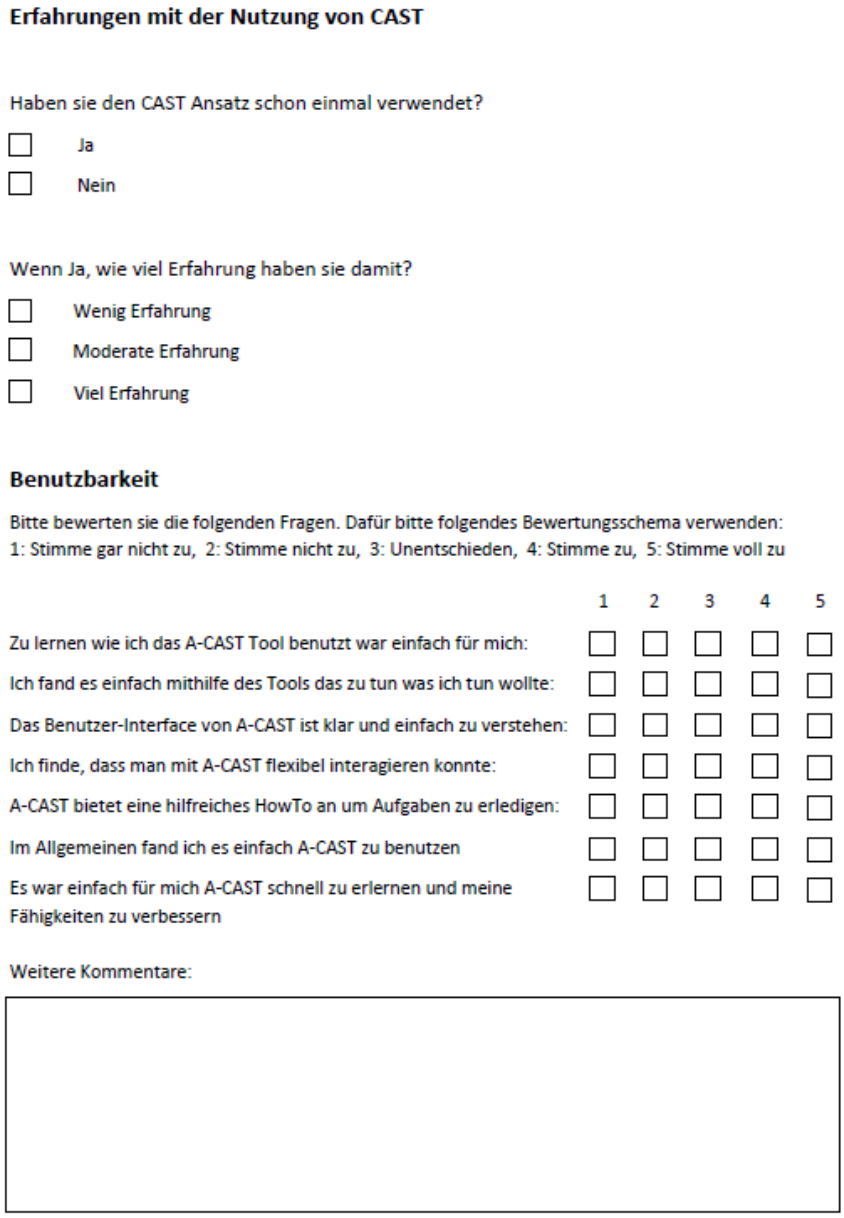


Abbildung 5.1: Hilfe-Fenster von A-CAST. Hier werden alle grundlegenden Funktionen erläutert.

5.1.2 Fragebogen

Den Testern wurde nach Eintragen der Beispiel-Daten in A-CAST folgender Fragebogen ausgeteilt, welcher in den Abbildungen 5.2, 5.3, 5.4, und 5.5 zu sehen ist.



Erfahrungen mit der Nutzung von CAST

Haben sie den CAST Ansatz schon einmal verwendet?

