Integrating Security Concerns into Safety Analysis of Embedded Systems Using Component Fault Trees

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Inf. Max Steiner

Datum der wissenschaftlichen Aussprache: 29. August 2016

Dekan:	Prof. Dr. Klaus Schneider
Erster Berichterstatter:	Prof. Dr. Peter Liggesmeyer
Zweiter Berichterstatter:	Prof. Dr. Karsten Berns

D 386

Zusammenfassung

Nahezu alle neu entwickelten Systeme enthalten heutzutage eingebettete Systeme zur Steuerung von Systemfunktionen. Ein eingebettetes System nimmt seine Umwelt über Sensoren wahr und interagiert mit ihr mittels Aktoren. Systeme, die ihrer Umwelt durch fehlerhaftes Verhalten Schaden zufügen können, werden üblicherweise auf ihre Betriebssicherheit (Safety) hin untersucht. Die Angriffs- oder Datensicherheit (Security) von eingebetteten Systemen wird meistens überhaupt nicht betrachtet. Durch die neuen Entwicklungen im Bereich *Industrie 4.0* und *Internet of Things* werden solche Systeme immer stärker miteinander vernetzt. Dadurch kommen neue Ursachen für Systemausfälle zum Tragen: Schwachstellen in den Software- und Kommunikationskomponenten können von Angreifern ausgenutzt werden, um einerseits die Kontrolle über ein System zu erlangen und andererseits durch gezielte Eingriffe das System in einen kritischen Zustand zu bringen, der entweder dem System selbst oder der Umwelt schadet. Beispiele solcher Schwachstellen und auch erfolgreiche Angriffe werden in letzter Zeit immer häufiger bekannt.

Aus diesem Grund muss man bei eingebetteten Systemen bei der Analyse der Betriebssicherheit (Safety) auch die Datensicherheit (Security) zumindest soweit berücksichtigen, wie sie Auswirkungen auf Ausfälle von Systemkomponenten haben kann.

Ziel dieser Arbeit ist es, in einem Modell zu beschreiben, wie sich Bedrohungen aus Securitysicht auf die Safetyeigenschaft eines Systems auswirken können. Da aber weiterhin die Betriebssicherheit der Systeme im Vordergrund steht, wird die Safetyanalyse erweitert, um Bedrohungen der Security mit zu berücksichtigen, die eine Wirkung auf die Safety des Systems haben können. Komponentenfehlerbäume eignen sich sehr gut dazu, Ursachen eines Ausfalls zu untersuchen und Ausfallszenarien zu finden. Ein Komponentenfehlerbaum eines zu untersuchenden Systems wird um zusätzliche Ereignisse erweitert, die sich auch durch gezielte Angriffe auslösen lassen. Qualitative und quantitative Analysen werden erweitert, um die zusätzlichen Securityereignisse zu berücksichtigen. Dadurch lassen sich Ursachen für Ausfälle finden, die auf Safety- und/oder Securityproblemen basieren. Quantitative oder Semi-quantitative

Analysen ermöglichen es, Securitymaßnahmen besser zu bewerten und die Notwendigkeit solcher zu begründen.

Der Ansatz wurde in mehreren Analysen von Beispielsystemen angewendet: Die Sicherheitskette des Off-road Roboters RAVON, ein adaptiver Tempomat, ein Smart Farming Szenario und ein Modell einer generischen Infusionspumpe wurden untersucht. Das Ergebnis war bei allen Beispielen, dass dadurch zusätzliche Ausfallursachen gefunden werden konnten, die in einem klassischen Komponentenfehlerbaum nicht auftauchen würden. Teilweise wurden auch Ausfallszenarien gefunden, die nur durch einen Angriff ausgelöst werden können und nicht von Ausfällen von Systemkomponenten anhängig sind. Das sind besonders kritische Szenarien, die in dieser Form nicht vorkommen sollten und durch eine klassische Analyse nicht gefunden werden. Dadurch zeigt sich ein Mehrwert des Ansatzes bei einer Sicherheitsanalyse, der sich durch die Anwendung etablierter Techniken auch mit wenig zusätzlichem Aufwand erreichen lässt.

Abstract

Nowadays, almost every newly developed system contains embedded systems for controlling system functions. An embedded system perceives its environment via sensors, and interacts with it using actuators such as motors. For systems that might damage their environment by faulty behavior usually a safety analysis is performed. Security properties of embedded systems are usually not analyzed at all. New developments in the area of *Industry 4.0* and *Internet of Things* lead to more and more networking of embedded systems. Thereby, new causes for system failures emerge: Vulnerabilities in software and communication components might be exploited by attackers to obtain control over a system. By targeted actions a system may also be brought into a critical state in which it might harm itself or its environment. Examples for such vulnerabilities, and also successful attacks, became known over the last few years.

For this reason, in embedded systems safety as well as security has to be analyzed at least as far as it may cause safety critical failures of system components.

The goal of this thesis is to describe in one model how vulnerabilities from the security point of view might influence the safety of a system. The focus lies on safety analysis of systems, so the safety analysis is extended to encompass security problems that may have an effect on the safety of a system. Component Fault Trees are very well suited to examine causes of a failure and to find failure scenarios composed of combinations of faults. A Component Fault Tree of an analyzed system is extended by additional Basic Events that may be caused by targeted attacks. Qualitative and quantitative analyses are extended to take the additional security events into account. Thereby, causes of failures that are based on safety as well as security problems may be found. Quantitative or at least semi-quantitative analyses allow to evaluate security measures more detailed, and to justify the need of such.

The approach was applied to several example systems: The safety chain of the off-road robot RAVON, an adaptive cruise control, a smart farming scenario, and a model of a generic infusion pump were analyzed. The result of

all example analyses was that additional failure causes were found which would not have been detected in traditional Component Fault Trees. In the analyses also failure scenarios were found that are caused solely by attacks, and that are not depending on failures of system components. These are especially critical scenarios which should not happen in this way, as they are not found in a classical safety analysis. Thus the approach shows its additional benefit to a safety analysis which is achieved by the application of established techniques with only little additional effort.

Acknowledgements

I would like to take the opportunity to thank everyone who supported me during the time I worked on this thesis.

First of all, I would like to thank my supervisor Prof. Dr. Peter Liggesmeyer who made this work possible in the first place. He inspired me to write this thesis about the combination of safety and security analysis. Special thanks go to Prof. Dr. Karsten Berns who initially referred me to Prof. Liggesmeyer, and who agreed to be my second referee on such short notice.

I would also like to thank my past and present colleagues at the chair Software Engineering: Dependability: Patric Keller for the support during the early stages and the motivation to write the first publication, Kai Bizik for the good times in our shared office and the many helpful conversations about our thesis topics, and Michael Roth who motivated me with his working morale and his everlasting high spirits to finish the thesis.

Special thanks also go to Thomas Schneider and Caroline Frey for the continuing support with technical or organizational obstacles and especially for the organization of the scholarship for the last six months of this work.

I also thank my other colleagues for the good teamwork that lasted for years.

Contents

List of Figures

List of Tables

1

Introduction

Nowadays, almost everything, from a simple toaster to complex humanoid robots contains embedded systems. An embedded system is able to monitor its environment using different kinds of sensors, and interacts with it using actuators such as electric motors. The sensor inputs are evaluated via controller components that calculate outputs for the actuator components (cf. Fig. 1.1).

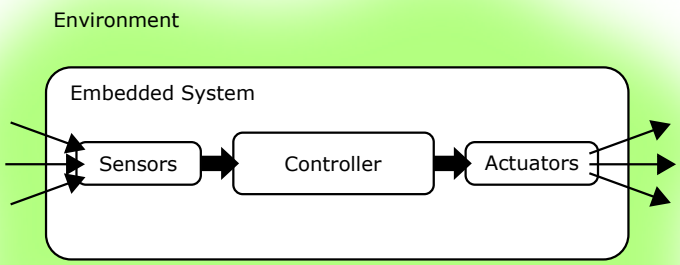


Fig. 1.1. An embedded system

Embedded systems are systems composed of software (controller) and hardware components (sensors, actuators). For systems that might cause harm to its environment, or people in its vicinity, due to system failures, tradition-

ally a safety analysis is necessary. By a safety analysis risks can be detected, and then they can be reduced to an acceptable level by suitable countermeasures. During the approval process of new systems it has to be ensured that only acceptable risks for the system's environment remain in the finished system.

For a safety analysis of a system there are analysis techniques that are partly decades old, accordingly widespread, and accepted by industry and researchers. One prominent technique is Fault Tree Analysis (FTA) which this thesis uses as a basis. Additional to the hardware components, which most of the techniques were initially developed for, in embedded systems there are software components that have to be taken into account too. The initially hardware-specialized analysis techniques were adapted and extended to encompass software components too. That was the first big change in safety analysis.

1.1 Motivation

The newest development in embedded systems is the *Internet of Things* and its application to industry under the term *Industry 4.0*. Embedded systems are connected to larger networks of systems, mostly via open networks as the Internet. Such connected embedded systems are often called Cyber-Physical Systems (CPSs) (cf. Fig. 1.2). Not only newly developed systems are connected but also existing, older systems are retrofitted to be connected to larger systems. This offers the ability to remotely manage or maintain systems.

But the downside is that most older systems were developed for isolated conditions where the access control was achieved by physically locking the system in some kind of box. An authorized technician had a key to that box, and only then he had complete access to the system. If such devices are connected to the Internet, it might be that an attacker who can connect to a device immediately gains complete access to control the system. That such systems exist show several examples which became known over the last few years (more about that later). To conclude so far: Networking embedded systems introduces new vulnerabilities into the systems.

That would be manageable if this fact would be taken into account during the retrofitting of existing systems, or the development of new systems. The current safety analyses only take into account failures that result from random faults, but not targeted manipulations of systems or system components. By using software components, and at the same time more and more networking of different systems, new causes for system failures appear: Vulnerabilities in software components that are accessible via remote connections may be exploited by attackers to gain control over a system, or to bring it into a critical system state by targeted actions.

Examples for vulnerabilities, but also successful attacks, became increasingly known over the last few years. Attackers not only attack web servers to

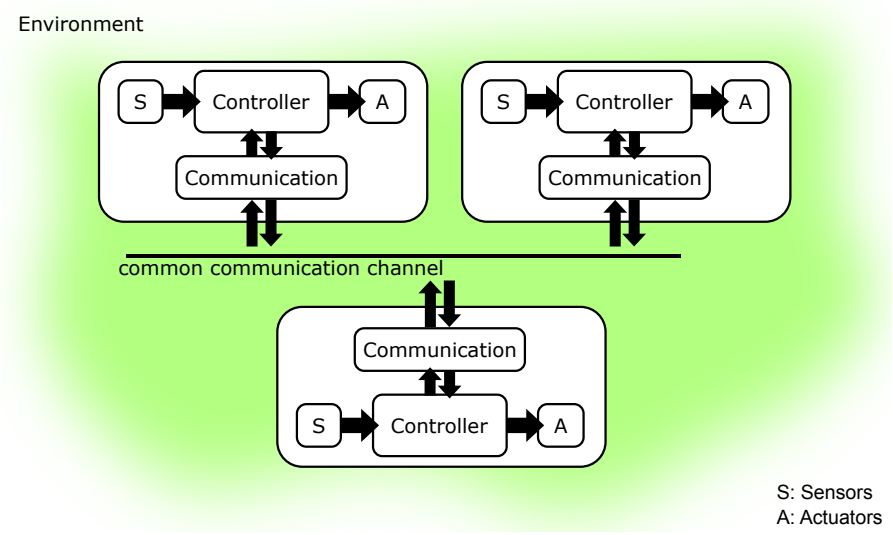


Fig. 1.2. Networked embedded systems

obtain information from connected databases as they used to, but now also control systems of industrial plants, cars, trains, or planes are targets for attacks. Most of the reports about possible attacks refer to security researchers that found critical vulnerabilities in embedded systems. Those found vulnerabilities are often not actively exploited because the necessary effort makes those attack scenarios mostly theoretical.

There are a number of reports about vulnerabilities in embedded systems that are used to control industrial plants. The widely used control system Siemens Simatic had a few vulnerabilities that allowed a possible attacker full access to the system which is used to control whole production cycles (e.g.: in chemical plants) [Bachfeld 11a, Bachfeld 11b, Schmidt 11, ICS-CERT 11, Schirrmacher 15, Klick 15].

A presentation at the Chaos Communication Congress 2014 on security in industrial plants shows that there are a lot of vulnerabilities in such control systems. But most of them can only be exploited by an attacker with detailed background knowledge and a considerable amount of effort [Kleinz 14]. A general overview about security problems of industrial control systems is given in [Kargl 14].

By the increasing availability of networked devices of the Internet of Things such attacks become dangerous to private citizens also. A possible attack on the upcoming smart grid may cripple the supply of electricity or water. A heating system from the company Vaillant that used controllers from the company Saia-Burgess was vulnerable to attacks over the Internet that allowed an attacker full access to the burner control [Stahl 13]. It was possible to shut off the heating, or overheat it which would cause damage to the whole system.

Other related examples of vulnerabilities with possibly catastrophic consequences are in the medical sector where failing devices may directly influence life or death of people. Infusion pumps allow attackers to re-program them via Wireless Local Area Network (WLAN) [Scherschel 15, Bergert 15, Zetter 15]. Other medical devices have hard-coded, never changing, default passwords, or are running on old operating systems with known vulnerabilities that are several years old [Pauli 15, ICS-CERT 13].

Beside industrial control systems, more and more cars, and even airplanes are vulnerable. Hugo Teso gave a presentation at the Hack-in-the-box conference 2013 on how to manipulate the course of a flying airplane using a smartphone and some preparations [Teso 13].

The growing functionality of a car's on-board computer also introduces some critical vulnerabilities. In 2010 security researchers found out that tire pressure measuring systems could be manipulated [Rouf 10]. By itself this is not yet safety critical, but it might be possible to convince drivers to stop their car, or to overwhelm them with false alarms that they ignore real ones. Checkoway, Koscher et al. did an experimental analysis of modern cars to show what might happen if an attacker gains access to one of the control system components [Koscher 10, Checkoway 11]. They conclude that in general attackers can influence all of the car's systems if they gain access to one of the internal components of a car.

In 2015 publications about vulnerabilities in cars heap up. Some are only security relevant, but others also have implications on the safety of the car. The vehicle immobilizer in cars of several brands has a flaw that allows to circumvent it with only low effort [Verdult 15]. This work was already done and ready to publish in 2013, but it was held back until 2015 because the manufacturer enforced a holdback period via court order. This is an example for the problems with publishing security vulnerabilities. Companies do not want them published at all because they fear a decrease in reputation.

Other researchers found ways to remotely access the car's operating system. They were able to unlock current model BMWs equipped with the ConnectedDrive technology via the mobile phone network [Holland 15], or to send commands to the on-board computer using a diagnose dongle of a Corvette via SMS [Foster 15].

Car manufacturers such as General Motors add remote control capabilities via smartphone applications to their cars. Those applications running on the smartphone can contain new vulnerabilities [Kamkar 15] that allow an attacker to control the car.

The most recent and most demonstrative example for safety critical vulnerabilities is the hack of a Jeep Cherokee [Greenberg 15]. The researchers Miller and Valasek were able to exploit a vulnerability in the infotainment system of the Jeep using the mobile network. The flaw in the infotainment system allowed them to access the CAN bus, and they were able to control system functions such as acceleration, brake, door locks, and during reverse drive even the steering.

Besides those mentioned vulnerabilities that to my knowledge were not actively exploited so far, there are some real attacks on running systems. Maybe the best known attack on an industrial control system was the sabotage of uranium centrifuges in Iran using the malware Stuxnet [Langner 11].

In 2014 there were reports about an attack in a German steel mill that resulted in heavy damage to the facility. The reports did not mention which steel mill was attacked and all operating companies denied the attack [Scherschel 14].

Another example is a pipeline in Turkey that exploded as the consequence of an attack on the control system of a valve [Robertson 14]. An attack on the network of an unnamed railroad company in the US disrupted their train service [Zetter 12].

The problem is not only that current control systems are vulnerable, but patching known vulnerabilities is harder than in pure software systems. Often simple update mechanisms are missing [Schneier 14]. A lot of the previously mentioned vulnerabilities remained untreated for years. What makes this even worse is that there are search engines such as Shodan[1] which simplify the search for vulnerable control systems significantly.

1.2 Problem Statement

Security flaws in existing systems such as the examples mentioned earlier remained mostly undetected because in a traditional safety analysis such causes for safety failures are not found. This results from the fact that there are separate experts and analysis techniques for safety and security. If a safety expert analyzes a system according to safety, security is not taken into account, and vice versa. Control systems are usually analyzed for safety only, not for security. A current safety analysis only investigates random faults, deliberate attacks are not taken into account. Countermeasures resulting from a safety analysis are only implemented to mitigate causes that are the most probable according to the analysis. Attacks on the other hand may cause events that otherwise would only come to happen with an extremely low probability which would not have required countermeasures. Thereby an attack that eventually would also be easily conducted without requiring much resources could bring the system into a critical state which might cause substantial damage. For this reason it is especially important that during the development of new systems a safety analysis also takes security causes into account.

1.3 Contribution

The goal of this thesis is to describe in a comprehensive model how vulnerabilities of the security of systems can have an effect on the safety of said systems.

[1] `https://www.shodan.io/`

To this end Component Fault Trees (CFTs) developed for safety analysis are extended by additional causes that arise from security problems. In the resulting enhanced CFT the same qualitative analyses can be conducted as in pure safety CFTs. Additionally, it is investigated to what extend quantitative analysis techniques for CFTs can be adapted for the enhanced CFTs. By those analyses that take into account safety as well as security aspects, causes for failures can be identified that depend on safety as well as security problems. Quantitative, or at least semi-quantitative, analyses make it possible to estimate security measures and to justify the necessity thereof. The extension of established analysis methods facilitates the use by experienced safety analysts that know how to analyze CFTs.

1.4 Structure

This thesis is structured into 6 chapters. Following this introduction, Chap. 2 discusses the state of the art in safety and security analysis. The state of the art is divided into three parts. The first part (Sect. 2.1) explains safety analysis techniques starting with FTA and its derivatives, followed by supporting techniques Failure Mode and Effects Analysis (FMEA) and Hazard and Operability Study (HAZOP) that will be used for this thesis. The second part (Sect. 2.2) is handling security analysis based on Attack Trees (ATs), followed by threat modeling techniques that are used to develop the Attack Trees. And the third part (Sect. 2.3) shows the related work about combining safety and security analysis. The main contribution is distributed over Chaps. 3 and 4. It starts with the overall process how to model a system taking additional security events into account. Then different combinations and approaches for building the combined trees of Component Fault Trees and Attack Trees are discussed, and ends with the description of the final modeling approach. After modeling the combined trees, in Chap. 4 the extensions to qualitative and quantitative analysis are shown including a discussion of ratings for events, and calculation rules. The whole approach is applied to several examples in Chap. 5. And finally, Chap. 6 concludes the thesis.

2

Related Work and Fundamentals

In the past, most embedded systems used to run in encapsulated environments. Maintenance actions needed physical access to the system itself to even connect to it. Access control was implicitly provided by limiting physical access to the existing maintenance interfaces. Login mechanisms or encrypted communication were not a high priority.

Nowadays, on the other hand, the majority of embedded systems is connected to a network, so maintenance or monitoring of the systems becomes easier. If that is done with systems developed without security in mind, attackers with different motivations may threaten the system's ability to run according to the specifications. An example for an embedded system with such vulnerabilities was the Siemens Simatic micro programmable logic controller [ICS-CERT 11].

To avoid such problems, not only safety but also effects of security threats on safety has to be taken into account during the development of new systems. The main contribution of this thesis is an approach to integrate security aspects into safety analysis.

This chapter will first provide an overview about safety analysis (Sect. 2.1) as well as security analysis (Sect. 2.2), and will later discuss approaches that include both safety and security aspects (Sect. 2.3).

The focus lies on Fault Trees (FTs) and their derivations because they are well established and widely accepted by industry and researchers to conduct safety analyses. The component-oriented Component Fault Trees (CFTs) allow to model and analyze complex systems with acceptable effort. They are an appropriate instrument to analyze interrelationships between faults and to find combinations of causes that lead to a failure. Therefore, they are very well suited to include additional causes for failures and their effects on safety of the system under study.

2.1 Safety Analysis Techniques

According to [IEC 61508-4 10] a system is safe if it is free from unacceptable risks. Safety analysis is required to determine whether a system is safe enough for its intended use.

Definition 1 (Safety) *In [IEC 61508-4 10] safety is defined as "freedom from unacceptable risk".*

Definition 2 (Risk) *Risk is defined as the "combination of the probability of occurrence of harm and the severity of that harm" [IEC 61508-4 10].*

Definition 3 (Harm) *Harm is defined as "physical injury or damage to the health of people or damage to property or the environment" [IEC 61508-4 10].*

Safety is also related to reliability in the sense that safety of a system depends on the reliability of safety-related system components.

Definition 4 (Reliability) *Reliability is the property of an entity regarding its qualification to fulfill the reliability requirements during or after given time periods with given application requirements [DIN 40041 90].*

Nicol et al. conducted a survey concerning existing model-based techniques for evaluating system dependability (respective reliability), and how they are extended to evaluate system security [Nicol 04]. They conclude that for system reliability there have been quantitative techniques for a long time, but for system security quantitative techniques are much less common. Over the years a lot of techniques to analyze safety were developed. They can be classified into techniques that find failures by examining the effects of faults on the system (inductive methods), and techniques that examine failures to find their causes (deductive methods).

Definition 5 (Fault) *A fault is an abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function [IEC 61508-4 10]. A fault may be the cause of a failure.*

Definition 6 (Failure) *A failure is either the termination of the ability of a functional unit to provide a required function, or operation of a functional unit in any way other than as required [IEC 61508-4 10]. A failure of a subcomponent can be seen as a fault in the context of the superordinate component or system.*

Some sources also define the term *error* as the underlying cause of a fault [Avizienis 04]. Although, there might be some confusion about the use of that term because of different definitions. This thesis uses the definitions of IEC 61508 as a reference which only distinguishes between faults and failures.

In the following, the focus lies on deductive methods and Fault Tree Analysis in particular because it is a technique that is widely used in industry. Failure Mode and Effects Analysis (FMEA) and Hazard and Operability Study (HAZOP) are explained as examples for inductive methods that can be used in combination with Fault Tree Analysis (FTA).

2.1.1 Fault Tree Analysis (FTA)

A method often used in industry for safety and reliability analysis is Fault Tree Analysis (FTA). FTA was developed in the 1960s in the Bell Labs (see [Ericson 99] for a history of Fault Trees (FTs)). The US Nuclear Regulatory Commission Regulation (NUREG) published an extensive compendium, the *NUREG-0492 Fault Tree Handbook* [Vesely 81] that treats FTA in detail. Later, NASA published a book, the *Fault Tree Handbook with Aerospace Applications* [Vesely 02] that deals with the application of FTA to the aerospace domain. Together, these books provide a comprehensive insight into FTA. The application of FTA is standardized in international standard [IEC 61025 06] and German national standard [DIN 25424-1 81, DIN 25424-2 90].

FTA is a deductive method for finding causes and combinations of causes that lead to system failures. It can be used in early phases of the development process, when only a few details are known about the system. Then, a preliminary qualitative analysis is possible. This preliminary FT can be used to make high-level decisions during system development. In later development stages, when more details are known, the preliminary FT can be extended to make a quantitative analysis possible. Analyses can cover systems that consist of hardware and/or software components.

An FT consists basically of events and gates that combine events. There are three types of events in an FT: Top Events, Basic Events and Intermediate Events. In standard FTs there are three basic gates: `AND`, `OR`, and `NOT`-gates that implement their corresponding logic functions. Derived from these basic gates, standard FTs may also include `XOR` or N-out-of-M (`NooM`) gates. But the `NOT` gate and the derived gates `XOR` and `NooM` should the handled with care [Andrews 00]:

A `NOT`-gate implies that working components can contribute to a failure of the system, and vice versa, a failed component can improve system reliability. This is counterintuitive for the engineer creating the FT to his perception of the functionality of systems. Using only `AND` and `OR`-gates makes an FT coherent. Adding additional failures cannot improve the overall reliability. Most works handling FTs use coherent FTs. Calculations in non-coherent FTs become much more complex than in coherent FTs. There may be, however, situations in which gates derived from `NOT` are necessary to model a particular circumstance. This is why the `NOT`-gate will also be defined in this thesis. But it will be avoided during modeling.

The dependencies between Basic Events (BEs) and Top Event (TE) are usually represented in a tree structure. The different event types will be defined in the following.

Definition 7 (Top Event (TE)) *The system failure modeled in a Fault Tree is called a* Top Event (TE)*. It is the root of the Fault Tree. Sometimes it is also called a* top level event*.*

Definition 8 (Basic Event (BE)) *The basic causes for a Top Event are called* Basic Events (BEs). *They are the leaves of the Fault Tree.*

Definition 9 (Intermediate Event (IE)) *If BEs are combined using logic gates, the result is an* Intermediate Event (IE). *Intermediate Events are sometimes separate nodes in the Fault Tree, as in the original definition from [Vesely 81], or the logic gates have also the function of Intermediate Events by attaching a label to the gates. In this thesis the term* gate *will be used synonymously with* Intermediate Event.

Figure 2.1 shows an example FT consisting of three BEs (BE_1, BE_2, BE_3), one `AND`-gate (AND_1), one `OR`-gate (OR_1) and a TE (Out_1). The example is modeled with the tool ESSaRel [ESSaRel 09] using the following symbols: Black triangles represent TEs, white boxes are gates (using the European notation) and white circles depict the BEs. All events should have a label that is the name of the event.

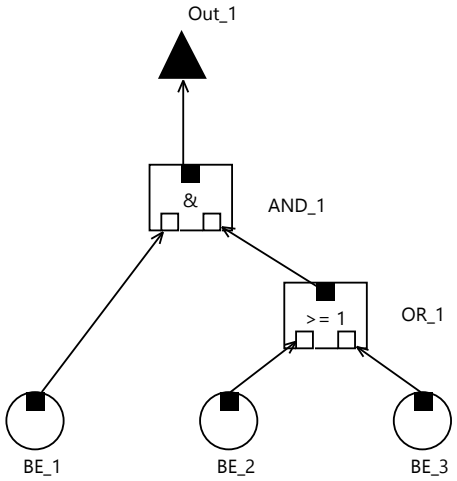


Fig. 2.1. An example Fault Tree

In this work CFTs are used, a derivative of FTs. The analyses that are described here for FTs are also applicable for CFTs [Kaiser 03b].

Component Fault Trees

A Component Fault Tree (CFT) is an extended FT with an additional focus on system components and reusability of subtrees [Kaiser 02, Kaiser 03b]. A system usually consists of several components which by themselves may consist of nested subcomponents. One CFT models one of these components (that may

also contain subcomponents) following the component hierarchy of the system. All examples in this work are modeled with the tool ESSaRel [ESSaRel 09].

In Fig. 2.2 an example CFT model is shown. It models a system with three subcomponents. Figure 2.2a shows the highest hierarchy level of the model. The components (gray rounded boxes) are interconnected via output ports (small black boxes) and input ports (small white boxes). The arrows of the connectors always point from output ports to input ports. The TE (black triangle) of a component defines an output port. The CFTs in Figs. 2.2b – 2.2d model the three subcomponents `Comp1`, `Comp2`, and `Comp3`. The CFT of `Comp1` models the same structure as the FT in Fig. 2.1. This shows the commonalities of FTs and CFTs. The CFTs of `Comp2` and `Comp3` show some differences. Figure 2.2c is an example that a CFT allows to model all failure modes of a component at once. This fact simply leads to more than one TE and therefore more output ports. And finally Fig. 2.2d shows that a CFT can have input ports (white triangles). The TE of `Comp3` depends on both internal (`BE_1`) and external causes (`In_1`, `In_2`, `In_3`).

If a CFT of one component is analyzed by itself, input ports can be seen as BEs. But if this CFT is analyzed as part of a higher level CFT, the input ports directly link to other CFTs as if the referenced CFT would be inserted instead of the input port.

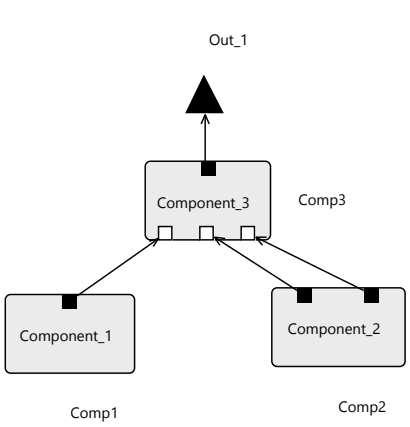
The component-wise construction of CFTs allows easier modeling of large systems than with FTs. Although the name suggests it, complex CFTs are not necessarily trees, but they are Directed Acyclic Graphs (DAGs). So, mesh-like structures are possible but cyclic dependencies have to be avoided. In the remainder of this thesis, the term *tree* is also used for CFTs.

The distinction of components allows the independent development of CFTs. The naming conventions ensure that events from different components can be distinguished. In the previous example `BE_1` appears in all three components `Comp1`, `Comp2`, and `Comp3`. They are referred to as `Comp1.BE_1`, `Comp2.BE_1`, and `Comp3.BE_1` outside their respective components. Despite these differences, CFTs are semantically equivalent to FTs, and the same analyses are possible.

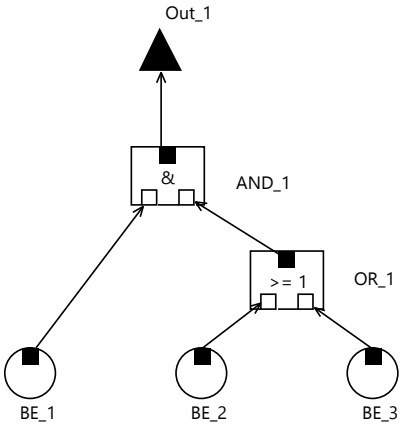
Qualitative Analysis

As a first analysis step, independent of probabilities or other ratings for BEs, a qualitative analysis may be conducted. It can already be done in early development phases using preliminary FTs. Some qualitative analysis techniques will be described in the following. In every FTA, the first step is to determine the Minimal Cut Sets (MCSs) for the TE.

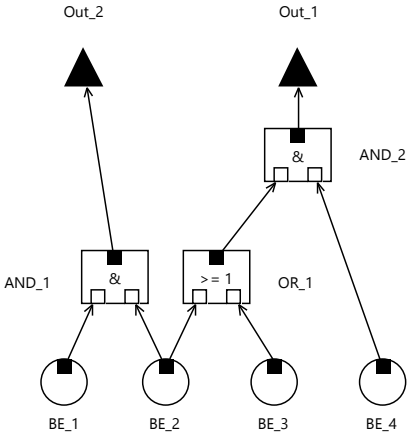
Definition 10 (Minimal Cut Set) *A Minimal Cut Set (MCS) is a smallest combination of Basic Events (BEs) which, if they all occur, will cause the Top Event (TE) to occur [Vesely 81].*



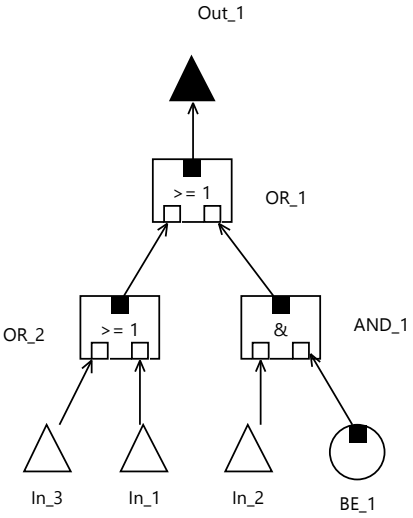
(a) An example CFT (Overview).



(b) An example CFT (Subtree for `Comp1`).



(c) An example CFT (Subtree for `Comp2`).



(d) An example CFT (Subtree for `Comp3`).

Fig. 2.2. A CFT consisting of three subcomponents

Strictly speaking, the term *Minimal Cut Set* is only defined for coherent FTs, thus FTs that do not contain `NOT`-gates or negated BEs. The more general term is *Prime Implicant (PI)* which also includes non-coherent FTs (see [Rauzy 01] for the mathematical foundations of PIs). PIs appear less commonly in failure analysis, because the use of negations in FTs is less intuitive as discussed earlier in Sect. 2.1.1. In this thesis, the term *Minimal Cut Set* is

used for minimal failure scenarios because no negated events were modeled. But still, all aspects of this work also apply to PIs, unless indicated otherwise.

Every MCS represents a minimal failure scenario. In general, an FT contains multiple MCSs corresponding to different failure scenarios. The TE of an FT can be depicted as the result of the disjunction of all Minimal Cut Sets MCS_i, $1 \leq i \leq n$, $n \in \mathbb{N}$.

$$TE = MCS_1 \vee MCS_2 \vee MCS_3 \vee \cdots \vee MCS_n \tag{2.1}$$

Whereas, MCS_i is a conjunction of Basic Events BE_j, $1 \leq j \leq m$, $m \in \mathbb{N}$.

$$MCS_i = BE_1 \wedge BE_2 \wedge BE_3 \wedge \cdots \wedge BE_m \tag{2.2}$$

MCSs can be determined using a variety of different algorithms. One of the most common is the *top-down* algorithm which is described in [Vesely 81, Vesely 02]. The FT is translated into Boolean equations. Starting with the equation for the TE, the Intermediate Events (IEs) are substituted by their expressions and the equation is expanded until the MCS expression for the TE is obtained. This step is repeated until all IEs are substituted. At last, redundancies are removed to obtain the MCSs.

Applied to the example FT depicted in Fig. 2.1, the following steps are needed: First, these are the Boolean equations for the FT:

$$Out_1 = BE_1 \wedge OR_1 \tag{2.3}$$

$$OR_1 = BE_2 \vee BE_3 \tag{2.4}$$

Equation 2.3 is already in MCS form, so the IE OR_1 is substituted with the right side of equation 2.4 and expanded which yields:

$$\begin{aligned} Out_1 &= BE_1 \wedge (BE_2 \vee BE_3) \\ &= BE_1 \wedge BE_2 \vee BE_1 \wedge BE_3 \end{aligned} \tag{2.5}$$

The MCSs resulting from equation 2.5 are:

$$\begin{aligned} MCS_1 &= \{BE_1, BE_2\} \\ MCS_2 &= \{BE_1, BE_3\} \end{aligned}$$

More sophisticated algorithms can also handle large FTs and calculate Prime Implicants for non-coherent FTs. In [Rauzy 01] algorithms based on Binary Decision Diagrams (BDDs) [Bryant 86] to calculate MCSs are described.

As mentioned earlier, FTs will most probably have several MCSs. Considering all of them can be time-consuming in large FTs. An analysis should focus on the most critical ones. So, it is part of the analysis to determine the most critical MCSs.

Qualitative rankings of MCSs can be determined according to their contributions to the TE (see [Vesely 81]). MCSs can be ordered according to

cardinality from small to large. For example, the larger an MCS is, the more BEs have to occur simultaneously, so that the TE is triggered. From that follows that the cardinality (or size) of an MCS can be a qualitative measure for criticality of an MCS. In absence of quantitative data, it can be said that a greater MCS size correlates with a lower criticality.

Definition 11 (MCS size) *The size of an MCS is defined as the number of events included in the MCS.*

The size of MCSs can be used to check design criteria, e.g.: "no single points of failure allowed" which is equal to "MCSs with size 1 have to be mitigated". The set of MCSs can also be used to find BEs that are part of several MCSs. A BE that is contained in many relevant[1] MCSs is more important than one that is contained in only one MCS. Finding such BEs can be used to determine the position of countermeasures: With countermeasures against one BE several MCSs may be mitigated. The decision which BE to counter depends on the cost-effect balance of the countermeasures. First, BEs from small MCSs (that are more likely to happen) are selected. From that list a combination of BEs is selected that counters all required MCSs with the least effort.

In terms of the previous example: Both MCSs have the same size, but BE_1 is included in both MCSs. So, a countermeasure against BE_1 mitigates both MCSs at once.

Quantitative Analysis

After finishing the qualitative analysis, a quantitative analysis of FTs may be conducted. For that purpose probabilities of failure, or failure rates, for BEs have to be available.

Definition 12 (Failure rate) *The failure rate $\lambda(t)$ of an entity is a reliability parameter. $\lambda(t) \times dt$ is the probability that this entity will fail within the time interval $[t, t + dt]$ under the condition that it did not fail during time interval $[0, t]$.*

Failure rates usually vary over the lifetime of a system and also depend on environmental conditions. A common description of failure rates over system lifetime is the so-called bathtub curve [Klutke 03]. It can be divided into three intervals: In the beginning there is a "break-in"-period during which the failure rate decreases because initial bugs are corrected. During the middle period the failure rate is nearly constant when the system runs under its intended working conditions. And the last interval is a "wear-out"-period when the system slowly degrades due to physical stress. For software systems the failure rate is also generally not constant over the whole lifetime. Whenever a bug is fixed, the failure rate decreases. If new functionality is introduced, often

[1] What *relevant* MCSs are has to be defined before. Usually MCSs up to a size of 2 or 3 are defined as relevant [Vesely 81].

new bugs emerge, so the failure rate increases. So, the failure rate can only be constant during a period of time when the software is not changed.

In classic FTs only constant failure rates per hour or per duty cycle are considered, while time dependent effects are ignored. A constant failure rate per hour λ leads to exponential failure probability distributions. A constant failure rate per duty cycle is equivalent to the failure probability. Components can have standby failure rates and operating failure rates which themselves are constant.

The quantitative analysis provides more detailed insight in the order of magnitude of the BE probabilities. Results can be probabilities of the TE or MCSs. It is used to check compliance to customer or regulatory requirements, and in case of violated requirements, to decide on which BEs the effort of countermeasures will be concentrated.

If BEs are connected via logic gates and probabilities are known for all BEs, the resulting probabilities are calculated as follows. For both the `AND` and `OR`-gates calculation rules for gates with two inputs (equations 2.6 and 2.8), and n inputs (equations 2.7 and 2.9) are presented. All BEs have to be stochastically independent from each other for these calculations to be correct. The probability calculation for the `NOT`-gate is shown in equation 2.10.

$$\texttt{AND} \quad P(A \wedge B) = P(A) \times P(B) \qquad A, B \in BE \quad (2.6)$$

$$P\left(\bigwedge_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} P(X_i) \qquad i, n \in \mathbb{N}, X_i \in BE \quad (2.7)$$

$$\texttt{OR} \quad P(A \vee B) = P(A) + P(B) - P(A) \times P(B) \qquad A, B \in BE \quad (2.8)$$

$$P\left(\bigvee_{i=1}^{n} X_i\right) = 1 - \prod_{i=1}^{n} (1 - P(X_i)) \qquad i, n \in \mathbb{N}, X_i \in BE \quad (2.9)$$

$$\texttt{NOT} \quad P(\overline{A}) = 1 - P(A) \qquad A \in BE \quad (2.10)$$

There are different possibilities to calculate the failure probability of a TE (see [Vesely 81]). Provided, the failure probabilities of all MCSs are already calculated, the failure probability of the TE can be approximated. An upper bound $P'(TE)$ for the TE failure probability is gained by summing up the failure probabilities of all MCSs. The TE results from an `OR` combination of all MCSs (see equation 2.1). Equation 2.8 shows the calculation of an `OR`-gate with two inputs. If instead the probabilities of the MCSs are simply summed up without subtracting the probability of the intersection of two MCSs, the result will always be greater or equal the exact value – in other words, it is a conservative approximation. For the purpose of this example the following probability values are arbitrarily chosen:

$$P(BE_1) = 10^{-5} \qquad P(BE_2) = 2 \times 10^{-3} \qquad P(BE_3) = 4 \times 10^{-2}$$

Applied to the example from Fig. 2.1, the results are:

$$
\begin{aligned}
P(MCS_1) &= P(BE_1 \wedge BE_2) \\
&= P(BE_1) \times P(BE_2) \qquad \text{(equation 2.6)} \\
&= 2 \times 10^{-8} \\
P(MCS_2) &= P(BE_1 \wedge BE_3) \\
&= P(BE_1) \times P(BE_3) \qquad \text{(equation 2.6)} \\
&= 4 \times 10^{-7} \\
P'(TE) &= P(MCS_1) + P(MCS_2) \\
&= 2 \times 10^{-8} + 4 \times 10^{-7} \\
&= 4.2 \times 10^{-7}
\end{aligned}
$$

If an approximation is not enough, the exact probability value can be calculated bottom-up based on the BE probabilities using the rules expressed in equations 2.6 to 2.10. The bottom-up calculation is now applied to the example FT:

$$
\begin{aligned}
P(OR_1) &= P(BE_2) + P(BE_3) - P(BE_2) \times P(BE_3) \qquad \text{(equation 2.8)} \\
&= 2 \times 10^{-3} + 4 \times 10^{-2} - 2 \times 10^{-3} \times 4 \times 10^{-2} \\
&= 4.192 \times 10^{-2} \\
P(TE) &= P(Out_1) = P(AND_1) \\
&= P(BE_1) \times P(OR_1) \\
&= 10^{-5} \times 4.192 \times 10^{-2} \\
&= 4.192 \times 10^{-7}
\end{aligned}
$$

The results show the difference between the approximation $P'(TE)$ and the exact result $P(TE)$.

$$
\begin{aligned}
P'(TE) &= 4.2 \times 10^{-7} \\
P(TE) &= 4.192 \times 10^{-7}
\end{aligned}
$$

There is another, more efficient, method to calculate the exact TE probabilities. The Boolean formula which is depicted by the FT with the BEs as variables, can be described by a BDD (see [Vesely 02] or [Rauzy 01] for a more detailed description on BDD algorithms). This BDD is simplified to a Reduced Ordered Binary Decision Diagram (ROBDD), to reduce calculation effort. For each BE e_i the probability $P(e_i)$ is assigned to the true branches, and $1 - P(e_i)$ is assigned to the false branches of the BDD. For each path leading to terminal symbol 1, the branch probabilities are multiplied, as they are combined via `AND`-gates. The results of all paths leading to terminal symbol 1 are summed up to receive the probability of the TE $P(TE)$. Because these minimal paths are disjoint, this summation yields the exact result for this `OR`-combination of paths.

