Christine Jakobs

Optimizing the Automotive Security Development Process
in Early Process Design Phases

Christine Jakobs

Optimizing the Automotive Security Development Process in Early Process Design Phases

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz
2023

Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über `https://www.dnb.de` abrufbar.

Die vorliegende Arbeit wurde von der Fakultät für Informatik der Technischen Universität Chemnitz als Dissertation zur Erlangung des akademischen Grades Dr.-Ing. genehmigt.

Gutachter:
Prof. Dr. Matthias Werner
Prof. Dr. Peter Tröger
Dr. Karsten Schmidt
Tag der Einreichung: 05.12.2022
Tag der Verteidigung: 20.06.2023

Universitätsverlag Chemnitz
`https://www.tu-chemnitz.de/ub/univerlag`

Print on Demand
Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-96100-190-3
DOI: 10.51382/978-3-96100-190-3
URN: urn:nbn:de:bsz:ch1-qucosa2-861067

Titelgrafik: Christine Jakobs

To all the people who supported me in finding my own way.
I would not be who I am without you.
Thank you for your trust and support.

Acknowledgement

Nach Fertigstellung des Exposés meinte mein Doktorvater zu mir: “Das hättest du, als du das erste Mal bei mir im Büro gestanden hast, auch nicht gedacht”. Damit lag er richtig, denn ohne die immer währende Förderung und Ermutigung wäre ich wahrscheinlich nie auf die Idee gekommen, den Traum der Promotion wirklich in Angriff zu nehmen. Daher gilt meine größte Anerkennung Prof. Matthias Werner.
Das Forschungsthema selbst verdanke ich meinem ehemaligen Kollegen, Prof. Dr. Peter Tröger. Mein Dank gilt Peter für seine Geduld, sein Einfühlungsvermögen und vor allem seinem Durchhaltevermögen bei unseren teils nächtlichen Forschungsdiskussionen, bereits während meines Masterstudiums. Ohne Peter wäre ich wohl vor einigen Hürden der Dissertation zurückgeschreckt. Danke, dass ich an deinen Erfahrungen teilhaben durfte und du mir gezeigt hast, wie faszinierend die Welt der Verlässlichkeit sein kann.
Sowohl Peter als auch Matthias haben meine wilden Ideen immer getragen und Begeisterung für jeden noch so verrückten Einfall gezeigt. Meinen Dank für eure unermüdliche Geduld meine Gedanken in die richtige Richtung zu lenken.
Nach der Meinung von Jenni(fer) Hofmann können wir Informatiker alle keine Kommasetzung und ohne Kaffee läuft bei uns gar nichts. So bekam ich sogar Nachts um 4 Uhr am Flughafen auf dem Weg nach Hawaii plötzlich Kaffee in die Hand gedrückt. Danke dafür, dass ich die Rechnung für die fehlenden Kommas nie bezahlen muss.
Zahlreiche Studenten und Kollegen haben durch ihre Mitarbeit bei Veröffentlichungen, das Korrekturlesen, ihre studentischen Arbeiten oder einfach nur dadurch, dass sie meine Kollegen waren dazu beigetragen, dass diese Arbeit nun endlich gedruckt wird. Mein Dank gilt Jafar Akhundov, Stefan Naumann, Laura Morgenstern, Julia Scharsich, Billy Naumann, Martin Richter, Theresa Werner, Jonas Henschel und allen weiteren Seminar-, Bachelor- und Masterstudenten, deren Arbeiten ich betreut habe.
Last but not least wäre die Arbeit rein theoretischer Natur und ohne Bezug zu einem realen System, wenn meine Kollegen bei der AUDI AG nicht wären. Meine Anerkennung an Dr. Karsten Schmidt. Seine offene Art hat mir bereits in früheren Projekten zahlreiche offene Probleme aufgezeigt, von denen ich nur noch eines auswählen musste. Seine Unterstützung bei der Realisierung des Doktorandenprojektes hat die Arbeit immer wieder geformt. Stefan Bauch, Jörg Herz und die Kollegen der I/EG-131 und I/EG-132 haben mir unermüdlich die Möglichkeit gegeben, am “lebenden Objekt” zu erfahren, dass Security so viel mehr als Kryptographie ist.
Nicht zuletzt kann so eine Arbeit niemals ohne die Unterstützung des privaten Umfelds entstehen. Vielen Dank an meine Mutter und an den besten Partner, den man sich im Leben wünschen kann. Ich will nicht wissen wie sehr er unter der Schlusszeit der Diss gelitten hat. Danke für deine Unterstützung und nein, du musst dieses Buch nicht lesen :).

Abstract

Security is a relatively new topic in the automotive industry. In the former days, the only security defense methods were the engine immobilizer and the anti-theft alarm system. The rising connection of vehicles to external networks made it necessary to extend the security effort by introducing security development processes. These processes include, among others, risk analysis and treatment steps. In parallel, the development of ISO/SAE 21434 and UNECE No. R155 started. The long development cycles in the automotive industry made it necessary to align the development processes' early designs with the standards' draft releases.
This work aims to design a new consistent, complete and efficient security development process, aligned with the normative references. The resulting development process design aligns with the overall development methodology of the underlying, evaluated development process. Use cases serve as a basis for evaluating improvements and the method designs. This work concentrates on the left leg of the V-Model. Nevertheless, future work targets extensions for a holistic development approach for safety and security.

Keywords: Automotive Systems, Dependability, Security, Development Process

Contents

List of Figures

List of Tables

Acronyms

ADAS	Advanced Driver Assistance Systems
AIS	Abbreviated Injury Scale
ARIS	Architecture of Integrated Information Systems
ASIL	Automotive Safety Integrity Level
Automotive SPICE	Automotive Software Process Improvement and Capability Determination
AUTOSAR	AUTomotive Open System ARchitecture
CAN	Controller Area Network
COTS	commercial off-the-shelf
CVSS	Common Vulnerability Scoring System
DES	Data Encryption Standard
DFD	Data Flow Diagrams
E/E	electrical and electronic
ECU	electronic control unit
ERP	enterprise resource planning
EU	European Union
EVITA	E-safety vehicle intrusion protected applications
FDS	Flashdatensicherheit (eng.: Flashware Protection)
FMEA	failure mode and effects analysis
HEAVENS	HEAling Vulnerabilities to ENhance Software Security and Safety
HSM	Hardware Security Module
IDS	intrusion detection system
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LAN	Local Area Network
LIN	Local Interconnect Network
MBSE	model-based systems engineering
MISRA	Motor Industry Software Reliability Association
MoRA	Modular Risk Assessment
MQTT	Message Queuing Telemetry Transport

NDA	non-disclosure agreement
OEM	original equipment manufacturer
OS	operating system
PASTA	Process for Attack Simulation and Threat Analysis
RAP	Required Attack Potential
RNG	random number generator
RTE	AUTOSAR Run-Time Environment
RU	Road User
SAE	Society of Automotive Engineers
SecOC	secure onboard communication
SOA	service-oriented architecture
SOME/IP	Scalable Service-Oriented Middleware over IP
SOTIF	Safety of the Intended Functionality
SPOF	single point of failure
SRA	security risk assessment
SRE	security relevance evaluation
SSL	Secure Sockets Layer
TEE	trusted execution environment
TLS	Transport Layer Security
TVRA	Threat, Vulnerability and Risk Assessment
UML	Unified Modeling Language
UNECE	United Nations Economic Commission for Europe
ViWi	Volkswagen Infotainment Web Interface
VKMS	vehicle key management system
VLAN	virtual local area network
WLAN	wireless local area network

Part I.

Foundation

The process of planning is very valuable, for forcing you to think hard about what you are doing, but the actual plan that results from it is probably useless.

Marc Andreessen

1 Introduction

Few quotes are going better with this dissertation than the one of Marc Andreessen. At the beginning of this dissertation project, the aim was a holistic process for architecture verification regarding the meta-functional aspects of safety and security. The project's idea was to incorporate both aspects into the development process in a unified way.

The necessity for such a new view on the development arises since modern automotive systems have an increasing amount of driving assistance features [79, 48] constantly moving towards automated driving. Statistics show that tens of millions of connected vehicles exist [27], something necessary for advanced driving assistance features. This shift leads to more complex functionality, primarily implemented in software [42]. The results are manifold. More software functionalities lead to a rising demand for hardware resources which is critical due to the limited space in the vehicle. High integration of software onto commercial off-the-shelf (COTS) hardware is a reasonable possibility to overcome this issue and reduce costs for specialized hardware [22]. The distributed development process with numerous suppliers demands rigorous approaches for coordination and consultation between all parties concerned [79, p. 5].

The described trends imply a shift from the closed world assumption of former automotive systems to open systems [48]. Since automotive systems are so-called risk-prone systems regarding safety and security, this shift raises the demand for quality management [79, p. 1]. Safety and security are traditionally disciplines that interrelate due to their dual view of the system. Safety prevents harm to the user and the environment, while security prevents harm to the system arising from the user and the environment. The necessary cohabitation of both so-called meta-functional aspects makes it necessary to rethink the development process designs: Otherwise,

it is impossible to identify interrelations as early as possible, raising the costs for design changes. Especially the transition from the design and implementation phase to validation and verification is essential in this case. It needs wholly defined and well-implemented development process steps.
While safety is a known field in the automotive industry, security is newly emerging. The introduction of security started with anti-theft alarm systems, electronic immobilizers, Remote Key-less Entry, and Key-less Go [104]. The emphasis was on theft protection and tuning prevention rather than analyzing system weaknesses. Nevertheless, the new trends make it necessary to see security development as a continuing process introduced early and holistically in the development process [33, 25]. This so-called system-level security or security-by-design targets the system's design to be robust to currently known security threats and tolerant in case of malicious attacks [87]. Also, the design must handle currently unknown threats as far as possible. Nowadays, on the ranking of security attacks, the automotive industry is in the third rank, which leads to high costs for handling attacks [27]. This trend is possible through the extended attack surface of vehicles enabled by the vehicle's connectivity [48].
The focus and release time of the normative references emphasize the necessity for a shift in security development. SAE J3061 *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems* [98] was the first published reference addressing security issues on the level of unauthorized access to data. The guidebook provided first recommendations for the cybersecurity process, too. Type approval in the automotive domain now relies on accomplishing the process standards ISO/SAE 21434 [54], and UNECE No. R155 [85]. The release of those standards was just recently. However, the long development times in the automotive domain made it necessary to adopt the outlined security development process from the normative references since its draft release.
Those standards demand a security development process that identifies unforeseen consequences and treat those risks [38, p. 4]. Thereby the security risk analysis tells a story of five questions:

- What can go wrong? - These are the damage scenarios endangering the system's correctness.
- How bad is this? - The impact of those damage scenarios.
- How can that happen? - The threat scenarios and attack paths an attacker might use to accomplish the damage scenarios.
- Who can do that? - The type of attacker which can successfully launch the attacks.
- How probable is this? - The feasibility of the attack.

