MÄLARDALEN UNIVERSITY
SWEDEN

LICENTIATE THESIS

Extending and Improving the Security Abstraction Model for Architectural Models of Autonomous Vehicles

Matthias Bergler
School of Innovation, Design and Engineering (IDT)
matthias.bergler@mdu.se
June 2023

Main Supervisor: Kristina Lundqvist
Co-supervisor: Ramin Tavakoli Kolagari

Abstract

Vehicle manufacturers and software developers are making considerable progress in the field of autonomous vehicle technology. Nevertheless, the highest level of autonomy, the fully autonomous vehicle without a driver, has not yet been achieved: according to media reports, there are still serious accidents involving autonomous vehicles, some of which are due to faults in the vehicle system. This results in a healthy scepticism, so that semi-autonomous vehicles have not yet been approved in many countries and there are also still concerns in society.

From a social point of view, the aspect of the dependability of the algorithms is central: since increasing digitization can not only lead to internal errors in vehicle control, but also lead to a risk of external attacks on vehicles due to the opening of many interfaces; this security aspect must already be taken into account in the early development phases of vehicle systems.

The EAST-ADL is a domain-specific architecture description language for automotive systems with comprehensive support for modeling the systems as well as diverse additional information such as requirements, safety, variability modeling, real-time behavior and so on. Current research activities at TH Nuremberg have added a Security Annex to EAST-ADL, the Security Abstraction Model "SAM".

In the context of the present work, the following two contributions were made: 1. addition of social engineering attacks to SAM, which are an increasing threat, and 2. elaboration of an EAST-ADL compatible modeling tool support in close cooperation with the tool vendor Metacase. Further additions to SAM became necessary to fully implement the ISO 21434 "Road vehicles - Cybersecurity engineering" standard; now SAM offers conceptually as well as tool-supported a comprehensive description of this standard and thus enables a seamless integration of a complete security model with the EAST-ADL modeling of the overall system.

The combination of integration and extension of the security abstraction model was subsequently evaluated in a study in cooperation with industry partners for its suitability for everyday use in the practical development of vehicle systems.

Contents

1 Introduction

In film and television, especially in the science fiction genre, people's futuristic dreams, desires and fears become reality. But these stories not only provide entertainment, they also provide research incentives for science. Devices such as smartphones or medical devices such as CT or MRI are based on devices from the Star Trek series [24]. Many of these stories also show futuristic cities with self-driving modes of transport in which passengers can pursue whatever activities they want. More and more vehicle manufacturers, such as Tesla, BMW, Mercedes etc., were influenced to also deal with the topic of autonomous driving. Tesla in particular is known for its advanced technology. Some success and failure stories have already been recorded by the media.
Companies like Tesla have shown that autonomous vehicles of the highest level, i.e., completely without a driver, are possible, and semi-autonomous vehicles are already being tested in traffic. To make this possible, data from several cameras and LIDAR (Light Detection and Ranging) systems are used to observe the vehicle environment. Internal processing units evaluate their surroundings in the form of images and sensor data and use pre-trained Machine Learning algorithms to calculate the necessary actions that have to be taken in order to get through the traffic accident-free. Unfortunately, these algorithms are not yet 100 percent reliable, maybe never will, and the tests in traffic repeatedly lead to malfunctions or failures. For example, obstacles in the form of a truck standing sideways were overlooked [17], curves were taken at the wrong angle and the vehicle left the lane or entered the opposite lane, or, as in April 2021 [42], the vehicle started to drive independently without a driver. These incidents illustrate how big the technological gap to a perfectly autonomous vehicle is. These technical deficiencies are partly software-related and partly hardware-related and must therefore be handled differently in terms of their safety. But not only the safety plays a major role, but also the security against the manipulation of such systems in order to prevent artificial caused accidents or, for example, people kidnapping. Examples worldwide show how control over vehicles can be taken over by the simplest means, for example by manipulating the infotainment system [12]. In combination with an autonomous vehicle, this can lead to devastating results. This work aims to address the need for secure automotive software systems by utilizing the Security Abstraction Model (SAM). To provide a better understanding of the development process of secure automotive systems, the background of various related standards, models, tools, and scoring systems is discussed in Chapter 2. This includes an overview of EAST-ADL, which is a domain-specific modeling language for automotive embedded systems, and AUTOSAR, an open-standard software architecture for automotive electronic control units. Additionally, the modeling tool MetaEdit+ is discussed, which allows for the creation of domain-specific languages and the integration of models and code generation.
Furthermore, the ISO21434 standard for automotive cybersecurity is introduced, which provides guidelines for the development of secure automotive systems. In Chapter 2, security is also discussed in more detail, including different types of attacks and countermeasures that can be used to mitigate these threats. Additionally, scoring systems, such as the Common Vulnerability Scoring System (CVSS) and new scores introduced with the ISO 21434, are presented as tools for evaluating the severity and likelihood of security vulnerabilities and risks.
In Chapter 3, the problem of implementing security by design with SAM is defined, highlighting the need for a comprehensive model that includes social-engineering attacks and their countermeasures. This leads to the formulation of research questions in Chapter 4, which guide the investigation of the different aspects of implementing security by design with SAM.
Chapter 5 provides an overview of the research methods and processes used to address the research questions. The contributions of this work are presented in Chapter 6, which includes the integration of social-engineering attacks into SAM, the integration of SAM with MetaEdit+ for

improved usability, and the development of different views for different stakeholders to ensure the comprehensive modeling of security aspects.
The state of the art in meta-modeling for IT-security in vehicles and social-engineering attacks is discussed in Chapter 7. This includes an overview of current research in these areas, as well as related tools and frameworks that can be used to enhance the development of secure automotive systems.
Finally, Chapter 8 concludes the work by outlining future directions for research in the field of secure automotive software systems. The integration of SAM with ISO21434 is discussed as a potential way to improve the overall security posture of automotive systems. Additionally, further work is needed to automate the newly introduced scores from ISO21434 and to evaluate the benefits of using SAM in conjunction with industrial partners.

2 Background

The design and development of modern automotive systems require a complex network of technologies and standards to ensure their safety and reliability. Two widely used standards for the design and development of automotive systems are the EAST-ADL and AUTOSAR standards. EAST-ADL is a domain-specific architecture description language for the development of automotive embedded systems, while AUTOSAR is a software architecture standard for automotive electronic control units. Both of these standards provide a framework for the design and development of automotive systems.
In order to develop secure automotive software systems, it is also important to have a clear understanding of security requirements and to implement effective security measures. To this end, several security-related standards and models have been developed, such as the Security Abstraction Model (SAM) and the ISO 21434 standard for automotive cybersecurity. SAM is a framework for modeling the security requirements of automotive software systems, while the ISO 21434 standard provides guidance on the implementation of cybersecurity in road vehicles. In addition to these standards, tools such as MetaEdit+ provide support for the creation, modification, and analysis of models such as SAM. These tools can help developers to identify potential security weaknesses and implement effective security measures.
Another important aspect of developing secure automotive systems is the use of security scoring systems. These scoring systems are used to evaluate the security posture of automotive systems and provide guidance on how to improve their security. Some examples of security scoring systems include the Common Vulnerability Scoring System (CVSS) and the Automotive Safety Integrity Level (ASIL) system.
Overall, a comprehensive understanding of these standards, models, tools, and scoring systems is crucial for the development of secure automotive software systems. In this Chapter, we will provide an overview of each of these topics and their relevance to the development of secure automotive systems.

2.1 EAST-ADL

As complexity and number of electronic components in automotive systems continue to increase, it becomes increasingly important to have effective tools and methods for designing and developing these systems. The Architecture Analysis and Design Language (AADL) [14] has been widely used in the design and development of safety-critical systems, including automotive systems. However, AADL has limitations in terms of supporting the design of automotive systems. To address this, the EAST-ADL (Embedded Automotive Systems and Technologies Architecture Description Language) was developed.
EAST-ADL is a domain-specific modeling language that provides a set of concepts, notations,

and guidelines for modeling automotive embedded systems. It was developed in the context of the European Artemis project, which aimed to develop new technologies for embedded systems. The language was developed to address the specific needs of the automotive industry, including the need for support for the modeling of safety and reliability aspects of automotive embedded systems [13].
EAST-ADL includes concepts for modeling the architecture, behavior, and requirements of automotive embedded systems. The language is based on a component and connector model, which allows for modular design and reusability of components. This approach to modeling provides a high level of abstraction, making it easier to understand and manage the complexity of automotive systems. EAST-ADL also provides support for modeling the interactions between components and connectors, as well as the functional and non-functional requirements of these systems.
One of the main benefits of EAST-ADL is its ability to support early-stage design activities, such as system architecture and requirements modeling. This can lead to a reduction in development time and cost. Additionally, the language has been designed to be compatible with other modeling languages, such as AADL and SysML. This makes it easier to integrate EAST-ADL models into existing development processes.
EAST-ADL has been used in a number of automotive development projects, including the development of electric and hybrid powertrains, advanced driver assistance systems (ADAS), and autonomous vehicles.

2.2 AUTOSAR

AUTOSAR (Automotive Open System Architecture) is a global development partnership of automotive industry companies and suppliers, which aims to develop and establish an open and standardized software architecture for automotive electronic control units (ECUs) [19].
The AUTOSAR standard defines a software architecture for ECUs that can be used across different automotive domains, such as powertrain, chassis, and infotainment. It provides a common language and framework for the development of automotive software, which can help to reduce development time and costs, as well as improve software quality and maintainability.
The AUTOSAR architecture is based on a layered approach, with each layer representing a different level of abstraction. The layers include the application layer, the basic software layer, and the runtime environment layer. The application layer contains the software components that implement the functionality of the ECU, while the basic software layer provides the basic software services that the application layer uses. The runtime environment layer provides the runtime environment for the ECU, including the operating system and communication services.
One of the key benefits of AUTOSAR is its ability to support the development of complex distributed systems. It provides a standardized communication protocol, known as the AUTOSAR communication stack, which enables ECUs to communicate with each other over a variety of communication buses, such as CAN, LIN, and Ethernet. This can help to simplify the development of distributed systems and reduce the risk of errors and compatibility issues.
In addition to the communication stack, AUTOSAR also provides a range of other software components and services, such as diagnostic services, calibration services, and security services. These components and services can help to improve the functionality, performance, and security of automotive software.
AUTOSAR is a widely adopted standard in the automotive industry, and is supported by a large number of companies and organizations. It is used in a wide range of automotive applications, from powertrain control to infotainment systems. The standard is continually evolving, with new releases being published on a regular basis [19].

2.3 MetaEdit+

MetaEdit+ is a powerful tool developed by Metacase in cooperation with the University of Jyväskylä for creating and maintaining domain-specific modeling languages (DSMLs) and code generators. It was developed by MetaCase, a Finnish software company, specialized in model-driven engineering (MDE) tools and services [35].
MetaEdit+ provides a graphical modeling environment where users can define their own DSMLs using a simple and intuitive notation. This notation is based on the concept of metamodeling, which is a technique for defining the structure and behavior of a modeling language. Metamodeling allows users to define their own language concepts and syntax, as well as the semantics of their language.
One of the main benefits of MetaEdit+ is its ability to generate code directly from DSML models. This code generation capability allows users to automate the generation of software code, eliminating the need for manual coding. This can lead to significant time and cost savings, as well as increased software quality.
MetaEdit+ also provides support for model transformations, which allow users to transform models from one language to another. This is particularly useful in situations where different stakeholders are using different modeling languages or tools. Model transformations can help to bridge the gap between different modeling languages, allowing stakeholders to share and exchange models more easily.
In addition to these features, MetaEdit+ provides a range of other tools and services for MDE. For example, it includes support for collaborative modeling, version control, and model debugging. It also provides a range of customization options, allowing users to tailor the tool to their specific needs and requirements.
MetaEdit+ has been used in a wide range of applications and domains, including telecommunications, automotive engineering, and software development. For example, it has been used to develop DSMLs for the design and development of telecom network protocols, automotive control systems, and software development processes.

2.4 ISO 21434

The ISO 21434 is an international standard that provides direction on the implementation of cybersecurity in road vehicles. It was developed by the International Organization for Standardization (ISO) in collaboration with the automotive industry, cybersecurity experts, and other stakeholders. The standard provides a framework for managing cybersecurity risks throughout the lifecycle of road vehicles, including design and development, production, operation, maintenance, and decommissioning [32].
ISO 21434 is based on a risk-based approach, which means that the level of cybersecurity protection required for a vehicle is determined by the level of risk associated with the vehicle. The standard provides guidance on how to assess the level of risk associated with a vehicle and how to select appropriate cybersecurity measures to mitigate those risks. It applies to all parties involved in the development, production, operation, and maintenance of road vehicles, including manufacturers, suppliers, and service providers.
The standard is designed to address the growing cybersecurity risks in the automotive industry. With the increasing use of connected and autonomous vehicles, the risk of cyber-attacks is becoming a major concern. The standard provides guidance on how to integrate cybersecurity into the overall development process for road vehicles, including the design of systems, components, and software [32].
The guidance provided in the ISO 21434 can help organizations in the automotive industry to increase the cybersecurity of their products and services, and to better manage cybersecurity risks. By implementing the guidance provided in the standard, organizations can reduce the

likelihood of cyber-attacks, protect their customers' privacy and safety, and maintain the integrity of their products and services.
In summary, the ISO 21434 is an international standard that provides guidance on the implementation of cybersecurity in road vehicles, with a risk-based approach to determine the level of cybersecurity protection required. The standard applies to all parties involved in the development, production, operation, and maintenance of road vehicles, and is designed to help organizations in the automotive industry to better manage cybersecurity risks and increase the cybersecurity of their products and services.

2.5 Security

According to the International Organization for Standardization (ISO), security is defined as "preservation of confidentiality, integrity and availability of information by applying a risk management process and giving assurance that the information and information processing systems continue to operate correctly in the face of various threats." [8]
The increasing use of connected and autonomous vehicles has highlighted the need for effective security measures to protect against cyber threats. As vehicles become more connected and reliant on electronic systems, the potential risks of cyber-attacks on these systems become greater. In order to address these risks, the automotive industry has developed a range of security measures and standards.
One of the key challenges of automotive security is the complexity of modern vehicles. Vehicles today include a range of electronic systems, including engine control units, infotainment systems, and advanced driver assistance systems. These systems communicate with each other and with external networks, creating a large attack surface for cyber criminals. As a result, security measures must be integrated throughout the vehicle design and development process [50].
The automotive industry has developed a range of security measures to address these risks, including secure communication protocols, firewalls, and intrusion detection systems. These measures are designed to protect against a range of cyber threats, such as malware, denial-of-service attacks, and unauthorized access.
In addition to these measures, there are also a number of industry standards that address automotive security. These include the ISO 21434 [32] standard for cybersecurity of road vehicles, which provides a framework for managing and mitigating cybersecurity risks throughout the entire vehicle lifecycle, and the SAE J3061 standard for cybersecurity engineering [11], which provides guidance on integrating cybersecurity into the vehicle development process.
Another important aspect of automotive security is the need for collaboration and information sharing between different stakeholders in the industry. The Automotive Information Sharing and Analysis Center (Auto-ISAC) was established to promote collaboration and sharing of cybersecurity information between automotive industry stakeholders, including manufacturers, suppliers, and government agencies.
In addition to these measures and standards, there is also a growing focus on the importance of cybersecurity training and awareness in the automotive industry. As cyber threats continue to evolve, it is important that all stakeholders in the industry, from engineers to executives, have a solid understanding of cybersecurity risks and best practices.

2.6 Security Scoring Systems

In today's world of cybersecurity, it is critical to have a way to measure the severity of vulnerabilities and their potential impact on systems and networks. Scoring systems provide a standardized method for assessing the severity of vulnerabilities, allowing organizations to

prioritize their response efforts and allocate resources accordingly.
One of the most widely used scoring systems is the Common Vulnerability Scoring System (CVSS), developed by the Forum of Incident Response and Security Teams (FIRST) [16]. CVSS provides a standardized method for assessing the severity of vulnerabilities based on a range of factors, including the ease of exploitation, the potential impact on systems and networks, and the level of required privileges.
In addition to these commonly used scoring systems, the recently developed ISO 21434 [32] standard includes an Impact Rating system, which provides a means of assessing the potential impact of a security threat on the safety of a vehicle. The Impact Rating system takes into account the severity of the threat, as well as the likelihood of it being realized and the potential consequences of a successful attack.
Overall, scoring systems such as CVSS and the Impact Rating from ISO 21434 play a critical role in modern cybersecurity by providing a standardized method for assessing the severity of vulnerabilities. By prioritizing response efforts and allocating resources accordingly, organizations can better protect their systems and networks against potential threats.

2.6.1 CVSS

The Common Vulnerability Scoring System (CVSS) is a framework for rating the severity of security vulnerabilities in computer systems. It was developed by the Forum of Incident Response and Security Teams (FIRST) to provide a standardized, open methodology for assessing the impact of security vulnerabilities. The CVSS framework assigns a score to each vulnerability based on its potential impact on confidentiality, integrity, and availability, as well as other factors such as exploitability and remediation level. The score ranges from 0 to 10, with higher scores indicating more severe vulnerabilities [16]. The CVSS is widely used in the industry to prioritize security vulnerabilities and to assist in the allocation of resources for vulnerability management. The CVSS score is calculated based on a set of metrics, which include the Base Score, Temporal Score, and Environmental Score. The Base Score provides a measure of the intrinsic severity of a vulnerability, while the Temporal Score takes into account the current state of the vulnerability, such as the availability of a patch. The Environmental Score considers the unique characteristics of an organization's systems and networks, such as the presence of mitigating controls. Since the Environmental Score isn't necessary for this thesis it is not listed below.

Algorithm 1 Base Score

```
If (Impact sub score <= 0) 0 else,
Scope Unchanged[4] Round up (Minimum [(Impact + Exploitability), 10])
Scope Changed Round up (Minimum [1.08 × (Impact + Exploitability), 10])

and the Impact sub score (ISC) is defined as,

Scope Unchanged 6.42 × $ISC_{Base}$
Scope Changed 7.52 × [$ISC_{Base}$-0.029] - 3.25 × [$ISC_{Base}$-0.02]$^{15}$

Where,

$ISC_{Base}$ = 1 - [(1-$Impact_{Conf}$) × (1-$Impact_{Integ}$) × (1-$Impact_{Avail}$)]
```

Algorithm 2 Temporal Score

Round up(BaseScore × ExploitCodeMaturity × RemediationLevel × ReportConfidence)

Attack feasibility rating	CVSS exploitability value
High	2.96-3.89
Medium	2.00-2.95
Low	1.06-1.99
Very low	0.12-1.05

Table 1: Example CVSS exploitability mapping.

2.6.2 ISO 21434 Scores

The ISO 21434 standard introduced three new scores for assessing the cybersecurity of automotive systems: the attack feasibility rating, the impact rating and the risk value determination[32].
The attack feasibility rating assesses the likelihood and ease of a successful cyber attack, considering factors such as the attacker's skills, resources, and motivation, as well as the system's vulnerabilities and defenses.
The impact rating evaluates the potential consequences of a successful cyber attack on a system or component, taking into account factors such as safety, financial impact, and operational disruption.
The risk value determination combines the impact and attack feasibility ratings to provide a quantitative measure of the overall risk posed by a particular cyber threat. It helps to prioritize cybersecurity measures and to communicate the risk to stakeholders.

Algorithm 3 Attack Feasibility Rating

$E = 8{,}22 \times V \times C \times P \times U$

where

E is the exploitability value;
V is the numerical value associated to the attack vector, ranging from 0,2 to 0,85;
C is the numerical value associated with the attack complexity, ranging from 0,44 to 0,77;
P is the numerical value associated with the privileges required, ranging from 0,27 to 0,85; and
U is the numerical value associated with user interaction, ranging from 0,62 to 0,85.

based on the numerical value it is possible to map the results in non-numerical values as seen in Table 1.

It is also possible to translate the attack feasibility and the impact into numerical values as seen in Table 2.

Algorithm 4 Impact Rating

The impact rating of a damage scenario shall be determined for each impact category to be one of the following:

- severe S3: i.e. life-threatening injuries (survival uncertain), fatal injuries;
- major S2: i.e. severe and life-threatening injuries (survival probable;
- moderate S1: i.e. light and moderate injuries; or
- negligible S0: i.e. no injuries

Attack feasibility rating	Numerical value F for attack feasibility
Very low	0
Low	1
Medium	1.5
High	2

Impact rating	Numerical value I for impact
Negligible	0
Moderate	1
Major	1.5
Severe	2

Table 2: Translation of attack feasibility rating and impact rating to numerical values.

Algorithm 5 Risk Value Determination

Based on the attack feasibility rating and the impact rating it is possible to determine the risk value as seen in Table 3.

		Attack feasibility rating			
		Very Low	Low	Medium	High
Impact rating	Severe	2	3	4	5
	Major	1	2	3	4
	Moderate	1	2	2	3
	Negligible	1	1	1	1

Table 3: Risk value determination matrix.

3 Problem Definition - Security by Design with SAM

The Security Abstraction Model SAM [6] is a specification of a modeling language for representing security-related properties in automotive software systems. This modeling language enables a security analysis of attack vectors in the automotive sector and allows for an in-depth risk analysis. With SAM both potential attacks and countermeasures against these attacks can be modeled. Furthermore, this allows the connection of security management and model-based systems engineering on an abstract description level according to the principles of automotive security modeling. SAM was defined based on security requirements from common industrial scenarios. It aims to be a solution for representing attack vectors on vehicles and provide a thorough security modeling for the automotive industry.
The Security Abstraction Model (SAM) is a modeling language that enables the integration of security-related information in system design. SAM provides a systematic way of specifying security-related requirements, policies, and mechanisms in the system architecture. The EAST-ADL, on the other hand, is an architectural modeling language used in the development of automotive systems.
SAM and EAST-ADL are connected through the concept of "Item". An Item in the EAST-ADL represents a functional or non-functional requirement of the system. SAM extends this concept by adding security properties to the Item, allowing the specification of security requirements and mechanisms that are necessary to satisfy the functional and non-functional requirements of the system.
In essence, SAM provides a way to integrate security considerations into the system architecture by defining security properties for Items in the EAST-ADL model. This enables the development of secure and reliable automotive systems that meet the required functional and non-functional requirements while also addressing potential security threats [4]. Item refers to a number of features of an automotive system. SAM tries to present all important criteria of the attack vectors, from the adversary's motivation up to the security breach. This allows a system to be represented from a security perspective in the early software development phase. In addition to the attack motivations, SAM also describes all intrinsic and temporal characteristics of an attack, e.g., effects on the security objectives (confidentiality, availability, integrity, etc.), the complexity of the attack, the affected object and the vulnerability. In the latest version, SAM can now also model social engineering attacks [7]. SAM acts as an extension to the EAST-ADL, because the EAST-ADL addresses relevant aspects of automotive systems (Being a major requirement for security modeling that is not offered by languages like SysML [51] or AADL [1], which only offers feature modeling); especially the features of a vehicle of any kind. In addition, the EAST-ADL speaks directly about functional safety and ISO 26262 in its Dependability Model. SAM identifies Items, Requirements as well as Hazards from architecture and dependability modeling and relates them to Attacks and Security Concepts.
Although SAM is developed as part of the EAST-ADL, it is not necessarily bound to EAST-ADL. SAM as a metamodel is independent of other languages but for connectivity links to 'Item' and 'Requirement' of the EAST-ADL. In addition, SAM can also be used independently of the rest of the system model in order to provide an overview of safety critical system parts before or at the beginning of the system engineering process. Models created according to SAM permit calculating a vulnerability score based on the Common Vulnerability Scoring System [15]. This scoring system allows a qualitative representation (such as low, medium, high and critical) of the severity of an attack and thus enables prioritization in the vulnerability management process. First attempts were to implement a generator that transfers the model data to an online tool. However, since this would have required a longer modeling time due to the transfer to the online tool and a permanent internet connection, this idea was rejected. In the current version, the CVSS calculator is integrated directly into the SAM modeling tool

MetaEdit+. The advantage of this is that no internet connection is required and the results can be viewed in real time next to the rest of SAM models. During the integration, same the color scheme of the CVSS was used. In this way, other analysis tools can also be integrated [7]. In the next development phase, a report system for safety-critical components of the system based on ISO 21434 is to be created.

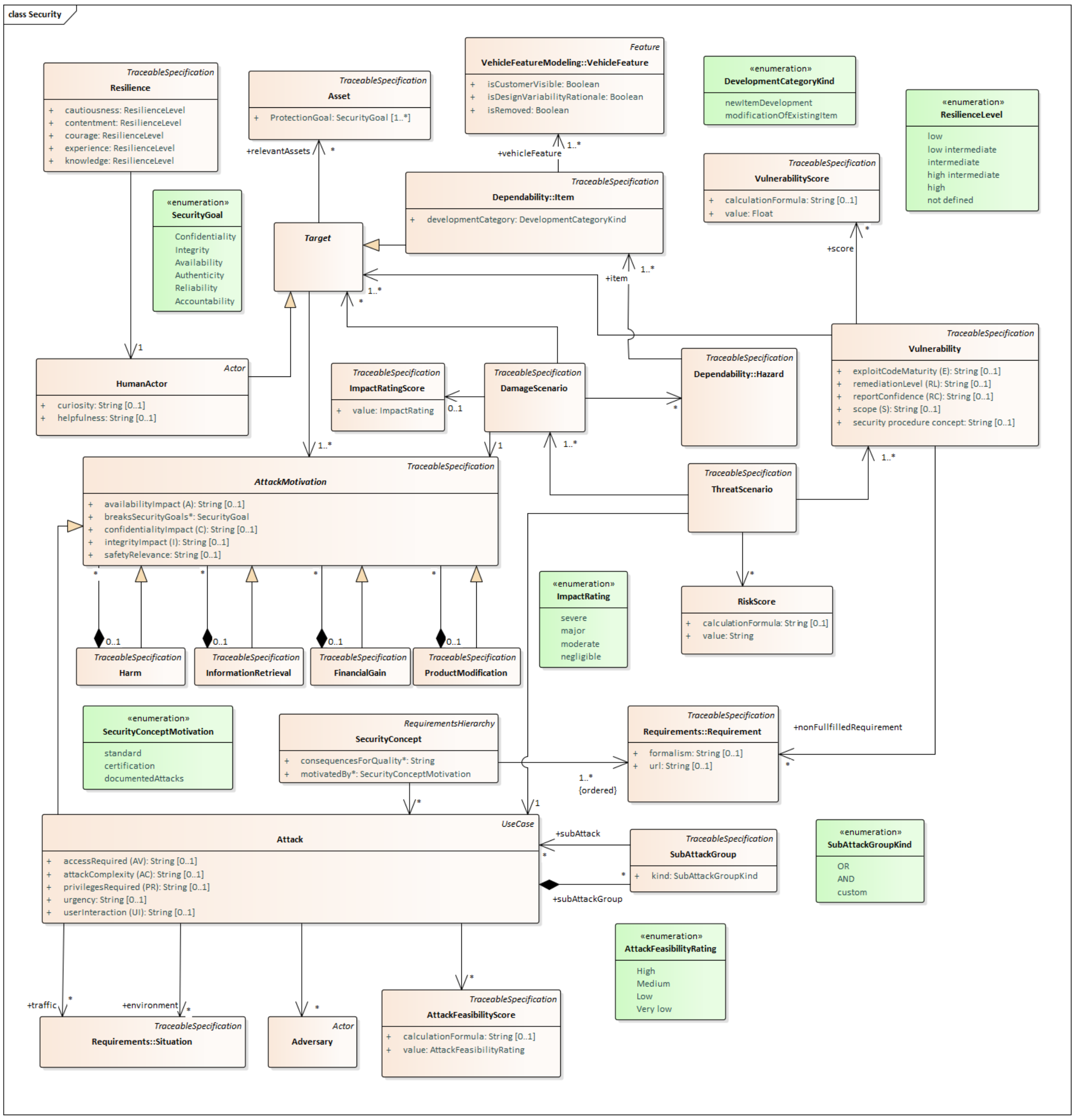


Figure 1: The SAM meta model. Higher resolution found at https://www.in.th-nuernberg.de/professors/BerglerMa/SAM

4 Research Questions

Security is a critical concern in the automotive industry, as the increasing reliance on electronic systems and connectivity expose vehicles to new and evolving security threats. In recent years, there has been a growing interest in developing security models and standards to address these threats. The Security Abstraction Model (SAM) is one such model that provides a systematic way of specifying security-related requirements, policies, and mechanisms in the system architecture of automotive software systems.
However, there are still some important questions that need to be addressed in order to fully leverage the capabilities of SAM and ensure that it remains relevant and effective in addressing the evolving security threats in the automotive industry.
One important area of focus is social engineering attacks. While SAM has been enhanced to address these types of attacks, there is still a need to explore what modeling support needs to be added to SAM to enable comprehensive modeling of social engineering attacks, countermeasures, and their relationships to the actor and the rest of the automotive system model. Such modeling support would be critical in developing effective countermeasures against these types of attacks.
Another important consideration is how to increase the applicability of the SAM model for practitioners. While SAM provides a systematic approach to security modeling, it is important that the model is easy to use and practical for practitioners in the automotive industry. Therefore, research should be conducted to explore how the SAM model can be made more user-friendly and how it can be integrated into existing development processes.
Finally, there is a need to explore the benefits and challenges of integrating the SAM model with a focus on cyber security into the ISO 21434 standard for automotive cybersecurity. While this integration would help to improve the overall security posture of automotive systems, there are likely to be challenges in implementing the SAM model within the context of the ISO 21434 standard. Therefore, it is important to explore the benefits and challenges of this integration and develop strategies for addressing any challenges that arise.
The SAM model is a valuable tool for addressing security threats in the automotive industry, but there are still important questions that need to be addressed in order to fully leverage the capabilities of this model. Based on the previous considerations, the following research questions arise:

- RQ1: What modeling support needs to be added to SAM in order to enable comprehensive modeling of social engineering attacks, countermeasures, and their relationships to the actor and the rest of the automotive system model?
- RQ2: How to increase the applicability of the Security Abstraction Model for the practitioner?
- RQ3: What are the benefits and challenges of integrating the Security Abstraction Model (SAM) with a focus on cyber security into the ISO 21434 standard for automotive cybersecurity, and how can this integration improve the overall security posture of automotive systems?