In Fig. 2.3 an ROBDD for the example with variable order BE_1, BE_2, BE_3 is shown. Solid lines are assigned to true branches, dashed lines to false branches. There are two paths leading to terminal symbol 1:

$$Path_1 = BE_1 BE_2$$
$$Path_2 = BE_1 \overline{BE_2} BE_3$$

The probability is calculated as:

$$\begin{aligned} P(Path_1) &= P(BE_1) \times P(BE_2) \\ &= 10^{-5} \quad \times 2 \times 10^{-3} \\ &= 2 \times 10^{-8} \\ P(Path_2) &= P(BE_1) \times (1 - P(BE_2)) \times P(BE_3) \\ &= 10^{-5} \times (1 - 2 \times 10^{-3}) \times 4 \times 10^{-2} \\ &= 3.992 \times 10^{-7} \\ P(TE) &= P(Path_1) + P(Path_2) \\ &= 2 \times 10^{-8} + 3.992 \times 10^{-7} \\ &= 4.192 \times 10^{-7} \end{aligned}$$

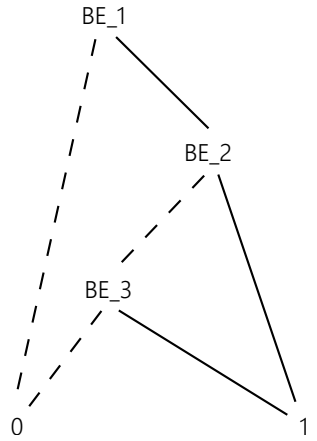


Fig. 2.3. A corresponding ROBDD for the example from Fig. 2.1

MCSs are determined to find combinations of faults leading to the system failure, or different failure scenarios. The failure probability of a TE is calculated to check if a system complies with given requirements. If the TE probability has to be decreased, some countermeasures have to be implemented against BEs. But not all BEs contribute to the TE probability to the

same degree. According to Vesely et al. only about 10–20% of the BEs contribute significantly to the TE probability [Vesely 02]. An importance analysis is conducted to find the BEs whose countering has the most effect on the TE probability.

Importance Measures

There are several different importance measures described in the literature [van der Borst 01, Vesely 02]. The four most common measures are Fussell-Vesely (FV) Importance, Risk Reduction Worth (RRW), Risk Achievement Worth (RAW) and Birnbaum's Importance Measure (BM).

The FV measure [Fussell 75] calculates the contribution of an event to the TE. It can be applied to all events in an FT, Basic Events as well as Intermediate Events. It is calculated by summing up the probabilities $P(MCS_e)$ of all MCSs that contain the particular event e and dividing the sum by the approximation of the TE probability $P'(TE)$. Whereas m_e is the number of MCSs that contain event e, and m the number of all MCSs.

$$FV_e = \frac{\sum_{i=1}^{m_e} P(MCS_{e_i})}{P'(TE)} = \frac{\sum_{i=1}^{m_e} P(MCS_{e_i})}{\sum_{j=1}^{m} P(MCS_j)} \tag{2.11}$$

The RRW measure calculates the decrease in the TE probability if a given event is prevented from occurring. First, the TE probability $P(TE)$ is calculated as usual. Then the probability of the event under study is set to 0 and the TE probability is calculated again. For the risk reduction a relative $\frac{P(TE)}{P(TE_{e=0})}$ and an absolute measure $P(TE) - P(TE_{e=0})$ can be calculated.

The RAW is the counterpart to the RRW. It calculates the increase in the TE probability if a given event is guaranteed to occur. The probability of the event under study is set to 1 and the TE probability is calculated. As for the RRW a relative $\frac{P(TE_{e=1})}{P(TE)}$ and an absolute measure $P(TE_{e=1}) - P(TE)$ can be calculated.

The last measure is BM. It is a combination of the absolute measures of RRW and RAW: BM = RRW + RAW. BM is equivalent to a sensitivity analysis: First, the probability of the event under study is set to 1 and the TE probability is calculated. Then the probability of the event under study is set to 0 and the TE probability is calculated again. The difference $P(TE_{e=1}) - P(TE_{e=0})$ is Birnbaum's Importance Measure.

Regarding the previous example (Fig. 2.1) the importance values for the BEs BE_1, BE_2, BE_3 are shown in Table 2.1. If the BEs are ordered according to these importance measures, the order is: $BE_1 > BE_3 > BE_2$. The calculation effort of the FV measure is the lowest of the presented ones, yet it results in the same order as the others. This is why the FV measure is the one used mostly.

Table 2.1. Importance values for the example BEs

Basic event	FV	RRW		RAW		BM
		abs	rel	abs	rel	
BE_1	1	4.192×10^{-7}	∞	4.192×10^{-2}	10^5	4.192×10^{-2}
BE_2	0.048	0.192×10^{-7}	1.048	9.581×10^{-6}	23.855	9.6×10^{-6}
BE_3	0.952	3.992×10^{-7}	20.96	9.581×10^{-6}	23.855	9.98×10^{-6}

Extensions of Fault Trees

One big disadvantage of FTs or CFTs is that they are unable to analyze sequences of events or timed behavior. It is, however, possible to combine Markov models with FTs to solve these problems [Vesely 02]. There also exist several extensions of FTs that introduce new gates that allow modeling of dynamic behavior. In Parametric Fault Trees (PFTs) identical subtrees can be replaced by one parameterized subtree to simplify and to reduce the overall FT [Bobbio 03].

Dynamic Fault Trees (DFTs) allow the modeling of functional or temporal dependencies between BEs [Bechta Dugan 92]. In this way it is possible to model prioritized events or sequences of events.

Repairable Fault Trees (RFTs) were developed to model especially repair processes, consisting of a set of components that can be triggered by failure events [Codetta-Raiteri 04].

Generalized Fault Trees (GFTs) combine logic gates from FTs, PFTs, DFTs, and RFTs to a unified model [Codetta-Raiteri 05]. Thereby dependencies, redundancies, symmetries, and repair mechanisms can be modeled using one modeling technique. For the evaluation the GFTs are transformed into Generalized Stochastic Petri Nets (GSPNs) and analyzed using the usual Petri net analysis methods [Marsan 95].

Another technique that combines CFTs and state-based approaches, are State/Event Fault Trees (SEFTs). Introduced by Kaiser in [Kaiser 03a], they can be used to model complex behavior of systems consisting of hardware and software [Steiner 12].

The quantitative analysis of FTs and their derivatives depends on failure rates or probabilities of occurrences of faults. Sometimes those are hard to obtain. There are approaches that use value intervals that are easier to estimate, instead of fixed numbers to evaluate FTs [Carreras 01]. Förster and Trapp [Förster 09] developed an FT-based method to cope with the problem of few or no information about failures early in the development. They base the analysis on CFTs to model system composition. The uncertainty of (software) safety probabilities for BEs is modeled with probability distributions over intervals instead of single probabilities. For calculating an overall probability distribution every interval is sampled numerously and the overall probability per sample is calculated.

To summarize, FTs are an important tool to conduct a safety analysis during the development of a system. They can be used to conduct high-level qualitative analyses as well as detailed quantitative analyses. With the mentioned extensions they can also handle state-based behavior of complex systems.

2.1.2 Inductive Safety Analysis Techniques

In standards such as [IEC 61025 06] or [IEC 60300-3-1 05] for Fault Tree Analysis a combination of inductive and deductive analysis techniques is suggested. They especially mention FMEA and HAZOP. These two techniques will be described in short in this section since they are used as supporting techniques to create the CFTs in this thesis.

Failure Mode and Effects Analysis

Failure Mode and Effects Analysis (FMEA) is a systematic approach to investigate cause-consequence relationships between component faults (cf. Definition 5 on p. 8) and system failures (cf. Definition 6 on p. 8). Standard [IEC 60812 06] describes the basic approach to conduct an FMEA. It is a widely used inductive analysis technique that can be applied to functions, interfaces, software, or whole systems [Bowles 01]. A disadvantage is that it cannot be used to find combinations of faults, only single faults can be found. Therefore, a subsequent FTA often is used to deduct combinations of causes for the found failure modes [IEC 61025 06, IEC 60300-3-1 05]. The result of an FMEA is a table which includes the failure modes of each system component, and for each failure mode possible causes and effects. Commonly used entries in this table are according to [IEC 60812 06]: investigated component, failure mode, description of the failure mode/local effect of the failure mode, possible cause for the failure, effect at the system level, recommended failure detection mechanism, recommended mitigation mechanism, and recommended design changes. The analysis is conducted by a moderated group of domain experts for all aspects of the product or process.

The system under study is divided into its components, and for each component the failure modes are identified. For each failure mode the effect on the system level and the cause (originating in the component itself or in other components) are also identified (see Fig. 2.4).

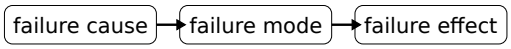


Fig. 2.4. The cause-effect chain of an FMEA

If a Failure Mode, Effects, and Criticality Analysis (FMECA) is conducted, an additional risk analysis is done. It consists of:

1. an estimation of the severity (S) of the failure effect on the system or customer,
2. an estimation of the probability of occurrence (O),
3. an estimation of the probability that the failure is not detected (D),
4. and the calculation of the Risk Priority Number (RPN)
   $RPN = S \times O \times D$, where $S, O, D \in \{1, 2, \ldots, 10\}$

The risk for the studied system is estimated by ranking the failure modes according to the RPN. The identified failure modes can be investigated in more detail by using them as TEs for an FTA.

Hazard and Operability Study

HAZOP is a qualitative technique to find derivations from intended behavior developed first by the British chemical industry in the 1960s. It is described in standard [IEC 61882 01]. It has since then been applied to many other fields. For example Software Hazard Analysis and Resolution in Design (SHARD) is a derivative for software safety analysis [Pumfrey 99]. Like FMEA it is a team-based inductive analysis technique that can be applied to systems consisting of hardware and software. This interdisciplinary team consists of five to ten persons. HAZOP can be conducted in early development phases as soon as first drafts of the component design, and the material and data flows are available. The analysis concentrates on the behavior of the flows between components rather than failure modes of the components themselves.

A set of guide words is combined with parameters of the flows to find questions about potential failure modes. Every combination of guide words and parameters is tested, but not all may make sense. The standard guide words of HAZOP for process plants are: *no*, *more*, *less*, *as well as*, *part of*, *other than*, and *reverse*. For different applications there can be different guide words. In the SHARD approach the guide words are: *omission*, *commission*, *early*, *late*, and *value*. The effect on the system is investigated for each deviation found. For problems that are not safety related, a justification why the design is acceptable is recorded. Safety problems have to be investigated further to find possible mitigations. The results are documented in tabular form [IEC 61882 01].

There are also applications of HAZOP to conduct security analyses. Lano et al. propose HAZOP guide words for the security analysis of object-oriented models in the Unified Modeling Language (UML) [Lano 02]. They provide new guide word interpretations for different UML diagram types, amongst others state-transition, class, and sequence diagrams. The basic elements of an analysis of state-transition diagrams are the transitions. For the attributes of transitions, events, and actions, the guide words are newly interpreted to take the interaction between software and hardware into account. Deviations from design intend described by a class diagram are covered by new guide words. They also developed guide words for sequence diagrams for message timing, message destination, and message conditions.

The HAZOP approach can be used during the development of CFTs to identify input and output ports of components (see also Sect. 2.1.1).

2.2 Security Analysis Techniques

This section deals with security analysis techniques that can be integrated with the safety analysis techniques described in Sect. 2.1. But before analysis techniques are discussed, a few definitions are necessary. First, what is security in the context of this thesis? The following definitions from the ISO/IEC 27000 standard series are used:

Definition 13 (Security) *Security is commonly defined as a combination of confidentiality, integrity, and availability (the C-I-A attributes) [Avizienis 04, ISO/IEC 27000 09].*

Definition 14 (Confidentiality) *Confidentiality is the property of a system that information is not made available or disclosed to unauthorized individuals, entities, or processes [ISO/IEC 27000 09].*

Definition 15 (Integrity) *Integrity is the property of safeguarding the accuracy and completeness of assets, and that assets are not improperly altered [Avizienis 04, ISO/IEC 27000 09].*

Definition 16 (Availability) *Availability is the property of being accessible and correctly usable upon demand by an authorized entity [ISO/IEC 27000 09].*

Safety was defined in a way that the operation of a system should not endanger its environment (cf. Sect. 2.1). Concerning security the effective direction is reversed: The environment of a system should not endanger the operation of the system. Another distinction between safety and security is that safety handles random accidental faults whereas security usually deals with intentional attacks [Saglietti 06].

In general, complex software-controlled systems will most likely contain some vulnerabilities. Those vulnerabilities may lead to undesired events which may compromise assets. Figure 2.5 shows the interrelations between the defined terms below.

Definition 17 (Vulnerability) *A vulnerability is a flaw or weakness of an asset or control that can be exploited by a threat [ISO/IEC 27000 09].*

Definition 18 (Asset) *An asset can be anything that has value to the organization [ISO/IEC 27000 09]. Assets can be information, software, hardware, services, people and their qualifications, skills, and experience, or reputation and image.*

Definition 19 (Control/Countermeasure) *A control is a means of managing risk, including policies, procedures, guidelines, practices or organizational structures which can be administrative, technical, management, or legal in nature [ISO/IEC 27000 09]. It is a mechanism to reduce the effects of vulnerabilities. Here they will be called countermeasures.*

Definition 20 (Threat) *[ISO/IEC 27000 09]: A threat is a potential cause of an unwanted incident which may result in harm to a system or organization.*

Definition 21 (Attack) *An attack is an attempt to destroy, expose, alter, disable, steal, or gain unauthorized access to or make unauthorized use of an asset [ISO/IEC 27000 09]. In other words, an attack is the exploitation or execution of a threat by a threat agent.*

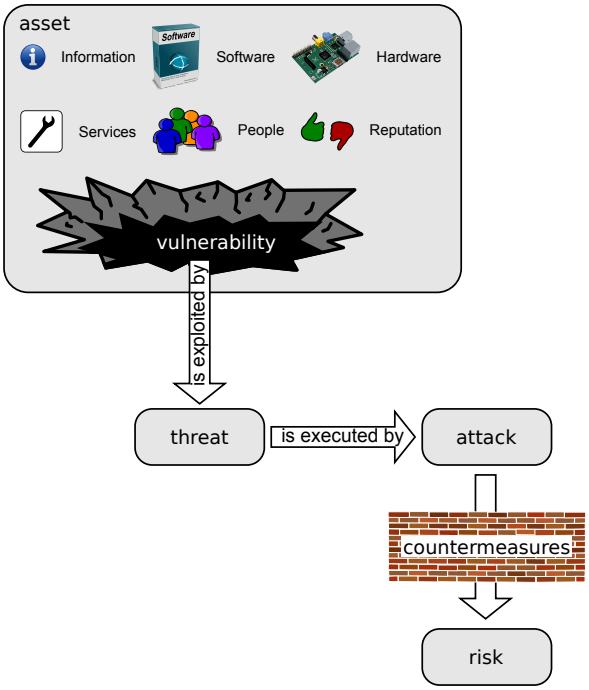


Fig. 2.5. An example to illustrate the interrelations of the definitions

Security analysis tries to identify the vulnerabilities of a system that may lead to compromised assets. An analysis shows what kind of threats may exploit vulnerabilities and which effects on the system they might have. Result of a security analysis is a ranked list of vulnerabilities for which countermeasures have to be implemented.

The focus in this section lies on security analysis techniques that can be easily integrated with safety analysis techniques based on Fault Trees (FTs) or Component Fault Trees (CFTs). The first part is about Attack Trees which are very similar to FTs. After that, threat modeling, a technique that can be used to find vulnerabilities in systems, is described.

2.2.1 Attack Tree Analysis

Attack Trees (ATs), sometimes also called Threat Trees [van Lamsweerde 04, Buldas 06], were introduced 1999 by Bruce Schneier in [Schneier 99]. Derived from FTs (see Sect. 2.1.1), ATs are a tool to model threats in a structured way. They represent attacks on a system in a tree structure with the attack goal as the root (Top Event (TE), see Definition 7 in Sect. 2.1.1), and partial attacks as sub-nodes (Intermediate Events (IEs), see Definition 9) and leaves (Basic Events (BEs), see Definition 8). To model combinations of events, as in FTs, there are `AND` and `OR`-gates as IEs. An `AND`-gate represents all required steps to reach the attack goal. An `OR`-gate represents the different alternatives for reaching the goal. Leaves, respective BEs in FT nomenclature, can be assigned with either Boolean or continuous values. Examples for values are shown in Table 2.2.

Table 2.2. Examples for values assigned to BEs in ATs from [Schneier 99]

Boolean values		continuous values
possible	impossible	probability of success of a given attack
		likelihood that an attacker will try a given attack
expensive	inexpensive	cost

Several other authors also use FT-like structures to model attacks in a security analysis [Brooke 03, Rushdi 04, Rushdi 05, Helmer 02]. Essentially, they do the same as Schneier with his ATs, they just do not call it that.

Figure 2.6 shows an example AT. The example models a computer system that is accessible via Local Area Network (LAN) or Wireless Local Area Network (WLAN). Additional credentials are necessary for authentication. Access to the LAN is limited by Media Access Control (MAC) address filters that only allow access to clients whose MAC address is on a list. Access to the WLAN is restricted by Wi-Fi Protected Access 2 (WPA2) encryption. The AT was modeled using the tool ESSaRel as are the FT examples in Sect. 2.1.1. The BEs e_1 to e_5 represent the basic attack actions which are combined by an attacker as modeled in the AT to gain access to the system.

According to Schneier the following steps are required to create an AT and to conduct an analysis:

1. identify possible attack goals and create one tree per goal

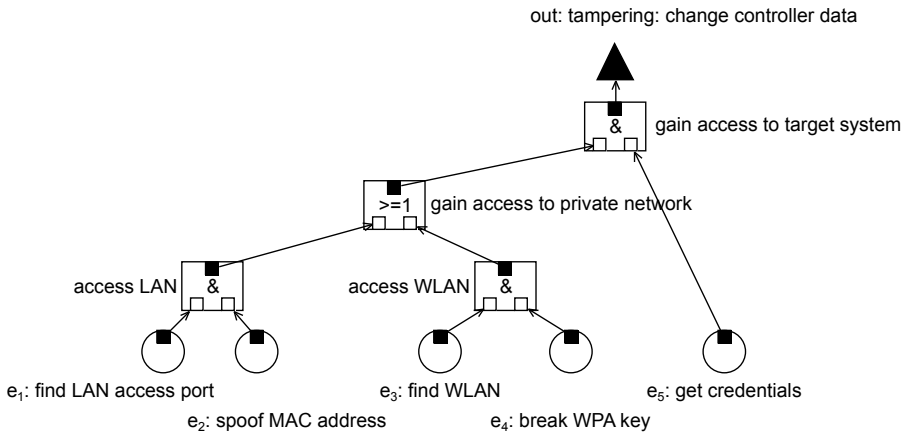


Fig. 2.6. An example Attack Tree (AT)

2. think of all attacks against a goal
3. refine the tree
4. add values (they can change over time)
5. compare and rank attacks

The main problem is to find the attack goals and to assign suitable values for the attributes. Results of the analysis can be an attribute value of the TE (attack goal) or a subtree fitting to a certain condition.

To rate the BEs, according to Schneier, knowledge about the potential attacker has to be taken into account. He mentions three example attacker types with different properties:

- bored graduate students who would not attempt illegal attacks such as bribery or blackmail,
- organized crime which is also prepared to attempt expensive attacks and to go to jail for an attack,
- or terrorists who are even willing to die to achieve the attack goal.

To determine attribute values of BEs, a certain knowledge about the attacker is required (a large intelligence service has greater resources than an angry neighbor). Examples for the use of ATs to analyze security are an analysis of the routing protocol of the Internet, the Border Gateway Protocol [Convery 02], or an analysis of vulnerabilities in Supervisory Control and Data Acquisition (SCADA) systems [Byres 04].

Attacker Modeling

In [Vigo 12] the author models attacker capabilities especially for Cyber-Physical Systems (CPSs). CPSs are systems that consist of sensors, actuators, and controlling components. Usually different system components are

networked to form a larger system. An example is a wireless sensor network in which different sensors send their data to one or more control units that evaluate it and send the data to other systems.

Vigo divides threats to CPSs into *physical* or *cyber* threats. The exploitation of physical threats require physical access to the system. Cyber threats on the other hand require only proximity to the transceivers of the system.

In contrast to other systems an attacker can simply influence the environment to affect the behavior of a CPS. Broadly distributed systems cannot be secured physically as well as monolithic systems. Additionally, the communication channels between system components have to be secured. Especially, a wireless sensor network can consist of a large number of single components. To keep costs relatively low, they are usually not tamper-proof which makes capturing a node easier.

Vigo models an attacker as a set of characteristics comprised of the attacker's location, capabilities, costs of attack actions, and needed resources. The paper also lists concrete physical attacks such as: removing a node from the network, reading/writing memory of a node, revealing content of the local control program, reprogramming the local control program, starving a node from energy, and inserting a new node into the network. The first four require direct physical access to a node, the last ones can be executed from a distance. Cyber-attacks such as: blocking messages, eavesdropping, or injecting new messages go against the communication between components (nodes).

A similar model for attackers of wireless sensor networks is described in [Benenson 08]. They describe attackers by two parameters: presence and intervention. Presence is the location of the attacker, and intervention represents his capabilities at that location. By combining these attributes they create attacker profiles.

Amenaza Technologies, a vendor of an Attack Tree modeling tool, quantify attacks using indicators such as cost of attack, probability of apprehension, probability of success, or technical difficulty [Ingoldsby 03, Ingoldsby 13]. Then they create different attacker profiles depending on the indicators used in the AT. Depending on the indicator values they can decide whether an attacker profile will be a threat to the system.

Attacks are evaluated similar to the calculation of the Risk Priority Number (RPN) in Failure Mode, Effects, and Criticality Analysis (FMECA) where weighted values from different properties are used (cf. Sect. 2.1.2).

Qualitative Analysis

As mentioned before, Attack Trees use the same tree structure and the same logic gates as Fault Trees. Therefore, the same qualitative analyses that depend only on the structure are possible. At first, the Minimal Cut Sets (MCSs) are determined (refer to Sect. 2.1.1 for a description how to determine MCSs). An MCS represents all actions that are necessary for an attack scenario to be successful. The size of an MCS determines its qualitative importance. The

smaller the MCS the more important it is to counter that specific MCS. From the list of MCSs the qualitative importance of BEs can also be determined. BEs that are part of a larger number of MCSs are more important than BEs that are only part of a smaller number of MCSs. Refer also to Sect. 2.1.1 for a description of the qualitative analysis of MCSs in Fault Trees.

Quantitative Analysis

Besides the modeling structure, Schneier also proposed some rules for quantitative analyses using ATs [Schneier 99]. Attributes of Intermediate Events and Top Events are calculated from BEs as in FTs. Schneier gives some examples for calculation rules: For Boolean attributes the value of a combination of two events is calculated as defined in equations 2.12 and 2.13 [Schneier 99].

$$\texttt{value}(AND) = \begin{cases} \texttt{possible} & , \texttt{if all child events are possible} \\ \texttt{impossible} & , \texttt{otherwise} \end{cases} \tag{2.12}$$

$$\texttt{value}(OR) = \begin{cases} \texttt{impossible} & , \texttt{if all child events are impossible} \\ \texttt{possible} & , \texttt{otherwise} \end{cases} \tag{2.13}$$

For the continuous valued attribute cost he proposes calculations as defined in equations 2.14 and 2.15.

$$\texttt{value}(AND) = \sum(costs\ of\ child\ events) \tag{2.14}$$

$$\texttt{value}(OR) = \min(costs\ of\ child\ events) \tag{2.15}$$

Several other researchers since then defined calculation rules for quantitative analyses in ATs. Based on Schneier's definition of ATs, Mauw and Oostdijk try to formalize ATs in [Mauw 06]. They use a slightly different nomenclature: An AT defines an *attack suite*, a collection of possible attacks. Basic Events are called *attack components*, the gates are *nodes* and a *bundle* is the equivalent to an MCS. In general, an attack suite is defined as a union of bundles. Or in other terms: Their ATs consist of a conjunction of MCSs and they show that every tree can be reduced to such a normalized tree. Mauw and Oostdijk try to formalize ATs as general as possible. They define general tree reduction and rearranging rules. For attribute calculation they also define general rules: For every attribute an operation for conjunction and disjunction of events has to exist so that attributes for TEs can be calculated. They do not define a negation operation (see also Sect. 2.1.1 for an explanation why). Some example operations for different attributes are shown in Table 2.3.

So far, the quantitative evaluations all use relatively crude values. The following works try to use more precise estimates in the evaluation. Buldas et al. propose a quantitative analysis by calculating the risk of an attack by equation 2.16 [Buldas 06]. Where τ is a security threat, $\Pr[\tau]$ the probability of τ and $\text{Loss}[\tau]$ the loss by τ.

Table 2.3. Operations for conjunction and disjunction for different attributes

attribute	domain and operations
cost of the cheapest attack	($\mathbb{N}$, min, +)
maximal damage	($\mathbb{N}$, max, +)
minimum skill required	($\mathbb{N}$, min, max)
possibility of attack	($\mathbb{B}$, $\wedge$, $\vee$)
special equipment required	($\mathbb{B}$, $\vee$, $\wedge$)

$$\text{Risk} = \sum_{\tau} \Pr[\tau] \times \text{Loss}[\tau] \tag{2.16}$$

Based on the calculation of the risk, they decide to put a countermeasure in place if either the risk is too high to be tolerated, or the cost of the measure is lower than the risk difference. Otherwise a countermeasure would not be reasonable. According to them, it is usually very hard to come by accurate probabilities, especially for targeted attacks that only happen once. Even if there is experienced data, companies are very reluctant on providing them to others out of fear for the loss of reputation towards their clients.

This might change in Germany in the future because of the "IT-Sicherheitsgesetz" (IT security law) which was passed in July 2015 [ITSG 15]. According to that law, providers of critical infrastructure have to report serious attacks to the BSI[2].

On the other hand, there are expert estimates for typical attacks. From these estimates for basic attacks, they want to determine probabilities of more complex attacks using ATs. They limit their effort to rational attackers who only go through with an attack if there is a positive outcome, and if so, they choose the one with the highest outcome. Therefore, they try to estimate the possibility of an attack using a cost-benefit calculation from the attacker's point of view. Their calculation of the outcome is shown in equation 2.17. They criticize that in previous works usually only one parameter is used in the trees and therefore the decision of an attacker to attack is not modeled very accurate. The attacker's decision is modeled with the parameters: attacker's gain, probability of success, probability of getting caught and possible penalties (listed in Table 2.4).

$$\text{Outcome} = p \times \text{Gains} - \text{Costs} - p \times \pi^{+} - (1-p) \times \pi^{-} \tag{2.17}$$

For two nodes with the parameters $\text{values}_i = (\text{Costs}_i, p_i, \pi_i^{+}, \pi_i^{-})$, and $(i = 1, 2)$ the following computation rules shown in equations 2.18 and 2.19 are established.

[2] Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)

Table 2.4. Parameters used to calculate the outcome of an attack according to Buldas et al. [Buldas 06]

parameter	description
p	probability that the attack is successful
Gains	the attacker's gain
Costs	the costs of the attack (material, bribes, ...)
π^+	penalty if the attack is successful (penalty × probability to be caught)
π^-	penalty if the attack is unsuccessful (penalty × probability to be caught)

$$\text{values}(\texttt{AND}) = \begin{pmatrix} \text{Costs} = \text{Costs}_1 + \text{Costs}_2, \\ p \quad = p_1 \times p_2, \\ \pi^+ \quad = \pi_1^+ + \pi_2^+, \\ \pi^- \quad = \frac{p_1(1-p_2)(\pi_1^+ + \pi_2^-) + (1-p_1)p_2(\pi_1^- + \pi_2^+) + (1-p_1)(1-p_2)(\pi_1^- + \pi_2^-)}{1-p_1p_2} \end{pmatrix} \tag{2.18}$$

$$\text{values}(\texttt{OR}) = \begin{cases} \text{values}_1, \text{ if Outcome}_1 > \text{Outcome}_2 \\ \text{values}_2, \text{ if Outcome}_1 \leq \text{Outcome}_2 \end{cases} \tag{2.19}$$

Whereas Outcome_i for $(i = 1, 2)$ is calculated according to equation 2.17. π^- is the average penalty if at least one of the child-attacks was not successful. Costs and penalties of node i can be combined to the value

$$\text{Expenses}_i = \text{Cost}_i + p_i \times \pi_i^+ + (1 - p_i) \times \pi_i^-$$

Jürgenson and Willemson do generally the same in [Jürgenson 08]. But they criticize that the calculations from [Buldas 06] are violating the general calculation rules from [Mauw 06]. To calculate the value of an `OR`-gate, the outcome of both inputs has to be compared for which the gain of each sub-attack is needed. Because this value is usually not available, Buldas et al. use the gain of the overall attack which is too high for this calculation. Also, they only consider the case that only one path of an `OR`-gate is executed. But Jürgenson and Willemson claim that in practice several attack paths are tried as long as they are feasible for the attacker. Therefore Buldas et al. underestimate the outcome of an attack and by this the attack probability. In [Jürgenson 08] they propose an improved approach that does not violate the criteria from [Mauw 06] but with a higher computational complexity. The computation rule for the outcome in [Buldas 06] is a lower bound for the maximum outcome as computed by Jürgenson and Willemson.

To summarize [Buldas 06] and [Jürgenson 08], they both do cost-benefit calculations from the attacker's point of view. Together with [Mauw 06], they propose a formal basis for calculations in attack trees. But they do not mention

how to obtain the probabilities for an attack to be successful other than by rough expert estimations. Without those, their approach is not feasible.

Extensions of Attack Trees and Related Techniques

As for FTs, there are extensions of ATs as well. Defense Trees (DTs) extended ATs with the ability to model countermeasures [Bistarelli 06]. BEs in a DT are annotated with a set of suitable countermeasures. They model DTs in Disjunctive Normal Form (DNF) which means DTs depict the set of possible attacks, or the combination of all MCSs in FT terminology. Their approach includes qualitative as well as quantitative evaluation of DTs. They use costs as a measure. Countermeasures are rated by a Return on Investment (ROI) from the defender's point of view and a Return on Attack (ROA) from the attacker's point of view. Interdependencies between sub-attacks are considered by rating the MCS/whole attack scenario instead of every BE by itself.

Previously mentioned Generalized Fault Trees (GFTs) (cf. Sect. 2.1.1) are another example of safety modeling techniques used to model security [Codetta-Raiteri 13]. Using GFTs repair and recovery mechanisms, and dynamic behavior can be modeled. This can lead to more accurate attack models, and if attack probabilities are available, the model can be calculated more accurate. But, the ATs based on GFTs are getting more complex. For an analysis they are transformed into Generalized Stochastic Petri Nets (GSPNs) and analyzed using the usual Petri net analysis methods [Marsan 95]. Therefore, they have the same advantages and drawbacks as Petri nets. Analyses can be very detailed, but the analysis effort increases exponentially with the size of the model. GFTs still depend on probabilities for attack events which are difficult to obtain.

There are state-based approaches that allow modeling dynamic behavior in attacks. Petri Net Attack Modelings (PENETs) [Pudar 09] is an extension of ATs with dynamic constructs such as time dependencies, or defense and repair mechanisms. They are analyzed by converting them into an equivalent Petri net.

In security attack statecharts [Ariss 11] states correspond to successful steps necessary to execute an attack. Events trigger transitions between the states. The decisions of an attacker are represented using statechart representations of logic gates such as `AND`, `OR`, and `Priority-AND`. It is possible to model conditional cycles, preconditions, concurrency, or collaboration.

Sheyner et al. describe attack graphs in [Sheyner 02] to analyze networks. The network is modeled as a finite state machine. Security properties are specified and state transitions correspond to atomic actions an attacker performs to violate these properties. The modeled state machines are evaluated using a model checker to find the most critical attacks, or the ones a defender has to prevent to counteract attacks. Other metrics for attack graphs are the shortest path to an attack goal or the number of paths [Idika 10].

McDermott models attacks using Petri nets as so-called attack nets in [McDermott 00]. It is basically a Petri net representation of ATs with the additional ability to model sequences.

ADversary VIew Security Evaluation (ADVISE) [LeMay 11] is an approach to generate an executable model, the attack execution graph, that represents the steps an attacker is likely to take to reach his attack goal. It is also a state-based model that extends the attack graph concept with attacker profiles. By executing attack steps, an attacker gains access to system components or reaches the goal. The graph also includes the skills and knowledge an attacker needs to execute an attack step. Attacker profiles are defined which include the attacker's attack preferences, attack goals, and attack skills. The choice which attack steps to take depends on cost, payoff, and probability of detection, and is calculated by an attack decision function. From an initial state the path with the most attractive attack steps is calculated which represents the most likely attack sequence.

State-based approaches are able to create more precise models than ATs, but they all have the problem of the state space explosion in larger systems. Therefore they are more suited to model partial aspects of a system in detail than the whole system. For the overall view of a system ATs are better suited, and they can be extended by state-based models for more detailed models.

2.2.2 Threat Modeling

The term *threat modeling* was coined by Microsoft as an "attack-focused analysis activity used to find security flaws in software" [Dhillon 11]. Threat modeling is effective to find classes of coding issues that stem from architectural or technology-selection decisions. It is recommended to apply threat modeling during early development phases, namely the architecture or design phase. The threat modeling process is described in four steps according to [Dhillon 11]:

1. Creating an annotated data flow diagram
2. Identifying and analyzing threats, guided by the STRIDE approach or a threat library
3. Assessing technical risks of threats
4. Mitigating threats to reduce risk

From an attacker's point of view data flows are interaction points with the system. Attacks especially are directed towards data flows. Data Flow Diagrams (DFDs) that are used for threat modeling should be annotated to simplify the following analysis steps. Useful annotations according to [Dhillon 11] are:

- different privilege levels
- embedded components
- programming languages
- critical security functions

- network and local data flows
- data flow type (HTTP, SQL, API call, ...)
- authenticated data flows
- encrypted and/or signed data flows/stores
- sensitive information in data stores
- how sensitive information is intended to flow through the system

DFDs consist of data flows, data stores, processes, interactors, and trust boundaries [Hernan 06]. Microsoft also provides general rules for the creation of data flow diagrams for threat modeling:

- there should be no "magic" data sources or sinks
- model all data transports, meaning there should always be a process that reads and writes data
- similar elements in one trust boundary should be collapsed for simpler modeling
- components on either side of a trust boundary should not be modeled simultaneously but separately with the respective other side as an external interactor

After developing the DFDs, the next step is to identify threats to system assets. In Microsoft's Security Development Lifecycle they use the STRIDE approach [Howard 06, Microsoft 10]. It is an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege (STRIDE). For each security property there is a specific threat which is shown in Table 2.5. The individual security properties are described in Table 2.6.

Table 2.5. Threats and security properties from [Hernan 06]

threat	security property
Spoofing	authentication
Tampering	integrity
Repudiation	non-repudiation
Information disclosure	confidentiality
Denial of service	availability
Elevation of privilege	authorization

For the STRIDE approach the system is divided into relevant components, each component is analyzed, and threats found are mitigated by additional measures. Table 2.7 shows which DFD elements are susceptible to which security threats.

For the analysis a security expert goes through the list of components and checks each for the threats listed in Table 2.7. The analyst has to find the individual threats fitting to the system in question. This step relies heavily on experience and can never be complete. But there are tools

Table 2.6. Descriptions of security properties from [Hernan 06]

property	description
confidentiality	data is only available to the people with the appropriate clearance
integrity	data and system resources are only changed in appropriate ways by appropriate people
availability	systems are ready when needed and perform acceptably
authentication	the identity of users is established
authorization	users are explicitly allowed or denied access to resources
non-repudiation	users cannot perform an action and later deny performing it

Table 2.7. Threats affecting data flow diagram elements according to [Hernan 06]

element	spoofing	tampering	repudiation	information disclosure	denial of service	elevation of privilege
data flows		X		X	X	
data stores		X		X	X	
processes	X	X	X	X	X	X
interactors	X		X			

that ease that task. Chapter 22 in Microsoft's "The Security Development Lifecycle" [Howard 06] has a number of threat tree patterns for the security properties of DFD elements. There are pre-built threat trees with design questions about all elements that help to understand the pattern. Common types of attacks are listed in libraries such as "Writing Secure Code" by Howard and LeBlanc [Howard 02] or the "Common Weakness Enumeration [MITRE 14].

The STRIDE taxonomy is general enough to encompass most attacks on software. But the downside is, to apply it a substantiated knowledge about software security is required. So, if normal developers should be able to apply threat modeling to their development process, they need a technique that requires less security knowledge. A simplification of the analysis is to focus threat modeling efforts on interactions between elements in the data flow diagrams. A threat library was developed that, at the time of the paper [Dhillon 11], included around 35 entries which are maintained and updated by security experts. It consists of the minimum considerations that are required for the systems that are developed by their company. A threat library can never be complete but it can be a good baseline of actions. Major criteria for including entries in the threat library are:

- identifiable during design time?
- at least moderate security risk?
- can actionable mitigation guidance be provided?

An entry of the library has the following fields:

- applicability field to show whether the issue is relevant
- threat description
- examples illustrating implications
- baseline risk with variation based on modeled system
- detailed prescriptive mitigation for the threat

After identifying the threats to the system, the risk resulting from them has to be assessed. In [Dhillon 11] they only assess the technical risk, business risk is not assessed because developers usually do not have the necessary knowledge to do that. They use a number of yes/no questions that are translated into numerical scores. The scale of these scores is divided into four intervals which then represent qualitative rankings of "critical", "high", "medium", or "low". The qualitative ranking makes it easier for developers to decide whether a threat has to be mitigated for a particular system.

For each entry in the threat library of [Dhillon 11] there is a description of mitigation strategies. This way developers do not have to solve the mitigation problem for known threats every time again.

2.3 Combinations of Safety and Security Analysis

As discussed in Sects. 2.1 and 2.2 there are several approaches that use Fault Trees (FTs) or derivatives thereof to analyze security. Most of them aim simply in using the FT method for security analysis, some qualitatively such as [Schneier 99, Helmer 02, Brooke 03], and some quantitatively such as [Rushdi 05, Buldas 06, Mauw 06, Jürgenson 08, Codetta-Raiteri 13]. The aim of this thesis is to analyze the effects security attacks may have on system safety. The previously mentioned approaches only analyze security on its own. They do not model interdependencies between security and safety concerns. In this section existing approaches that try to integrate safety and security analysis are discussed.

Eames and Moffett investigate how to integrate safety and security on the requirements specification level [Eames 99]. They argue that to obtain consistent and complete requirements safety and security requirements have to be defined simultaneously. A comparison of the safety and the security risk analysis processes is done and shows that safety and security risk analysis can be divided into common steps. These commonalities are shown in Table 2.8. From this they conclude that it is possible in general to do a simultaneous safety and security risk analysis. This claim can be supported by combined safety-security analyses using similar or the same techniques for safety and security, as will be shown by the following approaches. Sommerville argues that safety and security requirements should not even be treated as special requirements, but both should be integrated into the requirements engineering process because they have influence on the whole system [Sommerville 03]. Saglietti sketches an idea to extend FTs with causes from security attacks

Table 2.8. Comparison of safety and security risk analysis processes according to [Eames 99]

risk analysis processes		
safety	security	commonalities
functional and technical analysis	asset identification	system modeling
qualitative analysis	qualitative analysis	qualitative analysis
quantitative analysis	quantitative analysis	quantitative analysis
synthesis and conclusions	countermeasure evaluation	defining requirements

in [Saglietti 06]. A broader overview of the efforts of safety, security and safety-security co-analysis provides [Paul 15].

In the following, the first approaches are about combinations of FTs and Attack Trees (ATs) where ATs are attached to FTs as additional causes. After that, the next approaches handle ratings of security and safety risks in a combined manner. Next is Failure Mode, Vulnerabilities and Effects Analysis (FMVEA), an approach comparable to the one in this thesis, but based on Failure Mode and Effects Analysis (FMEA) instead of Fault Tree Analysis (FTA). The last section is a short overview over other analysis methods that combine safety and security analysis.

2.3.1 Integration of FTs and ATs

In this section two approaches that are combinations of FTs and ATs are shown. In both ATs are attached to FTs as additional causes. The first one shows the combination of two models, one FT and one AT. The second approach describes rules for the assignment of ratings and the calculations for a quantitative evaluation. A related approach is mentioned last, it combines FTs with Markov processes.

