Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

STPA Sec: Entwicklung eines Eclipse-Plugins zur Sicherheitsanalyse mit STPA-Sec

Domas Mikalkinas

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Asim Abdulkhaleq, M.Sc.
Beginn am:	1. November 2016
Beendet am:	3. Mai 2017
CR-Nummer:	I.6.1, I.6.3, K.6.5

Kurzfassung

Heutzutage ist Software ein wichtiger Bestandteil von Systemen, häufig auch bei sicherheitskritischen Systemen. Ein sicherheitskritisches System kann Gefahren für Sachwerte, Menschen und Umwelt verursachen, was wiederum in einem Schaden oder Verlust jener Werte resultieren kann. Daher ist die Gefahrenanalyse einer der wichtigsten Schritte bei der Entwicklung sicherheitskritischer Systeme. Für die Analyse der Betriebssicherheit existiert mit XSTAMPP bereits ein Werkzeug, welches die Gefahrenanalyse auf Basis des STPA-Ansatzes unterstützen soll. Jedoch existiert noch keines für STPA-Sec, mit welchem, als Abwandlung des STPA-Ansatzes, die Verwundbarkeiten durch Angriffe böswilliger Angreifer eines sicherheitskritischen Systems analysiert werden. Daher wird in dieser Arbeit ein Tool entwickelt, welches die Gefahrenanalyse der Angriffssicherheit von Systemen ermöglicht, um diese Lücke zu schließen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

CAST Causal Analysis based on STAMP. 15

CERT Computer Emergency Response Team. 37

CSV Comma-separated values. 46

NIST National Institute of Standards and Technology. 35

NIST SP 800-30 National Institute of Standards and Technology Special Publication 800-30. 21

OCTAVE Operationally Critical Threat, Asset,and Vulnerability Evaluation. 21

PDE Plugin Development Environment. 32

PDF Portable Document Format. 46

PNG Portable Network Graphics. 46

RCP Eclipse Rich Client Platform. 15

STAMP Systems-Theoretic Accident Model and Processes. 15

STPA System-Theoretic Process Analysis. 15

STPA-Sec STPA for Security. 15

XSD XML Schema Definition. 32

XSTAMPP An eXtensible STAMP Platform As Tool Support for Safety Engineering. 15

1 Einleitung

1.1 Motivation

Heutige Systeme werden immer komplexer. Dies ist vor allem der kontinuierlich steigenden Rechenkraft moderner Computer geschuldet. Unterstützt wird diese Entwicklung durch immer besser werdende Entwurfswerkzeuge und die Möglichkeit dadurch raffiniertere und an die jeweilige Situation angepasste Lösungen für neue Problemstellungen zu finden. Dadurch wird es ermöglicht mehr Software in den Systemen zu verbauen, um die präzisere und einfachere Steuerung sowie Überwachung von Systemen sicherzustellen. Dies kann insbesondere in sicherheitskritischen Systemen jedoch zu Problemen führen, denn es wird schwieriger den Überblick über diese Systeme zu behalten und es entstehen mehr Schwachstellen, welche zu Gefahren für Sachwerte, Umwelt und Menschen führen. Durch die größere Vernetzung aller Systeme entsteht auch die Gefahr, dass böswillige Angreifer über die Softwareschwachstellen die Betriebssicherheit der Systeme gefährden kann. Wenn man zum Beispiel die Entwicklung von Automobilen betrachtet, wird dies verdeutlicht. Automobile bestehen schon lange nicht mehr nur aus mechanischen Bauteilen, sondern sind komplexe, aus vielen miteinander verbundenen Einzelteilen bestehende, Systeme, bei welchen das Zusammenwirken häufig durch Software gesteuert wird. Durch das Aufkommen autonom fahrender Automobile kommt die Frage nach der Angriffssicherheit auf, da diese Automobile keine in sich geschlossenen Systeme darstellen, sondern mit der Umwelt kommunizieren müssen und somit viele angreifbare Schnittstellen zur Außenwelt besitzen. Wenn nun ein böswilliger Angreifer Zugriff auf das Automobil erlangt, kann schnell eine Gefahr für das Leben von Menschen entstehen. Daher ist es wichtig, dass man möglichst alle Schwachstellen von Systemen erkennt und entsprechende Sicherheitsvorkehrungen trifft. Dabei sollen nicht nur die Schwachstellen einzelner Komponenten eines Systems erkannt werden, sondern auch Schwachstellen, die in den Schnittstellen der einzelnen Teile entstehen. Die momentan verbreiteten Methoden sind dazu nicht in der Lage, jedoch gibt es einen neuen Ansatz namens STPA for Security (STPA-Sec), der auf System-Theoretic Process Analysis (STPA) basiert und genau diese Schwäche anderer Methoden beseitigt. Daher ist das Ziel dieser Bachelorarbeit ein Plugin für eine Methode zu entwickeln, die in der Lage ist, Schwachstellen von Systemen entdecken, die von Angreifern ausgenutzt werden können, um Gefahren durch das System zu erschaffen.

1.2 Problembeschreibung und Ziel

Zur Analyse von Gefahren in sicherheitskritischen Systemen wurde Systems-Theoretic Accident Model and Processes (STAMP) von Nancy Leveson entwickelt. STAMP besteht aus zwei Teilen: Der Unfallanalyse Causal Analysis based on STAMP (CAST) und der Gefahrenanalyse STPA. Dieser Ansatz ließ sich jedoch bisher nur für die Analyse der Betriebssicherheit verwenden. Daher wurde von William Young und Nancy Leveson eine Abwandlung dieses Ansatzes vorgeschlagen, die STPA dahingehend abändert, dass man nicht mehr die Gefahren in der Betriebssicherheit eines Systems analysiert, sondern die Angriffssicherheit eines Systems. Diese Angriffssicherheitsanalyse wird daher STPA-Sec genannt. Für dieses STPA-Sec besteht noch kein Programm, welches die Analyse unterstützt. Teil dieser Bachelorarbeit ist es, solch eines zu entwickeln. Dabei wird ein Plugin für das bereits existierende Programm An eXtensible STAMP Platform As Tool Support for Safety Engineering (XSTAMPP) entwickelt, welches bereits die normale STPA-Analyse, sowie eine CAST-Analyse unterstützt. Aufgrund der Ähnlichkeit von STPA und STPA-Sec wird das bereits existierende A-STPA-Plugin als Grundlage genommen und so abgeändert, dass es die Anforderungen für die STPA-Sec-Analyse erfüllt. Dadurch entsteht das neue Plugin STPA-Sec, welches die Möglichkeit bietet, eine Gefahrenanalyse für die Angriffssicherheit sicherheitskritischer Systeme durchzuführen. Dies alles wird in der Eclipse Rich Client Platform (RCP) gemacht, da XSTAMPP diese verwendet. Dann wird das Plugin noch anhand eines ausführlichen Beispiels im Bereich des Autonomen Fahrens getestet.

1.3 Aufbau der Ausarbeitung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: In diesem Kapitel werden die Grundlagen dieser Arbeit beschrieben. Dazu gehören die verwendeten Begrifflichkeiten, der momentane Stand der Security Analyse, sowie Erklärungen zur STAMP-Methode.

Kapitel 3 – Vergleich zwischen STPA for Security und anderen Methoden: In Kapitel 3 werden aktuelle Security Analysen vorgestellt und darauf aufbauend ein Vergleich der STPA-Sec-Methode mit anderen aktuellen Methoden der Security-Analyse durchgeführt.

Kapitel 4 – STPA-Sec Plugin: Hier werden die existierenden Werkzeuge auf technischer Ebene analysiert, sowie die Entwicklung des neuen STPA-Sec-Plugins beschrieben.

Kapitel 5 – Anwendung von STPA-Sec auf autonomes Fahren: Mit Hilfe des entwickelten Plugins wird eine Analyse an einem System zum autonomen Fahren durchgeführt.

Kapitel 6 – Verwendung des STPA-Sec Plugins: Dieses Kapitel zeigt auf, wie man das entwickelte Plugin verwenden kann.

Kapitel 7 – Zusammenfassung und Ausblick Zum Schluss werden die Ergebnisse der Arbeit zusammengefasst und Anknüpfungspunkte vorgestellt.

2 Grundlagen

2.1 Begrifflichkeiten

Im Englischen gibt es zwei Begriffe um über Sicherheit in Systemen zu sprechen: Safety und Security. Safety ist der Schutz der Umwelt vor dem System, also das Verhindern von Gefahren die durch Ausfall oder Fehlfunktion durch das System entstehen. Security hingegen ist der Schutz des Systems vor der Umwelt, also die Abwehr von Angreifern gegen das System [YL13]. Da die Begriffe Safety und Security auch in der deutschsprachigen Informatik weit verbreitet sind, werden sie im Folgenden auch verwendet. Der Begriff Sicherheit wird verwendet, wenn keine Unterscheidung zwischen Safety und Security gemacht wird. Weiterhin muss der Begriff Security im Kontext der Informatik genauer definiert werden, da sich die Bedeutung von Security in den verschiedenen Bereichen der Wissenschaft deutlich unterscheidet. Grundsätzlich werden der Security folgende Haupteigenschaften zugerechnet: Verfügbarkeit, Vertraulichkeit und Integrität. Diese sind im klassischen CIA-Modell festgelegt, wobei CIA für "Confidentiality" (Vertraulichkeit), "Integrity" (Integrität) und "Availability" (Verfügbarkeit) steht. Verfügbarkeit bezeichnet das Verhindern von Systemausfällen. Vertraulichkeit zielt auf das Verhindern unautorisierten Zugriffs auf Daten eines Systems ab und Integrität bedeutet das Verhindern des unbemerkten Änderns von Daten [Eck].

2.2 Security Analyse

Security war schon immer eine wichtige Eigenschaft von Systemen. Jedoch waren diese security-kritischen Systeme auf wenige Anwendungsbereiche beschränkt. Security ist besonders in der Kommunikation wichtig, um zu verhindern, dass Nachrichten abgefangen oder kompromittiert werden können. Gerade im Militär und bei Geheimdiensten konnte eine abgefangene Nachricht über Strategien oder Taktiken über Leben und Tod von Menschen entscheiden. Im alltäglichen Leben war Security jedoch weniger von Belang, da es wenige Kommunikationsmöglichkeiten gab und der Schaden durch diese begrenzt war. Ende des 20. und Anfang des 21. Jahrhunderts begann jedoch die rasante Entwicklung in den Bereichen der Computertechnik. Immer bessere, schnellere und kleinere Hardware wurde produziert. Angetrieben von dieser Entwicklung machte auch die Entwicklung von Software große Fortschritte. In Systemen wurde großflächig Computertechnik verbaut, dadurch wurden sie komplexer, unübersichtlicher und damit auch gefährlicher [LB07]. Dies wird besonders deutlich wenn man

die Entwicklung im Bereich der Energieerzeugung betrachtet. Frühere Dampfmaschinen waren relativ simple Konstrukte und leicht überschaubar. Bei einem Unfall waren auch die Folgen meist auf die nahe Umgebung beschränkt. Wenn man jedoch heutige Atomkraftwerke, die ohne Computersteuerung kaum vorstellbar wären, betrachtet, so hätte ein Unfall wesentlich weitreichendere Folgen. Es wäre nicht nur die nahe Umgebung für eine kurze Dauer beeinträchtigt, die Folgen könnten sich auf ganze Länder und über mehrere Jahrhunderte auswirken. Solch ein Unfall kann in einem Atomkraftwerk nicht nur durch die Fehlfunktion physischer Elemente entstehen, sondern auch durch softwarebezogene Fehlnutzung und durch böswillige Angriffe auf die Softwareinfrastruktur. Es ist nicht schwer sich vorzustellen, dass Security damit zu einer wichtigen Eigenschaft bei der Entwicklung solcher Systeme wurde.

Auch wurden die Geräte benutzerfreundlicher und fanden einen Weg in den privaten Bereich. Durch die Entwicklung des Internets und die zunehmende Vernetzung aller Geräte entstanden jedoch neue Gefahren, wie Cyberkriminalität. Damit wurde auch im privaten Bereich das Bedürfnis nach Security geweckt. Und diese Entwicklung ist rasant. Bestes Beispiel sind Systeme wie autonome Fahrzeuge, die langsam auf den Markt drängen, da sie sich durch eine hohe Vernetzung mit ihrer Umwelt und der starken Verwendung von Softwaresystemen auszeichnen. Eine Fehlfunktion in solchen Fahrzeugen, versehentlich oder hervorgerufen durch einen böswilligen Angreifer, kann schlimme Folgen haben [LB07].

2.2.1 Grundlagen der Security Analyse

Security als wissenschaftlicher Ansatz in der Informatik ist noch relativ neu. Daher kommen viele Personen, die an Security-Forschung arbeiten, aus anderen Disziplinen und bringen das Wissen und die Ansätze aus ihren ursprünglichen Bereichen mit. So stammt beispielsweise vieles aus der Terminologie und den Modellen aus dem militärischen Bereich. Man spricht beispielsweise von Verlusten und erstellt und verwendet Taktiken [YL13]. Das ist nicht verwunderlich, da man sich in beiden Fällen in einem Verhältnis von Angriff durch und Abwehr gegen einen Angreifer befindet. Ein weiterer Bereich der Security sehr beeinflusst, ist die Betriebswirtschaft. Der Schutz von Informationen und die Verfügbarkeit von Dienstleistungen über das Internet sind für Unternehmen wichtiger als je zuvor. Jeder erfolgreiche Angriff auf das Unternehmen oder seine Produkte hat den Verlust von Vermögenswerten zur Folge. Unternehmen haben also ein besonderes Interesse daran, ihre Systeme angriffssicher zu machen. Jedoch haben sie auch nur begrenzte Mittel zur Verfügung und müssen Abwägungen zwischen Gefahren und den Kosten zur Sicherung vor Gefahren machen. Aus diesem Grund basieren viele Security-Analyse-Methoden auf Risikomanagement. Im Risikomanagement wird die grundlegende Unterscheidung zwischen qualitativen und quantitativen Methoden gemacht [TM12].

2.2.2 Qualitative und quantitative Analysen

Qualitative Methoden verwenden nicht-numerische Werte, um Risiken zu beschreiben. Man geht dabei davon aus, dass Risiken und ihre Auswirkungen von Natur aus eine gewisse Ungenauigkeit haben und beschreibt diese daher in subjektiveren und vageren Werten. Der Vorteil dieser Methoden ist, dass sie kostengünstiger und weniger zeitintensiv sind, da die Werte annäherungsweise angegeben werden können. Ein weiterer Vorteil ist, dass Verbesserungen schneller identifiziert werden können. Der große Nachteil ist natürlich, dass sie subjektive Resultate hervorbringen und die Resultate wiederum Spielraum für Interpretation lassen.
Quantitative Methoden verwenden numerische Werte, um Risiken zu beschreiben. Den Informationen, beispielsweise den Wahrscheinlichkeiten eines Risikos oder den Kosten eines Verlustes, werden Werte zugeordnet, welche gemessen werden können. Die Messungen werden mittels mathematischen Modellen ausgewertet. Vorteil dieser Methoden ist, dass man konkrete, objektive Werte als Ergebnis bekommt, die miteinander verglichen werden können und auf deren Grundlage man Entscheidungen treffen kann. Nachteil ist jedoch, dass solche Methoden schnell komplex werden können und es insbesondere im Bereich der Informationssicherheit schwieriger ist, konkrete und korrekte Werte zu finden, die die Wirklichkeit widerspiegeln. Außerdem können konkrete Werte zu einer zu großen Zuversicht auf die Korrektheit der Analyse führen und zu Nachlässigkeit führen [Elk06]. Auch wenn man qualitative und quantitative Methoden gemeinsam verwenden kann, ist es schwierig die verschiedenen Prinzipien und Bewertungskriterien miteinander zu vermischen, um ein aussagekräftiges Ergebnis zu bekommen [TJYN11].

2.2.3 Risikomanagement

Risikomanagement ist aktuell einer der wichtigsten Ansätze in der Security Analyse. Risikomanagement ist ein Konzept, welches bereits schon länger in vielen anderen Bereichen angewendet wird. Die Risikomodelle in den Bereichen sind daher sehr gut ausgearbeitet und umfassen viele Szenarien [Whe11]. Aber auch in der Security konnten die Prinzipien in einigen Modellen sinnvoll umgesetzt werden. Beim Risikomanagement geht es darum, Risiken zu identifizieren, einzuschätzen und Wege zu finden, wie sie abgewehrt werden können. Die wesentlichen Schritte sind in den allgemein akzeptierten Modellen wie National Institute of Standards and Technology Special Publication 800-30 (NIST SP 800-30) oder Operationally Critical Threat, Asset,and Vulnerability Evaluation (OCTAVE), die in Kapitel 3 nochmal besprochen werden, gleich. Es werden Risiken und Werte, die Ereignisse, welche Bedrohungen gegen die Werte darstellen, sowie die Verwundbarkeiten und bestehenden Kontrollen im System identifiziert. Das Identifizieren dieser Dinge findet durch Sicherheitsexperten, Analysewerkzeuge und bereits vorhandene Daten statt. Da sich die grundlegenden Bedingungen, wie Technologien oder Gesetze, in diesem Bereich schnell ändern, ist dies ein schwieriges Unterfangen und bedarf höchster Expertise. Auch sind solche Einschätzungen höchst subjektiv und häufig gibt es keine vorhandenen Daten, die eine faktische Einschätzung deutlich erschweren. Danach werden die Wahrscheinlichkeiten eingeschätzt, mit welcher die Risiken eintreten. In diesem

Schritt werden hauptsächlich vorhandene Daten über bereits in anderen Fällen realisierte Risiken herangezogen. Auch hier gibt es das Problem, dass für manche Risiken keine Daten vorhanden sind, was wiederum bedeutet, dass das Einschätzen der Wahrscheinlichkeiten nur auf dem subjektiven Empfinden basieren. Und selbst wenn Daten vorhanden sind, heißt das nicht, dass die Daten statistisch repräsentativ sind. Des Weiteren müssen die Auswirkungen, die eingetretene Risiken haben, definiert werden. Hier werden häufig die Auswirkungen auf die Verfügbarkeit, die Vertraulichkeit und die Integrität betrachtet. Um Auswirkungen sinnvoll einschätzen zu können, müssen die Zusammenhänge zwischen Ereignissen und Bedingungen bekannt sein. In den modernen komplexen Systemen sind die Zusammenhänge jedoch schwer festzustellen, insbesondere Abhängigkeiten zwischen mehreren Teilen und Auswirkungen auf verschiedene Teile des Systems sind häufig nicht bekannt. Normalerweise werden auch bereits vorhandene Sicherheitsvorkehrungen in der Feststellung der Auswirkungen berücksichtigt. Diese werden bestimmten Gefahren oder Verwundbarkeiten zugeordnet, was jedoch zur Folge hat, dass Nebenwirkungen anderer Gefahren auf diese Sicherheitsvorkehrungen nicht erkannt werden. Auch wird in dieser Analyse nicht berücksichtigt, ob diese Sicherheitsvorkehrungen auch so verwendet werden, wie es geplant war. In einigen Methoden werden noch als ein Schritt Vorschläge zum Umgang mit den Risiken erzeugt. Das Ziel ist es immer Risiken auf ein akzeptables Niveau herunterzubringen. Dabei werden die vorhergehenden Ergebnisse zu Rate gezogen, sowie die Kosten, die die Reduzierung der Risiken mit sich bringen würden. Es gibt verschiedene Wege mit Risiken umzugehen. So kann man die Wahrscheinlichkeit oder die Auswirkungen eines Risikos reduzieren. Im Bereich der Informationssicherheit wäre dies beispielsweise ein Patch, der eine bekannte Schwachstelle in einem Programm schließt. Dies ist die meistgewählte Methode. Ein weiterer Weg ist es, Risiken an andere Parteien abzugeben. Hier werden beispielsweise Versicherungen abgeschlossen. Man kann Risiken auch einfach akzeptieren. Dies wird besonders dann gewählt, wenn die Risiken nicht besonders hoch sind oder die Abschwächung des Risikos besonders kostenintensiv ist. Letztlich kann man die Risiken auch vermeiden, indem man die Verwundbarkeit aus dem System entfernt. Wenn beispielsweise eine der Seiten auf einer Webseite eine bestimmte Verwundbarkeit aufzeigt, kann man diese entfernen und die verlorene Funktion durch einen anderen Mechanismus ersetzen [Elk06]. Insgesamt kann man das Risikomanagement als sehr subjektive Methode der Security Analyse bezeichnen, da sie stark von der Erfahrung und Ausbildung der ausführenden Personen abhängt. Auch wird deutlich, dass Risikomanagement häufig einen sehr stringenten kausalen Ablauf der Risiken annimmt. Das Eintreten eines Risikos ist das Geschehen eines Ereignisses, was sich auf eine Verwundbarkeit auswirkt und durch eine Sicherheitsvorkehrung reduziert werden kann. Dass es viel komplexere Zusammenhänge geben kann, wird nicht berücksichtigt. Daher ist es sinnvoll sich Gedanken darüber zu machen, ob man nicht andere Modelle findet, die nicht die Schwächen der bisherigen Methoden haben [TJYN11].

2.3 STAMP

STAMP steht für Systems-Theoretic Accident Model and Processes und wurde von Prof. Nancy Leveson am Massachusetts Institute of Technology entwickelt. STAMP ist ein Ansatz der vom alten ereignisbasierten Ansatz abweicht und sich der Systemtheorie bedient. STAMP basiert auf Prozessen und Kontrolle. Das STAMP-Modell geht von der Betrachtungsweise aus, dass Unfälle nicht durch bestimmte Ereignisse entstehen, sondern durch mangelnde Kontrolle der vorhandenen Prozesse. Durch Kontrolle der Systemkomponenten kann sichergestellt werden, dass Prozesse sicher bleiben. Da man von Fehlerketten weg und hin zur Kontrolle geht, werden auch Fehler entdeckt, die nicht durch die Fehlfunktion einzelner Komponenten entstehen, sondern durch die Zusammenarbeit und Interaktion mehrerer Komponenten. Deswegen werden im STAMP-Modell auch Risiken entdeckt, die durch klassische ereignisbasierte Ansätze nicht entdeckt werden. Zusätzlich werden durch STAMP jedoch auch die Risiken aufgedeckt, die durch die Fehlfunktion von Komponenten entstehen [YL13]. Außerdem bedeutet dies, dass das Verhindern von Unfällen nicht mehr durch das Verhindern von Ausfällen erreicht wird, sondern durch den Entwurf und das Implementieren von Kontrollstrukturen, die die notwendigen Sicherheitsauflagen erfüllen. Dahingehend gibt es drei Prinzipien, auf welchen STAMP basiert: Sicherheitsauflagen, hierarchische Sicherheits-Kontrollstrukturen und Prozess-Modelle, die in den nächsten Kapiteln genauer erklärt werden. STAMP besteht aus zwei Komponenten, STPA und CAST. CAST steht für Causal Accident Analysis und beschreibt die Analyse von bereits geschehenen Unfällen. CAST wird angewendet, um die Gründe und Umstände von Unfällen zu analysieren, um so zukünftige Unfälle verhindern zu können. STPA hingegen steht für System-Theoretic Process Analysis. Es handelt sich hierbei um eine Gefahrenanalyse, um die Gefahren eines Systems möglichst frühzeitig aufdecken und somit Gegenmaßnahmen treffen zu können. Auf diese beiden Teile wird später nochmals genauer eingegangen [Lev11].