5 Research Methods and Process

In this chapter, we present the methods utilized to achieve the research goals as seen in Figure 2. Initially, we describe the research process, followed by an explanation of the concrete methods used in this thesis. The research questions were formulated based on the current problem from Chapter 3 and open questions from research and industry in the field of vehicle safety and security of autonomous vehicles. Chapter 4 provides an overview of the research questions. The state-of-the-art and state-of-the-practice were then critically examined using formal research [60] and filtered to identify the most relevant techniques to solve the research questions. The next step involved defining solutions to various problems, designing and implementing them in the form of algorithms and new methods. Each solution was evaluated through a suitable study methodology, and the results were published as papers and reports. The research objectives were achieved when the results agreed with the expected outcome. If the results did not match, a new methodology was developed and tested based on the experience gained. Various research methods were utilized in achieving the research objectives. For instance, the "proof of concepts" method [27] was used to demonstrate that the solution concept works for the first research objective. On the other hand, the "proof by demonstration" method [27] was employed for research goals 3 and 4, where we presented solutions in the form of demonstrations to potential users. Additionally, since the solutions might change based on feedback from practitioners, we need to consider this during the development process.

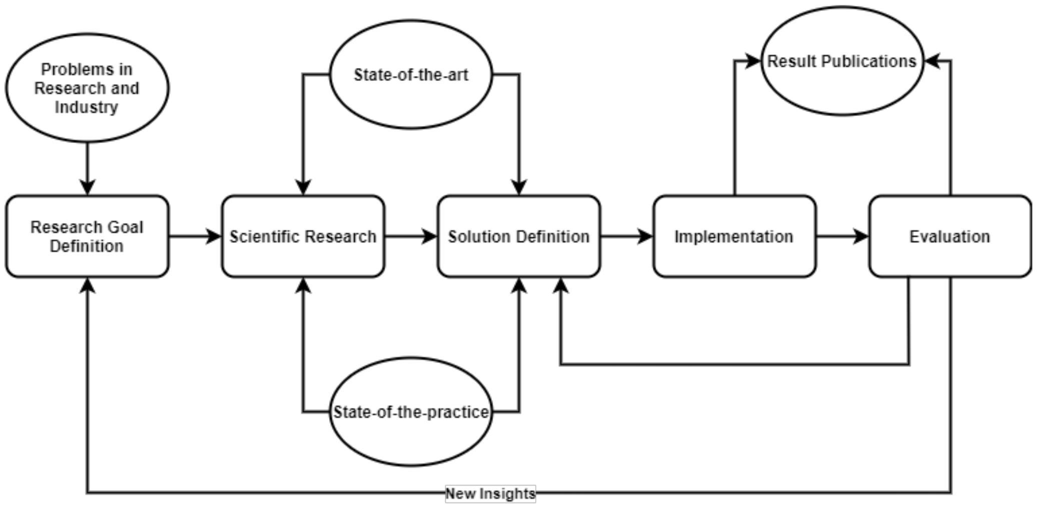


Figure 2: Research Process

6 Contributions

The Security Abstraction Model (SAM) is a well-established framework for modeling the security requirements of automotive software systems. However, SAM was not originally designed to model social engineering attacks, which can be a significant threat to the cybersecurity of automotive systems. To address this gap, we have developed an extension to SAM that enables the modeling of social engineering attacks and their relationships to the actors and the rest of the automotive system model. This modeling support enables a more comprehensive understanding of the security requirements of automotive software systems, and it can help developers to better anticipate and prevent social engineering attacks.
In addition to the modeling support added to SAM, we have also integrated the enhanced model with Meta-Edit+, a modeling tool that provides support for the creation, modification, and analysis of SAM models. This integration has enabled us to provide tool support for the enhanced SAM model, making it easier for developers to use and apply the new modeling features. With Meta-Edit+ support, developers can more easily create and modify SAM models that include social engineering attacks, and they can analyze the models to identify potential security weaknesses.
Finally, we have also integrated the ISO 21434 standard for automotive cybersecurity into SAM. This integration allows SAM to model the security requirements of automotive systems in a way that aligns with the ISO 21434 standard. By incorporating the ISO 21434 standard, SAM can provide guidance on the implementation of cybersecurity in road vehicles, with a risk-based approach to determine the level of cybersecurity protection required.

6.1 Research Question 1: Social-Engineering Attacks

Although vehicles cannot become direct victims of such attacks, can be impacted if the attacks are successfully carried out on employees involved in vehicle development or on private car owners. Social engineering attacks can provide attackers with access to individual vehicle parts or even the entire vehicle, and in many cases, these attacks can only be discovered but not prevented. For example, a quid pro quo attack, in which a victim provides information or access in exchange for other services, can be difficult to prevent.
Private car owners are a popular target for social engineering attacks due to the different motivations behind such attacks. An attacker may attempt to take control of the vehicle by tricking the owner into installing malicious software or hardware in the vehicle, as described in [12]. Alternatively, an attacker may seek valuable information about the driver that can be used for further social engineering attacks on targets related to the victim, such as the victim's employer. An attack on a vehicle owner may involve spying on the victim's vehicle type, contacting the victim with the identity of a service employee, and trying to obtain information about the vehicle and its usage behavior in a service conversation. The attacker then contacts the victim with the identity of a service employee and tries to find out more about the vehicle, the infotainment system and its usage behavior in a service conversation. He then offers the victim a free security update via CD, USB stick or mobile phone app, which the victim can download from a fake website or receive by post. This update actually installs malicious software that enables the attacker to read the victim's mobile phone data or to access the microphone of the hands-free system in order to record conversations and send them to the attacker via the mobile data connection of the mobile phone or vehicle. Alternatively, software can also be installed that gives the attacker control of the vehicle as in [21]. This example is modeled in Figure 3.
To prevent such attacks, possible points of attack via social engineering attacks must be identified and taken into account during the development of the components. With the current version of SAM, it is now possible to map social engineering attacks and develop a counter-

strategy. This Chapter describes how the original metamodel in SAM [62] was adapted to enable the representation of social engineering attacks. This was achieved by introducing a new abstract class called "Target," which generalizes the Item class already known in SAM and the new HumanActor class required for social engineering attacks. The HumanActor class has two attributes of the String type, which represent the exploitable human properties of curiosity and helpfulness. The use of these properties is measured by the ResilienceLevel type, which ranges from not defined (X), none (N), low (L), to high (H). An association of the Resilience class refers to the HumanActor class and represents the mental resistance to social engineering attacks. The Resilience class possesses attributes such as cautiousness, contentment, courage, experience, and knowledge, which correspond to the ResilienceLevel type and measure the extent of the property required to defend against the attack.
The Chapter concludes by emphasizing the need for adequate security concepts to counter social engineering attacks, particularly as the resilience of the victim plays a crucial role in the success of such attacks. By identifying the possible points of attack and developing counter-strategies using SAM, vehicle developers can better protect against social engineering attacks on vehicles. The Common Vulnerability Scoring System (CVSS) is not relevant for the assessment, as the values in the Resilience class show an assessment of the severity of the attack but an extended scoring system is under development.

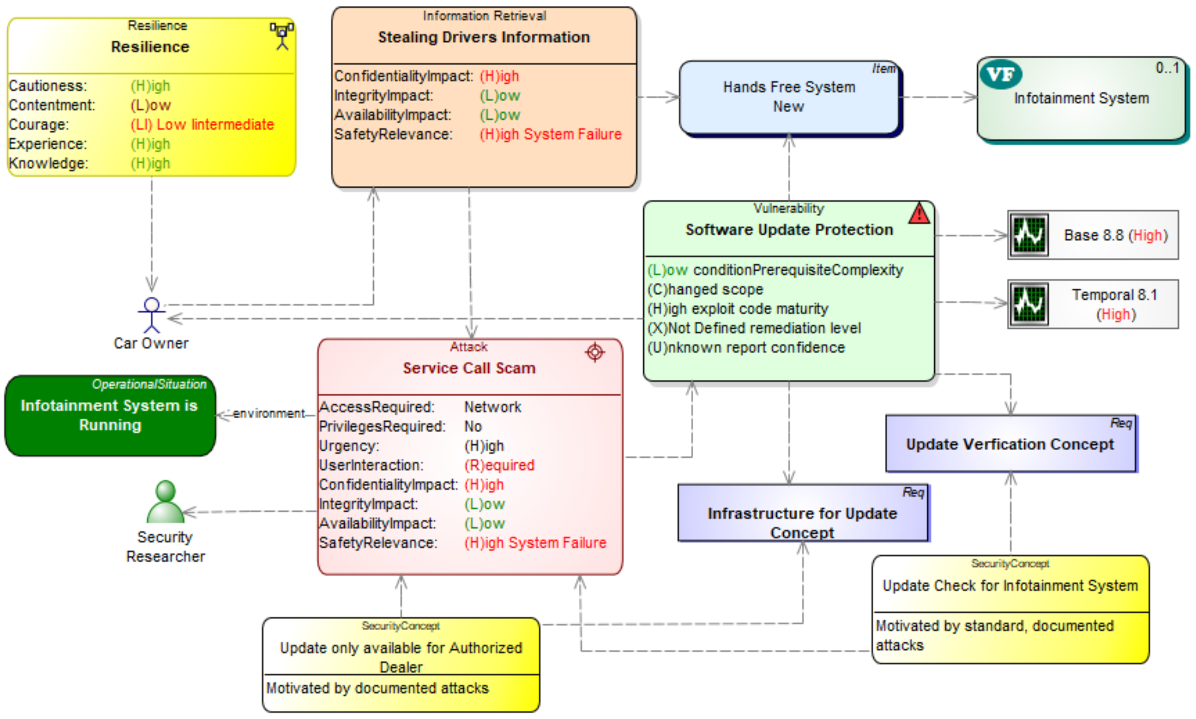

Resilience
Resilience
Cautioness: (H)igh
Contentment: (L)ow
Courage: (Ll) Low Intermediate
Experience: (H)igh
Knowledge: (H)igh
Information Retrieval
Stealing Drivers Information
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Item
Hands Free System
New
VF
0..1
Infotainment System
Vulnerability
Software Update Protection
(L)ow conditionPrerequisiteComplexity
(C)hanged scope
(H)igh exploit code maturity
(X)Not Defined remediation level
(U)nknown report confidence
Base 8.8 (High)
Temporal 8.1 (High)
Car Owner
OperationalSituation
Infotainment System is Running
environment
Attack
Service Call Scam
AccessRequired: Network
PrivilegesRequired: No
Urgency: (H)igh
UserInteraction: (R)equired
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Security Researcher
Req
Update Verfication Concept
Req
Infrastructure for Update Concept
SecurityConcept
Update Check for Infotainment System
Motivated by standard, documented attacks
SecurityConcept
Update only available for Authorized Dealer
Motivated by documented attacks

Figure 3: SAM model of Service Call Scam.

6.2 Research Question 2: Model Integration for Applicability Increase

Although it is possible to create tooling for the Security Abstraction Model (SAM) from scratch, MetaEdit+ [35], a commercial language workbench, was used to automate most of the required functionality. This approach only required defining the parts specific to SAM, such as its modeling concepts, rules, notation, and integration with other tools. Using MetaEdit+ not only speeds up implementation, but also enables easy evolution when the modeling needs change. Moreover, MetaEdit+ offers implementation of EAST-ADL [36] and other relevant functionality needed for automotive system development, such as collaborative modeling, ver-

sion control, integration with relevant tools applied in automotive, and availability of supporting services. However, some parts related to tooling, such as showing elements of SAM in the user interface and deciding how to indicate if constraints are not followed, deserved attention. Constraints that are considered mandatory are checked and reported at modeling time, while those not sensible to check, such as minimum cardinalities in the metamodel, are shown as recommendations in the live check pane at the bottom of the diagram. This way language users get immediate guidance to create security models.
SAM, like EAST-ADL, originally focused on language concepts and defining the exchange format via a metamodel. Defining the whole language required covering concrete syntax, all constraints, language usability topics, as well as integration with other tools. Half of the effort was on defining the metamodel, constraints, and notation, and almost the second half on implementing ways to calculate vulnerability scores. Feedback loops from two SAM knowledgeable individuals during the implementation phase led to revising the implementation and parts of the SAM metamodel. The tooling was tested and verified by using SAM to specify various kinds of systems and by comparing them with the reference test cases. The implementation started by integrating SAM in the existing metamodel of EAST-ADL, following the same conventions, such as security models having the same naming policies, all model elements having a globally unique identifier (UUID), and SAM following the same package structure as EAST-ADL uses to organize specifications. SAM concepts were also integrated with already existing EAST-ADL concepts, like Item from Dependability and ISO26262, VehicleFeature from variation models addressing product lines, and Requirements from specifying and tracing with system requirements.
MetaEdit+ allows to specify metamodels graphically [34] similarly to UML, SysML, EAST-ADL or AUTOSAR as well as SAM.This allowed us to integrate all the language constraints, notations, model checking, and generators in one place, which would have been specified separately and resulted in inconsistencies and low quality [58, 5]. By using MetaEdit+, we achieved a tight integration of the whole language definition, which significantly improved the language's quality. Figure 4 depicts the concepts of SAM, as defined in MetaEdit+. The list of Objects illustrates the key modeling elements of SAM, the list of Relationships shows the connections between these elements, and the list of Roles demonstrates how an object participates in relationships by being directed or undirected, having constraints or detailed properties.

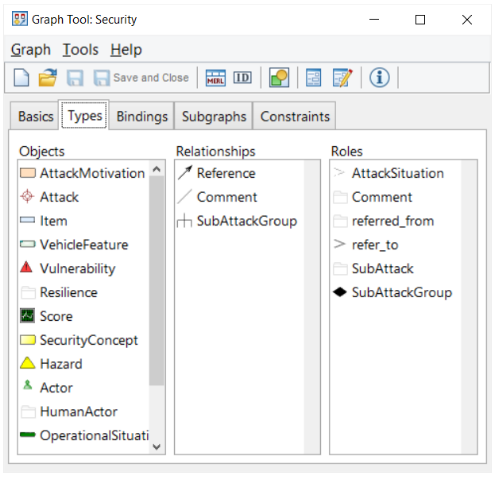
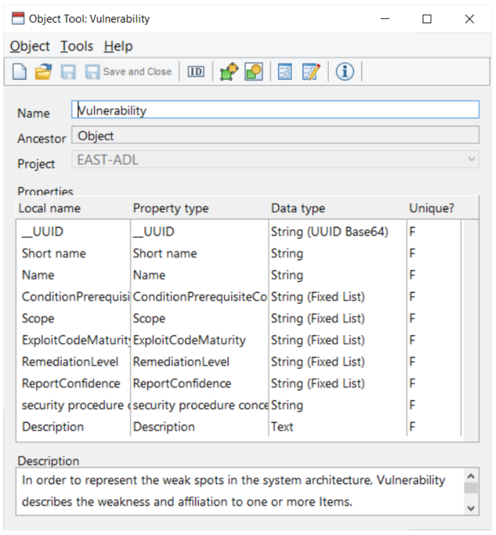


Figure 4: Defined language concepts of SAM.

To minimize the modeling effort the implementation defines one AttackMotivation and its concrete subtype is selected from a property. This way the type of AttackMotivation (Harm, Financial Gain etc.) can be changed without deleting the old one and creating and re-connecting a new one. This definition, compared to having a language construct for each subtype, is possible because all subtypes have the same properties and constraints. Also, for the reference from Attack to OperationalSituation, the role AttackSituation has a property to select if based on Traffic or Environment. Each element of SAM shown in Figure 4 are defined with further details. Figure 5 shows one such definition: The Vulnerability and its nine properties.



Name: Vulnerability

Ancestor: Object

Project: EAST-ADL

Properties

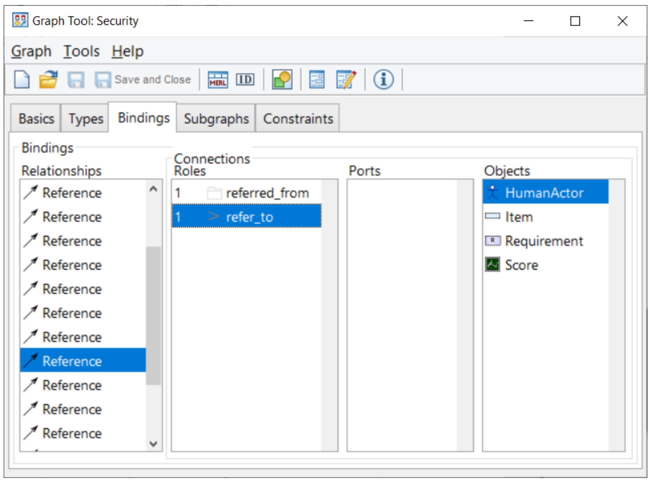
Local name	Property type	Data type	Unique?
__UUID	__UUID	String (UUID Base64)	F
Short name	Short name	String	F
Name	Name	String	F
ConditionPrerequisi	ConditionPrerequisiteCo	String (Fixed List)	F
Scope	Scope	String (Fixed List)	F
ExploitCodeMaturit	ExploitCodeMaturity	String (Fixed List)	F
RemediationLevel	RemediationLevel	String (Fixed List)	F
ReportConfidence	ReportConfidence	String (Fixed List)	F
security procedure (	security procedure conce	String	F
Description	Description	Text	F

Description

In order to represent the weak spots in the system architecture, Vulnerability describes the weakness and affiliation to one or more Items.

Figure 5: Definition of Vulnerability.

The first three properties are derived from the EAST-ADL and AUTOSAR metamodels, while the remaining ones are specific to SAM. These properties are subject to rules and constraints, such as the mandatory requirement for the "Short name" property to start with an alphabetical character, followed by possible characters, numbers or underscores, and the constraint of the "Scope of Vulnerability" property to have only two possible values (unchanged or changed). To enhance usability, we reordered the properties to match the order in which security engineers are expected to fill them, following the same order as the vulnerability analysis tool Common Vulnerability Scoring System CVSS [16]. An example of how the Vulnerability concept is used is illustrated in the modeled example of an social-engineering attack as seen in Figure 3 before. The definition of the Vulnerability modeling concept was completed by providing a description of the concept in SAM, which is available in the help system. Constraints in SAM can be applied either as part of the metamodel or via a model checker. For instance, rules on legal connections and uniqueness of element names can be checked and enforced during modeling time. The uniqueness of model elements is ensured by defining a constraint for each element. Figure 7 provides an illustration of the uniqueness constraint for the name of Vulnerability.



Bindings

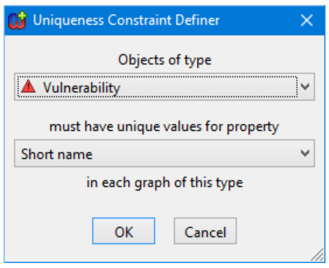
Relationships: Reference, Reference, Reference, Reference, Reference, Reference, Reference, Reference, Reference, Reference, Reference

Connections — Roles: 1 referred_from; 1 refer_to

Ports

Objects: HumanActor, Item, Requirement, Score

Figure 6: Legal references from Vulnerability.



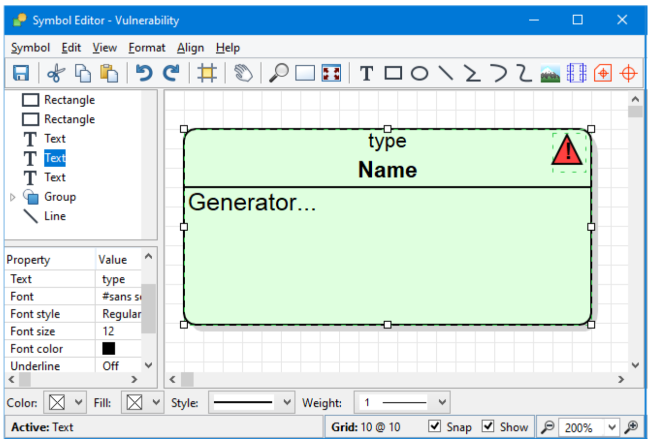
Objects of type Vulnerability must have unique values for property Short name in each graph of this type

Figure 7: Constraint for uniqueness of Vulnerability.

Examples of the latter kind of rules are those on completeness. For these kinds of rules model checking and guidance is provided for language users while it is applied. Implemented tool support for SAM covered the following constraints:

- Attacks and AttackMotivations may form a tree and cyclic structures are not allowed.
- Attack must refer to at least one Vulnerability.
- Vulnerability must refer to at least one Attack.
- Item must refer to at least one VehicleFeature.
- Item or HumanActor must refer to at least one Vulnerability.
- Item or HumanActor must refer to at least one AttackMotivation.
- Hazard must refer to at least one Item.
- SecurityConcept must refer to at least one Requirement.
- SecurityConcepts motivated by documentAttack must refer to an Attack.
- Requirement must be related to a SecurityConcept.
- Score must be connected to one Vulnerability.
- Resilience must refer to at least one HumanActor.

For the notation we applied a similar style as was already applied in EAST-ADL. In particular, we used various visual variables for the notation, such as shapes, colors and fonts, as guided by [38] improving readability, understanding and working with security models. Figure 8 illustrates the notation for the Vulnerability element. The notation is defined with the Symbol Editor of MetaEdit+.



Property	Value
Text	type
Font	#sans s
Font style	Regular
Font size	12
Font color	■
Underline	Off

Active: Text

Grid: 10 @ 10

Figure 8: Symbol definition for Vulnerability.

Alternatively, existing visualizations could be imported and applied. Initially the notation provided just the basics: A light green rounded rectangle showing the unique name of Vulnerability followed by its properties, like Remediation level. To differentiate better the various notational elements, the symbol definitions were extended with a visual clue on the upper right corner: See e.g., Software Update Protection vulnerability in Service Call Scam attack (Figure 3). SAM models can be utilized to analyze and compute vulnerability scores. To enable this, a generator was implemented to export data from models to the Common Vulnerability Scoring System (CVSS), which provides a numerical score indicating the severity of a vulnerability. This score can be translated into a qualitative representation, such as low, medium, high, and critical, to assist organizations in prioritizing their vulnerability management processes. Additionally, the integration of SAM with other analysis tools is feasible, provided that they offer programmable APIs or similar mechanisms, and depending on their capabilities, the results can be included back to the model. To illustrate, MetaEdit+ can display the score directly in the Vulnerability model element, making the information accessible when tracing from security properties to requirements, features, and system design in general. To avoid the slow modeling and score calculation process that results from using an external web-based calculator, we implemented the CVSS calculator into the SAM modeling tool. This was achieved by using the same generator system that produced vulnerability vectors to feed the existing CVSS calculator, which allowed vulnerability scores to be calculated in real-time during modeling. The Score element was employed to display the results, with the color schemes of CVSS used for the notation of Score. Figure 3 demonstrates the CVSS calculation during modeling time and the display of results directly in the model. SAM was tested by applying it to security modeling, using known security examples as reference cases. During this testing phase, the SAM metamodel was refined for usability and when used for CVSS calculation. Since the metamodel of SAM was integrated with the metamodel of EAST-ADL, SAM was also directly applied as an extension of EAST-ADL (see Figure 10).

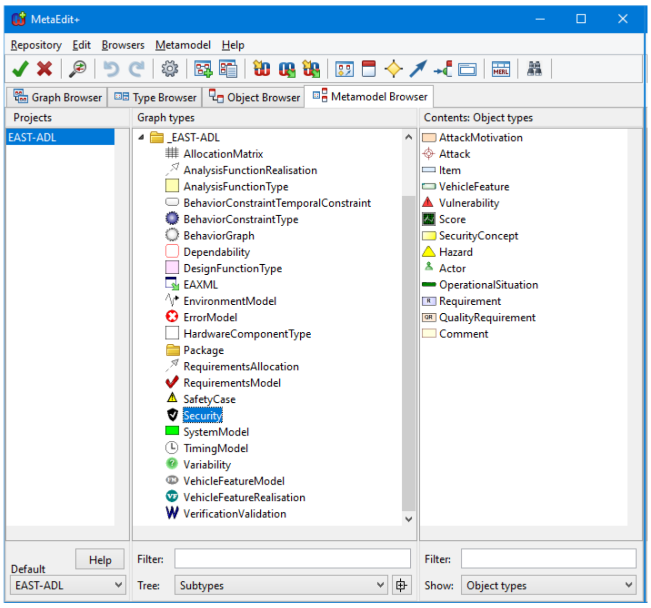


Figure 10: Security package as part of EAST-ADL (Security is selected and highlighted).

6.3 Research Question 3: Integrating ISO 21434 for Cybersecurity Threat Analysis

ISO 21434:2020 [32] is a standard that provides guidance for managing cybersecurity risks in the automotive industry, including vehicles with automation capabilities. With the rise of autonomous and semi-autonomous vehicles, the need for robust cybersecurity measures becomes increasingly critical. Vehicle automation relies heavily on software, connectivity, and data processing capabilities, which can be vulnerable to cyber threats. Therefore, integrating cybersecurity into vehicle automation systems is crucial to ensure the safety, security, and reliability of these vehicles.
ISO 21434 [32] provides a framework for incorporating cybersecurity into the development, production, operation, and maintenance of vehicles with automation capabilities. The standard emphasizes the need for a systematic approach to managing cybersecurity risks throughout the entire lifecycle of road vehicles, including those with automation capabilities. This involves identifying and assessing potential cybersecurity risks associated with vehicle automation, such as unauthorized access, data tampering, and remote manipulation of vehicle functions. It also requires developing appropriate mitigation measures to minimize the risk of cybersecurity threats impacting the safe operation of automated vehicles.
The standard provides guidelines for establishing secure development practices for automotive systems, including those related to vehicle automation. It emphasizes the importance of integrating cybersecurity into the entire development process, including secure coding practices, secure configuration management, and secure software supply chain management. This involves implementing secure coding standards and best practices specifically tailored for vehicle automation, securing communication channels between vehicle automation components, and ensuring the integrity and security of software components and updates used in automated systems.
ISO 21434 also highlights the need for rigorous testing and validation of automotive systems, including those related to vehicle automation. It provides guidelines for conducting vulnerability scanning, penetration testing, and security assessments to assess the security of automated systems. This involves testing the functionality and security of automated features, identifying potential vulnerabilities and weaknesses, and validating the effectiveness of cybersecurity mitigation measures in automated systems before deployment.

The standard emphasizes the importance of complying with relevant regulations and standards related to automotive cybersecurity, including those that apply to vehicle automation. This involves understanding and adhering to regulations and standards that specifically address cybersecurity in automated vehicles, such as UN Regulation No. 156 [55] Software Update Processes and Management Systems. Compliance with these regulations and standards can help ensure that automated vehicles meet the required cybersecurity requirements and operate securely and safely.
Finally, ISO 21434 underscores the need for comprehensive documentation and traceability of cybersecurity-related activities, including those associated with vehicle automation. This involves maintaining clear and traceable records of risk assessments, development practices, testing results, and compliance evidence specifically related to vehicle automation. Documentation and traceability are essential for audit and review purposes, as they provide evidence of compliance and accountability in ensuring the cybersecurity of automated vehicles.
The Security Abstraction Model (SAM) is a structured framework used by the automotive industry to manage cybersecurity risks in vehicles. It covers all stages of a vehicle's lifecycle, from design and development to production, operation, and maintenance. The emergence of ISO 21434, a standard focused on cybersecurity engineering in road vehicles, has made it crucial for the automotive industry to integrate ISO 21434 into SAM to ensure robust cybersecurity practices.
Integrating ISO 21434 into SAM offers several benefits, including standardization and consistency of cybersecurity practices across the organization, comprehensive risk management throughout the entire vehicle lifecycle, secure development practices at every stage, robust testing and validation processes, and compliance with industry regulations and standards related to automotive cybersecurity.
By incorporating the guidelines outlined in ISO 21434 into SAM, automakers can strengthen their cybersecurity posture and mitigate cybersecurity risks effectively in vehicles. The integration of ISO 21434 into SAM promotes a uniform approach to cybersecurity processes, methodologies, and documentation, making it easier to manage and assess cybersecurity risks consistently throughout a vehicle's lifecycle. It also helps automakers comply with industry regulations and standards related to automotive cybersecurity, which enhances the overall cybersecurity posture of the vehicles.
The integration of the ISO 21434 standard into SAM has changed the metamodel, and new items have been added to SAM to enable a more detailed and comprehensive risk analysis. The new items that have been added are Asset, Damage Scenario, Threat Scenario, ImpactRatingScore, RiskScore, AttackFeasibilityRating, and AttackFeasibilityScore. These items are illustrated in the example Figure 11, and the current meta-model can be viewed online [1].