The subsequent risk treatment step aims to consolidate the story. Thereby it weights the different elements and assigns defense methods to the system to reduce the risks arising from the impact and feasibility. Therefore, this step defines the implementation requirements regarding security.

Implementing such analysis and treatment steps demands a new development process. The core, but not necessarily the start of each development process design, is its definition, the methods, and tools needed to accomplish the process steps [66, p. 1]. Those methods help to achieve the necessary activities systematically [8]. Initial process descriptions typically arise from former development cycles and are subject to continuous evaluation and optimization to enhance efficiency and acceptance.
In the case of automotive security, there is no suitable tradition of modeling and analysis, like in safety. Initially, processes with this background are neither complete nor well-implemented and have high improvement potential. In this case, the processes were subject to constant change to align with the draft releases of the normative references.
Due to these issues, it got apparent that the original project idea is impossible with a proper relation to a real-world development scenario. Therefore, the project evermore moved to the left leg of the V-Model. It ended with the question of a complete, consistent, and efficient security development process design. The scope of the process design is the vehicle itself. Therefore, the vehicle's environment, backend servers, and production facilities are out of scope. The result builds the basis for accomplishing the original idea in future work.

1.1. Research Questions

This work targets the following research questions in order to design a complete, consistent and efficient security development process:

RQ 1: What is a suitable terminology for the security development process? Security is a traditional topic in information technology but new to the automotive domain. Literature from research and industry shows that there is no standard terminology. Therefore, it is necessary to define a distinct terminology to rate existing work and have a common understanding between the involved parties. The approach transfers the well-defined and commonly accepted *Laprie model* to the automotive security area.

RQ 2: What are the demands of the normative references? Automotive development demands alignment with normative references. Therefore, the starting point for defining or evaluating a development process is the analysis of those demands.

RQ 3: What are the necessary analysis and treatment steps? The demands of the normative references provide the abstract development process structure and process descriptions. Those enable the evaluation of possible implementations. In the case of already existing process steps, those need an evaluation regarding completeness. Missing process steps in the existing process make an extension necessary.

RQ 4: What are the possibilities to raise efficiency in the security development process? The complete process description allows for an evaluation of efficiency. This evaluation demands distinct properties and reveals mechanisms, e.g., to reduce repetition and raise automation.

RQ 5: Where is tracing necessary? How can tracing be introduced? A distributed development environment is subject to frequent adjustments of requirements. The certification demands of the normative references make it necessary for the security development process to support the tracing of decisions. Otherwise, it is impossible to reproduce process steps and work products.

1.2. Outline and Impact

The work begins with the background and the methodology (Part I). This part clarifies the development methodology of the evaluated security development process and the methodology used to achieve the contributions.
The next part about meta-functional aspects (Part II) aims to incorporate readers from the safety and security domain. It transfers the dependability tree of Laprie onto the security domain. It defines a taxonomy for a common understanding of the underlying security terminology in the automotive domain and the remaining work.
Part III uses the results from Part II for the security development process design. Thereby, this part uses the existing security development process structure with a state of September 2021 and includes later process changes as far as possible. The main contributions are a new design of the security relevance evaluation [60], improvements to the functional security risk analysis, a proposal for the former missing step of the system level risk analysis, and the risk treatment [59].
Use cases in Part IV provide examples for the security development steps of Part III and evaluate the achievements. The last part (Part V) discusses the results, concludes the work, and presents how the original project idea relies on the achievements of this work in the future work chapter. This part incorporates the contributions presented in former work [58, 57, 61].

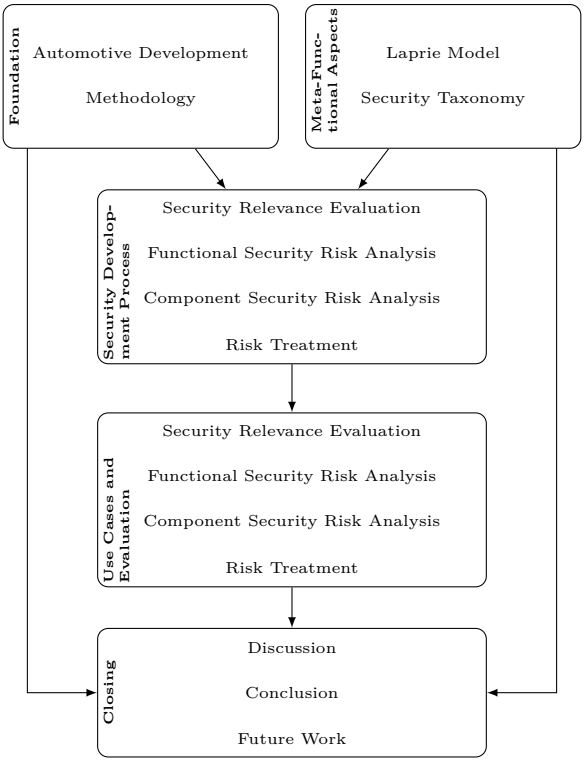

Figure 1.1.: Structure of this work.

What I wanted to do was build an automobile.
Carroll Shelby

2 Automotive Development

A development process for meta-functional aspects needs to align with the overall development methodology and follow a process model. Resulting from the normative references, the automotive development process commonly follows the V-Model (see Section 2.1). One constraint for this work is function-oriented development (see Section 2.2), embedded in a systems engineering approach (see Section 2.3). The commonalities of security add important points to the development process (see Section 2.4) to fulfill the demands of the normative references (see Section 2.5).

2.1. Development Models

Development models help structure the development steps by defining the beginning, end, and activities themselves [113, p. 32]. They follow an underlying development philosophy: sequential or incremental methods.

Sequential methods are those where the other system parts development happens in a series of steps following a downward fashion from initial requirements to the design, and the final product [100], [113, p. 33]. Such methods are suitable for large, distributed development teams since they force the developers to stick to the provided framework [113, p. 33]. They are also suitable for hardware development since they restrict iterations and thereby late changes of the design [100].
The advantages of sequential models are predictability, stability, repeatability, and high assurance [113, p. 33].

Incremental methods are typically also iterative by nature. They built upon the philosophy to gradually improve the system until the design incorporates all demanded capabilities [113, p. 36], [100].
This approach is especially suitable for projects with many development steps and concurrent development cycles. The latter is since the iterative nature allows the incorporation of lessons learned from other development strands meanwhile the different development phases [100].

Combined approaches are reasonable, depending on the development process. The automotive industry tends to use the V-Model as a sequential development philosophy candidate and combine it with the iterative nature of the spiral model [95, p. 12].
The normative references of the security engineering process rely on the V-Model, but in reality, the development process is not entirely sequential. Several prototypes and iterations accompany the vehicle development, which the spiral model covers.
Therefore, the remaining section covers the V-Model but also shortly introduces the idea of the spiral model. Interested readers may refer to the work of [121] for an extensive overview of process models.

2.1.1. Spiral Model

The spiral model is an iterative model of the design process [121] relying on a bi-directional development process. Top-down, the process derives requirements. Based on evaluating prototypes, necessary changes adjust the requirements bottom-up. Figure 2.1 illustrates the spiral model with one middle layer. In reality, the spiral has several layers with a new prototype each.
During the development cycle of the spiral model the development phases overlap [77, pp. 70 sq.], [37, p. 73]. The specifications validation (Definition 25) and verification (Definition 26) is part of the prototyping process. By that, the specifications refine along with the prototype while proceeding along the spiral to the outside [37, p. 73], [121].
The spiral model allows for high flexibility and user involvement in the development process. The iterative manner allows reacting very fast to changes in the requirements. The prototypes support this since they make stakeholder feedback easier.
On the other hand, the spiral model is only suitable for large-scale projects, which allow for constructing prototypes. The flexibility of the spiral model needs stringent management of the process with appropriate risk assessment, modeling, and analysis tools [77, pp. 71 sq.]. Otherwise, the project might end unfinished due to constant changes and new requirements.

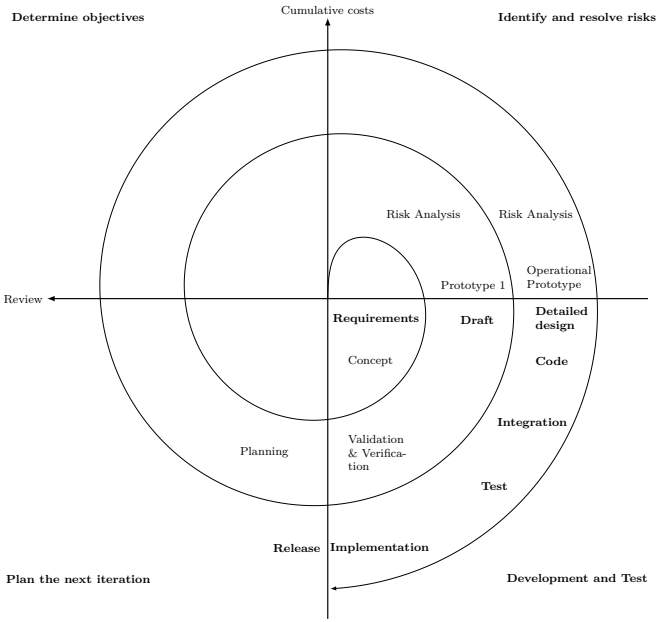

Figure 2.1.: Spiral Model, according to [37, p. 74].