☐ Ja

☐ Nein

Wenn Ja, wie viel Erfahrung haben sie damit?

☐ Wenig Erfahrung

☐ Moderate Erfahrung

☐ Viel Erfahrung

Benutzbarkeit

Bitte bewerten sie die folgenden Fragen. Dafür bitte folgendes Bewertungsschema verwenden:
1: Stimme gar nicht zu, 2: Stimme nicht zu, 3: Unentschieden, 4: Stimme zu, 5: Stimme voll zu

	1	2	3	4	5
Zu lernen wie ich das A-CAST Tool benutzt war einfach für mich:	☐	☐	☐	☐	☐
Ich fand es einfach mithilfe des Tools das zu tun was ich tun wollte:	☐	☐	☐	☐	☐
Das Benutzer-Interface von A-CAST ist klar und einfach zu verstehen:	☐	☐	☐	☐	☐
Ich finde, dass man mit A-CAST flexibel interagieren konnte:	☐	☐	☐	☐	☐
A-CAST bietet eine hilfreiches HowTo an um Aufgaben zu erledigen:	☐	☐	☐	☐	☐
Im Allgemeinen fand ich es einfach A-CAST zu benutzen	☐	☐	☐	☐	☐
Es war einfach für mich A-CAST schnell zu erlernen und meine Fähigkeiten zu verbessern	☐	☐	☐	☐	☐

Weitere Kommentare:

Abbildung 5.2: Erste Seite des Fragebogen zu A-CAST.

Schritte von A-CAST

Bitte bewerten sie die folgenden Fragen. Dafür bitte folgendes Bewertungsschema verwenden:
1: umständlich, 2: zu aufwändig, 3: befriedigend, 4: gut , 5: sehr gut

Wie einfach fanden sie es die einzelnen Schritte von A-CAST durchzuführen?

	1	2	3	4	5
Anlegen der Unfall-Beschreibung (Step 1):	☐	☐	☐	☐	☐
Anlegen der Safety Constraints (Step 1):	☐	☐	☐	☐	☐
Die Gefährdungen identifizieren (Step 1):	☐	☐	☐	☐	☐
Naheliegende Ereignisse festhalten (Step1):	☐	☐	☐	☐	☐
Kontrollstruktur anlegen (Step 2):	☐	☐	☐	☐	☐
Verantwortlichkeiten der Rollen anlegen (Step 2):	☐	☐	☐	☐	☐
Ergebnisse der Analyse festhalten (Step 3):	☐	☐	☐	☐	☐

Weitere Kommentare:

Abbildung 5.3: Zweite Seite des Fragebogen zu A-CAST.

