White Paper

# Safety & Security Engineering

A Model-Based Approach

fortiss

Authors:

**Dr. Vivek Nigam**
fortiss GmbH
Guerickestraße 25
80805 München

Supported by
**Dr. Harald Rueß**
**Prof. Dr. Alexander Pretschner**

kontakt@fortiss.org

fortiss
fortiss GmbH
White Paper, 2018

# Model-Based Safety and Security Engineering

Vivek Nigam[1]*, Alexander Pretschner[1,2], and Harald Ruess[1]

**Abstract**
By exploiting the increasing surface attack of systems, cyber-attacks can cause catastrophic events, such as, remotely disable safety mechanisms. This means that in order to avoid hazards, safety and security need to be integrated, exchanging information, such as, key hazards/threats, risk evaluations, mechanisms used. This white paper describes some steps towards this integration by using models. We start by identifying some key technical challenges. Then we demonstrate how models, such as Goal Structured Notation (GSN) for safety and Attack Defense Trees (ADT) for security, can address these challenges. In particular, (1) we demonstrate how to extract in an automated fashion security relevant information from safety assessments by translating GSNs into ADTs; (2) We show how security results can impact the confidence of safety assessments; (3) We propose a collaborative development process where safety and security assessments are built by incrementally taking into account safety and security analysis; (4) We describe how to carry out trade-off analysis in an automated fashion, such as identifying when safety and security arguments contradict each other and how to solve such contradictions. We conclude pointing out that these are the first steps towards a wide range of techniques to support Safety and Security Engineering. As a white paper, we avoid being too technical, preferring to illustrate features by using examples and thus being more accessible.

[1] *fortiss GmbH, Munich, Germany*
[2] *Technical Univeristy of Munich, Munich, Germany*
***Corresponding author**: nigam@fortiss.org

## Contents

## Introduction

The past years have witnessed a technological revolution interconnecting systems and people. This revolution is leading to new exciting services and business models. Managers can now remotely adapt manufacturing to customers needs in the so called Industry 4.0 paradigm. In the very near future, vehicles will operate with high levels of autonomy making decisions based on information exchanged with other vehicles and the available infrastructure. Similarly, autonomous UAVs will be used to transport cargo and people.

This technological revolution, however, leads to new challenges for safety and security. **The greater connectivity of systems increases their attack surface** [20], that is, the ways an intruder can carry out an attack. Whereas in conventional systems, attacks, such as car theft, require some proximity to their target, with interconnected systems, cyber-attacks, such as stealing sensitive data or hacking into safety-critical components, can be carried out remotely.

This increased attack surface has important consequences for system safety. Security can no longer be taken lightly when assessing the safety of a system.[1] **Indeed, cyber-attacks can lead to catastrophic events** [14, 15, 13, 16]. For example, cyber-attacks exploiting a connection to an autonomous vehicle may remotely disable safety features, such as airbags, thus placing passengers in danger [15]. **Therefore, both safety and security have to be taken into account in order to conclude the safety of a system.**

On the other hand, this increased surface attack also has important consequences to security itself. While traditional security concerns, such as, handling physical attacks, *e.g.*, car theft, remain important concerns, security engineers shall consider a wider range of cyber-attacks due to the increased attack surface and, in particular, cyber-attacks that lead to catastrophic events. This means that security engineers shall understand information normally contained in safety assessments, such as, which are the catastrophic events and how they can be caused/triggered. That is, **security analysis shall take into account safety concerns.**

Finally, **this technological revolution will also have an impact on system certification processes.** It is unreasonable to allow the delivery of products to consumers with-

[1]By, for example, enforcing that only authorized persons are near to sensitive parts of the system.

out considering attacks that may lead to catastrophic events. Companies will soon need to provide detailed security analysis arguing that the risk of such threats is acceptable. While current certification agencies do not demand such analysis, there have been initiatives towards this direction, *e.g.*, RTCA DO-326A [9], SAE J3061 [6], ISO 21434 [4], ISO 15408 [2]. **This change on certification processes will have an important impact to the processes and business models of companies.** It is, therefore, important to develop techniques that integrate safety and security that can facilitate the development of such safety and security arguments.

**Problem Statement** Safety and security are carried out with very different mind-sets. While safety is concerned in controlling catastrophic events caused by the malfunctioning of the system, security is concerned in mitigating attacks from malicious agents to the system.

The difference between safety and security is reflected in the types of techniques used for establishing safety and security assessments. Safety assessments are constructed by using analysis techniques such as Fault Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), System Theoretic Process Analysis (STPA), Goal Structured Notation [12], specific safety designs and mechanisms, *e.g.*, Voters, Watchdogs, etc. Security, on the other hand, uses different assessment techniques, such as Attack Trees [40], Attack Defense Trees [30], STRIDE, and security designs and mechanisms, *e.g.*, access control policies, integrity checks, etc.

It is not clear how these different techniques can be used in conjunction to build a general safety and security argument. Questions such as the following have to be answered: **What is the common language for safety and security? How can security engineers use safety assessments to build security arguments? What is the impact of security assessments to safety cases? What are the trade-offs to be considered? Which methods can be implemented within current practices?**

This difference in mentality is also reflected in the development process leading many times to **Poor Process Integration of Safety and Security**. Safety and security concerns are only integrated at very late stages of development when fewer solutions for conflicts are possible/available. For example, in secure by design processes, security engineers participate from the beginning proposing security design requirements. However, they do not take into account the impacts of such requirements on safety arguments, for example, adding mechanisms with unreasonable delays. **The lack of Integration of Safety and Security leads to increased development costs, delays and less safe and secure systems.**

**Benefits of Safety and Security Integration** Besides improving the safety and the security of systems, the integration of safety and security can lead to a number of benefits. We highlight some possible benefits:

- **Early-On Integration of Safety and Security:** Safety and security assessments can be carried out while the requirements of system features are established. Safety assessments provide concrete hazards which should be treated by security assessments, thus **helping security engineers to set priorities**. For example, a safety hazard shall be given a higher priority compared to other security attacks which do not cause catastrophic events;
- **Verification and Validation:** While safety has many well-established methods for verification, security verification relies mostly on penetration testing techniques, which are system dependent and therefore, resource intensive. The integration of Safety and Security can facilitate security verification. **Much of knowledge gathering can be retrieved from safety assessment, thus saving resources**. For example, FTAs describe the events leading to some hazardous event, while FMEAs describe single-points of failures. This information can be used by security engineers to plan penetration tests, *e.g.*, exploit single-point of failures described in FMEAs, thus leading to increased synergies and less development efforts;
- **Safety and Security Mechanisms Trade-Off Analysis:** By integrating safety and security analysis, it is possible to analyze trade-offs between control and counter-measures proposed to support safety and security arguments. On the one hand, **safety and security measures may support each other, making one of them superfluous.** For example [27, 36], there is not need to use CRC (Cyclic Redundancy Check) mechanisms for safety, if messages are being signed with MAC (Message Authentication Codes) as the latter already achieves the goal of checking for message corruption. On the other hand, **safety and security mechanisms may conflict with each other.** For example, emergency doors increase safety by allowing one to exit a building in case of fire, but it may decrease security by allowing unauthorized persons to enter the building. Such trade-off analysis can help solve conflicts as well as identify and remove redundancies reducing product and development costs.

**Outline** Model-Based Engineering (MBE) is widely used by industries such as automotive [3], avionics [10] and Industry 4.0 [8] for developing systems. The scope of the methods we propose will assume a MBE development where models play a central role. **As a white paper, we will avoid being too deep (and technical), preferring to illustrate the range of possibilities with examples.** In future works, however, we will describe in detail the techniques used as well as propose extensions.

Section 1 identifies key technical challenges for the integration of safety and security. Section 2 reviews briefly the main techniques used for safety and security. Section 3 describes how MBE provides the basic machinery for integrating safety and security. Section 4 describes, by example, how one can extract security relevant information from safety assessments. Section 5 describes how the evaluation of security assessments can impact the confidence in safety assessments. Section 6 builds on the material in the previous sections and proposes a *collaborative safety and security development process*. Section 7 describes how to carry out trade-off analysis between safety and security mechanisms. We illustrate how the detection of conflicts can be done automatically. Finally, Section 8 concludes by pointing out our next steps.