[1]https://www.in.th-nuernberg.de/professors/BerglerMa/SAM/

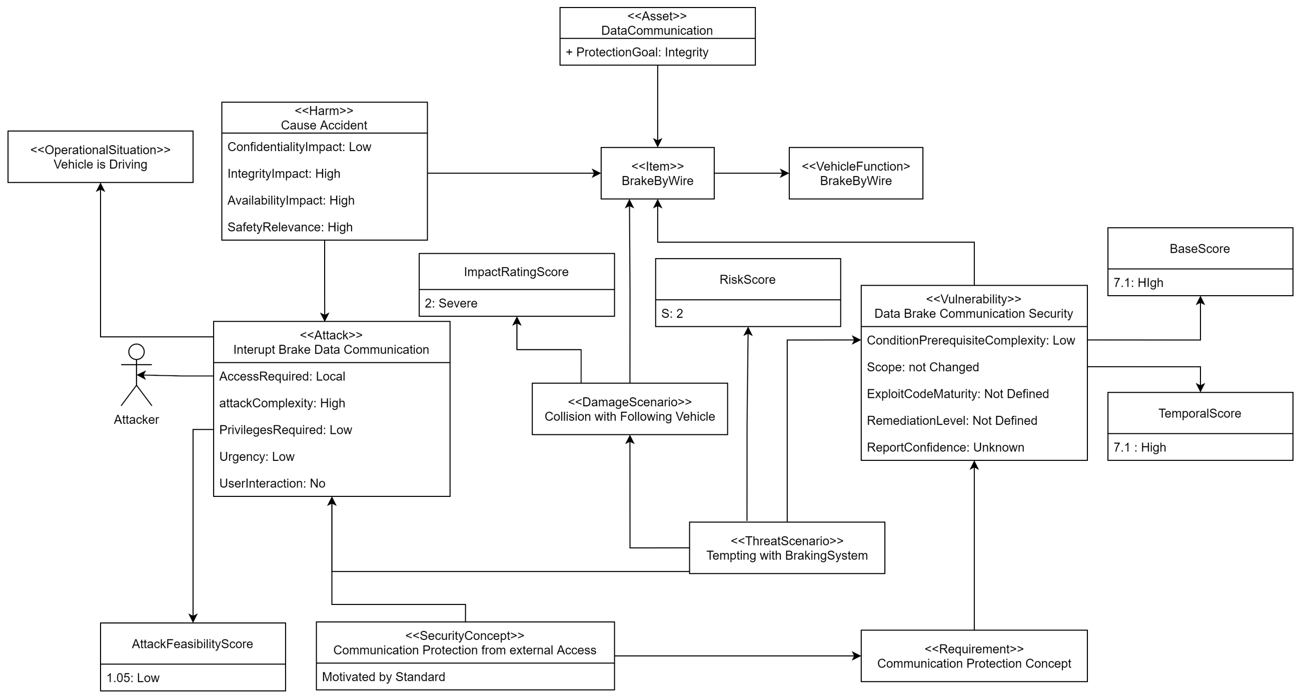

<<Asset>> DataCommunication
+ ProtectionGoal: Integrity
<<Harm>> Cause Accident
ConfidentialityImpact: Low
IntegrityImpact: High
AvailabilityImpact: High
SafetyRelevance: High
<<OperationalSituation>> Vehicle is Driving
<<Item>> BrakeByWire
<<VehicleFunction>> BrakeByWire
BaseScore
7.1: HIgh
ImpactRatingScore
2: Severe
RiskScore
S: 2
<<Vulnerability>> Data Brake Communication Security
ConditionPrerequisiteComplexity: Low
Scope: not Changed
ExploitCodeMaturity: Not Defined
RemediationLevel: Not Defined
ReportConfidence: Unknown
<<Attack>> Interupt Brake Data Communication
AccessRequired: Local
attackComplexity: High
PrivilegesRequired: Low
Urgency: Low
UserInteraction: No
Attacker
<<DamageScenario>> Collision with Following Vehicle
TemporalScore
7.1 : High
<<ThreatScenario>> Tempting with BrakingSystem
AttackFeasibilityScore
1.05: Low
<<SecurityConcept>> Communication Protection from external Access
Motivated by Standard
<<Requirement>> Communication Protection Concept

Figure 11: Security Model example of the newest version of SAM. The scenario describes a manipulation of the braking functionality.

The new item Asset complements the existing "Target"' in the metamodel, which can be both a "HumanActor" and an "Item" in the sense of ISO 26262. The Asset item refers to any entity that requires protection from cyber-attacks, such as a vehicle, a component of the vehicle, or any other element that is critical for the safe and secure operation of the vehicle.
According to ISO 21434, a damage scenario is a hypothetical event or sequence of events that could lead to harm to the vehicle, its occupants, or its surroundings [41]. It takes into account the potential sources of harm, the likelihood of the harm occurring, and the severity of the harm that could result. The purpose of defining damage scenarios is to identify the risks associated with the use of the vehicle and to establish measures to prevent or mitigate the effects of those risks.
With the introduction of the new item "DamageScenario," the consequences of a successful attack can now be modeled and additionally evaluated by the "ImpactRatingScore." The ImpactRatingScore assesses the impact of an attack scenario based on factors such as the severity of the damage caused, the extent of the damage, and the duration of the impact.
A threat scenario, according to ISO 21434, is a hypothetical situation or sequence of events that could lead to a security threat to a vehicle's functions, components, or data. It includes the potential sources of the threat, the probability of the threat occurring, and the severity of the consequences that could result. The goal of defining threat scenarios is to identify potential security risks and vulnerabilities and to establish measures to prevent or mitigate the effects of those risks.
By integrating the item "ThreatScenario," it is now possible to describe the attack scenario more precisely. In addition, in combination with the other newly introduced scores, an assessment of the risk for such a scenario can be given using the "RiskScore" item. The RiskScore item evaluates the overall risk associated with a given threat scenario based on the probability of the attack occurring and the potential consequences of the attack.
The already existing item "Attack" was supplemented by the "AttackFeasibilityScore," whereby the feasibility of an attack can be better assessed. Since the Common Vulnerability Scoring System (CVSS) is already integrated into SAM; this can be used as a basis for calculations.
Thanks to the successful integration of the ISO 21434 standard into SAM, it is now possible to

model and evaluate the following points with more accuracy and precision:

1. Item Definition
2. Asset Identification
3. Identification of Threat Scenarios
4. Impact Rating
5. Attack Path Analysis
6. Attack Feasibility Rating
7. Risk Value Determination
8. Risk Treatment Decision
9. Cybersecurity Concept

According to our research, it appears that SAM already addresses the two remaining topics of Cybersecurity Goals and Cybersecurity Claims. However, the standard's language regarding these points are not entirely clear, and additional research is needed to confirm this thesis with certainty.
Thanks to the successful integration of ISO 21434, it is not only possible to evaluate vulnerabilities, but also attacks and their impact on a system. For this purpose, new scores are introduced in the metamodel based on the ISO 21434 standard:

1. AttackFeasibilityScore: The AttackFeasibilityScore refers to the attack feasibility rating from the standard. This describes the feasibility of an attack on our system. The calculation basis for this is the already implemented CVSS score. According to CVSS, the ratings are mapped to numbers and used in the corresponding formula from the standard. The new formula is ($E = 8.22xVxCxPxU$) [41] where E is the exploitability value; V for the value of the attack vector; C for the attack complexity value; P for the value of the privileges required and U for the value of the user interaction. This value can then be mapped back to a textual evaluation based on the standard.

2. ImpactRatingScore: The ImpactRatingScore refers to the impact rating from ISO 21434. This value describes the severity of the consequences of a damage scenario. The impact rating can have the values Negligible (0), Moderate (1), Major (1.5) and Severe (2).

3. RiskScore: The RiskScore refers to the risk value determination from the standard. This value describes the risk that a threat scenario will occur. According to ISO 21434, this value can be determined either using a matrix or using your own calculation formulas. In both cases, the Impact Rating and the Attack Feasibility Rating are used. In our example, we use the risk matrix provided in the standard as seen in Table 3.

In our example (Figure 11), the following values result for the respective scores:

- BaseScore: 7.1 High
- TemporalScore: 7.1 High
- AttackFeasibilityScore: 1.05 Low ($E = 8.22x0.55x0.44x0.62x0.85$)
- ImpactRatingScore: 2: Severe (based on the definition in the standard)
- RiskScore: S: 2 (based on the evaluation matrix in the standard)

6.4 Summary

In summary, our contributions have been significant in enhancing the Security Abstraction Model (SAM) to model social engineering attacks, providing tool support through Meta-Edit+, and integrating the ISO 21434 standard for automotive cybersecurity. Our enhancements to SAM enable a more comprehensive understanding of the security requirements of automotive software systems and provide developers with the tools necessary to identify and prevent potential security weaknesses. By highlighting our contributions, we hope to provide a clear overview of the enhancements made to SAM and how they can improve the overall cybersecurity of automotive systems.

7 State of the Art

The importance of cybersecurity has grown significantly with the widespread use of technology in various domains. In recent years, the automotive industry has also been affected by this trend due to the increasing integration of digital components in vehicles. This has led to a greater need for IT-Security in Vehicles to prevent cyber-attacks and ensure the safety and reliability of modern vehicles. However, even with robust IT-Security measures in place, human vulnerabilities can still be exploited through Social-Engineering Attacks. Therefore, it is essential to educate employees and customers on cybersecurity best practices to prevent such attacks. In addition, Software Security Analysis Tools have been developed to assist in identifying and preventing software vulnerabilities during the development process. These tools provide automated analysis of software code to identify security flaws and can help ensure the safety and reliability of software used in vehicles. In this chapter, we will discuss the current state of the art in these three areas, including recent developments, challenges, and future directions.

7.1 IT-Security in Vehicles

The increase in the number of hacks on cars and other connected devices in recent years has raised the importance of security measures [54, 23, 20]. These security breaches are not just rare occurrences, as certain car models or their components, such as Toyota, Hyundai, Kia, and Volkswagen, have already been hacked [54, 23, 22]. Modern cars, with anti-theft systems, tire pressure monitoring systems, remote keys, Bluetooth, radios, and telematics access functionality, face numerous security challenges. Additionally, infotainment systems that offer access to third-party applications and the internet can be a potential target for social engineering attacks. As digitalization continues to advance, attackers can gain access to confidential data and smuggle malicious software into the development process [46]. Customer concerns and lawsuits on vulnerabilities push the automotive industry to prioritize the implementation of security measures [54]. However, developing and maintaining secure systems is not an easy task. Security cannot be an afterthought and should be designed from the very beginning of the system design [47]. The actual development of secure and trustworthy systems requires time, expertise from multiple fields and stakeholders, and needs to be integrated into other development processes and practices. To overcome these challenges, proper security design practices should be implemented to smoothly integrate security in the development processes of automotive companies. The increasing digitalization of vehicles resulted in modern vehicles becoming interconnected computer networks with advanced software components [3, 28, 59]. Autonomous vehicles will continue this trend towards more communication interfaces for reasons of functionality, safety, and comfort. Therefore, collective research efforts in the field of vehicle security are crucial for the protection of human lives [3, 28, 59].
The combination of state-of-the-art software components with legacy interfaces and hardware infrastructure decisions can be risky from an IT security perspective. Insecure and unencrypted protocols, such as Controller Area Network (CAN), were not designed in accordance with today's security principles, and secure automotive network architectures were not prioritized in the past due to the general prejudice of cars' security due to their technical complexity [51]. Development processes were sluggish, and the lack of standard guidelines and low societal pressure led to a rather slow transformation of automotive development processes taking the security-by-design principle systematically into consideration. Existing countermeasures against cyber-attacks, such as the use of message cryptography for encrypting, authenticating or randomizing vehicle-level network messages, focus mainly on concrete attacks and do not consider the complexity of the access options offered by modern vehicles [51].

Defining and enforcing security goals for the automotive system is necessary to improve overall security. The security goals that should be addressed include integrity, authenticity, confidentiality, reliability, availability, and accountability. Attack vectors, such as gaining remote control access to the vehicle using the OEMs cloud and/or mobile application's infrastructure, getting SecurityAccess via Unified Diagnostic Services (UDS), controlling the car via Onboard Diagnostic (OBD) injection, and remotely breaking into the telematics unit, can affect multiple security goals [44, 30, 43, 29, 26, 61, 18]. According to the SAE J 3601 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" [47], security should be considered throughout the entire development, production, and operation process of automotive systems. Techniques for threat analysis and risk assessment, threat modeling and vulnerability analysis, such as attack trees, table-based, use-case-based, and misuse case-based character, can be linked to requirements and design specifications in the early stages of development [47, 10, 31, 45, 33, 37].

7.2 Social-Engineering Attacks

Social engineering attacks are a type of cybersecurity attack that exploit human vulnerabilities to gain access to sensitive information or systems [48]. Unlike traditional attacks that rely on technical exploits, social engineering attacks manipulate human psychology and behavior to achieve their goals. Social engineering attacks can take many forms, including phishing, pretexting, baiting, and quid pro quo.
Phishing attacks are the most common form of social engineering attack. They typically involve the use of fraudulent emails or websites that trick users into disclosing sensitive information such as usernames, passwords, or financial data. Pretexting is another common tactic that involves creating a false pretext or scenario to obtain sensitive information or access to a system. Baiting involves the use of physical devices such as USB drives or CDs that contain malware or other malicious code. Quid pro quo attacks involve offering something in exchange for sensitive information or access [48].
To address the threat of social engineering attacks, various countermeasures and best practices have been developed [2, 9, 39]. These include security awareness training for employees, the use of multi-factor authentication, and the implementation of security controls such as firewalls and intrusion detection systems. In addition, security experts recommend that organizations adopt a "defense-in-depth" approach to cybersecurity, which involves implementing multiple layers of security controls to protect against a wide range of threats.
Even if the types of attack differ from one another, social engineering attacks always have the same pattern [40]:

1. Collect information about the target.
2. Develop relationship with the target.
3. Exploit the available information and execute the attack.
4. Exit with no traces.

Social engineering attacks can also be successful in the automotive industry, creating vulnerabilities for subsequent attacks [56]. Therefore, it is crucial to consider these weaknesses during the development of vehicle components.
Overall, social engineering attacks pose a significant threat to the security of automotive systems, and it is important for organizations to take proactive measures to prevent them. Ongoing research and development are needed to stay ahead of the evolving tactics and techniques used by attackers.

7.3 Software Security Analysis Tools

Software security analysis tools are designed to help developers identify and remediate security vulnerabilities in their code before it is deployed [52, 49]. Three popular examples of these tools are ThreatModeler, VectorCAST, and Code Sonar [53, 25, 57].
ThreatModeler is a tool that enables developers to model, analyze, and prioritize potential security threats in their software applications. By using ThreatModeler, developers can create a detailed and comprehensive threat model that takes into account the specific risks associated with their software, the assets that need protection, and the potential attackers. The tool provides an intuitive and user-friendly interface that allows developers to visualize and document potential threats, identify vulnerabilities, and prioritize remediation efforts. Additionally, ThreatModeler integrates with other security tools, such as SAST and DAST, to provide a complete and comprehensive security testing framework [53, 25].
VectorCast is a powerful software testing tool that helps identify and eliminate vulnerabilities in code by performing automated testing at various stages of the development process. VectorCast's static analysis engine can identify potential vulnerabilities in code by analyzing the source code and identifying potential issues such as buffer overflows, null pointer dereferences, and other common security issues. VectorCast also includes a dynamic testing engine that can simulate real-world conditions and identify vulnerabilities that may be missed by static analysis alone. Additionally, VectorCast can integrate with other security tools, such as SAST and DAST, to provide a complete and comprehensive security testing framework [57].
Code Sonar is a sophisticated static analysis tool that enables developers to identify and eliminate vulnerabilities in code by analyzing the codebase and identifying potential security issues. Code Sonar's analysis engine can identify a wide range of vulnerabilities, including buffer overflows, null pointer dereferences, and other common security issues. Additionally, Code Sonar includes a unique feature called the "taint tracker," which can track the flow of data through the codebase and identify potential security risks associated with the flow of sensitive data. Code Sonar also integrates with other security tools, such as SAST and DAST, to provide a complete and comprehensive security testing framework [57].
ThreatModeler, VectorCast, and Code Sonar are all powerful software security analysis tools, each with its own unique set of strengths and weaknesses.
One advantage of ThreatModeler is its ability to model and simulate complex threat scenarios, allowing developers to better understand potential vulnerabilities and make informed decisions about security measures. Its use of threat modeling also helps to identify potential issues earlier in the development process, which can save time and resources in the long run. However, the tool can be complex to use, and its output can be difficult to interpret without a deep understanding of security concepts [53].
VectorCast is known for its comprehensive testing capabilities, including support for both static and dynamic analysis, and its ability to automate a large portion of the testing process. Its intuitive user interface makes it easy to use for developers, and its integration with other development tools makes it an attractive choice for teams using a variety of tools. However, the tool can be expensive, and its comprehensive nature can lead to a large number of false positives, which can be time-consuming to sift through [57].
Code Sonar is a popular choice for its ability to identify complex software defects and vulnerabilities, particularly in C/C++ code. Its advanced static analysis engine is able to analyze code at a deep level, allowing it to detect issues that other tools may miss. However, its powerful engine comes at a cost - Code Sonar can be resource-intensive and slow to analyze large codebases. Its output can also be complex and difficult to understand for non-security experts [57].
Software security analysis tools such as ThreatModeler, VectorCast, and Code Sonar play a crucial role in ensuring the security and reliability of software applications. By using these

tools, developers can identify and eliminate vulnerabilities at various stages of the development process, reducing the risk of security breaches and improving the overall quality of the software. Overall, each of these tools offers unique advantages for software security analysis, but also has its own set of limitations. Choosing the right tool for a given project will depend on the specific needs and constraints of the project, as well as the expertise of the development team.

8 Conclusion and Future Work

The Security Abstraction Model (SAM) is a modeling language that provides a systematic way of specifying security-related requirements, policies, and mechanisms in the system architecture of automotive software systems. SAM was initially proposed to address the growing need for improved security in automotive systems, and its recent enhancements have further expanded its capabilities.

One of the most significant enhancements to the SAM model is the addition of social engineering attack modeling. Social engineering attacks are becoming increasingly common and pose a significant threat to the security of automotive systems. The SAM model now includes the ability to model and mitigate such attacks, which will help to improve the overall security posture of automotive systems.

Another important enhancement to the SAM model is the provision of tool support. The model includes tools that aid in identifying potential security threats and in implementing the necessary security measures to mitigate those threats. The tool support feature of the SAM model is essential in facilitating the integration of security considerations into the system architecture of automotive software systems.

Integration with the ISO 21434 standard for automotive cybersecurity is another significant enhancement to the SAM model. The ISO 21434 standard provides guidelines for addressing cybersecurity threats in the automotive industry, and the SAM model can now be used to address those threats more comprehensively. By integrating the SAM model with the ISO 21434 standard, the model can be used to develop more secure and reliable automotive software systems.

Future work in this area will focus on further refining and improving the SAM model. One area of focus will be the automation of the newly introduced scores from the ISO 21434 standard. This will help to streamline the process of addressing cybersecurity threats in automotive software systems. Additionally, an evaluation study of the effectiveness of the enhanced SAM model will be conducted with industry partners. This study will help to identify any areas where the SAM model can be improved and will help to ensure that the model remains effective in addressing security threats in the automotive industry.

In conclusion, the SAM model has been enhanced to address the growing need for improved security in automotive software systems. The addition of social engineering attack modeling, tool support, and integration with the ISO 21434 standard have greatly expanded the capabilities of the SAM model. Future work in this area will focus on further refining and improving the SAM model, and such work is critical in ensuring the continued success of this model in addressing security threats in the automotive industry.

Social Engineering Exploits in Automotive Software Security: Modeling Human-targeted Attacks with SAM

Matthias Bergler

Computer Science, Technische Hochschule Nürnberg, Germany. E-mail: matthias.bergler@th-nuernberg.de

Juha-Pekka Tolvanen

MetaCase, Finland. E-mail: jpt@metacase.com

Markus Zoppelt

Computer Science, Friedrich Alexander Universität Erlangen, Germany. E-mail: markus.zoppelt@fau.de

Ramin Tavakoli Kolagari

Computer Science, Technische Hochschule Nürnberg, Germany.
E-mail: ramin.tavakolikolagari@th-nuernberg.de

Security cannot be implemented into a system retrospectively without considerable effort, so security must be taken into consideration already at the beginning of the system development. The engineering of automotive software is by no means an exception to this rule. For addressing automotive security, the AUTOSAR and EAST-ADL standards for domain-specific system and component modeling provide the central foundation as a start. The EAST-ADL extension SAM enables fully integrated security modeling for traditional feature-targeted attacks. Due to the COVID-19 pandemic, the number of cyber-attacks has increased tremendously and of these, about 98 percent are based on social engineering attacks. These social engineering attacks exploit vulnerabilities in human behaviors, rather than vulnerabilities in a system, to inflict damage. And these social engineering attacks also play a relevant but nonetheless regularly neglected role for automotive software. The contribution of this paper is a novel modeling concept for social engineering attacks and their criticality assessment integrated into a general automotive software security modeling approach. This makes it possible to relate social engineering exploits with feature-related attacks. To elevate the practical usage, we implemented an integration of this concept into the established, domain-specific modeling tool MetaEdit+. The tool support enables collaboration between stakeholders, calculates vulnerability scores, and enables the specification of security objectives and measures to eliminate vulnerabilities.

Keywords: automotive systems, social engineering attacks, design, model-based development, modeling, security.

1. Introduction

The importance of security grew as hacks on cars and other connected devices became widespread and reported in the general public. Unfortunately, these are not rare special cases in certain car models or their particular components: “Hackers can clone millions of Toyota, Hyundai, and Kia keys” Greenberg (2020), “A new wireless hack can unlock 100 million Volkswagens” Greenberg (2016b), “Helpless in Jeep Cherokee” Timberg (2015). Since modern cars are computers on wheels, security challenges can be found from numerous systems, such as from anti-theft systems, tire pressure monitoring systems, remote keys, Bluetooth, radios (3G, 4G, 5G) and telematics access functionality. Also, infotainment systems tend to provide access to 3rd party applications as well as internet access. Social engineering attacks are also becoming increasingly popular due to increasing digitalization. The advancing digital exchange has made it easier for attackers to gain access to confidential data and thus smuggle malicious software into the development process PurpleSec (2021). The headlines on successful attacks are not only embarrassing, but customer concerns and lawsuits on vulnerabilities push the automotive industry to change Timberg (2015). Developing and maintaining secure systems, however, is not easy. First, security cannot be an afterthought, i.e., a component that may be added to or removed from an existing system. Second, security goals and measures to cope with vulnerabilities cannot be isolated from the rest of the design and development work. Instead, security must be designed in already from the very beginning of the system design SAE (2016). Third, the actual development of secure and trustworthy

Proceedings of the 31th European Safety and Reliability Conference.
Edited by Bruno Castanier, Marko Cepin, David Bigaud and Christophe Berenguer
Copyright © 2021 by ESREL2021 Organizers. *Published by* Research Publishing, Singapore
ISBN: 981-973-0000-00-0 :: doi: 10.3850/981-973-0000-00-0_"Social Engineering with SAM"

systems requires effort. It takes time, requires expertise from multiple fields and stakeholders, and needs to be linked with other development processes and practices. These challenges call for apt security design practices supporting companies to smoothly integrate security in their development processes. We propose a language-based approach and present a modeling language, called Security Abstraction Model (SAM) and its new extension for social engineering attacks with tool support for specifying security aspects. Unlike other approaches Cheah et al. (2017); Macher et al. (2016); Pattaranantakul et al. (2018); Matulevičius (2017); Microsoft-Corporation (2005), SAM is fully integrated into a standardized architecture description language, in this case the automotive-specific systems modeling language EAST-ADL Blom et al. (2013). It not only focuses on attacks, vulnerabilities and motivations for attacks but relates them with relevant parts of the automotive system design. Because of this integration, SAM also enables to start security design early on — already when the first high-level features of the vehicle are defined. This way, security is not an isolated afterthought but an integral part of the system development. Together with the tool support, SAM enables collaboration of system engineers with security engineers, traceability with system features, requirements, hazards as well as calculating vulnerability scores. SAM also helps in specifying security goals and measures to solve attacks. We demonstrate these via a case study along with identified benefits on applying SAM. We start by introducing the relevance of security in automotive and its current status. Then we go into social engineering attacks in automotive and the resulting danger. Afterwards we present SAM and the changes for covering social engineering, followed by the detailed tool implementation for all of its parts. Section 6 demonstrates the tool use with a practical example before the lessons learned and directions for future research are discussed.

2. Security in Automotive

Automotive system development has always had to adapt to the latest state of research and economic factors. Therefore, modern vehicles became interconnected computer networks in which many electronic control units (ECUs) communicate with one another and with the environment (Vehicle-to-X communication). Car manufacturers were producing vehicles that feature advanced desktop-grade software components. These vehicles have advanced algorithms for assistance systems and other computer-dominant extensions that can provide entry points and powerful tools for malicious attackers. It is thus not surprising that the number of scientific publications on automotive security has increased drastically Amendola (2004); Hubaux et al. (2004); Wolf et al. (2004). Considering the fact that autonomous vehicles will continue rather than reverse the trend towards more communication interfaces for reasons of functionality, safety and comfort, collective research efforts in the field of vehicle security are reasonable; after all, human lives are at stake every time these "driving computers" are the target of attacks. Combining state-of-the-art software components with legacy interfaces and hardware infrastructure decisions results in a risky set up from an IT security perspective. Legacy mechanisms like insecure and unencrypted protocols (e.g., Controller Area Network (CAN)) were originally not designed in accordance with today's security principles. Secure automotive network architectures were not prioritized in the past due to the general prejudice of cars' security due to their technical complexity (security by obscurity). Sluggish development processes, lack of standard guidelines and low societal pressure lead to a rather slow transformation of automotive development processes taking the security-by-design principle systematically into consideration. Most existing countermeasures against cyber-attacks, e.g., the use of message cryptography for encrypting, authenticating or randomizing vehicle-level network messages focus on concrete attacks and do not consider the complexity of the access options offered by modern vehicles Zoppelt and Kolagari (2019). This is mainly due to a solution-oriented approach to security problems. Defining and enforcing security goals for the automotive system helps to improve overall security. In the presented work, we address the security goals integrity, authenticity, confidentiality, reliability, availability and accountability. Many attack vectors often affect multiple security goals at once. Some of the attack vectors known to cause major threats to automotive systems include:

- Gaining remote control access to the vehicle using the OEMs cloud and/or mobile application's infrastructure Nie et al. (2018); Lab (2019); Nie et al. (2017); Lab (2018).
- Getting SecurityAccess via Unified Diagnostic Services (UDS) Van den Herrewegen and Garcia (2018).
- Controlling the car via Onboard Diagnostic (OBD) injection Zhang et al. (2016).
- Remotely breaking into the telematics unit Foster et al. (2015).
- Infecting the system with ransomware Ring (2015).

According to the SAE J 3601 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" SAE (2016), security affects the entire development, production and operation process of automotive systems. This is described in explicit analogy to ISO 26262-1:2018 (2018) the functional safety standard in the automotive sector, and results directly from the rule that security must be considered at the system design stage ("security by design"). With regard to concrete instructions in terms of the techniques to be used, the standard is fairly reserved, but in relation to model-based automotive system development, SAE (2016) refers in Appendices A-C to techniques for threat analysis and risk assessment, threat modeling and vulnerability analysis (e.g., attack trees) and explains when these should be used. The referenced techniques are relevant to the early stages of development in that they can be linked to requirements and design specifications by their illustrative (attack trees Cheah et al. (2017)), table-based Macher et al. (2016), use-case-based Pattaranantakul et al. (2018) and misuse case-based Matulevičius (2017) character. The development of threats and related information is typically performed by the STRIDE Threat modeling technique Microsoft-Corporation (2005) aiming to identify early possible security problems that may happen during the operation of a system. This approach is helpful even today, but what is true for this approach is equivalent to security modeling for enterprise systems: These approaches are not integrated into the design of the respective domain. Thus, it is not possible to identify the iterative cross-relationships between the designed system and security.

3. Social Engineering Attacks in Automotive

Thanks to the advanced digitization in communication, it is now possible to network and exchange ideas at any time. Especially during the COVID-19 pandemic, digitization is a key element in economy continuity. However, these digital communication systems are easily attackable and offer a weak point for engineering attacks, as we cannot identify our counterpart directly. Social engineering attacks in particular become increasingly popular with attackers, as they often lead to success despite the comparatively greater effort involved. This is because they are specially designed to exploit human weaknesses in behavior in order to obtain unauthorized access or sensitive data Mitnick and Simon (2003). The greater effort with some attacks depends on the fact that they have to be specially adapted to individual people, such as an employee of a certain company. Based on the technical report by the PurpleSec Association, the number of all kind of cyber-attacks has increased sixfold since the beginning of the pandemic and 98 percent of the cases are social engineering attacks PurpleSec (2021).