In [Guo 10] the authors describe an approach to identify security-safety requirements simultaneously, and they apply it as a case study to an off-road robot. They implement a combination of FTs and ATs as part of the derivation of security-safety requirements. Based on given functional requirements, a safety as well as a security analysis is conducted. For both, first an FMEA and then, based on the results, an FTA is done. They argue that by using the same techniques the analyses can be combined easier. Between safety and security terms they define equivalences such as *attacker – failure* modes and *weaknesses – failure causes*. The causal chain of the security FMEA leads from a security attack to a safety effect. The FTA yields a tree for the safety aspect and a tree for the security aspect. Those trees have to be merged to gain a combined tree for the final analysis. The safety tree is searched for events that match a Top Event (TE) of the security tree, and then the security tree is attached to this event as a subtree. The problem of matching trees is described in more detail in Sect. 3.3.1. The analysis results in necessary countermeasures that have to be built into the system. These countermeasures are gathered as additional system requirements.

The article only describes exemplarily the building of the FTs. A detailed analysis of the resulting trees is not presented. At least a qualitative analysis is straightforward as the trees are standard FTs which can be analyzed using Minimal Cut Sets (MCSs) as described in Sect. 2.1.1.

This next approach is about the calculations for a quantitative evaluation of a combination of FTs and ATs. Fovino et al. describe in [Fovino 09] an approach to integrate ATs and FTs to assess the influence of security-relevant events on safety TEs. They try to formalize ATs similar to [Mauw 06, Buldas 06, Jürgenson 08] to be able to calculate ratings for the TE of a combined tree. They call the root of an AT the *goal* of an attack. They partition Basic Events (BEs) into assertions and vulnerabilities. Assertions are defined as conditions under which an attack is possible. Vulnerabilities can be exploited by an attacker. For vulnerabilities the attributes name, type, description, vulnerability reference, list of affected components, countermeasures list, severity, likelihood, and resources required are recorded. All possible actions in the context of the system are called operations. Operations are conjunctions of assertions and vulnerabilities in the tree. Goals of ATs may appear as events in an FT. ATs and FTs can be integrated at that point via an `OR`-gate to which the FT event and the AT goal are connected (similar to [Guo 10]). Also similar to [Guo 10] they combine two existing trees instead of expanding one tree (see Sect. 3.3.1 for a discussion of that problem). It is mentioned that there could also be Intermediate Events (IEs) as side effects of an attack in an AT that correspond to other events in the FT. These are also merged using the IE the same way as a goal. They call that combined tree an Extended Fault Tree (EFT). Under the precondition that probabilities for all events exist in the EFT, security as well as safety-related, they use the usual rules of probabilistic Boolean logic to calculate probability of the safety-related TE of the FT.

The basic idea is very similar to the approach of this thesis, except for the following main differences: First, they say attack tree probabilities are difficult to determine, and they use likelihood values. Later they use the calculation rules as if there were probability values for all events. But as they state before, probabilities for attack trees are mostly unavailable, and therefore the calculation rules are merely theoretical. No qualitative analysis of the EFT is described that in some way takes into account that there are events of different types (deliberate attacks and random faults).

Boolean logic Driven Markov Processes (BDMPs) have a similar syntax to FTs and ATs. They are extended by triggers that model dependencies between events [Bouissou 03] and use `AND`, `OR`, and `PAND`-gates. If the origin of a trigger is true (the event occurs), the target of the trigger gets activated. Only activated events may become true. Triggers may activate or deactivate parts of a tree. Thereby it is possible to model simple dependencies, sequences, detections, reactions, and repair actions. Leaves of the tree (BEs) are modeled by Markov processes that represent the component's behavior depending on activation. BEs can model both failures in operation and failures on demand

with possible repair actions. BDMPs are able to model the same dependencies as Dynamic Fault Trees (DFTs), Repairable Fault Trees (RFTs), or other state-based approaches (cf. Sect. 2.1.1). In [Pièvre-Cambacédès 10a] the authors add security BEs such as *attacker actions*, *timed security events* and *instantaneous security events*. If an attacker action is activated, the attacker attempts an attack step. The timed security event models an event which influences an attack, but it is not under the attacker's control. The time to success for both is exponentially distributed. The instantaneous security event happens with probability γ directly after activation. Using those events BDMPs can be used to model complex attack sequences with interdependencies. In this way they are comparable to Defense Trees (DTs), an extension of ATs including defense mechanisms and countermeasures (cf. Sect. 2.2.1). The authors also combined safety and security models to model interdependencies between them [Pièvre-Cambacédès 10b]. They are able to model similar structures as described in [Guo 10] and [Fovino 09] such as adding security events as additional causes for safety failures. Because of the generalized model they also can model security failures that depend on safety causes. Results of the analysis depend on quantitative input data that are difficult to obtain for security events as mentioned earlier. Qualitative results are the possible failure sequences.

This approach is very similar to the one in [Fovino 09] and has the same drawbacks. It depends on attack probabilities respective attack rates to be useful, and they are hard to obtain. The qualitative results, the failure sequences, may be more detailed than in ATs. The usefulness depends on the required level of detail of an analysis. For an overview of a system, FTs and ATs are better suited. For more detailed parts of a system the greater effort of creating Markov Processes may pay off.

The next section will cover some approaches that use qualitative ratings instead of probability values, and they describe how to obtain the ratings.

2.3.2 Security Analysis with Safety Implications

The approaches of this section are mostly about how to obtain ratings for security and safety risks in a combined manner. How they obtain the scenarios they analyze is not in every case described. Some also use FTA to analyze their examples.

Bloomfield et al. conduct a security analysis of the European Railway Traffic Management System (ERTMS) including an impact analysis of the found vulnerabilities on the safety of the system [Bloomfield 12]. Attacks were rated according to several criteria:

- type of access required to exploit a vulnerability
- level of technical sophistication required to exploit a vulnerability
- type of failure caused by a successful attack
- scale of effect for a successful attack

- scalability of the attack from the attacker's perspective
- type of impact caused by a successful attack
- types of mitigation strategies that are possible
- level of difficulty for implementing each mitigation

For each criterion a qualitative rating along the lines of {HIGH,Medium, low} is used (see Table 2.9).

Table 2.9. Ratings of attack scenario criteria as used in [Bloomfield 12]

minimum access required	technical sophistication	type of failure	scale of effect
REMOTE ACCESS	LOW	LOSS OF LIFE	GLOBAL, NATIONAL
Access to infrastructure, but not the system itself	Medium	Denial of service	Regional
physical access to the system	high		local

scalability of the attack	type of impact	mitigation strategies	ease of mitigation
HIGH	SAFETY CRITICAL, ECONOMIC		HARD
Medium	Political	Reactive	Medium
low	psychological	preventive	easy

They do not try to use a compound rating, but instead they argue that the different ratings show the trade-offs a decision maker has to deal with. These ratings can be used to make more informed decisions.

In essence this is a security analysis in which safety implications are taken into account. The approach is directly inverse to the approach of this thesis. After the rating no further analysis is described, neither qualitative nor quantitative.

Also the same analysis direction as [Bloomfield 12], this approach from Casals et al. describes a security analysis that takes safety implications into account [Casals 12]. It is another example of an approach that uses qualitative ratings for the attack likelihood.

They describe an application of security risk assessment to avionics systems. As risk needs to be quantified to avoid unnecessary costs, they propose a quantitative risk assessment framework. Their methodology consists of six steps:

1. context establishment

2. preliminary risk assessment
3. vulnerability assessment
4. risk estimation
5. security requirements
6. risk treatment

The interesting step for this thesis is the risk estimation. The attack likelihood is rated using five likelihood levels: *frequent*, *probable*, *remote*, *extremely remote*, and *extremely improbable*. The level is determined by a combination of ratings for exposure of the asset to threats and attacker capabilities. Values for exposure and attacker capabilities have to be determined by experts. The risk, in turn, is determined by a combination of likelihood and safety impact. For the safety impact they also use five levels: *no safety effect*, *minor*, *major*, *hazardous*, and *catastrophic* [ED-202 10].

The authors use an FT structure to model the combinations of attacks that lead to the system failure (in this case an aircraft that crashes during take-off). Just as the authors of [Bloomfield 12], Casals et al. conduct a security analysis that includes the effects of security attacks on system safety. This can extend, but not replace, an existing safety analysis. A complete safety analysis is necessary nevertheless.

Reichenbach et al. also describe a security risk assessment approach that is extended to include the impact on system safety [Reichenbach 12]. They propose a combined approach for safety and security analysis to have a more consistent, less redundant analysis. Based on Threat Vulnerability and Risk Assessment (TVRA) [ETSI TS 102 165-1 11], they add the Safety Integrity Level (SIL) to get a rating of an attack that includes the impact on safety. Factors affecting the risk are time, expertise, knowledge, opportunity, equipment, asset impact, and intensity. The likelihood of an attack is based on the attack potential value which is based on time, expertise, knowledge, opportunity, and equipment. The impact of an attack depends on asset impact, attack intensity, and SIL. All these values have to be determined by experts during the analysis.

This is more a risk estimation approach than an analysis technique. It can be used after the risks are identified by other methods.

In this section the approaches used extensions of security risk assessment approaches to also assess the risk on the system under study. They cannot be used as standalone techniques but they may be used together with techniques such as FTA or FMEA that provide the failure scenarios. The next section on the other hand shows an approach that includes the scenario finding as well as the evaluation of ratings of risks.

2.3.3 Failure Mode, Vulnerabilities and Effects Analysis (FMVEA)

Schmittner et al. extend FMEA (see Sect. 2.1.2) to include vulnerabilities and attacks as additional causes in the analysis [Schmittner 14]. They describe a

combined analysis of safety and security based on the FMEA cause-effect chain. A generic set of security based failure modes called *threat modes* based on the STRIDE classification (see Sect. 2.2.2) is proposed. Table 2.10 shows corresponding FMEA and FMVEA elements. To the cause-effect chain of FMEA vulnerabilities, threat agent, threat mode, threat effect, and attack probability are added as additional elements. Figure 2.7 shows the cause-effect chain of FMVEA. The dashed areas show the security respective safety specific elements. The gray background indicates that probabilities are determined for failure causes as well as attacks which are composed of vulnerabilities and threat agents that exploit them.

Table 2.10. FMVEA elements and their correspondents in FMEA [Schmittner 14]

FMEA elements	FMVEA elements
failure cause	vulnerability
random event	threat agent
failure mode	threat mode
failure effect	threat effect
failure probability	attack probability

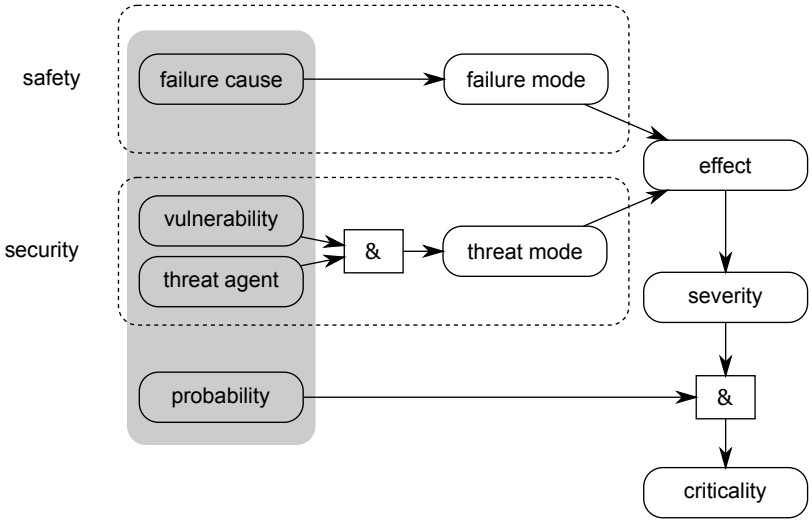


Fig. 2.7. The cause-effect chain of FMVEA from [Schmittner 14]

Like the approaches from the previous section, Schmittner et al. use a qualitative rating to describe the attack probability. The attack probability is calculated as the sum of *threat properties* and *system susceptibility*. The

value of the threat properties is determined by the sum of *motivation* and *capabilities*. System susceptibility is determined by the sum of *reachability* and *unusualness*. Schmittner et al. define scales with three values for all of these properties. Table 2.11 shows the properties and the corresponding rating values. The values of the resulting attack probability can then be interpreted

Table 2.11. Ratings of the properties necessary to determine the attack probability according to [Schmittner 14]

	property	rating values
threat properties	motivation	1 = opportunity target
		2 = mildly interested
		3 = main target
	capabilities	1 = low
		2 = medium
		3 = high
system susceptibility	reachability	1 = no network
		2 = private network
		3 = public network
	unusualness	1 = restricted
		2 = commercially available
		3 = standard

as *low* for values 4–6, *medium* for values 7–9, and *high* for values 10–12.

This is an approach that could be used in combination to the approach of this thesis. As with FMEA and FTA where the FMEA is used to find failure modes that are analyzed further using FTA, FMVEA could be used to find failure modes (and also some of the causes and effects) which then are analyzed further using the Security-enhanced Component Fault Trees (SeCFTs) of this thesis.

2.3.4 Other Related Approaches that Combine Safety and Security Analysis

In this section other approaches that in some way combine safety and security analysis are mentioned in a short overview.

In [Roth 13] the approach proposed in this thesis is extended by a security analysis from the attacker's point of view using FMECA-like ratings.

[Dobbing 06a, Dobbing 06b] describe a standard that integrates safety and security on an assurance case level. This standard is based on British Defense Standard [MoD DS 00-56 07] and Common Criteria [ISO/IEC 15408 12]. It was developed under the SafSec study of the British Ministry of Defense. However, it does not claim to be necessarily sufficient for safety or security certification, but it provides a framework how safety and security standards can be applied more cost efficient and with reduced risk.

Großpietsch and Silayeva propose architectural means to achieve safety and security in [Grosspietsch 04]. They analyze their approach using a Byzantine fault model to integrate security failure models into time-dependent reliability analysis using different attacker models. No concrete attacks are analyzed, but the effects of failed elements on the overall system due to attacks are analyzed.

[Lanotte 03] integrates safety and security analysis based on timed automata and symbolic model checking.

There exist only a few publications that try to combine FTs and ATs, despite of the similarities, to combine safety and security analyses. This thesis proposes an approach that enhances safety analysis to encompass security risks as additional causes for safety failures. Some researchers are doing the opposite approach by including safety effects into security analysis. There is work done on the level of standardization to include security into safety standards such as [IEC 61508 10]. But mostly security and safety are aspects of systems that are still analyzed separately and sometimes even in isolation.

2.4 Critique of Quantified Security Properties

In safety analysis it is common to conduct a quantitative analysis if more precise statements about a system are required. This practice has been tried to adapt to security analysis. Although quantitative approaches for security assessment have been intensively studied, quantification is still in its infancy [Förster 10].

In [Verendel 09], the author provides a representative overview of works between 1981 and 2008 containing security models, security metrics, or security frameworks for a quantitative analysis of system security. A few of them have been discussed in Sect. 2.2. According to him, the validity of most methods is unclear, and most of the methods were not successfully used in a real life analysis. A lot of the works Verendel studied for his survey claims that quantitative analysis of security is necessary. But that depends on the correct representation of security with quantitative information. In security analysis usual assumptions are that security related events are stochastically independent, that attackers are rational or choose the optimal way, or that security properties do not change over time. In general, these assumptions are not correct. Most of the surveyed work on quantification of security is not validated in a repeatable way. Based on the survey, Verendel states that at time of the paper (2009) security cannot be represented correctly with quantitative information.

Earlier described approaches such as [Mauw 06, Buldas 06, Jürgenson 08, Fovino 09, Piètre-Cambacédès 10b] illustrate that finding. They define calculation rules for probabilities of security events similar to the ones for safety analysis in FTs, but they do not discuss the fact that such probabilities are

usually not available. Without input values their whole calculation model is simply theoretical.

According to [Brooke 03], the major benefit of fault tree analysis is the identification of the relationship of events, not a quantitative evaluation of security. Quantitative analysis often has no additional value.

2.5 Conclusion

In the beginning of this state of the art, Fault Trees (FTs) are described that are suited to find and model causes and combinations of causes of failures. In combination with Failure Mode and Effects Analysis (FMEA) and Hazard and Operability Study (HAZOP) to find the failure modes, they are a well-established method in industry to analyze safety. They are also easily extendable to add the ability to model dynamic behavior using Markov processes. The extension to Component Fault Trees (CFTs) allows the component-wise modeling of larger systems.

The following sections show that Attack Trees (ATs) are a good match for a combined analysis. Even if they are not perfect to model security in much detail in their basic form, they can be extended to the needs of a modeler, just like CFTs. Comparable to safety analysis, ATs can be combined with Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) or STRIDE to find vulnerabilities. The approaches that directly provided the possibility to model dynamic behavior for both safety as well as security, were all more complicated in both modeling and analysis. Experience shows that CFTs in industry are hierarchically built. Only if it is necessary, more detailed models are built. So, using one of the dynamic modeling techniques such as Boolean logic Driven Markov Processes (BDMPs) to model the whole system creates much unnecessary overhead.

Also, using much too formal models for security does not help either because usually the necessary input data is not available. If such data would be available however, it can also be used without much additional effort in an AT.

3

A Comprehensive Safety Modeling Approach

In the introduction of this thesis it was established that networked embedded systems (or sometimes called Cyber-Physical Systems) introduce security vulnerabilities into systems that were not present in isolated systems. Embedded systems mostly are analyzed with regard to safety, but not to security. So the vulnerabilities newly introduced by networking systems together are not taken into account.

The idea of this thesis is to augment safety analysis based on Component Fault Trees (CFTs) by adding security concerns modeled as Attack Trees (ATs) that have an influence on the safety of the analyzed system. Fault Trees (FTs) in general, or CFTs in particular here, are a well-known tool to conduct a safety analysis. ATs are an established technique for security analysis. The motivation of this approach is to use established techniques with which engineers are familiar, and to extend these. First ideas of the proposed approach were described in [Förster 10] and [Roth 13].

In this chapter, the modeling part of the approach will be described. First, the extensions will be embedded in a usual analysis process for safety critical systems (Sect. 3.1). After that, different variants are discussed to combine FTs or rather CFTs with ATs in Sect. 3.2. Section 3.3 shows two approaches to create a combined tree, and Sect. 3.4 describes the selected approach in detail.

3.1 Overall Modeling and Analysis Process

Safety analysis using Fault Tree Analysis (FTA) is a process of several steps. The overall process is the same, independent of the use of FTs or CFTs.

Standards such as [IEC 61025 06] or [IEC 60300-3-1 05] recommend a combination of inductive and deductive techniques to minimize the potential of omitted failure modes. Inductive techniques as Failure Mode and Effects Analysis (FMEA) [IEC 60812 06] or Hazard and Operability Study

(HAZOP) [IEC 61882 01] can be used to find the Top Events (TEs). Deductive techniques as FTA [IEC 61025 06] are used to refine the analysis and to find causes and moreover combinations of causes that lead to the TE. The resulting tree is used to conduct qualitative and quantitative analyses. Figure 3.1 shows the general process in three steps. Strictly speaking, a CFT is not a tree but a Directed Acyclic Graph (DAG). To simplify naming matters CFTs will also be referred to as "trees" in the course of this thesis.

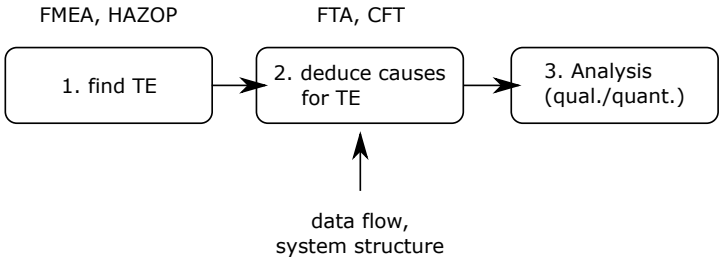


Fig. 3.1. The safety analysis process according to IEC 60300–3–1

The approach to introduce security aspects into safety analysis proposed in this thesis is based on CFTs. It extends the process described earlier by an additional step and modifies the analysis step. Figure 3.2 shows the extended analysis process. After developing the CFT, it is extended by possible security attacks as additional causes that also could lead to the safety-related TE. Those security attacks are found by analyzing data flow and interface specifications because most attacks are made at communication interfaces (cf. Sect. 2.2.1). Techniques to find security vulnerabilities that may lead to attacks are STRIDE (cf. Sect. 2.2.2) or Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) (cf. Sect. 2.3.3).

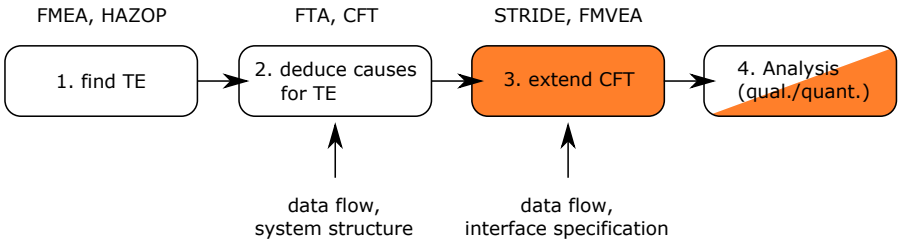


Fig. 3.2. The extended safety analysis process

3.2 Discussing Different Combinations of CFTs and ATs

Before describing the extension of CFTs, this chapter will discuss the issue of how safety and security concerns can be combined in an analysis. Depending on the focus of the analysis, two variants for combinations are possible: An AT as the main tree with additional branches from a CFT, or a CFT as the main tree with additional AT branches.

If the focus lies on security analysis, the TE will be an attack goal and the underlying tree will be an AT. In this AT there can be branches of CFTs if component failures are necessary for an attack to succeed (see Fig. 3.3a).

If, on the other hand, the focus lies on safety analysis, the TE will be a system failure and the underlying tree will be a CFT. In this CFT, there can be events that may also be caused by security attacks, hence in such a CFT there can be branches of ATs (see Fig. 3.3b).

Nesting of both variants is also imaginable (see Fig. 3.4). Both variants will be described in more detail in the next sections. But first, some definitions are needed to distinguish events of different types in the combined trees.

Definition 22 (Safety Event) *The term safety event is used for a Basic Event (BE) which occurs due to a random fault in the system [Förster 10]. The BEs in standard CFTs are all safety events.*

Definition 23 (Security Event) *The term security event is used for a Basic Event (BE) which is triggered from outside the system by a malicious attacker (or a random fault) [Förster 10]. The BEs in standard ATs are all security events.*

3.2.1 An AT Extended by a CFT

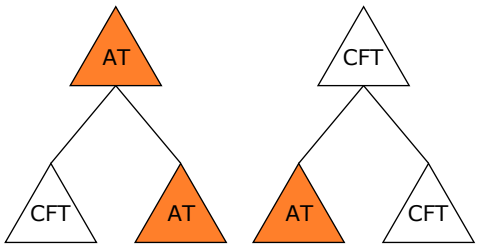
Figure 3.3a shows an AT with an attached CFT branch. The TE of this combination is an attack goal that is refined for security analysis. The interface between the AT and the CFT part is an event that can be caused by a security event as well as by a safety event. This is basically a normal AT as some attacks might depend on failures in the targeted system.

This combination variant is aiming at security analysis, and it is not examined further here because the goal of this thesis is to improve safety analysis.

3.2.2 A CFT Extended by an AT

Figure 3.3b shows a CFT with an AT branch attached to it as additional cause for the TE. The interface between the CFT and the AT branch is an event in the CFT that may be caused by a safety event as well as by a security event. This security event does not have to be the goal of an attack but it could also be a by-product, or an intermediate step of a more complex attack. This combination is used for a safety analysis that is extended to also include security events as causes for the TE, the system failure.

This will be the type of combination that will be used in the further course of this thesis. In general, it could also be possible to use the type from Fig. 3.4b. But this only would be necessary if an aspect of a security event would have to be analyzed in highest detail.



(a) An AT extended by a CFT.

(b) A CFT extended by an AT.

Fig. 3.3. Combinations of CFTs and ATs. Fig. 3.3b shows the combination that is used for this thesis.

3.2.3 Nested Trees

To make the list complete, Fig. 3.4a and 3.4b depict combinations of the previous two (Figs. 3.3a and 3.3b). As mentioned before, combination 3.4b could be interesting if and only if an attack that is causing a system failure has to be examined down to the last detail. Combination 3.4a is not pursued further here based on the same grounds as combination 3.3a. It focuses on security, in contrast to this thesis which has its focus on safety.

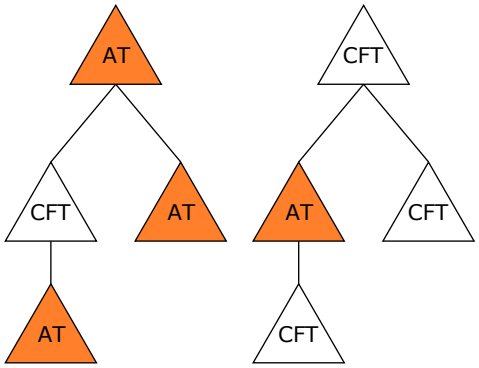
3.3 Tree Creation Approaches

After introducing the different combination variants in Sect. 3.2, this section is about the development of such a combined tree. In general there are two variants to generate a combined tree: The first variant is to model a CFT as well as a separate AT and to find points of intersection to combine the trees.

The second one is to generate a CFT for safety analysis and to extend it by adding security events as additional causes. In the following, both variants will be discussed.

3.3.1 Combination of two Complete Separate Trees

The first variant that comes to mind if a combined tree should be generated, is to create two separate trees and combine them afterwards. Figure 3.5 shows



(a) An AT extended by a CFT which is extended by another AT.

(b) A CFT extended by an AT which is extended by another CFT.

Fig. 3.4. Nested combinations of CFTs and ATs

the combination process for this approach on the basis of a CFT and an AT. In the first step safety experts build a CFT, and security experts build an AT using a set of predefined common rules for syntax and semantic. Such rules are necessary for the next step where the trees are searched for intersections, namely common events. These common events could be any combination of events. In Fig. 3.5 the example shows the combination at two Intermediate Events (IEs) i_1 and i_2. At the position of these events a new `OR`-gate (i_3) is inserted between the trees, and the events i_1 and i_2 and their respective subtrees are connected to it. That results in a combined tree shown in the lower part of the figure. In this example an IE is the interface between CFT and AT. This results in a graph with two TEs. Depending on the viewpoint of the analysis (safety or security) the appropriate TE is used.

This approach for the development of a combined tree has the advantage that if an analysis of either safety or security already exists, it could be reused under certain circumstances (provided the rules for constructing the trees are the same). Also, conducting two separate analyses in detail might lead to fewer omitted causes.

But it also has its disadvantages: The construction of the trees depends on two different views. In a CFT for a safety analysis failures are modeled as TEs whereas in an AT for security analysis attack goals are modeled. Attack goals can be different from failures because attackers may not want to crash the system but to gain access to it. Also, if the system is modeled modularly in a CFT, the AT is modeled for the whole system, similar to a traditional FT. This can lead to matching problems, as different parts of the AT can match to different components. Common events can also be events that have the same

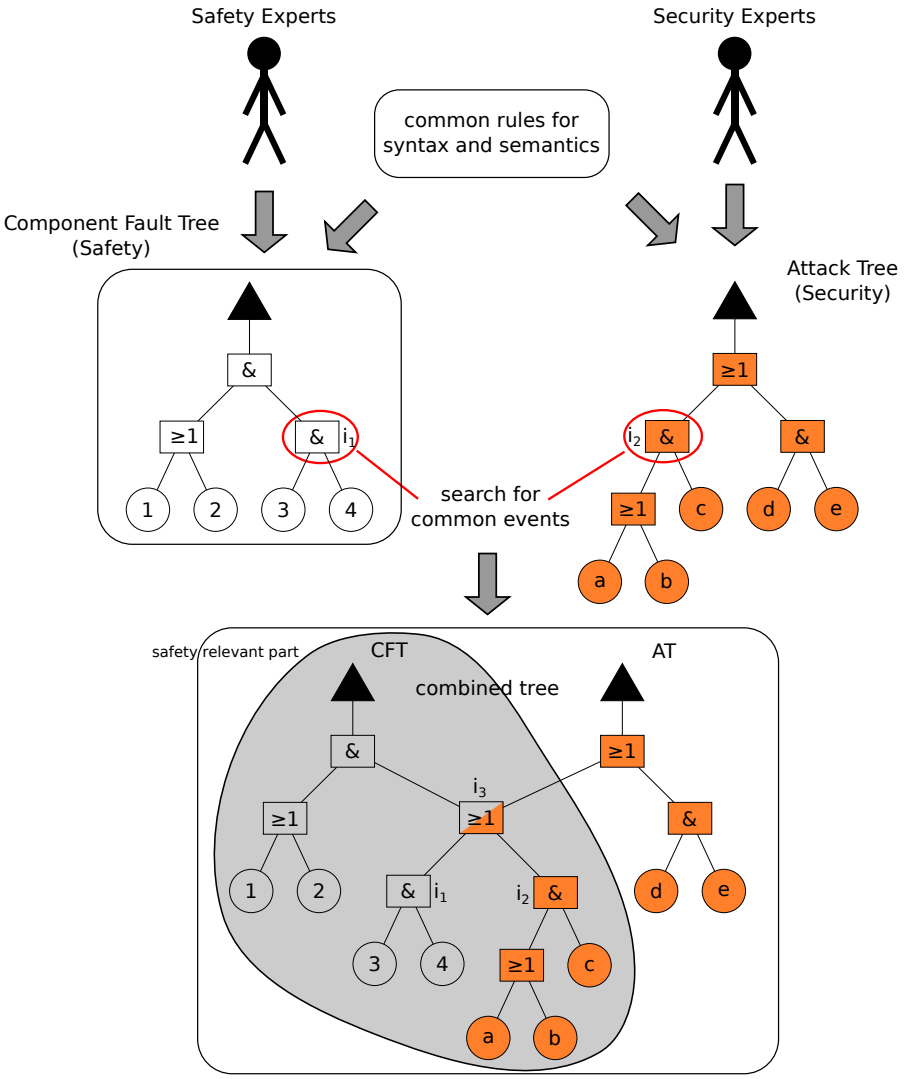


Fig. 3.5. The combination approach of a CFT and an separate AT

effect on the system but are named different. This is why naming conventions are especially important for this approach. These problems prevent a simple automatic merge of the trees because even same names not necessarily mean same semantics. So it can be difficult to find common events in the trees without manually going through the whole tree with both safety and security experts present. If only one analysis perspective (safety or security) is needed, this approach creates an unnecessary overhead. On the other hand, if the

system requires both perspectives, this approach may be beneficial to the completeness of the analyses.

3.3.2 Extension of a Tree

The second approach is the extension of an existing tree. An example is shown in Fig. 3.6 (FT version) and 3.7 (CFT version). Most of the time an analysis is either concerning security *or* safety. So it is not required nor desired to analyze both aspects completely because of the additional effort. The shown example is about the extension of an FT respective CFT for a safety analysis that includes security events as additional causes. An FT/CFT is extended by branches of an AT. In the first step of this approach safety experts build an FT/CFT (left side of Figs. 3.6 and 3.7). This FT/CFT is the basis of the next step, when security experts search through the tree and identify events that could additionally be caused by security events (circled `AND`-gate/IE i_1). To do that, additional information about the system is needed. Information about the data flow in the system and between the system and its environment is useful for security analysis because communication interfaces are preferred targets for an attack. This is why the specifications for the communication interfaces are also helpful at this stage.

In a traditional FT one would do the following: In place of the identified events, an `OR`-gate (i_2) is inserted into the tree and the former event i_1 and the newly found security event a are attached to it (right side of Fig. 3.6).

In a CFT however, the security event a (the vulnerability) should be modeled as an input port and the actual attack in a separate component (see Fig. 3.7). This separate component models the attacker that includes all relevant security events. These security events are accessible for other components via the output ports of the attacker component. So instead of a security event, an input port (sec_in_1) is attached to the `OR`-gate (i_2) (see Fig. 3.7).

In a first iteration of the combined tree the tree is finished now (right side of Fig. 3.6 or 3.7). If more detailed information about the new security event is needed, the attacker component can be refined just like any other component. For each TE, respective output port, of the attacker component an AT can be developed that describes the attack in more detail. Also, additional input ports are imaginable if the attack requires failures from other components (compare Fig. 3.4b).

The advantage of this approach is that the analysis team concentrates on the aspect that is important for them (in this case safety), and tries to find all possible causes for the TE. Only one aspect (either security or safety) is analyzed completely, so there is less analysis effort than for two complete analyses as in Sect. 3.3.1. Usually a system is either security critical, like databases, or safety critical, like embedded systems that control machines. So in practice only one aspect is interesting as analysis objective for a specific system.

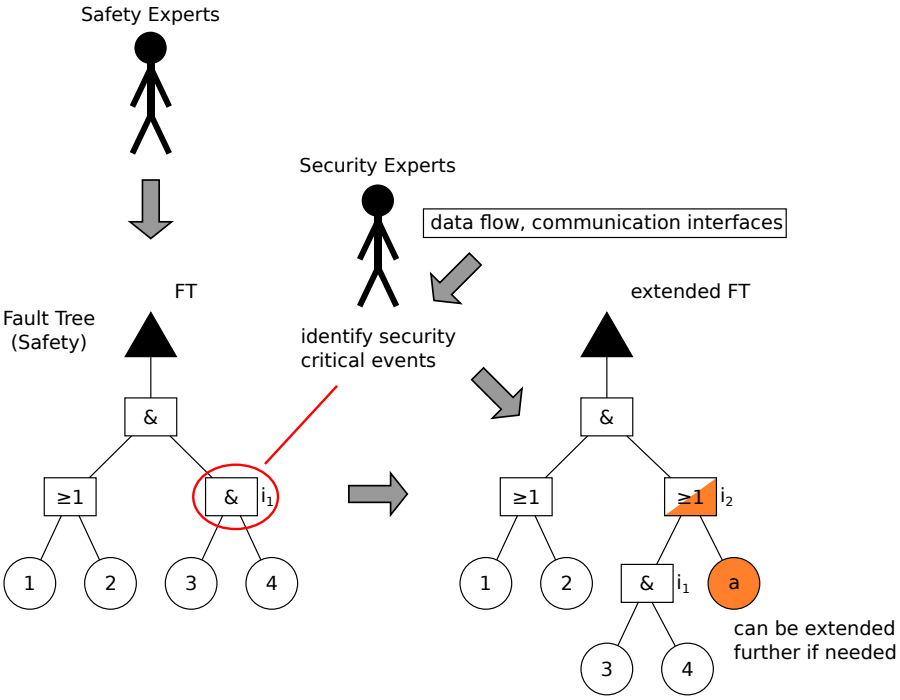


Fig. 3.6. The extension of a Fault Tree

3.4 Development of a Security-enhanced Component Fault Tree (SeCFT)

As stated earlier, the objective of this thesis is to improve safety analysis. Therefore, the second combination variant (Sect. 3.3.2) is the technique of choice. Figure 3.8 shows the development process of an SeCFT. First, a CFT is created by safety experts for the system or component failures under consideration. During the development of a CFT, some rules have to be obeyed. They are recalled in Sect. 3.5. The failure modes (TEs) of the CFT can be found using FMEA or HAZOP as supporting techniques (cf. Sect. 2.1.2) and a description of the system structure including data flows. When that CFT is finished so far, it is searched for events that may additionally be caused by security events (e.g. intentional manipulation of the system). This step can also be intertwined with the creation of the initial CFT. In the example IE i_1 is identified as an event that may be caused by an attacker (circled `AND`-gate).

Often, those events appear at interfaces to the system environment or between components. So, these interfaces should receive special consideration. Possible points to attack a system are (this list is not necessarily complete):

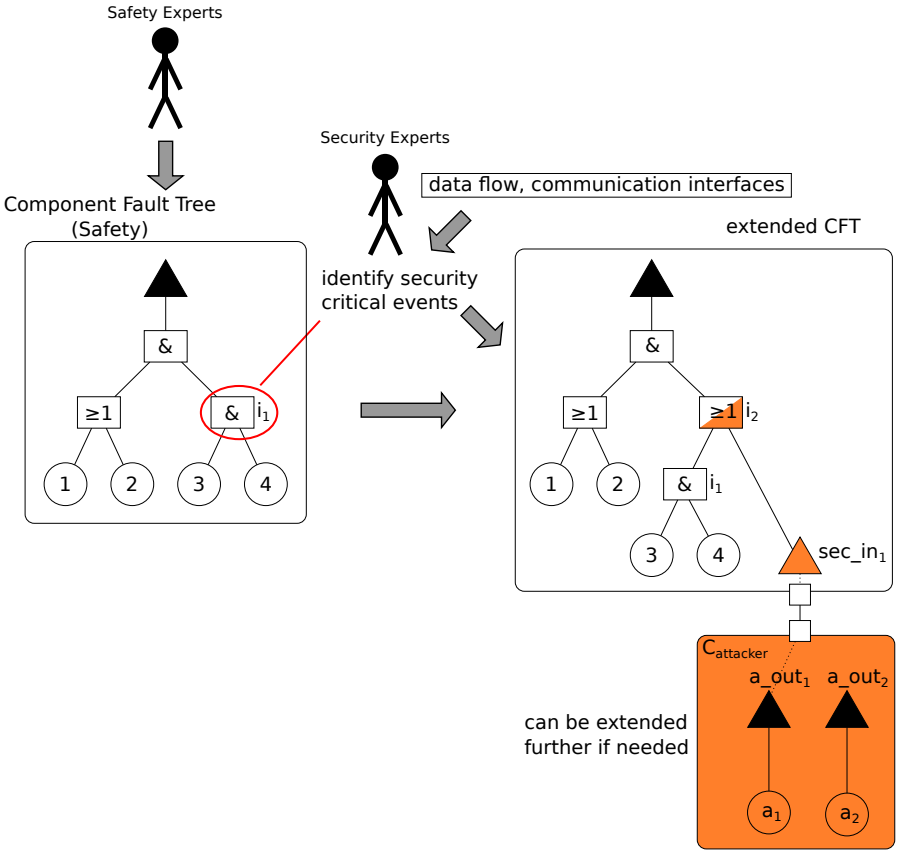


Fig. 3.7. The extension of a Component Fault Tree

- interfaces between system components and the environment (e.g. malformed data can be injected at sensor or actuator interfaces),
- communication channels that run through the system environment (meaning the channel is potentially accessible from the environment),
- and the software/hardware of the system components themselves (changing the behavior of the system itself).

From the attacker's point of view, the first two are usually the most promising points because they are accessible relatively easy from the system's environment. Spoofing or denial of service attacks can be aimed at sensors to bring the system in a vulnerable or dangerous state. An example could be a robot that uses infrared range finders to avoid obstacles. If the range finder is blinded by an infrared laser pointer, the robot might either ignore an existing obstacle or imagine a non-existing one. In both situations the robot might damage itself or its environment.

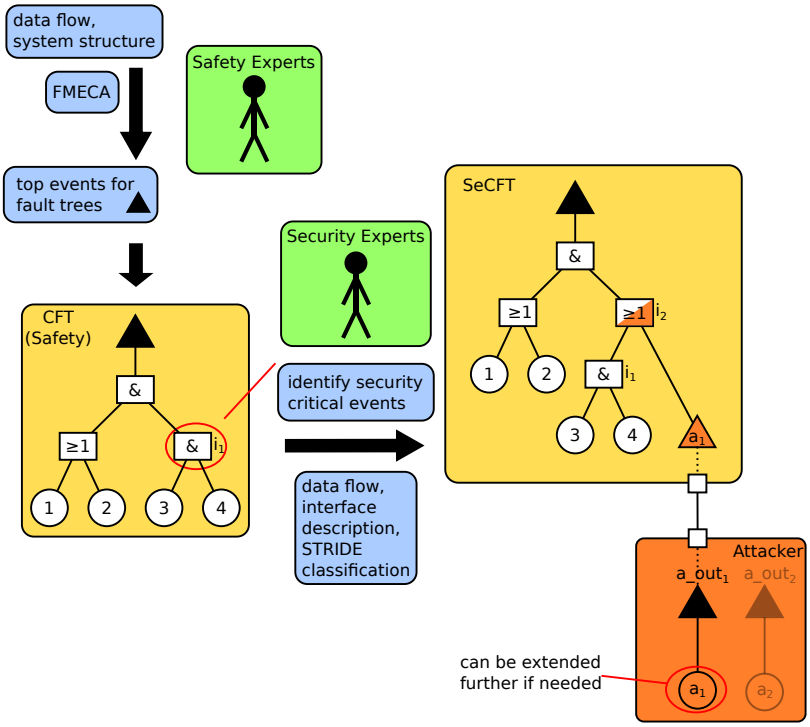


Fig. 3.8. The development of a Security-enhanced Component Fault Tree (SeCFT)

Communication channels that run through the environment (wired and radio signals alike) are susceptible to denial of service (jamming) or spoofing attacks (sending malicious signals). Denial of service attacks have the same effects as random faults of the communication. That means, this type should have already been handled by a traditional safety analysis. But it also could be the case that the probability of a random fault of the communication was determined as so low that no countermeasures were implemented. If attacks however would have been taken into account, the countermeasures would be necessary to avoid a deliberately caused failure.

Attacks on software/hardware level of a component are the ones which can have the most effects because the system behavior can change in an unforeseen manner. However, such attacks on the software are more difficult because they require unrestricted access to the software which usually requires either a previous attack on the communication or another way of access. Hardware attacks on the other hand are a special case, they require physical access to the system which can raise the necessary effort to high levels. Hardware attacks are also the ones that have almost unrestricted effects on the system, and they can hardly be detected unless they are foreseen during system development.