2.1.2. V-Model

The V-Model[1] or Vee-Model is an extension of the sequential waterfall model [77, pp. 72 sq.]. The characteristic shape (see Figure 2.2) results from splitting the development process into a design and an integration phase [77, pp. 72 sq.], [99, 101], [120, p. 7], [37, pp. 58 sq.] accompanied by verification and validation steps on each level [77, pp. 72 sq.].
The system design evolves downwards on the left-hand side, level by level [116], starting with the product requirement analysis and different architectural views (e.g., logical, technical, functional). The different development strands for the concrete implementation are at the end of the left leg. This part depicts the decomposition of the overall system into the subsystems for implementing requirements and showing feasibility [37, pp. 58 sq.]. In automotive systems, this would be the vehicle subsystems with its different parts of mechatronic, electrical and electronic (E/E), hardware and software development [37, pp. 58 sq.], [99]. The elements of the lowest level are subject to production or harvesting. Those are not further decomposed [116].
Parallel to the left leg of the V-Model, the integration phases compose the system accompanied by validation, and verification steps [99]. The right side ends with the overall system, validating that the initial requirements hold [77, pp. 72 sq.].
In conclusion, the V-Model levels illustrate the system development levels. Composing the left and right-hand provides a particular system view on each level [116]. From left to right (horizontally), the time and maturity of the product evolve [113, pp. 34 sq.]. Vertically, the levels of the system evolve. From the system context and requirements analysis over logical and physical levels to the system parts in the bottom layer [116].
The advantages of the V-Model are the easy addressable design verification against the implementation [77, pp. 72 sq.]. Iterations in the development process are possible on each level, on the left and right leg of the V [116]. This maintains high flexibility. Each level builds a consistent baseline, enabling early verification and concurrent engineering planning. The transfer of requirements downward the V and their validation maintains bidirectional communication between the incorporated parties [116].
In contrast to those advantages, the model does not address supporting processes, and therefore the process organization [77, pp. 72 sq.]. To overcome this issue, [101] formulates several views on the V-Model. Readers interested in those extensions may refer to [101] as a starting point for further information about V-Model views.

2.2. Function Oriented Development

Automotive development projects tend to have many different functionalities and variants of them. Those variants may be because of country dependencies or variations in the function range.

[1] The reader may not mix the V-Model with the German V-Modell [36] which is a development standard.

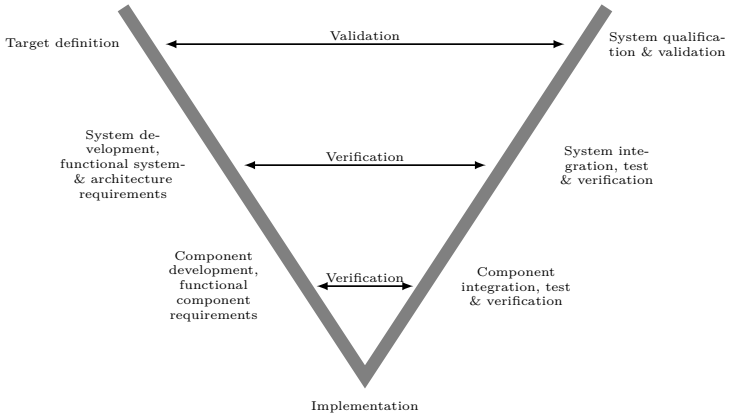

Figure 2.2.: V-Model, according to [37, p. 59].

Not every function variant is part of the resulting automotive architecture. Nevertheless, the development process must incorporate them all until the start of serial development. At this point, functions or function variants not part of this vehicle project drop out of the development process.
Function-oriented development targets this issue by incorporating functions independent of their definitive treatment. This development methodology is part of process organization and architecture generation. The idea is to decompose the system into functionalities and define the system architecture through functions, and sub-functions [113, pp. 190 sqq.] leaving the details of how to encompass them aside. By that, it is possible to develop the functions independent of the final architecture. This function-oriented view also reveals interdependencies, and action chains between the functions, supporting the decision processes on the different development process layers [114, 16].
Important in this approach is the independence of a concrete deployment. The functions' interdependencies and demands do not relate to the electronic control units (ECUs). The hardware is part of the solution space in the systems architecture. This independence allows the usage of the degrees of freedom in the deployment. Those are, among others, costs and weight [16].

2.3. Systems Engineering

Automotive development consists of hardware and software development. Several disciplines are essential in hardware development, e.g., E/E and mechanics. The functions are mostly implemented in software. Meta-functional aspects introduce additional dependencies into the development process and the resulting system.

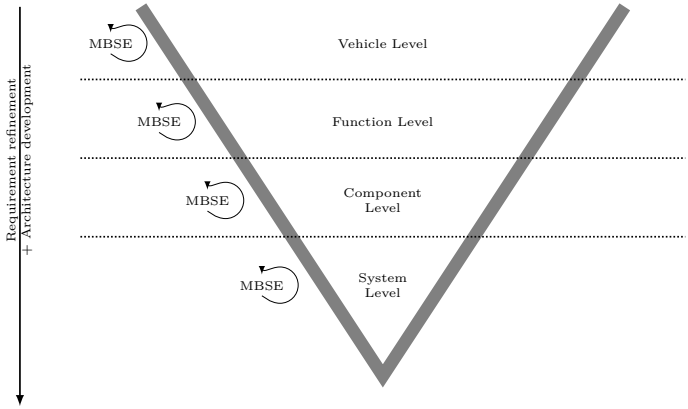

Figure 2.3.: Systems engineering concepts with the V-Model structure. The dotted line depicts the baseline of the different layers. Each layer uses model-based systems engineering (MBSE) for deriving the requirements.

Systems engineering is an interdisciplinary field that is especially suitable for such systems. Besides the inclusion of meta-functional aspects, it provides mechanisms for interdependencies between different engineering disciplines and to support distributed development environments [37, p. IX]. Thereby, systems engineering is no plan or method, but only a layout for system thinking and problem-solving [37, p. 3], [113, p. 11].
The idea behind systems engineering is to derive the system requirements top-down from the product properties in the stakeholder view. Those requirements refine and decompose step-wise until the subsystem level [77, p. 10]. The single subsystems integrate into functional clusters. The system structure depicts the project organization, the responsibilities, and the development team structure.
Combined with function-oriented development, systems engineering provides the layering and dependency relations between the overall system requirements and the development strands, ultimately supporting the definition of the system architecture in which the functions integrate (see Figure 2.3). The stakeholder and system requirements are the primary concern of systems engineering. Each level of the systems engineering top-down decomposition refines the requirements allocated to the functions (function-oriented development). The V-Model provides the structure in which the systems engineering approach operates. Therefore, it represents the layers from the vehicle view, function view, and component view in combination with the notion of baselines and verification/validation of the requirements and the system.

2.4. System Security Engineering

Systems engineering incorporates risk management in the system architecture. This management divides into risk identification of the product development risks and the product itself, as well as analysis and treatment of those risks [67, p. 184].
Therefore, systems engineering directly supports the management of meta-functional aspects like safety and security and allows suitable development processes for those aspects through viewpoints. The security viewpoint incorporates requirements related to the security-development which is the fundamental principle of security engineering [39, p. 26], [113, p. 234]. Therefore, systems engineering includes support for security-by-design. The reliance on modeling and analysis to build the systems architecture allows model-based analysis of the system related to security. This basis is a systematic, distinct and traceable way to manage security risks [39, pp. 16, 20].
The layout for the security engineering process used for this thesis starts at the top level of system engineering. In the beginning, the security concept is subject to updating. This update introduces new technologies and strategies into the security development process [113, pp. 234 sq.]. Conducting the risk assessment already in the requirement engineering phase introduces security requirements as early as possible [10]. This approach allows tailoring subsequent process steps related to the necessary effort [30]. The security assessment steps refine the security requirements regarding the system functionalities [38, p. 3] and validate/verify them [113, pp. 235 sq.]. Security engineering observes that the product realizes the introduced requirements during production. Throughout the complete life-cycle of the car, security engineering maintains the secure state of the car through updates and other support processes [113, pp. 234 sq.].

2.5. Normative References

Automotive security is, like most engineering disciplines, subject to the demands of standards, and state-of-the-art practice recommendations [113, p. 166].
At first, the Society of Automotive Engineers (SAE) published the SAE J3061, the *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems* [98]. This guide provides information about recommended practices on automotive security. Thereby, SAE J3061 addresses security issues on the level of unauthorized access to data and gives recommendations for the Cybersecurity process.
In 2016 International Organization for Standardization (ISO) and SAE started the development of the standard *Road vehicles – Cybersecurity engineering* and released it in 2021. The security standard for automotive systems relies on the recommendations of SAE J3061 and extends them to the CIA triad (Confidentiality, Integrity, and Availability) [54].

ISO/SAE 21434 serves as a guideline to implement UNECE No. R155, the *Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system* [85]. The first release of the UNECE No. R155 was in 2020 and became effective in 2021. UNECE No. R155 is binding for all automotive vehicle types beginning with July 2022 and from July 2024 on for all new vehicles [28].

2.5.1. ISO/SAE 21434

The ISO/SAE 21434 standard defines the process steps in automotive security development. Thereby the ISO/SAE 21434s main part for the product development aligns to the V-Model and outlines the development process steps, along with possible methods. Also, it provides information about approaches usable throughout the vehicle lifecycle [27]. Nevertheless, it does not provide details about efficiency [27]. This issue is up to the realization of the development process in the company. ISO/SAE 21434 defines the process steps typically as a triple: Input, requirements, and work products. While input describes the results of preceding process steps, the work products section defines the results of this process step. The requirements section defines this process's goal and possible methods.

2.5.2. UNECE No. R155

UNECE No. R155 describes the necessary parts of the security management systems to acquire the certification according to the standard. This certification is subject to revalidation every three years [85] and is required for type approval in the European Union (EU) and other countries.
Therefore, the UNECE No. R155 describes which measures the company need to take. In conclusion, it defines that the company needs to have a "systematic risk-based approach defining organizational processes, responsibilities, and governance to treat risk associated with cyber threats to vehicles and protect them from cyberattacks" [85].

2.5.3. Other Normative References

ISO/IEC 27001 covers information security systems [56] rather than automotive security. There are fundamental differences between both. As described by [102], automotive systems, in contrast to information systems, cannot be isolated from their environment. However, the requirements of ISO/IEC 27001 are not irrelevant in automotive development since it targets the backend-server and the information systems of the original equipment manufacturer (OEM).

UNECE No. R156, *Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system* [111] targets security related to the integrity of the update. Although this is relevant for type approval of automotive systems, it is not relevant for this work. A suitable security process for the related information systems and the automotive system covers this issue without further adjustments.

We can't solve problems by using the same kind of thinking we used when we created them.