**We also point out that the techniques described here have been implemented or under implementation as new features of the Model-Based tool AF3 [1] maintained by fortiss's MBSE department.**

## 1. Main Technical Challenges

This section introduces some of challenges that we believe are important for safety and security integration and therefore, shall and will be tackled in the following years.

In the Introduction, we mentioned that the difference in safety and security mind-sets lead to different techniques for carrying out safety and security assessments. One lacks a common ground, that is, a language that can be used to integrate both safety and security assessments. Without such common ground there is little hope to integrate safety and security in any meaningful way. This leads to our first challenge:

**Challenge 1:** *Develop a common language which can be used to integrated safety and security assessments.*

Safety assessments contain useful information for security engineers to evaluate how attacks can cause catastrophic events. Indeed, safety assessments contain the main hazards, how these can be triggered, control mechanisms installed, entry points, etc. However, safety assessments are written in the most varied forms and often in non-machine readable formats, *e.g.*, Word documents. This prevents security engineers to use safety assessments effectively in order to understand how cyber-attacks could affect the safety of a system. This leads to our second challenge:

**Challenge 2:** *Develop techniques allowing the (semi-)automated extraction of relevant security information from safety assessments.*

It is not reasonable to conclude the safety of an item without considering its security concerns. For example, an airbag cannot be considered safe if an intruder can *easily* deploy it at any time. This means that security assessments related to safety hazards shall impact security assessments. So, if a security assessment concludes that there is unacceptable security risk of the airbag being deployed, this conclusion should render the airbag safety unacceptable. To do so, we need techniques for incorporating the conclusions of the security assessments in safety assessments. This leads to our third challenge:

**Challenge 3:** *Develop techniques allowing the incorporation of relevant security assessment findings into safety assessments.*

Safety and security assessments often lead to changes on the architecture by, for example, incorporating control and mitigation mechanisms. Many times, these mechanisms may support each other to the point of being redundant. If this is the case, some mechanisms may be removed thus reducing costs. On the other hand, some mechanisms may conflict each other. Therefore, decisions may need to be taken, *e.g.*, finding alternative mechanisms or prioritizing safety over security or vice-versa. That is, trade-offs shall be carried out. However, there are no techniques for carrying out such trade-offs, leading to our fourth challenge:

**Challenge 4:** *Develop techniques allowing the identification of when safety and security mechanisms support, conflict or have no effect on each other, and to carry out trade-off analysis.*

In order to certify a system, developers have to provide enough evidence, *i.e.*, safety arguments, supporting system safety. As mentioned in the Introduction, with interconnected systems, safety arguments shall also consider security. Currently, safety arguments provide detailed quantitative evaluation for the safety of systems based on the probability faults. When taking security into account such quantitative evaluations no longer make sense as the probabilities of an attack to occur are in another order of magnitude as the probability of faults. Unfortunately, there are no techniques to present such quantitative safety arguments taking into account security. This leads to our last challenge:

**Challenge 5:** *Develop techniques for quantitative evaluation of system safety including acceptable risk from security threats.*

This white paper provides some ideas on how we plan to tackle these challenge. Challenge 1 and 2 are treated in Section 4; Challenge 3 and 5 are treated in Section 5. Challenge 4 is treated in Section 7. We plan in the next years to build and expand on these ideas.

## 2. Safety and Security Techniques

This section briefly reviews the main techniques used for establishment of safety and security as well as some proposals for integrating safety and security. Our goal is not to be comprehensive, but rather review established techniques that will be used in the subsequent sections.

### 2.1 Safety

We review some techniques used by engineers to evaluate and increase the safety of a system, namely, Fault Tree Analysis (FTA), Failure Modes and Effect Analysis (FMEA), Goal Structured Notation, and safety mechanisms.

**Fault Tree Analysis (FTA):** FTA is a top-down approach used in order to understand which events may lead to undesired events. It is one of the most important techniques used in safety engineering. An FTA is a tree with the root labeled with the top undesired event. The tree contains "and" and "or" nodes specifying the combination of events that can lead to the undesired event.

Consider, for example, the FTA depicted in Figure 1. The undesired event is Y placed at the root of the tree. A safety engineer is interested on the *cut sets of an FTA*, that is, the collections of events, normally component faults, that

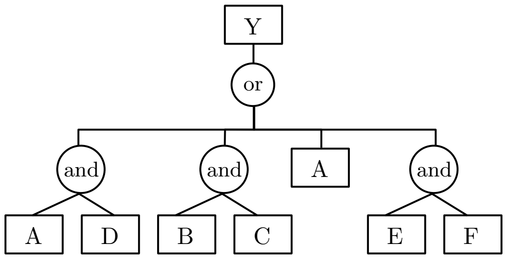

**Figure 1.** FTA Example.

lead to the undesired event. For this FTA example, the cut sets are:

$$\{A,D\}, \{B,C\}, \{A\}, \{E,F\}$$

as any of these combinations lead to the event Y. If both A and D happen at the same time, the left-most and branch is satisfied leading to the event Y.

From a FTA, one can compute the *minimum cut sets*, that is, the minimum set of cut sets that encompasses all possible combinations of triggering an undesired event. The minimum cut set for the given example is

$$\{B,C\}, \{A\}, \{E,F\}$$

Notice that the event A already triggers the event Y. Therefore, there is no need to consider the cut set {A,D}as it is subsumed by the cut set {A}.

Given the minimum cut sets, a safety engineer can, for example, show compliance with respect to the safety requirements. This may require placing control measures to reduce the probability of the corresponding undesired event.

As we argue in Section 4, FTA provides very useful information for security engineers. Indeed, an attack triggering, for example, the event A would lead to an undesired (possibly catastrophic) event. This means that penetration tests could be more focused in assessing how likely/easy it is to trigger event A, rather than finding this from scratch.

**Failure Modes and Effect Analysis (FMEA):** FMEA is a bottom-up approach (inductive method) used to systematically identify potential failures and their potential effects and causes. Thus FMEA complements FTA by instead of reasoning from top-level undesired events as in FTA, adopting a bottom-up approach by starting from faults/failures of sub-components to establish top level failures.

FMEAs are, normally, organized in a table containing the columns: Function, Failure Mode, Effect, Cause, Severity, Occurrence, Detection and the RPN value.

Failure modes are established for each function. Examples of failure modes include [7]:

- **Loss of Function**, that is, when the function is completely lost;
- **Erroneous**, that is, when the function does not behave as expected due to, for example, an implementation bug;
- **Unintended Action**, that is, the function takes the action which was not intended;
- **Partial Loss of Function**, that is, when the function does not operate at full operation, *e.g.*, some of the redundant components of the function are not operational.

Effect and cause are descriptions of, respectively, the impact of the failure mode of the function to safety and what could a cause for such failure be, *e.g.*, failures of sub-components. Severity, Occurrence and Detection are numbers, ranging normally from 1-10. The higher the value for severity the higher the impact of the failure. The higher the value for occurrence the higher is the likelihood of the failure. The higher the value of detection the less likely it is to observe (and consequently activate control mechanisms) the failure.

Finally, the value RPN is computed by multiplying the values for severity, occurrence and detection. It provides a quantitative way of classifying the importance of failure modes. The higher the value of RPN of a failure the higher its importance.

As we argue in Section 4, FMEAs also provide useful information for security engineers. For example, it describes the severity of failure modes and its causes. Therefore, security engineers can use this information to prioritize which attacks to consider. Notice, however, that occurrence does not play much importance for security engineers as occurrence in FMEA does not reflect the likelihood of attacks to occur, but rather the likelihood of faults/failures.

**Safety Mechanisms:** Safety mechanisms, such as voters, watchdogs, are often deployed in order to increase the safety of a system. For example, consider the hazard *unintended airbag deployment*. Instead of relying on a single signal, *e.g.*, crashing sensor, to deploy an airbag, a voter can be used to decide to deploy an airbag taking into account multiple (independent) signals, *e.g.*, crashing sensor and gyroscope, thus reducing the chances for this hazard.

However, as pointed out by Preschern *et al.* [38], safety mechanisms themselves can be subject to attacks. For example, an attacker may tamper the voter leading to a hazard. As we detail in Section 4, if security engineers are aware of the deployment of such mechanisms, they can assess how likely it is to attack them to trigger a hazard.