There are very different types of such attacks. These can be human-based or computer-based Xiangyu et al. (2017). Human-based means that there is direct interaction with a person, e.g. fake service calls or the well-known grandchildren's trick. Computer-based attacks are carried out using computer or email programs, e.g. phishing attacks. Even if the types of attack differ from one another, social engineering attacks always have the same pattern Mouton et al. (2016):

(i) Collect information about the target.
(ii) Develop relationship with the target.
(iii) Exploit the available information and execute the attack.
(iv) Exit with no traces.

In the automotive sector too, social engineering attacks may lead to success and create vulnerabilities for further attacks. These weaknesses have to be taken into account when developing the vehicle components. Although a vehicle cannot become a direct victim of a social engineering attack, successfully carried out attacks on employees in vehicle development or even a private person can lead to an attacker gaining access to individual vehicle parts or the entire vehicle. In many cases, these attacks can only be discovered but not prevented, e.g. in the case of a quid pro quo attack, in which a victim provides information or access in exchange for other services.

In addition to social engineering attacks on employees at a vehicle parts manufacturer in order to gain access to systems, private car owners are a popular target. There may be different motivations for the attack. Either an attempt can be made to take control of the vehicle himself by using social engineering attacks to trick the owner into installing malicious software or hardware in the vehicle, as described in Costantino et al. (2018), or valuable information about the driver can be obtained using vehicle data for further social engineering attacks on targets related to the victim, e.g. the victim's employer. An attack could look like this:

First of all, the vehicle type of the victim is spied out. The attacker then contacts the victim with the identity of a service employee and tries to find out more about the vehicle, the infotainment system and its usage behavior in a service conversation. He then offers the victim a free security update via CD, USB stick or mobile phone app, which the victim can download from a fake website or receive by post. This update actually installs malicious software that enables the attacker to read the victim's mobile phone data or to access the microphone of the hands-free system in order to record conversations and send them to the attacker via the mobile data connection of the mobile phone or vehicle. Alternatively, software can also be installed that gives the attacker control of the vehicle as in Greenberg (2016a). In order to prevent such attacks, possible points of attack via social engineering attacks must be identified and taken into account during the development of the components. With the latest expansion of SAM it is now also possible to map social engineering attacks and develop a counter strategy.

4. SAM: Automotive Domain Specific Security Modeling Approach

SAM is a modeling language for representing security-relevant properties of automotive software systems. It enables a security analysis of automotive attack vectors and conducts a thorough threat analysis. By means of systematic security analyses the effort for a potential attack can be quantified and appropriate counter measures can be modeled. The approach closely links security management and model-based system engineering by an abstract description of the principles of automotive security modeling. The resulting specification of SAM is based on security requirements that have been extracted from common industrial scenarios Zoppelt and Kolagari (2018). It aims to be a solution for representing attack vectors on vehicles and provide a thorough security modeling for the automotive industry.

4.1. *Feature-Targeted Attacks*

SAM has a close link to the architecture description via ‘Item’ entity (as in ISO 26262-1:2018 (2018) and Blom et al. (2013)) from the architecture model, playing the role of a ‘Feature’ from traditional security approaches. SAM aims to express all the important criteria of the attack vectors from the adversary’s motivation up to the security breach—to enable modeling of the system in early software development phases. In addition to attack motivations, SAM also describes all intrinsic and temporal characteristics of an attack, e.g., effects on the security objectives (confidentiality, availability, integrity, etc.), the complexity of the attack, the affected object and the vulnerability. The reason why we extended EAST-ADL rather than other languages like SysML or AADL is that EAST-ADL already addresses relevant aspects of automotive systems, namely its product line nature by specifying explicitly features that are either visible to customers (e.g., lane detection or regenerative braking) or on technical level (e.g., power generation control). EAST-ADL also directly addresses functional safety and ISO26262 with its Dependability Model. SAM identifies the same Items, Requirements and Hazards from architecture and dependability modeling and related them to Attacks and Security Concepts. Although SAM is developed as part of the EAST-ADL, it is not necessarily bound to EAST-ADL. SAM models are meant to be used and applied by anyone wanting to conduct a threat analysis of attacks in the automotive domain. SAM models have no aspiration to completeness with the rest of the systems model and can be used even in the very first phases of the system engineering process. Though it is advised to comply with the rest of the EAST-ADL systems model, SAM models can be used standalone.

4.2. *Human-Targeted Attacks*

In order to enable the representation of social engineering attacks in SAM, we had to adapt the original metamodel (Zoppelt and Kolagari (2018)) to provide the necessary classes[a]. For this purpose, the new abstract class ‘Target’ was introduced. This class generalizes the Item class already known in SAM, which represents the connection to the EAST-ADL, and the new HumanActor class, which is required for social engineering, as this can only be carried out on the basis of a person. The new class HumanActor has two attributes of the String type, which represent the exploitable human properties of curiosity and helpfulness. The use of these properties is either not defined (X), none (N), i.e. not used, low (L), used slightly, or high (H), used to a high degree. In addition, an association of the Resilience class refers to the HumanActor class. This class represents the mental resistance to social engineering attacks and possesses the attributes cautiousness, contentment, courage, experience and knowledge. They all correspond to the ResilienceLevel type, with which the extent of the property, required to defend against the attack, is measured. The different levels are low (L), low intermediate (LI), intermediate (I), high intermediate (HI), high (H) and not defined (X). The higher the demands on the resilience of a victim, the more successfully a social engineering attack can be carried out and the more adequate security concepts must be designed to counteract it. In section 6 an example of a social engineering attack with SAM is demonstrated. A security assessment through social engineering attacks can be carried out independently of the assessment of an Item, but it should definitely be considered. With the extension, SAM remains backwards compatible, as the use of social engineering attacks is optional in the application. The Common Vulnerability Scoring System (CVSS) is not relevant for the assessment, as the values in the Resilience class show an assessment of the severity of the attack but a extended scoring system is under development.

5. Language Implementation

While it is possible to create tooling for SAM from scratch, we applied MetaEdit+ MetaCase (2018b), a commercial language workbench, providing most of the needed functionality automatically: Only parts specific to SAM, its modeling concepts, rules, notation and integration

[a]https://www.in.th-nuernberg.de/professors/BerglerMa/SAM

with other tools, needed to be defined. This not only speeds up the implementation, but also enables easy evolution when the modeling needs change. MetaEdit+ provided also implementation of EAST-ADL MetaCase (2019) and other relevant functionality needed for automotive system development such as collaborative modeling, version control, integration with relevant tools applied in automotive (e.g., programming environments, Simulink, requirements management etc.) as well as having availability of supporting services. The only parts that deserved attention related to tooling were those parts not addressed by most metamodels, such as showing elements of SAM in the user interface (toolbar and browsers) based on their relevance, or decide how to indicate if constraints are not followed. In current definition, constraints that are considered mandatory are checked and reported at modeling time. Those constraints, not sensible to check like minimum cardinalities in the metamodel, are shown as recommendations in the live check pane at the bottom of the diagram. This way language users get immediate guidance to create security models. The original definition of SAM, similarly to the definition of EAST-ADL, focused on language concepts and on defining the exchange format via a metamodel. The definition of the whole language also needed to cover concrete syntax, all constraints, language usability topics as well as integration with other tools. These are detailed in the following subsections.

5.1. *Implementation of SAM concepts*

The implementation started by integrating SAM in the existing metamodel of EAST-ADL. First, the same conventions as applied in EAST-ADL and AUTOSAR were followed: Security models followed the same naming policies with short and optional longer names and all model elements had a globally unique identifier (UUID). Second, SAM was defined to follow the same package structure as EAST-ADL uses to organize specifications. Third, concepts of SAM were integrated with already existing EAST-ADL concepts, like Item from Dependability and ISO26262, VehicleFeature from variation models addressing product lines and Requirements from specifying and tracing with system requirements. Since MetaEdit+ allows to specify metamodels graphically (MetaCase (2018a)) similarly to UML, SysML, EAST-ADL or AUTOSAR as well as SAM we applied the form-based metamodeling tools of MetaEdit+. These allow the integration of all other related language constraints, notations as well as model checking and generators, which otherwise would be specified seperately, if at all, in addition to the metamodel. This tight integration of the whole language definition improves the quality of the language greatly. The typical problems from languages defined in an unintegrated manner, like inconsistencies and low quality Wilke and Demuth (2011); Bauerdick et al. (2004), can be more easily avoided. Figure 1 shows the concepts of SAM defined in MetaEdit+. The list of Objects shows the key modeling elements of SAM, the list of Relationships the connections between these elements, and the list of Roles how an object participates in the relationships, such as be directed or undirected, having constraints or detailed properties. To minimize the modeling effort the implementation defines one AttackMotivation and its concrete subtype is selected from a property. This way the type of AttackMotivation (Harm, Financial Gain etc.) can be changed without deleting the old one and creating and re-connecting a new one. This definition, compared to having a language construct for each subtype, is possible because all subtypes have the same properties and constraints. Also, for the reference from Attack to OperationalSituation, the role AttackSituation has a property to select if based on Traffic or Environment. Each element of SAM shown in Figure 1 are defined with further details. Figure 2 shows one such definition: The Vulnerability and its nine properties. The first three are obtained from EAST-ADL and AUTOSAR metamodel and the remaining from SAM. These properties have rules and constraints, such as 'Short name' being mandatory and starting with an alphabetical character followed by possible characters, numbers or underscores (defined as regular expression: `[a-zA-Z](_?[a-zA-Z0-9])*_?`. Another constraint of SAM is that Scope of Vulnerability may have only two possible values (unchanged or changed). To support usability, we slightly changed the ordering of the properties as set in the original metamodel to follow the order in which security engineers are expected to fill them and follow the same order as the vulnerability analysis tool Common Vulnerability Scoring System CVSS FIRST.Org (2019). The use

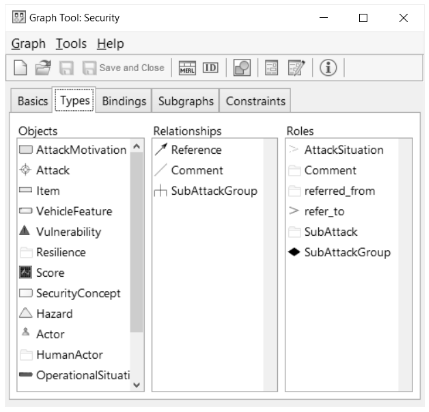
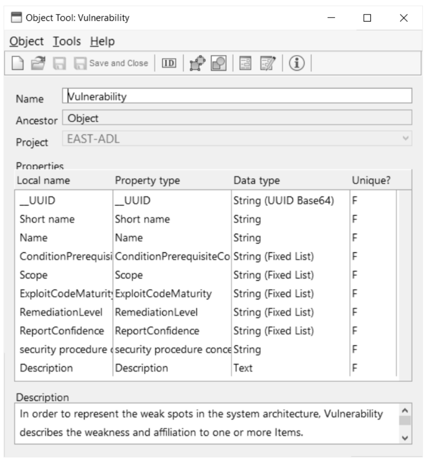


Fig. 1. Defined language concepts of SAM.

of Vulnerability concept is illustrated in Figure 3. The definition of the Vulnerability modeling concept was finalized by providing a description of this SAM concept (see Figure 2 the bottom of the window). Constraints of SAM are ensured in two different ways: either as part of the metamodel or applied via model checker. Examples of the former are rules on legal connections and uniqueness of element names. These constraints can be checked and ensured at modeling time, i.e., security models always follow them. As an example Vulnerability can refer to four model elements only (HumanActor, Item, Requirement and Score).

5.2. *Integration with Score Calculation*

Models made with SAM can be applied to analyze and calculate vulnerability scores. First we implemented a generator exporting data from models to CVSS to an online tool. CVSS provides a way to produce a numerical score reflecting the severity of vulnerability. The resulting numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes. In the same way integration with other analysis tools can be done, or if they provide other mechanisms, like programmable APIs, the modeling tool can apply them. Depending on the capabilities of the analysis system, the results can then be included back to the model. For example, MetaEdit+ can show the score directly in the Vulnerability model element. This information would then be also available when tracing from security properties to requirements, features and system design in general. Since integration from the external web-based calculator was not possible, and even if available would lead to slower modeling and score calculation process, also requiring an online connection, we implemented the CVSS calculator into the SAM modeling tool. This was done using the same generator system applied to produce vulnerability vectors to the existing CVSS calculator. The benefit of this latter approach is showing vulnerability scores immediately. In other words, scores are calculated real-time during modeling. We applied the existing Score element to show the results and followed also the color schemes of CVSS for the notation of Score. Figure 3 shows running the calculation at modeling time and displaying the results directly in the model: Base metric of CVSS in case of Service Call Scam Attack is 8.8 (high) and CVSS for temporal score is 8.1 (high).



Name: Vulnerability
Ancestor: Object
Project: EAST-ADL

Properties

Local name	Property type	Data type	Unique?
__UUID	__UUID	String (UUID Base64)	F
Short name	Short name	String	F
Name	Name	String	F
ConditionPrerequisi	ConditionPrerequisiteCo	String (Fixed List)	F
Scope	Scope	String (Fixed List)	F
ExploitCodeMaturity	ExploitCodeMaturity	String (Fixed List)	F
RemediationLevel	RemediationLevel	String (Fixed List)	F
ReportConfidence	ReportConfidence	String (Fixed List)	F
security procedure (	security procedure conce	String	F
Description	Description	Text	F

Description
In order to represent the weak spots in the system architecture, Vulnerability describes the weakness and affiliation to one or more Items.

Fig. 2. Definition of Vulnerability.

6. Case Study: Service Call Scam

In this example, we show a social engineering attack with the aim of getting the vehicle owner to update the infotainment system with malicious software. This software is supposed to access the function of the hands-free system via a back door and record phone calls or conversations made in the vehicle and forward them to the attacker via the internet connection of the smart phone as seen in Costantino et al. (2018). For this purpose, as in Figure 3, a service call from the car manufacturer is faked and the victim is provided with a software update for the infotainment system. Since the identity of the service employee cannot be confirmed over a telephone call, the victim must have high values in cautiousness, experience and knowledge. What makes preventing such an attack on the part of vehicle manufacturers difficult and dangerous. Much more defective software could also be installed, which would result in a loss of control over the vehicle as seen in Greenberg (2016a). Since CVSS is also implemented into the modeling tool, the same metrics are shown inside the metric element connected to individual Vulnerabilities (both base and temporal metric values are ‘high’). This way the resulting metric scores are not only seen at modeling time but also versioned, reported and documented along with the rest of the system design. Based on the specified security model, we can analyze the attack properties and try to derive SecurityConcept with requirements that must be satisfied to fix the vulnerabilities. In this case the requirements are different choices for update handling.

7. Conclusion

Security must be designed in at an early stage, be part of the rest of system designs, and be easily integrated with existing system development practices. We presented the extension of SAM for social engineering attacks and addressed the danger of social engineering attacks to create vulnerabilities. We also introduced a tool support

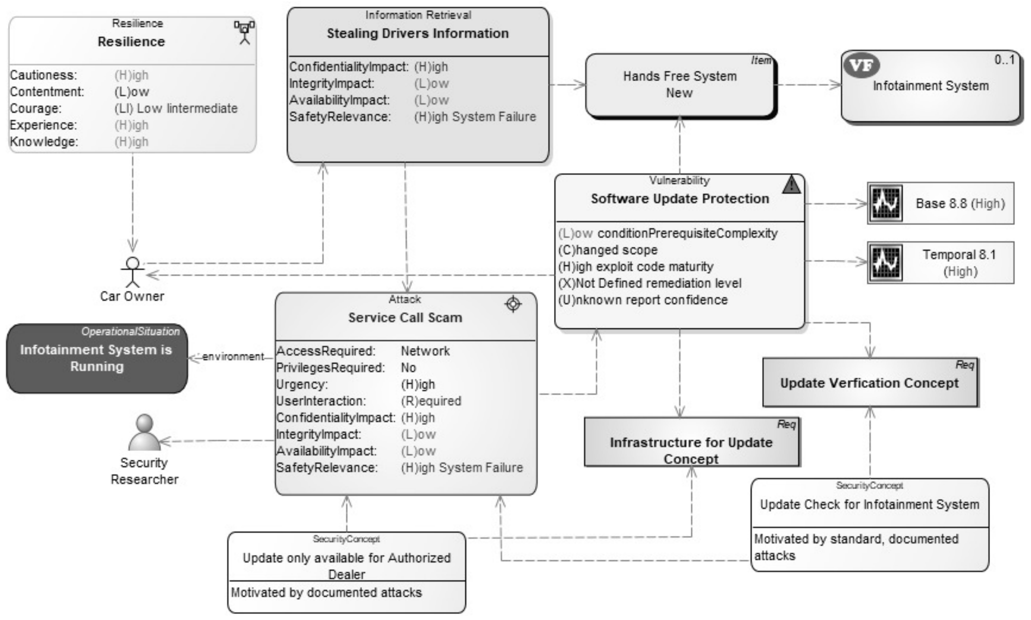

Resilience
Resilience
Cautioness: (H)igh
Contentment: (L)ow
Courage: (Ll) Low Intermediate
Experience: (H)igh
Knowledge: (H)igh
Information Retrieval
Stealing Drivers Information
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Item
Hands Free System
New
0..1
Infotainment System
Vulnerability
Software Update Protection
(L)ow conditionPrerequisiteComplexity
(C)hanged scope
(H)igh exploit code maturity
(X)Not Defined remediation level
(U)nknown report confidence
Base 8.8 (High)
Temporal 8.1 (High)
Car Owner
OperationalSituation
Infotainment System is Running
environment
Attack
Service Call Scam
AccessRequired: Network
PrivilegesRequired: No
Urgency: (H)igh
UserInteraction: (R)equired
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Security Researcher
Req
Update Verfication Concept
Req
Infrastructure for Update Concept
SecurityConcept
Update Check for Infotainment System
Motivated by standard, documented attacks
SecurityConcept
Update only available for Authorized Dealer
Motivated by documented attacks

Fig. 3. SAM model of Service Call Scam.

for developing secure automotive systems. The unique part of the proposed approach is its integration in existing architecture modeling languages applied in the automotive industry. In our case we demonstrated the implementation with EAST-ADL tooling. The developed modeling support provides several benefits for development teams:

- Security issues can be considered in relation to the system design—starting when the first customer visible features are identified.
- The proposed security language and tool support, guides developers by considering relevant aspects such as attacks, their motivation, vulnerabilities and relation to vehicle features.
- The security models are related with the rest of the system designs, with traceability.
- Vulnerability scores can be calculated immediately and be part of security models.
- Support communication and collaboration within a team.

We also presented the language implementation process covering the metamodel with rules, visual notation and integration with model analyzer. Because the investment on implementing tool support is modest, it pays off quickly as all the other developers can then model with the language, link with existing designs and produce vulnerability metrics on the identified attacks. This also indicates that the effort in implementing tool support for SAM with other modeling languages is modest—at least with current tooling and tools having access to the full metamodel. As both the modeling language and generators are fully open for modifications, the presented approach also gives full control for the company enabling future updates, like targeting other vulnerability analysis tools. Decision makers such as managers who allocate resources to security and engineers to implement countermeasures, or people without a security background who are not interested in all the details of an attack, may be overwhelmed by the level of detail of SAM, so due to collaborative nature of design work, the main users are both security engineers and system developers. At the same time, however, it is essential, especially for decision-makers, to understand and correctly assess the threat posed by attacks. This is why an additional view is needed that does not show all the details that a fully blown SAM model offers but can provide a quick overview of the mechanisms of operation and the severity of an attack. This provides a reliable initial basis for deciding on the allocation of resources for the creation of countermeasures. Our future work also focuses on process: Provide a methodology with a step-by-step refinement of the available information, whereby the refinement results on the one hand in the security expert gaining knowledge about details of the attack (over time) and on the other hand in extending the CVSS to consider resilience for social engineering attacks.

References

Amendola, S. (2004). Improving automotive security by evaluation—from security health check to common criteria. *White paper, Security Research & Consulting GmbH 176*.

Bauerdick, H., M. Gogolla, and F. Gutsche (2004). Detecting ocl traps in the uml 2.0 superstructure: An experience report. In *International Conference on the Unified Modeling Language*, pp. 188–196. Springer.

Blom, H., H. Lönn, F. Hagl, Y. Papadopoulos, M.-O. Reiser, C.-J. Sjöstedt, D.-J. Chen, F. Tagliabo, S. Torchiaro, S. Tucci, et al. (2013). East-adl: An architecture description language for automotive software-intensive systems. In *Embedded Computing Systems: Applications, Optimization, and Advanced Design*, pp. 456–470. IGI Global.

Cheah, M., H. N. Nguyen, J. Bryans, and S. A. Shaikh (2017). Formalising systematic security evaluations using attack trees for automotive applications. In *IFIP International Conference on Information Security Theory and Practice*, pp. 113–129. Springer.

Costantino, G., A. La Marra, F. Martinelli, and I. Matteucci (2018). Candy: A social engineering attack to leak information from infotainment system. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5.

FIRST.Org, I. (2019). First, common vulnerability scoring system, version 3.1.

Foster, I., A. Prudhomme, K. Koscher, and S. Savage (2015). Fast and vulnerable: A story of telematic failures. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*.

Greenberg, A. (2016a, January). *The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse*. Wired.

Greenberg, A. (2016b, March). *Radio Attack Lets Hackers Steal 24 Different Car Models*. Wired.

Greenberg, A. (2020). *Hackers can clone millions of Toyota, Hyundai, and Kia keys*. Wired.

Hubaux, J.-P., S. Capkun, and J. Luo (2004). The security and privacy of smart vehicles. *IEEE Security & Privacy 2*(3), 49–55.

ISO 26262-1:2018 (2018, December). Road vehicles — Functional safety. Standard, International Organization for Standardization, Geneva, CH.

Lab, T. K. S. (2018). Experimental security assessment of bmw cars: A summary report.

Lab, T. K. S. (2019). Experimental security research of tesla autopilot.

Macher, G., E. Armengaud, E. Brenner, and C. Kreiner (2016). A review of threat analysis and risk assessment methods in the automotive context. In *International Conference on Computer Safety, Reliability, and Security*, pp. 130–141. Springer.

Matulevičius, R. (2017). Security risk-oriented misuse cases. In *Fundamentals of Secure System Modelling*, pp. 93–105. Springer.

MetaCase (2018a). The graphical metamodeling example.

MetaCase (2018b). Metaedit+ user's guide.

MetaCase (2019). East-adl tutorial.

Microsoft-Corporation (2005). The stride thread model.

Mitnick, K. D. and W. L. Simon (2003). *The art of deception: Controlling the human element of security*. John Wiley & Sons.

Mouton, F., L. Leenen, and H. Venter (2016). Social engineering attack examples, templates and scenarios. *Computers & Security 59*, 186–209.

Nie, S., L. Liu, and Y. Du (2017). Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA 25*, 1–16.

Nie, S., L. Liu, Y. Du, and W. Zhang (2018). Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars. *Briefing, Black Hat USA*.

Pattaranantakul, M., R. He, Q. Song, Z. Zhang, and A. Meddahi (2018). Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys & Tutorials 20*(4), 3330–3368.

PurpleSec (2021). 2021 cyber security statistics the ultimate list of stats, data & trends.

Ring, T. (2015). Connected cars - the next target for hackers. *NetworkSecurity 11*, 11–16.

SAE, S. (2016). j3061, cybersecurity guidebook for cyber-physical vehicle systems. *Nr 1*, 52.

Timberg, C. (2015, July). *Hacks on the Highway*. Washington Post.

Van den Herrewegen, J. and F. D. Garcia (2018). Beneath the bonnet: A breakdown of diagnostic security. In *European Symposium on Research in Computer Security*, pp. 305–324. Springer.

Wilke, C. and B. Demuth (2011). Uml is still inconsistent! how to improve ocl constraints in the uml 2.3 superstructure. *Electronic Communications of the EASST 44*.

Wolf, M., A. Weimerskirch, and C. Paar (2004). Security in automotive bus systems. In *Workshop on Embedded Security in Cars*, pp. 1–13. Citeseer.

Xiangyu, L., L. Qiuyang, and S. Chandel (2017). Social engineering and insider threats. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 25–34.

Zhang, Y., B. Ge, X. Li, B. Shi, and B. Li (2016). Controlling a car through obd injection. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 26–29. IEEE.

Zoppelt, M. and R. T. Kolagari (2018). Sam: a security abstraction model for automotive software systems. In *Security and Safety Interplay of Intelligent Software Systems*, pp. 59–74. Springer.

Zoppelt, M. and R. T. Kolagari (2019). What today's serious cyber attacks on cars tell us: consequences for automotive security and dependability. In *International Symposium on Model-Based Safety and Assessment*, pp. 270–285. Springer.

Integrating Security and Safety with Systems Engineering: a Model-Based Approach

Matthias Bergler
Technische Hochschule Nürnberg
Nürnberg, Germany
matthias.bergler@th-nuernberg.de

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland
jpt@metacase.com

Ramin Tavakoli Kolagari
Technische Hochschule Nürnberg
Nürnberg, Germany
ramin.tavakolikolagari@th-nuernberg.de

Abstract—**Development of reliable systems requires that safety and security concerns are acknowledged during system development. Adding them afterwards is risky as many concerns are missed if not elicited together with the system requirements. Unfortunately, languages for systems engineering, like SysML, typically ignore security and safety forcing development teams to split the work into different formats, languages and tools without easy collaboration, with limited traceability, separate versioning and restricted use of automation that tools can provide. We present a model-based approach targeting automotive that integrates safety and security aspects with other system development practices. This is achieved via a comprehensive domain-specific modeling language that is extendable by language users. We demonstrate this approach with practical examples on how security and safety concerns are recognized along with traditional system design and analysis phases.**

Keywords—model-based development; security, safety, domain-specific language; system engineering; software engineering

I. Introduction

The development of reliable systems requires the consideration of safety aspects during system development. Adding them afterwards is difficult and error-prone, as important stakeholder concerns can be lost. Unfortunately, systems engineering languages such as SysML [12] typically ignore safety and security, forcing development teams to divide the work into distinct subsections: Safety and security are defined in different formalisms and languages, without clear traceability, collaborative work and use of other automations that tools can provide.

We present a model-based approach targeting the automotive domain that integrates safety and security aspects into the rest of system development practices. This is achieved through a modeling language that combines systems engineering concepts with those of safety and security. Thus, modeling is not only about specifying a system with blocks, signal connections, etc., but at the language level, security-relevant aspects such as attacks and vulnerabilities as well as safety-relevant aspects such as hazardous events and feature flaws are considered too. Safety and security are thus directly present in the development language as first-class citizens, just like the other aspects relevant for the specification and modeling of systems.

Our work on integrated language development is based on collaborative work over several years with automotive companies, researchers, and tool providers [2]. We demonstrate the approach with practical examples on how security and safety concerns are recognized with system design — covering traceability and automatically conducting several relevant safety-related methods such as ISO26262 [4], FTA [8] and FMEA [13], and security-related methods such as vulnerability scores [3]. The integrated model-based approach helps to ensure that safety and security is done for the intended system currently being developed and assists engineers with automation that tools can provide, covering editing, tracing, versioning and various analysis and reporting.

We start by presenting our solution by describing how the modeling languages are defined and integrated with existing languages. This is followed by case studies demonstrating the approach with practical examples. First, we describe how an existing automotive language is extended with security modeling: covering both attacks from the social engineering side and from the technical side. This is followed by describing language extensions for safety following strictly the functional safety as defined in ISO26262. For both language extensions we show examples on their use. We conclude by sharing our experiences on defining languages for safety and security.

II. Defining Modeling Languages and Their Integration

In this section, we give a brief overview of the possibilities of describing modeling languages (Section II-A) and the underlying modeling concepts that have been considered in the creation of safety/security aspects in the automotive context in the present language (Section II-B and II-C). The section concludes with a summary of the major benefits of an integrated modeling approach (Section II-D).

A. Metamodelling: description of the allowed structure and syntax

All computer-based languages, including programming languages and modeling languages, are defined with some formalism. For model-based development the languages are typically defined by metamodels [6]. A metamodel describes the allowed structure and syntax with which we can create models. Consider Fig. 1 illustrating a very small fragment of the

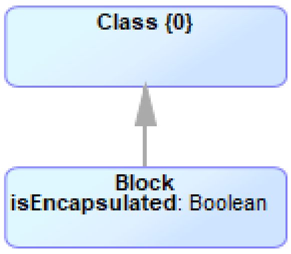


Fig. 1. Small part of SysML metamodel

metamodel taken from SysML: A ‘Block’ has one Boolean property called ‘isEncapsulated’ to define if a block is treated as a black box element [12]. Based on this metamodel, the ‘Block’ does not have any other properties as those that are defined by its supertype ‘Class’— and these ‘Class’-specific properties are omitted from this small metamodel.

Language definition normally also includes constraints to ensure that the created models are syntactically complete, correct and consistent. Some of these constraints are expressed directly in the metamodel whereas others are defined with additional scripts or constraint definitions. An example of the former is that an attribute is mandatory and an example of the latter that an element must have a certain number of connections.