Export von A-CAST

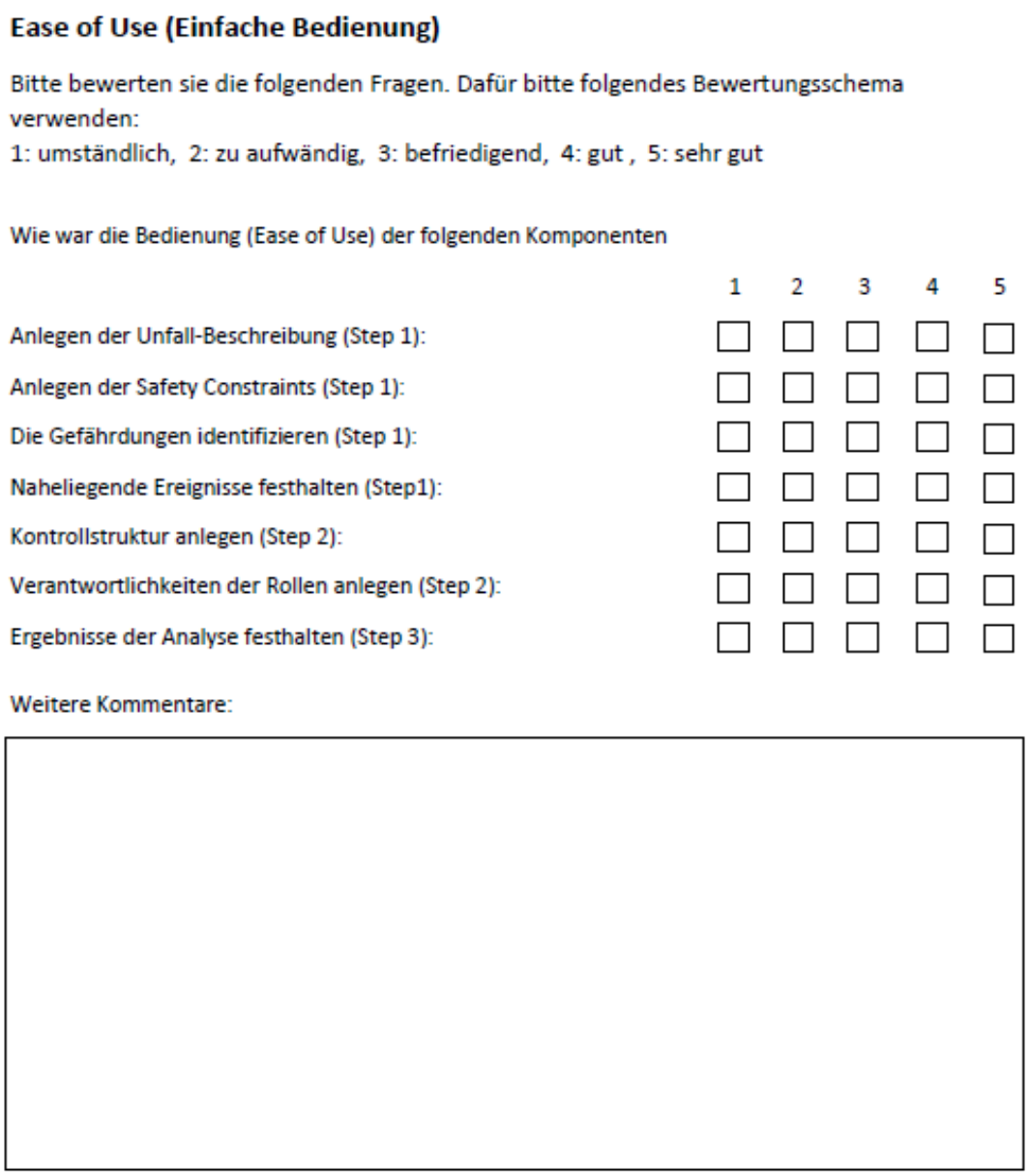
Bitte bewerten sie die folgenden Fragen. Dafür bitte folgendes Bewertungsschema verwenden:
1: umständlich, 2: zu aufwändig, 3: befriedigend, 4: gut , 5: sehr gut

Wie einfach fanden sie es die folgenden Funktionen von A-CAST durchzuführen und wie würden sie das Ergebnis beurteilen?

	1	2	3	4	5
Export als PDF:	☐	☐	☐	☐	☐
Export als PNG:	☐	☐	☐	☐	☐
Export als CSV:	☐	☐	☐	☐	☐

Weitere Kommentare:

Abbildung 5.4: Dritte Seite des Fragebogen zu A-CAST.



Ease of Use (Einfache Bedienung)

Bitte bewerten sie die folgenden Fragen. Dafür bitte folgendes Bewertungsschema verwenden:
1: umständlich, 2: zu aufwändig, 3: befriedigend, 4: gut , 5: sehr gut

Wie war die Bedienung (Ease of Use) der folgenden Komponenten

	1	2	3	4	5
Anlegen der Unfall-Beschreibung (Step 1):	☐	☐	☐	☐	☐
Anlegen der Safety Constraints (Step 1):	☐	☐	☐	☐	☐
Die Gefährdungen identifizieren (Step 1):	☐	☐	☐	☐	☐
Naheliegende Ereignisse festhalten (Step1):	☐	☐	☐	☐	☐
Kontrollstruktur anlegen (Step 2):	☐	☐	☐	☐	☐
Verantwortlichkeiten der Rollen anlegen (Step 2):	☐	☐	☐	☐	☐
Ergebnisse der Analyse festhalten (Step 3):	☐	☐	☐	☐	☐

Weitere Kommentare:

Abbildung 5.5: Vierte Seite des Fragebogen zu A-CAST.