**Goal Structured Notation (GSN):** Safety assessments are complex, breaking an item safety goal into safety sub-goals, *e.g.*, considering different hazards, and often applying different methods, *e.g.*, FTA, FMEA, Safety Mechanisms. GSN [12] is a formalism introduced to present safety assessments in a semi-formal fashion.

Since its introduction, different safety arguments have been mapped to GSN patterns. Consider, for example, the GSN pattern depicted in Figure 2. It specifies the argument by analysing all the possible/known hazards to an item's safety. It is assumed that all the hazards are known. For each hazard a safety argument, also represented by a GSN, is specified. At the leaves of the GSN, one describes the *solutions* that have been taken, *e.g.*, carry out FTA, FMEA, safety designs, etc.

Clearly, such safety arguments can provide important information for security. For example, it contains the key

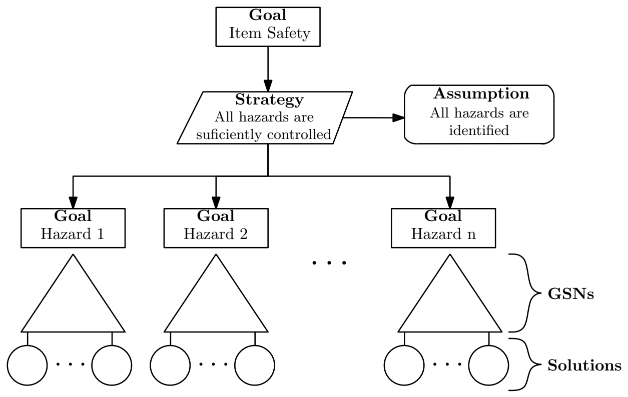

**Figure 2.** GSN Hazard Pattern.

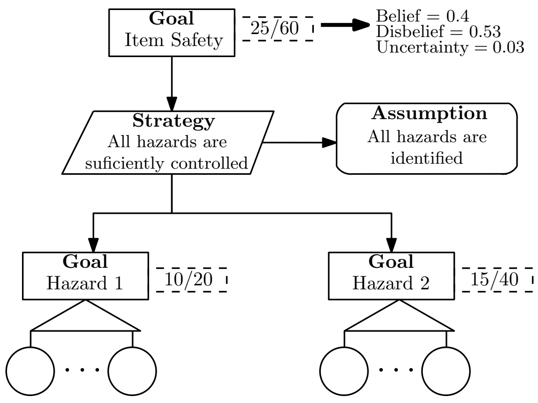

**Figure 3.** Example of GSN with Quantitative Information. Here the pair $m/n$ attached to goals specifies, respectively, the number of defeaters outruled and the total number of identified defeaters.

safety hazards of an item. It also contains what type of solutions and analysis have been carried out. However, a problem of GSN is the lack of more precise semantics. The semantics is basically the text contained in the GSN, which may be enough for a human to understand, but it does not provide enough structure for extracting automatically security-relevant information. In Section 4, we extend GSNs and constructing security models, namely, Attack Trees, from a GSN.

Finally, recent works [44, 23] have proposed mechanisms for associating GSN with quantitative values denoting its confidence level. These values are expressed as Dempfer-Schafer theories [22] containing three values for, respectively, the Belief, Disbelief, and Uncertainty on the safety assessment. These values may be assigned by safety experts [44] or be computed from the total number of identified defeaters[2] and the number of defeaters one was able to outrule [23].

We illustrate the approach proposed by Duan *et al.* [23]. Consider the GSN depicted in Figure 3. It contains a main goal which is broken down into two sub-goals. GSN goals are annotated with the number of defeaters outruled and the total number of defeaters. Intuitively, the greater the total number of defeaters, the lower is the uncertainty. Moreover, the greater the number of outruled defeaters the greater the belief on the GSN and the lower the disbelief. In Figure 3, a total of 60 = 20 + 40 defeaters have been identified and only 25 = 10 + 15 have been outruled. These values yield a Belief of 0.4, Disbelief of 0.53 and Uncertainty of 0.03.[3] If further 20 defeaters are outruled, then the Belief is increased to 0.73, the Disbelief reduces to 0.24 and the Uncertainty remains the same value 0.03.

Intuitively, only arguments that have high belief, thus low uncertainty and low disbelief, shall be accepted. As we argue in Section 5, such a quantitative information can be used to incorporate the results of security assessments in safety assessments. For example, if no security assessment has been carried out for a particular item, then the associated uncertainty shall increase. On the other hand, if a security has been carried out establishing that the item is secure, then the belief on the safety of the item shall increase. Otherwise, if an attack is found that could compromise the safety of the item, then the disbelief shall increase.

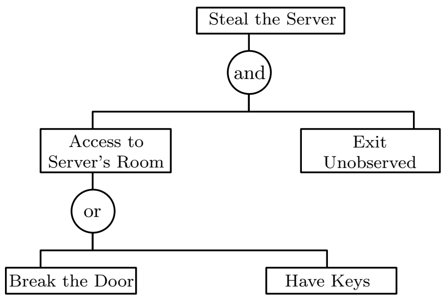

**Figure 4.** Attack Tree Example.

## 2.2 Security

We review some models used for carrying out threat analysis. More details can be found in Shostack's book [41] and in the papers cited.

**Attack Trees:** First proposed by Schneier [40], attack trees and its extensions [19, 30] are among the main security methods for carrying out threat analysis. An attack tree specifies how an attacker could pose a threat to a system. It is analogous to GSN [12] but, instead of arguing for the safety of a system, an attack tree breaks down the possibilities of how an attack could be carried out.

Consider, for example, the Attack Tree depicted in Figure 4. It describes how an intruder can successfully steal a server. He needs to have access to the server's room and be able to exit the building without being noticed. Moreover, n order to access to the server's room, he can break the door or obtain the keys.

**Attack Defense Trees (ADTs):** Attack defense trees [30] extend attack trees by allowing to include counter-measures to attack trees. Consider the attack defense tree depicted in Figure 5 which extends the attack tree depicted in Figure 4. It specifies counter-measures, represented by the dotted edges, to the possible attacks. For example, "breaking the door" can be mitigated by installing a security door which

[2]A defeater is a belief that may invalidate an argument.

[3]We refer to the work of Jøsang [29] on how exactly these values are computed.

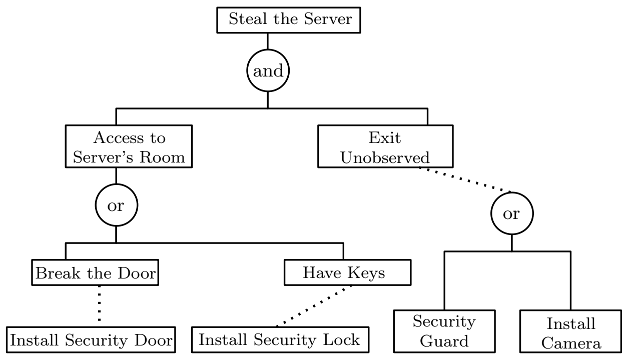

**Figure 5.** Attack Defense Tree Example.

is harder to break into. Similarly, installing a security camera or hiring a security guard can mitigate that the attacker leaves the building undetected. Attack defense trees also allow to model how attackers could attack mitigation mechanisms. For example, a cyber-attack on the security camera causing it to record the same image reduces the camera's effectiveness.

**Quantitative Attack Defense Trees** Attack Defense Trees are not only a qualitative threat analysis technique, but they provide quantitative information [17, 19]. Quantitative information can represent, for example, "what is the likelihood and impact of the attack?", "what are the costs of attacking a system?", "how long does the attack require?". Bagnato *et al.* [17] propose mechanisms to associate quantitative information to attack defense trees and to carry out computation to answer such questions. From the quantitative information, security engineers shall decide whether the security risk is acceptable.

## 2.3 Safety and Security

The problem of safety and security has been known for some time already and techniques have been proposed. They fall into the following two main categories:

**General Models for Both Safety and Security:** A number of works [35, 28, 34] have proposed using general models encompassing both safety and security concerns. For example, GSN extensions with security features, so that in a single framework, one can express both security and safety [34].

Although it is an appealing approach, it does not take into account the different mind-sets between safety and security, which poses serious doubts on the practicality of such approach. On the one hand, security engineers do use GSNs for threat modeling and it is hard to expect them to combine security threats with solutions such as FTA, FMEA, etc. On the other hand, safety engineers are not security experts, so it is hard to expect that they would develop deep security analysis.