2.3.1 Sicherheitsauflagen

Sicherheitsauflagen sind Begrenzungen, die gelten müssen, um Sicherheit gewährleisten zu können. Sie werden frühzeitig im Entwicklungsprozess formuliert. Die Verletzung einer Sicherheitsauflage kann Unfälle oder Ausfälle zur Folge haben. Diese Auflagen können dabei verschiedene Formen annehmen: es kann sich um physikalische Auflagen handeln, beispielsweise um eine Sicherung, welche bei Gefahr den Stromkreislauf in einem Stromnetz unterbricht. Es können jedoch auch soziale oder organisatorische Auflagen sein, die das Zusammenwirken zwischen Menschen und zwischen Menschen und Maschinen regeln. Dabei kann es sich sowohl um Gesetze handeln, die von außen an ein Unternehmen herangetragen werden, als auch Richtlinien oder Befehle, die innerhalb des Unternehmens festgelegt werden. Sicherheitsauflagen haben außerdem einen Geltungsbereich. Es können Auflagen aufgestellt werden, welche für das gesamte System über die volle Länge der Systemaktivität gelten, sie können jedoch auch nur für kleine Teilbereiche eines Systems für eine begrenzte Zeit gelten. Sicherheitsauflagen gelten, sobald der kontrollierte Prozess beginnt. Eine Schwierigkeit im Entwicklungsprozess

ist es, die richtigen Sicherheitsauflagen zu finden und adäquate Kontrollen dafür zu finden. In früheren, weniger komplexen, Systemen waren Auflagen häufig bereits durch technologische Grenzen gegeben. Die Entwicklung modernerer Technologien und besserer Materialien haben diese Grenzen jedoch so weit verschoben, dass Grenzen durch die Entwickler festgelegt werden müssen und Kontrollen dafür gefunden werden müssen. So haben früher häufig passive Kontrollen ausgereicht, die die Sicherheit allein durch ihre Anwesenheit gewährleisten sollen. Durch die technologische Entwicklung werden aber aktive Kontrollen immer häufiger. Aktive Kontrollen durchlaufen eine Kette an Aktionen. Zunächst muss eine aktive Kontrolle ein gefährliches Ereignis entdecken, dann muss sie Messungen durchführen, ob die kritischen Variablen nicht überschritten werden, daraufhin müssen die Messungen interpretiert werden und letztlich eine passende Reaktion auf das Ergebnis durchgeführt werden. Dies ist ein deutlich längerer und anfälligerer Prozess als bei einer passiven Kontrolle und, im Gegensatz zur passiven Kontrolle, unterliegt er auch einer zeitlichen Einschränkung. Häufig werden aktive Kontrollen durch softwaregesteuerte Kontrollsysteme gesteuert. Dadurch, dass passive Kontrollen den Prozess automatisch in einen sicheren Zustand bringen, werden sie in manchen Bereichen, die besonders sicherheitskritisch sind, bevorzugt. Diese besonders starke Sicherheit wird jedoch auf Kosten der Flexibilität erkauft. Aktive Kontrollen können eine effektivere Reaktion finden oder auch über weitere Strecken bedient werden. Grundsätzlich werden Sicherheitsauflagen zunächst auf Systemebene festgelegt. Diese Auflagen können in weitere, kleinere, Sicherheitsauflagen unterteilt werden, die für Teile des Systems gelten. Dieser Vorgang wird wiederholt, bis das festgelegte Level an Sicherheit gewährleistet ist oder bis die Auflagen nicht weiter aufgeteilt werden können. Wenn die Sicherheitsauflagen gefunden sind, können die notwendigen Kontrollen entworfen werden [Lev11].

2.3.2 Hierarchische Kontroll-Sicherheitsstrukturen

Die hierarchische Kontroll-Sicherheitsstruktur ist in STAMP ein abstraktes Modell des Systemdesigns. In Abbildung 2.1 ist ein Beispiel solch einer Struktur zu sehen. Es ist hierbei ein Abbild der funktionalen Struktur und nicht zwangsläufig ein Abbild der physikalischen Struktur eines Systems. Es werden die technischen Elemente dargestellt, aber auch beispielsweise der Fahrer eines Autos. Das Modell besteht aus mehreren hierarchisch strukturierten Ebenen, bei denen die höheren Ebenen eine größere Verantwortung und Autorität haben und den niedriger stehenden Ebenen Auflagen setzen, die diese umsetzen müssen. Die niedrigeren Ebenen geben hingegen Rückmeldungen an die höheren Ebenen. Somit entstehen Kontroll-Rückmeldungs-Beziehungen zwischen den Ebenen, die mittels Kontrollschleifen dargestellt werden. In der Kontrollstruktur werden Dinge wie Kontrollprozesse, kontrollierte Prozesse, Sensoren, Aktuatoren und Kontrollaktionen dargestellt. In Abbildung 2.1 ist beispielsweise eine simple Kontrollstruktur abgebildet, in welcher ein Fahrer ein Fahrzeug steuert. Dem Fahrer werden Sicherheitsauflagen von höheren Ebenen auferlegt, etwa Geschwindigkeitsbegrenzungen. Diese Sicherheitsauflagen werden wiederum durch Gesetze vom Gesetzgeber bestimmt. Der Fahrer übt dann Kontrolle über das Fahrzeug aus, während er jene Sicherheitsauflagen einhält und bekommt Rückmeldung vom Fahrzeug über wichtige Variablen. Wenn die Kontrollstruktur

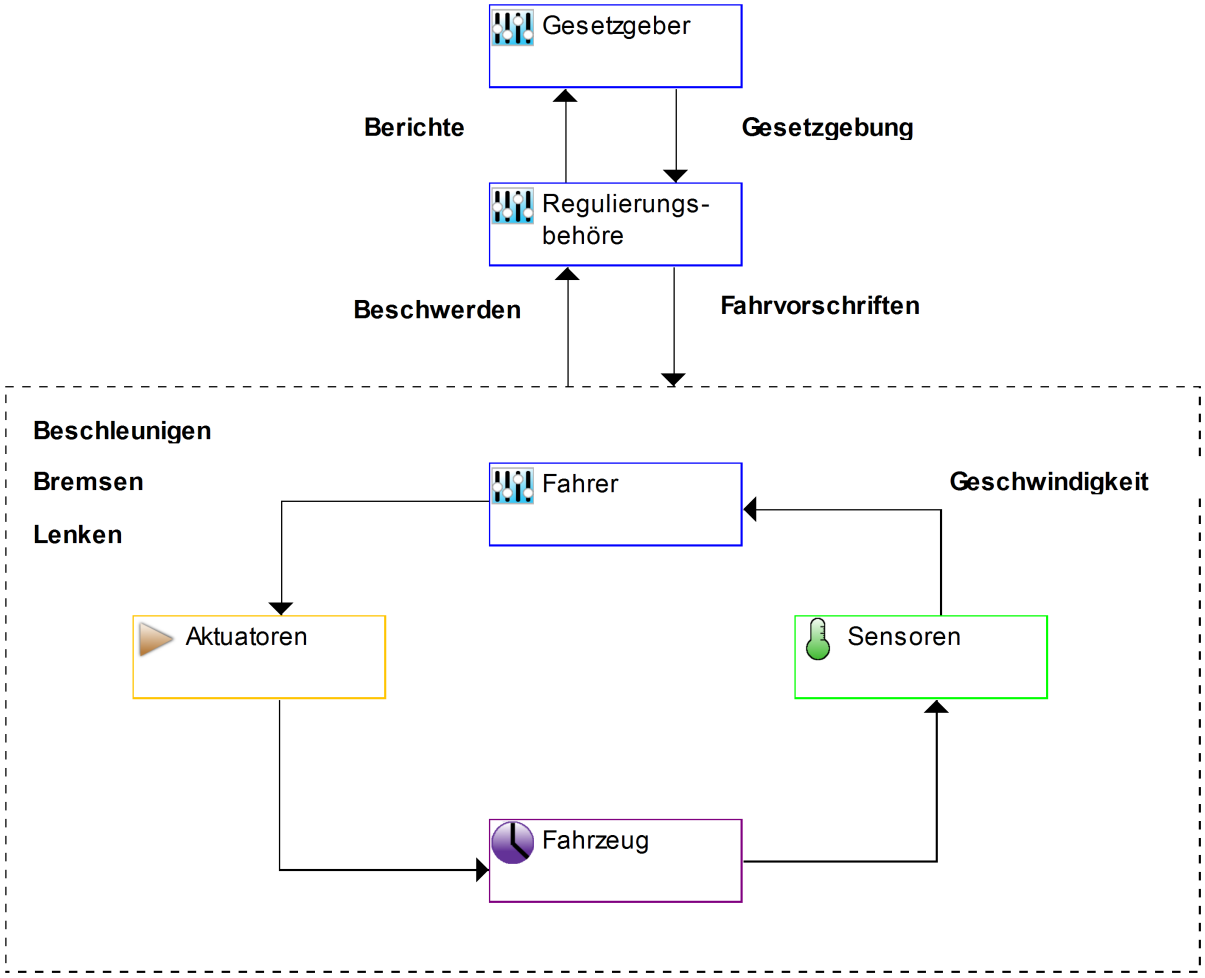


Abbildung 2.1: Kontrollstruktur eines Fahrzeugs mit Fahrer

beim Entwurf zu komplex wird, kann sie aufgeteilt werden, um die Übersicht über das System zu behalten. Für das Zusammenarbeiten der verschiedenen Teile wird ein weiteres Modell dargestellt[Lev11].

2.3.3 Prozessmodelle

In den Kontrollschleifen sind Kontrollprozesse dafür zuständig, die Sicherheitsauflagen, die die höheren Ebenen auferlegen, durchzusetzen. Die Kontrollprozesse bekommen die Rückmeldungen von den kontrollierten Prozessen, auf deren Grundlage sie Kontrollaktionen durchführen. Um die richtigen Kontrollaktionen dabei durchführen zu können, müssen die Kontrollprozesse ein Modell des kontrollierten Prozesses haben, um den Prozess effektiv kontrollieren zu können. Dies ist das Prozessmodell. Das Prozessmodell enthält Informationen über die Systemvariablen, den momentanen Zustand des Prozesses und die Möglichkeiten wie der Prozess seinen Zustand

ändern kann. Die Menge dieser Information kann dabei von wenigen Variablen für kleine Prozesse bis zu großen Modellen mit einer Vielzahl an Variablen und Zuständen für komplexe Prozesse reichen. Durch Rückmeldung des kontrollierten Prozesses wird das Prozessmodell aktualisiert und der Kontrollprozess entscheidet über etwaige Aktionen. Unfälle passieren, wenn Kontrollaktionen inkorrekt, nicht, zur falschen Zeit, zu kurz oder zu lang gesendet werden und somit die Sicherheitsauflagen der unteren Ebenen verletzen. Häufig geschehen Unfälle, weil das Prozessmodell nicht mehr mit dem kontrollierten Prozess übereinstimmt [Lev11].

2.3.4 STAMP Methoden

2.3.4.1 CAST

CAST ist eine der Komponenten von STAMP. CAST wird durchgeführt, um Unfälle zu analysieren und den Grund für das Eintreten des Unfalles herauszufinden. Es ist also eine rückblickende Betrachtung, um aus dem Geschehenen neue Erkenntnisse für zukünftige Systeme zu gewinnen. Dabei greift CAST auf die oben aufgeführten Strukturen von STAMP zurück. CAST besteht aus 9 Schritten: 1. Identifikation des Systems und der System-Gefährdungen. 2. Identifikation der Sicherheitsauflagen. 3. Konstruktion der Kontrollstruktur. 4. Beschreibung der naheliegenden Ereignisse. 5. Analyse des Verlustes auf der Ebene des physischen Systems. 6. Feststellung, wie und warum jede Ebene zur falschen Kontrollaktion der unteren Ebene beigetragen hat. 7. Begutachtung der allgemeinen Koordination und Kommunikation. 9. Erstellung von Empfehlungen. Diese Schritte müssen jedoch nicht zwingend in dieser Reihenfolge durchgeführt werden, sondern es können auch schon Schritte durchgeführt werden, bevor vorherige Schritte fertiggestellt worden sind [Lev11].

2.3.4.2 STPA

STPA ist die Methode zur Gefährdungsanalyse von STAMP. Wie auch CAST greift die Methode dabei auf die grundlegenden Strukturen von STAMP zurück. Das Ziel ist das Gleiche wie bei anderen Methoden, nämlich Informationen über Gefährdungen gewinnen. Das erworbene Wissen kann genutzt werden, um die Gefährdungen besser zu umgehen. STPA ist eine qualitative Methode, man bekommt keine Zahl, die die Gefährdung ausdrückt. STPA ist ein Top-Down-Ansatz und kann deswegen schon früh im Entwicklungsprozess angewendet werden. Es kann jedoch auch bei bereits existierenden Systemen verwendet werden, um nachträglich Gefährdungen zu finden und diese Gefährdungen abzuschwächen. STPA kann sowohl beim technischen, als auch beim organisatorischen Design angewendet werden. Die STPA-Methode besteht aus folgenden Schritten:

1. Die Systemgrundlagen für die Analyse und die Systementwicklung festlegen.
2. Unsichere Kontrollaktionen identifizieren.

3. Die unsicheren Kontrollaktionen nutzen, um Sicherheitsauflagen zu definieren.
4. Gründe für unsichere Kontrollaktionen definieren.

Im Folgenden wird die Methode nochmals genau beschrieben.

2.3.4.2.1 Die Systemgrundlage festlegen Zunächst muss definiert werden was Unfälle in einem System darstellen, sowie die Gefährdungen die damit im Zusammenhang stehen, und dazugehörige Sicherheitsauflagen. Diese Schritte werden bei fast allen Gefährdungsanalysen durchgeführt. Dann wird die vorläufige Kontrollstruktur modelliert. Die Systemgrundlagen festlegen ist nur die Vorbereitung für den Prozess und zählt nicht zum Prozess selbst. Unfälle sind laut Leveson unerwünschte und ungeplante Ereignisse, die zu einem Verlust führen. In einem System, was ein autonomes Fahrzeug steuert, wäre solch ein Unfall beispielsweise die Kollision zweier Fahrzeuge.

Unfälle können sehr eng gefasst werden, so kann ein Unfall nur das Ereignis sein, was den Verlust eines Menschenlebens verursacht, es kann aber auch so definiert werden, dass ein finanzieller Schaden auch einen Verlust darstellt. Was als Unfall gilt muss von denjenigen definiert werden, die die Analyse durchführen, da es für jedes System und je nach Einsatz etwas anderes ist. Wenn die Verluste identifiziert worden sind, müssen die dazugehörigen Gefährdungen untersucht werden. In STPA sind Gefährdungen Zustände oder Umstände, die niemals eintreten dürfen und die, verbunden mit einem Worst-Case-Szenario an Umwelteinflüssen, zu einem Verlust führen. Im Beispiel des autonomen Fahrzeugs könnte die zur Kollision zweier Fahrzeuge gehörende Gefährdung sein, dass ein unzureichender Abstand zwischen zwei Fahrzeugen gelassen wurde. Von diesen Gefährdungen können Sicherheitsauflagen abgeleitet werden. Diese Auflagen dienen den Ingenieuren als Richtlinien. Sicherheitsauflagen in diesem Schritt sind allgemeine Sicherheitsauflagen, die für das gesamte System gelten. Im Beispiel des unzureichenden Abstands wäre eine Sicherheitsauflage, dass Fahrzeuge niemals den minimalen Abstand unterschreiten dürfen.

Als nächstes wird die Kontrollstruktur erstellt. Die Kontrollstruktur ist ebenfalls kein Teil des STPA-Prozesses, sondern ein Teil der Dokumentation und somit ein Hilfsmittel beim Prozess. Da die Kontrollstruktur eine Abbildung der Funktion eines Systems ist, ist sie meist weniger komplex als ein Modell der technischen Umsetzung eines Systems. Das macht es einfacher ein System zu verstehen. Sinnvoll ist es mit einem simplen Modell zu starten und dann das Modell zu verfeinern. Wenn die Kontrollstruktur ausreichen modelliert wurde, muss noch ein Prozessmodell für jeden Kontrollprozess definieren werden [Lev11].

2.3.4.2.2 Unsichere Kontrollaktionen definieren Ausgehend von den Kontrollaktionen die in der Kontrollstruktur modelliert wurden, werden nun die unsicheren Kontrollaktionen definiert. Es gibt vier Möglichkeiten von unsicheren Kontrollaktionen:

- Eine Kontrollaktion nicht gegeben

Kontrollaktion	Nicht gegeben führt zur Gefährdung	Gegeben führt zur Gefährdung	Zu früh/zu spät führt zur Gefährdung	Zu früh gestoppt/zu lange angewandt führt zur Gefährdung
Bremsen	Der Fahrer bremst nicht, obwohl sich das Fahrzeug einem anderen Fahrzeug nähert, welches langsamer ist	Der Fahrer bremst, obwohl sich kein Hindernis vor dem Fahrzeug befindet und hinter dem Fahrzeug andere Fahrzeuge sind	Der Fahrer bremst erst, als das Fahrzeug sich bereits zu nah an einem vorausfahrenden Fahrzeug befindet	Der Fahrer bremst zu kurz, sodass sich das Fahrzeug immer noch einem langsamer fahrenden Fahrzeug voraus nähert

Tabelle 2.1: Tabelle der unsicheren Kontrollaktionen

- Gegeben aber Parameter verfälscht
- Gegeben, aber zu kurz/lang
- Erwartet, aber zu früh/zu spät gegeben

Eine sinnvolle Art dies darzustellen ist in einer Tabelle wie in Tabelle 2.1. Hierbei werden in den Kästen rechts neben der Kontrollaktion die Bedingungen angegeben, unter denen eine Kontrollaktion unsicher werden kann. Es kann auch sein, dass eine Kontrollaktion unter keinen Umständen unsicher ist. Die Einträge können wiederum in Sicherheitsauflagen umgesetzt werden. Diese sind jedoch feiner als die Sicherheitsauflagen, die schon in den Systemgrundlagen definiert wurden. Sie gelten nur für einen Teil eines Systems oder sogar nur für eine einzelne Komponente. Hier erkennt man auch den Top-Down-Ansatz. Während man zunächst systemweite Auflagen definiert, werden auf den niedrigeren Ebenen des Systems nach und nach kleinere Auflagen gesetzt. Außerdem zeigt es auch, dass die Methode noch in sehr früheren Phasen eingesetzt werden kann, um das Sicherheitsdesign mit zu beeinflussen und nicht nur um ein bereits existierendes Konzept auf Gefährdungen zu überprüfen.

2.3.4.2.3 Gründe für unsichere Kontrollaktionen definieren Die Gründe für unsichere Kontrollaktionen können vielseitig sein. Neben den 4 bereits erwähnten unsicheren Kontrollaktionen werden hier auch Kontrollaktionen die gegeben wurden, aber nicht ausgeführt wurden, berücksichtigt. Dieser Schritt erfordert viel Erfahrung, um mögliche Szenarien zu erkennen. Ein Hilfsmittel ist die Abbildung 2.2, welches mögliche Szenarios beschreibt. Dies ist jedoch keine abschließende List an möglichen Szenarios, sondern bietet nur eine anfängliche Hilfestellung. Aus den gefundenen Szenarios müssen nun wiederum Sicherheitsauflagen definiert werden.

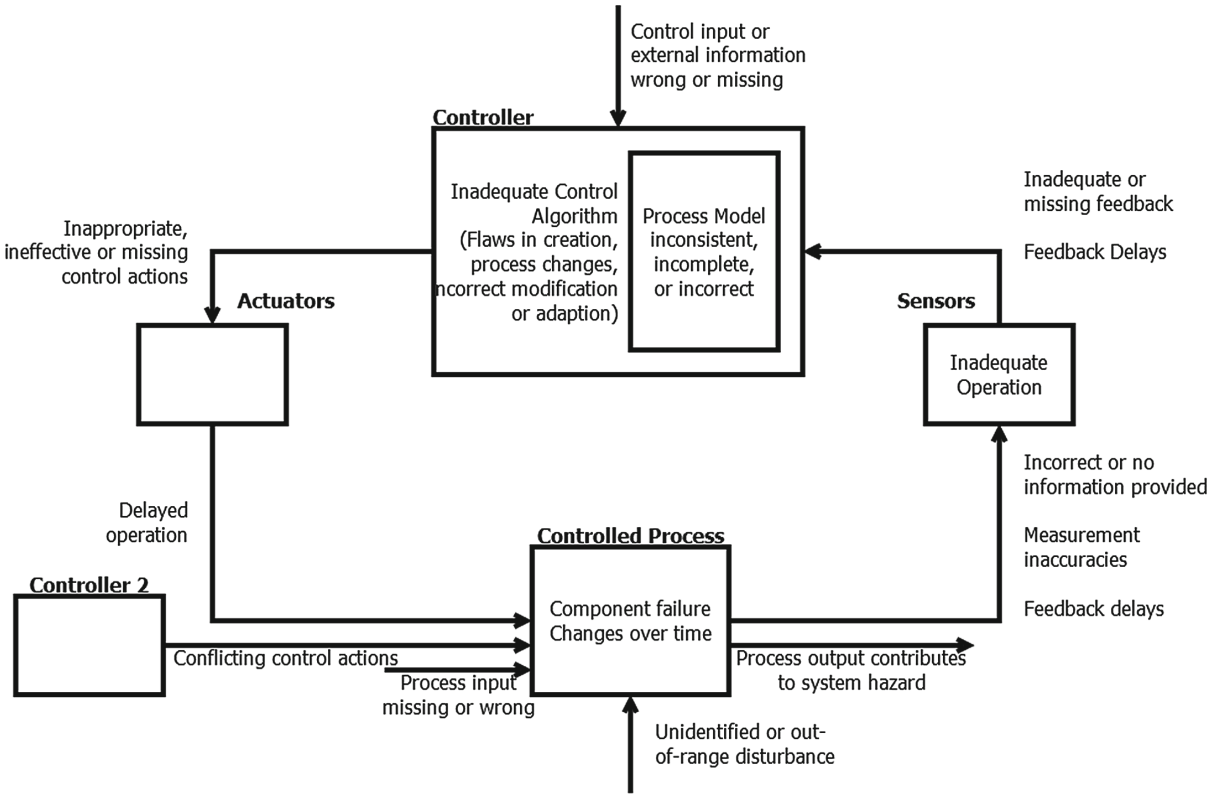


Abbildung 2.2: Szenarien für unsichere Kontrollaktionen [LDDM03]

Als Ergebnis bekommt man von STPA eine Liste von möglichen Szenarios, die zu beachten sind, sowie eine Liste von Sicherheitsauflagen, die gelten müssen. Aus diesen Ergebnissen können Ingenieure ein System bauen. Die genaue Umsetzung bleibt dabei den Ingenieuren selbst überlassen, da STPA keine Vorschläge bietet, was für Gegenmaßnahmen getroffen werden müssen, um ein System sicherer zu machen [Lev11].

2.3.4.3 STPA-Sec: STPA for Security

STPA-Sec ist der STPA-Ansatz angepasst auf die Security-Betrachtung. STPA-Sec unterscheidet sich in einigen Punkten deutlich von bisherigen security-basierten Ansätzen. So richtet STPA-Sec den Schwerpunkt der Analyse nicht auf das Verteidigen gegen Attacken, sondern auf das Schließen von soziotechnischen Schwachstellen. Der Fokus liegt also nicht auf den Gefahren die von Gegenspielern ausgehen, da man diese sowieso nicht kontrollieren kann. Vielmehr geht es um die Kontrolle der Systemteile, die im eigenen Einflussbereich liegen. STPA-Sec identifiziert die benötigten Sicherheitsauflagen für unsichere Systemzustände, die das System in einen Zustand der Angreifbarkeit bringen, wenn sie Störungen ausgesetzt werden. Dabei kann es sich um bewusste oder unbewusste Störungen handeln. Der grundlegende Ablauf von STPA-Sec ist der gleiche wie bei STPA. Jedoch unterscheiden sich die beiden Ansätze in

den einzelnen Schritten. So werden bei STPA-Sec nicht Unfälle definiert. Unfälle sind unbeabsichtigte Ereignisse. Jedoch werden in der Security insbesondere auch absichtliche Angriffe böswilliger Gegenspieler betrachtet. Daher ist es notwendig, vom Unfallbegriff abzulassen und die Betrachtung auf Verluste zu lenken, denn in der Security ist eines der fundamentalen Ziele, Verluste durch Angriffe zu vermeiden. Ein weiterer Unterschied wird bei der Analyse von Gefährdungen gemacht. Gefährdungen sind ein Begriff aus der Safety. In der Security geht es dahingehend darum, Verwundbarkeiten zu identifizieren und diese zu kontrollieren. Daher wird in STPA-Sec Ansatz der Fokus auf Verwundbarkeiten gelegt. Auch die unsicheren Kontrollaktionen unterscheiden sich in den beiden Ansätzen. Während "unsicher" bei STPA im Sinne der Safety gemeint ist, müssen unsichere Kontrollaktionen bei STPA-Sec im Sinne der Security identifiziert werden. Dies wirkt sich auch auf die Gründe für die unsicheren Kontrollaktionen aus. Während die Gründe in STPA mit unbeabsichtigten Szenarien modelliert werden, müssen in STPA-Sec absichtliche Szenarien beachtet werden [YL13]. Verbesserungen von STPA-Sec wurden von Schmittner et al. [SMP16] vorgeschlagen. Es wurde bemängelt, dass die Terminologie von STPA-Sec sich von der Terminologie des restlichen Security-Engineerings unterscheidet. Beispielsweise ist eine Verwundbarkeit im Security-Engineering eine Schwäche, die von einer Gefährdung missbraucht werden kann. Bei STPA-Sec hingegen ist eine Verwundbarkeit ein gefährlicher Systemzustand. Auch wurde die Liste an Szenarien, die zu Problemen führen können, um absichtliche Szenarien erweitert. Dies soll eine Hilfestellung sein, um die Analyse starten zu können. In der Abbildung 2.3 sind die neuen Szenarien in roter Farbe geschrieben. Wie schon in der ursprünglichen Liste ist diese nicht vollständig, der Nutzer muss ausgehend von den beschriebenen Szenarien aus neue Szenarien finden und diese in seiner Analyse beachten.