Albert Einstein

3 Methodology

This thesis aims to design a security development process using the evaluation of an existing one and the normative references. The following chapter describes the methodology used to accomplish this task, whereby a methodology is seen as a set of "processes, methods, and tools used to support a specific discipline" [113, p. 190].

3.1. Representation in the Thesis

Since automotive security is a relatively new field, no common terminology exists. The first step in this thesis is to determine a security taxonomy to overcome this issue and provide a distinct basis. This taxonomy transfers the existing dependability tree of the *Laprie model* onto security. The result is the foundation for the subsequent development process design.
The following process description targets each process step one at a time. Thereby, it uses a structure of five steps: The normative references for the process step, the current implementation, and related work give rise to issues and improvement potentials. Those improvement potentials and issues are subject to the subsequent solution approach, which suitable use cases accompany.

Therefore, each process step is part of two chapters with the following structure:

- Process Step (Part IV)
  - Normative references
  - Current process implementation
  - Related work (if existing)
  - Problem description and solution approach
  - Contributions
- Use Cases (Part V)
  - Application and results for the process step
  - Evaluation by reference to evaluation criteria and requirements

The structure illustrates the methodology of the process design. The aim is to evaluate the demands of the normative references and general process requirements. Those requirements depict the responsibilities, approach, methods, and tools. The evaluation aims to systematically identify problems with the current process implementation, and possible solutions to overcome these issues [108]. The contributions part of each process chapter aims to implement the solutions, followed by a use case including a nominal-actual comparison in the evaluation.

3.2. Information Sources

The evaluation of the state-of-the-art involves several sources. From the normative reference side, ISO/SAE 21434 [54] and UNECE No. R155 [85, 86] are the main sources. Those describe the demanded security development process steps as well as possible methods. Therefore, those sources are the baseline to which the development process needs to align.

An Architecture of Integrated Information Systems (ARIS) model illustrates the current development process implementation in the form of a business process model. The swimlane diagram shows the different process steps as well as preparational steps. Also, it incorporates responsibilities between the different security development roles. This model provides the development process structure, which must align with the development process steps of the normative references.

The definition of the development steps is in an internal wiki. Due to the company's global availability of the wiki, the responsible persons in the development process and all interested employees can engage with the topic. This wiki documents the process steps' activities and the tools and methods used.

Another source for information about the current process implementation is the development schedule. It gives rise to the points in time where the process steps occur, which allows identifying if may need to provide the notation of uncertainties. For the development process tracking, the security department uses the process and issue tracking software Jira[1]. With that tool, the development team can manage the security development process in a scrum-like fashion. By that, the process keeps basic traceability through the development process of the functionalities and over the complete progress.

[1] https://www.atlassian.com/software/jira

In order to prevent unaware changes on the work products, the security department uses a document management system based on Documentum[2]. One necessary key feature of this software is its ability to work with confidential information.
The information in the process tracking software and the document management system gives rise to the current implementation of the security development process. By that, it is possible to evaluate current templates and process results. The criteria are the alignment to the normative references, the time effort, and the support of model-based development and security-by-design.
By evaluating the mentioned sources, issues regarding the defined criteria are identifiable. Those provide the improvement potential. Suggested improvements do, if possible, not introduce new methods or tools but align and streamline the existing methods and tools. Therefore, the methodologies solution part describes the adjustments to the methods and tools and possible extensions.
The use case part serves two purposes. On the one hand, it shows how the methods and tools in the improved version apply in practice. On the other hand, the use case gives rise to accomplishing the improvement goals. Since the information used throughout the work is subject to non-disclosure agreement (NDA), the use case section sometimes lacks details. It is necessary to use approximated numbers and reduce the content of the use case input data and work products for this work.

[2] https://www.opentext.de/produkte-und-loesungen/produkte/enterprise-content-management/documentum-platform/documentum-d2

Part II.

Meta-Functional Aspects

The literature commonly uses the term *non-functional properties* which indicates that these are properties of a not functioning (broken) system. Therefore, this thesis decided to use another term to emphasize that these properties do not directly relate to the system's functionality: *meta-functional aspects*. Those aspects depict certain viewpoints to the system [37, p. 9] related to properties besides the specified functionality. For example, timing as a system aspect narrows the usefulness or correctness of the function regarding the viewpoint of timing requirements, not the functionality itself.

Safe is a relative construct. Safe has inconsistent meanings.

Star Trek Discovery, Season 2, Episode 8

4 Dependability as an Umbrella-Term

Dependability describes the extent to which the user can trust that a system's behavior follows its specification. The dependability attributes describe certain types of trust in the system behavior. Impairments endanger these system properties, while dependability means help to prevent these impairments (see Figure 4.1).
Research and industry discussed the terms related to the concept of dependability to an enormous extent with varying definitions. In order to prevent misunderstandings in the remaining work, this chapter discusses the dependability impairments, attributes, and means regarding their definitions according to the *Laprie model*, the applicable standard in the automotive industry (ISO 26262:2018 [53]) and related literature as a reference point. Thereby, the security-related terms are left out. They are the focus of the following Chapter 5, the security taxonomy.

4.1. Related Work

The research community commonly follows the approach by Laprie/Avižienis et al. [12, 15, 14, 13, 70, 71, 72, 69] [1] who group the different system attributes regarding the quality of service into dependability attributes and according to impairments and means (see Figure 4.1). Laprie et al. provide a distinct understanding between the general meaning and the quantification of the concepts of dependability as well as emphasize the commonality of the different perceptions of the attributes, which aim to give trust in the examined systems behavior [70]. The reliance on the system behavior includes both the correctness and the continuity of delivery.

[97] calls the *Laprie model* the dependability approach, which targets ultra-reliable and fault-tolerant systems. While the typical approach includes safety into dependability, [97] differentiates the dependability approach and the safety approach. The dependability approach "maximizes the extent to which the system works well, while the safety engineering tries to minimize the extent to which it can fail badly" [97]. In his paper, he categorizes both approaches according to their use case. While the dependability approach is usable in systems "where there is no safe alternative to normal service", e.g., aircraft systems, the latter approach is used in systems like military use cases "where there are specific undesired events" [97].

On the one hand, this categorization is interesting since it reflects different priorities in the industry. On the other hand, the differentiation between safety and dependability is not distinct. For example, the automotive industry emphasizes safety and leaves the other attributes aside. Nevertheless, they cover reliability and availability indirectly through the analysis in the development process. It is impossible to follow only one system attribute and leave the other entirely aside. The question is rather what the priority of the analysis results is. This interrelation also shows an evaluation of [97] since the presentation of the safety approach is mainly industry-related. In contrast, the dependability approach covers research and practice following the *Laprie model*.

[65] attaches to the definitions of Laprie and his system model and interprets the dependability concept in a behavioral context. The work introduces the concept of trustability to cover the behavioral attributes of dependable systems.

Different standards arose from the industry's perspective, applicable to different industry areas. They use their vocabulary, which differs between themselves and the research literature. This controversy leads to a diverse mindset about the different dependability terms, definitions, and interrelationships. When comparing literature out of industry standards and research like done by [109, 110] one can see that there are different goals the system design has to emphasize and the names for the different dependability attributes and impairments vary. However, at its core, the goal is the same. The user should be able to trust the system because it functions correctly in the behalves of its specification.

[1] Although the authors of the papers to the *Laprie model* changed and mentioned Avižienis more often as first author, the model itself is commonly called the Laprie model. Therefore, the author decided only to use the name Laprie in the text.

In the automotive industry, the standard ISO 26262:2018 ("Road vehicles – Functional safety") is the derived version of the International Electrotechnical Commission (IEC) 61508 series ("Functional safety of electrical/electronic/programmable electronic safety-related-systems"), the originating functional safety norm series.
Like in the research literature, the standards are selective in the defined terms. Although safety analysis typically uses results from reliability analysis, ISO 26262:2018 [53] does not define the term reliability.
ISO 21448:2022 [52] targets unexpected safety impairments, or the Safety of the Intended Functionality (SOTIF). Those are performance impairments, wrong situation assessments, and missing robustness against environmental influences. The target functions are emergency intervention systems and Advanced Driver Assistance Systems (ADAS) systems on levels 1 and 2.

4.2. Dependability Impairments

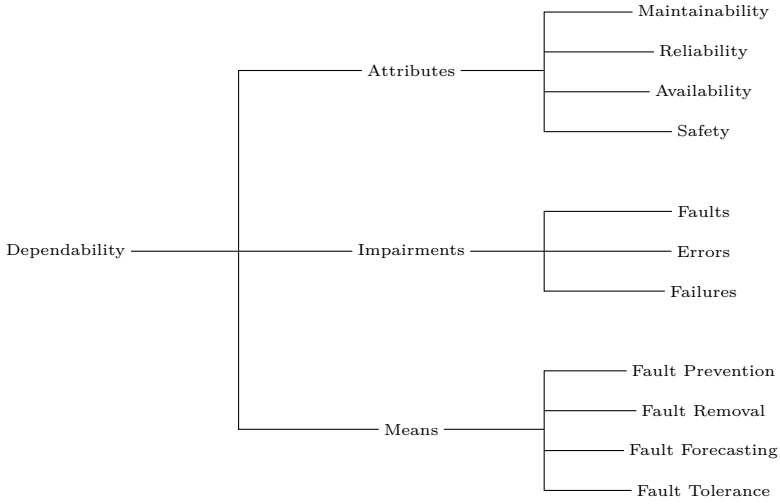

Figure 4.1.: Laprie Model, according to [13] with attributes reduced by the security objectives confidentiality and integrity.

Following the *Laprie model* (Figure 4.1), there are three categories to describe dependability impairments: Faults, errors, and failures. These relate to the underlying quality of the systems delivered service. It is the correct service if the system implements the system specification [13]. If this service deviates, the system transits from correct (regular) to incorrect behavior (error) and may ultimately lead to a system failure.