5.2 Auswertung des Fragebogens

Die Auswertung des Fragebogens wird in den folgenden Abschnitten vorgestellt. Dabei wird jede Frage separat ausgewertet.

5.2.1 Auswertung Benutzbarkeit

Die Benutzbarkeit von A-CAST (siehe Abbildung 5.2) wurde auf folgende Weise bewertet. Für die Bewertung wurde das Schema aus Abbildung 5.2 verwendet. Der Mittelwert aus allen Fragen lag bei ungefähr 4,1. Die Benutzbarkeit von A-CAST wurde also gut bewertet. Die niedrigste vergebene Bewertung lag bei circa 3,6, die beste vergebene Bewertung bei 4,6. Es gab keine große Schwankungen in der Bewertung.

5.2.2 Auswertung der Durchführung der Schritte von A-CAST

Schritt 1: Unfall-Beschreibung

Die Unfall Beschreibung wurde von allen Teilnehmern mit 5 Punkten bewertet. In der Frage ging es um die Einfachheit der Bedienung (siehe Abbildung 5.3). Die Unfall-Beschreibung wurde am besten von allen einzelnen Schritten bewertet. 5 Punkte entsprachen dem Wert *sehr gut* im Bewertungsschema.

Schritt 1: Sicherheitsauflagen

Das Anlegen der Sicherheitsauflagen wurde im Durchschnitt mit 3,8 Punkten bewertet. Auch hier ging es um die Einfachheit der Bedienung (siehe Abbildung 5.3). Ein Kritikpunkt als Kommentar war, dass die Buttons zum Hinzufügen und Löschen nicht oben in der Leiste platziert waren sondern unten und so nicht gleich zu sehen waren. 4 Punkte im Bewertungsschema entsprachen dem Wert *gut*.

Schritt 1: Gefährdungen

Das Anlegen den Gefährdungen ist genau so aufgebaut wie das Anlegen der Sicherheitsauflagen. Deswegen gab es hier auch die gleiche Bewertung, nämlich 3,8 Punkte. Die genannten Kritikpunkte bei den Sicherheitsauflagen gelten auch hier.

Schritt 1: Naheliegende Events

Der Editor für die naheliegenden Events wurde im Durchschnitt mit 4,2 Punkten bewertet. Die Frage bezog sich wieder auf die Einfachheit der Bedienung (siehe Abbildung 5.3). Es wurden keine zusätzlichen Kommentare festgehalten. 4 Punkte entsprechen dem Wert *gut* im Bewertungsschema.

Schritt 2: Kontrollstruktur

Das Anlegen der Kontrollstruktur wurde im Durchschnitt mit 4,6 Punkten bewertet. Die Teilnehmer fanden die Bedienung einfach und leicht verständlich. 5 Punkte im Bewertungsschema entsprachen dem Wert *sehr gut*.

Schritt 2: Verantwortlichkeiten

Der Editor für die Verantwortlichkeiten wurde im Durchschnitt mit 4,2 Punkten bewertet. Dies entspricht im Bewertungsschema (siehe Abbildung 5.3) dem Wert *gut*. Es gab keine zusätzliche Kommentare in der Auswertung.

Schritt 3: Ergebnisse der Analyse

Die Bewertung des Editors, für die Ergebnisse der Analyse, entspricht der Bewertung des Editors für die Verantwortlichkeiten, da diese genau gleich aufgebaut sind.