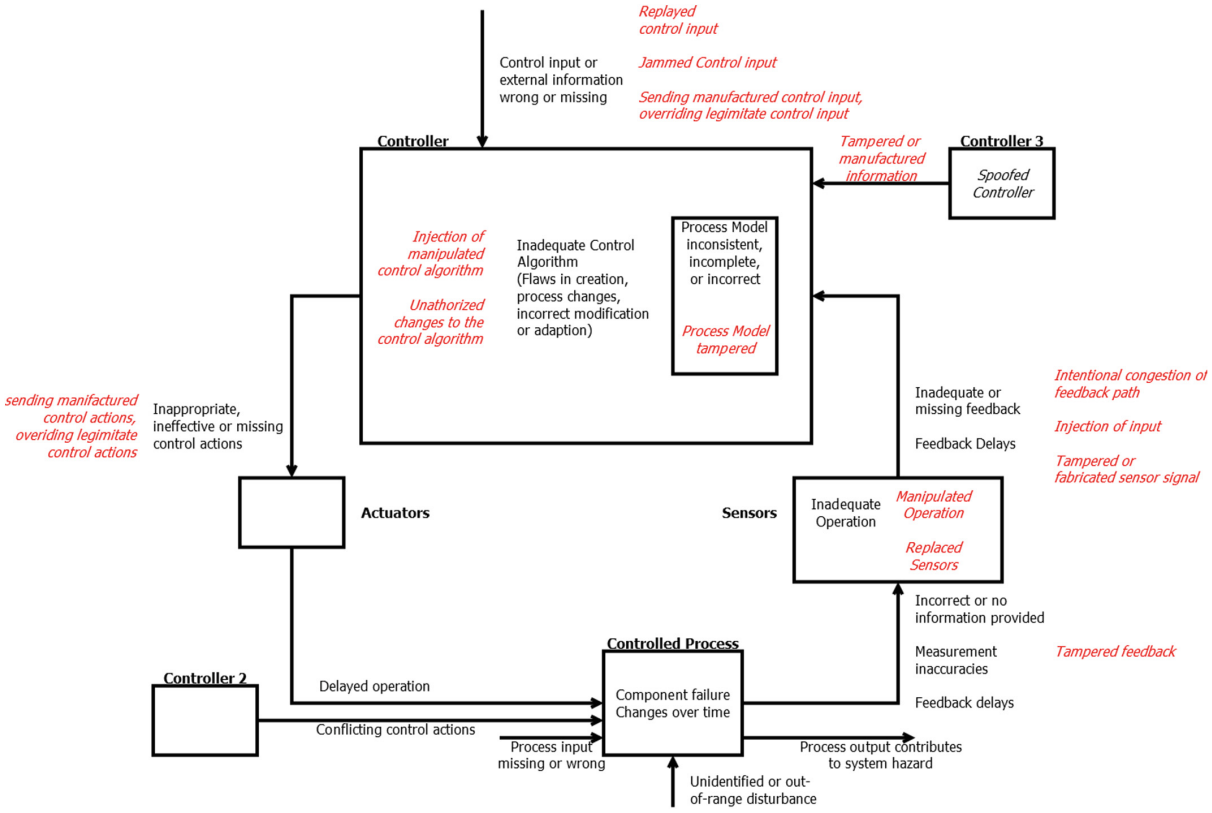


Abbildung 2.3: Szenarien für unsichere Kontrollaktionen bei STPA-Sec [SMP16]

2.4 Existierende Tools

2.4.1 XSTAMPP

XSTAMPP steht für „An eXtensible STAMP Platform As Tool Support for Safety Engineering“ und ist ein Programm, welches an der Universität Stuttgart durch die Promotion von Asim Abdulkhaleq entwickelt wurde. Es bildet eine Plattform für das STAMP-Modell und bietet mehrere Werkzeuge für die Anwendung der Methoden (STPA und CAST) dieses Modells. Ursprünglich wurde das Werkzeug A-STPA im Jahr 2014 im Rahmen eines Studentenprojekts entwickelt, welches vorerst nur die Funktionen der STPA-Methode unterstützte [AW14]. Dieses wurde auch erfolgreich in vielen Industriezweigen verwendet. Jedoch erkannte man, dass A-STPA auch Schwachstellen hatte. A-STPA implementierte nur die grundlegenden Schritte der STPA-Methode und konnte auch nicht erweitert werden, um Verbesserungen am STPA-Ablauf vorzunehmen oder gar neue Methoden von STAMP zu implementieren. So kam, mit Blick auf die Schwachstellen von A-STPA, der Entschluss auf, XSTAMPP auf der Grundlage von A-STPA zu entwickeln und die Funktionen von A-STPA als Plug-in in die neue Plattform aufzunehmen [AW15].

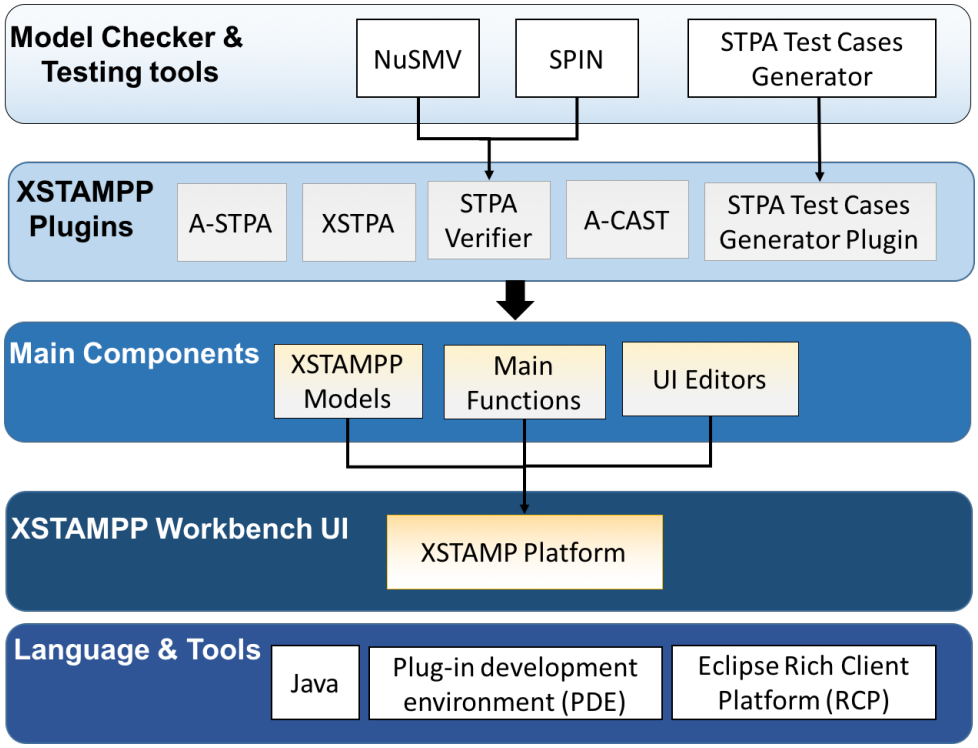


Abbildung 2.4: Architektur von XSTAMPP und seinen Plugins [Abd17]

Um die Erweiterbarkeit sicherzustellen wurde XSTAMPP mithilfe der RCP und der Plugin Development Environment (PDE) entwickelt. Diese bieten nützliche Funktionen, um eine Plattform zu bauen, welche mithilfe von Plugins erweitert werden kann. So stellt die RCP eine Reihe von Kerndateien dar, die minimal nötig sind, um eine Anwendung zu bauen, sowie bereits vorhandene und geprüfte Komponenten zur Verfügung, die nach Bedarf modular eingebunden werden können und die die Entwicklung der Anwendung beschleunigen. Die Plugins sind selbstständige Jar-Dateien, welche alle nötigen Ressourcen beinhalten, um ausgeführt zu werden. Die Plugins greifen im Fall von XSTAMPP teilweise auf Funktionalitäten des STAMP-Modells zu, welche in der Plattform selbst bereits implementiert sind.

In der Abbildung 2.5 ist die Benutzeroberfläche von XSTAMPP zu sehen. Hierbei ist es die Standardoberfläche, die von der RCP geboten wird. Die Oberfläche kann um weitere Ansichten erweitert und an die jeweiligen Bedürfnisse angepasst werden. Links werden die Projekte der Plugins angezeigt, welche auf XSTAMPP eingebunden sind. In XSTAMPP erstellte Projekte werden in XML-Dateien abgespeichert und können später mittels XML Schema Definition (XSD) eingelesen und in der Anwendung angezeigt werden. Außerdem bietet die XSTAMPP-Plattform Interfaces, auf die andere Plugins aufbauen müssen.

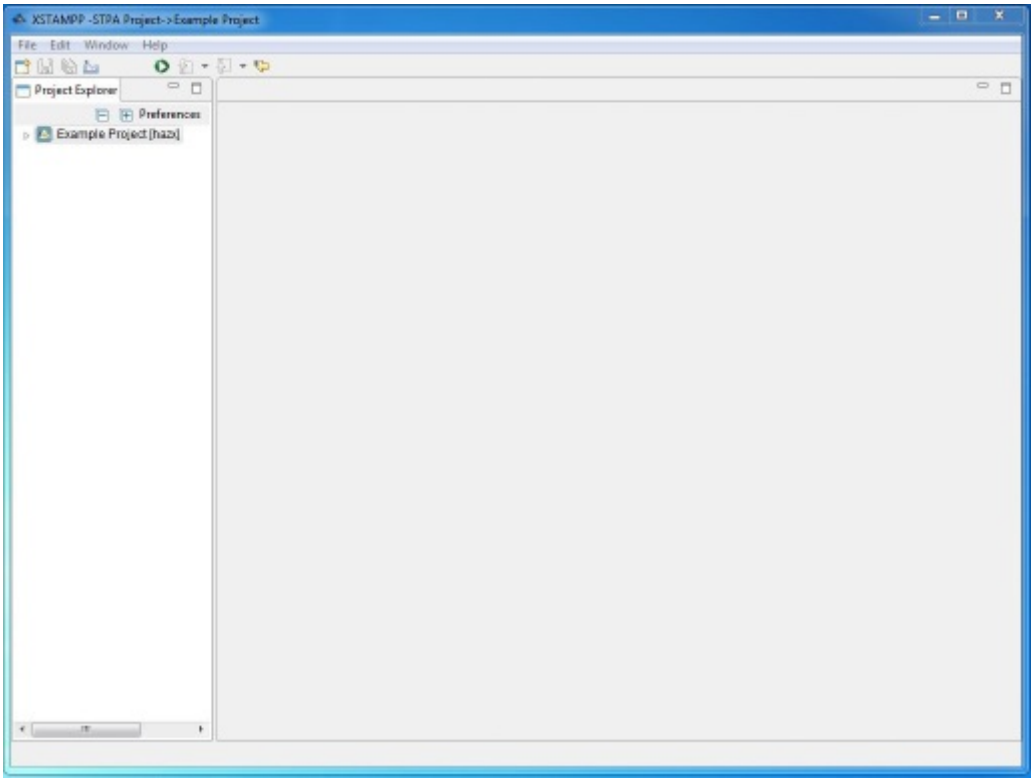


Abbildung 2.5: Benutzeroberfläche von XSTAMPP

2.4.2 A-STPA

Das A-STPA-Plugin ist eines der bereits vorhandenen Plugins für XSTAMPP und implementiert die STPA-Methode. Es ist gleichnamig mit der ursprünglichen Anwendung aus welcher XSTAMPP entstanden ist und beinhaltet die Funktionen aus der Anwendung, die nicht in der Plattform selbst implementiert wurden. Der Ablauf von A-STPA ist in Abbildung 2.6 zu sehen. Auch wurde das A-STPA-Plugin mit einem weiteren Plugin namens XSTPA erweitert. Dieses Plugin implementiert einen erweiterten Ansatz von STPA, welches es ermöglicht automatisch Kontexttabellen mit den verschiedenen Variablen aus dem Prozessmodell zu erstellen. Aus diesen Kontexttabellen lassen sich verfeinerte Kontrollaktionen und daraus wiederum verfeinerte Sicherheitsauflagen ableiten, aus denen sich Formeln in linearer temporaler Logik erstellen lassen. Diese Schritte sind im A-STPA-Ablauf unter der Kategorie „Causal Analysis“ unter dem Schritt „Control Structure With Process Model“ in Abbildung 2.6 angegeben.

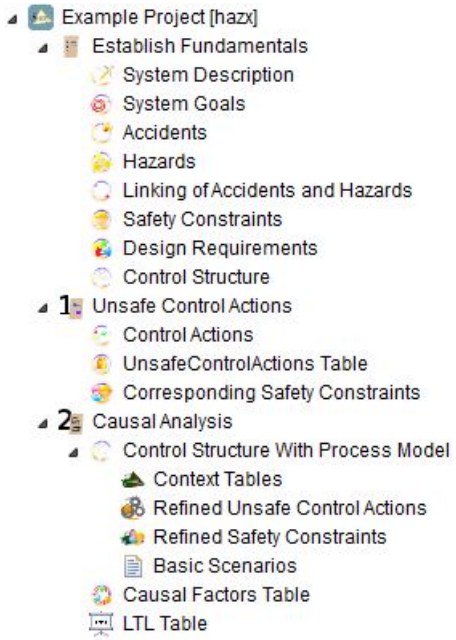


Abbildung 2.6: Ablauf von A-STPA

3 Vergleich zwischen STPA for Security und anderen Methoden

In diesem Kapitel wird der Vergleich mit bekannten und häufig verwendeten Methoden der Security-Analyse durchgeführt. Zunächst werden die Methoden einzeln vorgestellt, dann werden diese mit STPA-Sec verglichen und letztlich wird das Ergebnis des Vergleichs besprochen.

3.1 NIST 800-30

NIST SP 800-30 ist eine Methode zur Security-Analyse speziell für IT-Systeme, die vom National Institute of Standards and Technology (NIST), welches vom Handelsministerium der Vereinigten Staaten finanziert wird, entwickelt wurde. Es wurde für staatliche Sicherheitssysteme entwickelt, um staatlichen Richtlinien zu entsprechen und wird dementsprechend insbesondere von US-Bundesbehörden und Organisationen, die staatlichen Regulierungen unterworfen sind, verwendet, jedoch verwenden auch privaten Unternehmen diese Methode [TM12]. Es ist einer der ersten Ansätze, die auf dem Risikomanagement, das schon im Abschnitt 2.2.3 besprochen wurde, basieren und hat nachfolgende Methoden beeinflusst und wird auch weiterhin verbessert und angepasst. NIST SP 800-30 ist ein qualitativer Ansatz und betrachtet das System hauptsächlich von einer technischen Perspektive. Es gibt neun Schritte die durchzuführen sind:
1. Systemcharakterisierung: Im ersten Schritt werden alle (technischen) Komponenten des Systems, die relevant für die Analyse sind, wie etwa Hardware, Software oder Daten aufgelistet. NIST 800-30 gibt Empfehlungen vor, wie man diese Informationen gewinnen kann. Dazu gehören Techniken wie Fragebögen und Befragungen, aber auch automatische Scan-Werkzeuge.
2. Identifizierung von Gefahren: Im zweiten Schritten sollen alle Gefahren für das System identifiziert werden. Eine Gefahr ist alles, was eine Schwachstelle im System ausnutzen kann, beispielsweise ein Hacker. Auch hier bietet NIST 800-30 einen Katalog an möglichen Gefahrenquellen. Diese Gefahrenquellen sind aufgeteilt in die Kategorien natürliche Gefahren, menschliche Gefahren und Gefahren durch die Umwelt. Weiterhin verweist der Ansatz auf verschiedene Informationsquellen zu Gefahren, wie etwa Geheimdienste oder Security-Webseiten.
3. Identifizierung von Verwundbarkeiten: Den Gefahren aus dem zweiten Schritt werden im dritten Schritt Verwundbarkeiten, die von den Gefahren ausgenutzt werden können, zugeordnet. So wäre eine Verwundbarkeit die ein Hacker ausnutzen könnte, das Ausbleiben von

Ausbesserungen im System. Auch in diesem Schritt bietet der Ansatz einen Katalog an möglichen Informationsquellen, wie vorherige Risikoeinschätzungen oder Checklisten.
4. Analyse der Kontrollen: Im nächsten Schritt werden vorhandene und geplante Kontrollen aufgelistet. Kontrollen sollen verhindern, dass Verwundbarkeiten ausgenutzt werden. Verweise zu anderen Dokumenten mit möglichen Kontrollen bietet NIST SP 800-30 auch hier.
5. Festlegung der Wahrscheinlichkeit: Nun werden die Wahrscheinlichkeiten festgelegt, dass eine Verwundbarkeit ausgenutzt wird, abhängig von den Gefahren, Umwelteinflüssen und den Stärken der Kontrollen. Zu jeder Paarung von Verwundbarkeiten und Gefahren müssen Wahrscheinlichkeiten festgelegt werden. Diese Wahrscheinlichkeiten werden entweder durch qualitative Werte, wie hoch, mittel und niedrig, oder, abweichend vom qualitativen Ansatz, durch semi-quantitative Werte, also geschätzte numerische Werte angegeben.
6. Analyse der Auswirkungen: Das Ziel dieses Schrittes ist es, einzuschätzen wie sich die Ausnutzung einer Verwundbarkeit durch eine Gefahr auswirkt. Dabei wird hauptsächlich die Auswirkung auf Vermögenswerte, Individuen, Vorgänge oder Organisationen betrachtet. Auch hier wird die Auswirkung wieder qualitativ oder semi-quantitativ angegeben.
7. Einschätzung der Risiken: Zur Einschätzung der Risiken werden die Auswirkungen und die Wahrscheinlichkeiten verwendet. Auch in diesem Punkt gibt es eine quantitative oder semi-quantitative Werte, die die Risiken einschätzen, sowie eine Beschreibung, was die jeweiligen Stufen zu bedeuten haben.
8. Kontrollempfehlungen: In Schritt 8 werden Empfehlungen für Kontrollen erstellt, um mit den Risiken umzugehen. Hilfestellungen liefert der Ansatz jedoch nicht, dies bleibt den durchführenden Personen überlassen.
9. Ergebnisdokumentation: Im letzten Schritt wird ein Bericht erzeugt. Dieser hilft den Entscheidungsträgern die richtigen Entscheidungen zu treffen. Es gibt keine Vorgabe, was alles in den Bericht muss, jedoch wird empfohlen die Gefahren, Verwundbarkeiten, die Risikoeinschätzung und die empfohlenen Kontrollen hinzuzufügen [SGF02].

3.1.1 Vorteile von NIST 800-30

NIST 800-30 ist einer der am meisten ausgearbeiteten Methoden. Er liefert sehr viele Hilfestellung zur Durchführung der Analyse in Form von Informationsquellen und Beschreibungen zu den jeweiligen Schritten. Im Gegensatz zu vielen anderen Ansätzen werden auch Wege zum Umgang mit Risiken erstellt. Er ist auch flexibel und lässt sich gut an die jeweiligen Bedürfnisse anpassen. Außerdem ist er weit verbreitet und wird somit weiter gepflegt.

3.1.2 Nachteile von NIST 800-30

NIST 800-30 ist ein sehr subjektiver Ansatz und hängt somit stark von den durchführenden Personen ab. Dies führt auch dazu, dass die Ergebnisse nicht so klar definiert sind und wiederum interpretiert werden können. Er liefert auch nicht viele Beispiele zur konkreten Umsetzung der Kontrollen. Außerdem wurde er hauptsächlich für staatliche Organisationen entwickelt,

darum sind viele Beispiele für Gefahren für staatliche und militärische Szenarios ausgelegt [TM12].

3.2 OCTAVE Allegro

Eine weitere Methode zur Security-Analyse ist OCTAVE Allegro. OCTAVE Allegro ist ein qualitativer Ansatz und basiert ebenfalls auf dem Risikomanagement. OCTAVE Allegro ist eine Abwandlung des OCTAVE-Ansatzes, welcher speziell für die Information-Security vom Carnegie Mellon University's Software Engineering Institute in Zusammenarbeit mit Computer Emergency Response Team (CERT) entwickelt wurde. OCTAVE-Allegro nutzt, wie auch der originale OCTAVE-Ansatz, eine Reihe von Workshops, um die Analyse durchzuführen. Jedoch wurde OCTAVE Allegro so entwickelt, dass es sich auch von Einzelpersonen durchführen lassen kann. OCTAVE Allegro bietet eine Vielzahl von Hilfestellungen, wie Arbeitsblätter und Fragebögen, um den Prozess zu unterstützen. Der Ansatz wird in acht Schritten durchgeführt:
1. Kriterien für Risikoabschätzung festlegen: Im ersten Schritt wird festgelegt, wie die Risiken gemessen werden sollen. Die Risiken werden für verschiedene Bereiche eines Unternehmens identifiziert, also beispielsweise Risiken für den Ruf des Unternehmens oder für die Finanzen. Außerdem werden die verschiedenen Bereiche nach ihrer Wichtigkeit für die Ziele des Unternehmens gewertet. Für diesen Schritt bietet OCTAVE Allegro Vorlagen für Arbeitsblätter.
2. Profil der Informationsgüter entwickeln: Im zweiten Schritt werden die Profile für Informationsgüter entwickelt. Diese Profile enthalten beispielsweise einzigartige Eigenschaften, Qualitäten, Wichtigkeiten und Sicherheitsanforderungen der Güter und sollen die Güter klar voneinander abgrenzen.
3. Container der Informationsgüter identifizieren: Container beschreiben die Orte an denen Informationsgüter gespeichert, transportiert und verarbeitet werden. Das kann Hardware sein, aber auch ein Datencenter oder auch konkrete Menschen. Außerdem können die Orte innerhalb der Organisation liegen oder auch außerhalb, etwa wenn externe Dienstleister die Güter verwalten.
4. Betroffene Bereiche identifizieren: Im nächsten Schritt müssen mögliche Umstände oder Situationen ausgearbeitet werden, die eine Gefahr für die Informationsgüter darstellen. Ein Beispiel wäre etwa die schlechte Verschlüsselung von Daten auf einem über das Internet zugreifbaren Server.
5. Gefahrenszenarios identifizieren: Ausgehend von den betroffenen Bereichen können Gefahrenszenarios gefunden werden. Die Gefahrenszenarios bilden die Erweiterung der betroffenen Bereiche um eine tatsächliche Gefährdung der Güter. Im vorherigen Beispiel wäre ein solches Szenario: die schlechte Verschlüsselung von Daten auf einem über das Internet zugreifbaren Server könnte zu dem unautorisierten Zugriff auf die Daten durch einen Hacker führen. Die Szenarios können in Baumdiagrammen dargestellt werden. Für diesen Schritt hält OCTAVE Allegro als Hilfestellung Fragebögen bereit.
6. Risiken identifizieren: In diesem Schritt werden die Konsequenzen, also die Auswirkungen auf die Organisation, identifiziert für den Fall dass eine Gefahr realisiert wird. Gefahren können

auf mehrere Bereiche eines Unternehmens eine Auswirkung haben.
7. Risiken analysieren: Nachdem die Risiken identifiziert wurden, werden sie analysiert. Dazu wird dem Umfang der Auswirkungen auf ein Unternehmen durch die Realisierung einer Gefahr ein numerischer Wert zugewiesen. Anschließend wird anhand dieses Wertes und der ebenfalls durch einen numerischen Wert ausgedrückten Wichtigkeit des betroffenen Bereiches ein Risikowert berechnet. Dies dient lediglich dazu eine Priorisierung der Risiken zu erzeugen und nicht, um den Risiken einen tatsächlichen Risikowert zuzuweisen. Interessant ist, dass die Wahrscheinlichkeit eines Risikos nicht zwingend als Bewertungsgrundlage vorgeschrieben wird, sondern der Organisation die Entscheidung über ihre Verwendung überlässt.
8: Milderungsansatz wählen: Im letzten Schritt wird beschlossen, welche Risiken behandelt werden müssen und es wird eine Strategie für den Umgang mit den Risiken erstellt. Dabei werden die Risiken nach ihrer Priorisierung abgehandelt. Die Strategien werden anhand des Wertes der Güter, ihren Sicherheitsanforderungen, den Containern und der Umgebung der Organisation ausgewählt [CSYW07].

3.2.1 Vorteile von OCTAVE Allegro

OCTAVE Allegro bietet sehr viel Hilfestellung in Form von Arbeitsblättern, Fragebögen und Beschreibungen. Dadurch benötigt man nicht so viel Vorbereitungszeit. Die einzelnen Schritte von OCTAVE Allegro sind sehr klar definiert und sind einfach zu verstehen. Auch werden Strategien für den Umgang mit Risiken erstellt.

3.2.2 Vorteile von OCTAVE Allegro

Es gibt keinen Fragenkatalog den man abarbeiten kann, die Gefahren werden anhand von Fragebögen identifiziert, was sehr subjektiv ist und somit von den durchführenden Personen abhängt. Zur Identifizierung von möglichen Szenarios benötigen die Personen auch eine sehr gute Übersicht über das System [TM12].