Definition of a general-purpose modeling language, like SysML or UML, contains hundreds of elements like ‘Class’ and ‘Block’ and their properties like ‘isEncapsulated’. Although they can be considered as large languages neither of them covers

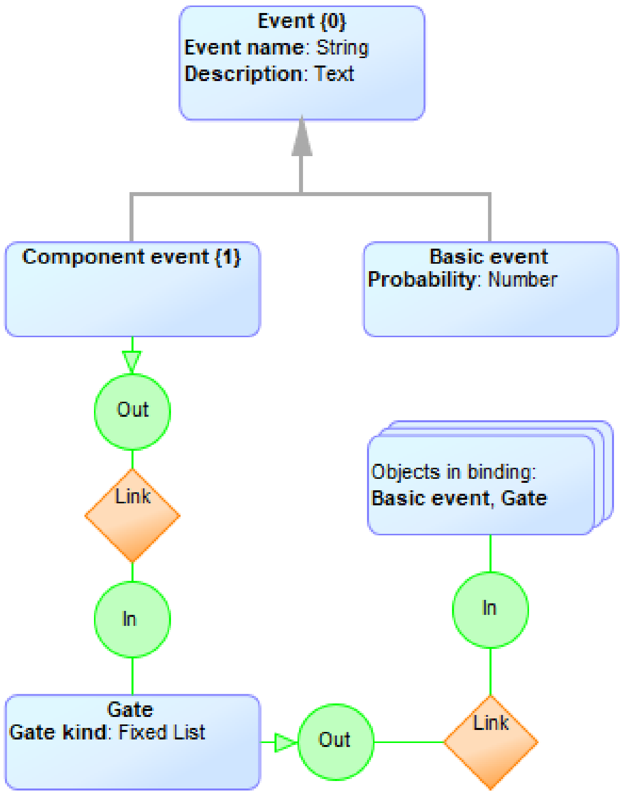


Fig. 2. Complete metamodel of Fault Tree modeling language

aspects relevant to safety or security. For example, a common practice for safety engineering is defining fault trees and performing Fault Tree Analysis (FTA) [8]. Fault trees is also a language having own metamodel that is independent and thus unrelated with SysML. Fig. 2 illustrates a metamodel of the fault tree. This metamodel contains just a few elements but it still defines the complete FTA language: There are two types of events: a ‘Component event’ acting as a root for the fault tree and a number of ‘Basic events’ causing an error with some probability. Both elements have ‘Event name’ and ‘Description’ properties that are inherited from the abstract supertype ‘Event’. In addition, ‘Basic event’ has a ‘Probability’ property to indicate how often the event can occur. Third element is a ‘Gate’ to indicate relationships among these three elements. Gate has a property ‘Gate kind’ with values like AND, OR for Boolean logic. Finally, a ‘Link’ element allows relating a root component event to a ‘Gate’ and those can be again linked with other gates or basic events.

In addition to the definition of a metamodel and related constraints, modeling languages also have notation for humans to create and read the models. Once notational symbols are given, a modern tool can provide the desired modeling support. Fig. 3 shows an example of fault tree modeling in a tool based on the metamodel defined in Fig. 2. This modeling editor, and related functionality, is provided automatically based on the language definition. In the example model ‘flat burns down’ is a component event that is the single root element, and there are two different types of gates with five different basic events. As basic events have failure probabilities, the tool can subsequently calculate the probability of the component failure.

If fault tree modeling should support other aspects, like link to other subtrees of faults the metamodel could be extended. Therefore, tools giving access to the metamodel (as in Fig. 2) enable users to extend the modeling capabilities directly without any lock or waiting for features from the tool vendor. This gives engineers tooling that can fit exactly to the needs and gives control over the ways systems are specified. Engineers using the tool can dictate, not tool providers.

In our example, the metamodel of fault tree and SysML are, however, disconnected which is understandable as capturing risks and safety concerns were not targeted when SysML was defined. However, we can identify connections between these two languages. For example, the root component of the fault tree most likely should refer to at least one of the blocks in SysML so that failure of the system block could be measured. Also, if working in automotive systems and on their functional safety the modeling support should be able to express ‘Item’, ‘Hazard’, ‘HazardousEvents’ that are all defined in the safety standard [4]. In the modeling approach presented in this paper, this desired connection between systems and safety as well as security modeling is fully supported via the language definition.

We present next language extensions that cover security and safety concerns related to automotive systems using EAST-ADL language that is made for developing automotive systems [2]. The principle of language extension and the practices described here are not limited to any particular modeling language though.

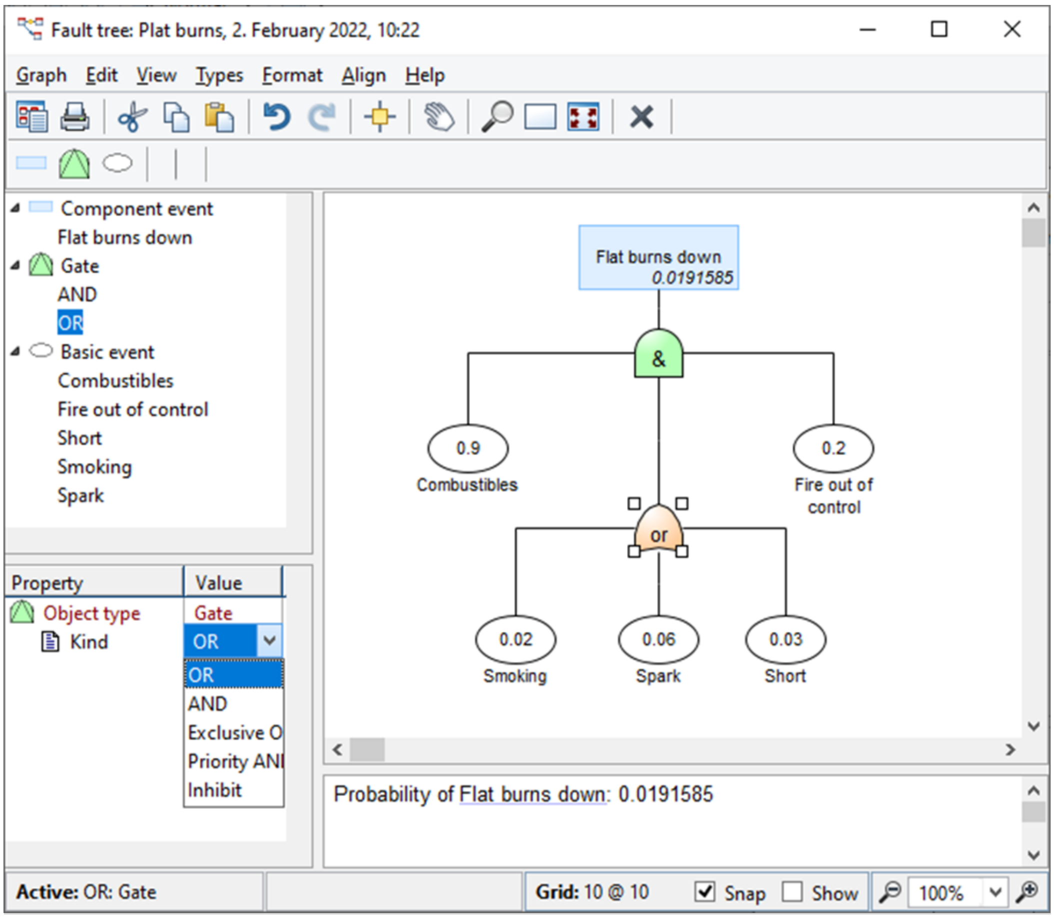


Fig. 3. Fault tree modeling based on the metamodel defined in Fig. 2.

B. *Security related capabilities of this approach*

Security Abstraction Model (SAM) targets representing security-related properties in automotive software systems. This modeling language enables a security analysis of attack vectors in the automotive sector and allows for an in-depth risk analysis. With SAM both potential attacks and countermeasures against these attacks can be specified. This allows the connection of security management and model-based systems engineering on an abstract description level according to the principles of automotive security modeling. SAM was defined based on security requirements from common industrial scenarios. It aims to be a solution for representing attack vectors on vehicles and provide a thorough security modeling for the automotive industry.

SAM has a close link to the system architecture description via the modeling entity 'Item' being part of the architecture model of EAST-ADL, an Architecture Description Language, aligned with the AUTOSAR automotive standard [2]. Item refers to a number of features of an automotive system. SAM tries to present all important criteria of the attack vectors, from the adversary's motivation up to the security breach. This allows a system to be represented from a security perspective in the early software development phase. In addition to the 'Attack motivations', SAM also describes all intrinsic and temporal characteristics of an 'Attack', e.g., effects on the security objectives (confidentiality, availability, integrity, etc.), the complexity of the attack, the affected object and the 'Vulnerability'. The latest version of SAM address also social engineering attacks [1].

Fig. 4 describes the metamodel of SAM. SAM acts as an extension to the EAST-ADL, because the EAST-ADL addresses relevant aspects of automotive systems (being a major requirement for security modeling that is not offered by languages like SysML [12] or AADL [15], which only offers feature modeling); especially the features of a vehicle of any kind. In addition, the EAST-ADL speaks directly about functional safety and ISO 26262 in its Dependability Model. SAM identifies the same 'Item', 'Requirement' and 'Hazard' from architecture and dependability modeling and relates them to 'Attacks' and 'Security Concepts'.

Although SAM is developed as part of the EAST-ADL, it is not necessarily bound to EAST-ADL. SAM as a metamodel is independent of other languages but for connectivity links to 'Item' and 'Requirement' of the EAST-ADL. In addition, SAM can also be used independently of the rest of the system model to provide an overview of safety critical system parts before or at the beginning of the system engineering process. For more information about SAM please see [16][17].

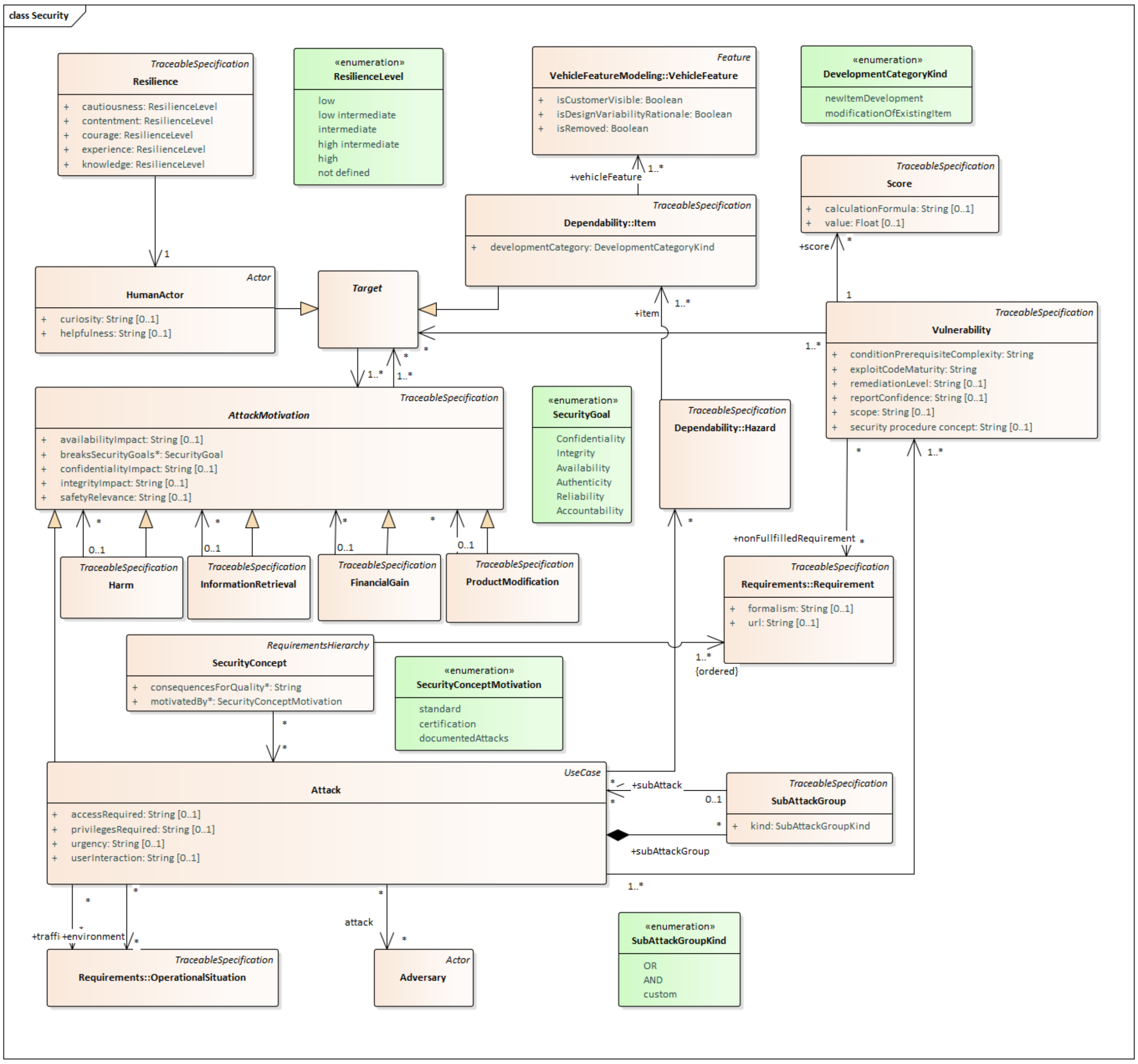


Fig. 4. Security metamodel. View Online at https://www.in.th-nuernberg.de/professors/BerglerMa/SAM/

Models created according to SAM permit calculating a vulnerability score based on the Common Vulnerability Scoring System [3]. This scoring system allows a qualitative representation (such as low, medium, high and critical) of the severity of an attack enabling prioritization in the vulnerability management process. First attempts were to implement a generator that transfers the model data to an online tool. However, since this would have required a longer modeling time due to the transfer to the online tool and a permanent internet connection, this idea was rejected. In the current version, the CVSS calculator is integrated directly into the SAM modeling tool MetaEdit+. The advantage of this is that no internet connection is required and the results can be viewed in real time next to the rest models. During the integration, we oriented ourselves to the color scheme of the CVSS. In this way, other analysis tools can also be integrated [1]. We demonstrate the use of the security modeling extension and CVSS calculation with examples in Section III.

C. Safety-related capabilities of this approach

To integrate aspects of functional safety, we followed functional safety standard ISO 26262 applied in automotive [4]. As in ISO 26262 an ‘Item’ is related to ‘Hazards’ and these are related to ‘Hazardous events’, which in turn can be classified according to ‘Severity’, ‘Exposure’ and ‘Controllability’. All these elements are also elements in the metamodel, like modeling objects or their properties. Fig. 5 shows the overview of the complete metamodel for dependability modeling for safety as defined in EAST-ADL.

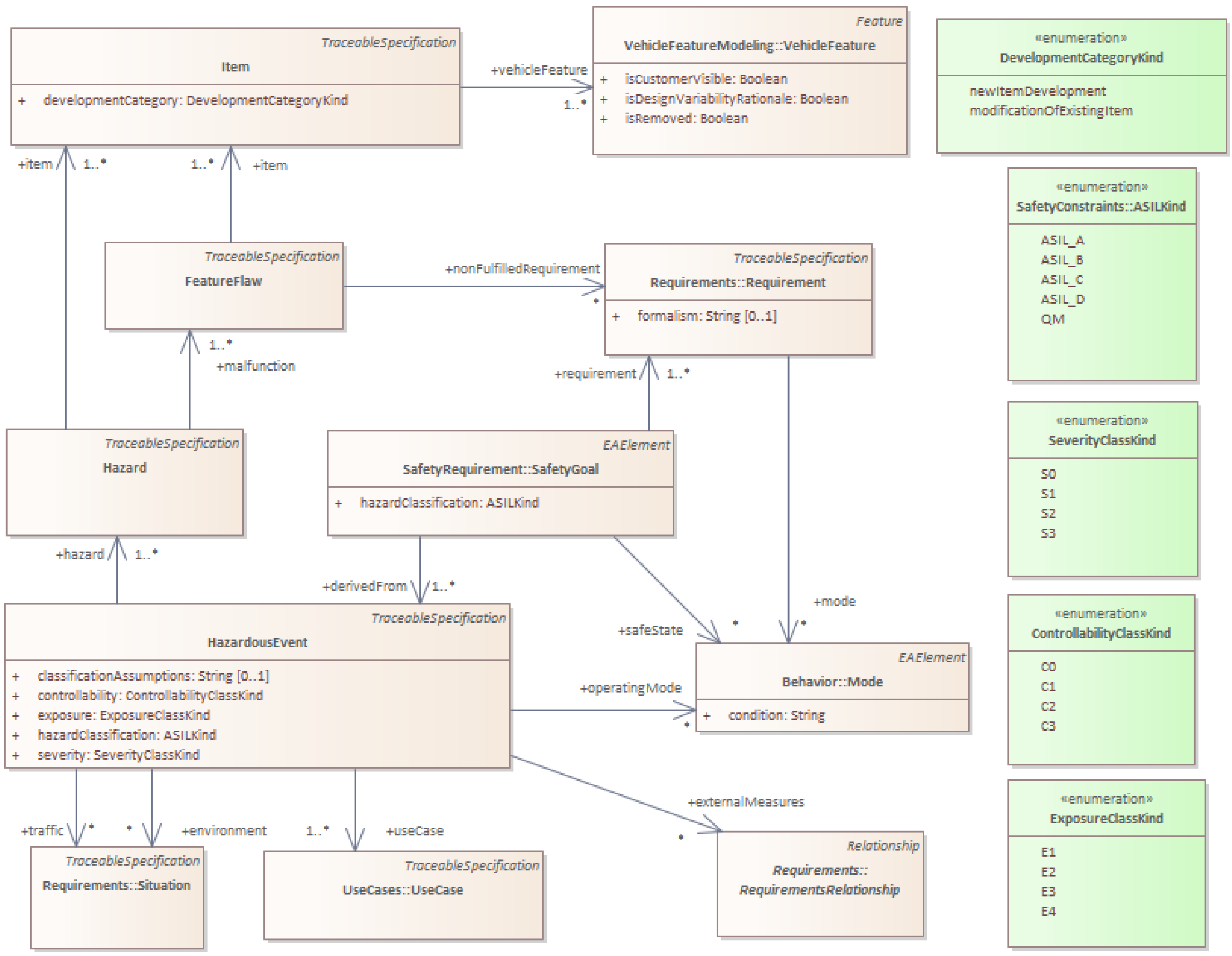


Fig. 5. Dependability package of the metamodel

The dependability package includes support for defining and classifying safety requirements through preliminary Hazard Analysis Risk Assessment (HARA), tracing and categorizing safety requirements according to their role in the safety life-cycle, as well as formalizing safety requirements using safety constraints. The dependability package itself is an extension to the automotive architecture modeling language EAST-ADL which already covers the modeling support for features, functions, hardware and related allocations. This full metamodel is described at http://east-adl.info/Specification.html along with the new version currently under review.

The metamodel shows how the integration with system architecture specification is established: the 'Item' (as defined in ISO 26262) is connected to features of the vehicle. As illustrated in Fig. 5 also constraints for this connection are defined making it mandatory (multiplicity is 1..*). In the metamodel of EAST-ADL, these individual features and their subfeatures are again connected via their context to any element in system architecture, such as to HardwareComponentTypes or FunctionTypes. Another form of connecting dependability modeling for safety is to apply 'Requirement' defined already in EAST-ADL and relate it with 'Safety Goals' and 'Feature Flaws'.

To formalize and assess fault propagation within the system, the dependability package also includes support for error modeling and organizing evidence of safety in a Safety Case. These are defined also via metamodels albeit not presented in Fig. 5. The integration of the system developed, the nominal system, and the error models is defined in the metamodel with trace links. Also, to minimize the modeling effort, automated error model generation based on system models is possible - and described with examples in Section IV.

D. *Benefits from integrated languages*

Integration of security and safety concerns with the language for system development offers several benefits over using separated languages — and tools and formats:

- Trace and analysis: A change in system engineering models or in safety/security-related models can be traced and analyzed. For example, all elements in system design that the safety-related 'Item' and its 'Hazard' can be related with can be identified. Similarly, if the system design is changed, e.g., by removing a feature, the related security or safety models can be identified and removed too — as they have become obsolete.
- Collaboration: Access to the system design as well as to security and safety aspects enables collaboration with fast feedback loop. If the modeling tools are not file-based, but apply a shared repository, then collaboration is even possible real-time. It is also up to tool features if access and collaboration is managed in some form, like allowing safety engineers to view system designs but not change them.
- Versioning: All models can be versioned together. There is no need to work with possible different formats, versioning systems, or collect data from different sources to get the complete picture at a particular point of time.
- Once the models share the same metamodel it is possible to run model checking and model transformations, like automatically produce initial safety models for the currently designed system. This way safety engineers don't need to manually create all safety models and they can better assure that their safety analysis is based on the planned system. With traces they can also follow what changes are made in the planned system too. Also, security aspects can be analyzed in the same way such as calculating vulnerability scores as described with example in Section III.
- Last and most importantly, security and safety design become tightly related to system designs sharing the same model structure.

III. Security Example

To illustrate the practical application, we show two scenarios of malware injections as examples for possible attacks: One via social engineering attacks and one via vehicle-to-vehicle attacks. In the first example an attacker attacks vehicle-to-vehicle communication in autonomous vehicles to install malware. The second attacker uses a social engineering attack to trick the owner into installing malicious software.

Fig. 6. shows the first case on how vehicle-to-vehicle communication is used in autonomous driving vehicles to infiltrate malware into the attacked vehicle via an attacking vehicle and what suitable countermeasures are available. Autonomous vehicles must know themselves and their surroundings very well in order to navigate through traffic as safely as possible. This requires not only many sensors and their evaluated data, but also the ability to communicate with other vehicles (Vehicle 2 Vehicle) or their environment (Vehicle 2 Everything) in an emergency. Many development projects in the field of autonomous driving are already taking place and aim to increase driving safety. But it is precisely the additional communication interfaces here that give rise to the possibility of attacks from outside: the attacker only must pose as a

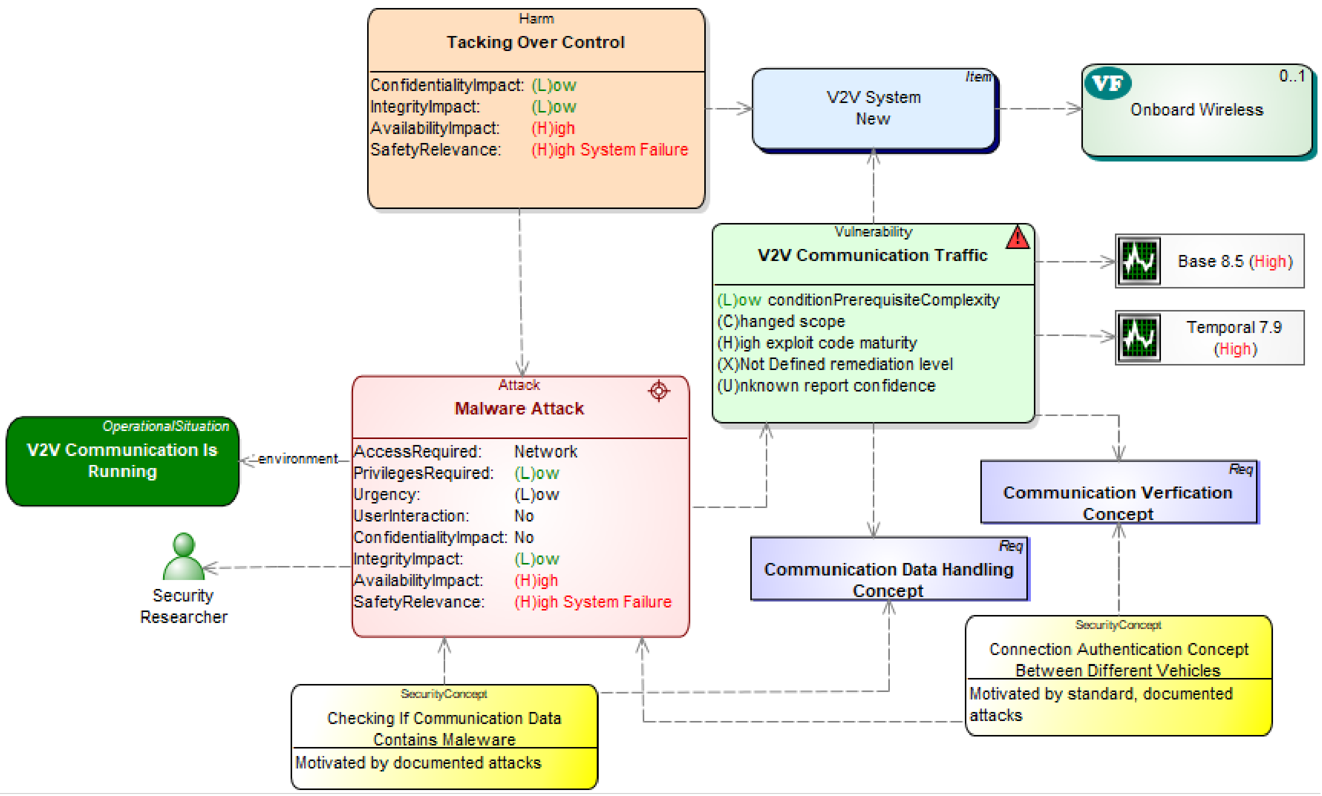

Harm
Tacking Over Control
ConfidentialityImpact: (L)ow
IntegrityImpact: (L)ow
AvailabilityImpact: (H)igh
SafetyRelevance: (H)igh System Failure
Item
V2V System New
VF
0..1
Onboard Wireless
Vulnerability
V2V Communication Traffic
(L)ow conditionPrerequisiteComplexity
(C)hanged scope
(H)igh exploit code maturity
(X)Not Defined remediation level
(U)nknown report confidence
Base 8.5 (High)
Temporal 7.9 (High)
Attack
Malware Attack
AccessRequired: Network
PrivilegesRequired: (L)ow
Urgency: (L)ow
UserInteraction: No
ConfidentialityImpact: No
IntegrityImpact: (L)ow
AvailabilityImpact: (H)igh
SafetyRelevance: (H)igh System Failure
OperationalSituation
V2V Communication Is Running
environment
Security Researcher
Req
Communication Verfication Concept
Req
Communication Data Handling Concept
SecurityConcept
Connection Authentication Concept Between Different Vehicles
Motivated by standard, documented attacks
SecurityConcept
Checking If Communication Data Contains Maleware
Motivated by documented attacks

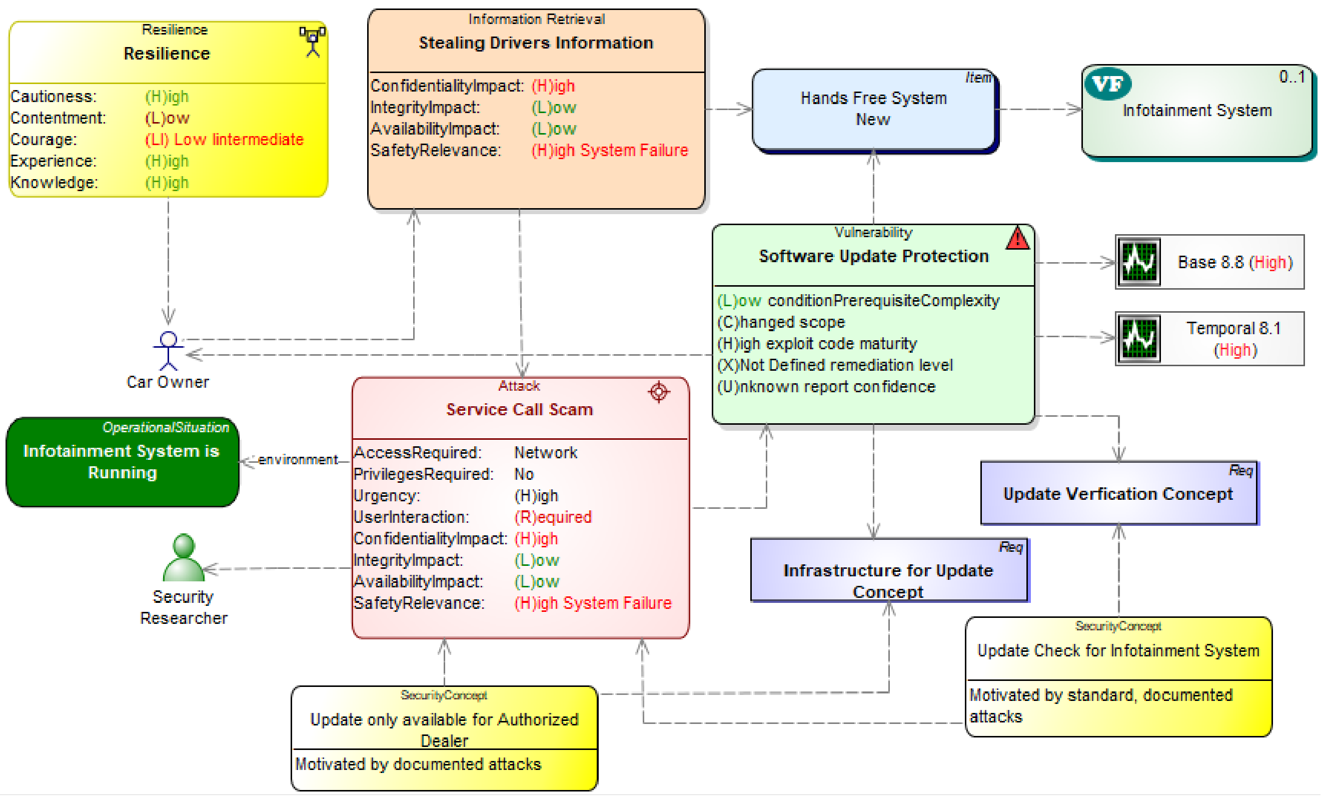
Fig. 6. Injecting Malware via V2V communication

trustworthy vehicle or environmental object and can thus establish a data connection to the attacked vehicle. The attacker can then use this data connection to install malware or steal sensitive user data. Therefore, it must be ensured that both the communication between the vehicles is sufficiently authenticated beforehand and that the transmitted data is checked for possible attacks such as the direct installation of malware or access to internal program structures through a memory overflow.