4.2.1. Faults

Definition 1 (Fault). *A fault is an unintended system condition that may result in an error. Internal faults reside in the system, while external faults originate from the system's environment.*

Faults are the actual or hypothesized causes of consecutive errors [13, 45] [66, p. 17]. They are typically seen as imperfections in the system [117] or not intended conditions which may result in a system failure [53, Part I]. These definitions state that faults are the beginning of the dependability chain leading ultimately to system failure.
Laprie et al. state that every fault is a design fault in the underlying system specification [69]. From a pragmatic point of view, this is not beneficial in modeling and analysis. Therefore, there is a distinction between *internal faults* and *external faults*. An internal fault is an event that leads to the system being still operational, and the behavior seems to be correct. Nevertheless, the system has an underlying defect in its design. Internal faults reside in the system and are activatable by internal control flows. External faults are environmental conditions introducing an error cause through interference or interaction [65, 13]. The distinction between internal and external faults helps in system design. Internal faults are those in the sphere of influence of the system designer and maintainer. External faults are coverable in the system design and analysis but are out of reach during the system's run-time. This differentiation covers also the dependability means (see Section 4.4).
In literature, error or defect is sometimes used instead of fault. [109, p. 85] provides a good overview of the deviations from Lapries definitions.

4.2.2. Errors

Definition 2 (Error). *The internal system state deviates from the correct state.*

In case of an activated fault the system state deviates from the correct state [53, Part I], [13, 45], "from accuracy or correctness" [117]. This is an atomic transition into the error state [65, 117, 13]. The error state itself is an internal system state which can be triggered through internal or external faults [65]. Errors are, therefore, the next stage in the dependability chain. The system's internal state deviates, but the deviation is not recognizable by the user.
An error can possibly be *detected* which is typically made recognizable through a log or error message [66, p. 18]. In this case, there is the option to *handle* the error in a way that leaves the system operational [109, p. 97]. As long as the error is not detected, it is called a latent error [13]. Those definitions describe the different states an error might have. Whether the system detects and handles an error depends on the system implementation. The dependability means cover that, e.g., by fault-tolerance approaches (see Section 4.4).

4.2.3. Failures

Definition 3 (Failure). *A failure transition leads to the system service deviating from the intended one.*

Not detected errors might lead to the event of the system behavior deviating in a way in which the wrong system state gets externally recognized by the environment (e.g., the user) [65, 13, 45] [53, Part I]. This transition transforms the system state from operational into failed according to the system specification [65, 13]. Thereby, failures are the last stage in the dependability chain. The system behavior deviates from the intended one and violates the dependability attributes.
Depending on the severity of the failure, the primary differentiation is benign/minor failure and malign/catastrophic failure. The former denotes failures "where the harmful consequences are of similar cost as the benefit provided by correct service delivery" [13]. The latter are those, "where the cost of harmful consequences is orders of magnitude, or even incommensurately, higher than the benefit provided by correct service delivery" [13]. This differentiation depends on how much a system must fulfill the dependability attributes. For example, safety-critical systems aim to prevent malign failures in any case.

4.2.4. Error Propagation

The *Laprie model* calls the correlation between the dependability impairments in different system parts error propagation. [109, pp. 86 sqq.] shows that this idea is usable for the propagation between system layers, e.g., hardware, operating system, middle-ware, and application. These layer abstractions are the system and execution layers (see Figure 4.2).
An internal fault gets activated during system execution. When the internal system state deviates through external faults or control flows triggering internal faults, the system goes into the error state. This error is the cause of a failure in the system layer (internal).
The failure of the system layer results in an external fault introduction in the execution layer. This external fault leads to an error state and may result in a consecutive failure of the execution layer. If this is recognizable to the user, the system fails and goes into the outage state.
An example from [13] is as follows: "The result of an error by a programmer leads to a failure to write the correct instruction or data, that in turn results in a [...] fault in the written software (faulty instruction(s) or data); upon activation (invoking the component where the fault resides and triggering the faulty instruction, instruction sequence or data by an appropriate input pattern) the fault becomes active and produces an error; if and when the error affects the delivered service (in information content and/or in the timing of delivery), a failure occurs." [13].

4.3. Dependability Attributes

The trust in the system behavior differentiates into several attributes. Those stress the different properties of dependability a system designer may emphasize. For

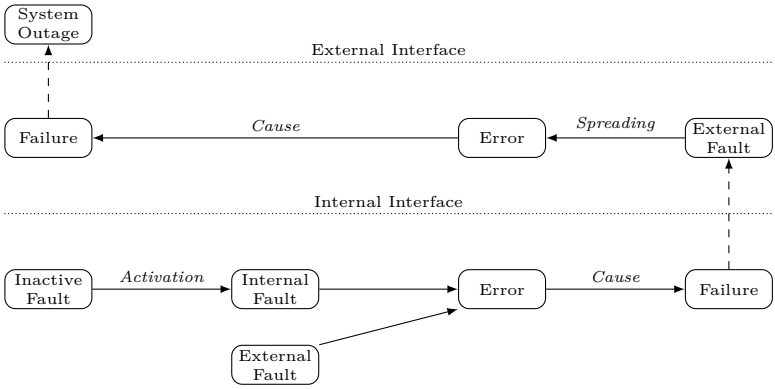

Figure 4.2.: Dependability chain of Laprie, transformed to the propagation between system layers. Taken from [109, p. 89].

example, web servers do not show a high degree of safety but high availability and maintainability. On the other hand, in automotive systems, maintainability is not a primary concern but safety.
As already stated, this chapter defines the basic terms of dependability, excluding the security-related ones. Therefore, this section defines dependability attributes, leaving aside confidentiality and integrity. Availability's definition is regarding its impact on the general system service. Section 5.4 describes security-related aspects of availability.

4.3.1. Maintainability

Definition 4 (Maintainability). *Maintainability is a system's design ability to change in order to stay in a working state.*

Maintainability is the property of a system to be repairable or changeable during its life-cycle. [70] at first define maintainability as "continuous service interruption" [70]. The later work of the *Laprie model* [71, 72, 14, 13, 15] defines maintainability as "ability to undergo repairs and modifications". [113, p. 228] arguments that maintainability is a system property of being restored or retained in a correct working state. It is, therefore, a property of system design while maintenance is the action, only possible in a properly designed system.
Although this is somewhat important also in automotive systems, the ISO standards (ISO 26262:2018 [53], ISO/SAE 21434 [54], and ISO/IEC 27000:2018 [55]) do not define this term.

4.3.2. Reliability

Definition 5 (Reliability). *Reliability is the property of continuity of correct system behavior and results.*

At first, the *Laprie model* defines reliability as "continuity of correct service" [69]. This definition is consistent with the consecutive papers [71, 72, 14, 13, 15].
While ISO 26262:2018 [53] and ISO/SAE 21434 [54] do not define the term reliability, ISO/IEC 27000:2018 [55] defines it as "property of consistent intended behavior and results" [55].
On the first view, the definition in ISO/IEC 27000:2018 seems slightly different from the one given in the *Laprie model*, but at a second glance, it is the same. The *Laprie model* describes service as the system behavior. Correctness in terms of the *Laprie model* includes the correctness and the continuity of delivery [13]. Since the system fails, if the external behavior deviates from the intended one, correctness means the intended behavior and the computation results. Therefore, the definition of ISO/IEC 27000:2018 is transformable into the one of Laprie and vice versa. Nevertheless, the one of ISO/IEC 27000:2018 states its meaning more apparent.

4.3.3. Availability

Definition 6 (Availability). *Availability is a system's readiness for correct service.*

Availability as "readiness for usage" [69, 71, 72] is the first definition provided in the *Laprie model*. Later the authors concretize the definition as "readiness for correct service" [14, 13, 15]. This definition includes the interrelationship of availability and reliability. Reliability describes the continuity of service without interruption. Availability assumes repairable systems and describes that the system is currently able to provide the intended service.
Other sources demand a certain timely manner of the provided service [41, 63], [43, p. 14]. [91, p. 12] has the most comprehensive description of influences on availability. It includes the usability and time, the fairness of resource allocation, and aspects of fault tolerance in the discussion. [92] also emphasizes the aspect of fault tolerance concerning data loss. Nevertheless, those are impairments or means for availability rather than a definition.
The definition of availability in ISO 26262:2018 is twofold. On the one hand, it defines the property of availability but also includes the quantifiable measure: "capability of a product to be in a state to execute the function required under given conditions, at a particular time or in a given period, supposing the required external resources are available" [53, Part I]. The first part of the definition fits the definitions of the other sources, focusing on the readiness to provide the expected functionality. The latter part includes the availability of external resources, which is not a core point of availability as a system property. The loss of external resources is a fault that the system analysis must incorporate into the system design.

It is no excuse if the system is unavailable. If there is the risk of a resource being unavailable, the system has to mitigate this, e.g., using fault tolerance mechanisms.

4.3.4. Safety

Definition 7 (Safety). *Safety denotes the absence of catastrophic consequences of a system failure that result in death, injury, illness, damage to or loss of property, or environmental harm.*

According to the *Laprie model*, safety is the "absence of catastrophic consequences on the user(s) and the environment" [14, 15, 13]. This definition results from the continuing refinement of the level to which these consequences should be degradable. At first, the authors use the word avoidance [69] which means "the action of preventing something from happening" [89] or "action of keeping away from or not doing something" [89] which emphasizes the lack of observation of catastrophic consequences. Later, they use the word non-occurrence [71, 72] which dual occurrence means "an incident or event" [90] or "the fact of something existing or being found in a place or under a particular set of conditions" [90]. This refinement focuses on safety as a negative outcome or influence on the system's environment. The word absence [14, 15, 13], like used in the later publications, means "the state of being away from a place or person" [88] or "the non-existence or lack of something" [88] and thereby emphasizes the result of the system modeling and analysis process: The complete absence of risks for catastrophic consequences.
[65] follows the *Laprie model* definition: "safety denotes the system's ability to fail in such a way that avoids catastrophic consequences. Thus safety is reliability with respect to catastrophic failures" [65]. The author regards safety as a "denial-of-service to the [u]ser and thus a violation of the specification" if this disruption of the delivery-of-service leads to catastrophic consequences [65]. On the other hand, safety may also relate to the illegitimate users in the way of a confidentiality failure which may also lead to catastrophic consequences [62].
The definition of the *Laprie model* and [62] are comparable to the definition of ISO 26262:2018: "absence of unreasonable risk" [53, Part I]. ISO 26262:2018 also focuses on reducing possible risks to the absolute minimum, whereby unreasonable risk is an ambiguous term. It is defined as "[r]isk judged to be unacceptable in a specific context according to valid societal moral concepts" [53, Part I]. Both definitions (*Laprie model* and ISO 26262:2018) are interpretable in their subject of the consequences. While the *Laprie model* authors do not precisely define catastrophic consequences, the ISO 26262:2018 leaves the definition to the societal moral concepts. Again, this is not exact since the moral concepts differ between countries and societies, especially regarding environmental consequences.
[18] on the other hand defines safety as "logical correctness (with respect to input/output specification) and temporal correctness (further requirement of [Real-time]-applications)" [18]. The source focuses on the way to analyze safety rather than giving a distinction to reliability and availability. Logical and temporal cor-

rectness might be usable to analyze reliability and availability, but safety can be seen as an attribute whether failure may lead to severe results.
[97] defines safety as an attribute "concerned with the occurrence of accidents or mishaps" [97], where mishaps are defined as "unplanned events that result in death, injury, illness, damage to or loss of property or environmental harm" [97]. By that, the definition is far more comprehensive regarding focusing on the consequences. Nevertheless, the core of the different sources is the same with different emphasis and subjects of the consequences: catastrophic consequences sourced by the system under consideration. Within this thesis, safety as an attribute is as described in the *Laprie model* with the definition of the user(s) and environment of [97]. Therefore, incorporating environmental consequences, as well as people's property.