3.3 Attack Trees

Bei Attack Trees handelt es sich um eine formale Methode, die die Security eines Systems beschreiben. Attack Trees wurden das erste Mal von Bruce Schneier 1999 beschrieben. Bei Attack Trees handelt es sich um Baumdiagramme, sie enthalten eine Wurzel, Zweige, Knoten und Blätter. Ein Attack Tree beschreibt dabei einen Angriff auf ein System. Die Wurzel stellt das Ziel des Angriffs dar, und die Knoten und Blätter sind die verschiedenen Wege, um das Ziel zu erreichen. Attack Trees enthalten UND und ODER Knoten. Ein UND Knoten zeigt, dass alle Kinderknoten erfüllt sein müssen, um diesen Weg zu erfüllen, ein ODER Knoten bedeutet, dass mindestens einer der Kinderknoten erfüllt sein muss. Den Knoten können auch

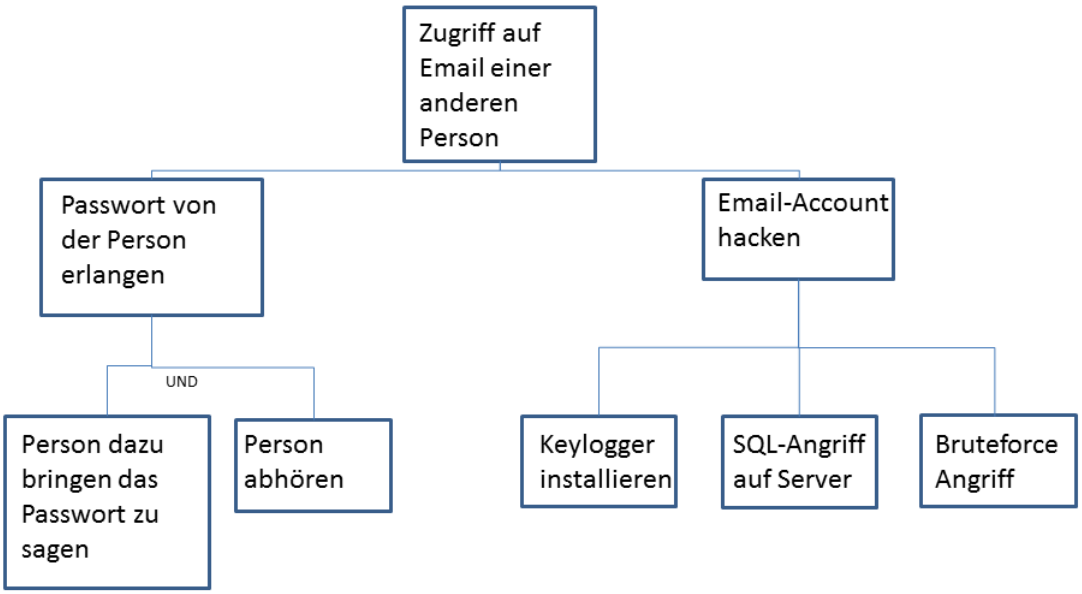


Abbildung 3.1: Beispiel eines Attack Trees

noch binäre oder kontinuierliche Werte zugewiesen werden. Diese Werte pflanzen sich von den Blättern des Baumes nach oben bis zur Wurzel fort. Jedem Knoten können auch mehrere Werte zugewiesen werden. Zunächst wird das Ziel des Angriffs als Wurzel festgelegt. Dann müssen alle möglichen Wege, das Ziel zu erreichen, durchdacht werden und als Knoten und Blätter hinzugefügt werden. Falls es noch benötigt wird, können den Knoten Werte zugewiesen werden, um die Knoten zu gewichten. Die Möglichkeit Werte zuzuweisen, macht Attack Trees zu einem Ansatz, der sowohl quantitativ als auch qualitativ ist. Das Ergebnis eines Attack Trees ist ein vollständig ausgearbeiteter Baum, für ein bestimmtes Ziel eines Angriffs anhand dessen man Gegenmaßnahmen treffen kann [Sch99].

3.3.1 Vorteile von Attack Trees

Attack Trees sind gut für kleine oder bekannte Angriffe, da sie eine gute Übersicht über die verschiedenen Angriffswege bieten. Außerdem kann man Attack Trees als Muster in verschiedenen Systemen wiederverwenden.

3.3.2 Nachteile von Attack Trees

Die Identifizierung von möglichen Szenarien ist sehr subjektiv und der Ansatz liefert keine Hilfestellung. Außerdem bekommt man als Ergebnis nur Angriffsszenarios.

3.4 Vergleich der Methoden

NIST 800-30 und OCTAVE Allegro sind jeweils für bestimmte Organisationen ausgelegt, NIST 800-30 auf staatliche und OCTAVE Allegro auf private. STPA-Sec und Attack Trees können überall angewendet werden. Bis auf die Attack Trees bieten alle Ansätze viele Hilfestellungen zur Durchführung, NIST 800-30 und OCTAVE Allegro jedoch mehr als STPA-Sec, was jedoch auch damit zusammenhängt, dass STPA-Sec ein noch sehr neuer Ansatz ist. Auch die Abläufe der Ansätze sind bei den zuletzt genannten gut definiert, doch auch hier sind die Abläufe von NIST 800-30 und OCTAVE Allegro aufgrund ihres Alters besser definiert als bei STPA-Sec. Dies führt dazu, dass man sich besser mit STPA-Sec auskennen muss, um die Analyse durchzuführen. STPA-Sec und OCTAVE Allegro benötigen eine sehr viel bessere Kenntnis vom System als die beiden anderen Ansätze. STPA-Sec liefert bessere Ergebnisse als alle anderen Ansätze, da es auch Verwundbarkeiten erkennt, die durch die Interaktion von Komponenten entstehen und nicht nur durch das Versagen von Komponenten. NIST 800-30 und OCTAVE Allegro hingegen liefern auch Schritte zum Umgang mit Risiken mit. Die Ergebnisse eines Attack Trees können als einziges in unterschiedlichen Systemen wiederverwendet werden.

3.5 Ergebnis des Vergleichs

STPA-Sec liefert mehr Ergebnisse als die anderen Ansätze. Es ist auch nicht nur auf bestimmte Organisationen beschränkt. Jedoch ist es nicht so klar definiert wie die anderen Ansätze und verfügt über weniger Hilfestellungen. Dies ist dadurch bedingt, dass der Ansatz noch vergleichsweise neu ist und es noch nicht viel Erfahrung zu diesem Ansatz gibt, dieser Nachteil wird aber zukünftig durch weitere Forschung vermindert werden.

Der Vergleich ist nochmals in Tabelle 3.1 zusammengefasst.

	NIST 800-30	OCTAVE Allegro	STPA-Sec	Attack Trees
Jahr	2002, letzte Revision 2012	OCTAVE 1999, OCTAVE Allegro 2007	2014	1999
Basis	Risikomanagement	Risikomanagement	Systemtheorie; hierarchische Kontrollstrukturen und unsichere Kontrollaktionen	Baumdiagramm
Qualitativ/ Quantitativ	Qualitativ	Qualitativ	Qualitativ	Beides möglich
Eingabe	Systemdokumentation Vergangene Systemattacken Daten von Geheimdiensten	Systemdokumentation Subjektive Einschätzung von Gefahren	STAMP Modell	Angriffsziele
Ausgabe	Risiken und Risikoeinschätzung Empfohlene Kontrollen	Verwundbarkeiten und Konsequenzen Vorschläge zur Milderung	STAMP Modell Unsichere Kontrollaktionen Kausale Faktoren	Angriffsszenarios

Prozess	1. Systemcharakterisierung 2. Gefahren identifizieren 3. Verwundbarkeiten identifizieren 4. Kontrollen analysieren 5. Wahrscheinlichkeiten festlegen 6. Auswirkungen analysieren 7. Risiken einschätzen 8. Kontrollen vorschlagen 9. Ergebnisdokumentation	1. Kriterien für Risikoabschätzung festlegen 2. Profil der Informationswerte entwickeln 3. Container der Informationswerte identifizieren 4. Betroffene Bereiche identifizieren 5. Gefahrenszenarios identifizieren 6. Risiken Identifizieren 7. Risiken analysieren 8. Milderungsansatz wählen	1. Verluste identifizieren 2. Verwundbarkeiten identifizieren 3. Systemweite Sicherheitsauflagen definieren 4. Kontrollstruktur erstellen 5. Unsichere Kontrollaktion identifizieren 6. Sicherheitsauflagen verfeinern 7. Kausale Szenarien identifizieren	1. Angriffsziele bestimmen 2. Angriffswege identifizieren (3. Angriffswegen Werte zuweisen)
Schwächen	Keine Bewertung der Organisation möglich Nur technische Perspektive	Umfangreiche Struktur kann unübersichtlich sein Wenig Hilfestellung für Gefahren Benötigt gute Kenntnis vom System	Benötigt viel Input um STAMP Modell zu bauen Benötigt sehr gute Kenntnis der Methode Kann zeitintensiv sein Noch wenig Erfahrungen zur Methode vorhanden	Sehr subjektiv Identifiziert nur Szenarios

Stärken	Umfangreich Flexibel Gut definierter Ablauf Gut für längerfristige Security-Einschätzungen Bietet viele Hilfestellungen	Umfangreich Flexibel Gut definierter Ablauf Vorteilhaft bei Projekten und einmaligen Beurteilungen Bietet viele Hilfestellungen	Erkennt Verwundbarkeiten, die andere Methoden nicht erkennen können Bietet viele Hilfestellungen	Gut für kleine oder bekannte Angriffe Lässt sich gut wiederverwenden

Tabelle 3.1: Vergleich der Methoden

4 STPA-Sec Plugin

Dieses Kapitel beschäftigt sich mit der Entwicklung des STPA-Sec-Plugins. Da sich die STPA-Sec-Methode eng an der normalen STPA-Methode orientiert wird das A-STPA-Plugin als Grundlage genommen, um das STPA-Sec-Plugin zu entwickeln. Daher wird zunächst das A-STPA-Plugin auf technischer Ebene analysiert.

4.1 Technische Umsetzung des A-STPA-Plugins

A-STPA ist nach dem Model-View-Controller-Entwicklungsmuster aufgebaut, dementsprechend besteht es aus drei Teilen.

Den ersten Teil bilden die Modelle. Modelle sind all die Klassen, welche die einzelnen Schritte des STPA-Ansatzes modellieren. Dies sind die Modelle für die Systembeschreibung, die Systemziele, Unfälle, die Verknüpfung der Unfälle und der Gefährdungen, die Kontrollstruktur, die unsicheren Kontrollaktionen, die entsprechenden Sicherheitsauflagen, das Prozessmodell und die kausalen Faktoren. Von XSTAMPP bekommt A-STPA die Modelle für die Gefährdungen und Sicherheitsauflagen zur Verfügung gestellt. Die Modelle sind notwendig, um das Speichern eines Projekts zu ermöglichen. Dazu werden die einzelnen Modelle mit XML-Annotationen versehen, welche das Modell auf eine XML-Modellierung abbilden. In A-STPA werden die Daten in dem eigens entworfenen hazx-Modell gespeichert. Der nächste Teil sind die Views. Das sind alle Benutzeroberflächen des Plugins, also die graphische Abbildung der einzelnen Schritte von A-STPA. Da alle Views in A-STPA editierbar sind, werden hierfür Editoren verwendet. In A-STPA werden die einzelnen Editoren von der Klasse *StandartEditorPart* abgeleitet, welche von XSTAMPP zur Verfügung gestellt wird und bereits vorgefertigte Funktionen bereitstellt die die Editoren benötigen.

Über die Editoren werden vom Nutzer Änderungen an den Daten vorgenommen. Da die Eclipse Rich Client Platform verwendet wird, sind die aus Eclipse bekannten Funktionen für die Manipulation der Editoren möglich. So ist es möglich die Größe der Editoren zu ändern oder mehrere Editoren nebeneinander darzustellen. Den letzten Teil bilden die Controller. Controller sind alle Klassen, welche Abläufe in A-STPA implementieren. Dies sind Funktionen wie das Hinzufügen, Bearbeiten oder Löschen. In A-STPA werden diese Funktionen in den DataModel-Interfaces für die einzelnen Modelle definiert und in der *DataModelController*-Klasse zusammengeführt. Sowohl die Interfaces als auch die *DataModelController*-Klasse implementieren dabei das *IDataModel*-Interface von XSTAMPP, um sicherzustellen, dass das

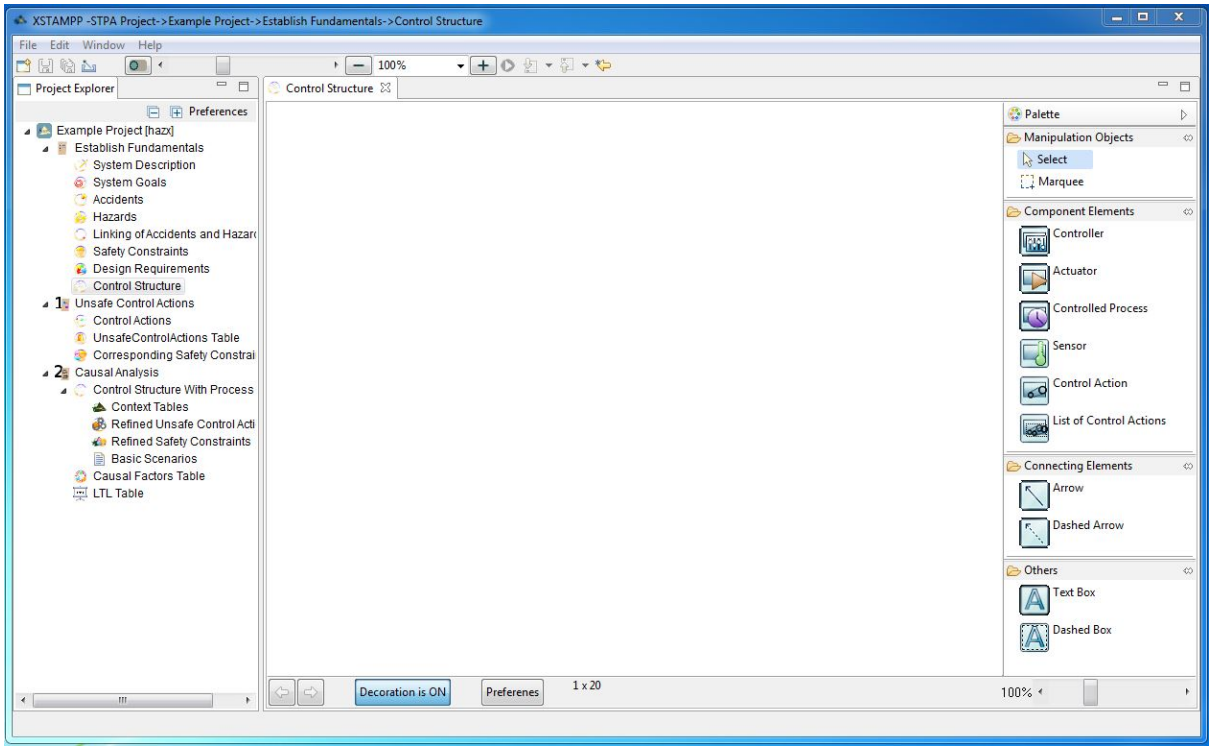


Abbildung 4.1: Der Kontrollstruktur-View

Entwurfsmuster des Beobachters, welches dazu dient, dass Änderungen an einem Objekt an von diesem abhängige andere Objekte weitergegeben werden, eingehalten wird. Für jedes Modell gibt es noch eine eigene Controller-Klasse, die die nötigen Funktionen der einzelnen Teile implementieren. Dadurch lassen sich die Controller für die Modelle flexibel gestalten, während die einzelnen Teile ohne Konflikte mit anderen Teilen ausgetauscht oder geändert werden können. Da die ganze Anwendung auf der Eclipse Rich Client Platform basiert, ist die Verwendung einer *plugin.xml*-Datei notwendig. Diese beschreibt, wie das Plugin die Plattform erweitert, welche Erweiterungspunkte es selbst ermöglicht und wie seine Funktionen implementiert werden. Hier werden auch die Informationen angegeben, die auf der Benutzeroberfläche angezeigt werden, wie Icons. Die Implementierung der Funktionen geschieht jedoch weiterhin in den einzelnen Java-Klassen. Wie die Erweiterungen des Plugins aussehen, ist in Abbildung 4.2 angegeben.

Im *xstampp.extension.steppedProcess* müssen beispielsweise die einzelnen Schritte hinzugefügt werden, welche im ProjectExplorer angezeigt werden. Diese Schritte haben eine eindeutige *editorId*, welche auf ebenfalls hier hinzugefügte Editors verweisen. Die Editors verweisen wiederrum auf die Klassen, in welchen die konkrete Implementierung der Klassen stattfindet. Auf diese Art und Weise wird das Plugin zusammengebaut. A-STPA bietet auch noch den Export der im Plugin erstellten Projekte an. Es lassen sich sowohl ganze Projekte, als auch einzelne Schritte exportieren. Die angebotenen Formate sind Portable Document Format (PDF), Comma-separated values (CSV) und Portable Network Graphics (PNG). Da die Projekte in einem

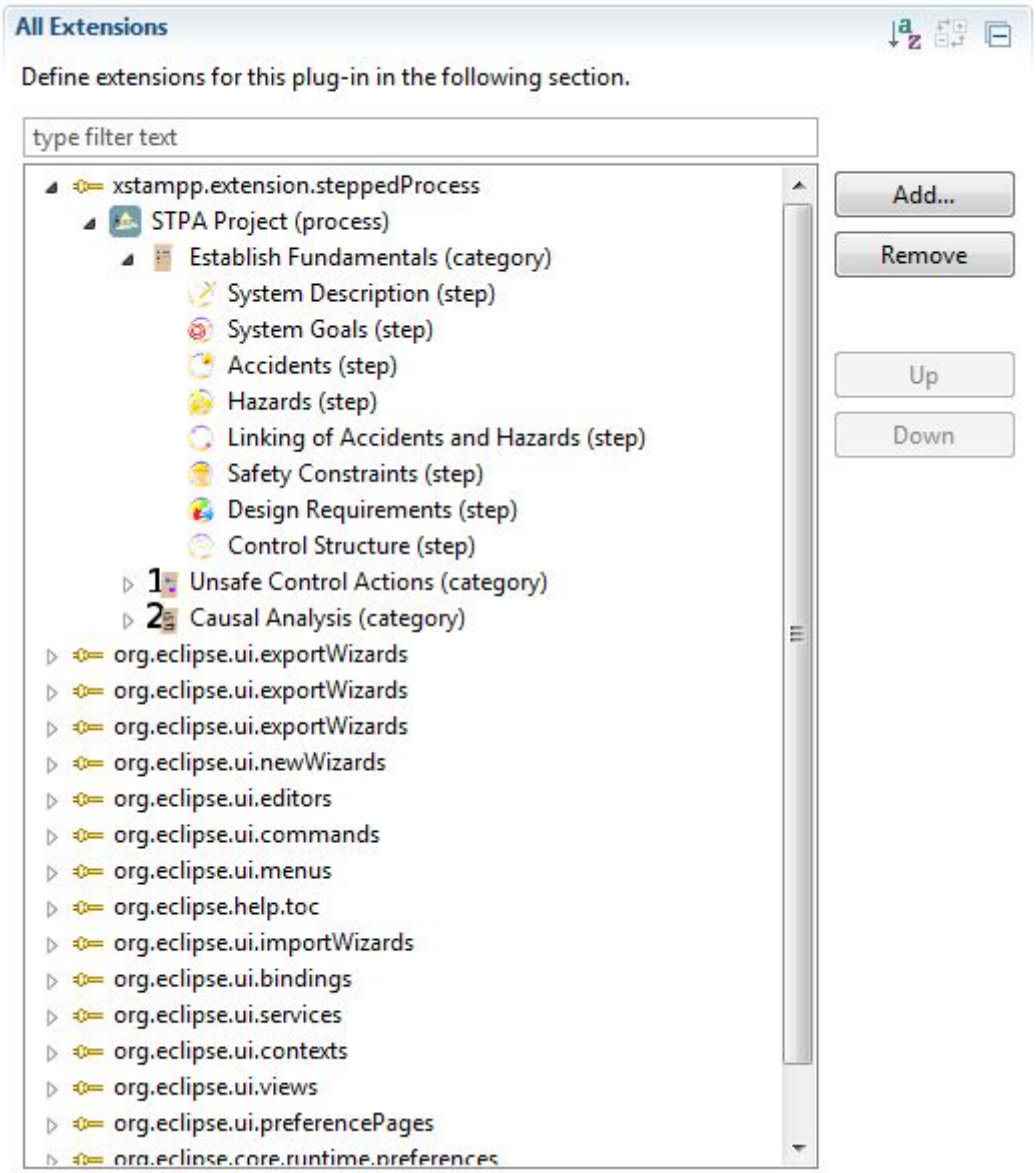


Abbildung 4.2: Editor für die Extensions

XML-Format gespeichert sind, wurde die „Extensible Stylesheet Language Transformation“ verwendet, um die Projekte in die anderen Formate umwandeln zu können. Dafür wurden für jeden Schritt XSL-Dateien angelegt, in denen alle nötigen Eigenschaften definiert sind, um die Umwandlung einer XML-Datei in ein anderes Dateiformat zu ermöglichen.

4.2 Implementierung des STPA-Sec-Plugins

Da sich STPA und STPA-Sec in einigen Punkten ähneln, bietet A-STPA-Plugin bereits viele Funktionen, die auch für ein STPA-Sec-Plugin nötig sind. Daher bietet es sich an, das STPA-Sec-Plugin auf dem A-STPA-Plugin aufzubauen und möglichst viele Komponenten wiederzuverwenden. Da sich das A-STPA-Plugin auch noch in der Entwicklung befindet, ist es sinnvoll, direkt auf die Komponenten des Plugins zuzugreifen und diese zu verwenden, anstatt lediglich eine Kopie des Plugins zu machen und die nötigen Änderungen durchzuführen. Dies ist auch möglich, da die Plugins ihre Klassen zur Laufzeit zur Verfügung stellen können.

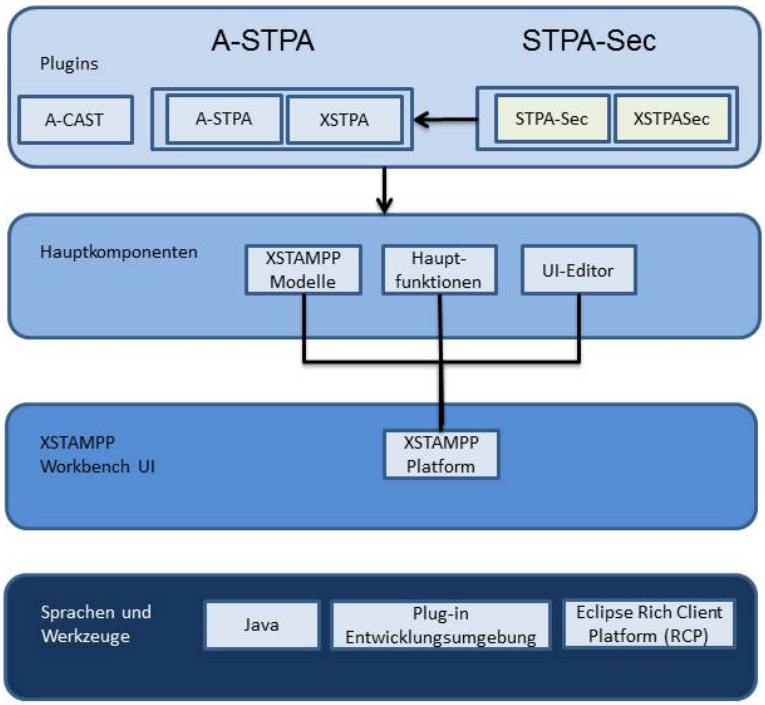


Abbildung 4.3: Erweiterung von XSTAMPP um das STPA-Sec Plugin

Das A-STPA-Plugin besteht insgesamt aus 44 Paketen, 294 Klassen, 15 für den Export benötigten xml-Schemas, 155 Bildern, einer bundle.properties-Datei, einer MANIFEST.MF-Datei und einer plugin.xml-Datei, sowie 9 HTML-Dateien, 2 xml-Dateien und 10 Bildern für die Hilfe-Seiten. Das XSTPA-Plugin besteht nochmals aus 7 Paketen, 35 Klassen, 24 Bildern, einer bundle.properties-Datei, einer MANIFEST.MF-Datei und einer plugin.xml-Datei, sowie 8 HTML-Dateien, 6 xml-Dateien und 16 Bildern für die Hilfe-Seiten. Der erste Schritt ist die Identifizierung der Elemente, die wiederverwendet werden können. Dazu muss man die Schritte vergleichen, in denen sich STPA-Sec und STPA unterscheiden. Diese sind aus dem Kapitel 2.3.4.3 zu entnehmen. Diese sind insbesondere in der Terminologie zu finden. Die Terminologie wurde für STPA-Sec anhand des Papers von Schmittner [SMP16] angepasst und jeder Unterschied im Plugin gefunden und in Tabelle 4.1 aufgelistet.