In the second example Fig. 7 shows a social engineering attack with the aim of getting the vehicle owner to update the infotainment system with malicious software. Due to the advanced digitization worldwide, more and more things are being done online without meeting the person opposite in person. It is also possible to install updates for infotainment systems simply from a downloaded file on a USB. However, this is precisely where the danger lies that one becomes the victim of a social engineering attack and corrupted software is installed on one's system, which serves as a gateway to further attacks. Fig. 7 shows such an example. The attacker first spies on the vehicle type, license plate number and other details of the intended victim. Contact is then made with information about a required service update for the vehicle's on-board computer, which the owner can easily install himself. If the victim is persuaded, the attacker will send them the software via a fake website, a CD or a USB stick. The victim installs the malicious software and the attacker gains access to the system via a back door and can, for example, listen to calls made by the victim via the hands-free kit or use the navigation system to track the victim's location. Countermeasures for this would be, for example, an update lock, which only official car workshops can bypass with special devices, but also a warning from the infotainment system itself that no new one is necessary and thus warns the victim well before something is installed. To model this, the latest extension of the SAM metamodel for social engineering attacks was used (https://www.in.th-nuernberg.de/professors/BerglerMa/SAM/).

Modeling support for safety covers calculating and displaying CVSS score for the attacked components. In Fig 7 the base and temporal scores are both ‘high’ for the vulnerability with values 8.8 and 8.1 respectively. While the calculations can be omitted, showing them at modeling time helps security engineers easier to identify possible weak points and initiate countermeasures as early as the design phase.

Since SAM supports comprehensive threat modeling that captures very detailed interrelationships between the properties of the attack and the vulnerability as well as their relationships to the architecture, some of which are not known in practice or should not be captured in all details for pragmatic reasons, a step-by-step approach to modeling with SAM makes sense. Therefore, we introduce different levels of modeling details, where Level 1 only models the motivation of the attacker, whereas Levels 2 and 3 are based on more details and information and thus enable a better representation of the threat situation; of course, this more detailed representation is accompanied by a higher development effort. Modeling tool also assist here security engineer to complete the specification by informing what elements can be considered next.



Resilience
Resilience
Cautioness: (H)igh
Contentment: (L)ow
Courage: (LI) Low Iintermediate
Experience: (H)igh
Knowledge: (H)igh
Information Retrieval
Stealing Drivers Information
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Item
Hands Free System New
VF
0..1
Infotainment System
Vulnerability
Software Update Protection
(L)ow conditionPrerequisiteComplexity
(C)hanged scope
(H)igh exploit code maturity
(X)Not Defined remediation level
(U)nknown report confidence
Base 8.8 (High)
Temporal 8.1 (High)
Car Owner
OperationalSituation
Infotainment System is Running
environment
Attack
Service Call Scam
AccessRequired: Network
PrivilegesRequired: No
Urgency: (H)igh
UserInteraction: (R)equired
ConfidentialityImpact: (H)igh
IntegrityImpact: (L)ow
AvailabilityImpact: (L)ow
SafetyRelevance: (H)igh System Failure
Security Researcher
Req
Update Verfication Concept
Req
Infrastructure for Update Concept
SecurityConcept
Update Check for Infotainment System
Motivated by standard, documented attacks
SecurityConcept
Update only available for Authorized Dealer
Motivated by documented attacks

Fig. 7. Injecting Maleware via Social Engineering Attack

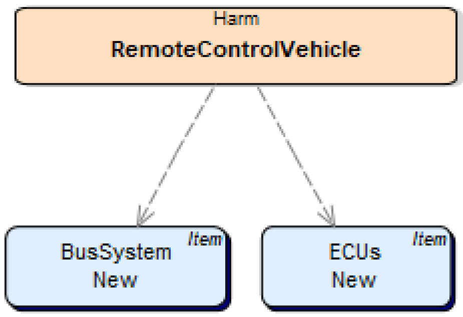


Fig. 8. Level 1 Representation of an attack in SAM

Level 1 is an entry level when working with SAM. It is to be used if only the motivation of the attacker is known about the attack(s) in question. This information already allows a pragmatic overview of the situation, a reference to the architecture of the automotive software system and an assessment of the severity of the attack, which is particularly relevant from a management perspective. As in Fig. 8, in Level 1 only the motivation is specified; specifically, this is one (or more) of the subclasses of 'AttackMotivation', i.e. either 'Harm', 'InformationRetrieval', 'ProductModification' or 'FinancialGain'.

The link between attack motivation and item plays a central role here. It arises from considerations of which item(s) is/are endangered by the attack. In Level 1, no details are known yet, so the item can be an umbrella term. For example, it is known that the car's bus system needs to be attacked for a given attack. However, it is not yet possible to say exactly which items are affected. Therefore, a "bus system" item can be created that includes a group of items. However, the item must be specified because it forms the connection to the rest of the automotive software architecture. Without the item, the threat modeling would be detached and would have no context to the rest of the architecture.

At Level 2, a more detailed threat analysis already takes place. The methodical orientation on Level 2 is recommended as soon as details about the attack are known, especially regarding the attribute 'breaksSecurityGoal'. The collection of (selected) details about the attack(s) are on the one hand not very time-consuming, but on the other hand offer a significantly better assessment of the threat situation compared to Level 1, since the focus of analysis is now no longer on human experience, but on technical feasibility. The entity Attack inherits the attribute breaksSecurityGoals of AttackMotivation. However, it is important to note that a single motivation for an attack can be divided into several subattacks, which can have different breaksSecurityGoals.

In addition to the breaksSecurityGoals, the reference to the followUpAttacks must also be modeled and the reference to the affected item must be established. The risk associated with the attack can be assessed qualitatively (which SecurityGoals are broken) or quantitatively (how many SecurityGoals are broken); this represents an improved assessment compared to Level 1,

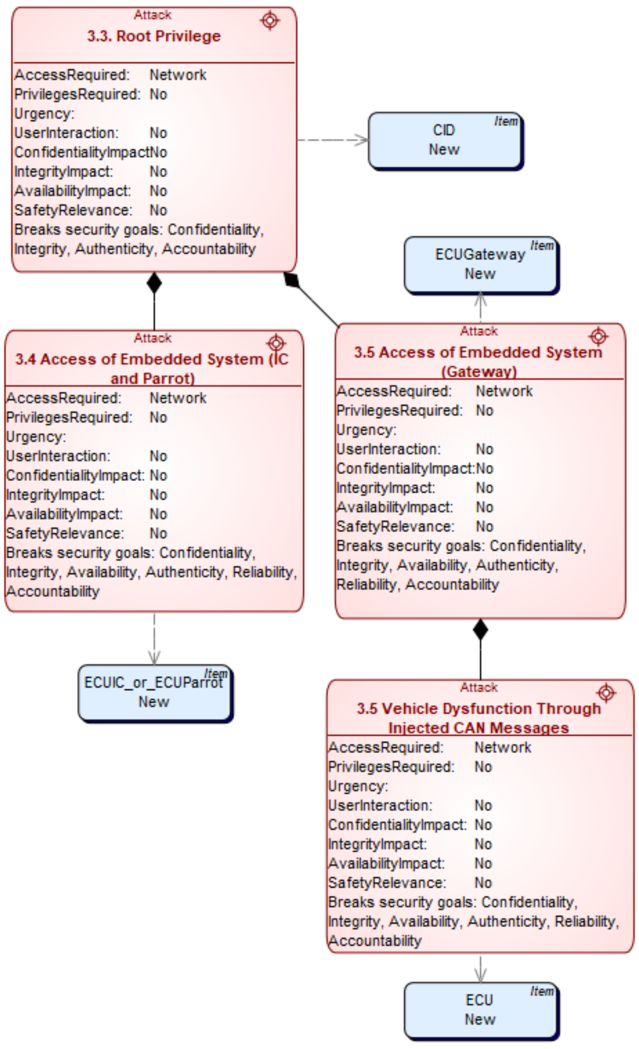


Fig. 9. Level 2 Representation of an attack in SAM

where the analysis is only carried out based on the motivation of the attack.

A better assessment of the severity of an attack results from the calculation of a score, for example analogous to the Common Vulnerability Scoring System (CVSS, see level 3). Level 2 is not detailed enough to calculate this score, but an initial assessment based on the experience of security experts can already be made and documented in the Score entity. To make it clear that this score is experience-based and has not been calculated, the attribute 'calculationFormula' is left empty in this case. Score can be omitted if no experience values are available.

ISO/SAE 21434 has been in place since 2021 to ensure a complete risk assessment analysis of cyber-attacks on vehicle systems [5]. This standard was developed because of the need to counteract against cyber-attacks due to the increasing networking of vehicle systems. The standard is related to the UNECE regulation R 155 "Cyber security and cyber security management system". The application of ISO 21434 is considered as a building block to facilitate certification. However, ISO 21434 does not cover all the requirements of R

155. To develop a reporting system for SAM that is understandable for different viewers, it makes sense to do this based on ISO 21434. The main steps in performing an ISO/SAE 21434 compliant threat analysis and risk assessment are (in order of an idealized linear execution):

1. Item Definition (Section 9.3)
2. Asset Identification (Section 15.3)
3. Identification of threat scenarios (Section 15.4)
4. Impact Rating (Section 15.5)
5. Attack Path Analysis (Section 15.6)
6. Attack Feasibility Rating (Section 15.7)
7. Risk Value Determination (Section 15.8)
8. Risk Treatment Decision (Section 15.9)
9. Cyber Security Goals (Section 9.4) [WP-09-03 & RQ-09-07]
10. Cyber Security Claims [WP-09-04 & RQ-09-06]
11. Cyber Security Concept (Section 9.5)

Most of the points listed are already implemented by SAM, including points 1, 2, 3, 4, 9, 10 & 11. Therefore, only the following points 5, 6, 7 & 8 have to be added to SAM to be able to create a reporting system based on ISO 21434. The points 9 to 11 are also fulfilled by the fact that they are applied holistically and not just domain-specifically and thus also cover the aspect of cybersecur1ity.

IV. SAFETY EXAMPLE

We focus next on safety and illustrate dependability modeling, error modeling and automated FTA/FMEA analysis that are possible due to metamodel extensions for safety. Fig. 10 shows the dependability model for PowerWindowController. This dependability model closely follows functional safety standard ISO26262 as has been defined in the metamodel (see Fig 5). In top of Fig. 10, PowerWindowController is considered as an item. Next, a hazard ‘Window obstacle not detected’ is specified and related to HazardousEvent ‘Window does not stop’. This Hazardous event is related to the use case that window action is requested. It may also be linked - albeit not described in the Fig. 10 - to other scenarios linked to traffic situation, environment or to operating mode. In other words, the model follows the metamodel as defined based on ISO 26262.

The example also describes the work on hazard analysis in which Severity, Exposure and Controllability and ASIL values are being defined. The dependability model also defines the safety goal with a safe state “PreventMovement”. It reduces the ASIL level to an acceptable level (B) with Requirements #7 and #9 that are already defined in the requirements model.

Thanks to the integrated metamodel safety design is not separated from the rest of the systems modeling. The integration is achieved by an ‘Item’ called PowerWindowController that, according to the metamodel, can refer to one or more features. In our example the item refers to the PowerWindow feature of the vehicle. This feature is defined in the feature model of EAST-ADL and is illustrated in Fig. 11.

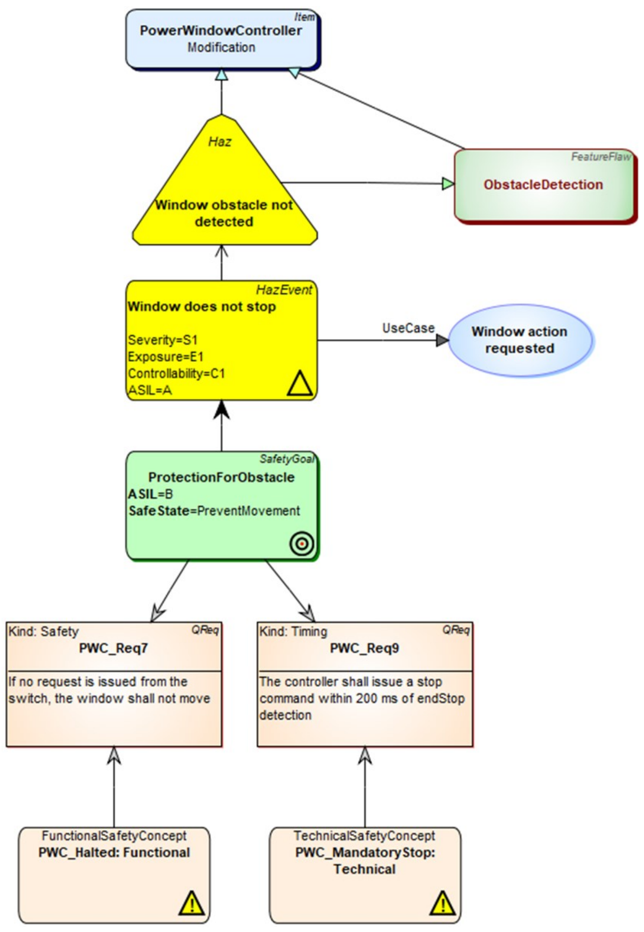


Fig. 10. Dependability model of PowerWIndowController

The features of Power Window are defined by systems engineers along with its functional architecture (as illustrated in Fig 12). Features in the feature model map to individual elements in the functional architecture. This enables traceability between system designs and safety designs.

System design as in Fig. 12, however, is not directly suitable for safety design as it does not identify and capture typical safety aspects like faults, failures or failure rates. Design models also include information that is not relevant for safety work adding unnecessary complexity for safety engineers. Consider for example the small system illustrated in Fig. 12. It shows the functional architecture of a PowerWindow controller: its functions, ports and connections among them. Functions are classified, have a more detailed internal hierarchical structure, and their ports have interfaces and data types. Some of these aspects, like classification of functions or data type definitions, are not directly relevant in safety design, but are necessary to generate the implementation (like AUTOSAR ARXML, Simulink models etc.).

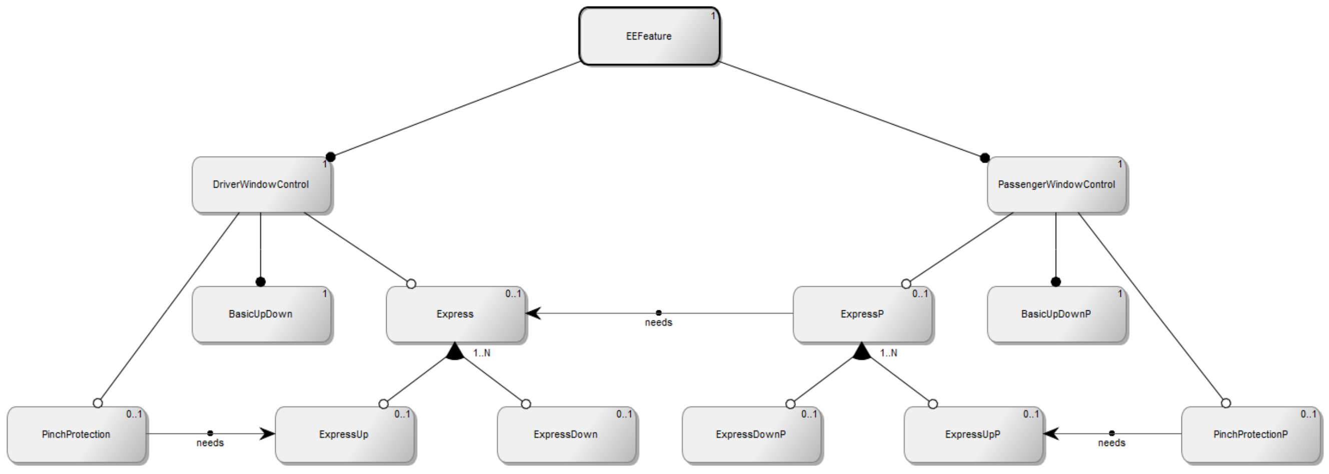


Fig. 11. Features of power window

Adding safety-related information, like faults or failure logic, directly to the design model is often not practical because it would quickly make the model complex with too many details. For safety analysis, unlike design, we also usually need several models covering different analysis scenarios. One solution is to generate the initial safety models from design specifications. For example, Fig. 13 shows the result of such a transformation: an initial error model produced from the system design shown in Fig. 12. The error model is then detailed for safety analysis focusing on failures, faults, and error types. Safety engineers may then add error behavior to this model or add other aspects of safety, like e.g., in Fig. 13 a FailureOut port is added to analyze an error on obstacle detection (top right in the model with blue thick border for propagation).

With error modeling, safety engineers can specify various faults and failure logic without changing the actual system designs. Yet, there is a link from error models back to nominal planned system descriptions. As error models can specify failure logic (Boolean and temporal) the models also serve as the basis for automated Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) [8][13]. This means that rather than creating fault tree diagrams manually (as in Fig. 3 earlier), they can be generated. For this purpose, we implemented model transformation that takes error models and translates them to the formats needed by FTA/FMEA tools. Since the transformations are fully automated the cost and effort to carry FTA and FMEA are greatly reduced. Fig. 14 shows the result of running the transformation from error models to analysis tools for FMEA.

Development of safety-critical functions with the model-based approach starts with hazard analysis and risk assessment in the dependability model, is detailed with error models for FMEA, and ends with verification of safety goals and safety requirements. Since models contain the needed information, it is possible to generate the documents like functional and technical safety concepts as well as verification and validation of safety goals (as done e.g., in [14]).

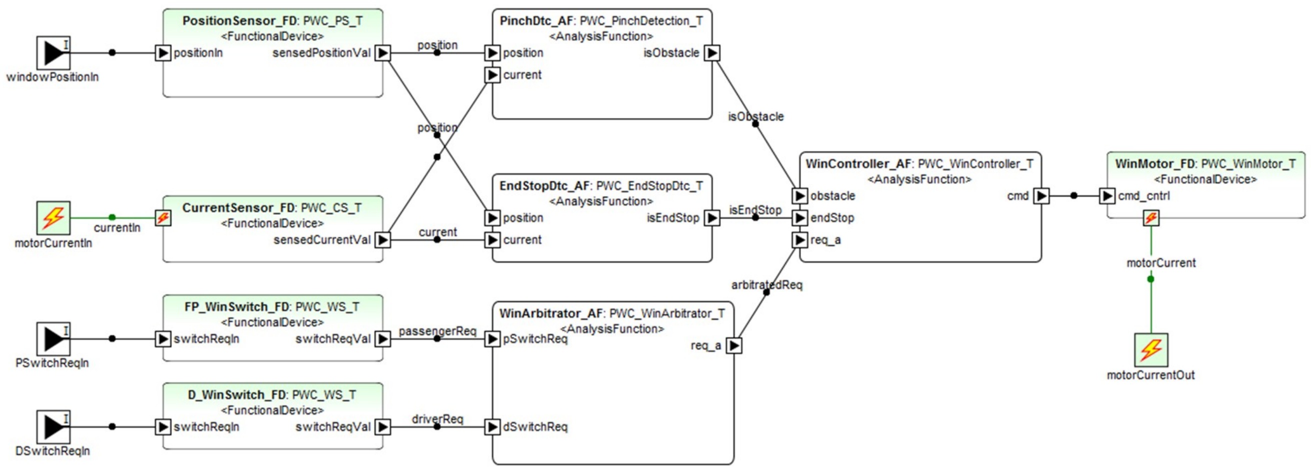
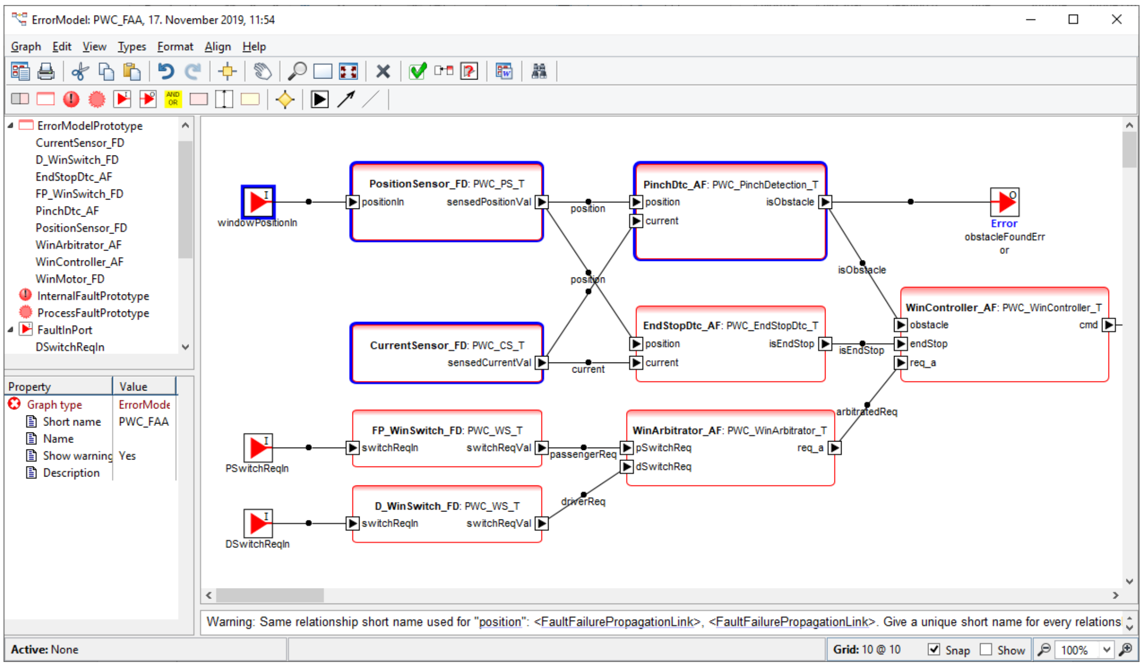


Fig. 12. Functional architecture of PowerWindow controller



Property	Value
Graph type	ErrorMode
Short name	PWC_FAA
Name	
Show warning	Yes
Description	

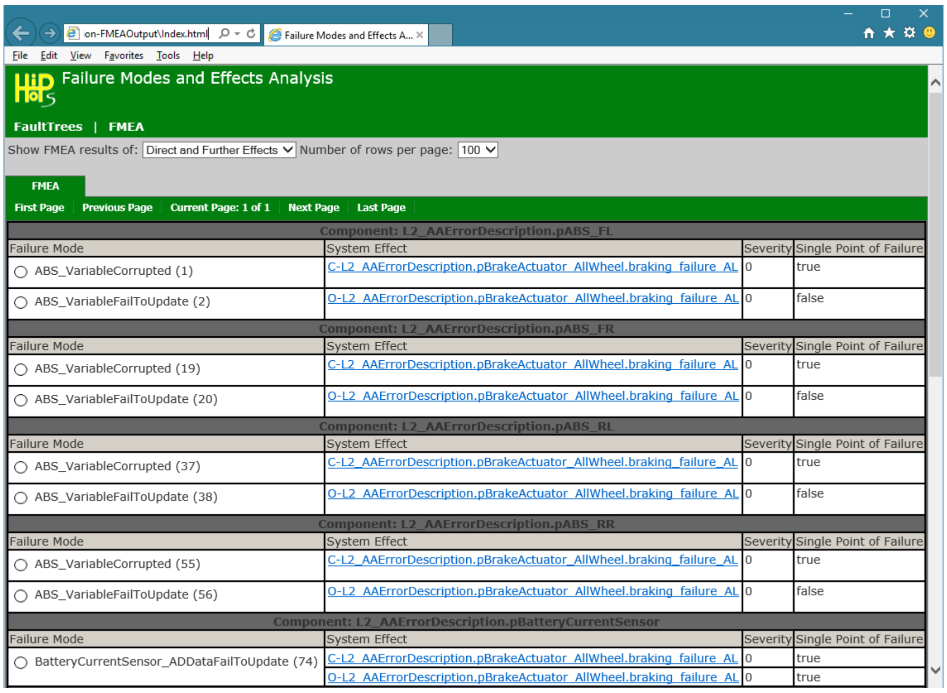
Warning: Same relationship short name used for "position": <FaultFailurePropagationLink>, <FaultFailurePropagationLink>. Give a unique short name for every relations

Fig. 13. Functional architecture of PowerWindow controller

These automations make safety work easier and faster to do as well as reduce manual error-prone routine tasks. Perhaps most importantly, they enable feedback from safety analysis earlier to be acknowledged in the system design.

V. Experiences on the Language Definition

Our efforts on implementing security and safety modeling have been related to existing automotive system development language, namely to EAST-ADL. For this reason, the actual language implementation required us to only specify the extensions (as in Fig. 4 and 5) and link them to the existing metamodel. The effort and process would be largely the same as if these would be added to other languages, like to AADL or SysML - given that these languages would have language constructs (elements in the metamodel) that are suitable for extension and integration with safety and security.


on-FMEAOutput\Index.html
Failure Modes and Effects A...
File Edit View Favorites Tools Help
HiP-HOPS

Failure Modes and Effects Analysis

FaultTrees | FMEA

Show FMEA results of: Direct and Further Effects Number of rows per page: 100

FMEA

First Page | Previous Page | Current Page: 1 of 1 | Next Page | Last Page

Failure Mode	System Effect	Severity	Single Point of Failure
Component: L2_AAErrorDescription.pABS_FL			
ABS_VariableCorrupted (1)	C-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true
ABS_VariableFailToUpdate (2)	O-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	false
Component: L2_AAErrorDescription.pABS_FR			
Failure Mode	System Effect	Severity	Single Point of Failure
ABS_VariableCorrupted (19)	C-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true
ABS_VariableFailToUpdate (20)	O-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	false
Component: L2_AAErrorDescription.pABS_RL			
Failure Mode	System Effect	Severity	Single Point of Failure
ABS_VariableCorrupted (37)	C-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true
ABS_VariableFailToUpdate (38)	O-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	false
Component: L2_AAErrorDescription.pABS_RR			
Failure Mode	System Effect	Severity	Single Point of Failure
ABS_VariableCorrupted (55)	C-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true
ABS_VariableFailToUpdate (56)	O-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	false
Component: L2_AAErrorDescription.pBatteryCurrentSensor			
Failure Mode	System Effect	Severity	Single Point of Failure
BatteryCurrentSensor_ADDataFailToUpdate (74)	C-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true
	O-L2_AAErrorDescription.pBrakeActuator_AllWheel.braking_failure_AL	0	true

Fig. 14. Results of fault tree analysis and failure modes and effects analysis.

The language definition steps consist of:

1. Defining the metamodel for the extensions and linking them with the existing language definition. Linking enables reuse, references and traces between model elements.

2. Setting constraints to keep specifications consistent and ensure syntactic completeness and correctness.

3. Defining notation that fits or resembles the domain being addressed (e.g., safety, security).

4. Implementing generators for model checking and trace as well as producing various kinds of artifacts like data for FMEA or CVSS.

In this paper we described the first two steps via the metamodel in Section II. Defining the notation in step 3 deals with symbol definitions and the work by Moody [10] can be applied directly here. The actual implementation of notations then varies on tools as some require programming them whereas for others they can be imported as images without much additional work [9].

The last step on generators then brings in more automation possibilities for checking, tracing, reporting and generating code, simulations etc. In our case the generators targeted external tools for performing Failure Mode and Effects Analysis (FMEA) as well as calculating vulnerability scores. In both cases implementing the actual generators were straightforward as for both needs specifications of the formats to be generated were available. The second kind of generators were those reporting the work in trace reports or various other documents. These were performed in the same MetaEdit+ tool.

The effort to create modeling support goes mostly to identifying and testing the suitable level of abstraction. We did not measure the effort on the safety side, but for the security side after having the initial metamodel (as in Fig. 4), its implementation into a modeling tool was done in the period of two calendar weeks by one person. Verification and validation were done by other people using the language for typical modeling cases.