**Safety Assessments used for Security Analysis:** Instead of building a general model for both safety and security, some approaches [25, 39, 43] propose the development of safety assessments and then "passing the ball" to security engineers to carry out security analysis based on the safety assessments.

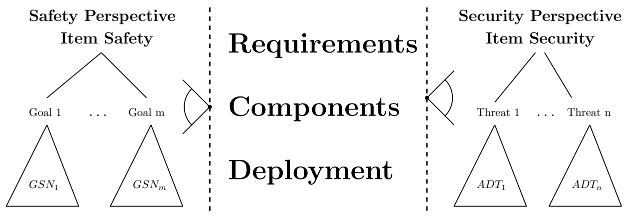

**Figure 6.** Safety and Security Lenses.

An example of this approach is the use of standard (natural) language, such as Guide Words [25], with information in safety assessments relevant for carrying out security assessments. For example, HAZOP uses guide words to systematically describe the hazards, such as under which condition it may occur. This information can provide hints for carrying out security analysis.

Recently, Dürrwang *et al.* [25] have proposed the following set of guide words for embedded systems: *disclosure*, *disconnected*, *delay*, *deletion*, *stopping*, *denial*, *trigger*, *insertion*, *reset*, and *manipulation*. These words provide a suitable interface between safety and security terminology thus allowing security engineers to better understand and reuse work carried out by safety engineers. This methodology has been used [24] to find a security flaw in airbag devices.

## 3. Integrating Safety and Security using MBE

Model-Based Engineering (MBE) proposes development by using (domain-specific) models, such as GSNs [12], Attack Defense Trees [40], Matlab Simulink [5], SysML [11] and AF3 [1]. These models are used to exchange information between engineers to, for example, further refine requirements, implement software/systems/architecture, including software deployment.

This is illustrated by Figure 6. Requirements are traced to components that are then embedded into the given hardware. These model-elements (requirements, components and deployments) reflect different development aspects, including safety an security. Safety arguments expressed as GSN are reflected into the model-elements. For example, the handling of hazards shall be expressed as *safety requirements* and safety designs, such as voters, as *safety design requirements*. Similarly, threats identified by security arguments shall yield *security requirements* and countermeasures as *security design requirements*.

**Features of our Approach:** As we will illustrate in the following sections, MBE provides a general framework for the integration safety and security through model-elements. We enumerate below some of the differences/features of our approach with respect to existing approaches described in Section 2.3:

1. **GSN and ADT Integration:** Instead of natural language as in a Guide Words approach (see Section 2.3), we use models, GSNs and ADTs. Models contain much more information than Guide Words, *e.g.*, traces to components, logical relation of solutions and hazards, quantitative evaluation. Furthermore, as we

describe in Section 4 and 5, information from GSN can be used to construct ADTs automatically and evaluations of ADTs can be incorporated into the evaluation of GSNs impacting a GSN's confidence;

2. **Development as a Game:** On the one hand, models that contain both safety and security annotations, like security extensions of GSN [35], require that safety and security work closely together in a single model, instead of using specialized techniques and models for safety and security. On the other hand, Guide Words allow safety and security to use their own techniques, but collaboration resumes to a single "passing the ball" from safety to security. This means that security is not taken into account for safety.

   Our development proposal has the advantages of both methods above. It is a collaborative process where the "ball is passed" between safety and security engineers until an acceptable risk is reached. Moreover, it also allow safety and security engineers to use their own specialized models (GSN, FTA, FMEA, etc for safety and Attack Defense Trees for security). We describe our process in Section 6 being illustrated by Figure 10

3. **Trade-Off Analysis:** Models also help to carry out trade-off analysis. In particular, GSNs contain solutions, such as safety mechanisms, and ADTs contain counter-measures, such as security mechanisms (such as counter-measures). As we illustrate in Section 7, we can identify when safety and security mechanisms contradict each other. Once a conflict is identified, compromises should be found, *e.g.*, finding other mechanisms or prioritizing safety over security. We describe how such contradiction can be solved.

## 4. Safety to Security

This section describes how safety assessments, expressed as GSNs, can be used by security engineers. As described in Section 2.1, a GSN contains safety details, such as the key hazards, safety methods (FTA, FMEA), and safety mechanisms used (Voters, Watchdogs) to control hazards. These details can be very useful for carrying out security assessments, such as understand which are the hazards, how they can be triggered, which safety mechanisms could be attacked. Our main goal here is to illustrate how a GSN can be transformed into Attack Tree specifying a preliminary security assessment for the item assessed by the GSN.

However, the first obstacle we face is that GSNs are syntactic objects, where its nodes are described with (arbitrary) text, lacking thus more precise semantics. It is, therefore, not possible to extract systematically from a GSN security relevant information. That is, GSN lacks a common language for safety and security integration (Challenge 1).

We overcome this obstacle by assigning semantics to GSN nodes, inspired by the work on Guide Words (Section 2.3).[4] We illustrate this with an example. Consider the GSN depicted to the left in Figure 7 derived from the Dürrwang *et al.*'s recent work on airbag security [24]. There are two main safety hazards to be considered for airbag safety:

- *Unintended Prevention of Airbag Deployment*, that is, during a safety critical event, *e.g.*, an accident, the airbag is not deployed. The failure to deploy the airbag reduces the passenger safety during accidents. Notice, however, that other safety mechanisms, *e.g.*, safety belt, may still be enough to ensure passenger safety;

- *Unintended Airbag Deployment*, that is, the airbag is deployed in a situation not critical. A passenger, *e.g.*, a kid, sitting while the car is parked may be hurt if the airbag is deployed. Different from the previous hazard, safety mechanisms, *e.g.*, safety belt, do not ensure the passenger safety. Additional safety mechanisms shall be implemented, such as Voters, as depicted in the GSN.

All this information is just described textually in the GSN. However, they shall be reflected in safety and safety design requirements as depicted in Figure 7 by the dashed lines. We propose adding additional meta-data to these requirements, called *domain specific requirements*. The exact meta-data may vary depending on the domain. For embedded systems, safety requirements shall contain data such as:

- *Hazard Impact*, which specifies how serious the corresponding hazard is to the item safety. From the reasoning above, the hazard *Unintended Prevention of Airbag Deployment* has a low impact, while *Unintended Airbag Deployment* has a high impact;

- *Mechanism* which may be one of the Guide Words detailed in Section 2.3. For example, the hazard *Unintended Airbag Deployment* is caused by triggering of the air-bag component;

- *Trace* from requirement to component is already part of the MBE development. It relates a requirement to a component in the architecture. In this case, the GSN nodes refer to the airbag component.

Similarly, solutions, such as voters, are mapped to safety design requirements, for which, we also attach some meta-data. Different types of solutions (FTA, FMEA, Safety Mechanisms) would involve different meta-data. In the case of voters, one specifies the signals used by the voters ($\text{Sig}_1, \ldots, \text{Sig}_N$), the threshold, $M$, used for deciding when the voter is activated. In our Airbag example, its voter uses signals from the Gyroscope and the Crash detector. Only if all of them indicate a crash situation, then the airbag is deployed.

Notice that the meta-data attached to domain specific requirements basically reflect the content in the GSN node, but in a uniform format which can be machine-readable. **This meta-data provides semantic information to GSN**

[4]One could attempt to provide a more general semantics to GSN trees instead of only its nodes. However, it is not clear yet how this can be done and left to future work. We focus here, instead, on adding enough/minimal meta-data in order to provide useful safety and security integration.