STPA	STPA for Security	Unterschiede
Establish Fundamentals	Establish Fundamentals	Nein
System Description	System Description	Nein
System Goals	System Goals	Nein
Accidents	Losses	Ja
Hazards	Vulnerabilities	Ja
Linking of Accidents and Hazards	Linking of Losses and Vulnerabilities	Ja
Design Requirements	Design Requirements	Nein
Control Structure	High Level Control Structure Model	Ja
Unsafe Control Actions	Unsafe Control Actions	Ja
Unsafe Control Actions Table	Unsecure Control Actions Table	Ja
Corresponding Safety Constraints	Corresponding Security Constraints	Ja
Causal Analysis	Causal Analysis	Nein
Control Structure With Process Model	Control Structure With Process Model	Nein
Context Tables	Context Tables	Ja
Refined Unsafe Control Actions	Refined Unsafe Control Actions	Ja
Refined Safety Constraints	Refined Security Constraints	Ja
Causal Factors Table	Causal Factors Table	Ja
LTL Formula Table	LTL Formula Table	Nein

Tabelle 4.1: Unterschiede im Ablauf zwischen A-STPA und STPA-Sec

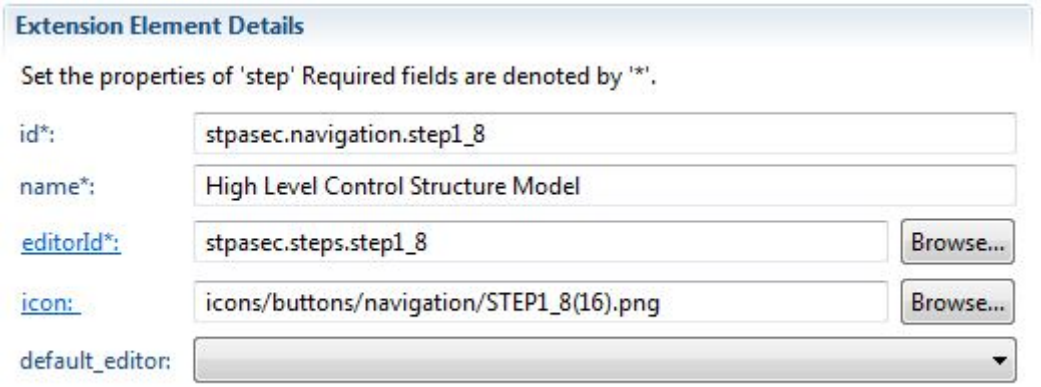


Abbildung 4.4: Editor für die einzelnen Extensions

Wie man sieht unterscheiden sich viele der Schritte voneinander. Im Fall der Kontrollstruktur unterscheidet sich die Bezeichnung jedoch nur in der Bezeichnung des Schrittes, der Editor ist davon nicht betroffen. Ebenso ist der Schritt *Unsecure Control Actions*, wie auch *Establish Fundamentals* und *Causal Analysis*, lediglich eine Kategorie und verfügt über keinen eigenen Editor, ist also auch nur eine Änderung der Bezeichnung. Diese Änderungen werden in der *plugin.xml*-Datei im *Extensions*-Fenster (siehe Abbildung 4.4) gemacht. Damit können alle Klassen, die die Editoren für die Schritte *System Description*, *System Goals*, *Design Requirements*, *High Level Control Structure Model*, *Control Actions*, *Control Structure with Process Model*, *LTL Formula Table* und ihre dazugehörigen Hilfsklassen wiederverwendet werden. Des Weiteren besitzt das A-STPA-Plugin für jedes Modell ein eigenes Interface. Die Interfaces enthalten keine Informationen, die für das STPA-Sec-Plugin abgeändert werden müssen, also kann man diese ebenfalls wiederverwenden. Auch bei den Klassen, die die Modelle implementieren, gibt es einige die man direkt wiederverwenden kann. So sind das unter anderem alle Klassen, die Modelle für die wiederverwendeten Schritte bereitstellen, also Modelle für *System Description*, *System Goals*, etc. Dies gilt auch für ihre Hilfsklassen. Im Falle des Modells für die *Causal Factors* befinden sich die Daten, die man für STPA-Sec abändern muss, in Hilfsklassen. Daher ist es nicht nötig ein eigenständiges Modell für diese zu implementieren, sondern man kann auf die Klasse aus A-STPA zurückgreifen. Eine weitere Art von Klassen sind die Controller der Modelle. Diese enthalten ebenfalls keine für STPA-Sec abzuändernden Informationen und können wiederverwendet werden. Schließlich können auch die Export-Klassen der Klassen aus A-STPA inklusive ihrer dazugehörigen xsl-Schemas wiederverwendet werden. Klassen die von STPA-Sec selbst implementiert werden müssen sind somit:
Editoren für *Losses*, *Vulnerabilities*, *Linking of Losses and Vulnerabilities*, *Security Constraints*, *Unsecure Control Actions Table*, *Corresponding Security Constraints*, *Context Tables*, *Refined Unsecure Control Actions*, *Refined Security Constraints* und *Causal Factors Table*.
Modelle und Controller für *Losses*, *Vulnerabilities*, *Security Constraints*, *Unsecure Control Actions*, die Hilfsklassen für *Causal Factors*, sowie den Controller für das gesamte Datenmodell.
Export-Klassen und dazugehörige xsl-Dateien für *Losses*, *Vulnerabilities*, *Linking of Losses and*

Vulnerabilities, *Security Constraints*, *Unsecure Control Actions*, *Corresponding Security Constraints*, *Context Tables*, *Refined Unsecure Control Actions*, *Refined Security Constraints* und *Context Tables*. Letztlich benötigt STPA-Sec noch grundlegende Klassen, die für die Funktionalität nötig sind. Dies wären: eine *Activator*-Klasse, die den Startpunkt des Plugins darstellt, eine Klasse für externalisierte Strings, eine Klasse für die Einstellungen, sowie Wizard-Klassen, die für das Erstellen eines neuen Plugins und das Durchführen von Import und Export nötig sind.

Listing 4.1 Klasse, die Verluste modelliert

```
/**
 * Class for losses
 *
 */
@XmlRootElement(name = "loss")
public class Loss extends ATableModel {

	public Loss(String title, String description, int number) {
		super(title, description, number);
	}

	public Loss() {
		// empty constructor for JAXB
	}

	@Override
	public String getIdString() {
		return "L-" + this.getNumber();
	}
}
```

Ein Beispiel für eine Klasse, die ein Modell implementiert ist in Listing 4.1 zu sehen. Es handelt sich hierbei um die Klasse die Verluste modelliert. Sie erweitert die abstrakte *ATableModel*-Klasse aus dem A-STPA Plugin, die bereits notwendige Attribute und Funktionen implementiert, etwa den Namen des Verlustes oder die Beschreibung. Die Klasse benötigt auch noch eine XML-Annotation, sodass ein Verlust-Objekt beim Speichern in der XML-Datei gespeichert werden kann.

Listing 4.2 Klasse, die den Editor für Verluste modelliert

```
/**
 * Class for the editor of Losses
 */
public class LossesView extends CommonTableView<IAccidentViewDataModel> {

	public static final String ID = "stpasec.steps.step1_2";

	// the accident currently displayed in the text widget
	private Loss displayedLoss;

	public LossesView() {

	}

	/**
	 * Create contents of the view part.
	 * @param parent The parent composite
	 */
	@Override
	public void createPartControl(Composite parent) {
	...
```

Ein weiteres Beispiel ist in Listing 4.2 für die Implementierung des Editors für Verluste angegeben. Die Klasse erweitert die *CommonTableView*-Klasse, die die Standardmethoden und Attribute für Editoren des A-STPA Plugins implementieren. Sie nutzt auch das Interface *IAccidentViewDataModel* von A-STPA, welches nötige Methoden für das Abrufen und Manipulieren der Daten des Editors vorschreibt, die in einem separaten Controller implementiert werden. Weiterhin benötigt die Klasse eine ID, über die das Plugin über die plugin.xml den Editor aufrufen kann. In der *createPartControl*-Methode wird die grafische Darstellung des Editors auf SWT-Basis implementiert.

4.3 Erweiterung des STPA-Sec-Plugins

Da sich security-bezogene Verwundbarkeiten auch auf die Safety eines Systems auswirken können, soll die Erweiterung des Plugins die Möglichkeit bieten, dies auch darzustellen. Dazu wird das Modell der unsicheren Kontrollaktionen um zwei weitere Attribute erweitert, die markieren, ob die Kontrollaktion unsicher im Sinne der Security oder der Safety ist. Außerdem wird das Plugin um zwei weitere Editoren erweitert. Der erste Editor in Abbildung 4.5 erweitert

die Kontrollstruktur und gibt eine Übersicht der Kontrollaktionen, sowie der dazugehörenden unsicheren Kontrollaktionen und ob die unsichere Kontrollaktion unsicher in Bezug auf Security oder unsicher in Bezug auf Safety ist. Die beiden letzten Attribute lassen sich hier editieren.

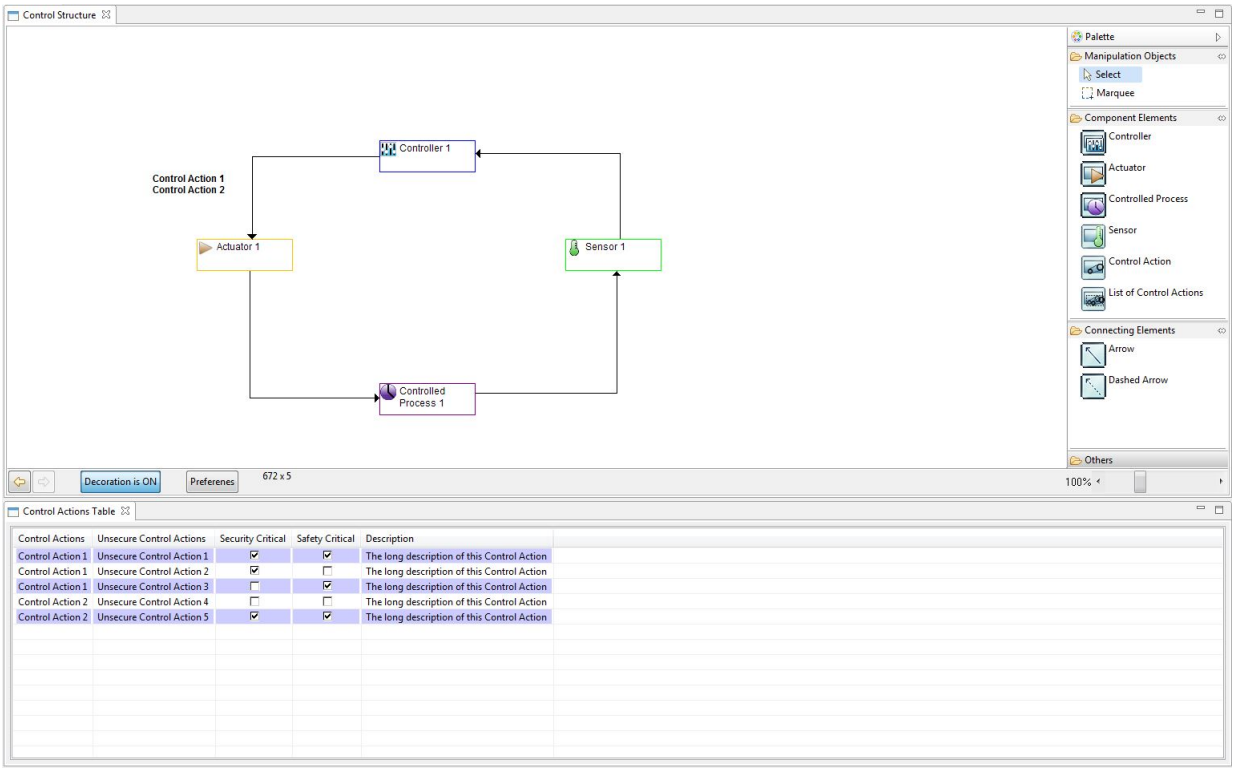
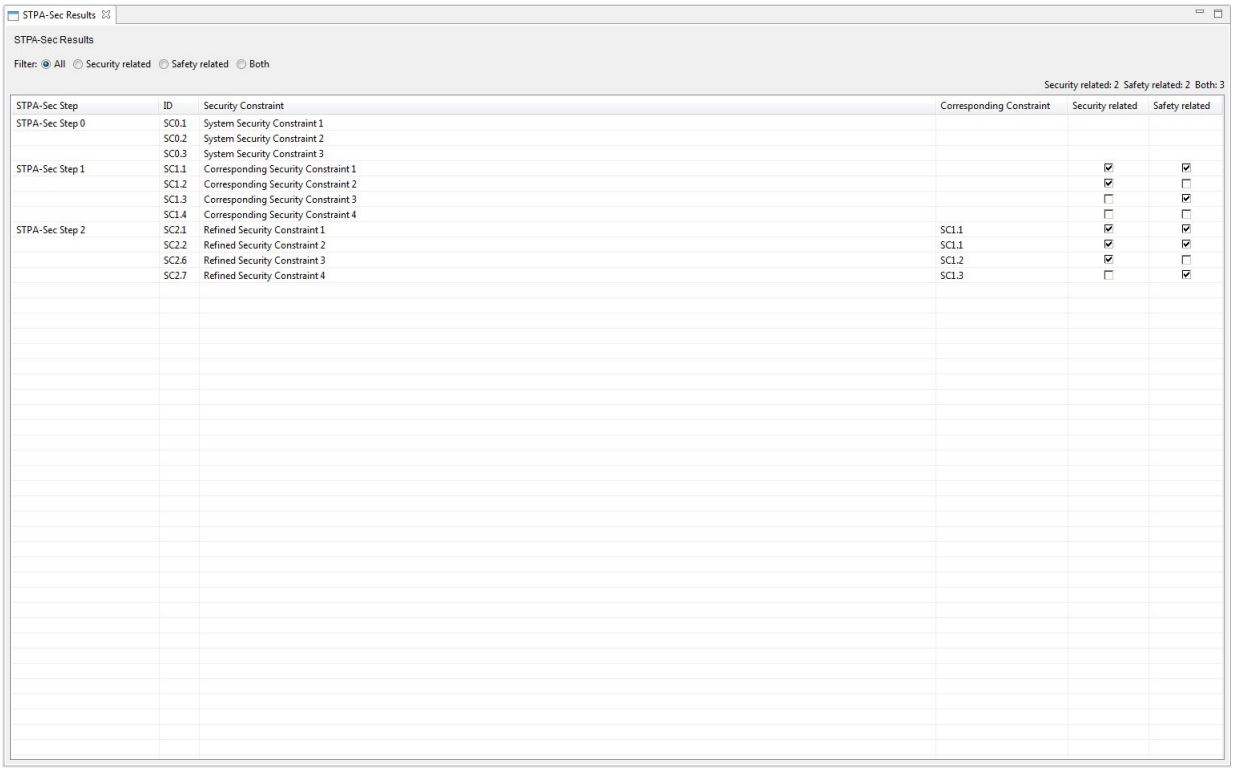


Abbildung 4.5: Editor für die unsicheren Kontrollaktionen

Der zweite Editor in Abbildung 4.6 zeigt die Sicherheitsauflagen, die in allen Schritten identifiziert wurden. In der Liste wird angegeben in welchem Schritt eine Sicherheitsauflage identifiziert wurde, die ID, die Beschreibung, ob sie sich auf die Sicherheit im Sinne der Security oder der Safety auf das System auswirkt, sowie für verfeinerte Sicherheitsauflagen die Sicherheitsauflagen, von denen die Sicherheitsauflage abhängt. Die Liste lässt sich filtern, sodass alle Sicherheitsauflagen, nur die safety-bezogenen, nur die security-bezogenen oder safety- und security-bezogenen angezeigt werden. Auch wird die Anzahl der jeweiligen Sicherheitsauflagen angezeigt. Die zwei letzten Attribute lassen sich ebenfalls bearbeiten. Nach Abschluss der Bearbeitung lässt sich die Liste letztlich auch exportieren.



STPA-Sec Step	ID	Security Constraint	Corresponding Constraint	Security related	Safety related
STPA-Sec Step 0	SC0.1	System Security Constraint 1			
	SC0.2	System Security Constraint 2			
	SC0.3	System Security Constraint 3			
STPA-Sec Step 1	SC1.1	Corresponding Security Constraint 1		☑	☑
	SC1.2	Corresponding Security Constraint 2		☑	☐
	SC1.3	Corresponding Security Constraint 3		☐	☑
	SC1.4	Corresponding Security Constraint 4		☐	☐
STPA-Sec Step 2	SC2.1	Refined Security Constraint 1	SC1.1	☑	☑
	SC2.2	Refined Security Constraint 2	SC1.1	☑	☑
	SC2.6	Refined Security Constraint 3	SC1.2	☑	☐
	SC2.7	Refined Security Constraint 4	SC1.3	☐	☑

Abbildung 4.6: Liste der Sicherheitsauflagen

5 Anwendung von STPA-Sec auf autonomes Fahren

In diesem Kapitel soll das STPA-Sec-Plugin anhand eines aktuellen Beispiels getestet werden. Dazu wird das System eines autonomen Fahrzeugs verwendet.

5.1 Grundlagen von autonomen Fahrzeugen

Das Automobil ist eine der wichtigsten Erfindungen und hatte spätestens seit dem 20. Jahrhundert einen enormen Einfluss auf die Menschheit. Täglich verwenden hunderte Millionen Menschen Automobile als Transportmittel. Ursprünglich nur aus dem nötigsten, wie Motor, Karosserie und Rädern, bestehend, wurden Automobile im Rahmen der technologischen Entwicklung immer komplexer und leistungsfähiger. So sind bereits heutzutage moderne Fahrzeuge nicht nur rein mechanische Systeme, sondern sie werden mit Computern ausgestattet, die zur Steuerung der verschiedenen Bauteile innerhalb der Systeme verwenden werden. Die Software zur Steuerung in diesen Fahrzeugen besteht aus Millionen Zeilen Code [KCR+10]. Doch trotz all dieser modernen Erweiterungen, sind Automobile noch nicht in der Lage selbstständig ohne einen menschlichen Fahrer zu fahren. Die Entwicklung von autonomen Fahrzeugen ist also eine der größten Herausforderungen der heutigen Zeit. Die Herausforderung daran liegt weniger darin, ein solches Automobil zu bauen, da die Technologie dafür bereits weitgehend vorhanden ist. Das Problem liegt vielmehr darin, die richtige Software zu entwickeln, sodass diese das Automobil sicher steuern kann. Das Fahrzeug muss Straßen erkennen, auf diesen beschleunigen, bremsen und die Spur halten, andere Verkehrsteilnehmer erkennen, zum richtigen Ort navigieren und das Ganze in einer Umwelt, in der viele Störungen auf das Fahrzeug einwirken, Entscheidungen in Sekundenbruchteilen getroffen werden müssen und diese weitreichende Konsequenzen nach sich ziehen können. Solch eine komplexe Software ist jedoch auch sehr fehleranfällig und angreifbar. Es ist also geboten bei der Entwicklung besonders großen Wert auf die Sicherheit dieser Systeme zu legen. Dies macht autonome Fahrzeuge zu einem perfekten Beispiel für einen Test des STPA-Sec-Plugins.

5.2 Technischer Hintergrund

Zunächst muss kurz geklärt werden, wie moderne Systeme für autonomes Fahren aussehen. Nahezu alle autonomen Fahrzeuge verfügen über fünf grundlegende Funktionen für das Fahren.
Wahrnehmung: Wahrnehmung ist das Erkennen der Umwelt des Fahrzeugs. Dazu werden verschiedene Sensoren verwendet, wie Radar, Lidar oder Kameras.
Ortung: Die Ortung bestimmt die Position des Fahrzeugs mithilfe von GPS, der Bewegungsrichtung und Geschwindigkeit des Fahrzeugs und anhand von Kartenmaterial.
Planung: Die Planung bestimmt die Bewegung und das Verhalten des Fahrzeugs, die es anhand der Informationen aus der Wahrnehmung und Ortung berechnet.
Kontrolle: Die Kontrolle ist die Ausübung der Befehle, die von der Planung berechnet wurden, etwa das Bremsen, Beschleunigen oder Steuern.
Systemmanagement: Das Systemmanagement kontrolliert schließlich sämtliche Funktionen des Fahrzeugs.
Jede dieser Funktionen verfügt über mehrere Komponenten, die einzelne Unterfunktionen implementieren. Diese Komponenten können auf zwei verschiedene Arten implementiert werden: Zentralisiert oder dezentralisiert. Bei einer zentralisierten Architektur werden sämtliche Komponenten an eine Recheneinheit angeschlossen, welche sämtlichen notwendigen Berechnungen durchführt. Der Vorteil bei einer solchen Architektur ist, dass das Systemmodell einfacher ist. Außerdem benötigt man keine zusätzlichen Kommunikationswege, was den Verlust und die Geschwindigkeit des Teilens von Information verringert. Der Nachteil hingegen ist, dass die Recheneinheit eine hohe Rechenbelastung hat und dementsprechend stark sein muss. Außerdem bedeutet ein Ausfall der Recheneinheit den Ausfall des gesamten Systems. Bei einer dezentralisierten Architektur hingegen werden verschiedene Komponenten von verschiedenen Recheneinheiten gesteuert. Die Informationen aus den Komponenten werden über ein gemeinsames Kommunikationssystem an die anderen Komponenten geschickt. Vorteil ist, dass die Rechenlast aufgeteilt wird. Außerdem kann der Ausfall einer Recheneinheit besser abgefangen werden. Nachteilig ist, dass das System komplexer wird und dass eventuell Kompatibilitätsprobleme zwischen den Komponenten auftauchen können. Diese Nachteile lassen sich jedoch durch eine gute Planung verringern [JKK+14].

5.3 Durchführung der STPA-Sec-Analyse

Anhand der vorhergehenden Beschreibungen wird die Analyse mithilfe des Plugins durchgeführt.

5.3.1 Grundlagen festlegen

5.3.1.1 Systembeschreibung

Ein autonomes Auto ist ein System für den Transport, welches in der Lage ist autonom zu navigieren, abhängig von Informationen die es durch die Umwelt erhält.

5.3.1.2 Systemziele

Das Grundziel eines Fahrzeuges ist klar die Sicherheit der Passagiere, sowie das Navigieren an den erforderlichen Ort. In dem System eines Fahrzeugs werden jedoch auch viele persönliche Informationen über die Passagiere gespeichert, die einen Wert für einen böswilligen Angreifer darstellen können. Deswegen gehört der Schutz dieser Informationen ebenfalls zu den Zielen des Systems.

ID	Systemziele
1	Für die Sicherheit der Passagiere sorgen
2	Die Passagiere zum korrekten Ort bringen
3	Keine persönlichen Informationen an unautorisierte Nutzer geben
4	Angeforderte Informationen an autorisierte Nutzer geben
5	Über angriffssichere Kanäle kommunizieren

Tabelle 5.1: Systemziele

5.3.1.3 Verluste

Anhand der Systemziele wurden folgende Verluste identifiziert:

ID	Verluste	Verknüpfung mit Verwundbarkeiten
L-1	Verlust des Lebens eines Passagiers	V-1,V-2, V-3, V-4, V-9
L-2	Verlust des Lebens einer außenstehenden Person	V-1,V-2, V-3, V-4, V-9
L-3	Physische Verletzung an einer Person	V-1,V-3, V-4, V-4, V-9
L-4	Schaden an Objekten	V-1,V-2, V-3, V-4, V-5, V-6,V-9
L-5	Verlust von Information	V-7, V-8, V-9

Tabelle 5.2: Verluste

5.3.1.4 Verwundbarkeiten

ID	Verwundbarkeiten	Verknüpfung mit Verlusten
V-1	Auto fährt mit einer unsicheren Geschwindigkeit	L-1, L-2, L-3, L-4
V-2	Auto fährt in eine unsichere Richtung	L-1, L-2, L-3, L-4
V-3	Das System sendet unsichere Signale an die Umwelt	L-1, L-2, L-3, L-4
V-4	Das Auto fährt zu einem unsicheren Ort	L-1, L-2, L-3, L-4
V-5	Das Auto fährt in einem unsicheren Gang	L-4
V-6	Der Automotor operiert außerhalb der bestimmten Grenzen	L-4
V-7	Persönliche Informationen werden gegenüber unautorisierten Personen exponiert	L-5
V-8	Information ist autorisierten Personen nicht verfügbar	L-5
V-9	Ein Hacker bekommt Zugriff auf das System	L-1, L-2, L-3, L-4, L-5

Tabelle 5.3: Verwundbarkeiten

5.3.1.5 Sicherheitsauflagen

ID	Sicherheitsauflagen
SC0.1	Das Auto darf niemals mit einer unsicheren Geschwindigkeit fahren
SC0.2	Das Auto darf niemals in eine unsichere Richtung fahren
SC0.3	Das System darf niemals unsichere Signale an die Umwelt senden
SC0.4	Das Auto darf niemals zu einem unsicheren Ort fahren
SC0.5	Das Auto darf niemals in einem unsicheren Gang fahren
SC0.6	Der Automotor darf niemals außerhalb der bestimmten Grenzen operieren
SC0.7	Persönliche Informationen dürfen niemals gegenüber unautorisierten Personen exponiert werden
SC0.8	Information muss autorisierten Personen immer verfügbar sein
SC0.9	Ein Hacker darf niemals Zugriff auf das System bekommen

Tabelle 5.4: Sicherheitsauflagen

5.3.1.6 Kontrollstruktur

Die Kontrollstruktur wurde anhand der Informationen aus Kapitel 4.2. erstellt. Es wird nochmals darauf hingewiesen, dass die Kontrollstruktur lediglich eine funktionale und keine technische Darstellung des Systems ist. Es wurde eine dezentralisierte Architektur als Grundlage angenommen, daher werden hier auch Berechnungen in den einzelnen Komponenten des Systems durchgeführt und lediglich die benötigten Informationen an die Planungskomponente „Selfdriving Software“ weitergeleitet. Des Weiteren verfügt das System über die im Abschnitt 5.2 besprochenen Sensoren, die in der Komponente „Environment Sensors“ zusammengefasst wurde. Das Automobil verfügt aber auch noch über interne Sensoren, welche solche Dinge, wie Motordrehzahl oder die Temperatur der Bremsen messen und die Messungen an die „Selfdriving Software“ weiterleiten. Die Kontrolle über das Automobil übt die Sofware

über den „Autonomous Driving Actuator“ aus, welcher Kontrollaktionen empfängt und diese mechanisch umsetzt. Das System hat außerdem auch noch eine Anbindung an einen Server, auf den es über eine Schnittstelle zugreift, welche auch GPS-Daten für die Sofware zur Verfügung stellt. Dieses System besitzt außerdem noch über eine Schnittstelle zur Kommunikation mit anderen Fahrzeugen, sowie eine Schnittstelle über die Passagiere auf das Fahrzeug zugreifen können, etwa um persönliche Daten abzurufen und auf dem Bildschirm des Autos darzustellen. Letztlich verfügt das Automobil noch über eine USB-Verbindung, über die Nutzer Zugriff auf das Fahrzeug bekommen können.