Having identified potential points of attacks by analyzing information such as data flow or interface descriptions together with the CFT, the next step is to find out which attack types are possible at those locations. Known attacks are collected in online databases such as the Open Web Application Security Project [OWASP 15] or vulnerability databases provided by several Computer Emergency Response Teams (CERT). The STRIDE classification is a tool to identify attacks (cf. Sect. 2.2.2). Table 3.1 shows the categories of the STRIDE classification.

Table 3.1. STRIDE maps threats to security properties

Threats	Security properties
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

With potential points of attack in mind, an analyst searches the CFT from TE to BEs for events that can be triggered by an attack. For each event in question the STRIDE categories are checked to determine which attack types are possible.

No events should be left out from the beginning, unless the causes for that are documented in the tree or in a separate document that is linked to the tree. If an event is found that can also be caused by an attack, the tree is extended by an OR-gate to which the previous subtree and the new security cause are attached (see Fig. 3.8). The security causes are modeled as input ports which are connected to an attacker component (see Fig. 3.9) in which the actual security events are located.

In terms of CFTs the attacker would then be a new component which is interacting with the system. From a semantic view point this makes sense because the attacks are not part of the system components as they could be seen if they would be modeled as BEs in the components. The input port models the vulnerability, and the attacker component models the actual attack. Modeling the attacker as a separate component therefore fits to the component concept of CFTs [Kaiser 03b] and to the security model where an attack exploits a vulnerability. Also, this eases necessary changes if new attacks are known because the only changes in the CFT of the system are new input ports and additional attacks in the attacker component.

The attacker component as shown in Fig. 3.9 can simply be a CFT with as many output ports as there are different attacks. The attacks themselves can be modeled simply as BEs (a_1) or be refined as an AT (a_2).

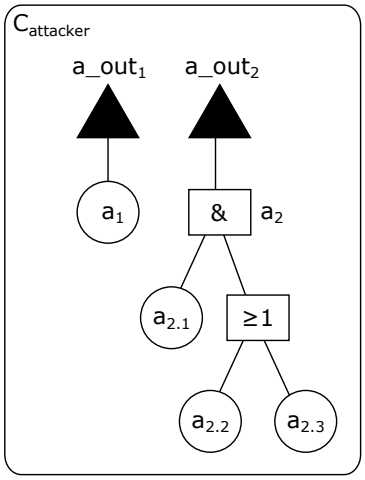


Fig. 3.9. An attacker component with two possible attacks

It is also imaginable that there is more than one attacker component modeled. If so, they can interact just like CFT components. This analysis process was also described in a paper for a workshop at the SafeComp conference 2013 [Steiner 13a].

The modeling process can be summarized as follows:

1. create a CFT
2. search for points of attacks/vulnerabilities
3. identify the types of vulnerabilities
4. develop ATs to model the attacks

3.5 Rules for the Development of an SeCFT

For the development of an SeCFT the same rules apply as defined in [Kaiser 06] for CFTs. Precondition for a quantitative analysis is that all input ports of a component are connected. Otherwise necessary information is missing if a component depends on external inputs. Also, BEs should be independent from each other because of the same reasons they should be in FTs (cf. [Vesely 81]). Kaiser defined some rules for the creation of CFTs:

- only one edge can lead to a target, more than one edge is forbidden
- but one or more edges can start from the same point
- if a target should have more than one input, a gate has to be used to join the edges
- directed cycles are not allowed
- a component cannot contain itself as a subcomponent

Also, he stated that small and self-contained subcomponents are preferred for later reuse. Therefore only BEs that really belong to the current component should be added there. For BEs that are in fact external causes, an input port should be added and the actual event should be placed in another component (thus the external attacker component mentioned earlier).

4

The Analysis of Security-enhanced Component Fault Trees

In Chap. 3 the modeling approach for an analysis with Security-enhanced Component Fault Trees (SeCFTs) is described. This chapter deals with the analysis of such SeCFTs. Qualitative and quantitative analysis methods will be described. The analysis process is also handled in [Steiner 15].

To be able to conduct a quantitative analysis, a comprehensive rating of all the events in an SeCFT has to be present. For that purpose different rating schemes for the security events of the SeCFTs are introduced, and it is discussed how they can be integrated with the safety rating schemes stemming from a Component Fault Tree (CFT). Using a comprehensive rating for all events the individual impact of an attack on the occurrence of the Top Event (TE) can be determined.

This chapter is built-up as follows: First, different rating schemes that can be included in an SeCFT are discussed. There are probabilities for safety events, likelihood values for security events, and a combination of both to be able to rate Minimal Cut Sets (MCSs) according to safety and security.

Then, calculations rules for the different rating schemes are described. Calculation rules used in CFTs for probabilities are recalled, new rules for likelihood values are discussed, and they are applied to the tuple rating that is composed of probabilities and likelihood.

After that the application of the established rules for qualitative and quantitative analysis is dealt with.

In the end the whole analysis process is summarized, and some ideas how to decide on countermeasures are devised.

4.1 Ratings of Basic Events in SeCFTs

In a CFT several ratings can be assigned to elements. Usually, ratings are assigned to Basic Events (BEs). Ratings of MCSs are calculated from the ratings of their corresponding BEs. The rating of a TE is calculated by the

combination of all BEs or less accurate, but more conservative, from the sum of the ratings of all MCSs.

In CFTs BEs are typically rated using probabilities or reliability values (see also Sect. 2.1.1). These are used to calculate the respective values for MCSs and the TE.

The modular composition of CFTs (one CFT per system component) allows for the calculation of ratings for subcomponents of a larger system and for later reuse of these values. This can be done if a CFT is not depending on inputs from other CFTs. Then, MCSs as well as other ratings can be calculated for the output ports respective the TEs of the CFT. Especially for basic components that are used in several systems this makes sense, and it can reduce the analysis effort [Kaiser 06]. Measures for the contribution of components to the overall system can also be calculated.

In an Attack Tree (AT) the same basic elements exist as in a CFT. Either Boolean or continuous values can be assigned to BEs. As Boolean values value pairs such as *possible – impossible* or *expensive – not expensive* are used. Continuous values for BEs can be *costs to execute an attack*, *probability of success of a given attack*, or *probability that an attack is actually conducted*. The latter probability however is problematic as a rating for BEs in an AT (cf. Sect. 2.2.1).

Figure 4.1 shows an example of an attacker component with two output ports out_1 and out_2. For the output port out_2 it is basically an AT which consists of 4 gates and 5 BEs. Two MCSs are present which represent two different attack scenarios: $\{e_1, e_2, e_5\}$ and $\{e_3, e_4, e_5\}$.

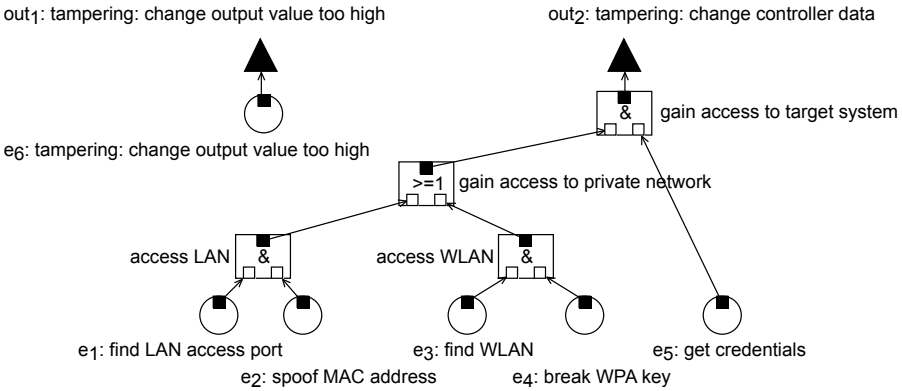


Fig. 4.1. An example attacker component

If an attack is consisting of several actions an attacker has to perform, like the ones for output port out_2 in the example, these actions are not stochastically independent in general. The usual assumption is if an attacker plans to attack a given system, and that attack requires him to execute different

actions (security events, sub-attacks), it is most likely that he will at least try all sub-attacks that are necessary to reach his goal (an attacker acts rational [Buldas 06]). In terms of the given example this means if an attacker chooses to try BE e_1 and he succeeds, he most probably will also try e_2 and e_5. In general this means, in an AT the events contained in an MCS are not independent from each other.

Therefore, it can make more sense to assign a probability to a whole MCS which represents the attack, instead of the BEs. The other rating values (other than probabilities) can be calculated for the MCSs from their respective BEs.

A first result from a safety analysis based on SeCFTs is the set of MCSs, as they are all combinations of BEs that lead to the analyzed system failure – the failure scenarios. To decide which of these combinations have to be mitigated to reach a certain safety level, this set of MCSs has to be prioritized. And of course to decide whether a system is complying to a given safety level from a standard or another requirement, the TE rating of the CFT has to be calculated.

Depending on available data, different rating schemes can be used. They will be described in the next sections.

4.1.1 Probabilities

Probabilities of occurrence, just like they are used in CFTs, are the first thing that comes to mind when thinking about ratings for BEs in SeCFTs. If those would be available for all events in the SeCFT – safety as well as security – in a comparable accuracy, a probability of occurrence for the TE and all MCSs could be easily calculated (see also [Mauw 06, Buldas 06, Fovino 09]). For safety events this comes naturally because often there are tables with failure probabilities for different system components. Of course, for software components they are more difficult to find than for hardware components. There exist some approaches that try to handle this fact by using easier to estimate probability intervals instead of single probability values [Carreras 01, Förster 09]. But probabilities for security events lead to several problems: The probability of occurrence for a security event corresponds directly to the combination of the probability that an attacker conducts an attack and the probability that this attack is successful. A probability that an attack is successful could be determined from expert knowledge and experienced data just like failure probabilities (cf. Sect. 2.2.1). But even the success probability is difficult to determine. There is only a small portion of the data about successful attacks available. Most successful attacks are not published because companies fear the loss of trust of their customers (cf. Sect. 2.2.1).

The bigger problem is the probability that an attacker actually conducts an attack. First, this probability depends on different aspects: the attacker's motivation, experience, availability of assets/money/knowledge, and accessibility of the system. And second of all, if this attack requires several distinct

actions that are modeled as separate security events, these events are not independent, as it would be required for most calculation algorithms for CFTs. Some estimate of the attack frequency of specific systems can be obtained by measuring the frequency of attacks to honey pots that are built to mimic real, vulnerable systems [Wilhoit 13].

There are some approaches that try to deal with the uncertainty of probabilities for some events of a CFT which can also be applied to an SeCFT. One approach is to derive a pseudo-probability from security indicators [Förster 10]. Together with other available probabilities, a probability for the TE can be calculated. This approach has the advantage that the same calculation rules can be applied as if there would be probabilities available for all BEs. But there are also some problems: Security attacks that are part of a larger compound attack directly depend on each other and therefore do not fulfill the independence criterion for probabilities in CFTs. Also, it is important to find the correct order of magnitude for the probabilities of security events, otherwise the resulting probability can deviate in a matter that makes the whole analysis obsolete. And last but not least, the numerical results from such an analysis suggest a level of accuracy that can be misleading.

Another approach is to do a random-sampling instead of using precise values (cf. Sect. 2.1.1). For every BE a probability interval is chosen instead of a precise probability. The stochastic distribution of probabilities in those intervals can be chosen as well (it should be uniform, if nothing is known about the BE). By randomly sampling values from the probability intervals of the BEs the TE probability of the TE is calculated for a large number of times, thus resulting in the probability distribution of the TE. This method is described in detail in [Förster 09].

To summarize, probabilities of the required accuracy for safety events are relatively easy to obtain. But for security events they are either not as accurate as the ones for safety events and therefore not directly comparable, or they are not independent from each other, so the usual calculation algorithms do not work. The next sections show different approaches that avoid problems that arise with probabilities for security events.

4.1.2 Likelihood

Instead of trying to assign probabilities to security events, it is a better idea to use a more coarse scale with only a few discrete values. [IEC 61025 06] states for Fault Tree Analysis (FTA) that in case when probabilities cannot be assigned to BEs, a "descriptive likelihood of occurrence" can be used with values such as: *highly probable*, *very probable*, *medium probability*, *remote probability*, etc. Different standards use different numbers of distinct values, e.g. [ED-202 10] considers five levels: *extremely improbable*, *extremely remote*, *remote*, *probable*, and *frequent* [Casals 12]. This likelihood can be used to rate security events in an SeCFT. Assigning more precise numerical values would only add fictitious precision which can be misleading in the interpretation of

the results as mentioned earlier in Sect. 4.1.1. This also has to be kept in mind when calculating values for combinations of events that are rated with likelihood values.

The values of that likelihood are determined from several indicators such as: *attack cost*, *attack resources*, *attacker motivation*, *accessibility*, or *attacker type*. For a first step a three-step scale is proposed which can be easily extended if required. The scale represents the likelihood of a security event to occur. To each value a numerical value is assigned for easier comparisons. From this follows that the likelihood would be mapped to integer values from the interval $[1, m]$ with $m \in \mathbb{N}$ the number of discrete values. In case of the IEC 61025 scale this means *highly probable* := 4 and *remote probability* := 1. A high rating corresponds with a high likelihood of occurrence and a low rating with a low one. The number of different values depends on the granularity of the used security indicators. If almost nothing is known about an attack, a three-step scale is sufficient. But if more detailed indicators are available, it makes sense to expand the scale. For example if six levels of accessibility can be distinguished for a system, the likelihood derived from that should have also 6 levels (e.g.: *remote via Internet*, *remote via local wired network*, *remote via local wireless network*, *easy physical access* (no access restrictions), *hard physical access* (security doors), *no access*). The methodology is independent of the number of likelihood steps.

The used likelihood scale as well as the origin of the values has to be defined before the analysis. A possible method is to calculate the likelihood in a similar way as the Risk Priority Number (RPN) of a Failure Mode, Effects, and Criticality Analysis (FMECA) (cf. Sect. 2.1.2).

The attack scenarios are rated by security experts according to known indicators. A predefined rating scheme is required for each indicator for a consistent rating. The relative importance of each indicator should be determined in advance. As it is difficult to achieve consensus among experts, it is generally recommended to restrict rating schemes to rather coarse scales. As mentioned before, it should only be as detailed as there is distinct data available. The value v_i of each indicator i can be determined on a scale from 1 to n, where 1 indicates a small likelihood that the attack is carried out according to indicator i, and n indicates a high likelihood. For example, potential damage or risk of attack is typically rated as *low*, *medium*, or *high*, thus $n = 3$. The relative importance of each indicator results in an influence factor $f_i \in \mathbb{R}^+$ which has to be determined for each indicator. The influence factor depends also on the type of attacker. Depending on his resources, capabilities, motives, and his preparedness to take risks, the factor is increased or decreased. For each attacker type taken into account a set of influence factors has to be created. An overall likelihood for an attack could then be calculated as $L = \sum_i v_i \times f_i$ where i is the index of the indicator [Förster 10]. An example for this approach is the rating used in Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) (cf. Sect. 2.3.3)

A possibility to achieve a common rating between safety and security events, other than probabilities, is to use the likelihood for both safety and security events. The advantage of this approach is that values for all BEs can be determined relatively easy, and comparisons of likelihood are easily performed. The disadvantage is that the accuracy coming from rating safety events with probabilities is lost.

To use the advantages of both, probabilities for safety events and likelihood for security events, the next section will describe an approach using a combination of both probability and likelihood.

4.1.3 Tuple Rating

In Sect. 4.1.1 it was established that rating BEs with probabilities is the preferred way for safety events. Section 4.1.2 concluded that for security events a rating with likelihoods is more feasible.

Hence in an SeCFT there can be both likelihoods and probabilities for different events. When MCSs are determined in a tree that includes safety as well as security events, there can be three types of MCSs as defined in the following:

Definition 24 (Safety MCS) *A safety Minimal Cut Set contains only safety events (cf. Definition 22 on p. 47).*

Definition 25 (Security MCS) *A security Minimal Cut Set contains only security events (cf. Definition 23 on p. 47).*

Definition 26 (Mixed MCS) *A mixed Minimal Cut Set contains safety events as well as security events.*

The TE will most certainly depend on safety as well as security events. Therefore a combination of both probabilities and likelihood is needed to calculate ratings for MCSs and TEs. Figure 4.2 shows an example SeCFT that uses the attacker component from Fig. 4.1. For the TE `controller value too high` there are MCSs of all three types in this example (see Table 4.1).

Table 4.1. MCSs according to type, whereas `controller` and `attacker` refer to the appropriate components

MCS type	Basic Events
safety MCS:	{`controller`.e_1}
security MCSs:	{`attacker`.e_1, `attacker`.e_2, `attacker`.e_5},
	{`attacker`.e_3, `attacker`.e_4, `attacker`.e_5}
mixed MCS:	{`attacker`.e_6, `controller`.e_2}

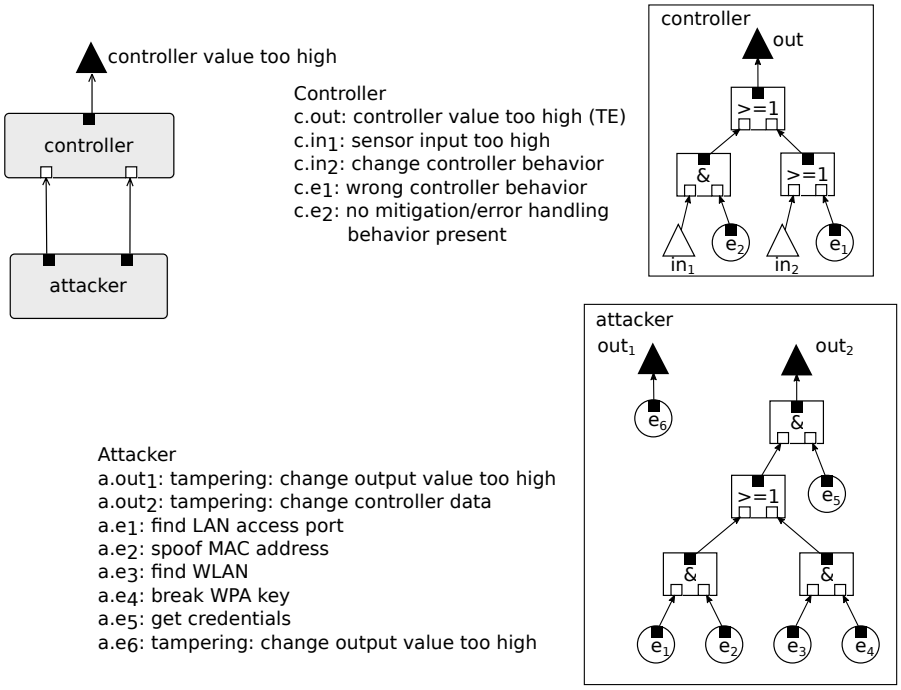


Fig. 4.2. An example SeCFT consisting of a controller component and an attacker component

Events in a safety MCS are rated with probabilities. Therefore the overall rating of a safety MCS is also a probability. Events in a security MCS are rated with likelihoods. So the overall rating of a security MCS is also a likelihood. In a mixed MCS however, there are both probabilities and likelihoods. As they are not directly comparable, the rating of a mixed MCS is a tuple consisting of the overall probability of all included safety events and the overall likelihood of all included security events. For TEs in an SeCFT, the same holds as for mixed MCSs.

The next section will introduce the extensions for the calculation rules needed for an SeCFT to handle the tuples of probabilities and likelihoods.

4.2 Calculation Rules for SeCFTs

In this section the rules how to perform the calculations needed to rate MCSs and TEs are described. The partitioning of this section is the same as the section before. First, the rules for calculating probabilities are recollected. Then, new rules to calculate likelihood results of `AND` and `OR`-gates are discussed, and at last the rules are adapted to the tuples needed in SeCFTs.

4.2.1 Probabilities

For the calculation of the probabilities at least calculation rules for the gates `AND`, `OR`, and `NOT` are required. Other gates such as `XOR` or voter gates can be constructed from these three basic gates. Their calculation rules result from them accordingly.

For recollection, the calculation rules for the gates `AND`, and `OR` will follow. The outcome of an `AND`-gate with two independent input events A, B, or more general n independent input events X_i, is calculated as follows:

$$P(A \wedge B) = P(A) \times P(B) \tag{4.1}$$

$$P\left(\bigwedge_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} P(X_i) \quad , i, n \in \mathbb{N} \tag{4.2}$$

The outcome of an `OR`-gate with two independent input events A, B, or more general n independent input events X_i, is calculated as follows:

$$P(A \vee B) = P(A) + P(B) - P(A) \times P(B) \tag{4.3}$$

$$P\left(\bigvee_{i=1}^{n} X_i\right) = 1 - \prod_{i=1}^{n} (1 - P(X_i)) \quad , i, n \in \mathbb{N} \tag{4.4}$$

As already discussed in Sect. 2.1.1 the `NOT`-gate will be avoided in SeCFT models. The calculation of probabilities for the basic gates is discussed in detail in Sect. 2.1.1. In Sect. 4.1 it was established that it is not feasible to use probabilities for security events. On the other hand, if probabilities exist for safety events, it is recommended to use them in an analysis.

Using these probabilities a partial quantitative analysis can be performed. Resulting probabilities can be calculated for MCSs and TEs if events without probability values are ignored. This means that only safety events are taken into account for the calculation of a resulting probability. Thereby an upper bound for the probability of MCSs is calculated.

4.2.2 Likelihood

Likelihood was defined as a qualitative probability for the occurrence of a security event. Security events are in most cases attacks conducted by an attacker. The following definitions for combinations are used:

Definition 27 (Likelihood `AND`-gate) *All subordinate events have to occur in order that the combined event occurs. Therefore, the event with the lowest likelihood determines the likelihood L of the combined event.*

$$L\left(\bigwedge_{i=1}^{n} X_i\right) = \min_{i=1}^{n}[L(X_i)] \quad , i, n \in \mathbb{N} \tag{4.5}$$

Another calculation can be derived from the calculation for the probability of an AND-gate (cf. Sect. 4.2.1). Values $P(X_i)$ are from the interval $[0, 1]$. Equation 4.2 holds for those values. The values $L(X_i)$ are from the integer interval $[1, m]$, where $m \in \mathbb{N}$ is the maximum likelihood value defined. Values $L(X_i)$ can be transformed from interval $[1, m]$ to interval $\left[\frac{1}{m}, \frac{m}{m}\right]$ by dividing by m. For $m \to \infty$ follows $\frac{1}{m} \to 0$. In that new interval $\left[\frac{1}{m}, 1\right]$, the same functions that are used for the probability calculations can be used. For n independent input events X_i, the likelihood of an AND-gate is calculated as follows:

$$L'\left(\bigwedge_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} \frac{L(X_i)}{m} \quad , i, n, m \in \mathbb{N} \tag{4.6}$$

The result of the AND-gate can only have the precision of the inputs. So, it has to be transformed back to the integer interval $[1, m]$. This is done by multiplying the result L' with m and rounding up. Rounding up ensures that no underestimation occurs.

$$L(x) = \lceil m \times L'(x) \rceil \quad , m \in \mathbb{N} \tag{4.7}$$

In comparison to this calculation, the minimum function from Definition 27 is a worst-case estimation.

The OR-gate can be defined along the same lines. The first idea assumed that if an attacker can choose which attack to perform, he would choose the most profitable or easiest one, and ignore the other ones [Steiner 15]. Then the outcome of an OR-gate can be calculated using the maximum likelihood value of all inputs. But then one might underestimate the likelihood of an attack because it does not take into account that several attackers might attack at the same time.

Definition 28 (Likelihood OR-gate) *At least one of the subordinate events has to occur in order that the combined event occurs. An attacker will most likely choose the event with the highest likelihood. But more than one attacker may attack at the same time. This equation only takes into account single attackers that always only choose one attack path:*

$$L\left(\bigvee_{i=1}^{n} X_i\right) = \max_{i=1}^{n}[L(X_i)] \quad , i, n \in \mathbb{N} \tag{4.8}$$

If more than one attacker or attackers that try several paths at once should be taken into account, the following calculation should be used. It is again derived from the probability calculation, and it depends on the transformation of the value interval described for the AND-gate.

$$L'\left(\bigvee_{i=1}^{n} X_i\right) = 1 - \prod_{i=1}^{n}\left(1 - \frac{L(X_i)}{m}\right) \quad , i, n, m \in \mathbb{N} \tag{4.9}$$

And again the result $L'(x)$ is transformed back to the value interval $[1, m]$ by equation 4.7. This might result in values greater than m which is the maximum value. This together with equation 4.9 results in

$$L(x) = \min\left(\lceil m \times L'(x) \rceil, m\right) \quad , m \in \mathbb{N} \tag{4.10}$$

for the likelihood outcome of an `OR`*-gate.*

The likelihood is propagated through the SeCFT like the probabilities: All events without a likelihood are ignored. It is expected that every safety event has a probability, and every security event has a likelihood assigned to them. For the calculation of a resulting likelihood only the security events are taken into account.

4.2.3 Tuple Rating

Following the calculation rules for probabilities (Sect. 4.2.1) and likelihood (Sect. 4.2.2), rules how to handle the combination of both in terms of the tuples described in Sect. 4.1.3 are defined. To do that all ratings of BEs are interpreted as tuples (P, L), where P is a probability and L a likelihood. For safety events e_{saf} there is no likelihood leading to $(P_{e_{saf}}, -)$ with an undefined $L_{e_{saf}}$, and for security events e_{sec} there is no probability value leading to $(-, L_{e_{sec}})$ with an undefined $P_{e_{sec}}$. Undefined values will be ignored in the calculation of the rating.

This has to be explained further: The alternative to undefined values, would be values that do not influence the order between the events. To achieve this, an *identity element* or *neutral element* for all possible gate-operations would be needed. This would mean in terms of probabilities, a value is needed which is the identity element for addition and multiplication. Such a value does not exist because the identity element for the addition is 0, and the identity element for the multiplication is 1. The same problem arises for the likelihood operations. These values exclude each other, so no value is selected, and the undefined values are ignored during the calculation.

The tuple elements of a combination of events by logic gates are calculated independent of each other according to the rules established in Sects. 4.2.1 and 4.2.2. Figure 4.3 shows the SeCFT from earlier with ratings for all BEs, safety as well as security. Likelihood in this example is a three-level scale of {low, medium, high} with corresponding values of {1,2,3}. Now, it will be shown exemplarily how to calculate the tuple ratings of the MCSs and the TE.

The ratings of the four MCSs result in the following:

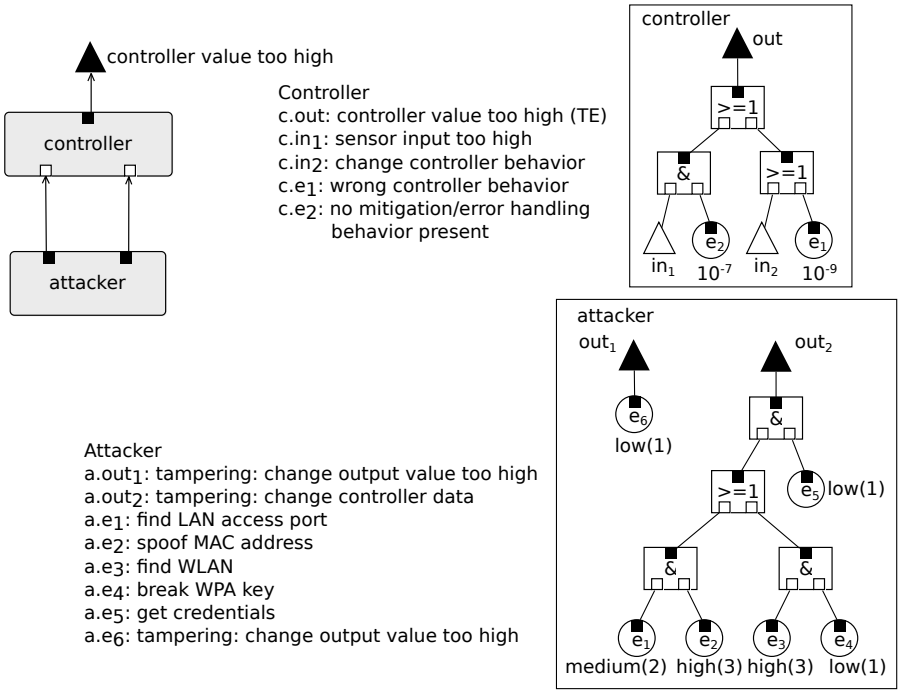


Fig. 4.3. The example from Fig. 4.2 with ratings for safety and security BEs

$$
\begin{aligned}
&MCS_1 : \{\mathtt{c.e_1}\} \\
&\left.\begin{aligned} P(MCS_1) &= P(\mathtt{c.e_1}) = 10^{-9} \\ L_1(MCS_1) &= undefined \end{aligned}\right\} (10^{-9}, -) \\
&MCS_2 : \{\mathtt{a.e_1}, \mathtt{a.e_2}, \mathtt{a.e_5}\} \\
&\left.\begin{aligned} P(MCS_2) &= undefined \\ L(MCS_2) &= \min(L(\mathtt{a.e_1}), L(\mathtt{a.e_2}), L(\mathtt{a.e_5})) \\ &= \min(2,3,1) = 1 \end{aligned}\right\} (-, 1) \\
&MCS_3 : \{\mathtt{a.e_3}, \mathtt{a.e_4}, \mathtt{a.e_5}\} \\
&\left.\begin{aligned} P(MCS_3) &= undefined \\ L(MCS_3) &= \min(L(\mathtt{a.e_3}), L(\mathtt{a.e_4}), L(\mathtt{a.e_5})) \\ &= \min(3,1,1) = 1 \end{aligned}\right\} (-, 1) \\
&MCS_4 : \{\mathtt{a.e_6}, \mathtt{c.e_2}\} \\
&\left.\begin{aligned} P(MCS_4) &= P(\mathtt{c.e_2}) = 10^{-7} \\ L(MCS_4) &= L(\mathtt{a.e_6}) = 1 \end{aligned}\right\} (10^{-7}, 1)
\end{aligned}
$$

The rating of the TE can be calculated from a disjunction of all MCSs:

$$\left.\begin{aligned} P(TE) &= 1-(1-P(MCS_1))(1-(MCS_4)) \\ &= 10^{-9}+10^{-7}-10^{-16} = 1.009999999\times 10^{-7} \\ L'(TE) &= 1-(1-\tfrac{L(MCS_2)}{3})(1-\tfrac{L(MCS_3)}{3})(1-\tfrac{L(MCS_4)}{3}) \\ &= 1-(1-\tfrac{1}{3})(1-\tfrac{1}{3})(1-\tfrac{1}{3}) = 0.7 \\ L(TE) &= \min\left(\lceil 3\times 0.7\rceil, 3\right) = 3 \end{aligned}\right\} (1.01\times 10^{-7}, 3)$$

The calculation of the TE using the maximum function would result in an overall likelihood of 1 in comparison to 3 from the sum calculation. For smaller scales such as the one used in the example the sum calculation of the OR-gate tends to overestimate the likelihood. So it can be useful to additionally calculate the maximum of the involved likelihood values to get a notion of the possible range of likelihoods. The undefined values $P(MCS_2), P(MCS_3)$ and $L(MCS_1)$ are ignored in this calculation. How these tuple ratings are used in a quantitative analysis of an SeCFT will be described in Sect. 4.4.

4.3 Qualitative Analysis of SeCFTs

The most important activity of a qualitative analysis in CFTs and SeCFTs is the determination of MCSs. They describe minimal failure scenarios of the system. MCSs also are used to derive a coarse classification of the criticality of failure scenarios and BEs, and they allow to make statements about the general endangerment of the analyzed system (see also Sect. 2.1.1). MCSs are also an important starting point for a following quantitative analysis. Based on the MCSs a basic importance analysis of BEs and MCSs can be conducted.

In Sect. 2.1.1 the qualitative analysis of traditional Fault Trees (FTs) (or CFTs) is described. This section deals with necessary extensions of the qualitative analysis to cope with additional security events in the SeCFT. The first step is again the determination of the MCSs. The interpretation of an MCS is the same as in CFTs: A minimal safety failure scenario (but possibly depending also on security attacks). As mentioned earlier in Sect. 2.1.1 a CFT (and therefore an SeCFT as well) can be transformed into a set of MCSs that represents all failure scenarios which are relevant for the system. In general, a tree contains multiple MCSs corresponding to different failure scenarios.

4.3.1 Minimal Cut Set Analysis

An analysis of MCSs of an SeCFT differs from the analysis of a traditional CFT in a few points. The first step is to calculate the MCSs per TE. The second step is to sort the MCSs according to their size. The result of a qualitative analysis is an ordered list of MCSs.

As discussed in detail in Sects. 4.1 and 4.2, ratings of safety and security events cannot be compared directly. Therefore it makes sense to sort the MCSs according to safety events and security events. Then, one receives three lists of MCSs:

1. safety MCSs (cf. Definition 24)
2. security MCSs (cf. Definition 25)
3. mixed MCSs (cf. Definition 26)

Safety MCSs are analyzed as usual: A qualitative analysis starts with an ordering according to the size of the MCS (Definition 11 in Sect. 2.1.1). The smaller an MCS is, the more critically it should be analyzed. This is explained by the fact that all events in an MCS have to occur, so that the TE occurs and the system fails. The lesser events have to occur, the more the TE depends on individual events. So events in smaller MCSs deserve more attention in an analysis. An especially critical case is an MCS with only one event – a single point of failure which itself can directly lead to the system failure.

Security MCSs are a more special case: In this case a system failure only depends on external influences and does not depend on failures of internal system components. But here also holds that smaller MCSs are more critical than larger ones. Pure security MCSs are not more critical per se than pure safety MCSs, but the uncertainty of the modeling of an attack is relatively high. Depending on the threat scenario and the attacker type the likelihood value changes over time. Necessary tools become better available and cheaper over time which can make an attack more probable in the future. Also, the attacker type, the attacker's motivation and capabilities can and will change over time – potentially to the disadvantage of the system. This is why pure security MCSs should be avoided by adding suited countermeasures which convert security MCSs to mixed MCSs.

Mixed MCSs on the other hand can be managed better than security MCSs: For the occurrence of the TE all events of a mixed MCS have to occur which means regular safety events have to occur. These occurrences of safety events can be managed with the usual methods such as redundancy or monitoring. The probability for a mixed MCS to cause the TE has an upper bound: the probability of the contained safety events. This way the criticality of security events can be mitigated by safety events with low probability. The probability of statistically independent safety events is multiplied to obtain the probability of the TE (cf. Sect. 4.2.1). That means, the more statistically independent safety events an MCS contains, the lesser probable it is to cause the TE. With this in mind, adding more events to an MCS increases the safety of the system without knowing the exact probabilities.

Figure 4.4 shows the example SeCFT with highlighted MCSs. MCS_1 is a safety MCS, MCS_2 and MCS_3 are security MCSs, and MCS_4 is a mixed MCS (see Table 4.2).

To summarize the qualitative analysis of MCSs: There are three types of MCSs which differ in the level of controllability of their BEs. Controllability in this context means how much a failure scenario (MCS) depends on faults of the system as opposed to external factors as for example attacks. In descending order according to their controllability these are: safety MCSs, mixed MCSs and security MCSs. Resulting from that, additionally to MCSs containing only

Table 4.2. MCSs according to type, whereas `controller` and `attacker` refer to the appropriate components

MCS number	Basic Events	MCS type
MCS_1	{`controller`.e_1}	safety MCS
MCS_4	{`attacker`.e_6, `controller`.e_2}	mixed MCS
MCS_2	{`attacker`.e_1, `attacker`.e_2, `attacker`.e_5},	security MCSs
MCS_3	{`attacker`.e_3, `attacker`.e_4, `attacker`.e_5}	

one event (single points of failure) also plain security MCSs should be avoided by adding more (safety) BEs. Also, the more MCSs exist in a given SeCFT,

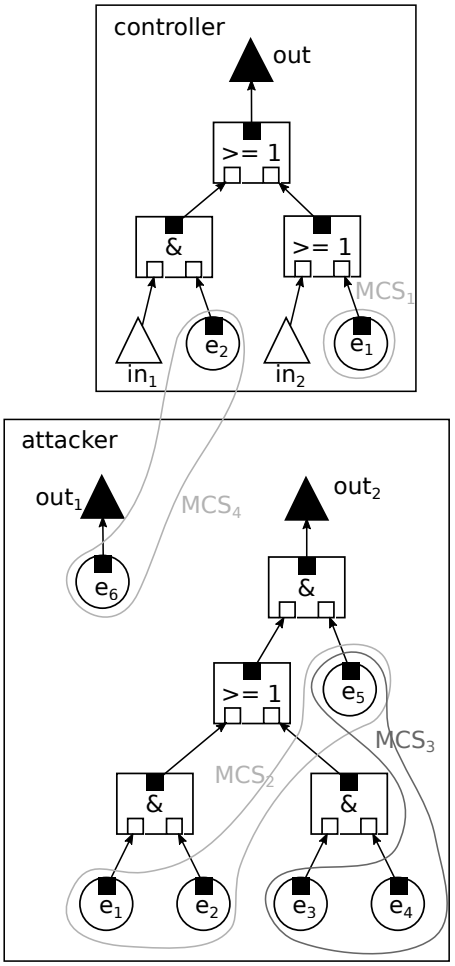


Fig. 4.4. An SeCFT with highlighted MCSs from the example in Fig. 4.2

the more opportunities for the TE exist which indicates a higher vulnerability of the system with respect to this TE.

4.3.2 Importance of Basic Events

Another goal of an analysis is to determine the importance of BEs. The importance shows how much of an impact a BE has on a TE. BEs that are part of more than one MCS are more important than the ones that are only part of one MCS. But the size of MCSs is also a factor. BEs in smaller MCSs are more important than the ones in larger MCSs. More accurate importance analysis is possible within a quantitative analysis. Referred to the example, if the importance of BEs within MCSs of the same size is compared, this results in the order of importance shown in Table 4.3.

Table 4.3. Qualitative importance of BEs

MCS size	basic events (in descending order of importance)
1	$c.e_1$
2	$\{c.e_2, a.e_6\}$
3	$a.e_5 > \{a.e_1, a.e_2, a.e_3, a.e_4\}$

4.4 Quantitative Analysis of SeCFTs

A quantitative analysis is conducted if more accurate statements about the system safety are necessary than the results from a qualitative analysis which are mainly the determination and preliminary ordering of MCSs. Further analysis of the MCSs brings more insight into the contribution of single failure scenarios (MCSs) to the vulnerability of the system. Systems have to fulfill customer requirements or to comply with certain standards which may include minimum sizes of MCSs or maximum values for system failure probabilities. A quantitative analysis therefore has several goals [Vesely 81, IEC 61025 06]:

- to determine the rating of the TE under consideration to compare it to the given requirements from standards or customers,
- to determine ratings of the individual MCSs to determine the MCS that has the biggest contribution to the TE (the most probable failure scenario),
- and derived from the previous ones: to determine where countermeasures would have the most effect.

A quantitative analysis of an SeCFT starts with a quantitative evaluation of its MCSs. The first step here is to assign probabilities to safety events and likelihoods to security events. During the assignment of likelihood values to

security events it should be kept in mind that those security events belonging to the same MCS can influence each other.

After the determination of the MCSs there are two possibilities to order them: According to size and type (see qualitative analysis in Sect. 4.3) or according to type and ratings (probability and likelihood). An ordering simply according to the ratings is not possible for all MCSs in general because of the incomparability of probabilities and likelihoods (see also Sect. 4.1). For each MCS a tuple rating (P, L) is calculated according to the rules described in Sect. 4.2.3. For probabilities this means the value for the MCS is the product of all probabilities of the contained events. (Under the precondition that all events are independent which is usually given for safety events.) For the likelihood of an MCS the minimum of all likelihoods of the included events is determined. This approach was previously described in [Steiner 13a] and [Steiner 15].

Each type of MCSs can be ordered by itself. To compare two Minimal Cut Sets MCS_1 and MCS_2 with tuple ratings (P_1, L_1) and (P_2, L_2) Table 4.4 is used. The table defines a partial order for MCSs.

Table 4.4. Conditions for an order of mixed MCSs according to two tuples (P_1, L_1) and (P_2, L_2)

	$P_1 < P_2$	$P_1 = P_2$	$P_1 > P_2$
$L_1 < L_2$	$\texttt{MCS}_1 < \texttt{MCS}_2$	$\texttt{MCS}_1 < \texttt{MCS}_2$	undefined
$L_1 = L_2$	$\texttt{MCS}_1 < \texttt{MCS}_2$	$\texttt{MCS}_1 = \texttt{MCS}_2$	$\texttt{MCS}_1 > \texttt{MCS}_2$
$L_1 > L_2$	undefined	$\texttt{MCS}_1 > \texttt{MCS}_2$	$\texttt{MCS}_1 > \texttt{MCS}_2$

The two states named *undefined* have to be analyzed further. To order two MCSs that fall into these states, the ordering has to be prioritized either according to probability or to likelihood. The resulting ordered list of MCSs reflects the relative criticality of the represented failure scenarios. Higher ratings here correspond to a higher criticality and vice versa. To find out if the system complies with the given requirements, the list of MCSs is filtered according to the requirements (e.g.: "show me all MCSs with size ≤ 2", "$P > 10^{-7}$" or "$L \geq$ medium"). The results then are the failure scenarios that require countermeasures. Table 4.5 shows the MCSs from the example ordered according to probability and likelihood with the values for the tuple rating calculated in Sect. 4.2.3.