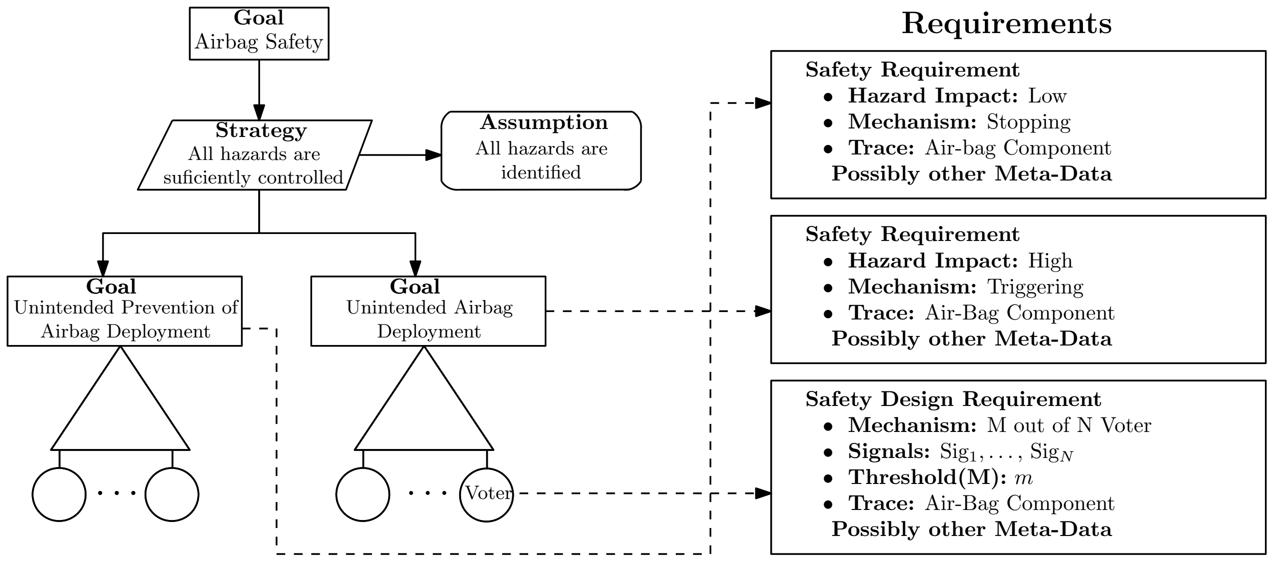
Goal
Airbag Safety
Strategy
All hazards are suficiently controlled
Assumption
All hazards are identified
Goal
Unintended Prevention of Airbag Deployment
Goal
Unintended Airbag Deployment
Voter
Requirements
Safety Requirement
• Hazard Impact: Low
• Mechanism: Stopping
• Trace: Air-bag Component
Possibly other Meta-Data
Safety Requirement
• Hazard Impact: High
• Mechanism: Triggering
• Trace: Air-Bag Component
Possibly other Meta-Data
Safety Design Requirement
• Mechanism: M out of N Voter
• Signals: $\text{Sig}_1, \ldots, \text{Sig}_N$
• Threshold(M): $m$
• Trace: Air-Bag Component
Possibly other Meta-Data

**Figure 7.** GSN: Airbag Deployment GSN and example of attaching semantics to GSN using domain specific requirements.

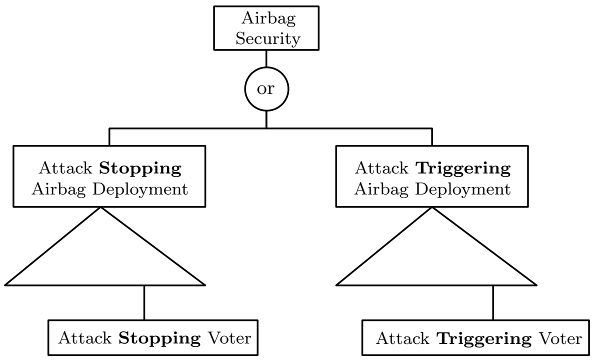
Airbag Security
or
Attack Stopping Airbag Deployment
Attack Triggering Airbag Deployment
Attack Stopping Voter
Attack Triggering Voter

**Figure 8.** Attack Tree for the Airbag Item.

**nodes.** For example, the meta-data in the requirement associated with the node *Unintended Airbag Deployment* specifies that the node represents a hazard of high impact which can be the result of triggering the airbag component.[5]

The information associated to GSN is enough to extract useful information for security engineers, allowing to construct (automatically) an attack tree on the security of an item from its corresponding GSN. For example, the attack tree depicted in Figure 8 can be extracted from the Airbag-GSN depicted in Figure 7. From the attack tree, security engineers can identify two different types of attacks, stopping airbags or triggering airbag deployment. Notice how the guide words **stopping** and **triggering** are used in the construction of the trees. Moreover, from the impact information, security engineers can understand the impact of these attacks, namely, triggering is more harmful than stopping airbag deployment, thus helping prioritize resources, *e.g.*, penetration testing.

Notice that while the voter only appears in one branch of the airbag GSN, attacks appear in both branches of the airbag attack tree. This is because an attack stopping the voter stops airbag deployment. This can be automatically inferred by the meta-data of the voter, first, specifying that it is a M out of N voter and that it is traced to the airbag component.

Solutions, such as, voters, FTA, FMEA, can also be translated to attack sub-trees.

- **Safety Mechanisms:** A safety mechanisms can normally be subject to a large number of attacks as enumerated by Preschern *et al.* [38]. We can use Guide Words to reduce this list to those attacks that are relevant. For example, an *attack triggering a voter* M out of N can be achieved by *spoofing* M signals or by tampering the voter. It is not necessary to consider denial of service attacks. On the other hand, *stopping the voter* may be achieved by carrying out a denial of service attack on the voter;
- **FTA:** The minimum cut-sets (see Section 2.1) resulting from the FTA can be used to construct attack trees. For example, if $\{ev_1, \ldots, ev_n\}$ is a minimum cut-set, then an attack would consist of carrying out attacks to trigger all events $ev_1, \ldots, ev_n$, by, for example, spoofing them;
- **FMEA:** The table of failures composing an FMEA (see Section 2.1) can also be used to construct an attack tree. In particular, the field failure mode specifies the type of attack on the corresponding function. For example, a *loss of function* entry can be achieved by denying service or tampering the function. Similarly, the severity field indicates how serious the failure mode is and the detection field indicates how disguised the attack can be. It seems possible to transform this information into quantitative information attached to attack trees [17, 19]. This is left for future work.

Finally, notice that the attack tree constructed from a GSN provides a preliminary attack tree on the item in ques-

[5]We are taking extra care to develop domain specific requirements to contain simple, but useful meta-data. While one could be more formal and express requirements in formal languages, such as Linear Temporal Logic [37], our experience shows that they are not effective in practice as few engineers and even specialists can write such formulas.

tion. This tree may be extended considering other possible attacks and attaching counter-measures.

## 5. Security to Safety

As described in Section 2.1, it is possible to attach quantitative evaluation to GSN based on the number of *defeaters* identified and overruled. The result of the quantitative evaluation are three non-negative real values, $B,D,U$, in $[0,1]$ such that $B+D+U=1$: $B$ is the belief on the GSN, $D$ the disbelief and $U$ the uncertainty. (See the work of Duan *et al.* [23] for more details.) A GSN shall only be acceptable if it has a high enough level of belief and low enough levels of disbelief and uncertainty. The exact degree of belief may depend on different factors, such as, how critical the item.

Security threats are possible defeaters for GSNs as they may invalidate the safety argument. There are the following possibilities according to the security assessment carried out:

- **No Security Assessment for the Item:** If no security assessment has been performed, then it is a defeater that has not yet been outruled and therefore, the uncertainty of the GSN shall be increased.
- **Existing Security Assessment for the item:** There are two possibilities[6]:
  - **Acceptable Security Risk:** If the security assessment concludes that there is acceptable security risk, that is, identified threats are sufficiently mitigated, then this shall have a positive effect on the belief of the corresponding GSN;
  - **Unacceptable Security Risk:** On the other hand, if the identified threats are not sufficiently mitigated, leading to an unacceptable risk, the disbelief of the safety case shall be reduced.

Figure 9 illustrates how one can integrate GSNs and ADTs. The value $w$ is a non-negative value specifying the importance of the security assessment for the item safety. The greater the value of $w$, the greater is the impact of the security assessment. For instance, if $w$ is zero, then the impact of the security assessment on the safety assessment is negligible. Depending on the evaluation of the item security as described above, the levels of confidence of the GSN are updated to $B_1,U_1,D_1$.

We illustrate this with an example implementing a simple update function. Notice, however, that other functions can be used (and subject to future work). Consider that $B=0.70, D=0.20, U=0.10$ and $w=2$. The values for belief, disbelief and uncertainty are updated taking into account the security assessment for the item in question if there is any as detailed below and the weight $w$:

[6]There are many ways to quantify an attack defense tree, *e.g.*, the effort, time, cost required by the attacker to attack an item. Based on these values together with other parameters, *e.g.*, the value of the item, security engineers can evaluate whether the risk is acceptable or not. For example, if all identified attacks to an item take too long to take place, then the risk of such attacks is acceptable.