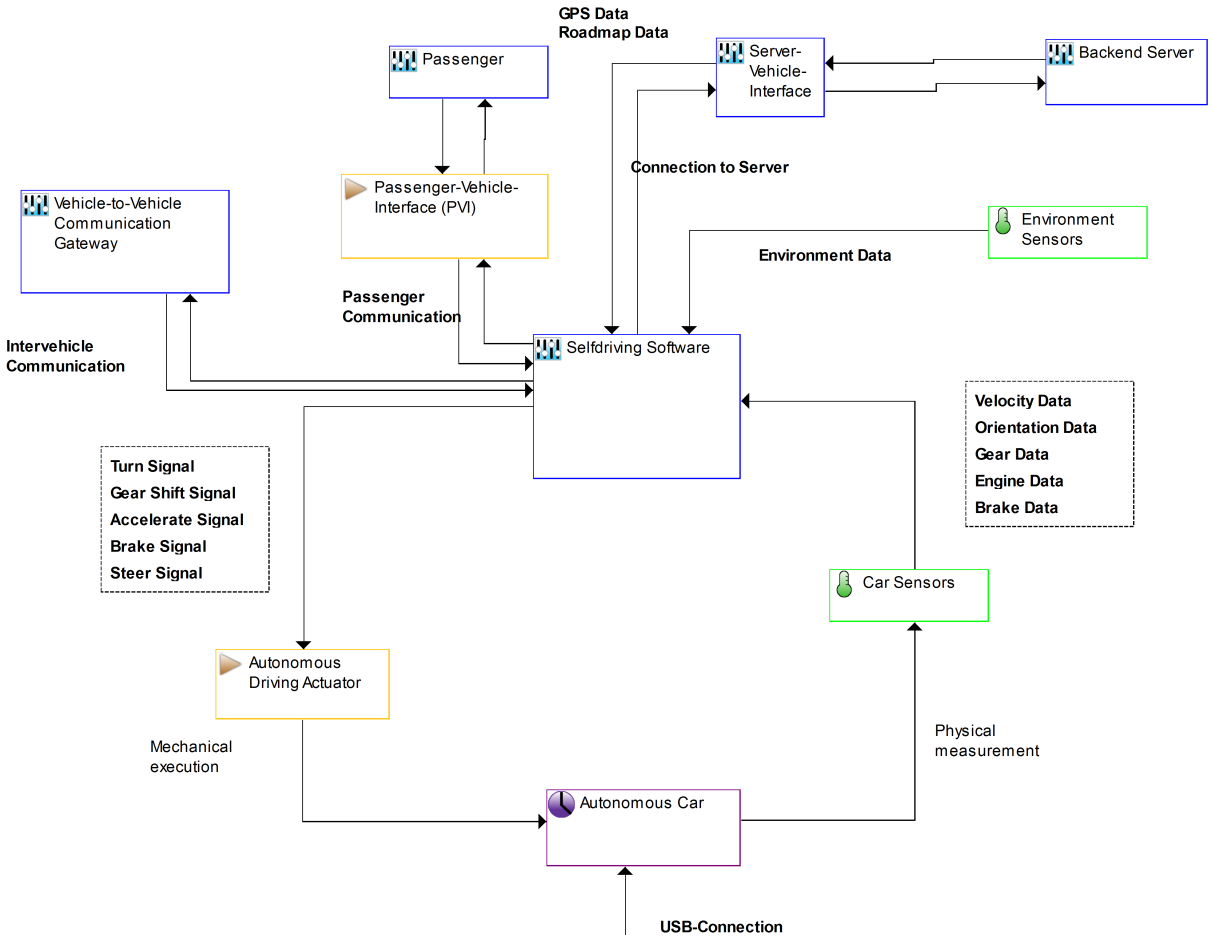


Abbildung 5.1: Kontrollstruktur eines autonomen Fahrzeugs

5.3.2 Unsichere Kontrollaktionen

5.3.2.1 Kontrollaktionen

Die Kontrollaktionen, die in der Kontrollstruktur identifiziert wurden, sind:

ID	Kontrollaktionen	Quelle	Ziel
1	Steer Signal (Steuersignal)	Selfdriving Software	Autonomous Driving Actuator
2	Brake Signal (Bremssignal)	Selfdriving Software	Autonomous Driving Actuator
3	Accelerate Signal (Beschleunigungssignal)	Selfdriving Software	Autonomous Driving Actuator
4	Gear Shift Signal (Signal zum Gang schalten)	Selfdriving Software	Autonomous Driving Actuator
5	Turn Signal (Signal zum Blinker setzen)	Selfdriving Software	Autonomous Driving Actuator
6	Brake Data (Bremsdaten)	Car Sensors	Selfdriving Software
7	Engine Data (Motordaten)	Car Sensors	Selfdriving Software
8	Gear Data (Gangdaten)	Car Sensors	Selfdriving Software
9	Orientation Data (Ausrichtungsdaten)	Car Sensors	Selfdriving Software
10	Roadmap Data (Kartendaten)	Server-Vehicle-Interface	Selfdriving Software
11	Velocity Data (Geschwindigkeitsdaten)	Car Sensors	Selfdriving Software
12	Environment Data (Umweltdaten)	Environment Sensors	Selfdriving Software
13	USB-Connection (USB-Verbindung)		Autonomous Car
14	Passenger Communication (Passagier-Kommunikation)	Passenger-Vehicle-Interface (PVI)	Selfdriving Software
15	Intervehicle Communication (Kommunikation zwischen Fahrzeugen)	Vehicle-to-Vehicle Communication Gateway	Selfdriving Software
16	GPS Data (GPS Daten)	Server-Vehicle-Interface	Selfdriving Software
17	Connection to Server (Verbindung zum Server)	Selfdriving Software	Server-Vehicle-Interface

Tabelle 5.5: Kontrollaktionen

5.3.2.2 Tabelle der unsicheren Kontrollaktionen

Es wurde sich darauf konzentriert, mögliche Angriffswege auf das System zu finden, dementsprechend zielen alle unsicheren Kontrollaktionen auf Absicht aus. Bei der Durchführung der Analyse wurden 60 verschiedene unsichere Kontrollaktionen gefunden. Da dies den Umfang dieser Arbeit sprengen würde, wird hier nur eine Auswahl abgebildet. In der Klammer unter den Beschreibungen sind die Verwundbarkeiten, auf die sich die unsichere Kontrollaktion auswirkt, angegeben.

Kontrollaktion	Nicht gegeben führt zur Verwundbarkeit	Gegeben führt zur Verwundbarkeit	Falsches Timing oder falsche Reihenfolge führt zur Verwundbarkeit	Zu früh gestoppt oder zu lange angewandt führt zur Verwundbarkeit
Steuersignal	Das Steuersignal wird von einem böswilligen Angreifer blockiert, obwohl das Fahrzeug den Kurs wechseln sollte [V-4]	Ein Steuersignal wird von einem böswilligen Angreifer injiziert, während das Fahrzeug auf einer geraden Strecke fährt [V-4, V-9]	Steuersignal wird von einem böswilligen Angreifer so manipuliert, dass es gesendet wird, bevor das Fahrzeug die von den Umweltdaten gemeldete Kurve erreicht [V-4, V-9]	Das Steuersignal wird von einem böswilligen Angreifer unterbrochen, während sich das Fahrzeug in einer Kurve befindet [V-4]
Bremssignal	Das Bremssignal wird von einem böswilligen Angreifer blockiert, während das Fahrzeug sich einem stationären oder langsameren Objekt nähert [V-1]	Ein Bremssignal wird von einem böswilligen Angreifer injiziert, obwohl sich vor dem Fahrzeug kein Objekt in einer gefährlichen Distanz befindet, sodass das Fahrzeug für den nachfolgenden Verkehr unberechenbar bremst [V-1, V-9]	Das Bremssignal wird von einem böswilligen Angreifer verzögert, sodass es sich einem stationären oder langsameren Objekt in einer immer noch gefährlichen Geschwindigkeit nähert [V-1, V-9]	Das Bremssignal wird durch einen böswilligen Angreifer zu früh gestoppt, sodass es sich einem stationären oder langsameren Objekt in einer immer noch gefährlichen Geschwindigkeit nähert [V-1]
Beschleunigungssignal	Das Beschleunigungssignal wird von einem böswilligen Angreifer geblockt, obwohl das Fahrzeug beschleunigen sollte und somit für den nachfolgenden Verkehr unberechenbar handelt [V-1]	Ein Beschleunigungssignal wird von einem böswilligen Angreifer injiziert, obwohl das Fahrzeug still steht und sich ein stationäres Objekt vor dem Fahrzeug befindet [V-1, V-9]	Das Beschleunigungssignal wird von einem böswilligen Angreifer so manipuliert, dass es ausgeführt wird, bevor überprüft wurde, ob sich ein Objekt vor dem Fahrzeug befindet [V-1, V-9]	Das Beschleunigungssignal wird von einem böswilligen Angreifer so manipuliert, dass es zu lang angewandt wird und das Fahrzeug sich einem langsameren oder stationären Objekt gefährlich nähert [V-1, V-9]
Geschwindigkeitsdaten	Geschwindigkeitsdaten werden von einem böswilligen Angreifer geblockt, sodass die Software von einer veralteten Geschwindigkeit ausgeht, obwohl die Geschwindigkeit verändert wurde [V-1]	Geschwindigkeitsdaten, die eine zu langsame Geschwindigkeit angeben, werden von einem böswilligen Angreifer injiziert, sodass das Fahrzeug weiter beschleunigt [V-1, V-9]	Geschwindigkeitsdaten werden von einem böswilligen Angreifer so verzögert, dass sie nach dem Beschleunigungssignal bei der Sofware ankommen und das Beschleunigungssignal mit veralteten Daten berechnet wurde [V-1, V-9]	Nicht unsicher
Motordaten	Motordaten werden von einem böswilligen Angreifer blockiert, obwohl der Motor mit einer zu hohen Drehzahl operiert und der Gang gewechselt werden sollte [V-6]	Motordaten, die angeben, dass die Drehzahl des Motors in einem sicheren Bereich ist, werden von einem böswilligen Angreifer injiziert, sodass die Software kein Signal zum Gang wechseln sendet [V-6, V-9]	Nicht unsicher	Nicht unsicher
Umweltdaten	Umweltdaten werden von einem böswilligen Angreifer geblockt, sodass die Software keine Information über ein sich vor dem Fahrzeug befindliches Objekt erhält [V-1]	Ein böswilliger Angreifer injiziert Umweltdaten, die kein Objekt vor dem Fahrzeug anzeigen, obwohl sich dort eines befindet [V-1, V-9]	Umweltdaten werden von einem böswilligen Angreifer so verzögert, dass sie erst eintreffen, nachdem ein Beschleunigungssignal gegeben wurde, sodass es mit alten Daten berechnet wurde, die kein Objekt vor dem Fahrzeug angeben, obwohl sich dort eines befindet [V-1]	
USB-Verbindung	USB-Verbindung wird von einem böswilligen Angreifer blockiert, sodass autorisierte Nutzer keinen Zugriff auf ihre Informationen bekommen [V-8]	USB-Verbindung wird von einem böswilligen Angreifer benutzt, um Schadsoftware zu installieren, die die persönlichen Daten der Passagiere überwacht [V-7, V-9]	Nicht unsicher	Nicht unsicher
Passagierkommunikation	Passagierkommunikation wird von einem böswilligen Angreifer blockiert, sodass autorisierte Nutzer keinen Zugriff auf ihre Informationen haben [V-8]	Falsche Passagierkommunikation wird von einem böswilligen Angreifer injiziert, sodass die Software persönliche Information an den Angreifer liefert [V-7, V-9]	Nicht unsicher	Nicht unsicher
Kartendaten	Kartendaten werden von einem böswilligen Angreifer blockiert, sodass die Software veraltete Karten verwendet [V-1, V-2, V-4]	Gefälschte Kartendaten werden von einem böswilligen Angreifer injiziert, sodass die Software falsche und gefährliche Karten verwendet [V-1, V-2, V-4, V-9]	Nicht unsicher	Nicht unsicher

Tabelle 5.6: Unsichere Kontrollaktionen

5.3.2.3 Dazugehörige Sicherheitsauflagen

ID	Unsichere Kontrollaktion	Dazugehörige Sicherheitsauflage
UCA1.1	Das Steuersignal wird von einem böswilligen Angreifer blockiert, obwohl das Fahrzeug den Kurs wechseln sollte	Das Steuersignal darf niemals blockiert werden, wenn das Fahrzeug den Kurs wechseln möchte
UCA1.2	Ein Steuersignal wird von einem böswilligen Angreifer injiziert, während das Fahrzeug auf einer geraden Strecke fährt	Es darf niemals ein falsches Steuersignal injiziert werden, während das Fahrzeug auf einer geraden Strecke fährt
UCA1.3	Steuersignal wird von einem böswilligen Angreifer so manipuliert, dass es gesendet wird, bevor das Fahrzeug die von den Umweltdaten gemeldete Kurve erreicht	Das Steuersignal darf niemals so manipuliert werden, dass es gesendet wird, bevor aktuelle Umweltdaten angekommen sind
UCA1.4	Das Steuersignal wird von einem böswilligen Angreifer unterbrochen, während sich das Fahrzeug in einer Kurve befindet	Das Steuersignal darf niemals unterbrochen werden, während sich das Fahrzeug in einer Kurve befindet
UCA1.5	Das Bremssignal wird von einem böswilligen Angreifer blockiert, während das Fahrzeug sich einem stationären oder langsameren Objekt nähert	Das Bremssignal darf niemals blockiert werden, während das Fahrzeug sich einem langsameren oder stationären Objekt nähert
UCA1.6	Ein Bremssignal wird von einem böswilligen Angreifer injiziert, obwohl sich vor dem Fahrzeug kein Objekt in einer gefährlichen Distanz befindet, sodass das Fahrzeug für den nachfolgenden Verkehr unberechenbar bremst	Es darf niemals ein falsches Bremssignal injiziert werden, wenn sich hinter dem Fahrzeug nachfolgender Verkehr befindet
UCA1.7	Das Bremssignal wird von einem böswilligen Angreifer verzögert, sodass es sich einem stationären oder langsameren Objekt in einer immer noch gefährlichen Geschwindigkeit nähert	Das Bremssignal darf niemals verzögert werden, wenn das Fahrzeug sich noch nicht an die Geschwindigkeit eines stationären oder langsameren Objekts vor dem Fahrzeug angepasst hat

UCA1.8	Das Bremssignal wird durch einen böswilligen Angreifer zu früh gestoppt, während es sich einem stationären oder langsameren Objekt in einer immer noch gefährlichen Geschwindigkeit nähert	Das Bremssignal darf niemals zu früh gestoppt werden, wenn das Fahrzeug sich noch nicht an die Geschwindigkeit eines vor ihm befindlichen stationären oder langsameren Objekts angepasst hat
UCA1.9	Das Beschleunigungssignal wird von einem böswilligen Angreifer geblockt, obwohl das Fahrzeug beschleunigen sollte und somit für den nachfolgenden Verkehr unberechenbar handelt	Das Beschleunigungssignal darf niemals blockiert werden, wenn das Fahrzeug beschleunigen sollte
UCA1.10	Ein Beschleunigungssignal wird von einem böswilligen Angreifer injiziert, obwohl das Fahrzeug still steht und sich ein stationäres Objekt vor dem Fahrzeug befindet	Es darf niemals ein Beschleunigungssignal injiziert werden, wenn das Fahrzeug still steht und sich ein stationäres Objekt vor dem Fahrzeug befindet
UCA1.11	Das Beschleunigungssignal wird von einem böswilligen Angreifer so manipuliert, dass es ausgeführt wird, bevor überprüft wurde, ob sich ein Objekt vor dem Fahrzeug befindet	Das Beschleunigungssignal darf niemals manipuliert werden, dass es ausgeführt wird, obwohl noch nicht überprüft wurde, ob sich ein Objekt vor dem Fahrzeug befindet
UCA1.12	Das Beschleunigungssignal wird von einem böswilligen Angreifer so manipuliert, dass es zu lang angewandt wird und das Fahrzeug sich einem langsameren oder stationären Objekt gefährlich nähert	Das Beschleunigungssignal darf niemals so manipuliert werden, dass es zu lange angewandt wird, während das Fahrzeug sich einem langsameren oder stationären Objekt gefährlich nähert
UCA1.13	Geschwindigkeitsdaten werden von einem böswilligen Angreifer geblockt, sodass die Software von einer veralteten Geschwindigkeit ausgeht, obwohl die Geschwindigkeit verändert wurde	Geschwindigkeitsdaten dürfen niemals blockiert werden, wenn die Geschwindigkeit verändert wurde
UCA1.14	Geschwindigkeitsdaten, die eine zu langsame Geschwindigkeit angeben, werden von einem böswilligen Angreifer injiziert, sodass das Fahrzeug weiter beschleunigt	Es dürfen niemals Geschwindigkeitsdaten, die eine zu langsame Geschwindigkeit angeben, injiziert werden

UCA1.15	Geschwindigkeitsdaten werden von einem böswilligen Angreifer so verzögert, dass sie nach dem Beschleunigungssignal bei der Software ankommen und das Beschleunigungssignal mit veralteten Daten berechnet wurde	Geschwindigkeitsdaten dürfen niemals so verzögert werden, dass sie nach dem Beschleunigungssignal ankommen
UCA1.16	Motordaten werden von einem böswilligen Angreifer blockiert, obwohl der Motor mit einer zu hohen Drehzahl operiert und der Gang gewechselt werden sollte	Motordaten dürfen nie blockiert werden, wenn der Motor mit einer zu hohen Drehzahl operiert
UCA1.17	Motordaten, die angeben, dass die Drehzahl des Motors in einem sicheren Bereich ist, werden von einem böswilligen Angreifer injiziert, sodass die Software kein Signal zum Gang wechseln sendet	Es dürfen niemals Motordaten, die eine Drehzahl in einem sicheren Bereich angeben, injiziert werden
UCA1.18	Umweltdaten werden von einem böswilligen Angreifer geblockt, sodass die Software keine Information über ein sich vor dem Fahrzeug befindliches Objekt erhält	Umweltdaten dürfen niemals blockiert werden, wenn sich vor dem Fahrzeug ein Objekt befindet
UCA1.19	Ein böswilliger Angreifer injiziert Umweltdaten, die kein Objekt vor dem Fahrzeug anzeigen, obwohl sich dort eines befindet	Es dürfen niemals Umweltdaten, die fälschlicherweise ein Objekt vor dem Fahrzeug anzeigen, injiziert werden
UCA1.20	Umweltdaten werden von einem böswilligen Angreifer so verzögert, dass sie erst eintreffen, nachdem ein Beschleunigungssignal gegeben wurde, sodass es mit alten Daten berechnet wurde, die kein Objekt vor dem Fahrzeug angeben, obwohl sich dort eines befindet	Umweltdaten dürfen niemals so verzögert werden, dass sie nach dem Beschleunigungssignal eintreffen
UCA1.21	USB-Verbindung wird von einem böswilligen Angreifer blockiert, sodass autorisierte Nutzer keinen Zugriff auf ihre Informationen bekommen	Die USB-Verbindung darf niemals blockiert werden, wenn ein autorisierter Nutzer auf seine Informationen zugreifen will

UCA1.22	USB-Verbindung wird von einem böswilligen Angreifer benutzt, um Schadsoftware zu installieren, die die persönlichen Daten der Passagiere überwacht	Die USB-Verbindung darf niemals verwendet werden, um persönliche Daten überwachende Schadsoftware zu installieren
UCA1.23	Passagierkommunikation wird von einem böswilligen Angreifer blockiert, sodass autorisierte Nutzer keinen Zugriff auf ihre Informationen haben	Passagierkommunikation darf niemals blockiert werden, wenn autorisierte Nutzer auf ihre Informationen zugreifen wollen
UCA1.24	Falsche Passagierkommunikation wird von einem böswilligen Angreifer injiziert, sodass die Software persönliche Information an den Angreifer liefert	Es darf niemals falsche Passagierkommunikation injiziert werden, sodass persönliche Informationen an einen Angreifer geliefert werden
UCA1.25	Kartendaten werden von einem böswilligen Angreifer blockiert, sodass die Software veraltete Karten verwendet	Kartendaten dürfen niemals blockiert werden, wenn die Karten, die die Software verwendet, veraltet sind
UCA1.26	Gefälschte Kartendaten werden von einem böswilligen Angreifer injiziert, sodass die Software falsche und gefährliche Karten verwendet	Es dürfen niemals gefälschte und gefährliche Kartendaten injiziert werden

Tabelle 5.7: Dazugehörende Sicherheitsauflagen

5.3.3 Kausale Analyse

5.3.3.1 Kontrollstruktur mit Prozessmodell

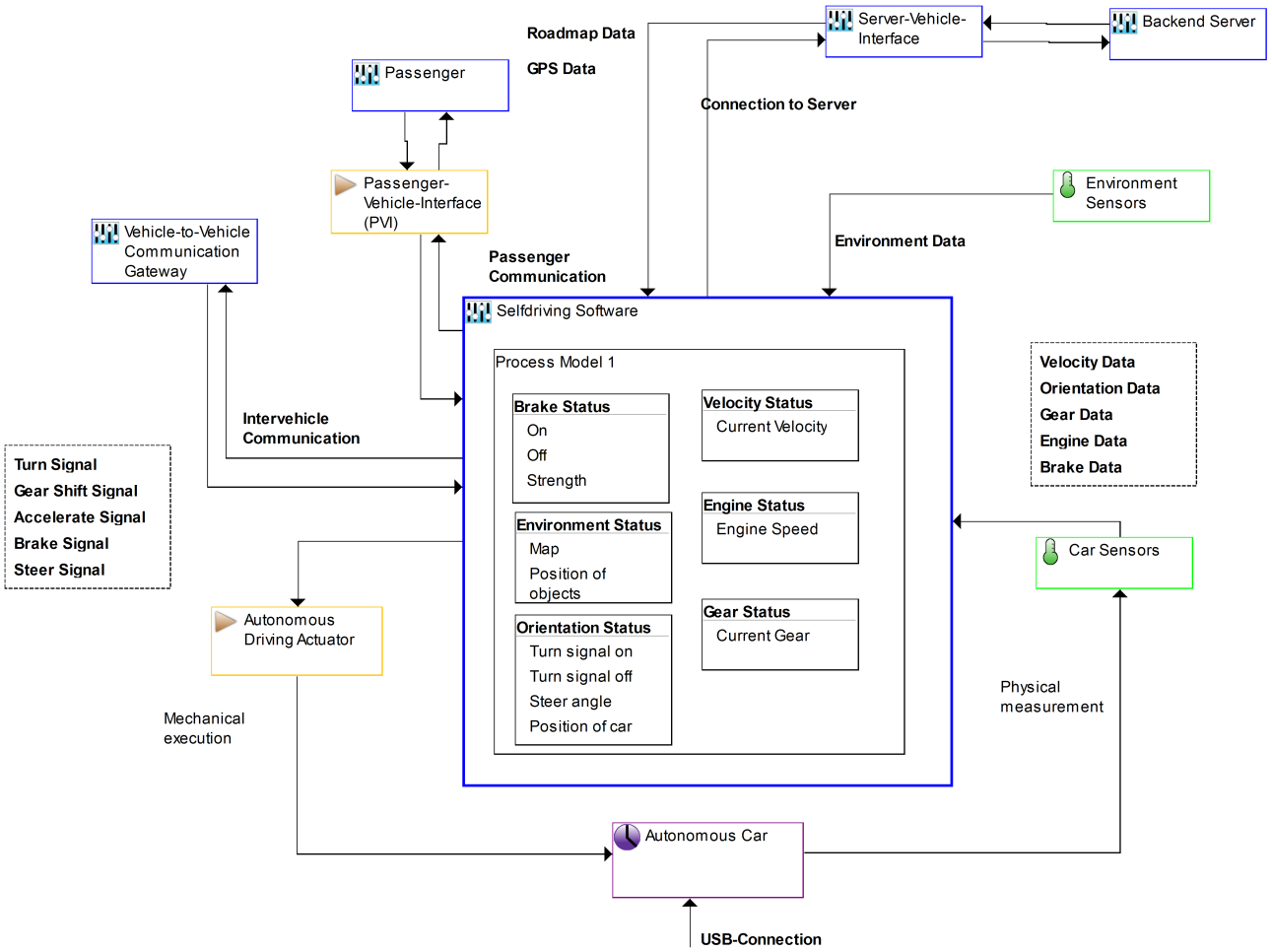


Abbildung 5.2: Kontrollstruktur mit Prozessmodell eines autonomen Fahrzeugs

5.3.3.2 Tabelle der kausalen Faktoren

Komponente	Kausaler Faktor	Unsichere Kontrollaktion	Verwundbarkeit	Kausales Szenario	Sicherheitsauflage
Server-Vehicle-Interface	Senden von manipuliertem Input	Gefälschte Kartendaten werden von einem böswilligen Angreifer injiziert, sodass die Software falsche und gefährliche Karten verwendet	V-1, V-2, V-4, V-9	Ein böswilliger Angreifer bekommt Zugriff zum Server und sendet gefälschte Kartendaten an das Interface	Ein böswilliger Angreifer darf niemals Zugriff auf den Server bekommen und gefälschte Kartendaten senden können
				Ein böswilliger Angreifer verwendet eine DDoS-Attacke, um das Interface zu überlasten, sodass dieses keine Daten mehr empfangen kann	Das Interface darf niemals durch eine DDoS-Attacke überlastet werden
	Verstopfte Kontrolleingaben	Kartendaten werden von einem böswilligen Angreifer blockiert, sodass die Software veraltete Karten verwendet	V-1, V-2, V-4	Ein böswilliger Angreifer verwendet eine DDoS-Attacke, um das Interface zu überlasten, sodass dieses keine Daten mehr empfangen kann	Das Interface darf niemals durch eine DDoS-Attacke überlastet werden