Functional Safety is a sub-term to safety. According to IEC 61508 [45, Part IV], it is a part of the overall safety regarding the system and the safety functions. This is also the explanation used by [43]: "Something must continue to function in order to keep the system safe" [43]. ISO 26262:2018 argues about the "absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E-Systems" [53]. This argumentation emphasizes that functional safety narrows the safety term. Safety aims to prevent catastrophic failures, while functional safety tries to provide any service in times of failure situations. Therefore, functional safety aims to transit the system into a safe state, e.g., by graceful degradation to prevent further harm [94]. This thesis considers the ISO 26262:2018 definition of functional safety if needed.

4.4. Dependability Means

Dependability means classify and group the methods and tools to achieve the dependability attributes. Classic literature groups them into fault intolerance and fault acceptance approaches. The former tries to prevent the introduction of faults in development. The latter admits that it is impossible to have a fault-free system and aims to detect and handle the resulting errors to prevent failures during their customer usage time. In modern systems, all means are also usable at run-time [109, p. 79]. Therefore, this work relinquishes the distinction between fault intolerance and fault acceptance.

4.4.1. Fault Prevention

The aim of fault prevention is to prevent the introduction of faults into the system [71, 72, 14, 15]. The idea is that in an ideal system, the designer knows each fault source, and the fault introduction is, therefore, preventable [109, pp. 77 sq.]. Since this is impossible, fault prevention targets organizational and normative policies and best practices (e.g., MISRA-C [84]) that help prevent fault introduction as far as possible.
During the development process, quality control verifies that the developer adheres to the policies [14]. As those, Laprie et al., like most dependability sources, do not further evaporate over fault prevention since they count it to systems engineering, which is a separate research area.

4.4.2. Fault Removal

Fault removal aims to "reduce the number and severity of faults" [14, 15]. The *Laprie model* states that fault removal consists of three steps. In the first step, the system verification analyses the functional and meta-functional properties. If the verification fails, the diagnosis step identifies the reason. The last step corrects the fault, and the cycle starts again with verification [69]. In later work, the *Laprie model* assigns the fault removal cycle to the development process. Fault removal during the use phase consists therefore of "corrective and preventive maintenance" [13]. Meaning the correction of reported miss-functions or, e.g., the preventive exchange of hard drives.

4.4.3. Fault Forecasting

In fault forecasting, the idea is to identify faults in the system. The aim is not only to evaluate whether faults exist in the system but also the likelihood and the consequences of errors [69, 71, 72, 14, 15, 13]. Usable are quantitative and qualitative models and analysis techniques. A good overview regarding dependability modeling provides [109] and [13].

4.4.4. Fault Tolerance

Fault tolerance describes the system's ability to work correctly besides existing faults. Since the system reacts after detecting an error, literature sometimes calls it error tolerance [109, pp. 78 sq.]. Typically, fault tolerance relates to redundancy [71], since many approaches use redundancy in time or space, e.g., TMR systems or backups. However, fault tolerance consists of error detection and handling [14].

4.4.5. Discussion

Like [69] states, the means of dependability have different targets and should always be a combined approach. Fault prevention is regarding the system design, but the best design methodologies do not entirely prevent the introduction of faults into the system. Therefore, fault removal, combined with fault forecasting methods, tries to evaluate and remove the faults from the system. Introducing fault tolerance approaches helps to improve confidence in the trustability of the resilience of the systems and helps to guide maintenance activities.

In order to form an immaculate member of a flock of sheep one must, above all, be a sheep.

Albert Einstein

5 Security Taxonomy

Security is a relatively new topic in the automotive industry. The typical approach in such cases is to transfer the terminology, methods, and tools from other areas. The vast amount of differing terminology in other domains makes this problematic. Additionally, the varying definitions limit working in a development team without an agreed-upon vocabulary.
Therefore, defining a *taxonomy* as a tool to organize the concepts is helpful to have a distinct understanding throughout the complete development process. Traditionally, dependability incorporates also security (see Figure 5.1). Nevertheless, security is a large field whether methodology and terms substantially differ from the classic dependability approach. Therefore, the approach in this work is to transform the Laprie models dependability tree (see Chapter 4) into the security domain of automotive systems. This method helps maintain completeness to the same degree as the Laprie model. Section 10.2 defines a taxonomy of security requirements and categorizes the standard terms for threat mitigation. Combining both taxonomies enables a precise understanding of the terms related to automotive security and therefore fulfills the demands of usefulness and completeness of taxonomies [50].

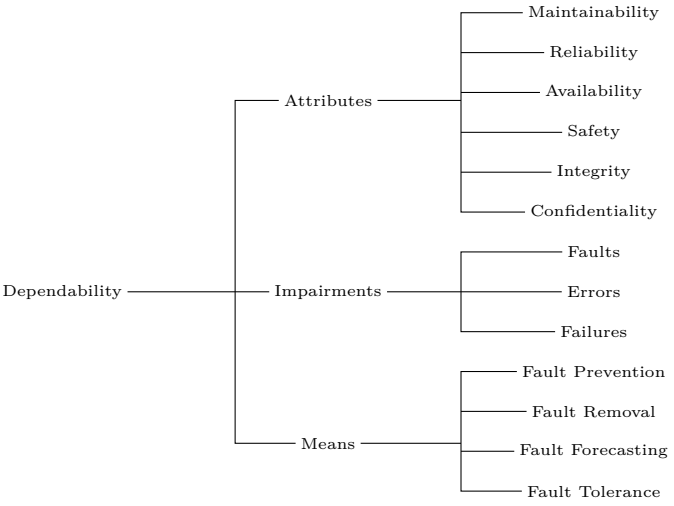

Figure 5.1.: Complete dependability tree of Laprie according to [13].

5.1. Related Work

At first, the *Laprie model* introduces security as a dependability attribute [69] before deleting security as a standalone attribute and integrating confidentiality and integrity [71]. This development reflects the growing discussions about security as demand for risk-prone systems [68]. Since [71] the definitions start to be continuous. Especially since [14] they stay the same.
[97] differentiates between the dependability and the security approach. The definition of the secure systems approach focuses on research on security kernels but also gives insights about similarities between fault tolerance and security kernels. The focus of [97] differs from the scope of this thesis, making the contributions not further applicable in this work.
[62] introduces the *User* and the *Non-user* to distinguish between the system's different types of service delivery. User and Non-user denote the system output regarding the delivery and the denial of service to the permitted user and the attacker.
ISO/IEC 27000:2018 ("Information technology – Security techniques – Information security management systems – Overview and vocabulary") is the norm for the vocabulary underlying the security standard series ISO/IEC 2700x. ISO/SAE 21434 is the standard "Road vehicles – Cybersecurity engineering" which partly adapts the terms from ISO/IEC 27000:2018 and ISO 26262:2018. In other parts, ISO/SAE 21434 redefines terms already defined by ISO/IEC 27000:2018. Especially ISO/SAE 21434 [54] seems pretty selective since it does not define confidentiality, integrity, and availability but relies on ISO/IEC 27000:2018.

5.2. Security

Literature provides several viewpoints to approach the general term security:

- from preventing threats and attacks
- the attributes of security
- the means for achieving security
- from resilience and preventing the result of an attack

ISO/SAE 21434 [54] as well as UNECE No. R155 [85] define security related to protecting assets from system threats. This viewpoint also follows [44] and [35] by stating that security is related to preventing threats and attacks on systems.
In [41] the term security is defined in terms of the properties a secure system should have: the classic CIA triad and secondary properties like authentication and authorization. The secure systems approach of [97] denotes, that secure systems have specific properties, the CIA triad. The same idea also follows ISO/IEC 27000:2018 [55] by defining security as a composition of confidentiality, integrity, and availability of information. Besides the varying focus on certain system parts, all approaches define security in terms of system properties.
Other sources approach security from a security violation's result or the other way around. [66] and [102] describe security as resilience to intentional faults. In [78], [63] and [76] the definition includes the environment as source for the negative influence to the system to be prevented.

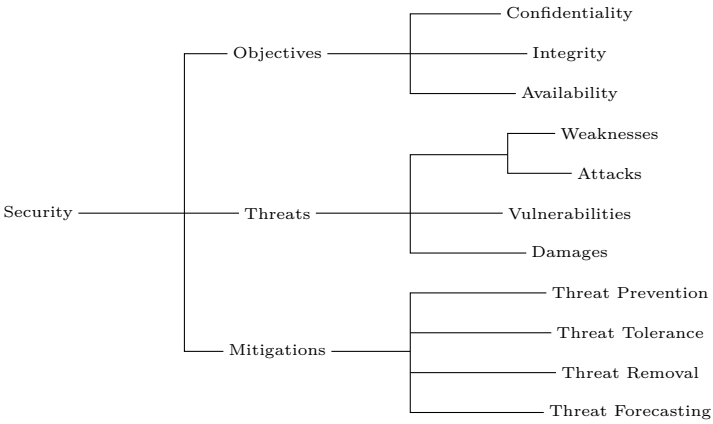

Figure 5.2.: Taxonomy of Security

Comparing the discussed approaches to define security leads to the conclusion that security is also an umbrella term [94]. The difference lies in the origin and type of faults the system needs to prevent. The attributes are a sub-set of the dependability attributes. At the same time, the means are derivable from the dependability means. The following sections describe the transformation of the dependability tree of Laprie into the security taxonomy (see Figure 5.2).