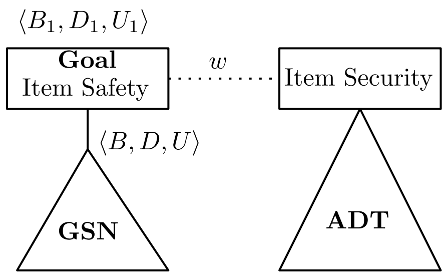

**Figure 9.** Illustration of GSN and ADT integration. Here, the values $B,U,D$ are, respectively the levels of belief, disbelief and uncertainty of the safety assessment expressed in the GSN. The new levels of belief, disbelief, and uncertainty, $B_1,U_1,D_1$, are obtained after integrating the security assessments (if any) taking into account the weight $w$, a non-negative number.

- **No Security Assessment:** In this case, the uncertainty shall increase. We do so by first updating the values for the belief and disbelief, reducing their values according to $w$ as follows:

  $$B_1 = B/(1+w) = 0.7/(1+2) = 0.23$$
  $$D_1 = D/(1+w) = 0.2/(1+2) = 0.07$$

  Then we compute the uncertainty as follows:

  $$\begin{aligned}U_1 &= U + (B - B_1) + (D - D_1)\\ &= 0.1 + (0.7 - 0.23) + (0.2 - 0.07) = 0.7.\end{aligned}$$

  where the uncertainty increases.

- **Acceptable Security Risk:** In this case, the belief shall increase. We do so by carrying out the following computations similar to above, where uncertainty and disbelief decrease:

  $$U_1 = U/(1+w) = 0.1/(1+2) = 0.03$$
  $$D_1 = D/(1+w) = 0.2/(1+2) = 0.07$$

  Then, we compute the new belief as follows:

  $$\begin{aligned}B_1 &= B + (D - D_1) + (U - U_1)\\ &= 0.7 + (0.2 - 0.07) + (0.1 - 0.03) = 0.9.\end{aligned}$$

  where the belief increases.

- **Unacceptable Security Risk:** In this case, the disbelief shall increase. We do so by carrying out the following computations similar to above, where belief and uncertainty decrease:

  $$B_1 = B/(1+w) = 0.7/(1+2) = 0.23$$
  $$U_1 = U/(1+w) = 0.2/(1+2) = 0.07$$

  Then, we compute the new disbelief as follows:

  $$\begin{aligned}D_1 &= D + (B - B_1) + (U - U_1)\\ &= 0.2 + (0.7 - 0.23) + (0.1 - 0.03) = 0.7.\end{aligned}$$

  where the disbelief increases.

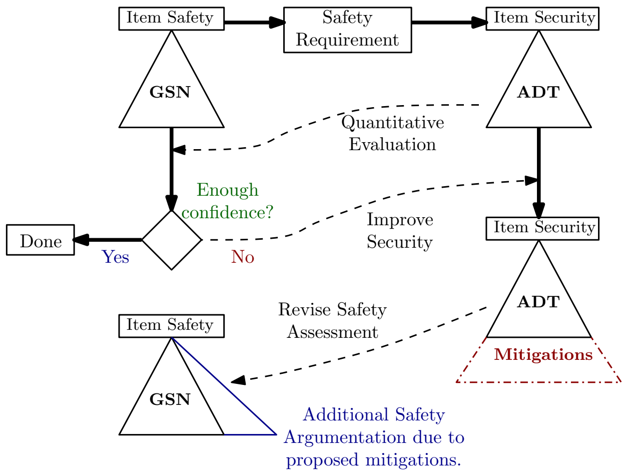

**Figure 10.** Collaborative Safety and Security Process Cycle.

Notice that in all cases the new values, $B_1, D_1, U_1$, remain within the interval [0,1] and $B_1 + D_1 + U_1 = 1$. Moreover, notice that if $w = 0$, then $B_1 = B, D_1 = D, U_1 = U$, that is, the security assessment does not affect the safety assessment.

The use of quantitative evaluations for GSN and ADT is a way to tackle Challenge 3 (incorporation of relevant security findings into safety assessments) and Challenge 5 (quantitative evaluation for safety including security assessments), as we are able to incorporate the conclusion of security asssessments into the quantitative evaluation of safety assessments and at the same time provide a quantification on the credibility of the safety case in terms of belief, disbelief and uncertainty.

## 6. Collaborative Process for Safety and Security

In Sections 4 and 5, we described how safety assessments in the form of GSNs can be used for constructing security assessments in the form of ADT, and moreover, how security assessment results can be integrated into safety assessment by modifying its levels of belief, disbelief and uncertainty. In this section, we describe how these techniques can be put together as a collaborative process for building an integrated safety and security assessments.

Consider the process cycle illustrated by Figure 10.

- **Initial Safety Assessment:** Assume given an initial safety assesssment for an item represented as a GSN. This starts the process by issuing safety requirements (with meta-data as described in Section 4);
- **Security Assessment:** Using the machinery described in Section 4, we can build an ADT for the item from the GSN. This ADT may be extended with new threats as well as with mitigation mechanisms;
- **Security Feedback to Safety:** Using the machinery described in Section 5, the evaluation of the security assessment is integrated into the GSN yielding values for belief, disbelief and uncertainty. One finishes the safety and security collaboration if these values are acceptable. Otherwise, there are two possibilities: Either refine the safety assessment, *e.g.*, outrruling more defeaters, or as depicted in Figure 10, request for a better security;
- **Additional Mitigations:** Once the request of improving security is received, security engineers can add further mitigation mechanisms in order to improve its security;
- **Safety Revision:** The mitigation mechanisms may impact the safety of the system, *e.g.*, add additional delays or add new single points of failure, etc. This may yield additional safety argumentation.

This collaborative development cycle repeats, possibly adding new safety and security mechanisms, until an acceptable security risk is reached.

**Airbag Example:** To illustrate this cycle, let us return to the Airbag safety assessment expressesd by the GSN depicted in Figure 7. From this GSN, we can construct corresponding attack tree in Figure 8. This ADT shall yield an unacceptable risk as the threats of stopping the voter and triggering the voter have not been further investigated. This impacts the safety assessment by reducing its belief, disbelief and uncertainty (as described in Section 5). Assume that these values are not acceptable. Thus, the security engineer is requested to improve the ADT.

In order to improve the ADT, the security engineer may evaluate the risk of, for example, stopping the voter and triggering the voter. However, from the information contained in the hazard meta-data (Figure 7), the impact of stopping the voter is lower than the impact of triggering the voter. Therefore, the security engineers may decide to further investigate the attack triggering the voter. They may discover, for example, the attack found by Dürrwant *et al.* [24] on the security access mechanism which poses a serious threat. In order to mitigate this threat, they can add as counter measure to perform plausibility checks as suggested by Dürrwant *et al.* [24], which would reduce the security risk.

As new counter-measures have been added, a request to revise safety assessments is issued. Safety engineers have to then argue that the plausibility checks are safe, that is, may not affect the airbag safety, by, for example, preventing airbag deployment. New safety mechanisms may be placed if necessary which may lead to new threats to be analyzed by security engineers. This process ends when the levels of belief, disbelief and uncertainty are acceptable.