As mentioned earlier, requirements can define boundary values for MCSs in size or rating, but usually the main requirement is a boundary value for the rating of the TEs: "The system shall not fail with a probability more than ..." The TE probability can be calculated either as the sum of the probabilities of all MCSs if only `AND` and `OR`-gates are used. This defines an upper boundary for the probability $P'(TE)$ (see equation 4.11 and also Sect. 2.1.1).

Table 4.5. MCSs from example 1 ordered according to probability and likelihood

MCS number	tuple rating (P, L)
MCS_4	$(10^{-7}, 1)$
MCS_1	$(10^{-9}, -)$
MCS_2	$(-, 1)$
MCS_3	$(-, 1)$

$$P(TE) \leq P'(TE) = \sum_{i=1}^{n} P(MCS_i) \qquad , i, n \in \mathbb{N}, n \text{ number of MCSs} \quad (4.11)$$

The other variant is to calculate $P(TE)$ using the Binary Decision Diagram (BDD) algorithm described in Sect. 2.1.1 which returns the exact probability value. To adapt the BDD algorithm to SeCFTs only the BEs with an assigned probability value are considered for the calculation as already discussed in Sect. 4.2.3.

The likelihood $L(TE)$ of the TE is calculated from the likelihoods of all MCSs as defined in equation 4.9 and 4.10 in Sect. 4.2.2:

$$L'\left(\bigvee_{i=1}^{n} X_i\right) = 1 - \prod_{i=1}^{n}(1 - \frac{L(X_i)}{m}) \quad , i, n, m \in \mathbb{N}$$

$$L(x) = \min\left(\lceil m \times L'(x) \rceil, m\right) \quad , m \in \mathbb{N}$$

A simpler estimation of the TE likelihood can be obtained by just summing the likelihood values of all MCSs.

Table 4.6 shows again the chosen values for probabilities and likelihood in the example. Concerning the example, the rating of the TE results in the

Table 4.6. BEs together with their ratings from the example

Basic Event	tuple rating (P, L)
$c.e_1$	$(10^{-9}, -)$
$c.e_2$	$(10^{-7}, -)$
$a.e_1$	$(-, 2)$
$a.e_2$	$(-, 3)$
$a.e_3$	$(-, 3)$
$a.e_4$	$(-, 1)$
$a.e_5$	$(-, 1)$
$a.e_6$	$(-, 1)$

following:

$$P'(TE) = 10^{-9} + 10^{-7} = 1.01 \times 10^{-7}$$
$$P(TE) = 1.009999999 \times 10^{-7}$$
$$L(TE) = 3$$

For all BEs an importance value can be calculated corresponding to the Fussell-Vesely (FV) importance. As with all the other calculations so far, the importance will be calculated separate for probability and likelihood. To recall, the FV importance of an event e is calculated as:

$$FV_e = \frac{\sum_{i=1}^{m_e} P(MCS_{e_i})}{\sum_{j=1}^{m} P(MCS_j)}$$

Whereas MCS_e are the MCSs that contain event e, m_e is the number thereof, and m the number of all MCSs. A similar value can be calculated for the likelihood:

$$FV_{L_e} = \frac{\sum_{i=1}^{m_e} L(MCS_{e_i})}{\sum_{j=1}^{m} L(MCS_j)}$$

Using these importance values for probabilities and likelihood, the events can be identified that have the most impact on the rating of the TE. This knowledge is used to decide where is the system countermeasures should be implemented.

As to the nature of MCSs, their probabilities can be reduced in two ways: either by reducing the probability of a single event (or more), or by adding more events. The likelihood decreases if a new event with a lower likelihood is added, or the minimum likelihood is decreased. Adding more events should also decrease the criticality of an MCS, but does not directly affect the likelihood value.

The decision for or against a countermeasure for a certain MCS can also be based on partially available data. For example an MCS that includes at least one extremely improbable event can be given a low priority even if not all probabilities of BEs are known. As probabilities of independent BEs are multiplied in order to calculate the MCS probability, the probability of the event with the lowest probability in an MCS defines an upper bound for the MCS probability.

$$P(MCS_i) \leq P(e_{ij}) \qquad , \forall e_{ij} \in MCS_i, i, j \in \mathbb{N} \tag{4.12}$$

If for the analysis of the security events indicators as attack cost or attacker type are used, those can also be used to prioritize MCSs: Known attack costs can be used as lower bounds of attacks, and systems can be designed to withstand only some attacker types.

With the proposed extensions of established techniques qualitative analyses of SeCFTs are possible taking into account safety as well as security causes of system failures.

4.5 Summary: Analysis Process of SeCFTs

This section will summarize the complete analysis process to give an overview of all required tasks. In Fig. 4.5 the analysis process is shown after the tree is created.

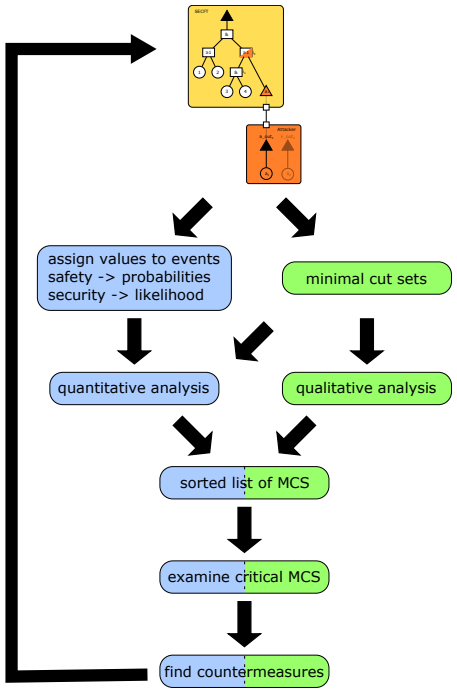


Fig. 4.5. The extended analysis process after the SeCFT is created

The analysis process can be divided into two parts. First, the MCSs are determined which serve as a basis for a qualitative and a quantitative analysis. In the qualitative analysis (right side of the figure) the MCSs are prioritized according to type and size to find the most critical ones. For the quantitative analysis (left side of the figure) values are assigned to all BEs, probabilities to safety events and likelihoods to security events. From these values the resulting ratings of the MCSs and the TEs are calculated. The ratings of the MCSs and TEs now can be compared to the requirements. If the requirements are met, the analysis is finished here, and it can serve as evidence for meeting the requirements. If not, a further analysis is done to find out where and how the system has to be improved. Using the ratings, the MCSs can be prioritized. The MCSs are sorted according to type and either size, probability or likelihood. Thereby the most critical MCSs (the ones that either directly violate the requirements or contribute substantially to it) are iden-

tified. Against these most critical MCSs countermeasures have to be found. After the implementation of those countermeasures the SeCFTs are extended to incorporate them and a new analysis begins with the new SeCFTs to check if the countermeasures have the desired effect.

5

Evaluation of the Approach

In the following chapter the analyses of the systems that were conducted during the course of this work within different projects are presented. The first on is an analysis of the robot RAVON, followed by an adaptive cruise control, a smart farming scenario, and an automatic infusion pump.

As the analyses of the examples were conducted during the development of the method, they reflect the particular development stages (cf. Sect. 3.3.2). The first analysis of RAVON was modeled similar to a Fault Tree (FT). The whole tree was modeled in one component, and the attacks were modeled as additional Basic Events (BEs). The second example, the adaptive cruise control, was modeled with separate components, but the attacks were still additional BEs. The last two examples, the smart farming scenario and the infusion pump, show the last iteration of the modeling process. The systems were modeled with different components, vulnerabilities were inserted as input ports, and the attacks were modeled in a separate attacker component.

The following section describes the workflow that was followed to conduct the analyses of the systems using Security-enhanced Component Fault Trees (SeCFTs).

5.1 Tool Support

The example systems were modeled using the CFT modeling tool ESSaRel in version 5.0.2 [ESSaRel 09]. Safety Basic Events are rated with a probability just like they would be in a Component Fault Tree (CFT) analysis. The distinction between safety BEs and security BEs is done via the description field of the BEs. For that purpose security BEs are marked in the description with the tag `[sec]`. Security BEs are rated with a likelihood value. This likelihood is also put into the description field, from where it can be read later in the analysis. From the viewpoint of ESSaRel the SeCFTs do not differ from standard CFTs, except that not all BEs have a probability value assigned to them. Thereby, it is possible to use ESSaRel to calculate the Minimal Cut

Sets (MCSs) for the desired Top Event. The resulting MCSs can be exported into an XML file which can be used for further analysis. Based on the exported list of MCSs, the first analyses were conducted manually using Excel respective LibreOffice Calc.

To simplify this second step, a small prototype tool was developed that uses the exported list of MCSs and the ESSaRel model to calculate the safety and security ratings for MCSs and Top Events. The tool requires the XML file containing the MCSs and the folder containing the ESSaRel model. This data is imported and all ratings are calculated. The tool returns a sorted list of the MCSs according to either size and type, safety probabilities, or security likelihoods. With that the most critical MCSs depending on the system requirements can be determined. Figure 5.1 shows a screenshot of the tool prototype.

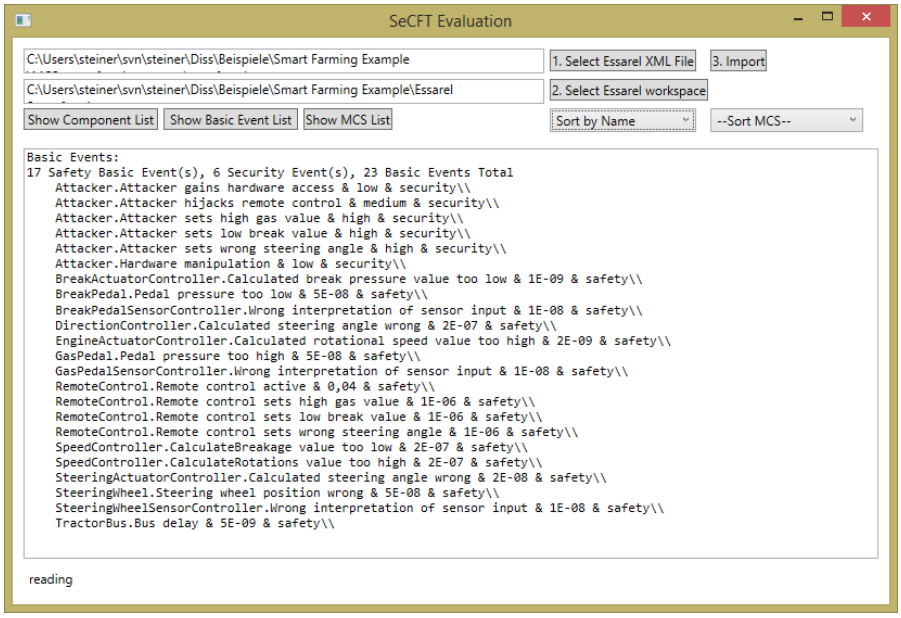


Fig. 5.1. A screenshot of the prototype tool for the analysis of SeCFTs

As this tool is a prototype with only a minimum of functionality, there are some restrictions in the usage. The tool cannot handle more than one proxy of a component, and the name of a proxy has to be the same as the name of the instantiated component. This is because ESSaRel does save the BEs included in an MCS using the proxy name, but the model of the component is saved under the component name. To merge the MCS information with the ratings of the BEs, the tool has to map proxy names to component names. The value of the description field also has to be in very strict bounds for the security

BEs. The tag `[sec]` has to be the first string value in the description, followed by either a space character or a comma and the rating of the security BE. This rating in the current version can be one of `[low]`, `[medium]`, or `[high]` in exactly that notation. Other values can also the used in theory, but they would have to be implemented in the code of the tool.

5.2 Analysis Example: RAVON

The first analysis using the approach of this thesis was done during the BMBF[1] funded project ViERforES[2]. In the next sections first the demonstrator RAVON is described, and afterwards the analysis and the results are presented.

5.2.1 Description RAVON

RAVON is a mobile robot developed by the Robotics Research Lab at the University of Kaiserslautern[3] to research off-road navigation of autonomous vehicles [Armbrust 09, Schäfer 11]. Using different sensor systems it is able to perceive its environment. Figure 5.2 shows the robot outside the university during a presentation. The technical data is summarized in Table 5.1.

Table 5.1. RAVON technical data

Length	2.35 m
Width	1.4 m
Height	1.8 m (highest point: GPS antenna)
Weight	750 kg
Power Supply	8 sealed lead batteries (12 V, 55 A h each)
Operation Time	about 4 h
Drive	four wheel drive with four independent electric motors
Steering	front and rear wheel steering via linear motors
Velocity	10 km h^{-1} max.
Controller	2 Motorola 56F803 DSPs
Computer	4 Industrial PCs
Floor Clearance	0.3 m
Wheel Diameter	0.75 m
Max. Slope	100 % (at 7 km h^{-1})

Laser scanners in the front and the back measure distance and detect obstacles. Two horizontally mounted 2D laser scanners are used for short range

[1] Bundesministerium für Bildung und Forschung (Federal Ministry of Education and Research)

[2] Virtuelle und Erweiterte Realität für höchste Sicherheit und Zuverlässigkeit Eingebetteter Systeme (ViERforES), `http://www.vivera.org/ViERforES/` (accessed 2015-11-09)

[3] `http://agrosy.informatik.uni-kl.de` (accessed 2015-11-09)

Fig. 5.2. The RAVON robot 2009 at the University of Kaiserslautern

obstacle detection. In the front of the robot two stereo camera systems are installed for mid-range obstacle detection (up to 20 m) and for far-range terrain classification (up to 40 m). In addition to the cameras a 3D laser scanner is used for better detection of obstacles in distances up to 15 m. Two spring-mounted bumpers in the front and the back are used for tactile detection of dense vegetation and act as emergency stop triggers.

The different sensor inputs are merged and evaluated in the control software. The control software is realized using the behavior-based control architecture iB2C [Proetzsch 10]. Depending on the evaluated sensor data it is decided if an obstacle is lying in the planned way of the robot, and if so, evasion behaviors make sure that the obstacle is not hit.

RAVON is driven by four battery-powered electric wheel hub motors bringing it to a maximum velocity of $10\,\mathrm{km\,h^{-1}}$. Its total mass of 750 kg brings up a potential risk of serious damage to itself and the environment including injuries in collisions with humans. It is therefore imperative that the system is analyzed for residual risks and to check if the built-in safeguards are sufficient. Unfavorable environmental conditions could lead to a non-detection of an obstacle by one or more sensors. As a last resort, if the sensors did not detect an obstacle, a safety chain has been implemented. It stops the robot immediately in case the bumper sensors are triggered. Figure 5.3 shows the structure of the safety chain implemented in RAVON.

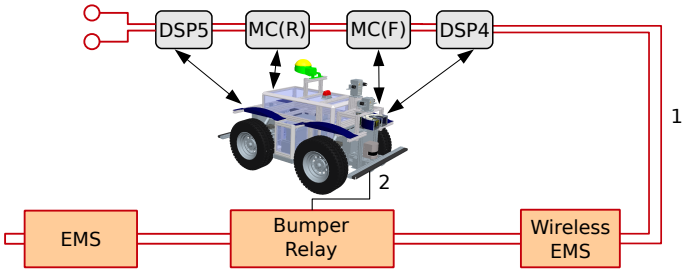


Fig. 5.3. The RAVON safety chain

The four motors are controlled by one motor controller each. The controllers for the front wheels are in the same box (MC(F)), as are the ones for the rear wheels (MC(R)). The motor controllers monitor the safety chain (connection 1 in Fig. 5.3). In case of a fault of the controllers, they also can interrupt the safety chain themselves. In case of an interruption of the safety chain the wheel motors are decelerated, and the holding brake blocks the wheels to prevent inadvertent movement. Two Digital Signal Processors (DSPs) are responsible for the steering. Additionally they monitor the safety chain and stop the steering motors in case of an interruption. In the front and the back of the robot the aforementioned bumpers are installed. On the outside of each bumper a safety edge is mounted that can interrupt the safety chain if too much pressure is applied to it. Both safety edges are connected to a bumper relay (connection 2 in Fig. 5.3) that switches the safety chain.

A human operator that is present during test drives also has the possibility to trigger an emergency stop via remote control (wireless EMS) or a button on the robot itself (EMS) which also interrupts the safety chain. Any interruption of the safety chain immediately stops all motors and should therefore prevent harm to the system or its environment.

5.2.2 Analysis

Because the safety chain implemented in RAVON is the last resort if the other sensors did not detect an obstacle, it is one of the most critical systems and was therefore chosen as an analysis example. The analysis of this example is the first application of the method described in Sect. 3.3.2. The system was modeled in one single component, comparable to a Fault Tree. Possible attacks were modeled as Basic Events. They were not refined further as this was not necessary for the modeled level of abstraction. Potential refinements would have been directly in that tree. In Fig. 5.4 the extended CFT is shown as an overview. The modeled failure scenario is that RAVON cannot come to a complete stop in front of an obstacle and therefore causes damage to the obstacle or to RAVON itself. In the figure safety events are unfilled circles, whereas attacks are depicted as filled circles that are additionally marked

with [sec] in the description. The tree can be divided into two parts: On the left side below the topmost OR-gate (*safety chain does not decelerate Ravon sufficiently*) the causes emanating from the motor controllers (*MC initiate no emergency stop*) are analyzed. Three classes of faults at the motor controllers are considered: faults in the control software of the controllers, wrong parameters for the brake maneuver, or not initiating the emergency stop because the corresponding signal is not interpreted correctly. For all three of those there is a possibility of random faults or targeted attacks as the cause.

The right subtree refers to the interruption (or lack thereof) of the safety chain (*safety chain is not interrupted*). The safety chain is divided in three parts that all have to fail together for the system to fail. An emergency stop button (*EMS does not interrupt SC*), a wireless emergency stop button (*wireless EMS does not interrupt SC*), and the bumper relay which is triggered by the safety edges on the bumpers (*bumper relay does not interrupt SC*) all have to fail so that the safety chain is not interrupted in case of an emergency. The emergency stop button could be not reachable, or the button could not be pressed due to mechanical or electrical reasons that could be caused by random faults or sabotage. The wireless emergency stop button is modeled as sender and receiver that could be sabotaged or fail randomly. Additionally, an attacker could jam the radio signal to hinder an emergency stop trigger. The bumper relay does not switch if it has a defect, it is bypassed, or the safety edges do not send the switching signal.

Plausible orders of magnitude were chosen for the probabilities of safety events. For emergency stop buttons there exist device specifications that state a failure probability of 1×10^{-7}. Based on that the values for the other system components, for which no values were available, were estimated. The security ratings in this model were chosen from a scale of three steps {low, medium, high}. The difficulty of all considered attacks depends directly on the level of access to the system. So the rating depends on the level of access an attacker has to the system. Direct physical access leads to a *low* likelihood because this is generally the most difficult to achieve. Access to the wireless sender of the wireless emergency stop button is rated with *medium*, and jamming the radio signal is rated with *high*, because this attack can be conducted without direct access to the system from a distance. There are 21 Basic Events (BEs), 11 thereof are safety BEs and 10 security BEs. The ratings of all considered BEs are listed in Table 5.2.

5.2.3 Results

During the qualitative analysis of the SeCFT of the safety chain of RAVON 126 Minimal Cut Sets (MCSs) were found. Table A.1 in Appendix A.1 shows all found MCSs in the order in which they were calculated by ESSaRel. The 6 MCSs shown in Table 5.3 have a size of 1 and are ordered according to their type. They are single points of failure in this model and should be analyzed further. The remaining 120 MCSs have a size of 3.

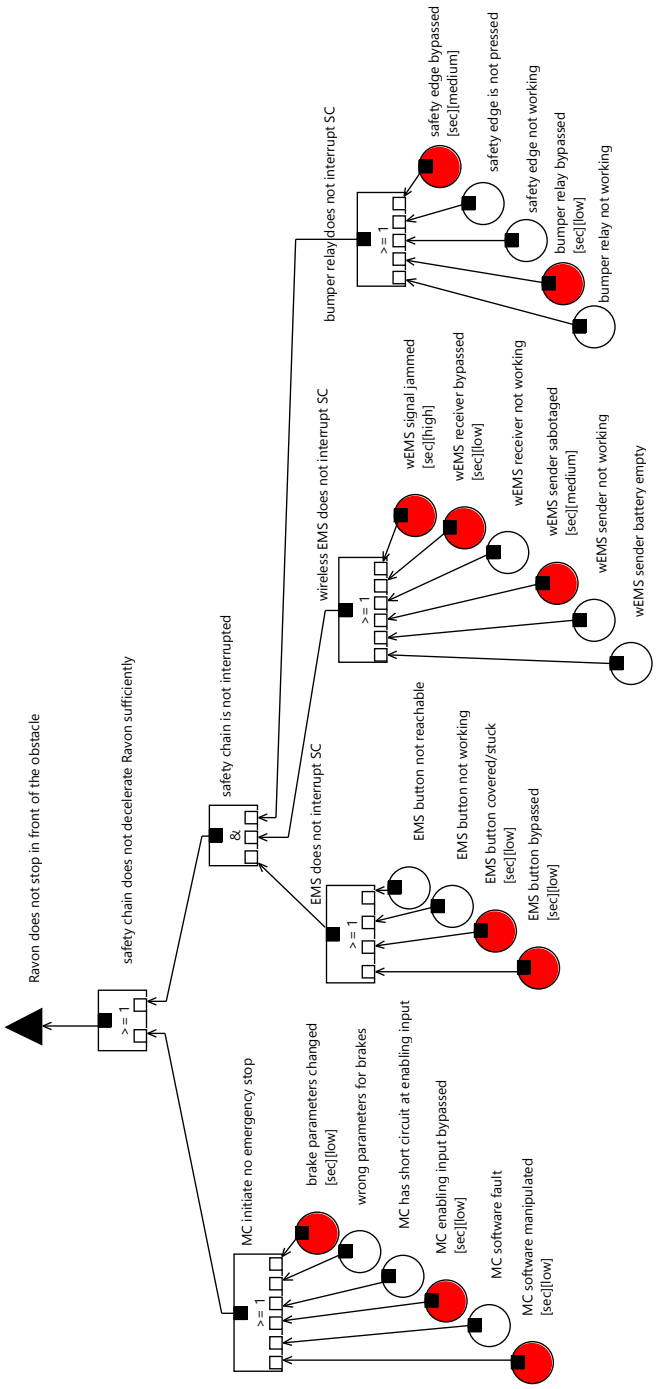


Fig. 5.4. A CFT of the RAVON safety chain

Table 5.2. Ratings of the Basic Events in the RAVON safety chain

Component.Basic event	Rating	Type
SafetyChain.brake parameters changed	low	security
SafetyChain.bumper relay bypassed	low	security
SafetyChain.bumper relay not working	1E-07	safety
SafetyChain.EMS button bypassed	low	security
SafetyChain.EMS button covered/stuck	low	security
SafetyChain.EMS button not reachable	1E-05	safety
SafetyChain.EMS button not working	1E-07	safety
SafetyChain.MC enabling input bypassed	low	security
SafetyChain.MC has short circuit at enabling input	1E-05	safety
SafetyChain.MC software fault	1E-07	safety
SafetyChain.MC software manipulated	low	security
SafetyChain.safety edge bypassed	medium	security
SafetyChain.safety edge is not pressed	1E-05	safety
SafetyChain.safety edge not working	1E-07	safety
SafetyChain.wEMS receiver bypassed	low	security
SafetyChain.wEMS receiver not working	1E-07	safety
SafetyChain.wEMS sender battery empty	1E-05	safety
SafetyChain.wEMS sender not working	1E-07	safety
SafetyChain.wEMS sender sabotaged	medium	security
SafetyChain.wEMS signal jammed	high	security
SafetyChain.wrong parameters for brakes	1E-05	safety

Table 5.3. MCSs of size 1 found during a qualitative analysis of the RAVON safety chain

MCS ID	Basic Event	Type
2	SafetyChain.brake parameters changed	security
3	SafetyChain.MC software manipulated	security
5	SafetyChain.MC enabling input bypassed	security
1	SafetyChain.wrong parameters for brakes	safety
4	SafetyChain.MC software fault	safety
6	SafetyChain.MC has short circuit at enabling input	safety

Overall, there are 21 safety MCSs and 90 mixed MCSs. 15 of all MCSs are security MCSs that would not have been found without the consideration of attacks as additional causes for failures. The security MCSs should receive special attention, as their occurrence only depends on the actions of an attacker. The later quantitative analysis showed that their likelihood is *low*. So countermeasures only have to be taken if MCSs of size 1 have to be avoided at all cost.

A following quantitative analysis yields a rating of the Top Event *Ravon does not stop in front of the obstacle* of (1.8×10^{-4}, high). The relative large number of security MCSs explains the high likelihood for the security part of the SeCFT. The most critical MCSs according to the security likelihood and the safety probability of this system were determined. Table 5.4 shows the MCSs with likelihood values of *high*. As can be seen in the table, those are mixed MCSs, and their safety probabilities are very low. Thus it appears

that these MCSs are not critical despite their high security likelihood, because they contain safety events with a very low probability of occurrence.

Table 5.4. Most critical MCSs according to the security likelihood of the RAVON safety chain

MCS ID	Type	Rating	Basic Events
55	mixed	(1E-10, high)	SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed
104	mixed	(1E-12, high)	SafetyChain.EMS button not reachable SafetyChain.wEMS signal jammed SafetyChain.safety edge not working
103	mixed	(1E-12, high)	SafetyChain.EMS button not reachable SafetyChain.wEMS signal jammed SafetyChain.bumper relay not working
59	mixed	(1E-12, high)	SafetyChain.EMS button not working SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed
108	mixed	(1E-14, high)	SafetyChain.EMS button not working SafetyChain.wEMS signal jammed SafetyChain.safety edge not working
107	mixed	(1E-14, high)	SafetyChain.EMS button not working SafetyChain.wEMS signal jammed SafetyChain.bumper relay not working

Table 5.5 shows the most critical MCSs with a probability of at least 1×10^{-5}. Those failure scenarios would also have been found using standard CFTs. As the table shows, the system has no failure scenarios with a higher probability than 1×10^{-5}.

The analysis of the RAVON safety chain using SeCFTs shows that there are security MCSs in the system which may directly cause the system failure. Those security MCSs would not have been found in a classical Fault Tree Analysis (FTA) using CFTs. Some of these attack scenarios also consist of only one security event. Depending on the requirements for the system, those have to be avoided or at least argued that they are not critical. The quantitative analysis rated those single event security MCSs with a *low* likelihood. So, as long as the grounds on which they were rated do not change, they can be seen as non-critical MCSs.

Table 5.5. Most critical MCSs according to the safety probability of the RAVON safety chain

MCS ID	Type	Rating	Basic Events
1	safety	(1E-05, –)	SafetyChain.wrong parameters for brakes
6	safety	(1E-05, –)	SafetyChain.MC has short circuit at enabling input
8	mixed	(1E-05, medium)	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed
57	mixed	(1E-05, medium)	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS signal jammed
63	mixed	(1E-05, low)	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver bypassed
15	mixed	(1E-05, low)	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed
75	mixed	(1E-05, low)	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed
83	mixed	(1E-05, low)	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed
79	mixed	(1E-05, low)	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed
29	mixed	(1E-05, low)	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed
21	mixed	(1E-05, low)	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed
24	mixed	(1E-05, low)	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed
19	mixed	(1E-05, low)	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed
30	mixed	(1E-05, low)	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed
26	mixed	(1E-05, low)	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed
49	mixed	(1E-05, low)	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS receiver bypassed
71	mixed	(1E-05, low)	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed
67	mixed	(1E-05, low)	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS signal jammed

5.3 Analysis Example: Adaptive Cruise Control

This second example describes the modeling of an Adaptive Cruise Control (ACC) of a car. The ACC example was created during the BMBF[4] funded project ViERforES II[5]. It was used in [Steiner 13a] to illustrate the modeling approach of this thesis.

5.3.1 Description Adaptive Cruise Control

In the considered system vehicles are to drive in a platoon with other vehicles. A defined distance should be kept between the vehicles. A vehicle sends its measured distances to both leading and following vehicles, and its velocity to the following vehicle. Figure 5.5 shows a schematic of the system with two cars.

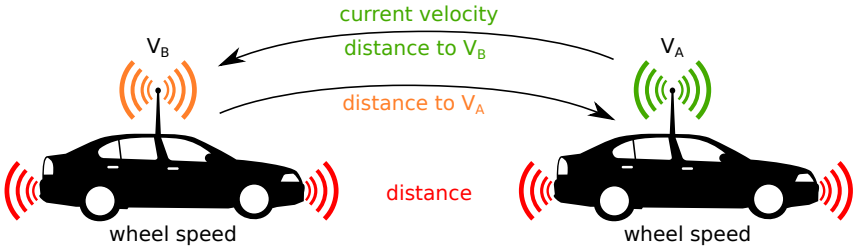


Fig. 5.5. The ACC example system

The used system architecture is shown in Fig. 5.6. Components not used in the analysis scenario are painted in gray. It is based on the system described in [Spang 10]. Four wheel speed sensors provide the own current speed of the vehicle to the speedometer. The distance to the other vehicle is provided by front and rear distance sensors. Each vehicle receives the measured velocity and distance values from the leading vehicle by an antenna component. Distance values and foreign velocities are used by the communication system which provides them to the control logic unit. From the distance and the velocity the new target velocity is calculated by the ACC component which then increases or decreases the vehicle speed. The received distance and the measured one are fused in the communication system. The received velocity is used as is.

[4] Bundesministerium für Bildung und Forschung (Federal Ministry of Education and Research)

[5] Virtuelle und Erweiterte Realität für höchste Sicherheit und Zuverlässigkeit Eingebetteter Systeme – Zweite Phase (ViERforES II), `http://www.vivera.org/ViERforES/` (accessed 2015-11-09)

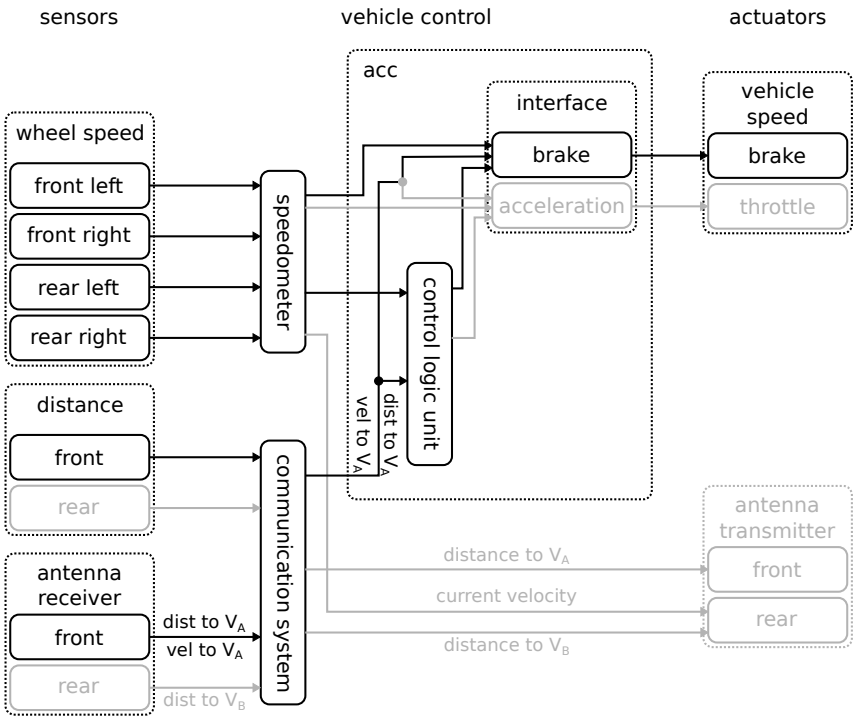


Fig. 5.6. The modeled architecture of the ACC

5.3.2 Analysis

The analyzed scenario is: There are two vehicles in a platoon, a leading vehicle V_A and a following vehicle V_B. Due to a failure in the ACC system V_B is too fast and crashes against V_A.

The components of this example system were modeled as separate CFTs. Attacks were added as additional Basic Events (BEs) in the respective CFTs. Comparable to the previous example, the attacks were not refined further than one BE. This modeling example illustrates an intermediate stage of the development of SeCFTs in between the modeling as FTs as in the first example in Sect. 5.2, and the final SeCFTs as in the following examples in Sects. 5.4 and 5.5.

In Fig. 5.7 one can see the high level overview of the CFT model of the ACC. An enlarged version including all subtrees can be found in Appendix A.2 in Figs. A.1 to A.11. The CFTs for the ACC system are built along the system components and their composition. Accordingly, there is one CFT per component and different nesting levels. In the highest level CFT shown in Fig. 5.7 directly below the Top Event *crash with* V_A is the brake CFT, followed by the vehicle CFT in which all control tasks are handled, and at the bottom there are the sensor CFTs. The vehicle CFT contains the CFTs

for speedometer, the actual ACC, and the communication system. Thereby the CFTs are built according to the data flow from sensors to actuators using an overall number of 10 components, whereas the wheel speed sensor CFT is instantiated four times. Possible attacks on the vehicle sensors and the wireless communication were considered as well as the more theoretical attacks to manipulate vehicle components directly.

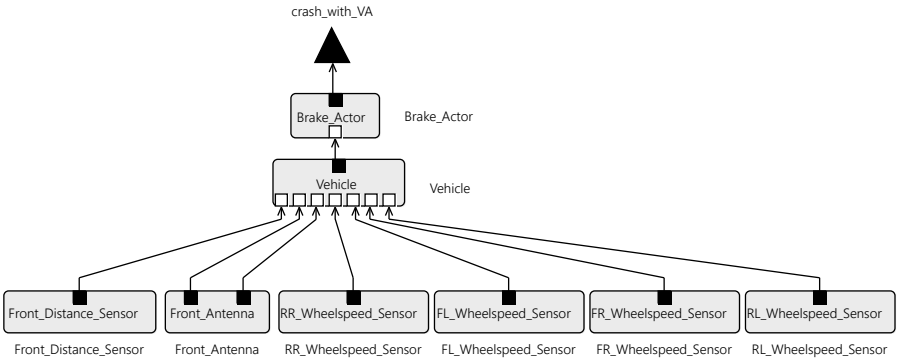


Fig. 5.7. A high level CFT of the ACC example system

The values for the safety events for the quantitative analysis are taken from [Spang 10]. Security events were rated using an estimation for difficulty of access to the system and difficulty of conducting the attack (high difficulty results in a low rating). The final rating is the average of both values. There are 39 Basic Events (BEs) in all CFTs, thereof 13 safety BEs and 26 security BEs. The ratings of all considered BEs are listed in Table 5.6.

5.3.3 Results

In a qualitative analysis of the SeCFT for the Top Event *crash with* V_A 124 separate MCSs were found. Table A.2 in Appendix A.2 shows all found MCSs in the order in which they were calculated by ESSaRel. In Table 5.7 the 14 MCSs of size of 1 are shown. They are single points of failure in this model and should be analyzed further. The remaining 110 MCSs have a size of 2.

Overall, there are 28 safety MCSs and 54 mixed MCSs. 42 of all MCSs are security MCSs that would not have been found without the consideration of attacks as additional causes for failures. The security MCSs, especially as there are a lot of them in this system, should receive special attention, as their occurrence only depends on the actions of an attacker.

A following quantitative analysis yields a rating for the Top Event *crash with* V_A of (0.05, high). The system has a very high failure probability, so further analysis is necessary to find the most critical MCSs to find countermeasures to minimize the failure probability. The relative large number of

Table 5.6. Ratings of the Basic Events in the ACC system

Component.Basic event	Rating	Type
ACC.tampering deceleration value too low	low	security
Brake Actor.Brake reacts not as expected	1E-05	safety
Brake Actor.spoofing a too low value	low	security
Brake Interface.tampering brake disabled/deceleration value too low	low	security
Brake Interface.tampering distance VA too high	low	security
Brake Interface.tampering own velocity too low	low	security
Brake Interface.tampering velocity VA too high	low	security
Communication System.tampering distance value too high	low	security
Communication System.tampering too high velocity value	low	security
Control Logic Unit.tampering distance value too high	low	security
Control Logic Unit.tampering output brake disabled	low	security
Control Logic Unit.tampering own velocity too low	low	security
Control Logic Unit.tampering velocity VA value too high	low	security
FL Wheelspeed Sensor.denial of service	low	security
FL Wheelspeed Sensor.Noise in Sensor Magnet Field	1E-05	safety
FL Wheelspeed Sensor.Sensor malfunction	1E-05	safety
FL Wheelspeed Sensor.Spoofing sensor values	low	security
FR Wheelspeed Sensor.denial of service	low	security
FR Wheelspeed Sensor.Noise in Sensor Magnet Field	1E-05	safety
FR Wheelspeed Sensor.Sensor malfunction	1E-05	safety
FR Wheelspeed Sensor.Spoofing sensor values	low	security
Front Antenna.Received distance to VA is too high	0.017	safety
Front Antenna.Received velocity of VA too high	0.015	safety
Front Antenna.Spoofing of a distance signal	medium	security
Front Antenna.Spoofing velocity too high	medium	security
Front Antenna.Tampering Distance Signal to a too high value	high	security
Front Antenna.Tampering Velocity too high value	high	security
Front Distance Sensor.Assumed sonic velocity too high	2E-05	safety
Front Distance Sensor.denial of service	high	security
Front Distance Sensor.Echo time too high	0.001	safety
Front Distance Sensor.Spoofing sensor values	medium	security
RL Wheelspeed Sensor.denial of service	low	security
RL Wheelspeed Sensor.Noise in Sensor Magnet Field	1E-05	safety
RL Wheelspeed Sensor.Sensor malfunction	1E-05	safety
RL Wheelspeed Sensor.Spoofing sensor values	low	security
RR Wheelspeed Sensor.denial of service	low	security
RR Wheelspeed Sensor.Noise in Sensor Magnet Field	1E-05	safety
RR Wheelspeed Sensor.Sensor malfunction	1E-05	safety
RR Wheelspeed Sensor.Spoofing sensor values	low	security
Speedometer.tampering velocity too low value	low	security

security MCSs explains the high likelihood for the security part of the SeCFT. The most critical MCSs according to the security likelihood and the safety probability of this system were determined. Table 5.8 shows the MCSs with likelihood values of *high*. MCS_{114} (a single event MCS) and MCS_6 are security MCSs that are rated with a high likelihood of occurrence. MCS_{120} and MCS_8 also have a high safety probability. Therefore they are critical from both security and safety points of view. Countermeasures against these failure scenarios have to be found. These should introduce new safety Basic Events into MCS_{114} and MCS_6 to transform them to mixed MCSs that are not only depending on the actions of an attacker, and reduce the probabilities of all MCSs to an acceptable level.

Table 5.9 shows the MCSs with a probability of at greater than 1×10^{-5}. There are additional 49 MCS with a probability of 1×10^{-5} which are shown

Table 5.7. MCSs of size 1 found during a qualitative analysis of the ACC

MCS ID	Type	Basic Event
3	security	Brake Interface.tampering brake disabled/deceleration value too low
4	security	Control Logic Unit.tampering output brake disabled
105	security	Brake Actor.spoofing a too low value
107	security	ACC.tampering deceleration value too low
108	security	Brake Interface.tampering distance VA too high
109	security	Brake Interface.tampering own velocity too low
110	security	Brake Interface.tampering velocity VA too high
111	security	Speedometer.tampering velocity too low value
112	security	Communication System.tampering distance value too high
113	security	Communication System.tampering too high velocity value
114	security	Front Antenna.Tampering Velocity too high value
115	security	Front Antenna.Spoofing velocity too high
106	safety	Brake Actor.Brake reacts not as expected
116	safety	Front Antenna.Received velocity of VA too high

Table 5.8. Most critical MCSs according to the security likelihood of the ACC

MCS ID	Type	Rating	Basic Events
114	security	(– , high)	Front Antenna.Tampering Velocity too high value
6	security	(– , high)	Front Antenna.Tampering Distance Signal to a too high value Front Distance Sensor.denial of service
120	mixed	(0.017, high)	Front Antenna.Received distance to VA is too high Front Distance Sensor.denial of service
8	mixed	(0.001, high)	Front Antenna.Tampering Distance Signal to a too high value Front Distance Sensor.Echo time too high
7	mixed	(2E-05, high)	Front Antenna.Tampering Distance Signal to a too high value Front Distance Sensor.Assumed sonic velocity too high

in Appendix A.2 in Table A.3. These MCSs would also have been found in a classical FTA using CFTs. Depending on the requirements for the system the probabilities of those MCSs should be lowered to an acceptable level. They should be dealt with accordingly.

Table 5.9. Most critical MCSs according to the safety probability of the ACC

MCS	Type	Rating	Basic Events
120	mixed	(0.017, high)	Front Antenna.Received distance to VA is too high Front Distance Sensor.denial of service
119	mixed	(0.017, medium)	Front Antenna.Received distance to VA is too high Front Distance Sensor.Spoofing sensor values
116	safety	(0.015, –)	Front Antenna.Received velocity of VA too high
8	mixed	(0.001, high)	Front Antenna.Tampering Distance Signal to a too high value Front Distance Sensor.Echo time too high
122	mixed	(0.001, medium)	Front Antenna.Spoofing of a distance signal Front Distance Sensor.Echo time too high
7	mixed	(2E-05, high)	Front Antenna.Tampering Distance Signal to a too high value Front Distance Sensor.Assumed sonic velocity too high
121	mixed	(2E-05, medium)	Front Antenna.Spoofing of a distance signal Front Distance Sensor.Assumed sonic velocity too high
124	safety	(1.7E-05, –)	Front Antenna.Received distance to VA is too high Front Distance Sensor.Echo time too high

The analysis of the Adaptive Cruise Control system using SeCFTs shows that there are security MCSs in the system which may directly cause the system failure. The qualitative analysis shows that the system contains attack scenarios that are single points of failure. The followed quantitative analysis even shows that those attack scenarios are very likely to happen. The detailed results show that the velocity value coming from the front antenna is critical because it can be tampered with easily, and it can be measured wrong with a high probability. It is even a single point of failure and can lead to a crash of two vehicles. The other value in the front antenna component, the distance to the other car, is not as critical as the velocity because it is measured additionally by the own car. But it is shown that both values can be manipulated by an attacker, and there are no countermeasures in the shown system design.