5.3. Security Threats

The term threat, like security, has varying viewpoints for definitions. One approach for defining threats is as "intentional, malicious and planned actions" [25]. [38, p. 20] classifies threats as events or circumstances which impact the system, thereby focusing on the result of the action. This focus follows [76] by stating, that a threat has "undesirable consequences"[76] and adds that security, in contrast to safety, focuses on intentional events. However, those intentional events are known as well as unknown scenarios. Including the former in the analysis procedures is easy. The latter are scenarios that may arise in the future [94]. Those relate to threats in the system currently unknown, e.g., faults in cryptographic procedures.
[63] definition of threats includes the environment as the intentional source of faults in the system. This view follows [63] by adding an environmental system (human being, other IT systems, or natural influence) as the source of the intentional events. [120, p. 2] on the other hand, targets only the source as a definition for the term threat.

Other definitions incorporate a threat's source and result for defining the term. [17] states that environmental systems force a violation of the systems security policies. In [91, p. 6] the definition targets the cause of an undesired event with the potential harm to the system.
UNECE No. R155 [85] and ISO/IEC 27000:2018 [55] define the term threats similar regarding the cause and the result. They extend the resulting harm, incorporating the system, the organization, and individuals. ISO/SAE 21434 [54] has a more elaborated definition than the other two norms. It defines threats regarding the impact on security attributes of system assets (see Definition 21), which results in a damage scenario.
While [66, p. 17] states that threats are intentional faults, the description and the examples fall into a different category. Malware as a threat example is an error, or even a system failure, depending on the viewpoint. Nevertheless, they define threats as part of a propagation chain.
When comparing the definitions of the term threat, it reveals that threat is, like impairments in the Laprie model, an umbrella term describing a chain of propagation from the hypothesized cause to the resulting security failure.
Following the preceding discussion, threats are comparable to the impairments of the Laprie model. Weaknesses, vulnerabilities, attacks, and damages are part of system impairments, like faults, errors, and failures. Therefore, the remaining work uses the following definition of threats, whether distinction and interrelation Figure 5.3 illustrates.

Definition 8 (Threat). *Security impairments with the potential to violate a security objective.*

5.3.1. Weakness

Definition 9 (Weakness). *A weakness is an internal fault propagating into the system's lifetime, leading to the system being in a vulnerable state.*

[13] does not directly define the term weakness or name the underlying fault of security threats. Nevertheless, the necessity for a vulnerability as an internal fault result states that there is a fault propagating into the system's lifetime. Another possibility are external faults introduced during system operation.
ISO/SAE 21434 defines weaknesses as a "defect or characteristic that can lead to undesirable behaviour" [54]. As already stated, the term defect may be a substitute for fault or failure [109, p. 85]. The most prominent sources supporting defect as a synonym for fault are the ODC [21] and the IEEE Standard Glossary [47]. Since ISO/SAE 21434 does not define other security threats, it is impossible to interpret defects compared to other threats definitions. Therefore, this work interprets a defect as a fault, like in the mentioned sources.
[38] relates weaknesses to "mistakes in the design of a system"[38, p. 22]. Following Laprie, each fault is regarding the system design (see Section 4.2) and may be internal or external. Internal faults originate in the internal system state [13].

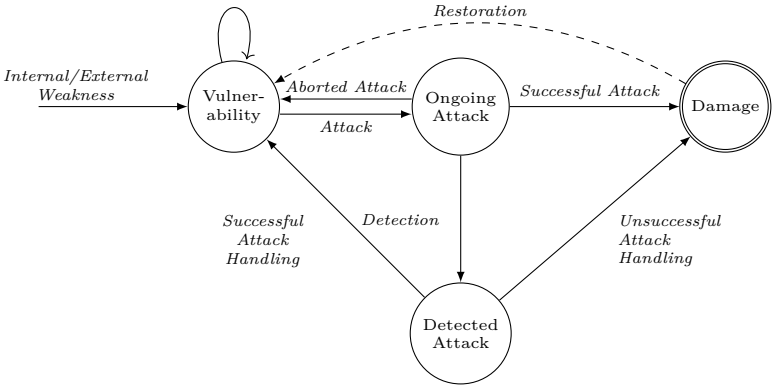

Figure 5.3.: Illustration of the security threat chain.

Comparing the definitions for weakness and vulnerability, provided by [38, p. 22] allows the interpretation of weakness as an internal fault.
UNECE No. R155 [85] defines the term weakness indirectly through vulnerability. A weakness relates to the system leading to an exploitable vulnerability. The link to the system allows for identifying weaknesses as internal faults residing in the system.
Literature shows that weakness is related to the term fault in the *Laprie model* and may have different sources, like faults. Mainly, the system design phase introduces weaknesses that propagate into the use phase. The nature of security-critical systems is that it is impossible to foresee every threat to the system. There is always the possibility of a newly revealed weakness residing in the system. Therefore, a security-critical system is in a vulnerable state during its lifetime.

5.3.2. Attack

Definition 10 (Attack). *An attack is a malicious environment action that targets introducing an external fault into the system.*

The security's viewpoint on intentional events demands an attacker to launch an attack by introducing a malicious external fault [65]. External faults are those integrated into the system by "interaction or interference" [13]. Therefore, like weaknesses, attacks are actions of fault introduction. Weaknesses introduce internal faults into the next system layer. Attacks introduce malicious external faults.
[120, p. 17] describes this mapping directly: "The attack itself is analogous to an error in that it is the manifestation of a vulnerability"[120, p. 17]. The vulnerability is the possibility of an exploit [91, p. 6].

It leverages an attack to the system [119, p. 17], [96, p. 58], [96, p. 17]. While [120, p. 17] describes the attack as a system state, it is rather a trace of system transitions, depending on the type of attack. The ongoing attack manifests in the system state, but each step is a system transition.
An attack may be successful or not [44, p. 53], [63]. Also, it can aim to introduce a weakness into the system [65]. In this case, the attack is successful if it adds a weakness to the system.
Depending on the type of attacker, an attacker may abort an attack. The typical case is whitehat hackers. That attacker aims to realize an attack but does not harm the system. They instead share their knowledge with the manufacturer [48].

Definition 11 (Attack Path). *An attack path is a set of actions describing how an attacker realizes damage to the system.*

An attack is a particular case of a system state where the attacker exploits the vulnerability and starts an attack path. A "set of deliberate actions to realize a threat scenario"[54] where a threat scenario is the cause of an assets damage scenario. Depending on the type of attack, the attack state may lead to another attack state. Therefore, the attack is an environmental action, starting a series of attack states whether length is not determinable beforehand.
Attack paths are also called attack vectors [38, p. 22]. They describe the way the attacker uses to gain its goal [94].

5.3.3. Vulnerability

Definition 12 (Vulnerability). *Vulnerability is an incorrect state in which a malicious external fault could attack the system.*

[13] states, that a necessity for a vulnerability is the existence of "an internal fault that enables an external fault to harm the system"[13]. This view also follows [63] as an "attack will be successful if it can exploit a vulnerability in the system"[63]. Further amendments in [13] are that vulnerabilities result from faults during development (internal faults) or use (external faults). The existence of a fault leads to the system being in a latent error state.
[38] denotes that vulnerabilities reside in the system and result from weaknesses: "Vulnerabilities represent a materialization of design weaknesses" [38, p. 23] which are usable for attackers. This definition allows to relate vulnerabilities to the active fault/latent error state in the *Laprie model* (see Section 4.2). [66, p. 17] states that a vulnerability is an exploitable fault, which supports this opinion. [91, p. 6] adds that a security control stops a threat in the vulnerability state.
While ISO/SAE 21434 does not define the term vulnerability, UNECE No. R155 defines it as "a weakness of an asset or mitigation that can be exploited by one or more threats" [85]. Vulnerability is an exploitable weakness, which means the weakness is in an active state. Compared to the Laprie model, this is an incorrect system state, an error. The incorporated threats are external faults that exploit the vulnerability, named attacks.

The evaluated literature shows that comparing the terms weakness/attack/vulnerability and fault/error is not straightforward. There is the need to change the viewpoint from general dependability to a certain sense of security. While dependability relates to unintentional events, security treats intentional actions. This intentional action introduces an external fault (attack) that exploits an incorrect system state (vulnerability) and may result in a system failure (damage). Beforehand, the system is already in an error state, the vulnerability state. The attack exploits the vulnerability and leaves the system in the error state of an ongoing attack. This state leads to damage, or detection and handling are possible. Therefore, vulnerability is a system state that deviates from the intended one because it violates security objectives. This deviation is not recognizable during regular operation until a detected or successful attack.

5.3.4. Damage

Definition 13 (Damage). *A successful attack can lead to a system state where the system behavior regarding one or more security objectives deviates from the intended one, recognizable by the user.*

As already stated, the evaluated literature sources leave out some of the terms needed for a complete taxonomy on the terms of security. A good example is the failure state. Typically the sources do not explicitly notice this fact since it is logically deducible. Sometimes, the sources use the term failure and give examples for security failures, e.g., [66, p. 18]. In this case, the term failure has the same meaning as in classic dependability: The system cannot fulfill its specified service, recognizable by the user.
[63] distinguishes between classic and security failures based on the system output regarding the intended user and the attacker. In ISO/SAE 21434, the threat scenario describes a successful attack and, therefore, a failure to the system. Where a threat scenario is the resulting system failure: “adverse consequence involving a vehicle or vehicle function and affecting a road user” [54].
The damage a successful attack may cause is regarding unauthorized access to the system resulting in [55, 54, 46]:

- a modification or destruction of the system
- the change or use of information

In cases where the attackers aim to plant a weakness in the system, the attack intends to exploit it later as part of another attack. This dependency depicts error propagation in the *Laprie model* (see Figure 4.2). The introduced external fault is a weakness, activated as a vulnerability and exploited through an attack.
Restoration relates to removing external faults introduced by an attack. From the system point of view, a damage state is always restorable. However, from the user’s point of view, it depends on the type of damage.
Attacks that only aim to place a weakness into the system, e.g., Viruses or Trojan horses, have no direct damage to the user. Those are removable by system restoration. The underlying cause for the fault introduction remains in the system.