Environment Sensors	Ersetzen von Sensoren	Umweltdaten werden von einem böswilligen Angreifer so verzögert, dass sie erst eintreffen, nachdem ein Beschleunigungssignal gegeben wurde, sodass es mit alten Daten berechnet wurde, die kein Objekt vor dem Fahrzeug angeben, obwohl sich dort eines befindet	V-1	Ein böswilliger Angriff bekommt Zugriff auf die Sensoren und ersetzt sie mit welchen, die das Senden von Messdaten verzögern	Das System muss immer erkennen, wenn Sensoren durch falsche Sensoren ersetzt wurden
	Manipulierte Operation	Umweltdaten werden von einem böswilligen Angreifer geblockt, sodass die Software keine Information über ein sich vor dem Fahrzeug befindliches Objekt erhält	V-1	Ein böswilliger Angreifer bekommt Zugriff auf die Sensoren und beschädigt sie so, dass sie keine Informationen mehr senden können	Beschädigte Environment Sensors müssen immer vom System erkannt werden, wenn sie keine Informationen mehr senden können
Passenger-Vehicle-Interface	Senden von manipuliertem Input	Falsche Passagierkommunikation wird von einem böswilligen Angreifer injiziert, sodass die Software persönliche Information an den Angreifer liefert	V-7, V-9	Ein böswilliger Angreifer gibt sich fälschlicherweise als Passagier aus, schickt eine Anfrage nach persönlichen Informationen, welche das System beantwortet	Das Interface darf niemals unautorisierte Nutzer als Passagiere anerkennen und diesen persönliche Informationen über echte Passagiere liefern

	Überschreiben von legitimem Input	Falsche Passagierkommunikation wird von einem böswilligen Angreifer injiziert, sodass die Software persönliche Information an den Angreifer liefert	V-7, V-9	Ein böswilliger Angreifer nutzt eine Man-in-the-Middle-Attacke, um die Kommunikation zwischen Interface und Software abzufangen und so abzuändern, dass ihm persönliche Informationen gesendet werden	Die Kommunikation zwischen Interface und Software darf niemals abgefangen und abgeändert werden
Self-driving Software	Injektion eines manipulierten Algorithmus	Ein Steuersignal wird von einem böswilligen Angreifer injiziert, während das Fahrzeug auf einer geraden Strecke fährt	V-2, V-9	Ein böswilliger Angreifer bekommt Zugriff zur Kontrollsoftware und überschreibt den korrekten Algorithmus mit einem, welcher ein falsches Steuersignal berechnet	Ein böswilliger Angreifer darf niemals in der Lage sein, Zugriff auf die Kontrollsoftware zu bekommen und den korrekten Algorithmus zu überschreiben
Car Sensors	Absichtliche Verstopfung des Rückmeldungskanals	Geschwindigkeitsdaten werden von einem böswilligen Angreifer geblockt, sodass die Software von einer veralteten Geschwindigkeit ausgeht, obwohl die Geschwindigkeit verändert wurde	V-1	Ein böswilliger Angreifer bekommt Zugriff auf das Auto und installiert ein Gerät, welches den Kommunikationskanal zwischen den Car Sensors und der Software so stört, dass Signale nicht mehr bei der Software ankommen	Der Kommunikationskanal zwischen den Car Sensors der Software darf niemals so gestört werden, dass Signale nicht mehr durchkommen

Autonomous Car	Physische Manipulation	USB-Verbindung wird von einem böswilligen Angreifer blockiert, sodass autorisierte Nutzer keinen Zugriff auf ihre Informationen bekommen	V-8	Ein böswilliger Angreifer bekommt Zugriff auf den USB-Port und beschädigt diesen	Der USB-Port darf niemals beschädigt werden

Tabelle 5.8: Kausale Faktoren

5.4 Ergebnis der STPA-Sec Analyse

Die Analyse hat viele verschiedene unsichere Kontrollaktionen und kausale Szenarien hervorgebracht. Dabei handelt es sich sowohl um Szenarien, die durch die Unsicherheit einer Komponente auftreten können, als auch um Szenarien, die durch die Kombination mehrerer Komponenten zu einer Verwundbarkeit führen. Jedoch ist diese hohe Anzahl nur bedingt sinnvoll, manche unsicheren Kontrollaktionen teilen sich die gleichen Szenarien. Bei der Analyse orientiert man sich hauptsächlich an der Kontrollstruktur, da diese eine funktionale Darstellung ist, ist es schwieriger, technische Schwächen des Systems aufzudecken. Dies gilt insbesondere für die Information-Security, da manche Fehler erst bei der Betrachtung des Systems auf der Datenfluss-Ebene aufgedeckt werden. Man benötigt sehr viel Erfahrung im Security-Engineering, um alle Verwundbarkeiten aufzudecken. Das Plugin selbst hat sich bei der Analyse als sehr wertvoll erwiesen, da es systematisch durch die Schritte der Methode führt und gleichzeitig Listen und Tabellen bereitstellt, in die man nur noch die Information eintragen muss. Positiv ist auch, dass man die Ergebnisse der einzelnen Schritte nicht nur als PDF, sondern auch als Bild und auch als CSV-Datei exportieren kann, da dies das Einbinden in andere Dokumente erleichtert.

6 Verwendung des STPA-Sec Plugins

In diesem Kapitel wird beschrieben, wie das STPA-Sec Plugin verwendet werden kann. Zunächst wird gezeigt, wie man das Plugin auf der XSTAMPP-Plattform installiert und danach einige Schritte des Plugins beschrieben.

6.1 Installation

Um das Plugin auf der XSTAMPP-Plattform zu installieren, muss das Plugin zunächst heruntergeladen werden. Die benötigte **updatesite.zip**-Datei findet man auf https://sourceforge.net/projects/stpasec-stpa-for-security/files/. Dann öffnet man XSTAMPP, geht dort auf **Help** und wählt **Install New Software** aus. Daraufhin öffnet sich das Fenster 6.1. Dort geht man auf **Add...**, klickt auf **Archive...** und wählt dort über das Dateiauswahlfenster die heruntergeladene **updatesite.zip**-Datei aus. Es tauchen zwei Einträge in dem Fenster auf, welche man beide auswählt und auf **Next >** klickt. Im nächsten Fenster akzeptiert man die Lizenzbedingungen und bestätigt das während der Installation auftauchende Fenster. Im letzten Schritt akzeptiert man den Neustart von XSTAMPP.

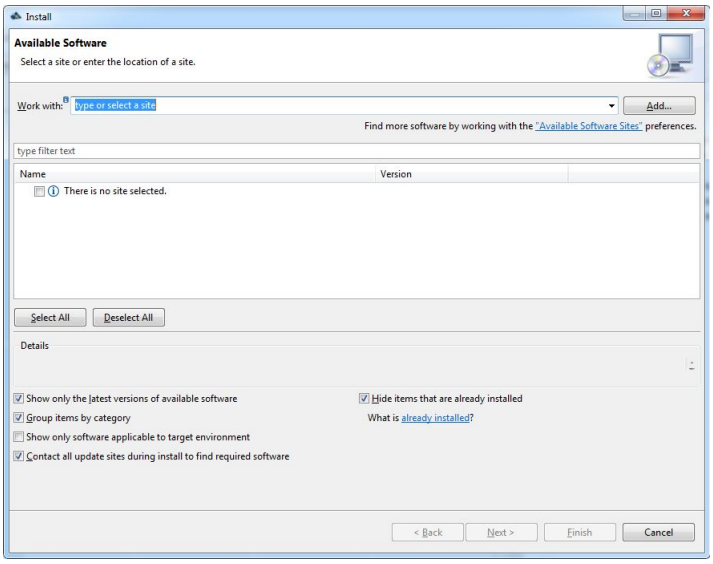


Abbildung 6.1: Installationsfenster von XSTAMPP

6.2 Verwendung

Um ein neues Projekt zu erstellen, klickt man auf **Data** und auf **New Project**. Im Fenster in Abbildung 6.2 wählt man **STPA for Security Project** aus und klickt auf **Next**. Dort gibt man den Namen des Projekts ein, wählt einen Speicherort aus und klickt auf **Finish**.

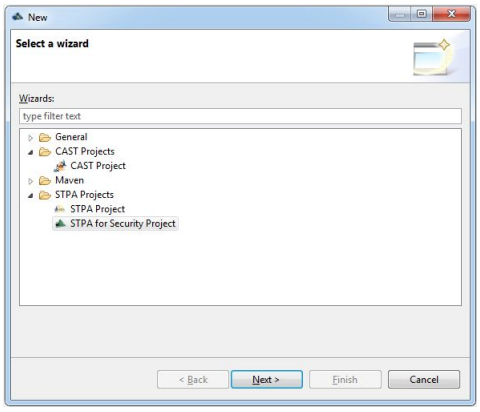


Abbildung 6.2: Erstellen eines neuen Projekts

Nachdem man ein neues Projekt erstellt hat, muss man zunächst das System beschreiben. Dazu gibt es ein Texteingabefenster (Abbildung 6.3) mit grundlegenden Werkzeugen zur Textbearbeitung, beispielsweise zur Anpassung der Größe der Worte oder zum Unterstreichen im Text.

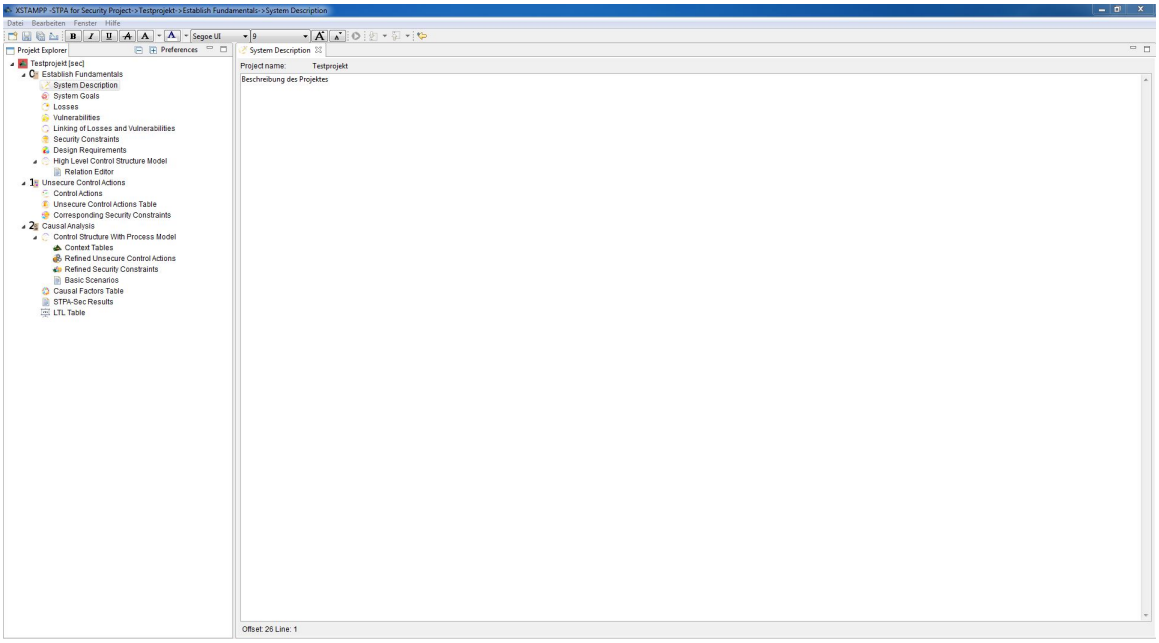


Abbildung 6.3: Beschreiben des Projekts

Die Fenster der Schritte *System Goals*, *Losses*, *Vulnerabilities*, *Security Constraints* und *Design Requirements* funktionieren alle gleich, darum wird hier beispielshaft das *Losses*-Fenster (Abbildung 6.4) erklärt. In diesem werden die Verluste bearbeitet. Mit dem Plus-Knopf unten links wird ein neuer Verlust hinzugefügt, mit dem X-Knopf rechts daneben wird ein Verlust wieder gelöscht und mit dem Doppel-X-Knopf rechts daneben werden alle Verluste entfernt. Die Beschreibung des Verlustes selbst lässt mit einem Klick darauf markieren und verändern. Rechts kann man eine ausführlichere Beschreibung des Verlustes hinzufügen. Mit dem Filterfenster oben in der Mitte kann man nach die Verluste nach einem eingegebenen Begriff filtern.

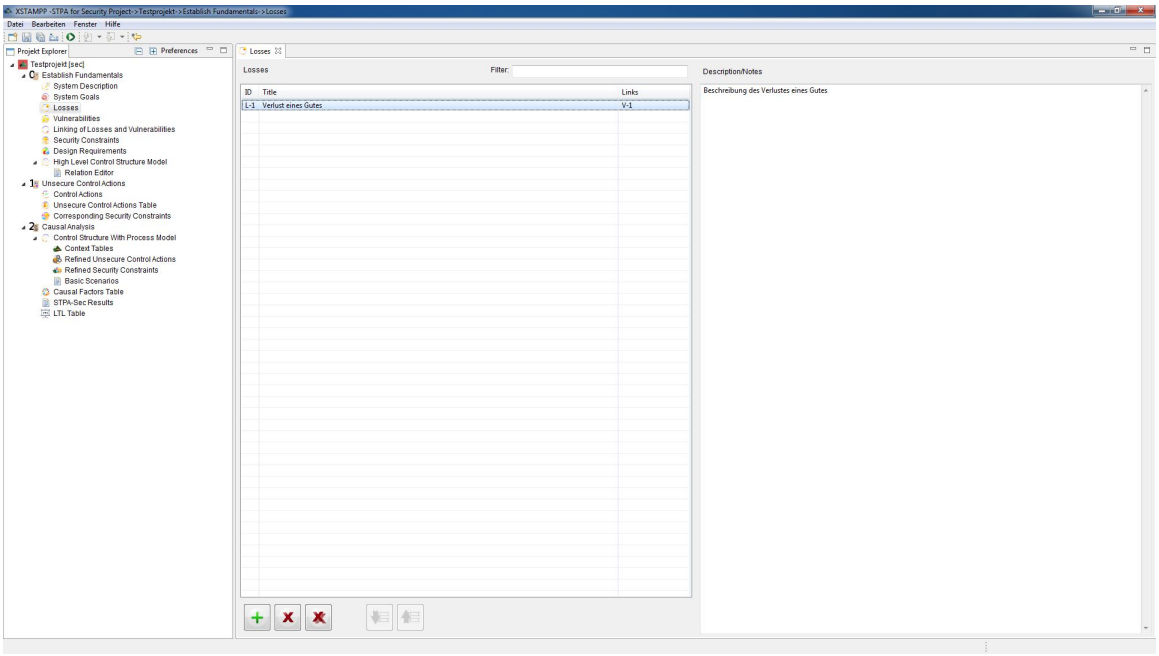


Abbildung 6.4: Hinzufügen von Verlusten

Im Schritt Linking of Losses and Vulnerabilities werden die Verluste und die Verwundbarkeiten verbunden. Man kann dabei mit dem Knopf **Switch to Vulnerabilities** auswählen, ob man die Verluste anzeigt und die Verwundbarkeiten zuordnet oder die Verwundbarkeiten anzeigt und die Verluste zuordnet. Zugeordnete Attribute lassen sich auch wieder entfernen.

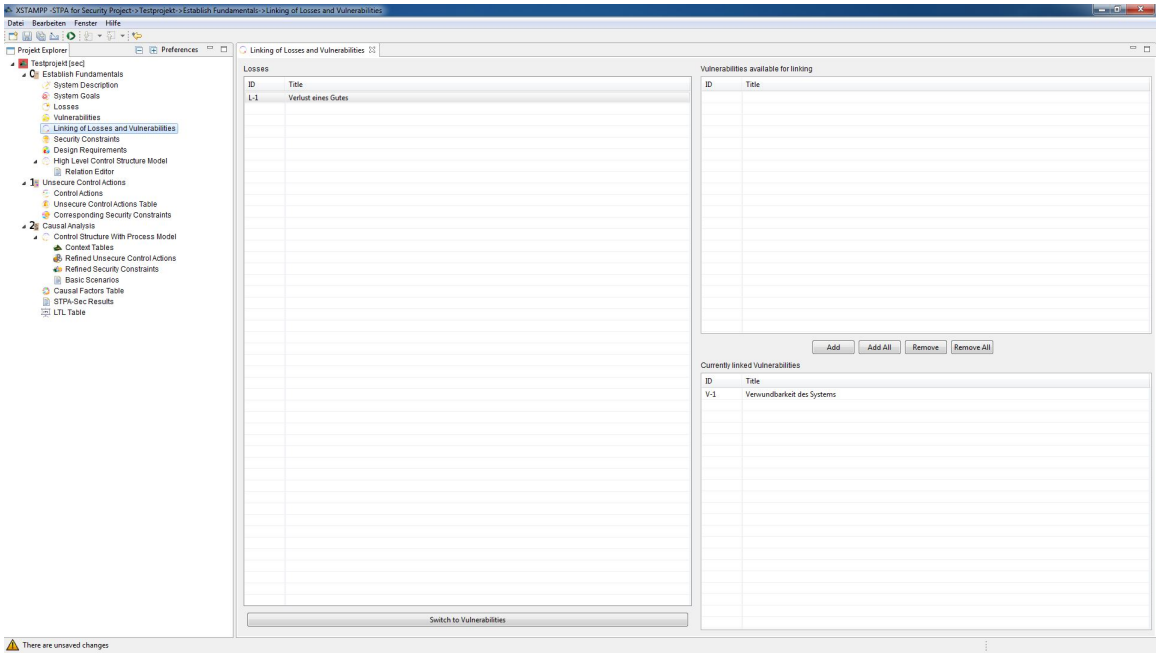


Abbildung 6.5: Verbinden von Verwundbarkeiten und Verlusten

Als nächstes wird die Kontrollstruktur erstellt. Man hat alle möglichen Komponenten zur Verfügung die man mittels Klick auswählen und platzieren kann. Die Komponenten lassen sich mit Pfeilen verbinden und den Pfeilen lassen sich Kontrollaktionen zuordnen. Außerdem kann man Textboxen als Kommentare setzen.

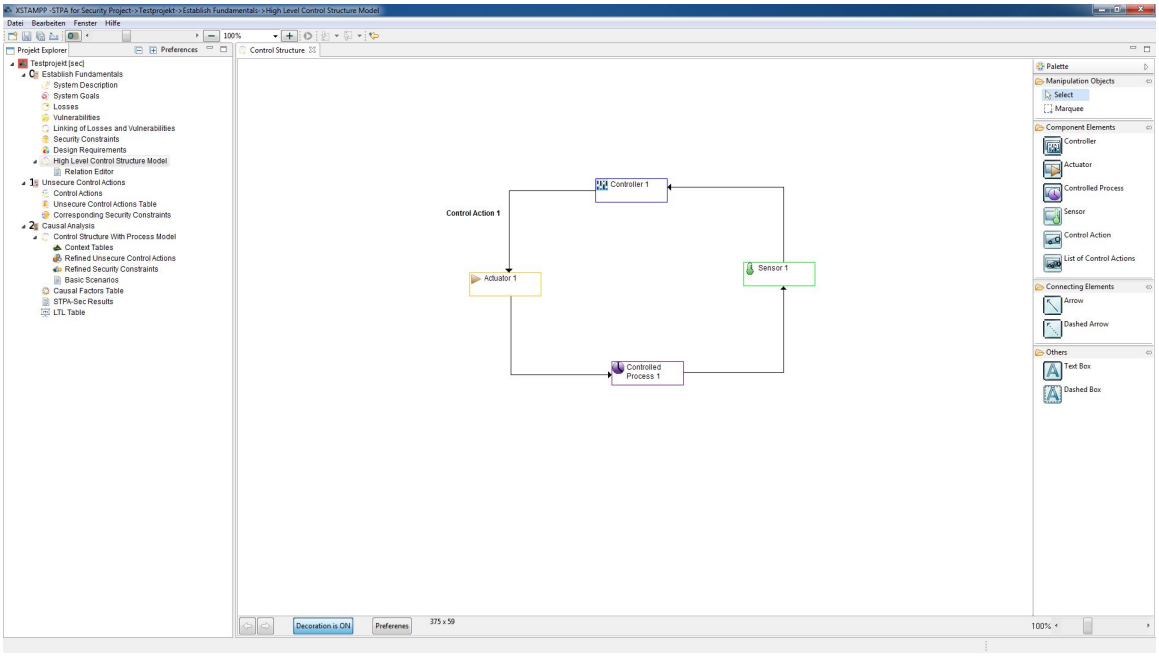


Abbildung 6.6: Erstellen der Kontrollstruktur

Im Schritt *Unsecure Control Actions Table* werden die unsicheren Kontrollaktionen identifiziert. Mit einem Klick auf das Plus wird eine neue unsichere Kontrollaktion erstellt, mit einem Klick auf das Textfeld lässt sich die Beschreibung dieser verändern. Sie lässt sich mit einem Klick auf den X-Knopf auch wieder löschen und mit dem blauen Knopf lässt sich der unsicheren Kontrollaktion eine Verwundbarkeit hinzufügen. Oben gibt es auch einen Filter, mit dem sich die Liste nach bestimmten Begriffen filtern lassen kann.

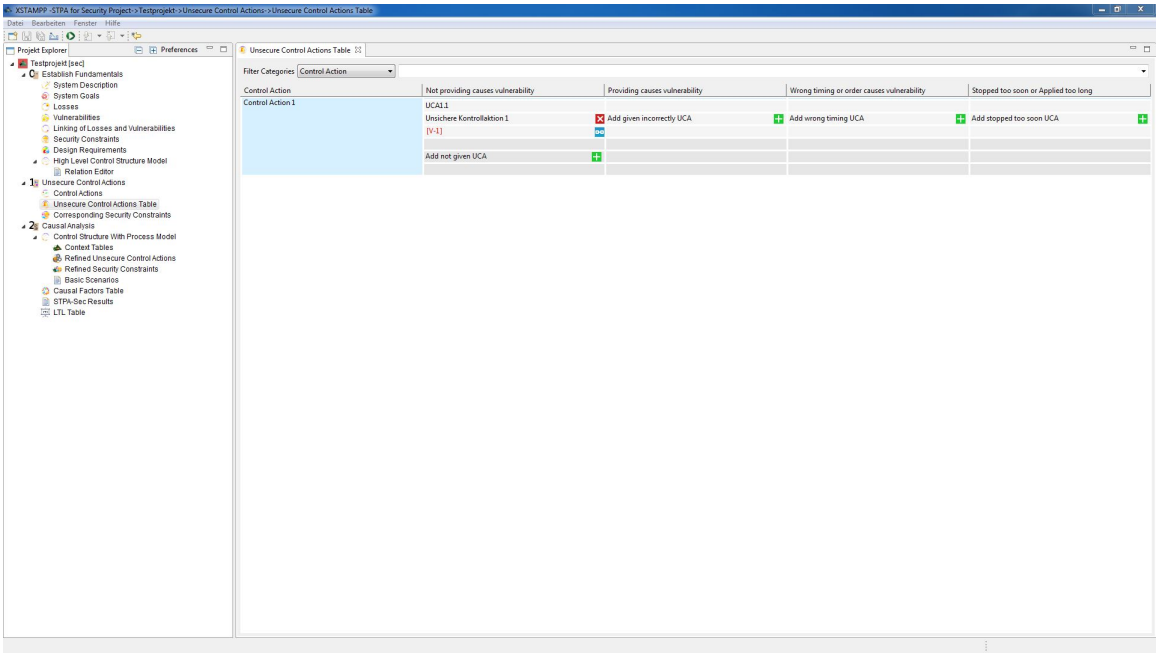


Abbildung 6.7: Identifizieren von unsicheren Kontrollaktionen

Wenn man die unsicheren Kontrollaktionen identifiziert hat, werden diesen Sicherheitsauflagen zugeordnet. Dazu gibt es eine Liste aller unsicheren Kontrollaktionen, zu denen man rechts über ein Textfeld die Auflagen hinzufügen kann. Man kann die Liste auch filtern.