5.2.3 Auswertung Export

Der Export in A-CAST wurde im Durchschnitt mit 5 Punkten bewertet. Dies entspricht dem Wert *sehr gut* im Bewertungsschema. Die Teilnehmer fanden den Export gut strukturiert und übersichtlich. In der Frage ging es um die Funktionalität vom Export und das Ergebnis dieses (siehe Abbildung 5.4).

5.2.4 Auswertung Bedienbarkeit

Die Bedienbarkeit in A-CAST wurde im Durchschnitt mit 4,25 Punkten bewertet. Das Bewertungsschema und die Frage sind in Abbildung 5.5 zu sehen. Die Bedienbarkeit wurde von den Teilnehmer als gut eingestuft.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde das Thema CAST und die dazu gehörenden Grundlagen (siehe Kapitel 2) vorgestellt. Dazu gehört das STAMP-Modell, ein Unfallmodell, das besonders für komplexe System geeignet ist. Darauf aufbauen wurde CAST näher erläutert und die einzelnen Schritte der Analyse vorgestellt. Ein weiterer Teil dieser Arbeit war die Entwicklung des Tools A-CAST, ein Plug-in für die XSTAMPP-Plattform. Dieses ermöglicht eine Analyse der Sicherheit eines komplexen Systemes. In XSTAMPP ist die Gefahrenanalyse (STPA) bereits enthalten und implementiert. Mit A-CAST sind nun CAST und STPA für die Analyse von Unfällen mithilfe der XSTAMPP-Plattform möglich. Es wurde die Implementierung und Entwicklung von A-CAST erläutert und gezeigt, wie A-CAST um weitere Funktionalitäten erweitert werden kann. Ebenso wurde die Benutzung von A-CAST anhand des Beispiels eines realen Unfalls gezeigt.

6.1 Fazit

Mithilfe von A-CAST lassen sich erfolgreich alle Schritte der Analyse mit CAST durchführen. In Kapitel 4 wurde dies anhand eines realen Beispiels verdeutlicht. Dabei wurde auf die Funktionsweise von A-CAST eingegangen aber auch die graphischen Oberflächen wurden gezeigt und erläutert. Weiterhin besteht die Möglichkeit die Daten aus A-CAST zu exportieren. Auf diese Weise lassen sich Informationen schnell und einfach zusammenfassen und bieten einen guten Überblick über das zu analysierende System. Da Formate wie PDF, CSV und PNG unterstützt werden, ist es möglich die Daten der Analyse einfach zu veröffentlichen und diese sind leicht zugänglich.

Zur Evaluation wurde eine kurze Test-Studie durchgeführt, in welcher fünf Softwaretechnik Studenten an der Universität Stuttgart teilnahmen. Die Teilnehmer bekamen Daten zu einem realem Unfall zur Verfügung gestellt und mussten diese mithilfe von A-CAST modellieren. Anschließend füllten sie einen Fragebogen (siehe Kapitel 5) aus. Bei der Auswertung dieses, ergaben sich durchweg positive Resultate. Sowohl die Usability als auch die Bedienbarkeit wurden als gut erachtet.

Die Test-Studie hatte fünf Teilnehmer, dies reicht durchaus um einen ersten Eindruck zu gewinnen, aber um A-CAST richtig bewerten zu können braucht es natürlich mehr Benutzer.

Deshalb wird es spannend zu sehen, wie oft und von wie vielen Benutzern A-CAST verwendet wird, nachdem es veröffentlicht wird.

A-CAST ist nun erfolgreich in XSTAMPP integriert. Das heißt mit der XSTAMPP-Plattform ist es jetzt möglich Unfälle mithilfe des CAST Ansatzes zu analysieren. Da XSTAMPP eine Open-Source Software ist, steht A-CAST jedem zu Verfügung und kann kostenlos genutzt werden.