It is important to note that if the tools provide access to the language definitions the modeling support can be extended incrementally based on the needs. For example, both the metamodel definitions addressing security and safety have evolved because of the changes in the modeling requirements and because learning from the language usage. This makes the suggested approach also future proof as we already know that possible new requirements can be addressed. Access to the metamodel and generators also gives possibilities to adapt the support for company specific needs - as described in [14]. We are also aware of a case in developing ADAS systems in which modeling support of EAST-ADL is extended with concepts like Safety Measures.

VI. Conclusions

We have presented a model-based approach for integrating safety and security concerns with the rest of system development. This is achieved via an integrated modeling language that provides modeling support for safety and security at language level similarly we have got used to with traditional system and software modeling languages. We demonstrated our approach been applied with practical examples. The benefits of integration include:

- Collaborative development: Access to the system design as well as to security and safety aspects enables collaboration with fast feedback loop.

- Trace and analysis are possible — even at modeling time — between different aspects of the developed system.

- All work items can be versioned together. There is no need to work with possible different formats, versioning systems, or collect data from different sources to get a complete picture.

- Automated analysis and transformations: combined models can be used as input for checking and model transformations, like automatically producing initial safety models for the currently planned system design.

- Security and safety design becomes tightly related to system designs sharing the same model structure.

With EAST-ADL and its extensions these benefits are already available [2], but the same principles can be applied if other modeling languages would be extended. For example, extend SysML instead of EAST-ADL, or add metamodel of RAAML from [11] instead of the dependability metamodel of EAST-ADL.

References

[1] Bergler, M. et al. Social Engineering Exploits in Automotive Software Security: Modeling Human-targeted Attacks with SAM. Proceedings of the 31st European Safety and Reliability Conference (ESREL 2021), 2021

[2] EAST-ADL, 2021, http://www.east-adl.info/Specification.html [Accessed 21 April 2022].

[3] First, Common Vulnerability Scoring System version 3.1: Specification Document, 2019, https://www.first.org/cvss/specification-document [Accessed 21 April 2022].

[4] ISO Functional Safety, 26262-1, 2018

[5] ISO/SAE 21434:2021 - "Road vehicles - Cybersecurity engineering" https://www.iso.org/standard/70918.html [Accessed 16 May 2022]

[6] Kelly, S., Tolvanen, J.-P., Domain-Specific Modeling: Enabling full code generation, Wiley-IEEE Computer Society Press, 2008

[7] Kritzinger, D., Fault tree analysis, in Aircraft System Safety, Elsevier, 2017

[8] Lee, W. S., Grosh, D. L., Tillman, F. A., Lie C. H., Fault Tree Analysis, Methods, and Applications - A Review. IEEE Transactions on Reliability, Volume: R-34, Issue: 3, Aug. 1985.

[9] MetaCase, MetaEdit+ User's Guide. [Online]. Available at: https://metacase.com/support/55/manuals/, 2018 [Accessed 21 April 2022].

[10] Moody, D., The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering, IEEE Transactions on Software Engineering, vol. 35, no. 6, 2009.

[11] OMG, Risk Analysis and Assessment Modeling Language (RAAML), https://www.omg.org/spec/RAAML/1.0/Beta1/PDF, 2021 [Accessed 21 Jan 2022].

[12] OMG, System Modeling Language, version 1.6. [online] Available at: https://www.omg.org/spec/SysML/, 2019 [Accessed 21 April 2022]

[13] Reifer, D., Software Failure Modes and Effects Analysis, IEEE Transactions on Reliability, Volume: R-28, Issue: 3, 1979.

[14] Sari, B., Fail-Operational Safety Architecture for ADAS/AD Systems and a Model-driven Approach for Dependent Failure Analysis. Springer, 2020.

[15] SEI, Architecture Analysis and Design Language (AADL), https://www.sei.cmu.edu/our-work/projects/display.cfm?customel_datapageid_4050=191439,191439 [Accessed 13 May 2022]

[16] Zoppelt, M. Tavakoli Kolagari, R., SAM: A Security Abstraction Model for Automotive Software Systems, ISSA/CSITS@ESORICS, 2018.

[17] Zoppelt, M., Tavakoli Kolagari, R., UnCle SAM: Modeling Cloud Attacks with the Automotive Security Abstraction Model, 2019.

Automotive Software Security Engineering based on the ISO 21434

Technical Report

Matthias Bergler and Ramin Tavakoli Kolagari

May 6, 2023

1 Introduction

The first chapter of the paper provides an overview of the importance of automotive security and the increasing threat posed by cyberattacks. It highlights the critical role of ISO 21434, a standard for automotive cybersecurity engineering, in ensuring the security of automotive systems throughout their lifecycle. The chapter also introduces the Security Abstraction Model (SAM), a framework for automotive system design and development, and discusses the need to integrate ISO 21434 with SAM to enable effective implementation of cybersecurity measures. Finally, the chapter outlines the goals and structure of this work, which aims to explore the practical application of integrating ISO 21434 into SAM and provide insights into the benefits and challenges of this approach.

1.1 Motivation Security and Automotive

The automotive industry has been rapidly evolving with the advent of connected and autonomous vehicles. While this has brought about many benefits, such as improved safety features and enhanced driving experiences, it has also raised concerns about the security of these advanced vehicles. The increasing reliance on software, communication networks, and data exchange in modern cars has made them vulnerable to various cybersecurity threats, including unauthorized access, data breaches, and remote manipulation Wang et al. (2021); Luo et al. (2021). As a result, the motivation for security in the automotive industry has become a critical concern for automakers, regulators, and consumers alike.

One of the primary motivations for security in the automotive industry is to protect the safety and privacy of vehicle owners and passengers. As connected vehicles collect and transmit large amounts of data, including personal information, location data, and driving behavior, there is a heightened risk of data breaches and privacy violations. Malicious actors could exploit vulnerabilities in the vehicle's software or communication networks to gain unauthorized access to sensitive data, leading to identity theft, financial fraud, and other serious consequences. Ensuring the security of connected vehicles is crucial to safeguard the privacy and personal safety of the users Schoettle and Sivak (2014).

Another motivation for security in the automotive industry is to prevent unauthorized access and tampering of the vehicle's critical systems, such as the engine, brakes, and steering. As connected vehicles are equipped with multiple electronic control units (ECUs) that communicate with each other, there is a risk of cyber attacks that could compromise the integrity and functionality of these systems. For example, hackers could remotely manipulate the vehicle's controls, leading to potential accidents or even life-threatening situations. Therefore, robust security measures are needed to prevent unauthorized access and tampering of critical systems in connected vehicles, ensuring their safe operation Guan et al. (2022).

Furthermore, protecting the reputation and brand image of automakers is another motivation for security in the automotive industry. A successful cyber attack on a connected vehicle could not only result in financial losses due to recalls, lawsuits, and damages but also have long-term consequences for the automaker's reputation. Consumers trust that their vehicles are secure and safe to use, and any breach of that trust could significantly impact the brand's image and customer loyalty. Therefore, automakers have a strong motivation to invest in robust security measures to protect their reputation and brand image Turel et al. (2007).

Moreover, regulatory requirements and industry standards also serve as a motivation for security in the automotive industry. Governments and regulatory bodies around the world are increasingly enacting laws and regulations to address the cybersecurity risks associated with connected vehicles. For instance, the European Union's General Data Protection Regulation (GDPR) includes provisions for data protection and privacy in connected vehicles Andraško et al. (2021); Hamulák et al. (2021), and the United States' National Highway Traffic Safety Administration (NHTSA) has issued guidelines for cybersecurity in vehicles Park and Choi (2020); Das et al. (2019). Additionally, industry organizations, such as the Society of Automotive Engineers (SAE) and the Automotive Information Sharing and Analysis Center (Auto-ISAC), have developed cybersecurity best practices and standards for the automotive industry Al-Jarrah et al. (2019). Compliance with these regulations and standards is essential for automakers to meet legal requirements and demonstrate their commitment to security.

The increasing connectivity and autonomy of vehicles have brought about a growing motivation for security in the automotive industry. Protecting the safety and privacy of vehicle users, preventing unauthorized access and tampering of critical systems, safeguarding the reputation of automakers, and ensuring compliance with regulatory requirements and industry standards are all key motivations for enhancing security in the automotive industry. As the reliance on software, communication networks, and data exchange in vehicles continues to grow, it is imperative for automakers to prioritize cybersecurity measures to mitigate potential risks and protect the integrity, safety, and privacy of connected vehicles and their users.

1.2 Importance of integrating ISO 21434 into SAM

The Security Abstraction Model (SAM) is a comprehensive approach used by the automotive industry to manage cybersecurity risks in vehicles. It provides a structured framework for identifying, assessing, and mitigating cybersecurity threats throughout the vehicle's lifecycle. With the emergence of ISO 21434 Macher et al. (2020), a standard specifically focused on cybersecurity engineering in road vehicles, integrating ISO 21434 into SAM becomes crucial for ensuring robust cybersecurity practices in the automotive industry for the following reasons:

1. Standardization and Consistency: ISO 21434 provides a globally recognized standard for managing cybersecurity risks in the automotive industry. By integrating ISO 21434 into SAM, automakers can ensure that their cybersecurity practices are standardized and consistent across the organization. This promotes uniformity in cybersecurity processes, methodologies, and documentation, making it easier to manage and assess cybersecurity risks consistently throughout the vehicle's lifecycle Macher et al. (2020).

2. Comprehensive Risk Management: ISO 21434 emphasizes the importance of identifying, assessing, and mitigating cybersecurity risks throughout the entire development and operational lifecycle of vehicles. By integrating ISO 21434 into SAM, automakers can establish a comprehensive risk management approach that covers all stages of the vehicle's lifecycle, from design and development to production, operation, and maintenance. This ensures that cybersecurity risks are effectively managed at every stage, reducing the likelihood of potential cybersecurity incidents Macher et al. (2020).

3. Secure Development Practices: ISO 21434 promotes the integration of cybersecurity into the entire development process of automotive systems. By integrating ISO 21434 into SAM, automakers can ensure that secure development practices are followed consistently across all stages of the vehicle's lifecycle. This includes secure coding, secure configuration management, and secure

software supply chain management, ensuring that cybersecurity is considered at every step of the development process Macher et al. (2020).

4. Testing and Validation: ISO 21434 emphasizes the importance of rigorous testing and validation of automotive systems to identify and fix potential cybersecurity vulnerabilities. By integrating ISO 21434 into SAM, automakers can establish a robust testing and validation process that includes vulnerability scanning, penetration testing, and security assessments. This helps in identifying and fixing cybersecurity vulnerabilities before the vehicles are deployed, reducing the risk of potential cyber attacks Macher et al. (2020).

5. Conformance to Regulations and Standards: ISO 21434 is gaining increasing recognition and adoption by regulatory bodies and industry organizations as a standard for managing cybersecurity risks in the automotive industry. By integrating ISO 21434 into SAM, automakers can ensure compliance with industry regulations and standards related to automotive cybersecurity. This includes requirements such as the UN Regulation No. 155 Cybersecurity and Software Updates, which mandates compliance with ISO 21434. Compliance with industry regulations and standards helps automakers demonstrate their commitment to cybersecurity and enhances the overall cybersecurity posture of the vehicles Macher et al. (2020).

Integrating ISO 21434 into the Security Abstraction Model (SAM) is essential for the automotive industry to ensure robust and consistent cybersecurity practices. It promotes standardization, comprehensive risk management, secure development practices, testing and validation, and compliance with industry regulations and standards. By incorporating the guidelines outlined in ISO 21434 into SAM, automakers can strengthen their cybersecurity posture and mitigate cybersecurity risks effectively in vehicles Macher et al. (2020).

1.3 Study Contributions

The goal of this paper is to provide an overview of the benefits, considerations, challenges, and practical applications of integrating the ISO 21434 standard into the existing Security Abstraction Model (SAM) processes for the automotive industry. Specifically, our research aims to:

1. Explain relevance and necessities of integrating ISO 21434 into SAM: Our efforts will provide an overview of the benefits of integrating ISO 21434 into SAM, which include improving the quality and safety of software in vehicles, increasing customer satisfaction, and reducing costs associated with software defects and security breaches.

2. Provide an overview of the key considerations and challenges of the full support of ISO 21434 by SAM: We will discuss the scope of ISO 21434 and how it fits within the SAM framework. It will also provide an overview of the key processes and activities in SAM that need to be adapted or augmented to integrate ISO 21434 effectively. Additionally, we will examine the specific challenges involved in integrating ISO 21434 into SAM, such as resistance to change, lack of resources, and the need to train personnel on new processes and tools.

3. Demonstrate the applicability of SAM that fully supports the ISO 21434: We will provide an automotive system model (the braking system) together with an attack model based on the latest SAM version reflecting various aspects of the standard considering nine process steps (of a total of eleven process steps defined by the standard).

2 Background

The second chapter discusses the growing importance of automotive cybersecurity in the context of the increasing use of software and connectivity in modern vehicles. The chapter introduces ISO 21434, a standard for automotive cybersecurity engineering that provides guidelines for the development and validation of secure automotive systems and the Security Abstraction Modell (SAM) and its role in facilitating the integration of various components and systems within a vehicle. The chapter concludes by emphasizing the need for integrating ISO 21434 with SAM to ensure the security of automotive systems throughout their lifecycle and to enable effective implementation of cybersecurity measures.

2.1 ISO 21434

ISO 21434 is a standard that specifically focuses on cybersecurity engineering in road vehicles. It was published in 2020 by the International Organization for Standardization (ISO) as ISO 21434:2020 - "Road vehicles – Cybersecurity engineering"Macher et al. (2020). This standard provides a comprehensive framework for managing cybersecurity risks in the automotive industry, with the ultimate goal of ensuring the security and integrity of vehicles and protecting them from cyber threats.

The objectives of ISO 21434 are to provide guidance and requirements for integrating cybersecurity engineering practices into the entire lifecycle of road vehicles. The standard aims to establish a systematic and proactive approach to cybersecurity, covering all stages of vehicle development, production, operation, and maintenance. ISO 21434 provides guidelines for managing cybersecurity risks, establishing secure development practices, conducting testing and validation, and ensuring compliance with regulations and standards.

The main objectives of ISO 21434 can be summarized as follows:

1. Cybersecurity Risk Management: ISO 21434 emphasizes the importance of identifying, assessing, and mitigating cybersecurity risks throughout the entire lifecycle of road vehicles. It provides guidance on establishing a systematic and proactive approach to cybersecurity risk management, including risk identification, risk assessment, risk mitigation, and risk monitoring. The objective is to ensure that cybersecurity risks are effectively managed throughout the entire lifecycle of the vehicle, reducing the likelihood of potential cybersecurity incidents.

2. Secure Development Practices: ISO 21434 promotes the integration of cybersecurity into the entire development process of automotive systems. It provides guidelines for establishing secure development practices, including secure coding, secure configuration management, and secure software supply chain management. The objective is to ensure that cybersecurity is considered at every step of the development process, reducing the risk of potential cybersecurity vulnerabilities in the final product.

3. Testing and Validation: ISO 21434 emphasizes the importance of rigorous testing and validation of automotive systems to identify and fix potential cybersecurity vulnerabilities. It provides guidance on conducting vulnerability scanning, penetration testing, and security assessments to identify and address potential cybersecurity risks. The objective is to ensure that vehicles are thoroughly tested and validated for cybersecurity before they are deployed, reducing the risk of potential cyber attacks.

4. Compliance with Regulations and Standards: ISO 21434 recognizes the importance of compliance with industry regulations and standards related to automotive cybersecurity. It provides guidance on ensuring compliance with applicable regulations and standards, such as UN Regulation No. 155 Cybersecurity and Software UpdatesUNECE (2021a). The objective is to ensure that vehicles meet the requirements of relevant regulations and standards, demonstrating the commitment to cybersecurity and enhancing the overall cybersecurity posture of the vehicles.

ISO 21434 is a standard that aims to provide a comprehensive framework for managing cybersecurity risks in the automotive industry. Its objectives include cybersecurity risk management, integration of secure development practices, testing and validation, and compliance with regulations and standards. By following the guidelines outlined in ISO 21434, the automotive industry can establish robust cybersecurity practices and ensure the security and integrity of vehicles in the face of evolving cyber threats.

2.2 Relationship between ISO 21434 and Vehicle Automation

As vehicle automation continues to advance, with the development of autonomous and semi-autonomous vehicles, the need for robust cybersecurity measures becomes increasingly critical. Vehicle automation relies heavily on software, connectivity, and data processing capabilities, which can be vulnerable to cyber threats. Therefore, integrating cybersecurity into vehicle automation systems is crucial to ensure the safety, security, and reliability of these vehicles Macher et al. (2020).

ISO 21434:2020, titled ”Road vehicles – Cybersecurity engineering,” is a standard that provides guidance for managing cybersecurity risks in the automotive industry, including vehicles with automation capabilities. This standard is designed to address the unique cybersecurity challenges associated with vehicle automation and provides a framework for incorporating cybersecurity into the development, production, operation, and maintenance of these vehicles.

The relationship between ISO 21434 and vehicle automation can be understood in the following key aspects:

1. Cybersecurity Risk Management for Vehicle Automation: ISO 21434 emphasizes the need for a systematic approach to managing cybersecurity risks throughout the entire lifecycle of road vehicles, including those with automation capabilities. This involves identifying and assessing potential cybersecurity risks associated with vehicle automation, such as unauthorized access, data tampering, and remote manipulation of vehicle functions. It also requires developing appropriate mitigation measures to minimize the risk of cybersecurity threats impacting the safe operation of automated vehicles.

2. Secure Development Practices for Vehicle Automation: ISO 21434 provides guidelines for establishing secure development practices for automotive systems, including those related to vehicle automation. It emphasizes the importance of integrating cybersecurity into the entire development process, including secure coding practices, secure configuration management, and secure software supply chain management. This involves implementing secure coding standards and best practices specifically tailored for vehicle automation, securing communication channels between vehicle automation components, and ensuring the integrity and security of software components and updates used in automated systems.

3. Testing and Validation for Vehicle Automation: ISO 21434 highlights the need for rigorous testing and validation of automotive systems, including those related to vehicle automation. It provides guidelines for conducting vulnerability scanning, penetration testing, and security assessments to assess the security of automated systems. This involves testing the functionality and security of automated features, identifying potential vulnerabilities and weaknesses, and validating the effectiveness of cybersecurity mitigation measures in automated systems before deployment.

4. Compliance with Regulations and Standards for Vehicle Automation: ISO 21434 emphasizes the importance of complying with relevant regulations and standards related to automotive cybersecurity, including those that apply to vehicle automation. This involves understanding and adhering to regulations and standards that specifically address cybersecurity in automated vehicles, such as UN Regulation No. 156 Software Update Processes and Management Systems UNECE (2021b). Compliance with these regulations and standards can help ensure that automated vehicles meet the required cybersecurity requirements and operate securely and safely.

5. Documentation and Traceability for Vehicle Automation: ISO 21434 underscores the need for comprehensive documentation and traceability of cybersecurity-related activities, including those associated with vehicle automation. This involves maintaining clear and traceable records of risk assessments, development practices, testing results, and compliance evidence specifically related to vehicle automation. Documentation and traceability are essential for audit and review purposes, as they provide evidence of compliance and accountability in ensuring the cybersecurity of automated vehicles.

ISO 21434 provides guidance for integrating cybersecurity into the development, production, operation, and maintenance of vehicles with automation capabilities. It emphasizes the importance of cybersecurity risk management, secure development practices, testing and validation, compliance with regulations and standards, and documentation and traceability, specifically tailored for vehicle automation. By following the guidelines provided by ISO 21434, automotive stakeholders can enhance the cybersecurity posture of automated vehicles and ensure their safe and secure operation.

2.3 EAST-ADL

The Architecture Analysis and Design Language (EAST-ADL) is a modeling language used in the automotive and embedded systems domain for describing the architecture and design of complex automotive systems. It provides a comprehensive and systematic approach for modeling the software and hardware components, their interactions, and their relationships with the environment in which they operate. EAST-ADL enables engineers to capture the architectural design decisions, analyze the system's behavior, and support the development of automotive systems with higher quality, safety, and efficiency Cuenot et al. (2010).

EAST-ADL was developed by a consortium of European automotive manufacturers, suppliers, and research institutes as part of the European research project "Model-Based Analysis and Design of Novel Architectures for Dependable Electric Vehicles" (MAENAD) Manead (2021). The project aimed to develop a modeling language and associated analysis techniques to support the design of advanced embedded control systems, particularly in the automotive domain.

Some key features of EAST-ADL include:

1. Comprehensive Modeling Capabilities: EAST-ADL provides a rich set of modeling concepts and notations for describing the architecture and design of automotive systems. It allows engineers to model the system's structure, behavior, and interactions, including the software and hardware components, their interfaces, their connections, their modes of operation, and their communication and data exchange mechanisms Cuenot et al. (2010).

2. Support for Functional and Non-functional Requirements: EAST-ADL allows engineers to capture both functional and non-functional requirements of automotive systems. This includes modeling the system's functionality, performance, safety, reliability, and other quality attributes. EAST-ADL also supports the modeling of timing, resource allocation, and other system-level constraints Cuenot et al. (2010).

3. Analysis and Simulation Capabilities: EAST-ADL provides analysis and simulation capabilities that enable engineers to evaluate the behavior and performance of the system at the architectural level. This includes support for model checking, simulation, and other analysis techniques to detect potential design flaws, performance bottlenecks, and other issues early in the development process Cuenot et al. (2010).

4. Integration with Other Modeling Languages: EAST-ADL is designed to be compatible and interoperable with other modeling languages commonly used in the automotive domain, such as the AUTOSAR (AUTomotive Open System ARchitecture) standard. This allows engineers to integrate EAST-ADL models with models developed in other modeling languages, facilitating system-level analysis and simulation Cuenot et al. (2010).

5. Tool Support: Several modeling tools and frameworks are available that support the use of EAST-ADL, providing engineers with a graphical user interface and automated analysis capabilities. These tools help streamline the modeling and analysis process, making it more efficient and effective Cuenot et al. (2010). The modeling tool MetaEdit+ Tolvanen and Rossi (2003) has a full integration of the EAST-ADL as well as a full integration of SAM. So a wholesome modeling process is ensured.

EAST-ADL is a powerful modeling language used in the automotive and embedded systems domain for describing the architecture and design of complex systems. It provides comprehensive modeling capabilities, supports functional and non-functional requirements, offers analysis and simulation capabilities, integrates with other modeling languages, and has tool support, making it a valuable tool for automotive system design Cuenot et al. (2010).

2.4 SAM

The automotive industry is undergoing a significant transformation with the rapid advancement of software-intensive technologies, such as connected cars, autonomous driving, and electric vehicles. However, this increasing complexity and connectivity also bring about new challenges in terms of cybersecurity, as vehicles become vulnerable to various cyber threats, including hacking, data breaches, and remote manipulation.

In response to these challenges, the Security Abstraction Model (SAM) Zoppelt and Tavakoli Kolagari (2019); Bergler et al. (2021); Bergler, Tolvanen, and Kolagari (Bergler et al.) has emerged as a significant concept in the automotive industry. SAM provides a structured approach to model, analyze, and enforce security aspects in the development of automotive software systems. It aims to improve the security assurance, streamline the development process, and enhance the resilience of vehicles against security threats.

The importance of SAM in the automotive industry can be highlighted in several key aspects:

1. Comprehensive Security Management: SAM offers a comprehensive framework for managing security concerns in automotive software development. It provides a systematic approach to identify, analyze, and mitigate security risks throughout the entire development lifecycle of vehicles, from design and implementation to testing and deployment. SAM helps automotive manufacturers and suppliers to effectively manage security aspects in their software systems, ensuring that the vehicles are protected against potential cyber threats.

2. Standardization and Consistency: SAM provides a standardized and consistent approach to modeling and analyzing security aspects in automotive software systems. It offers a common language and methodology for describing security threats, security architectures, and security analysis techniques. This standardization enables better collaboration among different stakeholders in the automotive industry, such as automakers, suppliers, and security experts, and helps to ensure a consistent and unified approach to security across different automotive systems.

3. Enhanced Security Assurance: SAM supports the development of more secure software systems by providing a structured approach to security assurance. It enables the identification of potential security risks early in the development process, allowing for timely mitigation measures to be implemented. SAM also provides a way to systematically verify and validate the security measures in place, improving the overall security assurance of automotive software systems.

4. Improved Efficiency and Resilience: SAM helps streamline the development process of automotive software systems by providing a structured approach to security. It reduces the complexity and ambiguity of security considerations, making it easier for developers to integrate security measures into their software designs. This improved efficiency allows for more effective development of secure software systems, reducing the risk of security breaches and enhancing the resilience of vehicles against threats.

5. Compliance with Industry Standards: SAM aligns with industry standards and guidelines for automotive security, such as ISO 21434, AUTOSAR, and EAST-ADL, which are widely adopted in the automotive industry. By incorporating SAM into the development process, automotive manufacturers and suppliers can ensure compliance with these standards and guidelines, demonstrating their commitment to security best practices.

SAM plays a crucial role in addressing the security challenges faced by the automotive industry. It provides a structured approach to model, analyze, and enforce security aspects in the development of automotive software systems, leading to improved security assurance, streamlined development process, and enhanced resilience against threats.

3 Case Study Braking System

The following scenario is described as an example of the use of SAM and the new features of ISO 21434: Imagine a situation where a hacker gains unauthorized access to the data communication network of a car's braking system. The hacker then intentionally interferes with the communication signals between the car's brake control module and the wheel speed sensors. As a result, the brake system receives incorrect information about the speed of the wheels and the car's braking distance. This causes the brake system to malfunction and fail to engage, even when the driver presses the brake pedal. Consequently, the car is unable to slow down or stop in time, and it collides with another vehicle or object, causing an accident. In this scenario, the hacker's malicious actions disrupt the normal operation of the car's braking system, creating a dangerous situation that puts the driver, passengers, and other road users at risk of injury or death.

3.1 EAST-ADL System Model of the Braking System

The EAST-ADL is an architecture description language that models the core of a system and has manifold complementary models that capture additional information, such as timing, variability, dependability and so on. The modeling of the core has two main and widely appreciated features in practice: first, the system model is described on predefined abstraction levels, starting with an abstract feature modeling as well as two architecture description levels, where the analysis level is a purely functional logical description and the design level already includes a description of hardware and software; second, the component-oriented approach to architecture description is based on a type/prototype concept, which allows for the reuse of components from a component library.

In this Figure 1 we see a small excerpt from the abstract level (the Vehicle Level) and the first architecture description level (the Analysis Level). The item "BrakeByWire", which is to be understood in accordance with the item definition from the standard ISO 26262, refers in this case to the vehicle feature "BrakeByWire" from the Vehicle Level and is realized by the function "BBW_FAA" from the Analysis Level. We can see the internal components of the function "BBW_FAA" in the lower part of the figure, consisting of a sensor "BrakePedal" and four actuators "ABS" (one actuator for each wheel) and a control unit.

3.2 Security Model of the Braking System

As previously discussed, the integration of the ISO 21434 standard into the Security Abstraction Model SAM has many advantages for the automotive industry. The integration of the ISO made some demands on the meta model. The following addressed points from the ISO therefore had to be checked or integrated to complete SAM:

1. Item Definition (ISO 21434 Section 9.3): The item definition is a central component of ISO 21434 and describes the definition and specification of safety-critical elements in a vehicle. It is about gaining a comprehensive understanding of the functions and characteristics of these elements in order to identify and eliminate potential vulnerabilities in the system. The item definition includes a comprehensive analysis of the safety-critical elements, including their functions, interfaces, data flows, requirements, and risks. It also considers the interaction of the elements with each other and with other systems. The item definition is a crucial step in the process of automotive cybersecurity development, ensuring that the systems meet the security requirements and are protected against potential threats.