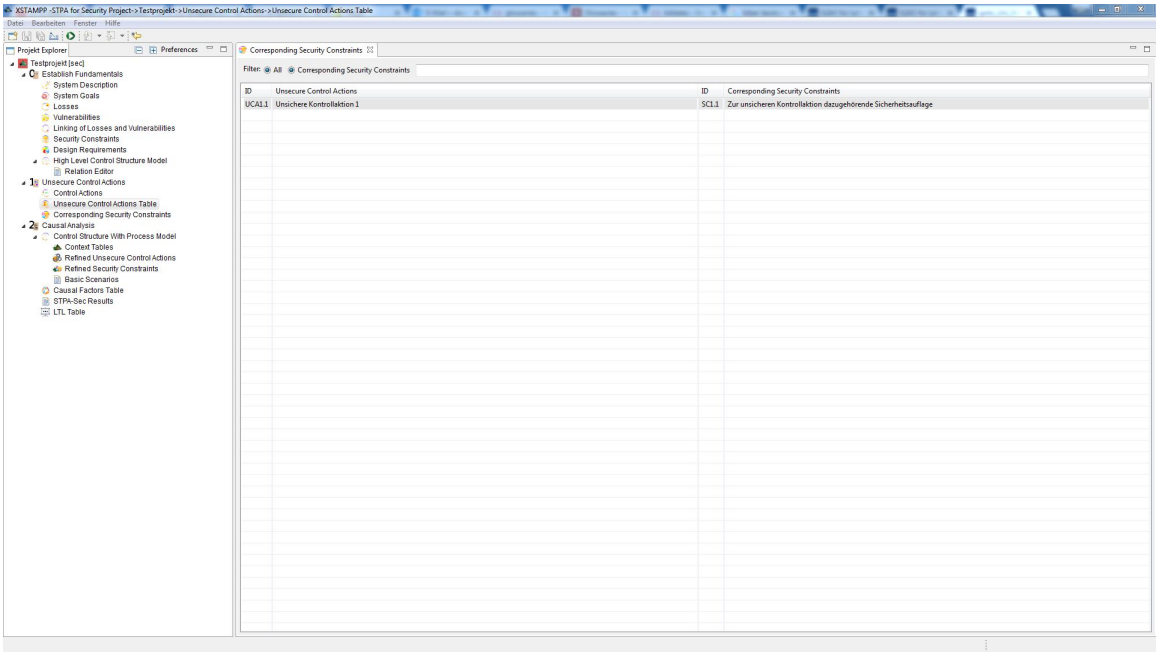


Abbildung 6.8: Identifizieren Sicherheitsauflagen für unsichere Kontrollaktionen

Als nächstes werden die unsicheren Kontrollaktionen als "unsicher" im Sinne der Security oder als "unsicher" im Sinne der Safety markiert. Man kann auch hier nochmals die Beschreibung der Kontrollaktion verändern.

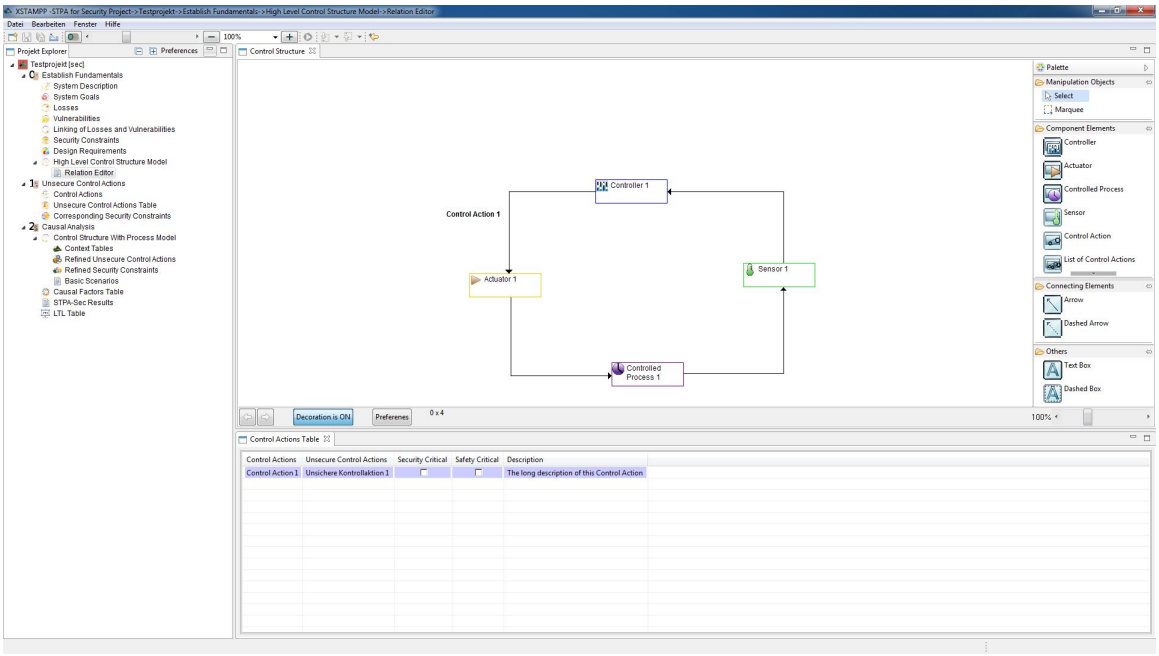


Abbildung 6.9: Markieren der unsicheren Kontrollaktionen

In dem Fenster in Abbildung 6.10 wird das Prozessmodell des Controllers erstellt. Das Hinzufügen von Objekten funktioniert wie in der Kontrollstruktur.

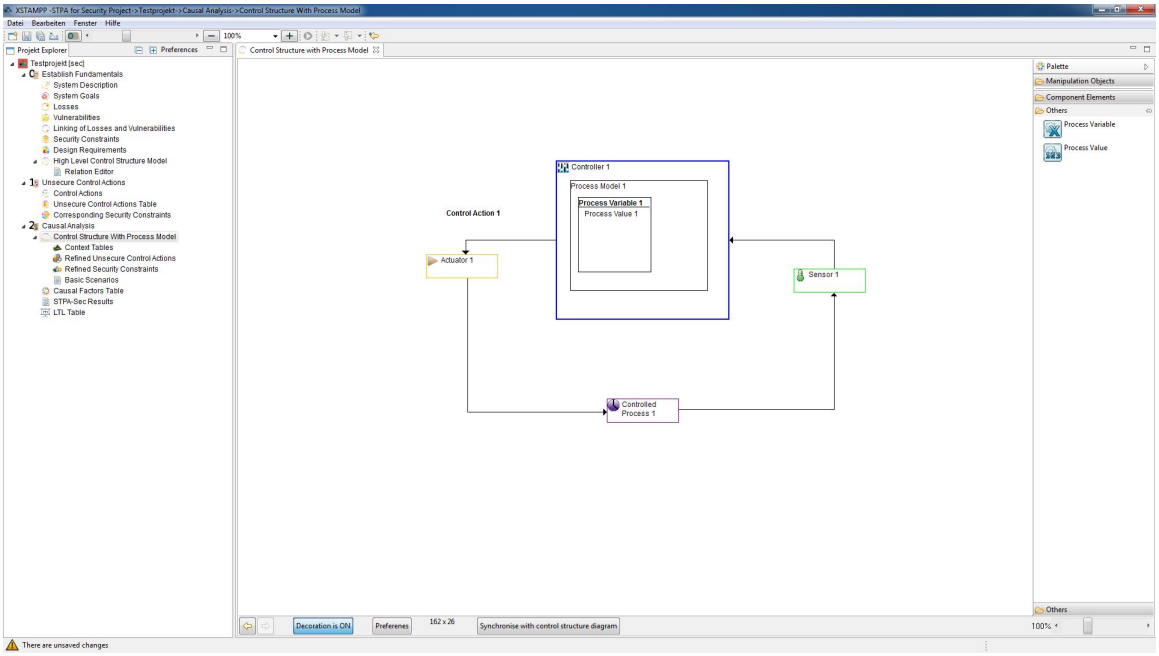


Abbildung 6.10: Erstellen des Prozessmodells

Der nächste Schritt identifiziert die kausalen Szenarios. In der Liste werden alle Komponenten der Kontrollstruktur angezeigt. Diesen kann man kausale Faktoren hinzufügen, die wiederum mit den unsicheren Kontrollaktionen verbunden werden. Den unsicheren Kontrollaktionen kann man kausale Szenarios hinzufügen mit deren Hilfe man verfeinerte Sicherheitsauflagen identifizieren kann. Man kann den unsicheren Kontrollaktionen noch Notizen hinzufügen. Auch lässt sich die Liste hier nach Begriffen filtern.

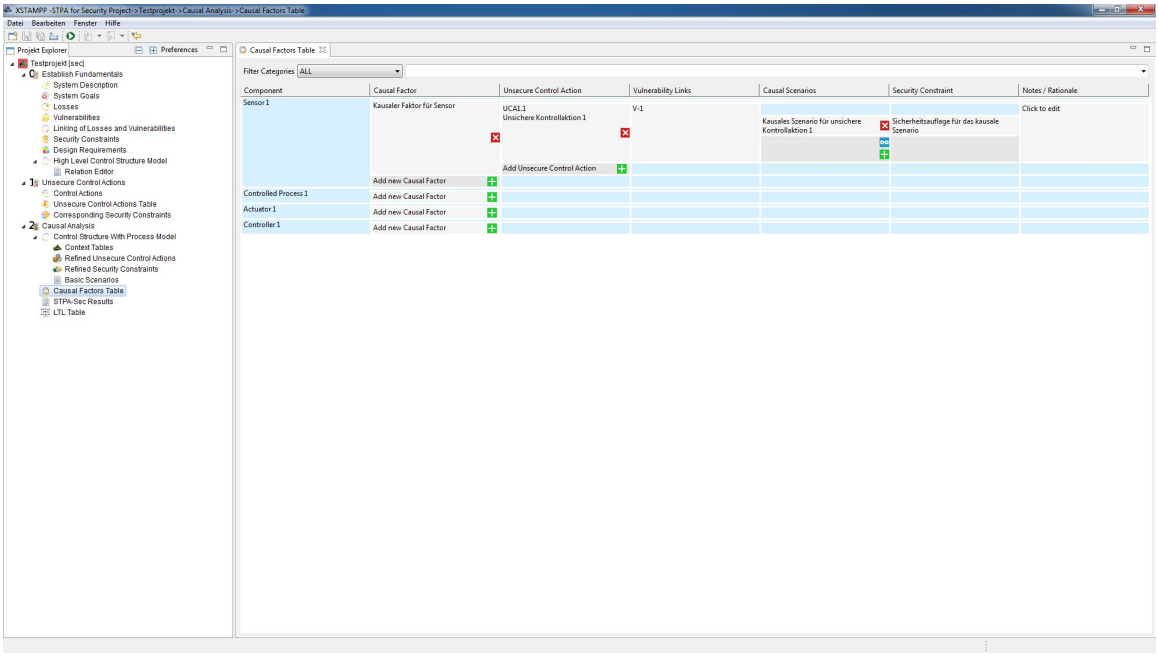


Abbildung 6.11: Identifizieren von kausalen Szenarios

Im letzten Schritt werden alle Sicherheitsauflagen angezeigt. Diese werden nach den Schritten aufgeteilt, in denen sie identifiziert wurden. Die Sicherheitsauflagen können hier als "unsicher" im Sinne der Security oder als "unsicher" im Sinne der Safety markiert werden und nach diesen Attributen gefiltert werden. Wenn man einen Eintrag markiert hat und auf den X-Knopf unten klickt, lässt sich der Eintrag löschen.

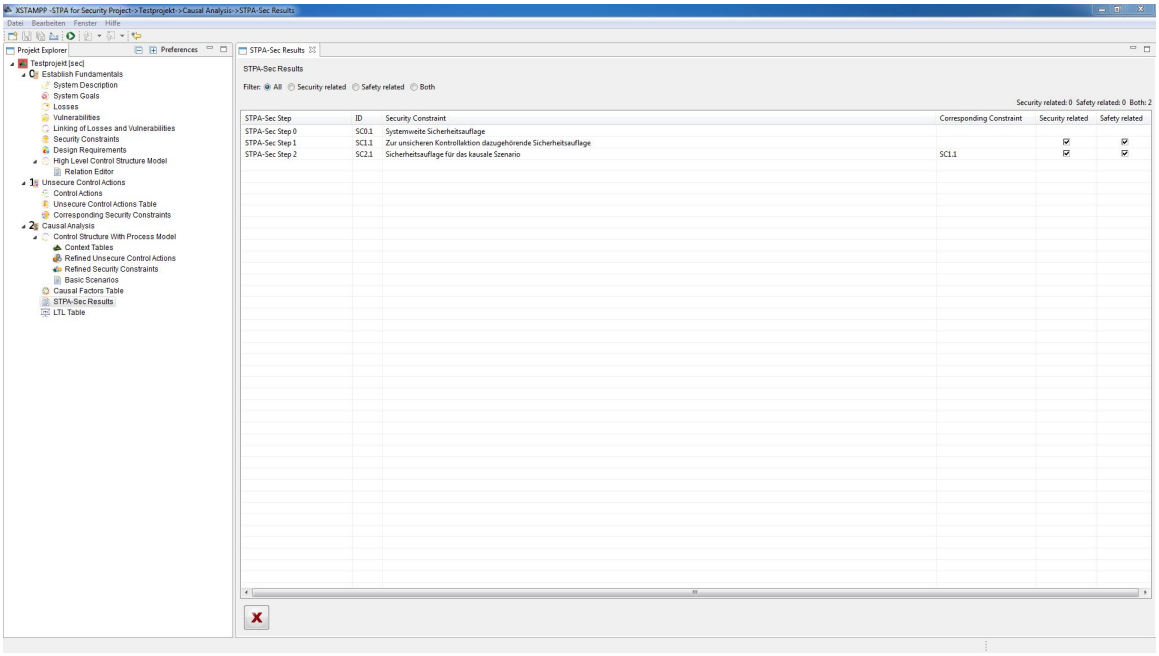


Abbildung 6.12: Markieren der Sicherheitsauflagen

Wenn man mit der Durchführung fertig ist, lässt sich das Projekt exportieren. Dazu klickt man mit rechts auf das Projekt und wählt **Export** aus. Daraufhin öffnet sich das Fenster 6.13. In diesem kann man auswählen, ob man das ganze Projekt oder nur einen einzelnen Schritt des Projekts exportieren kann.

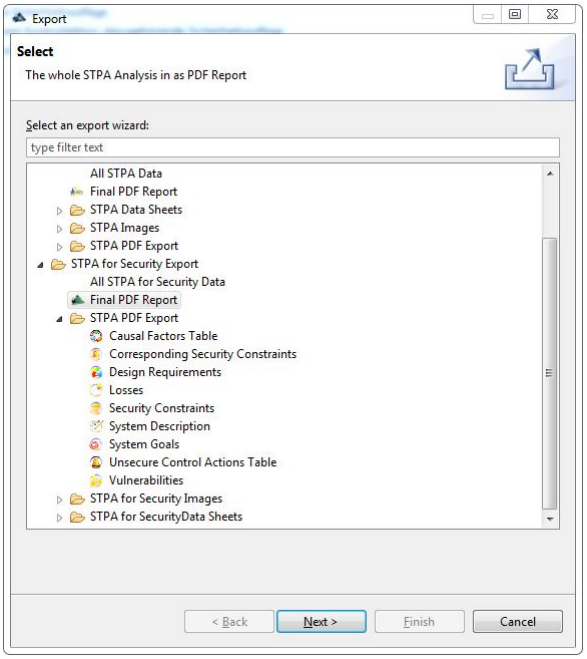


Abbildung 6.13: Exportfenster

Beispielshaft wird in Abbildung 6.14 der Export des ganzen Projekts als PDF angezeigt. Man kann den Namen eines Unternehmens sowie ein Logo hinzufügen. Man kann auch die Farbgestaltung der exportierten PDF ändern, sowie die Größe des Textes, die Ausrichtung der PDF und die Anzeige der Dekoration der Kontrollstrukturkomponenten. Letztlich wählt man einen Speicherort aus und klickt auf **Finish**.

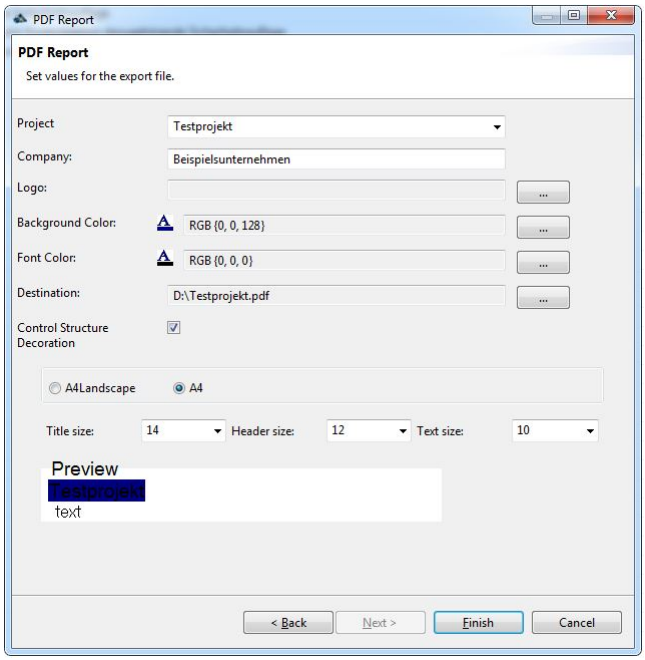


Abbildung 6.14: Export des Projekts als PDF

7 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde das Thema STPA-Sec und die dafür benötigten Grundlagen erklärt. Danach wurde STPA-Sec mit anderen bereits existierenden und weitläufig verwendeten Methoden zur Security-Analyse verglichen. Außerdem wurde in dieser Arbeit das STPA-Sec-Plugin, aufbauend auf dem bereits existierenden A-STPA-Plugin entwickelt. Abschließend wurde das Plugin und die Methode anhand des aktuellen Beispiels der autonomen Fahrzeuge ausführlich getestet.

Fazit

Anhand des Vergleichs und des Beispiels wurde gezeigt, dass STPA-Sec ein vielversprechender Ansatz zur Security-Analyse ist, der jedoch noch einige Schwächen hat. In Kapitel 4 wurde beschrieben,wie A-STPA aufgebaut ist und wie man ein Plugin auf Basis von A-STPA baut. Weiterhin wurde in Kapitel 5 gezeigt, dass sich mit dem STPA-Sec-Plugin alle Schritte der STPA-Sec-Methode erfolgreich durchführen lassen.

Ausblick

Zu STPA-Sec als Methode gibt es noch viel Forschungsbedarf, es ist jedoch abzusehen, dass STPA-Sec in Zukunft einige Relevanz haben wird für Security-Analysen. Das STPA-Sec-Plugin stellt ein nützliches Werkzeug dar, um die Analyse durchzuführen und funktioniert gut mit der aktuellen XSTAMPP-Architektur. Sowohl XSTAMPP als auch A-STPA befinden sich noch in der Entwicklung. Da das STPA-Sec-Plugin viele Komponenten von A-STPA direkt übernimmt, ist nicht zu erwarten, dass Probleme in der Kompatibilität zwischen dem Plugin und der Plattform auftreten, da sich Änderungen an A-STPA auch auf das STPA-Sec-Plugin auswirken. Auch das Erweitern des Plugins stellt kein Problem dar, wenn wichtige Änderungen an der STPA-Sec-Methode entwickelt werden.

Literaturverzeichnis

[Abd17] A. Abdulkhaleq. *XSTAMPP*. 2017. URL: http://www.xstampp.de/ (zitiert auf S. 32).

[AW14] A. Abdulkhaleq, S. Wagner. „Open tool support for System-Theoretic Process Analysis“. In: *2014 STAMP Workshop, MIT, Boston, USA*. 2014 (zitiert auf S. 31).

[AW15] A. Abdulkhaleq, S. Wagner. „XSTAMPP: An eXtensible STAMP platform as tool support for safety engineering“. In: *2015 STAMP Workshop, MIT, Boston, USA*. 2015 (zitiert auf S. 31).

[CSYW07] R. Caralli, J. Stevens, L. Young, W. Wilson. *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process*. Techn. Ber. CMU/SEI-2007-TR-012. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419 (zitiert auf S. 38).

[Eck] C. Eckert. *IT-Sicherheit, Konzepte - Verfahren - Protokolle*. (Zitiert auf S. 19).

[Elk06] S. Elky. „An Introduction to Information System Risk Management“. In: *SANS Institute InfoSec* (2006) (zitiert auf S. 21, 22).

[JKK+14] K. Jo, J. Kim, D. Kim, C. Jang, M. Sunwoo. „Development of Autonomous Car—Part I: Distributed System Architecture and Development Process“. In: *IEEE Trans. Ind. Electron., vol. 61, no. 12, pp. 7131-7140*. 2014 (zitiert auf S. 58).

[KCR+10] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage. „Experimental Security Analysis of a Modern Automobile“. In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. SP '10. Washington, DC, USA: IEEE Computer Society, 2010, S. 447–462. ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.34. URL: http://dx.doi.org/10.1109/SP.2010.34 (zitiert auf S. 57).

[LB07] K. de Leeuw, J. Bergstra. *The History of Information Security*. 2007 (zitiert auf S. 19, 20).

[LDDM03] N. Leveson, M. Daouk, N. Dulac, K. Marais. „A Systems Theoretic Approach to Safety Engineering“. In: *DEPT. OF AERONAUTICS AND ASTRONAUTICS, MASSACHUSETTS INST. OF TECHNOLOGY*. 2003 (zitiert auf S. 29).

[Lev11] N. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. 2011 (zitiert auf S. 23–27, 29).

[Sch99] B. Schneier. „Attack trees: Modeling security threats“. In: *Dr. Dobb's journal* (1999) (zitiert auf S. 39).

[SGF02] G. Stoneburner, A. Y. Goguen, A. Feringa. *SP 800-30. Risk Management Guide for Information Technology Systems*. Techn. Ber. Gaithersburg, MD, United States: NIST, 2002 (zitiert auf S. 36).

[SMP16] C. Schmittner, Z. Ma, P. Puschner. „Limitation and Improvement of STPA-Sec for Safety and Security Co-analysis“. In: *Computer Safety, Reliability, and Security: SAFECOMP 2016 Workshops, ASSURE, DECSoS, SASSUR, and TIPS, Trondheim, Norway, September 20, 2016, Proceedings*. Hrsg. von A. Skavhaug, J. Guiochet, E. Schoitsch, F. Bitsch. Cham: Springer International Publishing, 2016, S. 195–209. ISBN: 978-3-319-45480-1. DOI: 10.1007/978-3-319-45480-1_16. URL: http://dx.doi.org/10.1007/978-3-319-45480-1_16 (zitiert auf S. 30, 31, 49).

[TJYN11] S. Taubenberger, J. Jürjens, Y. Yu, B. Nuseibeh. „Problem Analysis of Traditional IT-Security Risk Assessment Methods – An Experience Report from the Insurance and Auditing Domain“. In: *Future Challenges in Security and Privacy for Academia and Industry: 26th IFIP TC 11 International Information Security Conference, SEC 2011, Lucerne, Switzerland, June 7-9, 2011. Proceedings*. Hrsg. von J. Camenisch, S. Fischer-Hübner, Y. Murayama, A. Portmann, C. Rieder. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 259–270. ISBN: 978-3-642-21424-0. DOI: 10.1007/978-3-642-21424-0_21. URL: http://dx.doi.org/10.1007/978-3-642-21424-0_21 (zitiert auf S. 21, 22).

[TM12] M. Talabis, J. Martin. *Information Security Risk Assessment Toolkit*. 2012 (zitiert auf S. 20, 35, 37, 38).

[Whe11] E. Wheeler. *Security Risk Management*. 2011 (zitiert auf S. 21).

[YL13] W. Young, N. Leveson. „Systems Thinking for Safety and Security“. In: *Proceedings of the 29th Annual Computer Security Applications Conference*. ACSAC '13. New Orleans, Louisiana, USA: ACM, 2013, S. 1–8. ISBN: 978-1-4503-2015-3. DOI: 10.1145/2523649.2530277. URL: http://doi.acm.org/10.1145/2523649.2530277 (zitiert auf S. 19, 20, 23, 30).

Alle URLs wurden zuletzt am 30. 04. 2017 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