## 7. Trade-off Analysis

In this section, we describe methods towards carrying out trade-off analysis between safety and security mechanisms. Such analysis may help decide which mechanisms to be implemented. It may be that there are synergies between safety and security mechanisms which would make them redundant. For example [27, 36], CRC checks used for checking the integrity of messages and MAC used to ensure that no message is corrupted. Therefore, MAC could

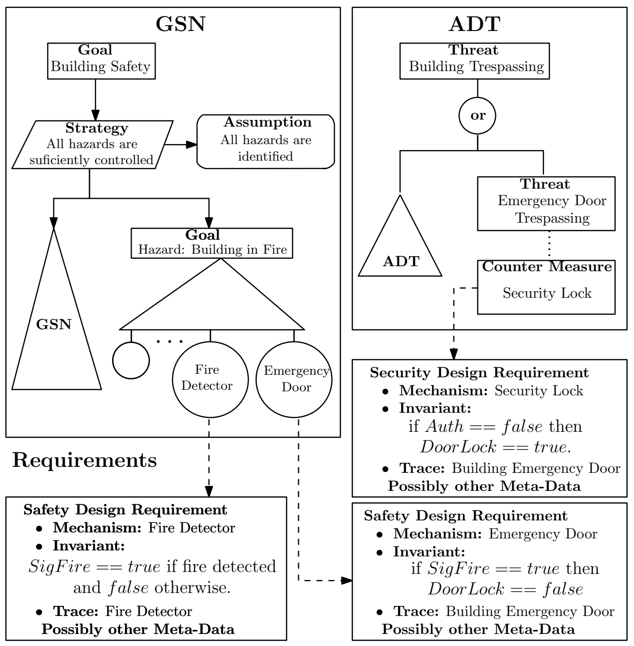
GSN
Goal
Building Safety
Strategy
All hazards are suficiently controlled
Assumption
All hazards are identified
Goal
Hazard: Building in Fire
GSN
. . .
Fire Detector
Emergency Door
ADT
Threat
Building Trespassing
or
ADT
Threat
Emergency Door Trespassing
Counter Measure
Security Lock
Requirements
Security Design Requirement
• Mechanism: Security Lock
• Invariant:
if $Auth == false$ then $DoorLock == true$.
• Trace: Building Emergency Door
Possibly other Meta-Data
Safety Design Requirement
• Mechanism: Fire Detector
• Invariant:
$SigFire == true$ if fire detected and $false$ otherwise.
• Trace: Fire Detector
Possibly other Meta-Data
Safety Design Requirement
• Mechanism: Emergency Door
• Invariant:
if $SigFire == true$ then $DoorLock == false$
• Trace: Building Emergency Door
Possibly other Meta-Data

**Figure 11.** Illustration of GSN and ADT for detecting conflicts on proposed safety and security mechanisms.

replace CRC, rendering CRC not needed. On the the other hand, safety and security mechanisms may conflict, that is, interfere with their purposes. In such cases, one may have to decide on alternative mechanisms or ultimately choose either safety or security.

We illustrate how MBE can be used to carry out these trade-offs with an example. Consider the safety and security arguments expressed by a GSN and an ADT and depicted in Figure 11 of a building. The arguments express the following concerns:

- **Building Safety:** The GSN specifies, among other possible hazards, controlling the hazard of people getting hurt when the building is in fire. It proposes as solutions installing a *fire detector* and an *emergency door*. These solutions are associated with domain specific safety requirements (pointed by the corresponding dashed lines). These requirements are functional requirements specifying that the *boolean signal SigFire* is true when a fire is detected and false otherwise and that when *SigFire* is true then the emergency door shall be unlocked, that is, the signal *DoorLock* is false.

- **Building Trespassing:** The ADT breaks down the threat of a malicious intruder trespassing in the building. A possibility is by entering the building using the emergency door. This threat is mitigated by installing a security lock in the emergency door with an *authentication mechanism* (*e.g.*, biometric, code, card). This mitigation is associated to a domain specific security requirement, specifying the function of the security door: If the authentication mechanism signal *Auth* is false, then the emergency door shall be locked.

Given these arguments (GSN and ADT) and its associated domain specific requirements, it is possible to identify potential conflicts: one simply needs to check whether the requirements have intersecting set of signal names. In this example, the *DoorLock* output signal is mentioned in both the security lock requirement and in the emergency door requirement. A priori, the fact that the same signals are mentioned does not mean that there is a conflict, but only that these are *potential candidates for conflicts*. This is just one possible method for identifying conflict candidates. Other methods may use the trace, the type of requirements, etc. It is important, however, to have simple mechanisms to determine these candidates as in a usual development a large number of requirements are specified.

Once the potential candidates are identified, it remains to check whether they are indeed conflicting. We illustrate how this can be done using off-the-shelf tools. First, we extract the logical clauses in the requirements, where *SigFire* and *Auth* are input signals and *DoorLock* is the output signal:

$$\begin{array}{ll} SigFire \Rightarrow \neg DoorLock & \text{from emer. door req.} \\ \neg Auth \Rightarrow DoorLock & \text{from sec. lock req.} \end{array}$$

The question is whether these clauses can entail contradicting facts. This is clearly the case as when *SigFire* is true implies that *DoorLock* is false, and when *Auth* is false implies that *DoorLock* is true, thus yielding a contradiction.

For such (more or less) simple specifications, one can detect such contradiction manually. However, as specification become more complicated and the number of requirements increase, checking all potential contradictions for actual contradictions becomes impractical. It is much better is to automate this process as we demonstrate with this example.

Before, however, we should point out that traditional classical logic (propositional logic) is not suitable for this problem because of the famous frame problem [33]. This is because only propositions that are supported by the extracted logical clauses shall be derivable. One way to solve this problem is to use the *Closed World Assumption* [32] from the knowledge representation literature [18]. We will use here the logic paradigm Answer-Set Programming (ASP) [26, 18] which allows specifications using the Closed World Assumption and the solver DLV [31][7] which supports ASP.

We start by adding for each predicate (*SigFire*, *DoorLock*, *Auth*), a *fresh predicate* with a prefix *neg* corresponding to its classical negation (*negSigFire*, *negDoorLoc*, *negAuth*). Thus it should not be possible that, for instance, *negDoorLock* and *DoorLock* are both inferred from the specification at the same time as this would be a contradiction. Second, we translate the clauses above into the following ASP program using DLV syntax:[8]

```
DoorLock :- negAuth.
negDoorLock :- SigFire.
negAuth v Auth.
negSigFire v SigFire.
contradiction :- DoorLock, negDoorLock.
:- not contradiction.
```

[7] http://www.dlvsystem.com/

[8] A logic programming clause of the form `A :- B1, ..., Bn` shall be interpreted as the clause $B_1,\ldots,B_n \Rightarrow A$.

The first two lines are direct translations of the clauses above. The lines 3 and 4 specify, respectively, that either *negAuth* or *Auth* is true[9] and either *negSigFire* or *SigFire* is true. Line 5 specifies that there is a *contradiction* if both *DoorLock* and *negDoorLock* can be derived. Finally, line 6 is a constraint specifying that only solutions that contain *contradiction* shall be considered. This is because for this example we are only interested in finding (logical) contradictions. If there is no such solution, then the theory is always consistent and therefore, the requirements are not contradicting.

For the program above, however, we obtain a single solution (answer-set) when running this program in DLV:

```
{DoorLock, negAuth, negDoorLock,
            SigFire, contradiction}
```

indicating the existence of a contradiction, namely, the one we expected where *Auth* is false and *SigFire* is true.

Once such contradictions are found, safety and security engineers have to modify their requirements. A possible solution is for the security lock to check whether there is a fire or not, that is, having the following invariant:

$$\text{if } Auth == false \text{ and } SigFire == false \text{ then } DoorLock == true.$$

which resolves the contradiction as can be checked by again using DLV.

Sun *et al.* [42] propose determining such conflicts by using the rewriting tool Maude [21]. While their encoding is much more involved than ours, the use of Maude has the potential of finding different types of conflicts, such as involving delays. This is because the encoding in Maude specifies part of the operational semantics of the system, while our encoding only takes into account the logical entailment.

Finally, this proof-of-concept example illustrates how conflicts are detected. It seems possible to also determine when requirements support each other, by adding suitable meta-data in domain specific requirements. For example, CRC and MAC solutions for the same communication channels. Further investigation is left for future work.

## 8. Conclusions

The main goal of this white paper is to set the stage for Safety and Security Engineering. We identified some key technical challenges in Section 1. We then illustrated with examples techniques that can help address some of these challenges. For example, we showed how to extract security relevant information of safety assessments by translating GSNs into ADTs. For this, we provided semantics to GSN nodes by using *domain-specific requirements*. We also showed how to use existing machinery on quantitative evaluation of GSN and ADTs to incorporate the findings of security assessments into safety assessments. We then proposed a collaborative development process where safety and security engineers incrementally build arguments using their own methods. Finally, we demonstrated how paradigms, such as Answer-Set Programming, can be used to identify when safety and security assessments are conflicting.

This is clearly the start of a long and interesting journey towards Safety and Security Engineering. Besides further developing the techniques illustrated in this white paper, we identify the following future work categorized into Techniques, Processes and Certification:

- **Techniques:** As pointed out throughout the white paper, the techniques we illustrate are going to be subject of intensive future work. We would like to answer questions such as: which meta-data should be added to domain-specific requirements or to models in order to enable further automated model translation? How can different security domains impact safety cases? How can we automatically detect other types of contradictios, such as timing contradictions? Finally, how can trade-off analysis be compiled so to facilitate conflict solving?