5.4 Analysis Example: Smart Farming

The third application example of the approach was an analysis of a smart farming system. It was also created during the course of the project ViERforES II[6], same as the previous example. In [Steiner 13b], a technical report for the project, it was used to test the whole approach from modeling to analysis.

5.4.1 Description: Smart Farming

The system under study is a living lab called *smart farming* from the Fraunhofer IESE[7]. It is used to show how the work with agricultural machines can be automated and linked to sensors and remote services. In the system there are different vehicles (Harvester, Mower, and Tractor), sensors (cameras, GPS, and weather sensors), implements for the tractor, and mobile devices for remote control and planning. In Fig. 5.8 the components of the smart farming ecosystem are shown.

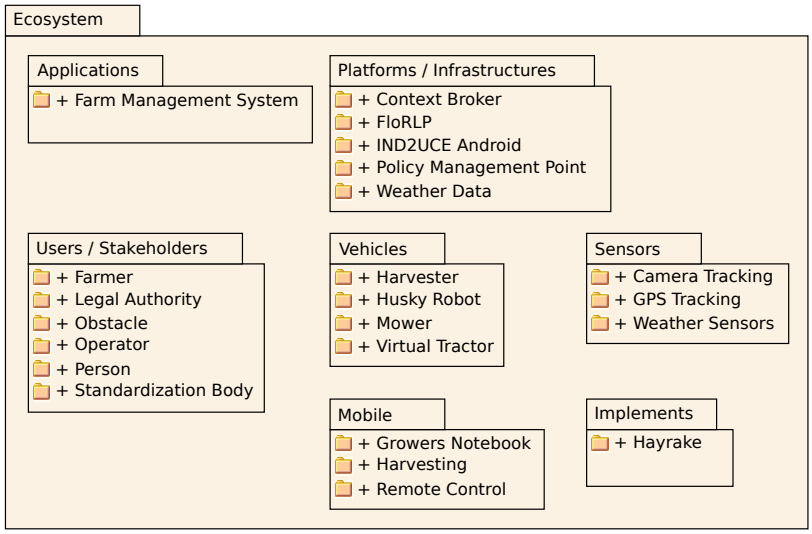


Fig. 5.8. An overview of the smart farming ecosystem

From the safety point of view the tractor and its remote control are especially interesting, because tractors are heavy machines moving around that

[6] Virtuelle und Erweiterte Realität für höchste Sicherheit und Zuverlässigkeit Eingebetteter Systeme – Zweite Phase (ViERforES II), `http://www.vivera.org/ViERforES/` (accessed 2015-11-09)

[7] `http://www.iese.fraunhofer.de/de/presse/press_archive/press_2013/PM_2013_06_050313_cebit.html` (accessed 2015-11-09)

may harm other machines or people. In the smart farming scenario the tractor drives a predefined trajectory over the field which avoids known stationary obstacles and the field's boundaries. Using the remote control it can be driven manually. The combination of automated tractor and remote control is analyzed using the SeCFT approach of this thesis.

First, the considered system components are described. Figure 5.9 shows an overview of the system components of the tractor and their interconnections. The hardware components of the sensors and actuators are not present in the model, the model ends at the respective controllers. There are four groups of components: sensors, logic components, actuators, and buses. The ones that are not used for the analysis are also mentioned for a complete overview, but they are set in brackets in the description and in gray in the figure.

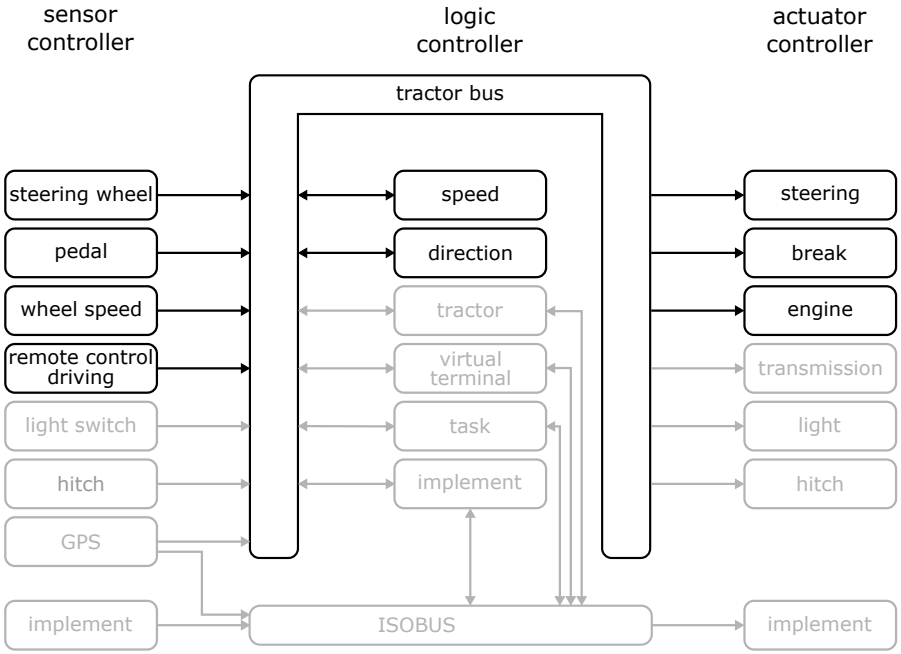


Fig. 5.9. An overview of the smart farming system components

Sensors:

Steering Wheel Sensor Controller: The steering wheel sensor controller collects the current rotation angle of the steering wheel and provides this information for further processing.

Pedal Sensor Controller: The pedal sensor controller collects information about the pedal position.

Wheel Speed Sensor Controller: The wheel speed sensor controller determines the current turning speed of the tractor's wheels.

Remote Control Driving Sensor Controller: The remote control driving sensor controller receives and processes control commands for steering, pedal, light, and hitch from a remote control system. It creates the corresponding sensor controller messages for the command as if it was generated from the actual sensor controller.

(GPS Sensor Controller): The GPS sensor controller determines the current position of the tractor based on GPS data.

(Implement Sensor Controller): The implement sensor controller is the software unit in the implement to monitor a physical sensor and send sensor data messages via the bus.

(Light Switch Sensor Controller): The light switch sensor controller determines the current position of the tractor's headlight and turn indication switches.

(Hitch Switch Sensor Controller): The hitch switch sensor controller detects the current position of the switch in the tractor to control the position of the hitch.

Logic:

Speed Controller: The speed controller serves as the main computing software unit to control the tractor's current driving speed. It processes the gas and break position data, computes required engine rotational speed and break pressure, and sends according control commands to the actuators.

Direction Controller: The direction controller serves as the main computing software unit to control the tractor's current direction. It processes steering wheel position data, computes the required wheel rotation angle, and creates and sends steering commands to the steering actuator controller.

(Tractor Controller): The tractor controller serves as the central computing unit for the tractor behavior. It takes sensor input signals and based on this computes control commands that it sends to actuator controllers. Thereby it also performs analyses to ensure correct and safe behavior of the tractor. Additionally it serves as the interface between the tractor bus and the ISOBUS.

(Virtual Terminal): The virtual terminal is the software control unit for the tractor's main graphical user interface. It visualizes tractor and control operational data as well as interfaces for the user to control the tractor and the implement. It also allows the user to create tasks.

(Task Controller): The task controller executes predefined tasks created with a farm management system or with the tractor display. It executes these commands by triggering the single actions a task consists of, based on timing or sensor data, such as the current position. To trigger the single actions, it creates control messages that are sent to the implement or tractor controller. It also logs farming activities and transfers the data to the farm management for later analysis.

(Implement Controller): The implement controller is the central computing software unit in the implement. It processes sensor data, computes an implement behavior, and creates and sends according control commands to the corresponding actuators.

Actuators:

Steering Actuator Controller: The steering actuator controller changes the angle of the tractors front wheels according to the direction controller's command.

Break Actuator Controller: The break actuator controller adjusts the break pressure according to the commands given by the speed controller.

Engine Actuator Controller: The engine actuator controller adjusts the engine's rotational speed.

(Transmission Actuator Controller): The transmission actuator controller is responsible to set the tractor's gear.

(Light Actuator Controller): The light actuator controller turns the tractor's headlights and turn indicator lights on or off.

(Hitch Actuator Controller): The hitch actuator controller is responsible for raising or lowering the hitch, and to physically connect the hitch to the tractor.

(Implement Actuator Controller): The implement actuator controller is responsible for moving the implement depending on the type of implement.

Buses:

Tractor Bus: The tractor bus is the channel for the communication between the controller components within the tractor.

(ISOBUS): The ISOBUS is the channel for communication between the tractor and the implement.

5.4.2 Analysis

The analyzed failure scenario in the smart farming example is that a tractor is hitting an obstacle in its path due to non-responding steering or too high speed. The tractor is either driven by someone on the driver's seat or via a remote control.

As stated before, this analysis was the first one that completely follows the modeling approach described in Sect. 3.4. For each system component a separate CFT was modeled. Potential vulnerabilities were considered as additional input ports of the respective components. The actual attacks are modeled in a separate attacker component.

Figure 5.10 shows the highest level SeCFT of the modeled system. An enlarged version including all subtrees can be found in Appendix A.3 in Figs. A.12 to A.26. The SeCFT shown in Fig. 5.10 is built in layers. For the tractor to crash into an obstacle, it has either to drive too fast, so that it cannot brake in time, or to steer in the wrong direction, so that the obstacle is hit. The SeCFT is modeled along the signal flow coming from the controls via the tractor bus to the actuators. Only the control elements are modeled as possible points of attack. In theory all other components could also be considered as points of attack. But such attacks would require fundamental changes in the physical systems which means great effort on one hand, and a high chance to be detected on the other. Therefore, they were not considered in this model. Besides, the analysis was about showing the additional risks of a connected vehicle with a possibility for remote control. The attacker was modeled as one that wants to harm the owner of the tractor with a minimal risk of detection. In total, the model consists of 13 system components and one attacker component.

The attacker component was kept simple in this example. How the attacks could be conducted in detail is only modeled as a next step if the analysis results in attacks that highly influence the safety of the system. For that step a security expert would be necessary. The attacker component modeled for this example is shown in Fig. 5.11.

Table 5.10 shows all 23 Basic Events (BEs) in the scenario. 17 safety BEs and 6 security BEs were identified. The ratings for safety BEs are probabilities of occurrence, for security BEs it is a three-valued scale high – medium – low representing a likelihood of occurrence. The probability of failure of the engine and break controllers were assumed to be in the range of Safety Integrity Level (SIL) 4. The other values were estimated accordingly. The likelihood values of the security BEs were estimated depending on the difficulty of accessing the necessary system components.

5.4.3 Results

In a qualitative analysis of the SeCFT for the Top Event *Tractor deviates from its planned trajectory and damages an object or hurts a person* 20 Minimal Cut

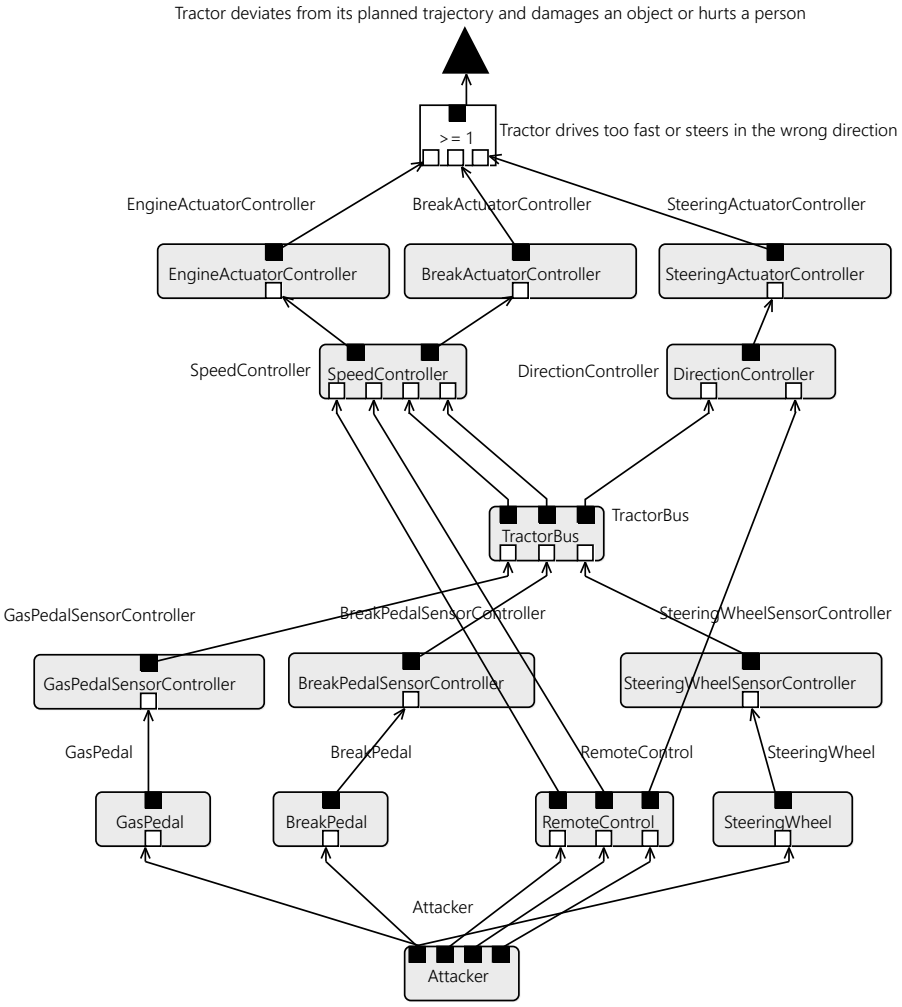


Fig. 5.10. A high level SeCFT of the smart farming example system

Sets (MCSs) were found. Table A.4 in Appendix A.3 shows all found MCSs in the order in which they were calculated by ESSaRel. 13 MCSs contained only one BE, 4 contained two BEs and 3 contained three BEs. The MCSs of size 1 are shown in Table 5.11. Those single points of failure are especially critical and should be avoided by altering the system to add countermeasures which result in more BEs in those MCSs.

The classification according to safety and security events results in 16 safety MCSs, 1 security MCS, and 3 mixed MCSs. The found security MCS describes a failure scenario that would not have been found without the consideration of attacks as additional causes for failures.

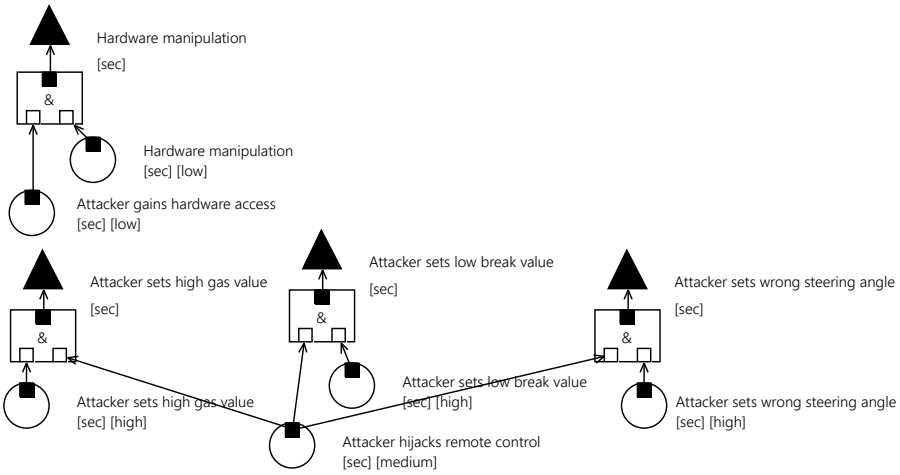


Fig. 5.11. The SeCFT of the attacker component in the smart farming example system

Table 5.10. Ratings of the Basic Events in the smart farming scenario

Component.Basic event	Rating	Type
Attacker.Attacker gains hardware access	low	security
Attacker.Attacker hijacks remote control	medium	security
Attacker.Attacker sets high gas value	high	security
Attacker.Attacker sets low break value	high	security
Attacker.Attacker sets wrong steering angle	high	security
Attacker.Hardware manipulation	low	security
BreakActuatorController.Calculated break pressure value too low	1E-09	safety
BreakPedal.Pedal pressure too low	5E-08	safety
BreakPedalSensorController.Wrong interpretation of sensor input	1E-08	safety
DirectionController.Calculated steering angle wrong	2E-07	safety
EngineActuatorController.Calculated rotational speed value too high	2E-09	safety
GasPedal.Pedal pressure too high	5E-08	safety
GasPedalSensorController.Wrong interpretation of sensor input	1E-08	safety
RemoteControl.Remote control active	0.04	safety
RemoteControl.Remote control sets high gas value	1E-06	safety
RemoteControl.Remote control sets low break value	1E-06	safety
RemoteControl.Remote control sets wrong steering angle	1E-06	safety
SpeedController.CalculateBreakage value too low	2E-07	safety
SpeedController.CalculateRotations value too high	2E-07	safety
SteeringActuatorController.Calculated steering angle wrong	2E-08	safety
SteeringWheel.Steering wheel position wrong	5E-08	safety
SteeringWheelSensorController.Wrong interpretation of sensor input	1E-08	safety
TractorBus.Bus delay	5E-09	safety

A following quantitative analysis yields a rating for the Top Event *Tractor deviates from its planned trajectory and damages an object or hurts a person* of (0.11, high). The high probability and likelihood requires measures to either lower the probability or likelihood. The most critical MCSs according to the security likelihood and the safety probability of this system were determined. The highest likelihood values in this analysis are *medium*, but their combination yields a *high* value for the Top Event. The results from the qual-

Table 5.11. MCSs of size 1 found during a qualitative analysis of the smart farming scenario

MCS ID	Basic Event	Type
8	safety	DirectionController.Calculated steering angle wrong
9	safety	SteeringWheelSensorController.Wrong interpretation of sensor input
10	safety	SteeringWheel.Steering wheel position wrong
11	safety	EngineActuatorController.Calculated rotational speed value too high
12	safety	SpeedController.CalculateRotations value too high
13	safety	TractorBus.Bus delay
14	safety	GasPedalSensorController.Wrong interpretation of sensor input
15	safety	GasPedal.Pedal pressure too high
16	safety	BreakActuatorController.Calculated break pressure value too low
17	safety	SpeedController.CalculateBreakage value too low
18	safety	BreakPedalSensorController.Wrong interpretation of sensor input
19	safety	BreakPedal.Pedal pressure too low
20	safety	SteeringActuatorController.Calculated steering angle wrong

itative analysis show that there are 4 MCSs which contain security events. One even consists of only security events (MCS_7). This is not that critical as it was rated with *low*. But the other three MCSs (MCS_2, MCS_3, MCS_4) have a *medium* security likelihood and a high safety probability (0.04). Table 5.12 shows the MCSs with the highest likelihood values. In this example, they are also the MCSs with the highest safety probabilities. The probabilities of these MCSs are so high that measures should be implemented to reduce the risk. The remote control is usually only a problem if it is not monitored. So it should only be activated if someone actively watches the machine. That someone could deactivate the remote control in case of a problem. By implementing a measure that monitors the remote control for unauthorized access the risk of a successful attack could also be reduced.

Table 5.12. Most critical MCSs according to the security likelihood and the safety probability of the smart farming scenario

MCS ID	Type	Rating	Basic Events
2	mixed	(0.04, medium)	RemoteControl.Remote control active Attacker.Attacker hijacks remote control Attacker.Attacker sets high gas value
3	mixed	(0.04, medium)	RemoteControl.Remote control active Attacker.Attacker hijacks remote control Attacker.Attacker sets low break value
4	mixed	(0.04, medium)	RemoteControl.Remote control active Attacker.Attacker hijacks remote control Attacker.Attacker sets wrong steering angle

This example shows that the additional consideration of security during safety analysis finds additional causes for system failures that were not found before. In this case, a fifth of the MCSs (4 out of 20) are containing security events which would not have been considered in a classical safety analysis.

5.5 Analysis Example: Infusion Pump

The last example is an analysis of a generic model of an automatic infusion pump which was enriched by a known vulnerability of an existing infusion pump. The analysis was conducted during the project *Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments*[8] *(EMC²)* and is used to illustrate the analysis method in [Steiner 15].

5.5.1 Description: Infusion Pump

Basis of the analysis is a model of a generic patient controlled analgesia pump developed at the University of Pennsylvania [Arney 09]. This complex model was simplified, and a subsystem of it was analyzed. Figure 5.12 shows a high-level view of the architecture of a generic patient controlled analgesia pump with all modeled components. The simplified model used for this analysis consists of a *PumpUnit* that receives inputs from a user (a patient or hospital staff) and controls a *PumpDriver* which executes the actual pumping task. The PumpUnit is monitored by a *FlowRateMonitor* that raises an alarm if the delivery rate of the pump exceeds predefined limits. During the year 2015 several vulnerabilities of real infusion pumps in use became known. The security part of the analysis is inspired by the security flaw of the infusion pump Hospira PCA3 that was published in [Scherschel 15].

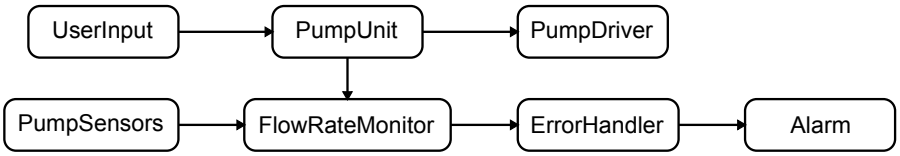


Fig. 5.12. A simplified architecture of a generic patient controlled analgesia pump based on [Arney 09]

5.5.2 Analysis

The analyzed scenario is that an automatic infusion pump delivers a wrong dosage of a drug, and no alarm is raised at the same time. A dosage too high might lead to poisoning of the patient. A dosage too low on the other hand might lead to increased pain, or a general deterioration of the patient's condition. Figure 5.13 shows the high-level SeCFT of the pump model. In Appendix A.4 the SeCFTs of the subcomponents (Figs. A.27 to A.35) can be found. For all of the 7 considered components from Fig. 5.12 one SeCFT was

[8] `http://www.artemis-emc2.eu/` (accessed 2015-11-09)

built. Potential vulnerabilities were added as input ports like in the previous example. The structure of the SeCFTs is according to the data flow of the pump model.

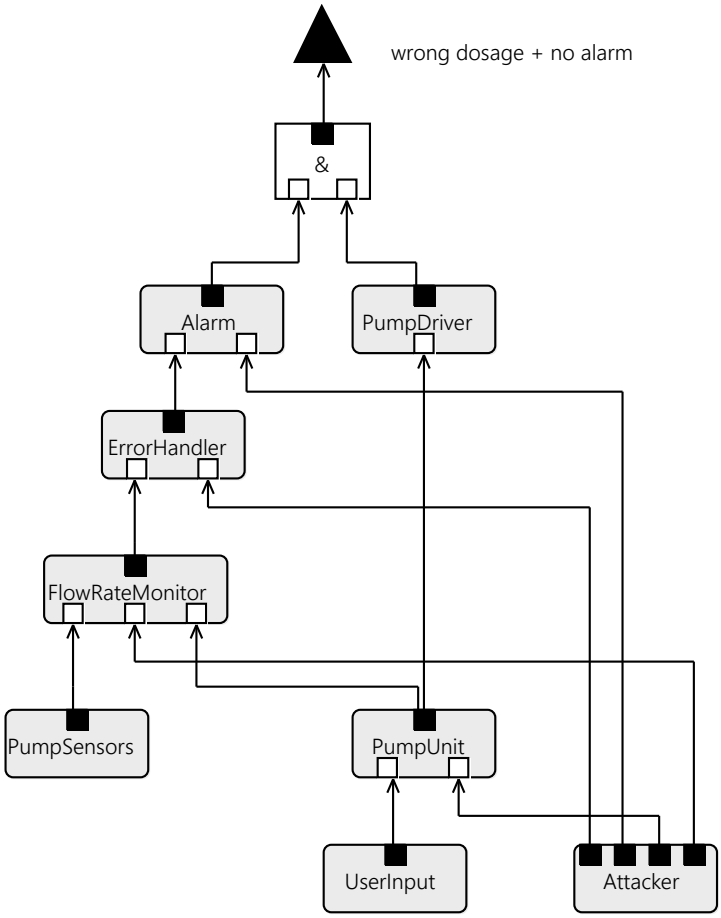


Fig. 5.13. A high-level SeCFT model of the pump model shown in Fig. 5.12

An additional attacker component was created that models the following attack scenario (see Fig. 5.14): The attacker gains access to the Ethernet connector of the infusion pump and logs into the pump operating system (a Linux derivative) via Telnet. In the previously named Hospira infusion pump there was no authentication of the Telnet access, so an attacker could directly log in with root privileges. Having root privileges on the pump operating system, the attacker has the possibility for any action imaginable. For example, an attacker might disable the alarms or change settings such as flow rates.

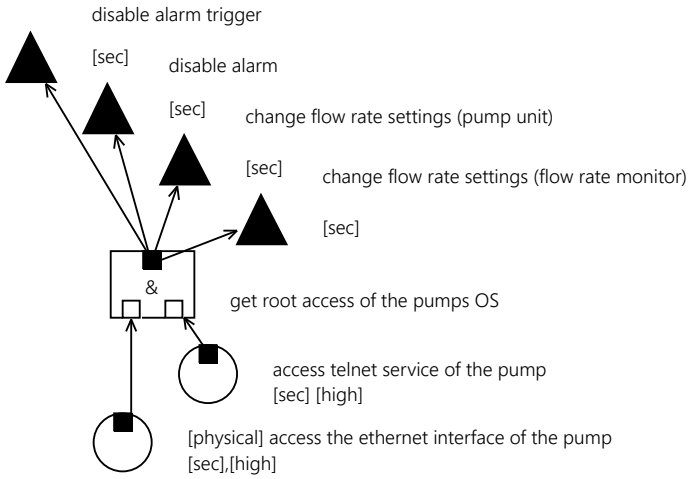


Fig. 5.14. The attacker component of the infusion pump analysis

Table 5.13 shows all 10 Basic Events (BEs) in the scenario. 8 safety BEs and 2 security BEs were identified. Suitable ratings for all BEs were chosen. The required Safety Integrity Level for the individual components was estimated and used as order of magnitude for the rating of the safety BEs. The rating of the security BEs was chosen due to the simplicity of the physical access (to attach an Ethernet cable) and Telnet access (no authentication necessary to get root access). The likelihood for the security BEs in this example is on a three-level scale of {low, medium, high}.

Table 5.13. Ratings of the Basic Events in the infusion pump example

Component.Basic event	Rating	Type
Alarm.alarm fails	1E-07	safety
Attacker.[physical] access the Ethernet interface of the pump	high	security
Attacker.access telnet service of the pump	high	security
ErrorHandler.error handler fails	1E-07	safety
FlowRateMonitor.flow rate monitor fails	1E-07	safety
PumpDriver.pump driver fails	1E-08	safety
PumpSensors.sensors provide wrong values	1E-07	safety
PumpUnit.check of user input fails	1E-07	safety
PumpUnit.pump unit sets wrong values	1E-07	safety
UserInput.user sets wrong values	1E-06	safety

5.5.3 Results

A qualitative analysis of the infusion pump example concerning the Top Event *wrong dosage + no alarm* results in 7 Minimal Cut Sets (MCSs). Of those 7 MCSs 6 are safety MCSs, and one is a security MCS. In this example there are no mixed MCSs. One MCS contains only one Basic Event (BE), and the other 6 contain two BEs. All MCSs are shown in Table 5.14 with their type and ratings.

Table 5.14. All MCSs including their ratings of the infusion pump scenario

MCS ID	Type	Rating	Basic Events
1	safety	(1E-07, –)	PumpUnit.pump unit sets wrong values
2	security	(– , high)	Attacker.[physical] access the Ethernet interface of the pump Attacker.access telnet service of the pump
3	safety	(1E-13, –)	UserInput.user sets wrong values PumpUnit.check of user input fails
4	safety	(1E-15, –)	PumpDriver.pump driver fails Alarm.alarm fails
5	safety	(1E-15, –)	PumpDriver.pump driver fails ErrorHandler.error handler fails
6	safety	(1E-15, –)	PumpDriver.pump driver fails FlowRateMonitor.flow rate monitor fails
7	safety	(1E-15, –)	PumpDriver.pump driver fails PumpSensors.sensors provide wrong values

This simple model is an example for systems that have security flaws that would not have been considered in classical CFTs, but will lead to a failure if an attack is conducted (MCS_2). A subsequent quantitative analysis shows that the safety MCSs are not very critical as the failure probabilities are all below or equal to 1×10^{-7}. But the single security MCS was rated with a likelihood of *high*. This leads to a rating for the Top Event *wrong dosage + no alarm* of (1×10^{-7}, high). The security MCS with the *high* likelihood requires countermeasures to either lower the likelihood, or to introduce safety events with a reasonably low probability into the MCS. Thereby the failure scenario does not only depend on actions of an attacker, but also on the failure of a countermeasure which can be controlled.

5.6 Conclusion

The examples in this chapter show that with a little more effort in comparison to CFTs, additional security causes for system failures can be found with SeCFTs. The initial analysis even can be conducted by analysts that are safety experts with little expertise in security analysis. Only if attacks have to be analyzed in more detail, a security expert should be present for the analysis. Because the safety analysis using SeCFTs is based on CFTs, a qualitative analysis can find all failure scenarios that would have been found with CFTs. Additional failure scenarios can be found that only depend on security Basic Events (BEs) and such that are caused by safety as well as security BEs. These scenarios would not have been found in an analysis using only CFTs. The quantitative analysis can give further insight on the criticality of a Minimal Cut Set (MCS). Even security MCSs consisting of only one BE might be not critical, based on the likelihood value, as long as the grounds for the determination of the likelihood rating do not change. The analysis of the RAVON safety chain is an example for that. So a quantitative analysis may safe resources that would have been spent otherwise.

In the analyzed example systems only likelihood values from a scale of {*low, medium, high*} were used. This is because no more precise or experienced data were available to the author. If such data exists for an analyzed system, it is reasonable that more fine-grained scales would be used. The calculations and analyses of this thesis are kept general enough to handle scales of different granularity. The minimalistic tool used to support the analyses, however, would have to be adapted.

6

Conclusion

This thesis has presented the extension of safety analysis based on Component Fault Trees (CFTs) by failure causes from a security point of view. The still existing separation of the viewpoints of safety and security analysis has shown in numerous examples that it is not suitable for the current and coming networked embedded systems (Cyber-Physical Systems). In most cases during the development of embedded systems only a safety analysis is conducted, security concerns are not even addressed. A complete security analysis on the other hand would mean substantial additional effort, and would produce an unnecessary overhead for most embedded systems. This is why in this thesis an existing safety analysis technique was extended so that no complete separate security analysis is necessary. Only security concerns that might lead to safety issues are analyzed. Therefore, the additional effort is minimized.

The extended safety analysis using Security-enhanced Component Fault Trees (SeCFTs) is based on safety analysis using CFTs and extended by parts of Attack Trees (ATs). In doing so, the scalability and modularity of CFTs is used. The qualitative analysis of SeCFTs does not differ significantly from the one of CFTs. The calculation of Minimal Cut Sets (MCSs) and the classification according to size can be done using the same tools as for CFTs. Only the classification according to the type of the MCSs (safety, security, or mixed) is added. In the quantitative analysis the ratings of MCSs and Top Events are calculated separately for safety probability and security likelihood. The security events are deliberately not rated using probabilities, because these cannot be determined with the same accuracy as for the safety events. If such probabilities would be available, however, they could be used in the analysis. But still then they should be separated from the safety probabilities because security probabilities may change whereas the safety probabilities may most likely not. To avoid misunderstandings about the accuracy of the values of the security likelihood, instead a more coarse scale is used. This results in a tuple of safety probabilities and security likelihoods as the rating for the MCSs and the Top Events. Failure scenarios are evaluated depending on their type. Safety MCSs are handled just as they would in classical CFTs, the risk

of mixed MCSs can be estimated by the probability value as a worst case, and security MCSs can at least be compared to each other to find the most critical ones. Having these prioritizations the application of countermeasures can also be prioritized, so they do not have to be used across-the-board. By the extension of established methods for modeling and analysis, SeCFTs can be applied by safety experts with only little learning effort. Security experts are only necessary for the eventual deeper analysis of critical attacks.

The approach was applied in several analyses of example systems, the safety chain of the off-road robot RAVON, an Adaptive Cruise Control system, a smart farming scenario, and a generic infusion pump. The result in all examples was that by using SeCFTs additional failure causes could be found that would not have been found using classical CFTs. In all of the analyzed examples whole failure scenarios were found which only depended on actions of an attacker, not on failures of system components. These cases should receive extra consideration because they would not have been found in a classical safety analysis. Thereby the additional value of the approach using SeCFTs for safety analysis is shown, and the improvement can be achieved with little extra effort.

Concluding it can be said that with the application of Security-enhanced Component Fault Trees instead of Component Fault Trees the safety of networked embedded systems which are becoming more popular every day can be improved.

A

Appendix

A.1 Evaluation Example: Ravon

Table A.1. MCSs of the SeCFT of the RAVON safety chain

ID	contained Basic Events
1	SafetyChain.wrong parameters for brakes
2	SafetyChain.brake parameters changed
3	SafetyChain.MC software manipulated
4	SafetyChain.MC software fault
5	SafetyChain.MC enabling input bypassed
6	SafetyChain.MC has short circuit at enabling input
7	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed
8	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed
9	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not working SafetyChain.safety edge is not pressed
10	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not working SafetyChain.safety edge bypassed
11	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed
12	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed
13	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not working SafetyChain.safety edge is not pressed
14	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not working SafetyChain.safety edge bypassed
15	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed
16	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not working SafetyChain.bumper relay bypassed
17	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed
18	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not working SafetyChain.bumper relay bypassed
19	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed
20	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed
21	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed
22	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed
23	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed
24	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed
25	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed
26	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed
27	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed
28	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed
29	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed
30	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed
31	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.bumper relay not working
32	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not reachable SafetyChain.safety edge not working
33	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not working SafetyChain.bumper relay not working
34	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button not working SafetyChain.safety edge not working
35	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not reachable SafetyChain.bumper relay not working
36	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not reachable SafetyChain.safety edge not working
37	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not working SafetyChain.bumper relay not working
38	SafetyChain.wEMS sender battery empty SafetyChain.EMS button not working SafetyChain.safety edge not working
39	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.bumper relay not working
40	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button bypassed SafetyChain.safety edge not working
41	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.bumper relay not working
42	SafetyChain.wEMS sender sabotaged SafetyChain.EMS button covered/stuck SafetyChain.safety edge not working
43	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.bumper relay not working
44	SafetyChain.wEMS sender battery empty SafetyChain.EMS button bypassed SafetyChain.safety edge not working
45	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.bumper relay not working
46	SafetyChain.wEMS sender battery empty SafetyChain.EMS button covered/stuck SafetyChain.safety edge not working
47	SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed
48	SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed SafetyChain.wEMS sender not working
49	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS receiver bypassed
50	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS sender not working
51	SafetyChain.EMS button not working SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed
52	SafetyChain.EMS button not working SafetyChain.safety edge is not pressed SafetyChain.wEMS sender not working
53	SafetyChain.EMS button not working SafetyChain.safety edge bypassed SafetyChain.wEMS receiver bypassed
54	SafetyChain.EMS button not working SafetyChain.safety edge bypassed SafetyChain.wEMS sender not working

ID	contained Basic Events	ID	contained Basic Events
55	SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed	82	SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed SafetyChain.wEMS receiver not working
56	SafetyChain.EMS button not reachable SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver not working	83	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed
57	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS signal jammed	84	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver not working
58	SafetyChain.EMS button not reachable SafetyChain.safety edge bypassed SafetyChain.wEMS receiver not working	85	SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed SafetyChain.wEMS signal jammed
59	SafetyChain.EMS button not working SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed	86	SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed SafetyChain.wEMS receiver not working
60	SafetyChain.EMS button not working SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver not working	87	SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver bypassed
61	SafetyChain.EMS button not working SafetyChain.safety edge bypassed SafetyChain.wEMS signal jammed	88	SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed SafetyChain.wEMS sender not working
62	SafetyChain.EMS button not working SafetyChain.safety edge bypassed SafetyChain.wEMS receiver not working	89	SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver bypassed
63	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver bypassed	90	SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed SafetyChain.wEMS sender not working
64	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS sender not working	91	SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed SafetyChain.wEMS signal jammed
65	SafetyChain.EMS button not working SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver bypassed	92	SafetyChain.EMS button bypassed SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver not working
66	SafetyChain.EMS button not working SafetyChain.bumper relay bypassed SafetyChain.wEMS sender not working	93	SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed SafetyChain.wEMS signal jammed
67	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS signal jammed	94	SafetyChain.EMS button covered/stuck SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver not working
68	SafetyChain.EMS button not reachable SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver not working	95	SafetyChain.EMS button not reachable SafetyChain.wEMS receiver bypassed SafetyChain.bumper relay not working
69	SafetyChain.EMS button not working SafetyChain.bumper relay bypassed SafetyChain.wEMS signal jammed	96	SafetyChain.EMS button not reachable SafetyChain.wEMS receiver bypassed SafetyChain.safety edge not working
70	SafetyChain.EMS button not working SafetyChain.bumper relay bypassed SafetyChain.wEMS receiver not working	97	SafetyChain.EMS button not reachable SafetyChain.wEMS sender not working SafetyChain.bumper relay not working
71	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed	98	SafetyChain.EMS button not reachable SafetyChain.wEMS sender not working SafetyChain.safety edge not working
72	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS sender not working	99	SafetyChain.EMS button not working SafetyChain.wEMS receiver bypassed SafetyChain.bumper relay not working
73	SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed SafetyChain.wEMS receiver bypassed	100	SafetyChain.EMS button not working SafetyChain.wEMS receiver bypassed SafetyChain.safety edge not working
74	SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed SafetyChain.wEMS sender not working	101	SafetyChain.EMS button not working SafetyChain.wEMS sender not working SafetyChain.bumper relay not working
75	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver bypassed	102	SafetyChain.EMS button not working SafetyChain.wEMS sender not working SafetyChain.safety edge not working
76	SafetyChain.EMS button covered/stuck SafetyChain.safety edge is not pressed SafetyChain.wEMS sender not working	103	SafetyChain.EMS button not reachable SafetyChain.wEMS signal jammed SafetyChain.bumper relay not working
77	SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed SafetyChain.wEMS receiver bypassed	104	SafetyChain.EMS button not reachable SafetyChain.wEMS signal jammed SafetyChain.safety edge not working
78	SafetyChain.EMS button covered/stuck SafetyChain.safety edge bypassed SafetyChain.wEMS sender not working	105	SafetyChain.EMS button not reachable SafetyChain.wEMS receiver not working SafetyChain.bumper relay not working
79	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS signal jammed	106	SafetyChain.EMS button not reachable SafetyChain.wEMS receiver not working SafetyChain.safety edge not working
80	SafetyChain.EMS button bypassed SafetyChain.safety edge is not pressed SafetyChain.wEMS receiver not working	107	SafetyChain.EMS button not working SafetyChain.wEMS signal jammed SafetyChain.bumper relay not working
81	SafetyChain.EMS button bypassed SafetyChain.safety edge bypassed SafetyChain.wEMS signal jammed	108	SafetyChain.EMS button not working SafetyChain.wEMS signal jammed SafetyChain.safety edge not working

ID	contained Basic Events
109	SafetyChain.EMS button not working
	SafetyChain.wEMS receiver not working
	SafetyChain.bumper relay not working
110	SafetyChain.EMS button not working
	SafetyChain.wEMS receiver not working
	SafetyChain.safety edge not working
111	SafetyChain.EMS button bypassed
	SafetyChain.wEMS receiver bypassed
	SafetyChain.bumper relay not working
112	SafetyChain.EMS button bypassed
	SafetyChain.wEMS receiver bypassed
	SafetyChain.safety edge not working
113	SafetyChain.EMS button bypassed
	SafetyChain.wEMS sender not working
	SafetyChain.bumper relay not working
114	SafetyChain.EMS button bypassed
	SafetyChain.wEMS sender not working
	SafetyChain.safety edge not working
115	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS receiver bypassed
	SafetyChain.bumper relay not working
116	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS receiver bypassed
	SafetyChain.safety edge not working
117	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS sender not working
	SafetyChain.bumper relay not working
118	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS sender not working
	SafetyChain.safety edge not working
119	SafetyChain.EMS button bypassed
	SafetyChain.wEMS signal jammed
	SafetyChain.bumper relay not working
120	SafetyChain.EMS button bypassed
	SafetyChain.wEMS signal jammed
	SafetyChain.safety edge not working
121	SafetyChain.EMS button bypassed
	SafetyChain.wEMS receiver not working
	SafetyChain.bumper relay not working
122	SafetyChain.EMS button bypassed
	SafetyChain.wEMS receiver not working
	SafetyChain.safety edge not working
123	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS signal jammed
	SafetyChain.bumper relay not working
124	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS signal jammed
	SafetyChain.safety edge not working
125	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS receiver not working
	SafetyChain.bumper relay not working
126	SafetyChain.EMS button covered/stuck
	SafetyChain.wEMS receiver not working
	SafetyChain.safety edge not working