6.2 Ausblick

Nach der Veröffentlichung von A-CAST wird sich herausstellen, ob alle nötigen Funktionalitäten vorhanden sind und es erfolgreich benutzt werden kann. Da A-CAST eine RCP-Anwendung ist und als Open-Source Software veröffentlicht wird, das heißt der Code steht jedem zur Verfügung, kann es leicht um gewünschte Funktionalitäten erweitert werden. Das Thema Erweiterung wurde ebenfalls in dieser Arbeit behandelt und erläutert. Dies sollte es jedem Entwickler möglich machen A-CAST schnell und einfach um gewünschte Funktionalitäten zu erweitern. Die RCP-Architektur unterstützt die Erweiterbarkeit noch zusätzlich durch ihre Modularität.

Auf diese Weise sollte es jedem möglich sein, den CAST Ansatz zu verwenden. Da XSTAMPP bereits einige Nutzer hat, werden diese auch A-CAST verwenden, sofern sie es benötigen sollten. So können Unfälle nochmal analysiert werden, um Schwachstellen in der Sicherheit im System aufzudecken und auf diese Weise auch zukünftige Unfälle verhindern zu können.

Literaturverzeichnis

[Abd15] A. Abdulkhaleq. XSTAMPP, 2015. URL http://www.iste.uni-stuttgart.de/se/werkzeuge/xstampp.html. (Zitiert auf Seite 12)

[AW] A. Abdulkhaleq, S. Wagner. XSTAMPP: An eXtensible STAMP Platform As Tool Support for Safety Engineering. (Zitiert auf den Seiten 24 und 25)

[AW14] A. Abdulkhaleq, S. Wagner. Open Tool Support for System-Theoretic Process Analysis. 2014. (Zitiert auf Seite 23)

[Bau06] D. Baule. Modellierung sicherheitskritischer Systeme durch Kopplung von Aufgabenmodellierung und dem Unfallanalyseverfahren STAMP. *Studienarbeit, Universität Paderborn*, 2006. (Zitiert auf Seite 15)

[Cle02] P. L. Clemens. Fault tree analysis. *JE Jacobs Severdurup*, 2002. (Zitiert auf Seite 13)

[D+12] A. Dong, et al. *Application of CAST and STPA to railroad safety in China.* Dissertation, Massachusetts Institute of Technology, 2012. (Zitiert auf Seite 41)

[Jed11] R. A. Jedick. Domino Model, 2011. URL http://goflightmedicine.com/wp-content/uploads/2014/10/domino-theory.jpg. (Zitiert auf Seite 14)

[LDDM03] N. Leveson, M. Daouk, N. Dulac, K. Marais. A systems theoretic approach to safety engineering. *Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge*, 2003. (Zitiert auf den Seiten 16, 17 und 18)

[Lev11] N. Leveson. *Engineering a safer world: Systems thinking applied to safety.* Mit Press, 2011. (Zitiert auf den Seiten 14, 15 und 18)

[LRH09] J. Lundberg, C. Rollenhagen, E. Hollnagel. What-You-Look-For-Is-What-You-Find–The consequences of underlying accident models in eight accident investigation manuals. *Safety Science*, 47(10):1297–1311, 2009. (Zitiert auf Seite 13)

[PSA15a] PSAS. CAST Tutorial, 2015. URL http://psas.scripts.mit.edu/home/wp-content/uploads/2015/03/2015-CAST-Tutorial.pdf. (Zitiert auf den Seiten 19 und 22)

[PSA15b] PSAS. STAMP Workshop, 2015. URL http://psas.scripts.mit.edu/home/stpa2015/. (Zitiert auf Seite 11)

[Sil09] V. Silva. *Practical Eclipse Rich Client Platform Projects*. Apress, 2009. (Zitiert auf Seite 27)

[Ulr88] W. Ulrich. Systems thinking, systems practice, and practical philosophy: A program of research. *Systems Practice*, 1(2):137–163, 1988. (Zitiert auf Seite 15)

[Wö15] D. Wörterbuch. TheFreeDictionary.com, 2015. URL http://de.thefreedictionary.com/Unfallbericht. (Zitiert auf Seite 13)

Alle URLs wurden zuletzt am 26. 10. 2015 geprüft.

Erklärung

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

Ort, Datum, Unterschrift