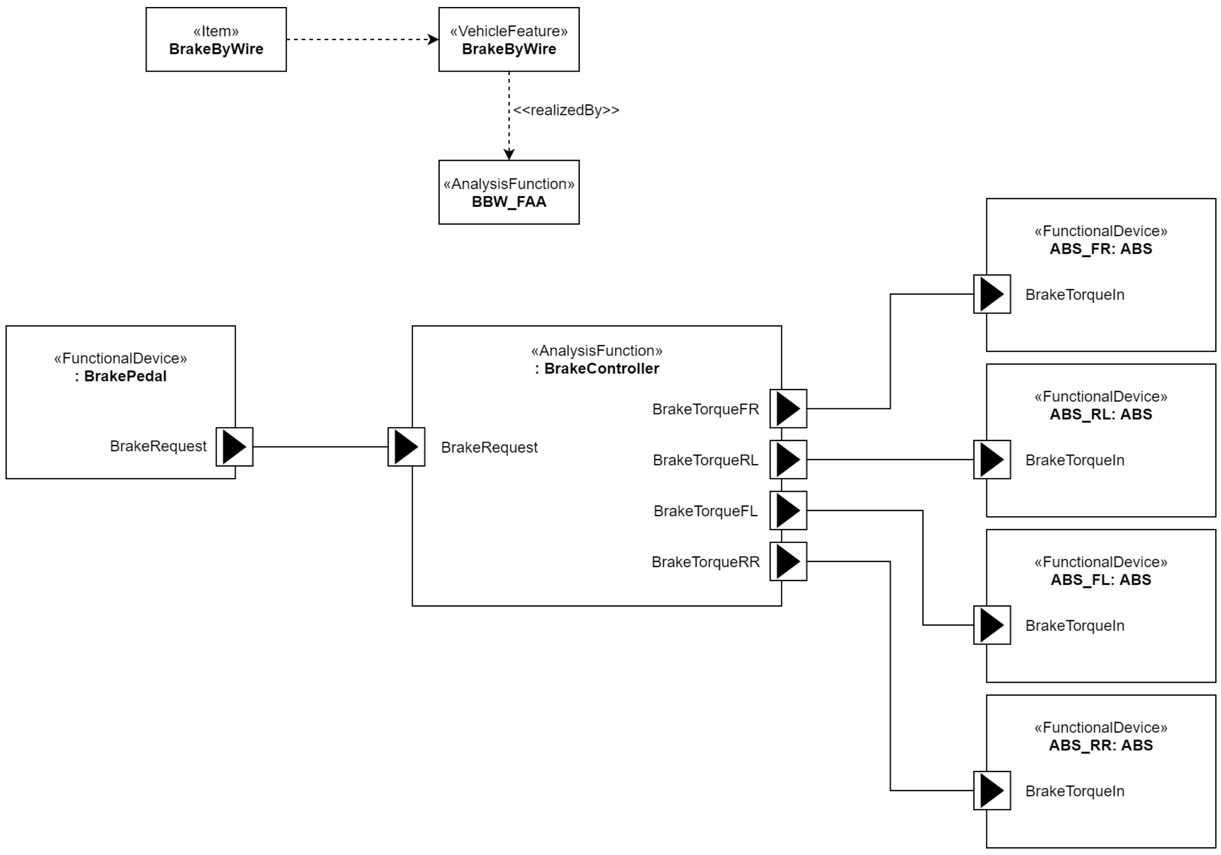


Figure 1: This figure shows the Analysis Architecture of the braking function based on EAST-ADL.

2. Asset Identification (ISO 21434 Section 15.3): Asset Identification is another critical component of ISO 21434 that plays a crucial role in the development of secure automotive systems. Asset Identification refers to the process of identifying the assets that need to be protected within the system. These assets may include hardware, software, data, or other resources that are critical to the proper functioning of the system. The goal of Asset Identification is to gain a comprehensive understanding of the assets and their characteristics in order to properly assess the risks associated with each asset and develop effective security measures. This process involves a detailed inventory of all assets and their associated risks, as well as the establishment of asset classifications and their criticality levels. By properly identifying and classifying assets, the development team can better understand the security requirements of the system and ensure that adequate security measures are implemented to protect the system against potential cyber threats.

3. Identification of Threat Scenarios (ISO 21434 Section 15.4): Threat scenarios are a critical part of ISO 21434 and involve identifying potential cybersecurity threats that could impact automotive systems. This includes identifying potential attack vectors and the potential impacts of a successful cyberattack. By using threat scenarios, the development team can develop more effective security measures and reduce the risk of a successful cyberattack. They are an essential part of ensuring the safety and security of automotive systems.

4. Impact Rating (ISO 21434 Section 15.5): Impact Rating is a process in ISO 21434 used to evaluate the potential impact of a cybersecurity threat on the system by assigning a severity level to each potential threat based on its potential harm and feasibility of mitigation. This helps prioritize security efforts and allocate resources accordingly to ensure the safety and security of automotive systems.

5. Attack Path Analysis (ISO 21434 Section 15.6): In the context of ISO 21434, Attack Path Anal-

ysis is an important tool for identifying potential cybersecurity threats to automotive systems and developing appropriate security measures to mitigate those threats. It is a critical component of the risk management process outlined in the standard and is used to help ensure the safety and security of automotive systems throughout their lifecycle.

6. Attack Feasibility Rating (ISO 21434 Section 15.7): The Attack Feasibility Rating is a process within ISO 21434 that evaluates the likelihood of a successful cybersecurity attack by assessing the attacker's resources and access to information. It helps prioritize security measures and allocate resources accordingly, based on the most significant threats to the system. This process helps to ensure the safety and security of automotive systems.

7. Risk Value Determination (ISO 21434 Section 15.8): ISO 21434 includes a process called risk value determination, which is used to evaluate the level of risk posed by cybersecurity threats to automotive systems. This process assigns a numerical value to each threat and helps the development team prioritize security measures and allocate resources effectively. By using risk value determination, the development team can identify and address the most significant threats to the system's safety and security, ensuring appropriate security measures are implemented. Overall, risk value determination is a vital aspect of ISO 21434, supporting the safe and secure development of automotive systems.

8. Risk Treatment Decision (ISO 21434 Section 15.9): The ISO 21434 standard outlines a process for making decisions about how to treat cybersecurity risks in automotive systems. This involves selecting and implementing security measures to reduce the risks to an acceptable level. The purpose of this process is to prioritize security measures and allocate resources effectively.

9. Cybersecurity Goals (ISO 21434 Section 9.4) [WP-09-03 & RQ-09-07]: The ISO 21434 standard defines cybersecurity goals for automotive systems. These goals help ensure that the system is secure against potential threats and that any vulnerabilities are identified and addressed appropriately. The cybersecurity goals specified in the standard are designed to be flexible and adaptable to different use cases and system architectures. By setting clear goals for cybersecurity, the standard aims to promote a more systematic and comprehensive approach to cybersecurity in the automotive industry.

10. Cybersecurity Claims (ISO 21434 Section 9.4) [WP-09-04 & RQ-09-06]: The cybersecurity claims defined in the ISO 21434 standard refer to the features and functions of automotive systems that provide cybersecurity protection. These claims serve as a means of communicating the level of cybersecurity protection to customers and stakeholders. To ensure that cybersecurity claims are accurate and reliable, the standard requires that they be based on evidence and that they are verifiable. This helps ensure that customers and stakeholders can make informed decisions about the security of the system. The standard also provides guidance on how to test and verify cybersecurity claims to ensure that they are accurate and reliable.

11. Cybersecurity Concept (ISO 21434 Section 9.5): The Cyber Security Concept is a description of the security objectives and measures that apply to an automotive system, as defined in the ISO 21434 standard. It serves as a high-level guide for the development of effective cybersecurity measures and is regularly reviewed and updated throughout the development process.

After making adjustments to the meta model, the previously created scenario has now been replicated as a security model (see Figure 2. This security model focuses on the "BrakeByWire" feature, which interfaces with the EAST-ADL in a vehicle. With the integration of ISO 21434 in SAM, new scores such as "AttackFeasibility", "ImpactRating", and "RiskValue" can now be calculated in addition to the Common Vulnerability Scoring Systems (CVSS) Mell et al. (2006) Base and Temporal score. These scores are essential for assessing the level of risk posed by potential security threats and vulnerabilities in the system. With the ability to calculate these scores in SAM, the system can be analyzed comprehensively and appropriate security measures can be implemented to mitigate potential risks and ensure the safety and security of the system and its users.

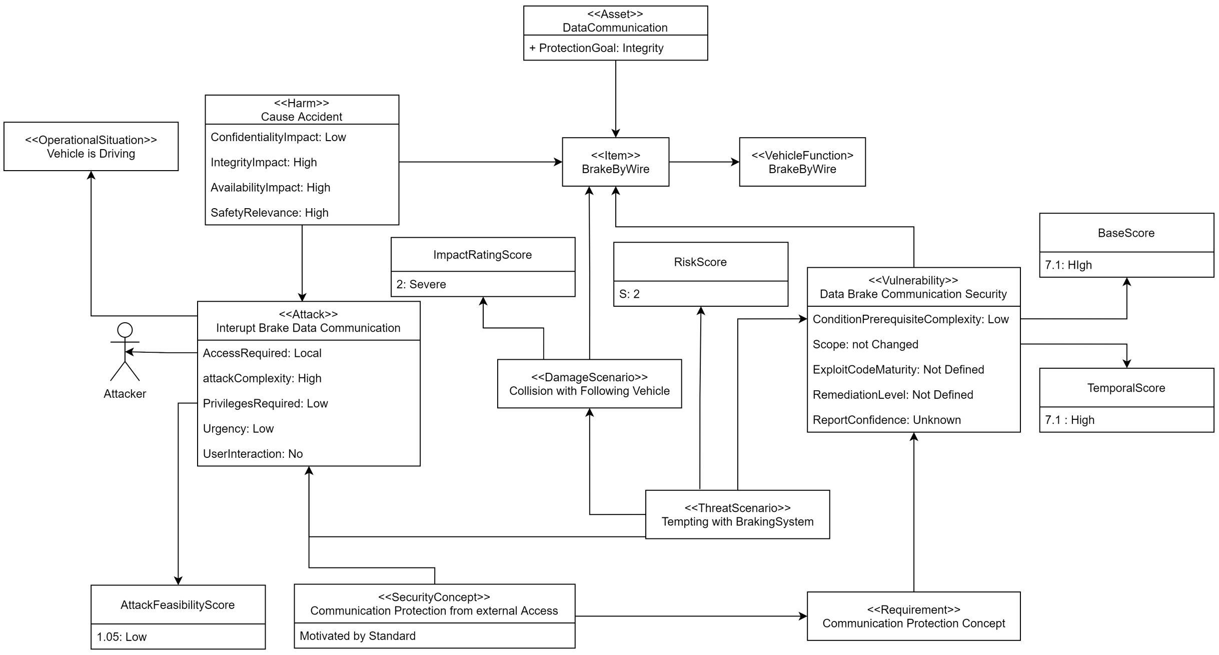


Figure 2: This figure shows the Security Model of the braking system based on SAM.

3.3 Scoring Calculation

Thanks to the successful integration of ISO 21434, it is not only possible to evaluate vulnerabilities, but also attacks and their impact on a system. For this purpose, new scores are introduced in the metamodel based on the ISO 21434 standard:

1. The Common Vulnerability Scoring System (CVSS) is a framework used to evaluate and measure the severity of security vulnerabilities. It provides a consistent and standardized way to assess the potential impact of a vulnerability and assign a score, which can be used to prioritize and plan security measures. CVSS was developed by the Forum of Incident Response and Security Teams (FIRST) and is widely used by security professionals and organizations to evaluate and communicate the severity of security vulnerabilities. This score is already integrated into SAM Bergler, Tolvanen, and Kolagari (Bergler et al.).

2. AttackFeasibilityScore: The AttackFeasibilityScore refers to the attack feasibility rating from the standard. This describes the feasibility of an attack on our system. The calculation basis for this is the already implemented CVSS score. According to CVSS, the ratings are mapped to numbers and used in the corresponding formula from the standard. The new formula is ($E = 8.22xVxCxPxU$) Macher et al. (2020) where E is the exploitability value; V for the value of the attack vector; C for the attack complexity value; P for the value of the privileges required and U for the value of the user interaction. This value can then be mapped back to a textual evaluation based on the standard.

3. ImpactRatingScore: The ImpactRatingScore refers to the impact rating from ISO 21434. This value describes the severity of the consequences of a damage scenario. The impact rating can have the values Negligible (0), Moderate (1), Major (1.5) and Severe (2).

4. RiskScore: The RiskScore refers to the risk value determination from the standard. This value describes the risk that a threat scenario will occur. According to ISO 21434, this value can be determined either using a matrix or using your own calculation formulas. In both cases, the Impact

Rating and the Attack Feasibility Rating are used. In our example, we use the risk matrix provided in the standard.

In our example, the following values result for the respective scores:

- BaseScore: 7.1 High
- TemporalScore: 7.1 High
- AttackFeasibilityScore: 1.05 Low ($E = 8.22x0.55x0.44x0.62x0.85$)
- ImpactRatingScore: 2: Severe (based on the definition in the standard)
- RiskScore: S: 2 (based on the evaluation matrix in the standard)

4 Integrating ISO 21434 into SAM

The integration of the standard into SAM has changed the metamodel and the items Asset, Damage Scenario, Threat Scenario, ImpactRatingScore, RiskScore, AttackFeasibilityRating and AttackFeasibilityScore needed to be added as seen in the Figure 2. The current meta model can be viewed online [1]. The new item asset complements the existing target in the metamodel, which can be both a ”HumanActor” and an ”Item” in the sense of ISO 26262. According to ISO 21434, a damage scenario refers to a hypothetical event or sequence of events that could lead to harm to the vehicle, its occupants, or its surroundings. It takes into account the potential sources of harm, the likelihood of the harm occurring, and the severity of the harm that could result. The purpose of defining damage scenarios is to identify the risks associated with the use of the vehicle and to establish measures to prevent or mitigate the effects of those risks. With the introduction of the new item ”DamageScenario”, the consequences of a successful attack can now be modeled and additionally evaluated by the ”ImpactRatingScore”. A threat scenario, according to ISO 21434, is a hypothetical situation or sequence of events that could lead to a security threat to a vehicle’s functions, components, or data. It includes the potential sources of the threat, the probability of the threat occurring, and the severity of the consequences that could result. The goal of defining threat scenarios is to identify potential security risks and vulnerabilities and to establish measures to prevent or mitigate the effects of those risks. Threat scenarios are an essential part of the risk analysis process in the development of secure vehicles. By integrating the item ”ThreatScenario” it is now possible to describe the attack scenario more precisely. In addition, in combination with the other newly introduced scores, an assessment of the risk for such a scenario can be given using the ”RiskScore” item. Based on these scores, a strategy can now be developed as to which measures are to be taken to avoid them. The already existing item ”Attack” was supplemented by the ”AttackFeasibilityScore”, whereby the feasibility of an attack can be better assessed. Since CVSS is already integrated into SAM, this can be used as a basis for calculations. The new scores are currently calculated manually because there is no tool support for the new SAM version. Thanks to the successful integration, the following points can now be reliably modeled with SAM:

1. Item Definition
2. Asset Identification
3. Identification of Threat Scenarios
4. Impact Rating
5. Attack Path Analysis
6. Attack Feasibility Rating

[1] https://www.in.th-nuernberg.de/professors/BerglerMa/SAM/

7. Risk Value Determination
8. Risk Treatment Decision
9. Cybersecurity Concept

As far as our research suggests, the two outstanding points Cybersecurity Goals and Cybersecurity Claims are already covered via SAM. However, the standard is not clear here and further research is therefore required to confirm the thesis with certainty.

5 Conclusion

By providing a structured approach for cybersecurity management of road vehicles, ISO 21434 can help ensure that cybersecurity considerations are integrated into the development process for automotive software. However, the process of integrating ISO 21434 into SAM also required adjustments to the existing meta model.

This paper has provided an overview of the key considerations when integrating ISO 21434 into SAM, including the benefits and challenges of the integration as well as the practical application of the integration. The paper has highlighted the importance of adapting or augmenting SAM processes and activities to integrate ISO 21434 effectively.

By preparing an example, the paper has demonstrated how integrating ISO 21434 into SAM may improve the quality and security of software in vehicles, which has the potential to result in increased customer satisfaction, improved brand reputation, and reduced costs associated with software defects and security breaches.

Next steps in development include a comprehensive evaluation study, as well as providing appropriate tool support with the new functionality to enable easier use.

The next development steps include both the provision of suitable tool support and an evaluation study with which the use of SAM in the development process is to be evaluated.

MetaEdit+ will continue to be used for tool support, as good experiences have already been made here in advance. Thanks to the integration in MetaEdit+, there is not only complete access to the contents of the EAST-ADL and thus ISO 26262 for functional safety, but also to the security supplement via SAM. In addition, the newly integrated scores from ISO 21434 can be calculated automatically with MetaEdit+, as has already been done with CVSS.

The evaluation study is to be carried out in cooperation with representatives from the industry. The development process of a vehicle component is to be exercised as an example by using the EAST-ADL and SAM using tool support through MetaEdit+. The representatives from the industry should then give an assessment and feedback on the usability and benefits of SAM.

References

Al-Jarrah, O. Y., C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis (2019). Intrusion detection systems for intra-vehicle networks: A review. *IEEE Access 7*, 21266–21289.

Andraško, J., O. Hamul'ák, M. Mesarčík, T. Kerikmäe, and A. Kajander (2021). Sustainable data governance for cooperative, connected and automated mobility in the european union. *Sustainability 13*(19), 10610.

Bergler, M., J.-P. Tolvanen, and R. T. Kolagari. Integrating security and safety with systems engineering: a model-based approach.

Bergler, M., J.-P. Tolvanen, M. Zoppelt, and R. T. Kolagari (2021). Social engineering exploits in automotive software security: Modeling human-targeted attacks with sam. In *31st European Safety and Reliability Conference, ESREL 2021*, pp. 2502–2509.

Cuenot, P., P. Frey, R. Johansson, H. Lönn, Y. Papadopoulos, M.-O. Reiser, A. Sandberg, D. Servat, R. Tavakoli Kolagari, M. Törngren, et al. (2010). The east-adl architecture description language for automotive embedded software. In *Model-Based Engineering of Embedded Real-Time Systems: International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers*, pp. 297–307. Springer.

Das, S., S. R. Geedipally, K. Dixon, X. Sun, and C. Ma (2019). Measuring the effectiveness of vehicle inspection regulations in different states of the us. *Transportation research record 2673*(5), 208–219.

Guan, T., Y. Han, N. Kang, N. Tang, X. Chen, and S. Wang (2022). An overview of vehicular cybersecurity for intelligent connected vehicles. *Sustainability 14*(9), 5211.

Hamulák, O., J. Andraško, and M. Mesarcik (2021). The digital development of the european union: data governance aspects of cooperative, connected and automated mobility. *IDP: revista de Internet, derecho y política= revista d'Internet, dret i política* (34), 7.

Luo, F., Y. Jiang, Z. Zhang, Y. Ren, and S. Hou (2021). Threat analysis and risk assessment for connected vehicles: A survey. *Security and Communication Networks 2021*, 1–19.

Macher, G., C. Schmittner, O. Veledar, and E. Brenner (2020). Iso/sae dis 21434 automotive cybersecurity standard-in a nutshell. In *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops: DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings 39*, pp. 123–135. Springer.

Manead, M. (2021). About.

Mell, P., K. Scarfone, and S. Romanosky (2006). Common vulnerability scoring system. *IEEE Security & Privacy 4*(6), 85–89.

Park, S. and J.-Y. Choi (2020). Malware detection in self-driving vehicles using machine learning algorithms. *Journal of advanced transportation 2020*, 1–9.

Schoettle, B. and M. Sivak (2014). A survey of public opinion about connected vehicles in the us, the uk, and australia. In *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 687–692. IEEE.

Tolvanen, J.-P. and M. Rossi (2003). Metaedit+ defining and using domain-specific modeling languages and code generators. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pp. 92–93.

Turel, H. S., E. M. Yigit, and I. Altug (2007). Evaluation of elderly people's requirements in public open spaces: A case study in bornova district (izmir, turkey). *Building and Environment 42*(5), 2035–2045.

UNECE, U. (2021a, Apr). Un regulation no. 155 - cyber security and cyber security management system.

UNECE, U. (2021b, Apr). Un regulation no. 156 - software update and software update management system.

Wang, Y., Y. Wang, H. Qin, H. Ji, Y. Zhang, and J. Wang (2021). A systematic risk assessment framework of automotive cybersecurity. *Automotive Innovation 4*, 253–261.

Zoppelt, M. and R. Tavakoli Kolagari (2019). Sam: a security abstraction model for automotive software systems. In *Security and Safety Interplay of Intelligent Software Systems: ESORICS 2018 International Workshops, ISSA 2018 and CSITS 2018, Barcelona, Spain, September 6–7, 2018, Revised Selected Papers*, pp. 59–74. Springer.

References

[1] AADL: Architecture analysis and design language (aadl) (2022), https://www.sei.cmu.edu/our-work/projects/display.cfm?customel_datapageid_4050=191439,191439

[2] Aburrous, M., Hossain, M.A., Dahal, K., Thabtah, F.: Experimental case studies for investigating e-banking phishing techniques and attack strategies. Cognitive Computation **2**, 242–253 (2010)

[3] Amendola, S.: Improving automotive security by evaluation—from security health check to common criteria. White paper, Security Research & Consulting GmbH **176** (2004)

[4] Association, E.A.: Common vulnerability scoring system version 3.1: Specification document (2019), http://www.east-adl.info/Specification.html

[5] Bauerdick, H., Gogolla, M., Gutsche, F.: Detecting ocl traps in the uml 2.0 superstructure: An experience report. In: International Conference on the Unified Modeling Language. pp. 188–196. Springer (2004)

[6] Bergler, M., Tolvanen, J.P., Zoppelt, M., Kolagari, R.T.: Social engineering exploits in automotive software security: Modeling human-targeted attacks with sam (2021)

[7] Bergler, M., Tolvanen, J.P., Zoppelt, M., Kolagari, R.T.: Social engineering exploits in automotive software security: Modeling human-targeted attacks with sam. Proceedings of the 31st European Safety and Reliability Conference (ESREL 2021) (2021)

[8] Brenner, J.: Iso 27001 risk management and compliance. Risk management **54**(1), 24–29 (2007)

[9] Campbell, C.C.: Solutions for counteracting human deception in social engineering attacks. Information Technology & People **32**(5), 1130–1152 (2019)

[10] Cheah, M., Nguyen, H.N., Bryans, J., Shaikh, S.A.: Formalising systematic security evaluations using attack trees for automotive applications. In: IFIP International Conference on Information Security Theory and Practice. pp. 113–129. Springer (2017)

[11] Committee, S.I.V.C.S.E., et al.: Sae j3061: Cybersecurity guidebook for cyber-physical vehicle systems (2016)

[12] Costantino, G., La Marra, A., Martinelli, F., Matteucci, I.: Candy: A social engineering attack to leak information from infotainment system. In: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring). pp. 1–5 (2018). https://doi.org/10.1109/VTCSpring.2018.8417879

[13] Cuenot, P., Frey, P., Johansson, R., Lönn, H., Papadopoulos, Y., Reiser, M.O., Sandberg, A., Servat, D., Tavakoli Kolagari, R., Törngren, M., et al.: 11 the east-adl architecture description language for automotive embedded software. In: Model-Based Engineering of Embedded Real-Time Systems: International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers. pp. 297–307. Springer (2010)

[14] Feiler, P.H., Lewis, B.A., Vestal, S.: The sae architecture analysis & design language (aadl) a standard for engineering performance critical systems. In: 2006 ieee conference on computer aided control system design, 2006 ieee international conference on control applications, 2006 ieee international symposium on intelligent control. pp. 1206–1211. IEEE (2006)

[15] FIRST: Architecture analysis and design language (aadl) (2022), https://www.first.org/cvss/specification-document

[16] FIRST.Org, I.: First, common vulnerability scoring system, version 3.1 (2019), https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf

[17] Forbes, B.T.: Tesla in taiwan crashes directly into overturned truck, ignores pedestrian, with autopilot on (2020), https://www.forbes.com/sites/bradtempleton/2020/06/02/tesla-in-taiwan-crashes-directly-into-overturned-truck-ignores-pedestrian-with-autopilot-on/?sh=3ec11c5758e5

[18] Foster, I., Prudhomme, A., Koscher, K., Savage, S.: Fast and vulnerable: A story of telematic failures. In: 9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15) (2015)

[19] Fürst, S., Bunzel, S.: Autosar. Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort pp. 105–122 (2015)

[20] Greenberg, A.: After Jeep Hack, Chrysler Recalls 1.4M Vehicles for Bug Fix. Wired (2015)

[21] Greenberg, A.: The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse. Wired (January 2016)

[22] Greenberg, A.: Radio Attack Lets Hackers Steal 24 Different Car Models. Wired (March 2016)

[23] Greenberg, A.: Hackers can clone millions of Toyota, Hyundai, and Kia keys. Wired (2020)

[24] Health, U.: 5 real-life medical devices inspired by science fiction (2020), https://www.usfhealthonline.com/resources/healthcare/5-real-life-medical-devices-inspired-by-science-fiction/

[25] Hemel, T.: Automated interactive threat analysis of it architectures (2019)

[26] Van den Herrewegen, J., Garcia, F.D.: Beneath the bonnet: A breakdown of diagnostic security. In: European Symposium on Research in Computer Security. pp. 305–324. Springer (2018)

[27] Holz, H.J., Applin, A., Haberman, B., Joyce, D., Purchase, H., Reed, C.: Research methods in computing: what are they, and how should we teach them? ACM SIGCSE Bulletin **38**(4), 96–114 (2006)

[28] Hubaux, J.P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Security & Privacy **2**(3), 49–55 (2004)

[29] Lab, T.K.S.: Experimental security assessment of bmw cars: A summary report (2018)

[30] Lab, T.K.S.: Experimental security research of tesla autopilot (2019)

[31] Macher, G., Armengaud, E., Brenner, E., Kreiner, C.: A review of threat analysis and risk assessment methods in the automotive context. In: International Conference on Computer Safety, Reliability, and Security. pp. 130–141. Springer (2016)

[32] Macher, G., Schmittner, C., Veledar, O., Brenner, E.: Iso/sae dis 21434 automotive cybersecurity standard-in a nutshell. In: Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops: DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings 39. pp. 123–135. Springer (2020)

[33] Matulevičius, R.: Security risk-oriented misuse cases. In: Fundamentals of Secure System Modelling, pp. 93–105. Springer (2017)

[34] MetaCase: The graphical metamodeling example (2018), `https://metacase.com/support/55/manuals/GraphicalMetamodeling.pdf`

[35] MetaCase: Metaedit+ user's guide (2018), `https://metacase.com/support/55/manuals/`

[36] MetaCase: East-adl tutorial (2019)

[37] Microsoft-Corporation: The stride thread model (2005), `https://msdn.microsoft.com/en-us/library/ee823878.aspx`

[38] Moody, D.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on software engineering **35**(6), 756–779 (2009)

[39] Mouton, F., Leenen, L., Venter, H.S.: Social engineering attack examples, templates and scenarios. Computers & Security **59**, 186–209 (2016)

[40] Mouton, F., Leenen, L., Venter, H.: Social engineering attack examples, templates and scenarios. Computers & Security **59**, 186–209 (2016). https://doi.org/https://doi.org/10.1016/j.cose.2016.03.004, `https://www.sciencedirect.com/science/article/pii/S0167404816300268`

[41] n, I..: Road vehicles - cybersecurity engineering (20222019), `https://www.iso.org/standard/70918.html`

[42] News, C.G.B.: Tesla's autopilot 'tricked' to operate without driver (2021), `https://www.bbc.com/news/technology-56854417`

[43] Nie, S., Liu, L., Du, Y.: Free-fall: Hacking tesla from wireless to can bus. Briefing, Black Hat USA **25**, 1–16 (2017)

[44] Nie, S., Liu, L., Du, Y., Zhang, W.: Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars. Briefing, Black Hat USA (2018)

[45] Pattaranantakul, M., He, R., Song, Q., Zhang, Z., Meddahi, A.: Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. IEEE Communications Surveys & Tutorials **20**(4), 3330–3368 (2018)

[46] PurpleSec: 2021 cyber security statistics the ultimate list of stats, data & trends (2021), `https://purplesec.us/resources/cyber-security-statistics/`

[47] SAE, S.: j3061, cybersecurity guidebook for cyber-physical vehicle systems. Nr **1**, 52 (2016)

[48] Salahdine, F., Kaabouch, N.: Social engineering attacks: A survey. Future Internet **11**(4), 89 (2019)

[49] Sion, L., Yskout, K., Van Landuyt, D., Joosen, W.: Risk-based design security analysis. In: Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment. pp. 11–18 (2018)

[50] Sommer, F., Dürrwang, J., Kriesten, R.: Survey and classification of automotive security attacks. Information **10**(4), 148 (2019)

[51] SysML: System modeling language (sysml) (2022), `https://sysml.org/`

[52] Tahaei, M., Vaniea, K., Beznosov, K., Wolters, M.K.: Security notifications in static analysis tools: Developers' attitudes, comprehension, and ability to act on them. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. pp. 1–17 (2021)

[53] ThreatModeler Software Inc, ThreatModeler® Software, I.: Automated threat modeling solution (May 2021), `https://threatmodeler.com/`

[54] Timberg, C.: Hacks on the Highway. Washington Post (July 2015)

[55] UNECE, U.: Un regulation no. 156 - software update and software update management system (Apr 2021), `https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update`

[56] Wang, Z., Zhu, H., Sun, L.: Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods. IEEE Access **9**, 11895–11910 (2021)

[57] Ward, C.D.: Software verification for a custom instrument using vectorcast and codesonar (2011)

[58] Wilke, C., Demuth, B.: Uml is still inconsistent! how to improve ocl constraints in the uml 2.3 superstructure. Electronic Communications of the EASST **44** (2011)

[59] Wolf, M., Weimerskirch, A., Paar, C.: Security in automotive bus systems. In: Workshop on Embedded Security in Cars. pp. 1–13. Citeseer (2004)

[60] Zelkowitz, M.V., Wallace, D.: Experimental validation in software engineering. Information and Software Technology **39**(11), 735–743 (1997)

[61] Zhang, Y., Ge, B., Li, X., Shi, B., Li, B.: Controlling a car through obd injection. In: 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). pp. 26–29. IEEE (2016)

[62] Zoppelt, M., Kolagari, R.T.: Sam: a security abstraction model for automotive software systems. In: Security and Safety Interplay of Intelligent Software Systems, pp. 59–74. Springer (2018)