A.2 Evaluation Example: Adaptive Cruise Control

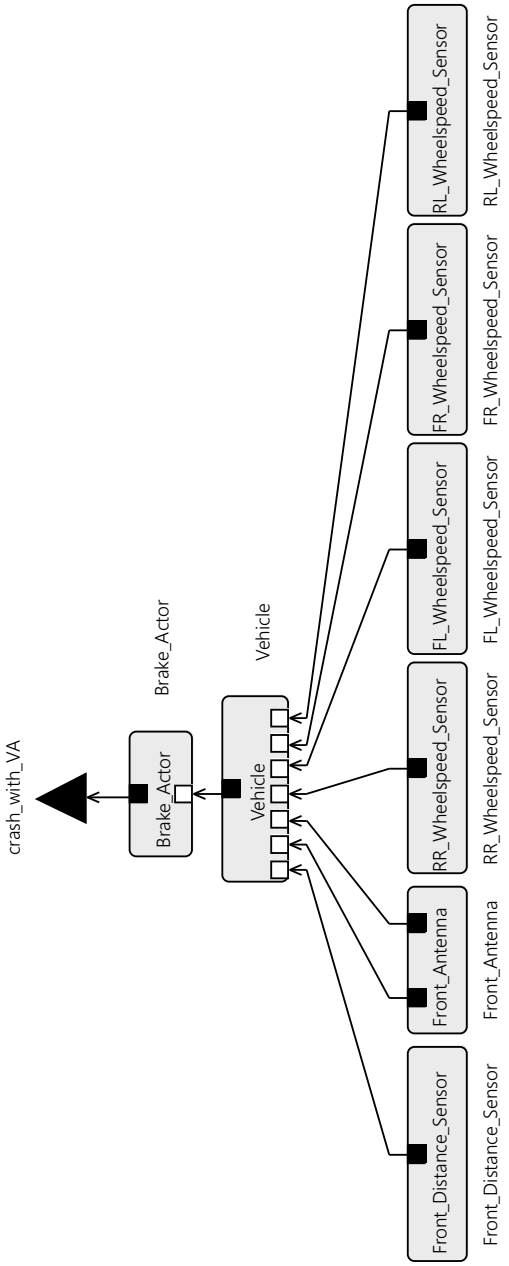


Fig. A.1. SeCFT: ACC.System

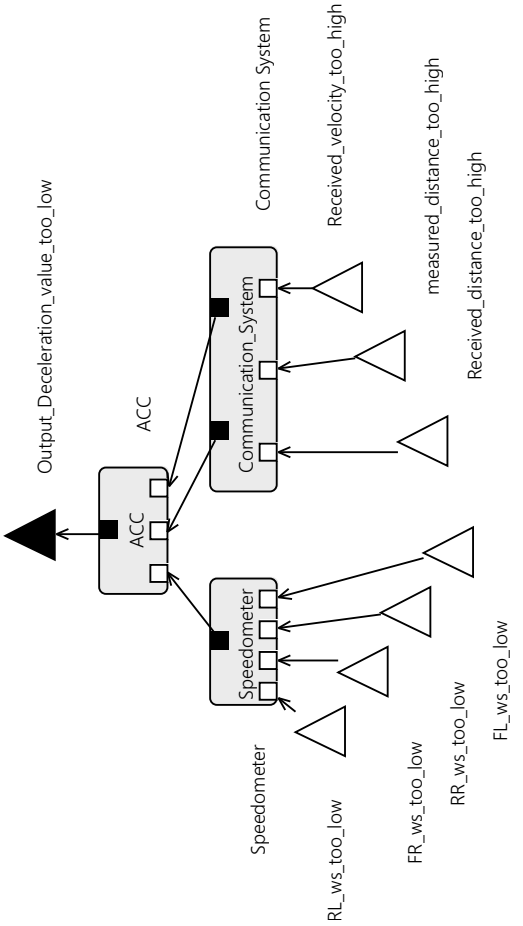


Fig. A.2. SeCFT: ACC.Vehicle

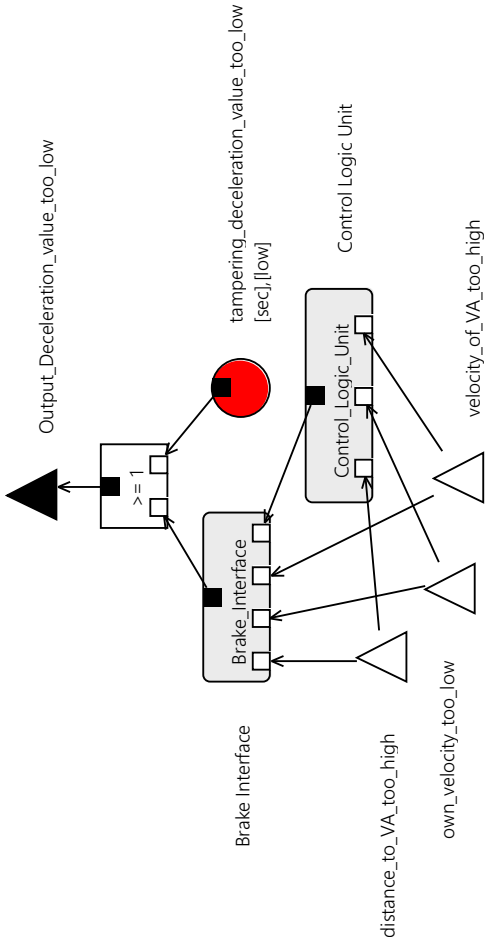


Fig. A.3. SeCFT: ACC.ACC

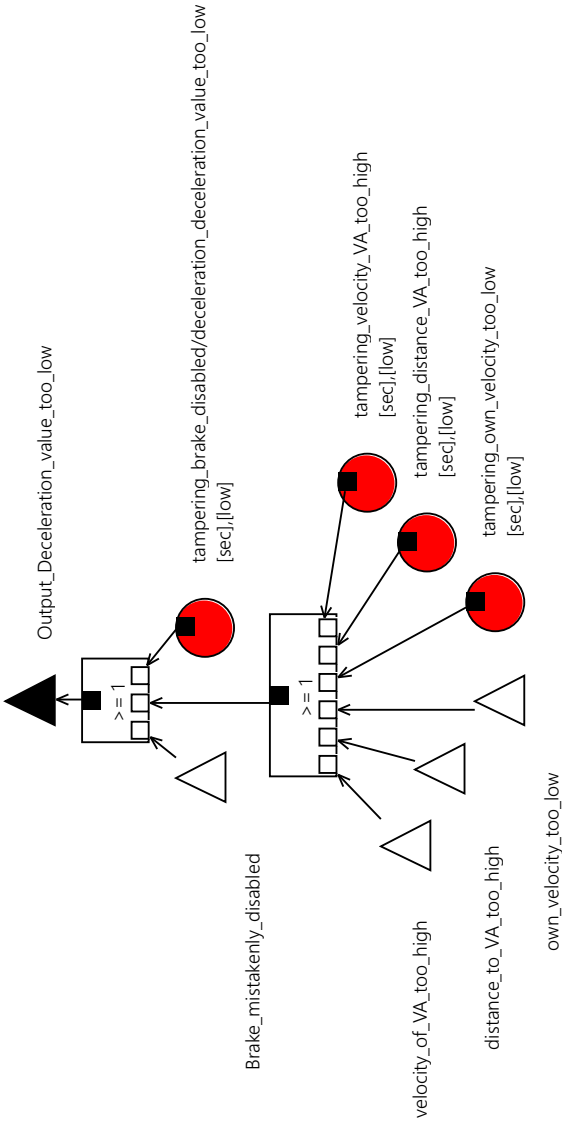


Fig. A.4. SeCFT: ACC.Brake Interface

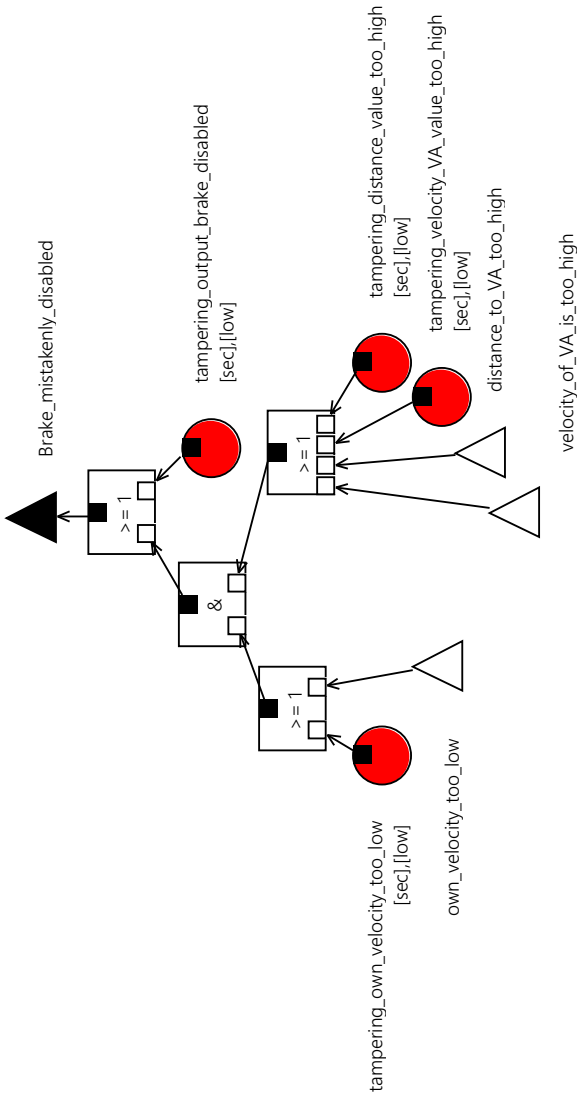


Fig. A.5. SeCFT: ACC.Control Logic Unit

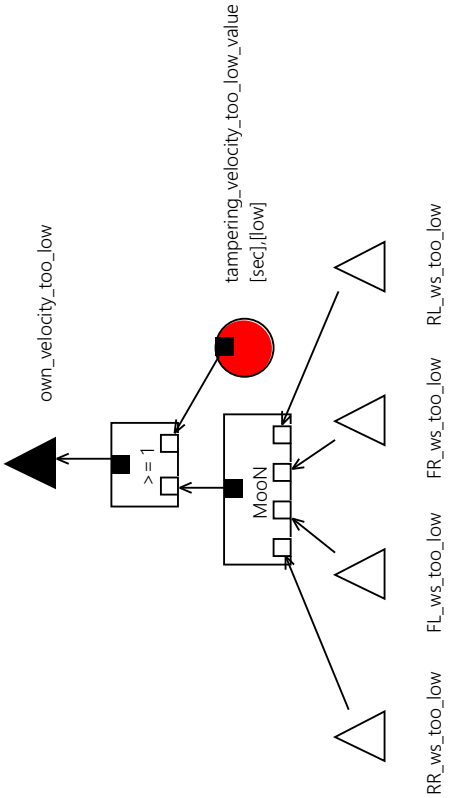


Fig. A.6. SeCFT: ACC.Speedometer

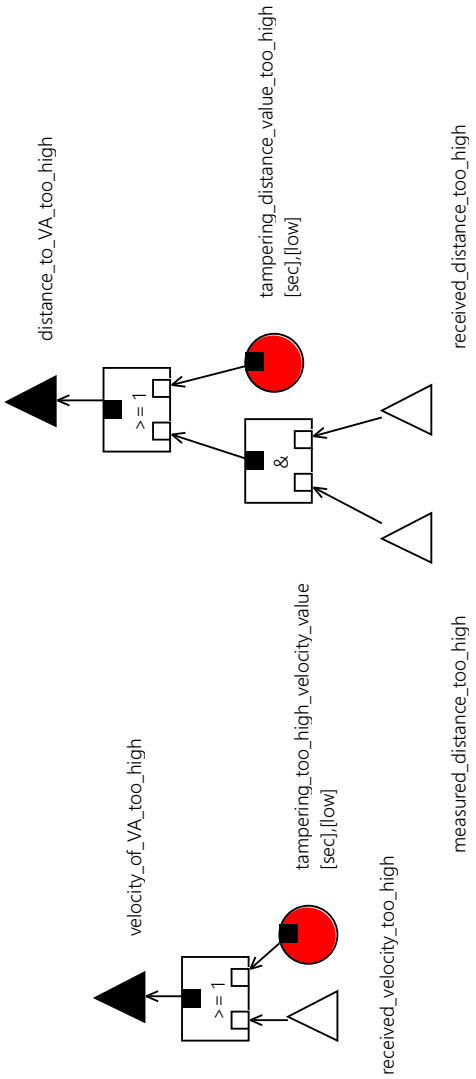


Fig. A.7. SeCFT: ACC.Communication System

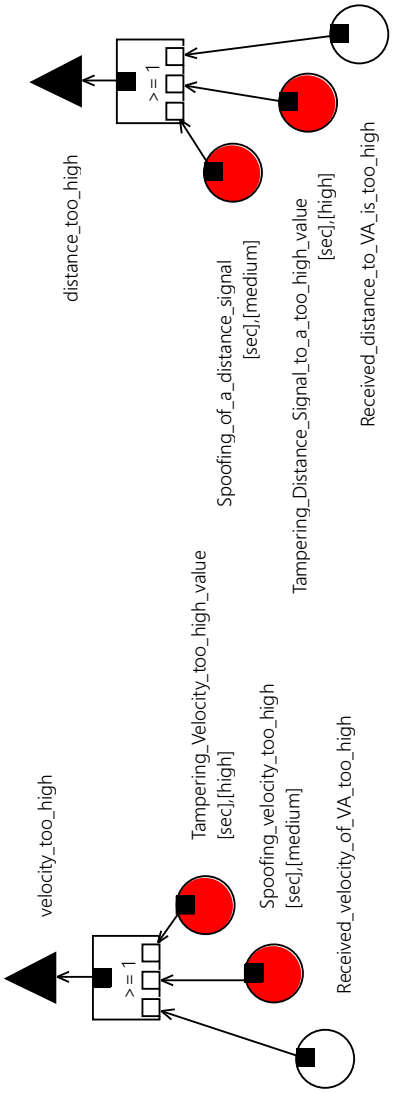


Fig. A.8. SeCFT: ACC.Front Antenna

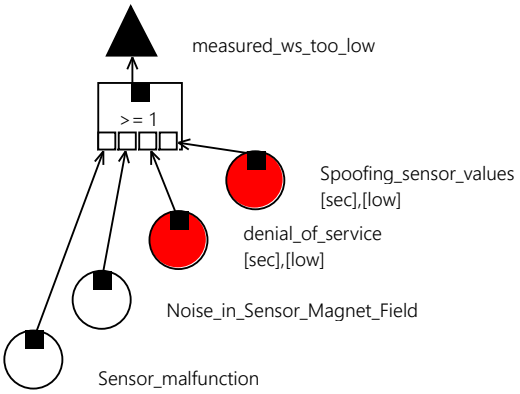


Fig. A.9. SeCFT: ACC.Wheelspeed Sensor

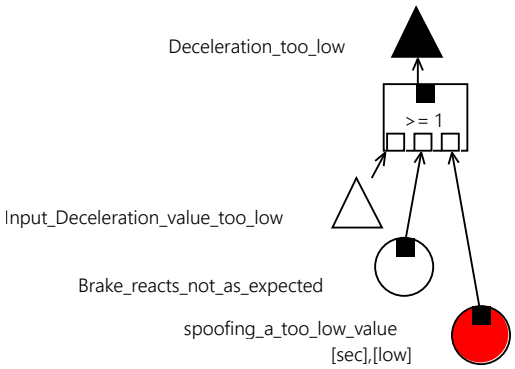


Fig. A.10. SeCFT: ACC.Brake Actor

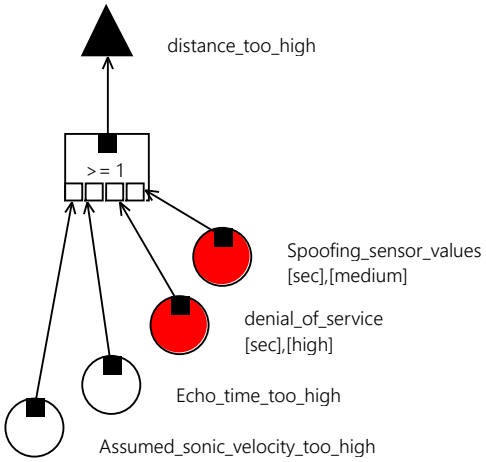


Fig. A.11. SeCFT: ACC.Distance Sensor

Table A.2. MCSs of the SeCFT of the Adaptive Cruise Control system

ID	contained Basic Events
1	Control Logic Unit.tampering own velocity too low
	Control Logic Unit.tampering distance value too high
2	Control Logic Unit.tampering own velocity too low
	Control Logic Unit.tampering velocity VA value too high
3	Brake Interface.tampering brake disabled/deceleration value too low
4	Control Logic Unit.tampering output brake disabled
5	Front Antenna.Tampering Distance Signal to a too high value
	Front Distance Sensor.Spoofing sensor values
6	Front Antenna.Tampering Distance Signal to a too high value
	Front Distance Sensor.denial of service
7	Front Antenna.Tampering Distance Signal to a too high value
	Front Distance Sensor.Assumed sonic velocity too high
8	Front Antenna.Tampering Distance Signal to a too high value
	Front Distance Sensor.Echo time too high
9	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
10	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	FL Wheelspeed Sensor.denial of service
11	FR Wheelspeed Sensor.denial of service
	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
12	FR Wheelspeed Sensor.denial of service
	FL Wheelspeed Sensor.denial of service
13	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	FL Wheelspeed Sensor.Spoofing sensor values
14	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	FL Wheelspeed Sensor.Sensor malfunction
15	FR Wheelspeed Sensor.denial of service
	FL Wheelspeed Sensor.Spoofing sensor values
16	FR Wheelspeed Sensor.denial of service
	FL Wheelspeed Sensor.Sensor malfunction
17	FR Wheelspeed Sensor.Spoofing sensor values
	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
18	FR Wheelspeed Sensor.Spoofing sensor values
	FL Wheelspeed Sensor.denial of service
19	FR Wheelspeed Sensor.Sensor malfunction
	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
20	FR Wheelspeed Sensor.Sensor malfunction
	FL Wheelspeed Sensor.denial of service
21	FR Wheelspeed Sensor.Spoofing sensor values
	FL Wheelspeed Sensor.Spoofing sensor values
22	FR Wheelspeed Sensor.Spoofing sensor values
	FL Wheelspeed Sensor.Sensor malfunction
23	FR Wheelspeed Sensor.Sensor malfunction
	FL Wheelspeed Sensor.Spoofing sensor values
24	FR Wheelspeed Sensor.Sensor malfunction
	FL Wheelspeed Sensor.Sensor malfunction
25	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
26	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.denial of service
27	FR Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
28	FR Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.denial of service
29	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Spoofing sensor values
30	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Sensor malfunction
31	FR Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Spoofing sensor values
32	FR Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Sensor malfunction
33	FR Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
34	FR Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.denial of service
35	FR Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
36	FR Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.denial of service
37	FR Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Spoofing sensor values
38	FR Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Sensor malfunction
39	FR Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Spoofing sensor values
40	FR Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Sensor malfunction
41	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field

ID	contained Basic Events
42	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.denial of service
43	FR Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
44	FR Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.denial of service
45	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Spoofing sensor values
46	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Sensor malfunction
47	FR Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Spoofing sensor values
48	FR Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Sensor malfunction
49	FR Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
50	FR Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.denial of service
51	FR Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
52	FR Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.denial of service
53	FR Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Spoofing sensor values
54	FR Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Sensor malfunction
55	FR Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Spoofing sensor values
56	FR Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Sensor malfunction
57	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
58	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.denial of service
59	FL Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
60	FL Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.denial of service
61	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Spoofing sensor values
62	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RL Wheelspeed Sensor.Sensor malfunction
63	FL Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Spoofing sensor values
64	FL Wheelspeed Sensor.denial of service
	RL Wheelspeed Sensor.Sensor malfunction
65	FL Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
66	FL Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.denial of service
67	FL Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
68	FL Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.denial of service
69	FL Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Spoofing sensor values
70	FL Wheelspeed Sensor.Spoofing sensor values
	RL Wheelspeed Sensor.Sensor malfunction
71	FL Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Spoofing sensor values
72	FL Wheelspeed Sensor.Sensor malfunction
	RL Wheelspeed Sensor.Sensor malfunction
73	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
74	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.denial of service
75	FL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
76	FL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.denial of service
77	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Spoofing sensor values
78	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Sensor malfunction
79	FL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Spoofing sensor values
80	FL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Sensor malfunction
81	FL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field

ID	contained Basic Events
82	FL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.denial of service
83	FL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
84	FL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.denial of service
85	FL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Spoofing sensor values
86	FL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Sensor malfunction
87	FL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Spoofing sensor values
88	FL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Sensor malfunction
89	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
90	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.denial of service
91	RL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
92	RL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.denial of service
93	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Spoofing sensor values
94	RL Wheelspeed Sensor.Noise in Sensor Magnet Field
	RR Wheelspeed Sensor.Sensor malfunction
95	RL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Spoofing sensor values
96	RL Wheelspeed Sensor.denial of service
	RR Wheelspeed Sensor.Sensor malfunction
97	RL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
98	RL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.denial of service
99	RL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Noise in Sensor Magnet Field
100	RL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.denial of service
101	RL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Spoofing sensor values
102	RL Wheelspeed Sensor.Spoofing sensor values
	RR Wheelspeed Sensor.Sensor malfunction
103	RL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Spoofing sensor values
104	RL Wheelspeed Sensor.Sensor malfunction
	RR Wheelspeed Sensor.Sensor malfunction
105	Brake Actor.spoofing a too low value
106	Brake Actor.Brake reacts not as expected
107	ACC.tampering deceleration value too low
108	Brake Interface.tampering distance VA too high
109	Brake Interface.tampering own velocity too low
110	Brake Interface.tampering velocity VA too high
111	Speedometer.tampering velocity too low value
112	Communication System.tampering distance value too high
113	Communication System.tampering too high velocity value
114	Front Antenna.Tampering Velocity too high value
115	Front Antenna.Spoofing velocity too high
116	Front Antenna.Received velocity of VA too high
117	Front Antenna.Spoofing of a distance signal
	Front Distance Sensor.Spoofing sensor values
118	Front Antenna.Spoofing of a distance signal
	Front Distance Sensor.denial of service
119	Front Antenna.Received distance to VA is too high
	Front Distance Sensor.Spoofing sensor values
120	Front Antenna.Received distance to VA is too high
	Front Distance Sensor.denial of service
121	Front Antenna.Spoofing of a distance signal
	Front Distance Sensor.Assumed sonic velocity too high
122	Front Antenna.Spoofing of a distance signal
	Front Distance Sensor.Echo time too high
123	Front Antenna.Received distance to VA is too high
	Front Distance Sensor.Assumed sonic velocity too high
124	Front Antenna.Received distance to VA is too high
	Front Distance Sensor.Echo time too high

Table A.3. Most critical MCSs according to the safety probability of the ACC

MCS ID	Type	Rating	Basic Events
120	mixed	(0.017, high)	Front Antenna.Received distance to VA is too high
			Front Distance Sensor.denial of service
119	mixed	(0.017, medium)	Front Antenna.Received distance to VA is too high
			Front Distance Sensor.Spoofing sensor values
116	safety	(0.015, –)	Front Antenna.Received velocity of VA too high
8	mixed	(0.001, high)	Front Antenna.Tampering Distance Signal to a too high value
			Front Distance Sensor.Echo time too high
122	mixed	(0.001, medium)	Front Antenna.Spoofing of a distance signal
			Front Distance Sensor.Echo time too high
7	mixed	(2E-05, high)	Front Antenna.Tampering Distance Signal to a too high value
			Front Distance Sensor.Assumed sonic velocity too high
121	mixed	(2E-05, medium)	Front Antenna.Spoofing of a distance signal
			Front Distance Sensor.Assumed sonic velocity too high
124	safety	(1.7E-05, –)	Front Antenna.Received distance to VA is too high
			Front Distance Sensor.Echo time too high
106	safety	(1E-05, –)	Brake Actor.Brake reacts not as expected
48	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service
			RR Wheelspeed Sensor.Sensor malfunction
81	mixed	(1E-05, low)	FL Wheelspeed Sensor.Spoofing sensor values
			RR Wheelspeed Sensor.Noise in Sensor Magnet Field
45	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
			RR Wheelspeed Sensor.Spoofing sensor values
80	mixed	(1E-05, low)	FL Wheelspeed Sensor.denial of service
			RR Wheelspeed Sensor.Sensor malfunction
49	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values
			RR Wheelspeed Sensor.Noise in Sensor Magnet Field
52	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction
			RR Wheelspeed Sensor.denial of service
84	mixed	(1E-05, low)	FL Wheelspeed Sensor.Sensor malfunction
			RR Wheelspeed Sensor.denial of service
87	mixed	(1E-05, low)	FL Wheelspeed Sensor.Sensor malfunction
			RR Wheelspeed Sensor.Spoofing sensor values
39	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction
			RL Wheelspeed Sensor.Spoofing sensor values
38	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values
			RL Wheelspeed Sensor.Sensor malfunction
43	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service
			RR Wheelspeed Sensor.Noise in Sensor Magnet Field
42	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
			RR Wheelspeed Sensor.denial of service
86	mixed	(1E-05, low)	FL Wheelspeed Sensor.Spoofing sensor values
			RR Wheelspeed Sensor.Sensor malfunction
64	mixed	(1E-05, low)	FL Wheelspeed Sensor.denial of service
			RL Wheelspeed Sensor.Sensor malfunction
71	mixed	(1E-05, low)	FL Wheelspeed Sensor.Sensor malfunction
			RL Wheelspeed Sensor.Spoofing sensor values
61	mixed	(1E-05, low)	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
			RL Wheelspeed Sensor.Spoofing sensor values
68	mixed	(1E-05, low)	FL Wheelspeed Sensor.Sensor malfunction
			RL Wheelspeed Sensor.denial of service
70	mixed	(1E-05, low)	FL Wheelspeed Sensor.Spoofing sensor values
			RL Wheelspeed Sensor.Sensor malfunction
65	mixed	(1E-05, low)	FL Wheelspeed Sensor.Spoofing sensor values
			RL Wheelspeed Sensor.Noise in Sensor Magnet Field
59	mixed	(1E-05, low)	FL Wheelspeed Sensor.denial of service
			RL Wheelspeed Sensor.Noise in Sensor Magnet Field
55	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction
			RR Wheelspeed Sensor.Spoofing sensor values
54	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values
			RR Wheelspeed Sensor.Sensor malfunction
77	mixed	(1E-05, low)	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
			RR Wheelspeed Sensor.Spoofing sensor values
58	mixed	(1E-05, low)	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
			RL Wheelspeed Sensor.denial of service
74	mixed	(1E-05, low)	FL Wheelspeed Sensor.Noise in Sensor Magnet Field
			RR Wheelspeed Sensor.denial of service
75	mixed	(1E-05, low)	FL Wheelspeed Sensor.denial of service
			RR Wheelspeed Sensor.Noise in Sensor Magnet Field
36	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction
			RL Wheelspeed Sensor.denial of service
103	mixed	(1E-05, low)	RL Wheelspeed Sensor.Sensor malfunction
			RR Wheelspeed Sensor.Spoofing sensor values
16	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service
			FL Wheelspeed Sensor.Sensor malfunction
97	mixed	(1E-05, low)	RL Wheelspeed Sensor.Spoofing sensor values
			RR Wheelspeed Sensor.Noise in Sensor Magnet Field
26	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field
			RL Wheelspeed Sensor.denial of service
96	mixed	(1E-05, low)	RL Wheelspeed Sensor.denial of service
			RR Wheelspeed Sensor.Sensor malfunction

MCS ID	Type	Rating	Basic Events
100	mixed	(1E-05, low)	RL Wheelspeed Sensor.Sensor malfunction RR Wheelspeed Sensor.denial of service
102	mixed	(1E-05, low)	RL Wheelspeed Sensor.Spoofing sensor values RR Wheelspeed Sensor.Sensor malfunction
20	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction FL Wheelspeed Sensor.denial of service
22	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values FL Wheelspeed Sensor.Sensor malfunction
17	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values FL Wheelspeed Sensor.Noise in Sensor Magnet Field
27	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service RL Wheelspeed Sensor.Noise in Sensor Magnet Field
33	mixed	(1E-05, low)	FR Wheelspeed Sensor.Spoofing sensor values RL Wheelspeed Sensor.Noise in Sensor Magnet Field
32	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service RL Wheelspeed Sensor.Sensor malfunction
90	mixed	(1E-05, low)	RL Wheelspeed Sensor.Noise in Sensor Magnet Field RR Wheelspeed Sensor.denial of service
23	mixed	(1E-05, low)	FR Wheelspeed Sensor.Sensor malfunction FL Wheelspeed Sensor.Spoofing sensor values
10	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field FL Wheelspeed Sensor.denial of service
11	mixed	(1E-05, low)	FR Wheelspeed Sensor.denial of service FL Wheelspeed Sensor.Noise in Sensor Magnet Field
13	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field FL Wheelspeed Sensor.Spoofing sensor values
29	mixed	(1E-05, low)	FR Wheelspeed Sensor.Noise in Sensor Magnet Field RL Wheelspeed Sensor.Spoofing sensor values
91	mixed	(1E-05, low)	RL Wheelspeed Sensor.denial of service RR Wheelspeed Sensor.Noise in Sensor Magnet Field
93	mixed	(1E-05, low)	RL Wheelspeed Sensor.Noise in Sensor Magnet Field RR Wheelspeed Sensor.Spoofing sensor values

A.3 Evaluation Example: Smart Farming

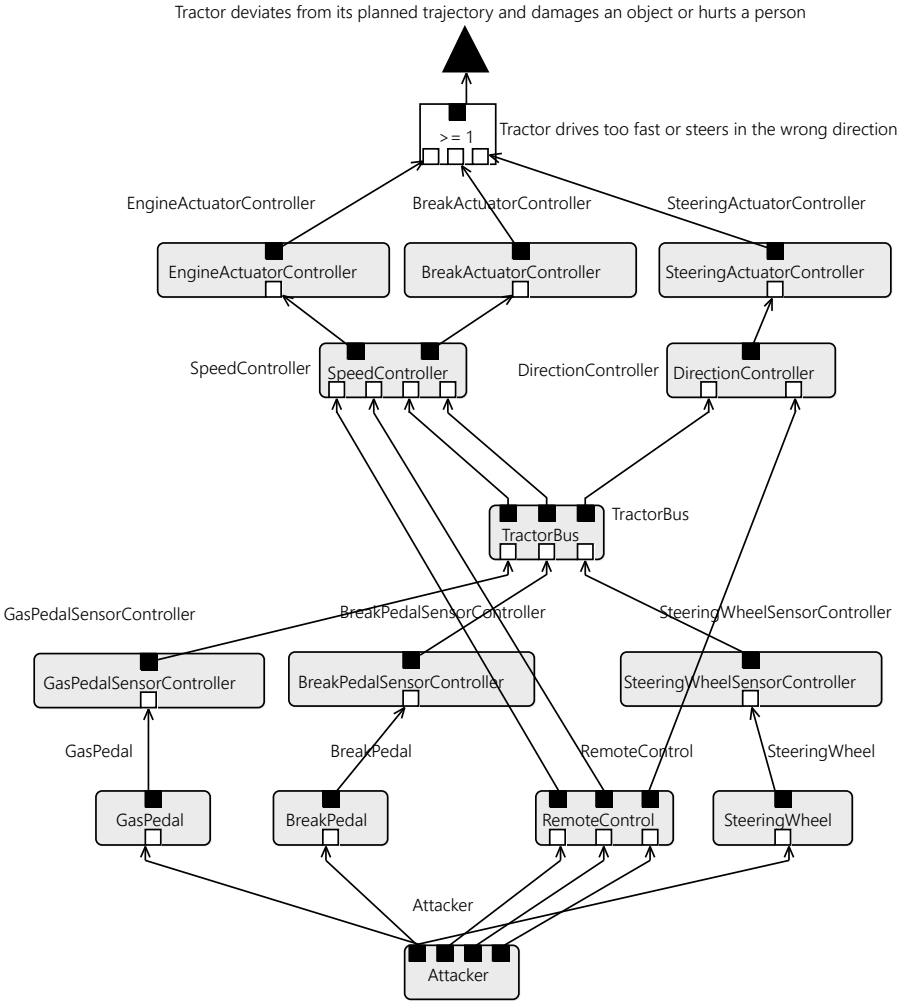


Fig. A.12. SeCFT: SmartFarming.Driving

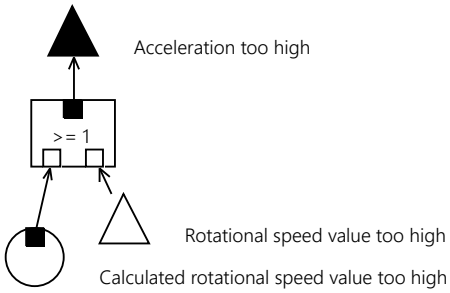


Fig. A.13. SeCFT: SmartFarming.EngineActuatorController

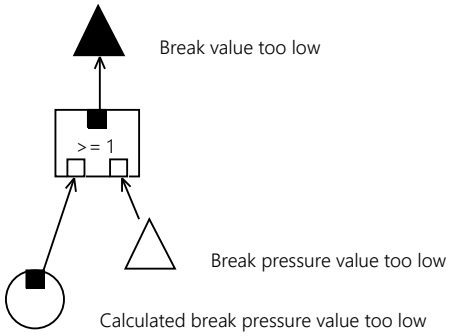


Fig. A.14. SeCFT: SmartFarming.BreakActuatorController

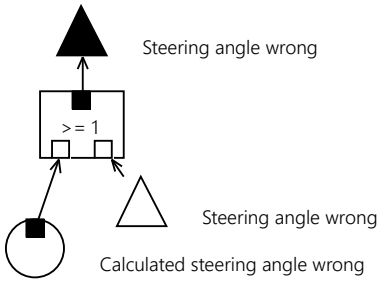


Fig. A.15. SeCFT: SmartFarming.SteeringActuatorController

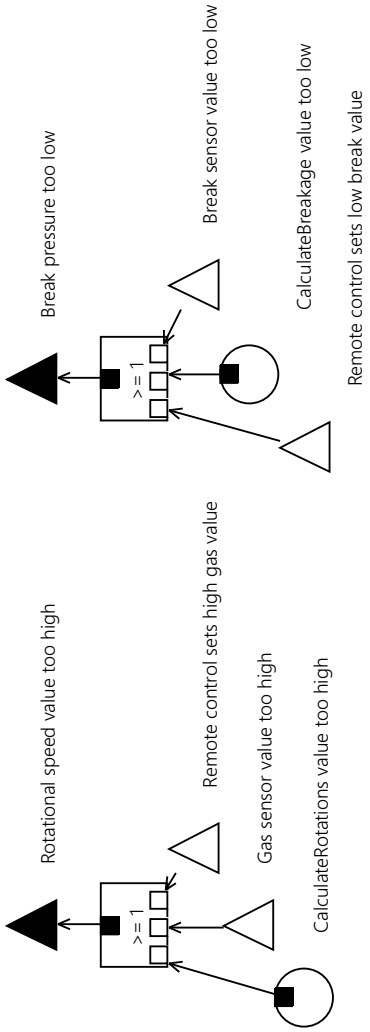


Fig. A.16. SeCFT: SmartFarming.SpeedController

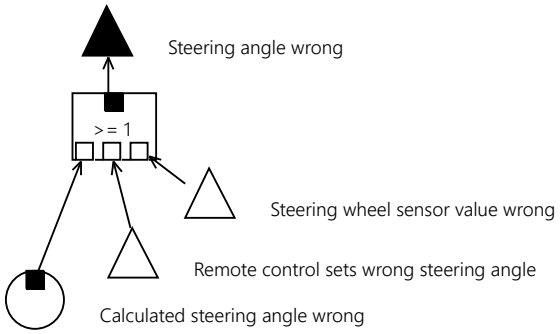


Fig. A.17. SeCFT: SmartFarming.DirectionController

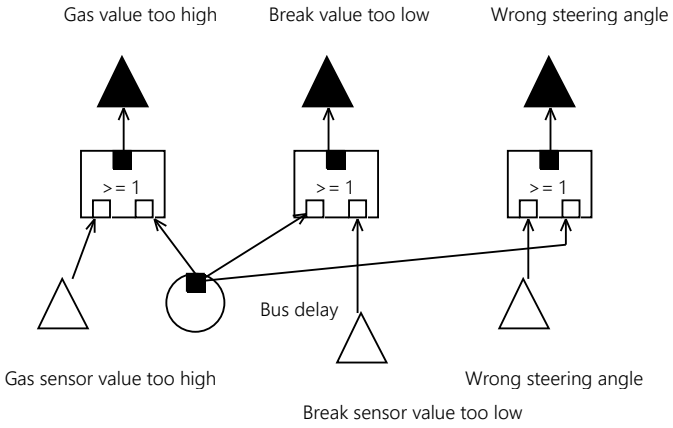


Fig. A.18. SeCFT: SmartFarming.TractorBus

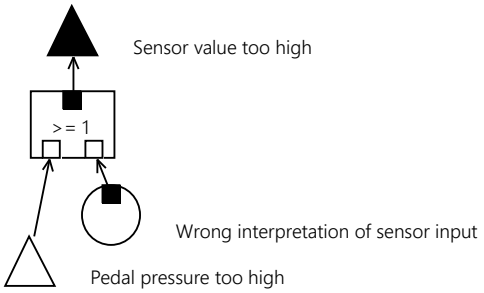


Fig. A.19. SeCFT: SmartFarming.GasPedalSensorController

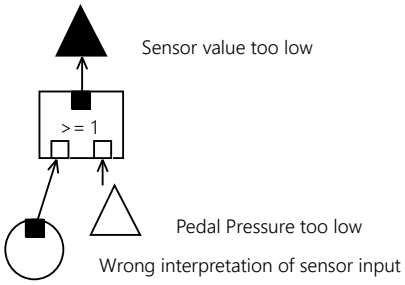


Fig. A.20. SeCFT: SmartFarming.BreakPedalSensorController

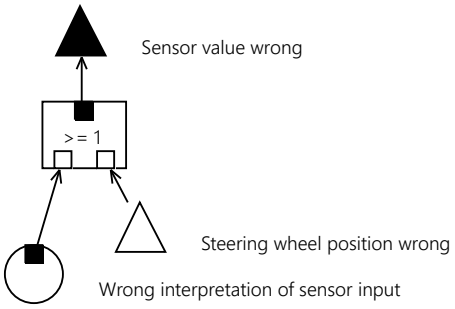


Fig. A.21. SeCFT: SmartFarming.SteeringWheelSensorController

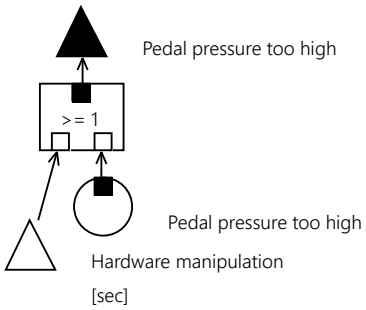


Fig. A.22. SeCFT: SmartFarming.GasPedal

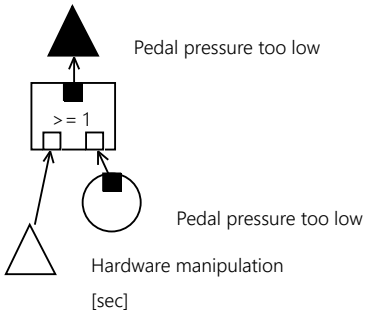


Fig. A.23. SeCFT: SmartFarming.BreakPedal

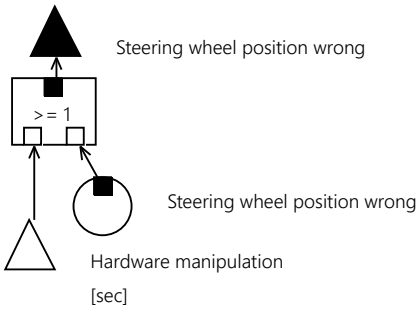


Fig. A.24. SeCFT: SmartFarming.SteeringWheel

Table A.4. MCSs of the SeCFT of the smart farming scenario

MCS ID	contained Basic Events
1	RemoteControl.Remote control active
	RemoteControl.Remote control sets high gas value
2	RemoteControl.Remote control active
	Attacker.Attacker hijacks remote control
	Attacker.Attacker sets high gas value
3	RemoteControl.Remote control active
	Attacker.Attacker hijacks remote control
	Attacker.Attacker sets low break value
4	RemoteControl.Remote control active
	Attacker.Attacker hijacks remote control
	Attacker.Attacker sets wrong steering angle
5	RemoteControl.Remote control active
	RemoteControl.Remote control sets low break value
6	RemoteControl.Remote control active
	RemoteControl.Remote control sets wrong steering angle
7	Attacker.Hardware manipulation
	Attacker.Attacker gains hardware access
8	DirectionController.Calculated steering angle wrong
9	SteeringWheelSensorController.Wrong interpretation of sensor input
10	SteeringWheel.Steering wheel position wrong
11	EngineActuatorController.Calculated rotational speed value too high
12	SpeedController.CalculateRotations value too high
13	TractorBus.Bus delay
14	GasPedalSensorController.Wrong interpretation of sensor input
15	GasPedal.Pedal pressure too high
16	BreakActuatorController.Calculated break pressure value too low
17	SpeedController.CalculateBreakage value too low
18	BreakPedalSensorController.Wrong interpretation of sensor input
19	BreakPedal.Pedal pressure too low
20	SteeringActuatorController.Calculated steering angle wrong