- **Collaborative Processes:** While here we illustrate a collaborative process involving safety and security concerns only, we are investigating how to extend this collaboration with other aspects, such as performance and quality. We are also investigating how better tooling can make the collaborative process go smoothly, *e.g.*, automated notifications;

- **Certification:** We are currently investigating how the techniques and the collaborative process cycle relate with certifications, such as the ISO 26262 [3]. A particular goal for future work is to build automated techniques that can be used to support the building of convincing safety and security assessments, complementing recent work [39] on the topic.

Finally, we plan to apply the techniques in extended use-cases from different domains. We will also report these results as scientific papers and technical reports to industry.

## Acknowledgments

We thank our industry partners, in particular, Airbus Defense and BMW Research, for valuable discussions. We also thank the AF3 team for helping us with the implementation of features in AF3. Finally, we also thank the fortiss Safety and Security Reading group.

## References

[1] AF3 – AutoFOCUS 3. More information at `https://af3.fortiss.org/`.

[2] ISO 15408, Information technology - Security techniques - Evaluation criteria for IT security (Common Criteria).

[3] ISO 26262, Road vehicles — Functional safety — Part 6: Product development: software level. Available from `https://www.iso.org/standard/43464.html`.

[9] The symbol v should not be interpreted as “or”, but more close to “x-or”, though not exactly. More details can be found at [31].

[4] ISO/SAE AWI 21434, Road Vehicles – Cybersecurity engineering. Under development.

[5] Matlab/Simulik. More information at https://in.mathworks.com/products/matlab.html.

[6] SAE J3061: Cybersecurity guidebook for cyber-physical vehicle systems. Available from https://www.sae.org/standards/content/j3061/.

[7] Standard ARP 4761: Guidelines and methods for conducting the safety assessment. Available from https://www.sae.org/standards/content/arp4761/.

[8] Standard IEC 61499: The new standard in automation. Available from http://www.iec61499.de/.

[9] Standard RTCA DO-326A: Airworthiness security process specification. Available from http://standards.globalspec.com/std/9869201/rtca-do-326.

[10] Standard RTCA DO-331: Model-based development and verification supplement to DO-178C and DO-278A. Available from https://standards.globalspec.com/std/1460383/rtca-do-331.

[11] SysML. More information at https://sysml.org/.

[12] *GSN Community Standard Version 1*. 2011. Available at http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf.

[13] Hackers remotely kill a Jeep on the highway—with me in it, 2015. Available at https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[14] Cyberattack on a German steel-mill, 2016. Available at https://www.sentryo.net/cyberattack-on-a-german-steel-mill/.

[15] A deep flaw in your car lets hackers shut dowm safety features, 2018. Available at https://www.wired.com/story/car-hack-shut-down-safety-features/.

[16] Hacks on a plane: Researchers warn it's only 'a matter of time' before aircraft get cyber attacked, 2018. Available at https://tinyurl.com/ycgfa3j8.

[17] Alessandra Bagnato, Barbara Kordy, Per Håkon Meland, and Patrick Schweitzer. Attribute decoration of attack-defense trees. *IJSSE*, 3(2):1–35, 2012.

[18] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2010.

[19] Stefano Bistarelli, Fabio Fioravanti, and Pamela Peretti. Defense tree for economic evaluations of security investment. In *ARES 06*, pages 416–423, 2006.

[20] Corrado Bordonali, Simone Ferraresi, and Wolf Richter. Shifting gears in cyber security for connected cars, 2017. McKinsey&Company.

[21] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude: A High-Performance Logical Framework*. LNCS. Springer, 2007.

[22] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 1967.

[23] Lian Duan, Sanjai Rayadurgam, Mats Heimdahl, Oleg Sokolsky, and Insup Lee. Representation of confidence in assurance cases using the beta distribution. 2016.

[24] J. Dürrwang, M. Braun, , R. Kriesten, and A. Pretschner. Enhancement of automotive penetration testing with threat analyses results. *SAE Intl. J. of Transportation Cybersecurity and Privacy*, 2018. To appear.

[25] Jürgen Dürrwang, Kristian Beckers, and Reiner Kriesten. A lightweight threat analysis approach intertwining safety and security for the automotive domain. In Stefano Tonetta, Erwin Schoitsch, and Friedemann Bitsch, editors, *SAFECOMP*, volume 10488 of *LNCS*, pages 305–319. Springer, 2017.

[26] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In *ICLP*, pages 579–597, 1990.

[27] Benjamin Glas, Carsten Gebauer, Jochen Hänger, Andreas Heyl, Jürgen Klarmann, Stefan Kriso, Priyamvadha Vembar, and Philipp Wörz. Automotive safety and security integration challenges. In Herbert Klenk, Hubert B. Keller, Erhard Plödereder, and Peter Dencker, editors, *Automotive - Safety & Security 2014 (2015), Sicherheit und Zuverlässigkeit für automobile Informationstechnik, Tagung, 21.-22.04.2015, Stuttgart, Germany*, volume 240 of *LNI*, pages 13–28. GI, 2014.

[28] Edward Griffor, editor. *Handbook of System Safety and Security*. 2016.

[29] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2001.

[30] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Foundations of attack-defense trees. pages 80–95, 2010.

[31] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7:499–562, 2006.

[32] Vladimir Lifschitz. Closed-world databases and circumscription. *Artif. Intell.*, 27(2):229–235, 1985.

[33] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. 1969.

[34] Per Håkon Meland, Elda Paja, Erlend Andreas Gjære, Stéphane Paul, Fabiano Dalpiaz, and Paolo Giorgini. Threat analysis in goal-oriented security requirements modelling. *Int. J. Secur. Softw. Eng.*, 5(2):1–19, 2014.

[35] Nicola Nostro, Andrea Bondavalli, and Nuno Silva. Adding security concerns to safety critical certification. In *Symposium on Software Reliability Engineering Workshops*, 2014.

[36] Thomas Novak, Albert Treytl, and Peter Palensky1. Common approach to functional safety and system security in building automation and control systems. 2007.

[37] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[38] Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. Security analysis of safety patterns. PLoP '13, pages 12:1–12:38, USA, 2013.

[39] Giedre Sabaliauskaite, Lin Shen Liew, and Jin Cui. Integrating autonomous vehicle safety and security analysis using STPA method and the six-step model. *International Journal on Advances in Security*, 11, 2018.

[40] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal of Software Tools*, 24:21–29, 1999.

[41] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley.

[42] Mu Sun, Sibin Mohan, Lui Sha, and Carl Gunter. Addressing safety and security contradictions in cyber-physical systems. Available at `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.3246`.

[43] Kenji Taguchi, Daisuke Souma, and Hideaki Nishihara. Safe & sec case patterns. In *SAFECOMP 2015 Workshops, ASSURE, DECSoS, ISSE, ReSA4CI, and SASSUR*, 2015.

[44] Rui Wang, Jérémie Guiochet, and Gilles Motet. Confidence assessment framework for safety arguments. In *SAFECOMP*, 2017.

# Imprint

**Published by**
fortiss GmbH
Guerickestraße 25
80805 München

**Project lead**
Dr. Vivek Nigam
fortiss GmbH

**Layout**
fortiss GmbH

**ISSN Print**
2699-1217

**ISSN Online**
2700-2977

**1st issue**
September 2018

**Photo credits**
Page 1: shutterstock ©HQuality
Page 3: ©fortissGmbH
Page 18: ©fortissGmbH

Find here more fortiss Whitepaper

fortiss is the Free State of Bavaria research institute for software-intensive systems based in Munich. The institute collaborates on research, development and transfer projects together with universities and technology companies in Bavaria and other parts of Germany, as well as across Europe. The research activities focus on state-of-the-art methods, techniques and tools used in Software & Systems-, AI- and IoT-Engineering and their application with cognitive cyber-physical systems.

fortiss is legally structured as a non-profit limited liability company (GmbH). The shareholders are the Free State of Bavaria (majority shareholder) and the Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

**fortiss GmbH**
Guerickestraße 25
80805 München
Deutschland
www.fortiss.org
Tel: +49 89 3603522 0
E-Mail: info@fortiss.org

fortiss