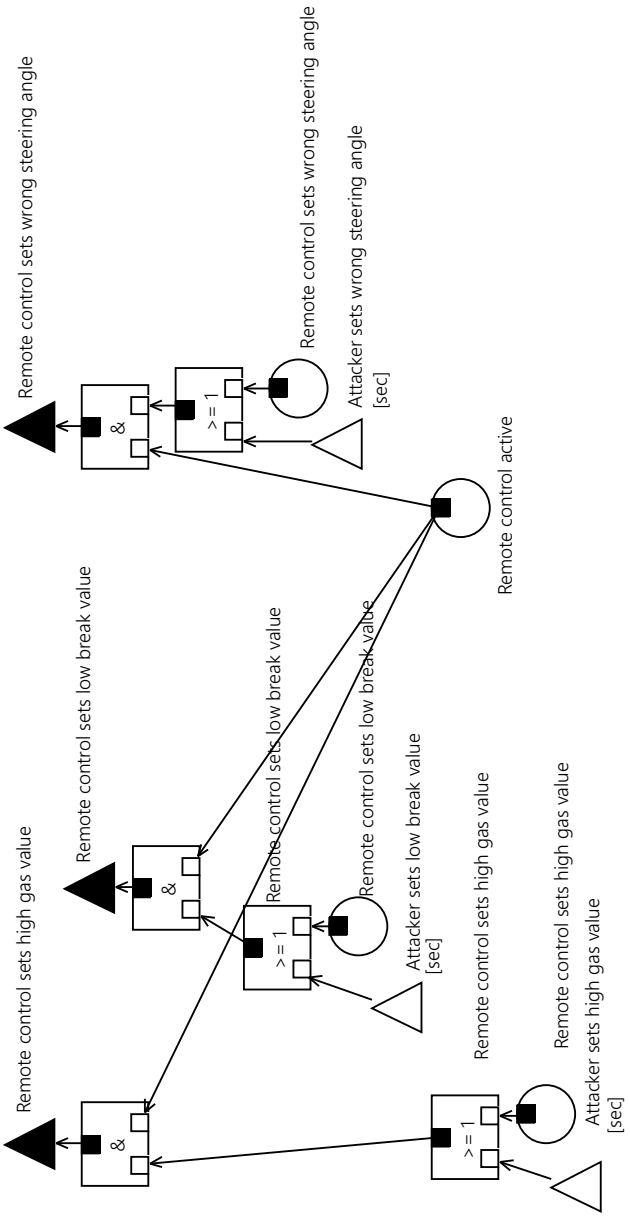


Fig. A.25. SeCFT: SmartFarming.RemoteControl

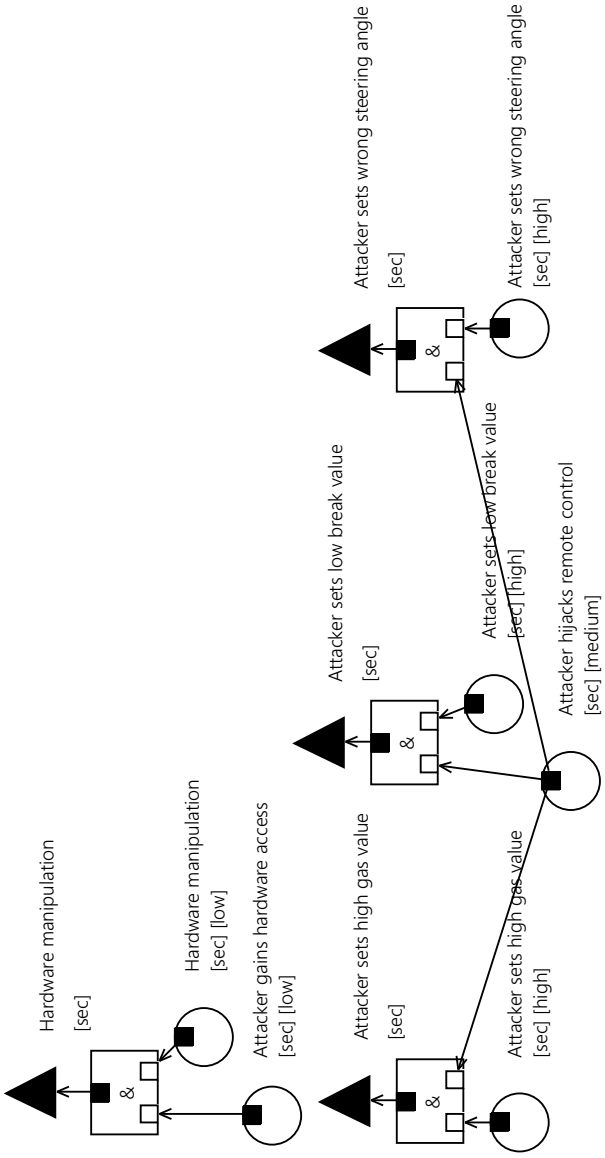


Fig. A.26. SeCFT: SmartFarming.Attacker

A.4 Evaluation Example: Infusion Pump

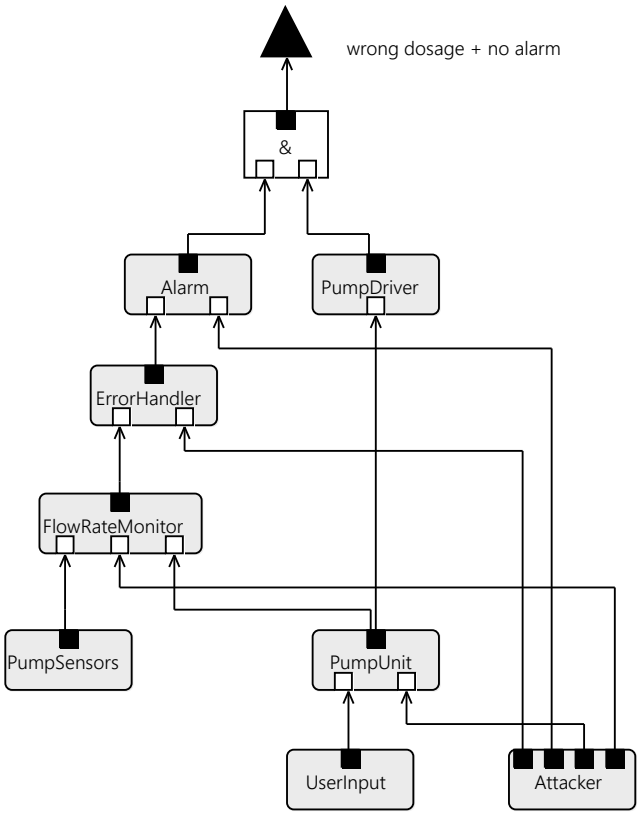


Fig. A.27. SeCFT: InfusionPump

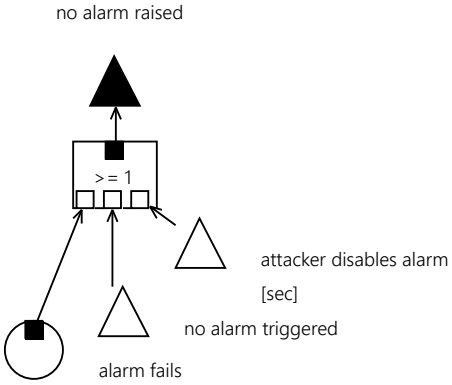


Fig. A.28. SeCFT: InfusionPump.Alarm

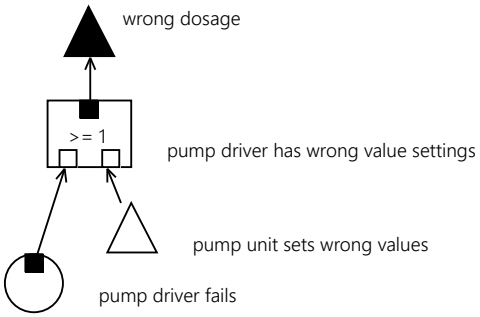


Fig. A.29. SeCFT: InfusionPump.PumpDriver

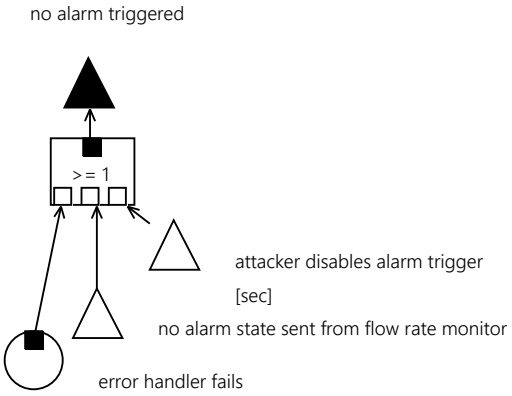


Fig. A.30. SeCFT: InfusionPump.ErrorHandler

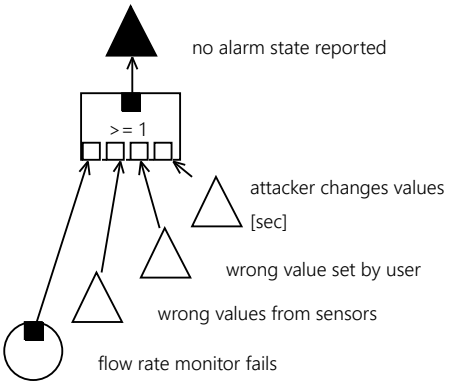


Fig. A.31. SeCFT: InfusionPump.FlowRateMonitor

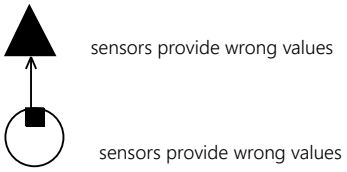


Fig. A.32. SeCFT: InfusionPump.PumpSensors

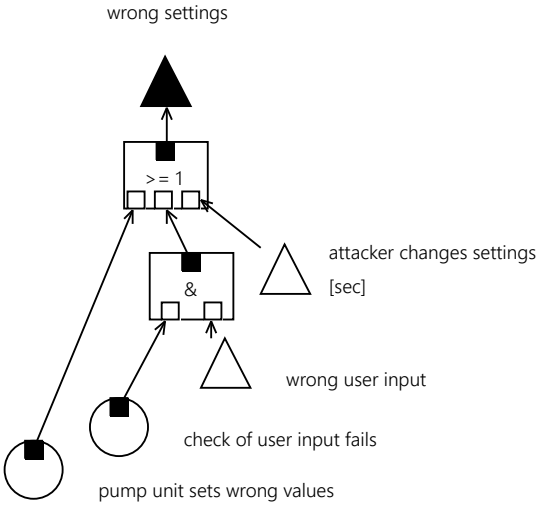


Fig. A.33. SeCFT: InfusionPump.PumpUnit

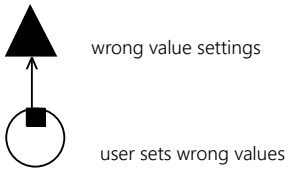


Fig. A.34. SeCFT: InfusionPump.UserInput

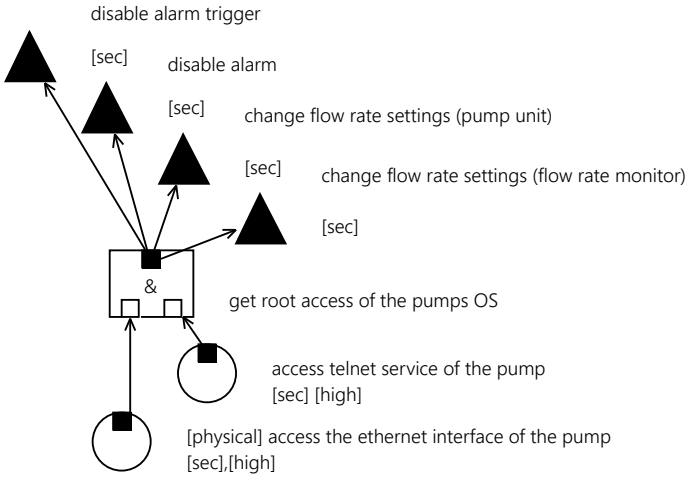


Fig. A.35. SeCFT: InfusionPump.Attacker

B

Abbreviations

ACC	Adaptive Cruise Control ... 89
ADVISE	ADversary VIew Security Evaluation ... 31
AT	Attack Tree ... 109
BDD	Binary Decision Diagram ... 75
BDMP	Boolean logic Driven Markov Process ... 43
BE	Basic Event ... 79
BM	Birnbaum's Importance Measure ... 18
CFT	Component Fault Tree ... 109
CPS	Cyber-Physical System ... 25
DAG	Directed Acyclic Graph ... 46
DFD	Data Flow Diagram ... 31
DFT	Dynamic Fault Tree ... 37
DNF	Disjunctive Normal Form ... 30
DSP	Digital Signal Processor ... 83
DT	Defense Tree ... 37
EFT	Extended Fault Tree ... 36
ERTMS	European Railway Traffic Management System ... 37
FMEA	Failure Mode and Effects Analysis ... 45
FMECA	Failure Mode, Effects, and Criticality Analysis ... 63
FMVEA	Failure Mode, Vulnerabilities and Effects Analysis ... 63
FTA	Fault Tree Analysis ... 87
FT	Fault Tree ... 79
FV	Fussell-Vesely ... 76
GFT	Generalized Fault Tree ... 30
GSPN	Generalized Stochastic Petri Net ... 30
HAZOP	Hazard and Operability Study ... 45
IE	Intermediate Event ... 49
LAN	Local Area Network ... 24
MAC	Media Access Control ... 24
MCS	Minimal Cut Set ... 109

NUREG	US Nuclear Regulatory Commission Regulation 9
PENET	Petri Net Attack Modeling 30
PFT	Parametric Fault Tree 19
PI	Prime Implicant 12
RAW	Risk Achievement Worth 18
RFT	Repairable Fault Tree 37
ROA	Return on Attack 30
ROBDD	Reduced Ordered Binary Decision Diagram 16
ROI	Return on Investment 30
RPN	Risk Priority Number 63
RRW	Risk Reduction Worth 18
SCADA	Supervisory Control and Data Acquisition 25
SeCFT	Security-enhanced Component Fault Tree 109
SEFT	State/Event Fault Tree 19
SHARD	Software Hazard Analysis and Resolution in Design 21
SIL	Safety Integrity Level 99
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege 32
TE	Top Event 59
TVRA	Threat Vulnerability and Risk Assessment 39
UML	Unified Modeling Language 21
WLAN	Wireless Local Area Network 24
WPA2	Wi-Fi Protected Access 2 24

Index

References

[Andrews 00] J. D. Andrews. To Not or Not to Not. In *18th International System Safety Conference*, Fort Worth Texas, Radisson Plaza, Sep 2000.

[Ariss 11] Omar El Ariss, Dianxiang Xu. Modeling Security Attacks with Statecharts. In *Proceedings of the joint ACM SIGSOFT conference – QoSA and ACM SIGSOFT symposium – ISARCS on Quality of software architectures – QoSA and architecting critical systems – ISARCS*, QoSA-ISARCS '11, pp. 123–132. ACM, 2011.

[Armbrust 09] Christopher Armbrust, Tim Braun, Tobias Föhst, Martin Proetzsch, Alexander Renner, Bernd-Helge Schäfer, Karsten Berns. RAVON — The Robust Autonomous Vehicle for Off-road Navigation. In *Proceedings of the IARP International Workshop on Robotics for Risky Interventions and Environmental Surveillance 2009 (RISE 2009)*, Brussels, Belgium, Jan 12–14 2009. IARP.

[Arney 09] David Arney, Raoul Jetley, Yi Zhang, Paul Jones, Oleg Sokolsky, Insup Lee, Arnab Ray. The Generic Patient Controlled Analgesia Pump Model. `http://rtg.cis.upenn.edu/gip.php3`, 2009. (accessed 2015-10-23).

[Avizienis 04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan-Mar 2004.

[Bachfeld 11a] Daniel Bachfeld. Siemens schließt Lücken in Automatisierungssystemen. `http://heise.de/-1259623`, 2011. (accessed 2015-10-23).

[Bachfeld 11b] Daniel Bachfeld. Weitere Siemens-Industriesteuerungen für Angriffe anfällig. `http://heise.de/-1274912`, 2011. (accessed 2015-10-23).

[Bechta Dugan 92] J. Bechta Dugan, Salvatore J. Bavuso, M.A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *Reliability, IEEE Transactions on*, 41(3):363–377, Sep 1992.

[Benenson 08] Zinaida Benenson, Peter M Cholewinski, Felix C Freiling. Vulnerabilities and attacks in wireless sensor networks. *Wireless Sensors Networks Security, Cryptology & Information Security Series (CIS)*, pp. 22–43, 2008.

[Bergert 15] Denise Bergert, Axel Kannenberg. Weitere Sicherheitslücke in Hospira-Infusionspumpen. `http://heise.de/-2682272`, 2015. (accessed 2015-10-23).

[Bistarelli 06] Stefano Bistarelli, Fabio Fioravanti, Pamela Peretti. Defense trees for economic evaluation of security investments. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pp. 8 pp.–, Apr 2006.

[Bloomfield 12] Richard Bloomfield, Robin Bloomfield, Ilir Gashi, Robert Stroud. How Secure Is ERTMS? In Frank Ortmeier, Peter Daniel (Hrsg.), *Computer Safety, Reliability, and Security* , Lecture Notes in Computer Science Bd. 7613. Springer Berlin Heidelberg, 2012.

[Bobbio 03] Andrea Bobbio, Giuliana Franceschinis, Rossano Gaeta, Luigi Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level Petri net semantics. *Software Engineering, IEEE Transactions on*, 29(3):270–287, Mar 2003.

[Bouissou 03] Marc Bouissou, Jean-Louis Bon. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliability Engineering & System Safety*, 82(2):149 – 163, 2003.

[Bowles 01] John B. Bowles, Chi Wan. Software Failure Modes and Effects Analysis For a Small Embedded Control System. In *2001 Proceedings Annual Reliability and Maintainability Symposium*. IEEE, 2001.

[Brooke 03] Phillip J. Brooke, Richard F. Paige. Fault trees for security system design and analysis. *Computers & Security*, 33:256–264, 2003.

[Bryant 86] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *Computers, IEEE Transactions on*, C-35(8):677–691, Aug 1986.

[Buldas 06] Ahto Buldas, Peeter Laud, Jaan Priisalu, Märt Saarepera, Jan Willemson. Rational Choice of Security Measures Via Multi-parameter Attack Trees. In *Critical Information Infrastructures Security*, 2006.

[Byres 04] Eric J. Byres, Matthew Franz, Darrin Miller. The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems. In *IEEE International Infrastructure Survivability Workshop*, 2004.

[Carreras 01] C. Carreras, I. D. Walker. Interval Methods for Fault-Tree Analyses in Robotics. *IEEE Transactions on Reliability*, 50(1):3–11, Mar 2001.

[Casals 12] Silvia Gil Casals, Philippe Owezarski, Gilles Descargues. Risk Assessment for Airworthiness Security. In Frank Ortmeier, Peter Daniel (Hrsg.), *Computer Safety, Reliability, and Security* , Lecture Notes in Computer Science Bd. 7612. Springer Berlin Heidelberg, 2012.

[Checkoway 11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno et al. Comprehensive Experimental

Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*, 2011.

[Codetta-Raiteri 04] D. Codetta-Raiteri, G. Franceschinis, M. Iacono, V. Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *Dependable Systems and Networks, 2004 International Conference on*, pp. 659–668, Jun 2004.

[Codetta-Raiteri 05] D Codetta-Raiteri. *Extended Fault Trees Analysis supported by Stochastic Petri Nets.* Dissertation, Università di Torino, Nov 2005.

[Codetta-Raiteri 13] Daniele Codetta-Raiteri. Generalized Fault-Trees: from reliability to security. In *International Workshop on Quantitative Aspects in Security Assurance (QASA)*, 2013.

[Convery 02] S. Convery, D. Cook, M. Franz. An Attack Tree for the Border Gateway Protocol. Technischer Bericht, Cisco, 2002.

[Dhillon 11] Danny Dhillon. Developer-Driven Threat Modeling: Lessons Learned in the Trenches. *IEEE Security and Privacy*, 9:41–47, 2011.

[DIN 25424-1 81] DIN 25424: Fehlerbaumanalyse Teil 1: Methode und Bildzeichen, Sep 1981.

[DIN 25424-2 90] DIN 25424: Fehlerbaumanalyse Teil 2: Handrechenverfahren zur Auswertung eines Fehlerbaumes, Apr 1990.

[DIN 40041 90] DIN 40041: Zuverlässigkeit - Begriffe, Dec 1990.

[Dobbing 06a] Brian Dobbing, Samantha Lautieri. SafSec: Integration of Safety & Security Certification - Methodology: Guidance Material. Technischer Bericht, Praxis High Integrity Systems, 2006.

[Dobbing 06b] Brian Dobbing, Samantha Lautieri. SafSec: Integration of Safety & Security Certification - Methodology: Standard. Technischer Bericht, Praxis High Integrity Systems, 2006.

[Eames 99] David Peter Eames, Jonathan Moffett. The Integration of Safety and Security Requirements. In *Computer Safety, Reliability and Security*, 1999.

[ED-202 10] ED-202: Airworthiness security process specification, 2010.

[Ericson 99] Clifton A. Ericson. Fault Tree Analysis - A History. In *Proceedings of The 17th International System Safety Conference*, The Boeing Company, Seattle, Washington, 1999.

[ESSaRel 09] Embedded System Safety and Reliability Analyzer (ESSaRel). `http://www.essarel.de`, 2009. (accessed 2015-10-23).

[ETSI TS 102 165-1 11] ETSI TS 102 165-1: TISPAN Methods and Protocols; Method and proforma for Threat, Risk, Vulnerability Analysis, 2011.

[Foster 15] Ian Foster, Andrew Prudhomme, Karl Koscher, Stefan Savage. Fast and Vulnerable: A Story of Telematic Failures. In *Proceedings of the 9th USENIX Conference on Offensive Technologies*, pp. 15–15. USENIX Association, 2015.

[Fovino 09] Igor Nai Fovino, Marcelo Masera, Alessio De Cian. Integrating cyber attacks within fault trees. *Reliability Engineering and System Safety*, 94:1394–1402, 2009.

[Fussell 75] J.B. Fussell. How to Hand-Calculate System Reliability and Safety Characteristics. *Reliability, IEEE Transactions on*, R-24(3):169–174, 1975.

[Förster 09] Marc Förster, Mario Trapp. Fault tree analysis of software-controlled component systems based on second-order probabilities. In *ISSRE 2009 proceedings*, 2009.

[Förster 10] Marc Förster, Reinhard Schwarz, Max Steiner. Integration of Modular Safety and Security Models for the Analysis of the Impact of Security on Safety. Technischer Bericht 078.10/E, Fraunhofer IESE, Technische Universität Kaiserslautern, Kaiserslautern, Germany, 2010.

[Greenberg 15] Andy Greenberg. Hackers Remotely Kill a Jeep on the Highway – With Me in It. `http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/`, 2015. (accessed 2015-10-23).

[Grosspietsch 04] Karl-Erwin Grosspietsch, Tanya A. Silayeva. A Combined Safety/Security Approach for Co-Operative Distributed Systems. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, 2004.

[Guo 10] Zhengsheng Guo, Dirk Zeckzer, Peter Liggesmeyer, Oliver Mäckel. Identification of Security-Safety Requirements for the Outdoor Robot RAVON Using Safety Analysis Techniques. In *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, pp. 508–513, 2010.

[Helmer 02] Guy Helmer, Johnny Wong, Mark Slagell, Vasant Honavar, Les Miller, Robyn Lutz. A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System. *Requirements Engineering Journal*, 7(4):207–220, 2002.

[Hernan 06] Shawn Hernan, Scott Lambert, Tomasz Ostwald, Adam Shostack. Uncover Security Design Flaws Using The STRIDE Approach. *MSDN Magazine*, Nov 2006.

[Holland 15] Martin Holland. ConnectedDrive: Der BMW-Hack im Detail. `http://heise.de/-2540786`, 2015. (accessed 2015-10-23).

[Howard 02] Michael Howard, David E. Leblanc. *Writing Secure Code*. Microsoft Press, Redmond, WA, USA, Ausgabe 2nd, 2002.

[Howard 06] Michael Howard, Steve Lipner. *The Security Development Lifecycle: A Process for Developing Demonstrably More Secure Software*. Microsoft Press, 2006.

[ICS-CERT 11] ICS-CERT. Alert (ICS-ALERT-11-161-01) - Siemens S7-1200 PLC Vulnerabilities. `https://ics-cert.us-cert.gov/alerts/ICS-ALERT-11-161-01`, Jun 2011. (accessed 2015-09-17).

[ICS-CERT 13] ICS-CERT. Alert (ICS-ALERT-13-164-01) - Medical Devices Hard-Coded Passwords. `https://ics-cert.us-cert.gov/alerts/ICS-ALERT-13-164-01`, Jun 2013. (accessed 2015-10-23).

[Idika 10] Nwokedi Idika, Bharat Bhargava. Extending Attack Graph-based Security Metrics and Aggregating Their Application. *IEEE Transactions on Dependable and Secure Computing*, 2010.

[IEC 60300-3-1 05] IEC 60300-3-1: Dependability management - Part 3-1: Application guide; Analysis techniques for dependability; Guide on methodology, May 2005.

[IEC 60812 06] IEC 60812: Analysis techniques for system reliability - Procedure for Failure Mode and Effects Analysis (FMEA), Jan 2006.

[IEC 61025 06] IEC 61025: Fault tree Analysis (FTA), Dec 2006.

[IEC 61508-4 10] IEC 61508-4: Functional safety of electrical / electronic / programmable electronic safety-related systems - Part 4: Definitions and abbreviations, Apr 2010.

[IEC 61508 10] IEC 61508: Functional safety of electrical / electronic / programmable electronic safety-related systems, 2010.

[IEC 61882 01] IEC 61882: Hazard and operability studies (HAZOP studies) - Application guide, May 2001.

[Ingoldsby 03] Terrance R Ingoldsby. Understanding Risks Through Attack Tree Analysis. *Amenaza Technologies Ltd. Copyright*, 2003. `http://amenaza.com/downloads/docs/Methodology.pdf`.

[Ingoldsby 13] Terrance R Ingoldsby. Attack tree-based threat risk analysis. *Amenaza Technologies Ltd. Copyright*, 2013. `https://amenaza.com/downloads/docs/AttackTreeThreatRiskAnalysis.pdf`.

[ISO/IEC 15408 12] ISO/IEC 15408 Common Criteria for Information Technology Security Evaluation, Sep 2012.

[ISO/IEC 27000 09] DIN ISO/IEC 27000 Information technology - Security techniques - Information security management systems - Overview and vocabulary, May 2009.

[ITSG 15] Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz). "`http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl115s1324.pdf`", Jul 2015. (accessed 2015-10-23).

[Jürgenson 08] Aivo Jürgenson, Jan Willemson. Computing Exact Outcomes of Multi-parameter Attack Trees. In *On the Move to Meaningful Internet Systems*, 2008.

[Kaiser 02] Bernhard Kaiser. Integration von Sicherheits- und Zuverlässigkeitsmodellen in den Entwicklungsprozess Eingebetteter Systeme. Technischer Bericht, Hasso-Plattner Institut für Softwaretechnik an der Universität Potsdam, 2002.

[Kaiser 03a] Bernhard Kaiser. A Fault-Tree Semantics to model Software-Controlled Systems. Technischer Bericht, Hasso-Plattner Institute for Software Systems Engineering at the University of Potsdam, 2003.

[Kaiser 03b] Bernhard Kaiser, Peter Liggesmeyer, Oliver Mäckel. A New Component Concept for Fault Trees. In *8th Australian Workshop on Safety Critical Systems and Software*, Canberra, Oct 2003.

[Kaiser 06] Bernhard Kaiser. *State/Event Fault Trees: A Safety and Reliability Analysis Technique for Software-Controlled Systems.* Dissertation, Technische Universität Kaiserslautern, 2006.

[Kamkar 15] Samy Kamkar. Drive It Like You Hacked It: New Attacks and Tools to Wirelessly Steal Cars. In *DEF CON 23*, 2015.

[Kargl 14] Frank Kargl, Rens W. van der Heijden, Hartmut König, Alfonso Valdes, Marc C. Dacier. Insights on the Security and Dependability of Industrial Control Systems. *Security Privacy, IEEE*, 12(6):75–78, Nov 2014.

[Kleinz 14] Torsten Kleinz, Hajo Schulz. 31C3: Wie man ein Chemiewerk hackt. `http://heise.de/-2507259`, 2014. (accessed 2015-10-23).

[Klick 15] Johannes Klick, Stephan Lau, Daniel Marzin, Jan-Ole Malchow, Volker Roth. Internet-facing PLCs - A New Back Orifice. In *Blackhat USA 2015,*, 2015.

[Klutke 03] G.A. Klutke, P.C. Kiessler, M.A. Wortman. A critical look at the bathtub curve. *Reliability, IEEE Transactions on*, 52(1):125–129, Mar 2003.

[Koscher 10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *2010 IEEE Symposium on Security and Privacy*, 2010.

[Langner 11] Ralph Langner. Stuxnet: Dissecting a Cyberwarfare Weapon. *Security Privacy, IEEE*, 9(3):49–51, May 2011.

[Lano 02] Kevin Lano, David Clark, Kelly Androutsopoulos. Safety and Security Analysis of Object-Oriented Models. In *Safecomp*, 2002.

[Lanotte 03] Ruggero Lanotte. *An automaton-theoretic approach to safety and security in real-time systems.* Dissertation, Università degli Studi di Pisa, 2003.

[LeMay 11] E. LeMay, M.D. Ford, K. Keefe, W.H. Sanders, C. Muehrcke. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pp. 191–200, Sep 2011.

[Marsan 95] Marco A. Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, Giuliana Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, Inc., 1995.

[Mauw 06] Sjouke Mauw, Martijn Oostdijk. Foundations of Attack Trees. In *Information Security and Cryptology - ICISC 2005*, 2006.

[McDermott 00] J. P. McDermott. Attack Net Penetration Testing. In *Proceedings of the 2000 Workshop on New Security Paradigms*, NSPW '00, pp. 15–21, New York, NY, USA, 2000. ACM.

[Microsoft 10] Microsoft. Security Development Lifecycle. 2010.

[MITRE 14] MITRE. Common Weakness Enumeration. `http://cwe.mitre.org`, 2014. (accessed 2015-10-23).

[MoD DS 00-56 07] Defence Standard 00-56, Safety Management Requirements for Defence Systems, Jun 2007.

[Nicol 04] David M. Nicol, William H. Sanders, Kishor S. Trivedi. Model-Based Evaluation: From Dependability to Security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48–65, Jan-Mar 2004.

[OWASP 15] Open Web Application Security Project (OWASP). https://www.owasp.org/, 2015. (accessed 2015-10-18).

[Paul 15] Stephane Paul, Laurent Rioux. Over 20 Years of Research in Cybersecurity and Safety Engineering: a short Bibliography. In *6th International Conference on Safety and Security Engineering (SAFE)*, 2015.

[Pauli 15] Darren Pauli. Thousands of 'directly hackable' hospital devices exposed online. http://www.theregister.co.uk/2015/09/29/thousands_of_directly_hackable_hospital_devices_found_exposed/, 2015. (accessed 2015-10-23).

[Pièvre-Cambacédès 10a] Ludovic Pièvre-Cambacédès, Marc Bouissou. Attack and Defense Modeling with BDMP. In Igor Kotenko, Victor Skormin (Hrsg.), *Computer Network Security* , Lecture Notes in Computer Science Bd. 6258, pp. 86–101. Springer Berlin Heidelberg, 2010.

[Pièvre-Cambacédès 10b] Ludovic Pièvre-Cambacédès, Marc Bouissou. Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes). In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 2852–2861, Oct 2010.

[Proetzsch 10] Martin Proetzsch, Tobias Luksch, Karsten Berns. Development of Complex Robotic Systems Using the Behavior-Based Control Architecture iB2C. *Robotics and Autonomous Systems*, 58(1):46–67, Jan 2010.

[Pudar 09] Srdjan Pudar, G. Manimaran, Chen-Ching Liu. PENET: A practical method and tool for integrated modeling of security attacks and countermeasures. *Computers & Security*, 28(8):754 – 771, 2009.

[Pumfrey 99] David John Pumfrey. *The Principled Design of Computer System Safety Analyses*. Dissertation, University of York, 1999.

[Rauzy 01] Antoine Rauzy. Mathematical Foundations of Minimal Cutsets. *IEEE Transactions on Reliability*, 50(4):389–396, Dec 2001.

[Reichenbach 12] F. Reichenbach, J. Endresen, M.M.R. Chowdhury, J. Rossebo. A Pragmatic Approach on Combined Safety and Security Risk Analysis. In *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, pp. 239–244, Nov 2012.

[Robertson 14] Jordan Robertson, Michael A. Riley. Mysterious '08 Turkey Pipeline Blast Opened New Cyberwar. http://www.bloomberg.com/news/articles/2014-12-10/mysterious-08-turkey-pipeline-blast-opened-new-cyberwar, 2014. (accessed 2015-10-23).

[Roth 13] Michael Roth, Max Steiner, Peter Liggesmeyer. Ein Ansatz zur integrierten Sicherheitsanalyse komplexer Systeme. In Wolfgang A. Halang (Hrsg.), *Kommunikation unter Echtzeitbedingungen*, Informatik aktuell. Springer Berlin Heidelberg, 2013.

[Rouf 10] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, travis Taylor, Sangho Oh, Wanyuan Xu, Marco Gruteser, Wade Trappe, Ivan Seskar. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring Case Study. Technischer Bericht, Dept. of CSE, Univ. of South

Carolina, Columbia, SC USA and WINLAB, Rutgers Univ. Pscataway, NJ USA, 2010.

[Rushdi 04] Ali M. Rushdi, Omar M. Ba-Rukab. A Doubly-Stochastic Fault-Tree Assessment of the Probabilities of Security Breaches in Computer Systems. In *Saudi Science Conference*, 2004.

[Rushdi 05] Ali M. Rushdi, Omar M. Ba-Rukab. Fault.-tree modelling of computer system security. *International Journal of Computer Mathematics*, 82:805–819, 2005.

[Saglietti 06] Francesca Saglietti. Interaktion zwischen funktionaler Sicherheit und Datensicherheit. In *Sicherheit - Schutz und Zuverlässigkeit (Sicherheit 2006)*, Lecture Notes in Informatics, pp. 373–383, Magedeburg, 2006. Gesellschaft für Informatik.

[Scherschel 14] Fabian Scherschel. BSI-Sicherheitsbericht: Erfolgreiche Cyber-Attacke auf deutsches Stahlwerk. `http://heise.de/-2498990`, 2014. (accessed 2015-10-23).

[Scherschel 15] Fabian Scherschel. Root-Shell im Krankenhaus: Hospira-Infusionspumpe mit Telnet-Lücke. `http://heise.de/-2633529`, 2015. (accessed 2015-10-23).

[Schirrmacher 15] Dennis Schirrmacher. Scada-Sicherheit: Siemens-PLC wird zum Einbruchswerkzeug. `http://heise.de/-2774812`, 2015. (accessed 2015-10-23).

[Schmidt 11] Jürgen Schmidt. Siemens bezieht Stellung zu den "SCADA-Affen". `http://heise.de/-1318547`, 2011. (accessed 2015-10-23).

[Schmittner 14] Christoph Schmittner, Thomas Gruber, Peter Puschner, Erwin Schoitsch. Security Application of Failure Mode and Effect Analysis (FMEA). In Andrea Bondavalli, Felicita Di Giandomenico (Hrsg.), *Computer Safety, Reliability, and Security* , Lecture Notes in Computer Science Bd. 8666, pp. 310–325. Springer International Publishing, 2014.

[Schneier 14] Bruce Schneier. The Internet of Things Is Wildly Insecure - And Often Unpatchable. `https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.html`, 2014. (accessed 2015-10-23).

[Schneier 99] Bruce Schneier. Attack Trees. *Dr. Dobb's Journal*, Dec 1999.

[Schäfer 11] Bernd Helge Schäfer. *Robot Control Design Schemata and their Application in Off-road Robotics.* Dissertation, TU Kaiserslautern, 2011.

[Sheyner 02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, Jeannette M. Wing. Automated Generation and Analysis of Attack Graphs. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pp. 273–284, 2002.

[Sommerville 03] Ian Sommerville. An Integrated Approach to Dependability Requirements Engineering. In *Proceedings of the 11th Safety-Critical Systems Symposium*, 2003.

[Spang 10] Silke Spang, Rasmus Adler, Tanvir Hussain, Robert Eschbach. Scrutinizing the impact of security on safety on an Communicating vehicle platoon. Technischer Bericht, Fraunhofer IESE, 2010.

[Stahl 13] Louis-F. Stahl, Ronald Eikenberg. Kritisches Sicherheitsupdate für 200.000 Industriesteuerungen. `http://heise.de/-1934787`, 2013. (accessed 2015-10-23).

[Steiner 12] Max Steiner, Patric Keller, Peter Liggesmeyer. Modeling the Effects of Software on Safety and Reliability in Complex Embedded Systems. In Frank Ortmeier, Peter Daniel (Hrsg.), *Computer Safety, Reliability, and Security*, Bd. 7613, pp. 454–465. Springer Berlin Heidelberg, 2012. Proceedings of the 3rd International Workshop in Digital Engineering.

[Steiner 13a] Max Steiner, Peter Liggesmeyer. Combination of Safety and Security Analysis - Finding Security Problems that Threaten the Safety of a System. In Matthieu ROY (Hrsg.), *Proceedings of Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[Steiner 13b] Max Steiner, Adrien Mouaffo, Davide Taibi. Empirische Evaluierung der Analysetechniken dokumentiert. Technischer Bericht, University of Kaiserslautern, 2013. Meilensteinbericht zum Arbeitspaket 6.3.2, Meilenstein 30.

[Steiner 15] Max Steiner, Peter Liggesmeyer. Qualitative and Quantitative Analysis of CFTs Taking Security Causes into Account. In Floor Koornneef, Coen van Gulijk (Hrsg.), *Computer Safety, Reliability, and Security* , Lecture Notes in Computer Science Bd. 9338, pp. 109–120. Springer International Publishing, 2015.

[Teso 13] Hugo Teso. Aircraft Hacking: Practical Aero Series. `http://conference.hitb.org/hitbsecconf2013ams/hugo-teso/`, 2013. (accessed 2015-10-23).

[van der Borst 01] M van der Borst, H Schoonakker. An overview of {PSA} importance measures. *Reliability Engineering & System Safety*, 72(3):241–245, 2001.

[van Lamsweerde 04] Axel van Lamsweerde. Elaborating Security Requirements by Construction of Intentional Anti-Models. In *International Conference on Software Engineering*, 2004.

[Verdult 15] Roel Verdult, Flavio D. Garcia, Barıs Ege. Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer. In *24nd USENIX Security Symposium*, 2015.

[Verendel 09] Vilhelm Verendel. Quantified security is a weak hypothesis: a critical survey of results and assumptions. In *NSPW '09: Proceedings of the 2009 workshop on New security paradigms workshop*, pp. 37–50, New York, NY, USA, 2009. ACM.

[Vesely 02] William Vesely. *Fault Tree Handbook with Aerospace Applications*. NASA, 2002.

[Vesely 81] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasl. *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission, 1981.

[Vigo 12] Roberto Vigo. The Cyber-Physical Attacker. In Frank Ortmeier, Peter Daniel (Hrsg.), *Computer Safety, Reliability, and Security*, Bd. 7613, pp. 347–356. Springer Berlin Heidelberg, 2012.

[Wilhoit 13] Kyle Wilhoit. Who's Really Attacking Your ICS Equipment? *Trend Micro Incorporated*, 2013.

[Zetter 12] Kim Zetter. Hackers Breached Railway Network, Disrupted Service. `http://www.wired.com/2012/01/railyway-hack/`, 2012. (accessed 2015-10-23).

[Zetter 15] Kim Zetter. Hacker Can Send Fatal Dose to Hospital Drug Pumps. `http://www.wired.com/2015/06/hackers-can-send-fatal-doses-hospital-drug-pumps/`, 2015. (accessed 2015-10-23).

Max Steiner

Curriculum Vitae

Name Max Philipp Steiner

Higher Education

2003 – 2009 Study of Computer Science at the University of Kaiserslautern

Diploma Thesis, *Integrating Reinforcement Learning into Behaviour-Based Control of Bipedal Robots*.

04/2009 **Diplom Informatik (Dipl.-Inf.)**, *University of Kaiserslautern*, Kaiserslautern.

08/2009 – 01/2016 PhD Student at University of Kaiserslautern at the Chair of Software Engineering: Dependability

Work Experience

08/2009 – 01/2016 **Researcher**, *University of Kaiserslautern*, Chair of Software Engineering: Dependability, Area of research: safety and security analysis of embedded systems.