This cause is only removable through a design or implementation change, e.g., a system update. In case of a successful attack regarding confidential information, the system returns to the vulnerable state after the attack. The damage to the user or manufacturer still exists. From their point of view, the system remains damaged. At first glance, restoration may be questionable in the case of security since it is useless to remove the damage without the underlying cause (the weakness). Restoration is always just the first step in handling a successful attack. Every attack (attempt) needs a consideration of the enabling weakness and measures to remove it. Besides the theoretical viewpoint that restoration and threat removal are two separate steps, they may be simultaneous in practice. Depending on the damage type, it is recommendable to do no roll-back with a subsequent threat removal but a re-design of the system. This re-design might be a new installation with a change, e.g., removing some functionality or introducing new defense methods.

5.3.5. Propagation Chain

The presented definitions of security threats lead to a transformation of the dependability propagation chain of Laprie. Figure 5.4 includes both viewpoints, dependability impairments, and security threats in one picture. Since propagation in security is regarding the development and the customer usage time, the figure differentiates not between system layers like in Figure 4.2.

In the traditional dependability concept, a fault in one layer (e.g., hardware) is an internal fault manifesting as an error and probable a failure in this layer. This failure is regarded an external fault in the next layer (e.g., software).

Transferred to stages of system development and customer usage time, this propagation into the next layer is no external fault introduction. The failure in development leads to an internal fault residing inside the system during customer usage time. Nevertheless, the propagation inside one layer stays the same. Solely the transfer between layers is different.

Development faults can be either malicious or non-malicious. The former denotes the intended introduction of faults into the system. The latter are common development faults. If not revealed during the development process, they manifest in the system, leading to a development failure since the system deviates from the intended design and function.

Non-malicious development faults manifest as internal faults in the lifetime and follow the Laprie propagation chain into an error state, possibly leading to system failure. A non-malicious external fault leads to a system's transition into an error state. Another possibility is that the external fault leads to a non-deliberate attack state, e.g., data corruption or confidentiality breach. In this case, the system interaction manifests as a state which deviates from the intended security behavior. As long as this state is not recognized, it is no damage. An example would be a user who accidentally leaks system information. If no other user reveals this information, there is no damage.

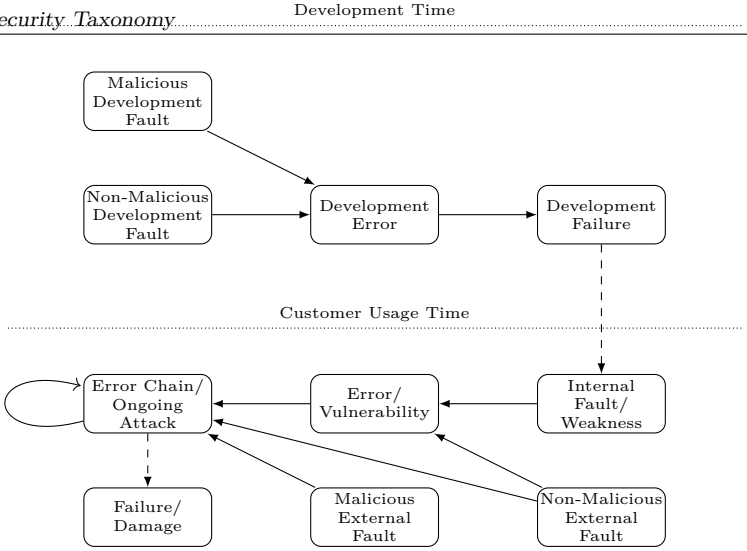

Figure 5.4.: Incorporated dependability and security propagation chain. The security threat chain of Figure 5.3 applies to each layer.

Malicious development faults manifest as weaknesses in the system during its lifetime and lead to the system being in a vulnerable state. Also, not every weakness is foreseeable. Therefore, from a security point of view, the system is always in a vulnerable state. A malicious external fault leads to an attack and the system state transition into an ongoing attack state. The system cycles in this state as long as the attack is not detected or successful. The system can sustain damage if the attack is successful.

5.4. Security Objectives

Security objectives have varying names, the most prominent are attributes or properties [38, p. 15], [54], [102]. This work uses the more common and general term objectives. Security objectives are not to confuse with security goals. Those are a combination of an objective and a threat scenario [54].
In security, the term CIA-triad describes the set of security objectives. There exists an ongoing discussion on whether the *A* stands for availability or authenticity. Nevertheless, more common is the opinion that the CIA triad incorporates availability [13, 97, 41, 63]. The *Laprie model* describes the coherence as "associating integrity and availability with respect to authorized actions, together with confidentiality leads to security" [71, 72]. Authenticity is a secondary attribute.

Vehicle functionalities rely on communication. Therefore, the automotive industry further distinguishes the CIA triad into data and ECUs, e.g., data integrity and ECU integrity. Such distinctions narrow the scope of the analysis but have no further benefit in the general term definitions. This work defines the security objectives regarding information flow in a system. Therefore, their focus is already on communication in this sense.

5.4.1. Confidentiality

Definition 14 (Confidentiality). *Confidentiality is a behavioral property describing the prevention of information flow to unintended users and the availability of information flow to intended users. Information flow in this context means everything related to knowledge of information.*

The *Laprie model* relates confidentiality to the "unauthorized disclosure of information"[69, 71, 14, 13, 15] which may be accidental or intentional [69]. Accidental confidentiality damages arise, e.g., from misconfiguration in the sense of wrong access control configurations.
[62, 65] relate confidentiality to system behavior as denying information access to illegitimate users. Therefore, confidentiality is a behavioral concept of the system's service, and its achievement demands denial-of-service to illegitimate users [63].
Other literature sources follow the definitions of the *Laprie model* and [62]: Confidentiality is commonly seen as the prevention of unauthorized disclosure of information [66, p. 18], [91, p. 10], [41, 92, 87, 76] [96, p. 6]. [91] emphasizes that information access in this context is "not only reading but also viewing, printing, or simply knowing that a particular asset exists"[91, p. 10]. [92] adds that confidentiality includes access control.
ISO/SAE 21434 relies on ISO/IEC 27000:2018, which defines confidentiality as "information is not made available or disclosed to unauthorized individuals entities or processes" [55]. This definition emphasizes both confidentiality aspects: the access to information to the intended user and the denial of information to the unintended user.
Although the definition of confidentiality seems straightforward at first glance, understanding what unauthorized access to information means is tricky. Preventing unauthorized access and availability of access demands access control to information. Besides the obvious use of access control in operating systems and file access, the question is how to define it concerning information flow in a general sense. Information flow is everything that leads to knowledge about information. Following the discussed literature, this knowledge extends from the existence of information to its use.
In conclusion, confidentiality damage results from a vulnerability where an external fault enables the information flow to unintended users or prevents the availability of information flow to intended users. Information flow in this context means everything related to knowledge of information. Examples are data from the OEM and implementation details.

The system sustains damage if the confidentiality breach results from a malicious source. Otherwise, a confidentiality breach is possibly non-malicious, and the system does not reach the damage state. The confidentiality damage is permanent damage that is not restorable. Restoration targets only the weakness – the originating weakness results from a development failure, typically due to missing or miss-implemented access control mechanisms.

5.4.2. Integrity

Definition 15 (Integrity). *Absence of improper system state modification.*

In 1992 [69] recognizes integrity as a main attribute of dependability. There, the definition of integrity is "the prevention of the unauthorized amendment or deletion of information" [69]. [69] argues that integrity is "the condition of being unimpaired, in the broad sense of the term: a) for either data or programs, and b) with respect to either accidental or intentional faults" [69]. Later, the *Laprie model* authors slightly change the definition to "non-occurrence of improper alternations of information"[71, 72]. Both definitions fit the argumentation that integrity is a part of protective security against fault introduction by the environment (intended and unintended user) [62, 65].
In the later papers, [14, 15, 13] define integrity as the absence of improper system alternations. Therefore, integrity "relates to the notion of authorized actions only" [15], focusing on information and the alternation of information by violating the authorization policy [14, 15, 13].
The general security literature provides several definitions for integrity. [91, p. 10] and [76] target integrity to authorization regarding the modification of assets. As with confidentiality, [91] explicitly states the types of modification: "writing, changing, changing status, deleting, and creating" [91, p. 10]. While [87] relates integrity to authorization of modification, the authors target only data integrity stating that the system is responsible for achieving integrity. [41] incorporates the correct modification of information . [92] subsumes and exceeds the initial scope of integrity to "consistency, coherence and configuration of information and systems, and preventing unauthorized changes to them" [92].
Analyzing publications in automotive security reveals that many articles immanently assume a definition of the term integrity. [6] focuses on verifying data integrity rather than the overall system and claims protection from unauthorized or undesired changes. [66] state that integrity relates to "intentional or accidental system alternations" [66, p. 18]. The type of alternation covers the idea that non-malicious faults can endanger security objectives.
The ISO/SAE 21434 [54] does not define integrity and just refers to ISO/IEC 27000:2018 [55] which defines integrity as the "property of accuracy and completeness"[55].

As shown in [61], the definitions target different abstraction levels and need a redefinition in the application domain context. For example, Avižienis defines integrity as a property of the system itself. ISO/IEC 27000:2018 has the most abstract definition, targeting the provision of service.
Integrity relates to an attack that aims to introduce a weakness into the system by targeting a system state modification. The weakness is permanent, where the attacker modifies the system's software or modifies run-time data. In the first case, the system sustains damage as a permanent modification if the attack or modification is not detected and the system restoration occurs. If the attacker modifies data in the system, the damage is temporary till the system recognizes the attack or deletes the data. Integrity damage can spread to other system parts if the damaged system communicates this data to other systems. In this case, the communicated data is an external fault introduction at the receiver side which starts an unintended attack.
An integrity attack may also be unintended. One example is a user who modifies data in the system in good faith without knowing that this modification is forbidden or wrong. Nevertheless, the result is the same as in the malicious case.

5.4.3. Availability

Definition 16 (Availability). *Availability is a system's readiness for usage to an authorized entity without unauthorized withholding of information.*

The security standard ISO/IEC 27000:2018 [55] defines availability towards the security aspect as the "property of being accessible and usable on demand by an authorized entity" [55]. Nevertheless, the common definition of availability is an extension emphasizing authorized entities. This follows the idea of [62]: "ability of the system to deliver its service to the authorized user" [62]. The author extends the property to preventing "unauthorized withholding of information" [62] which is the same as with [69]. This extension underpins the classification of [62] to see availability as an attribute of the correct system behavior for the user or the delivery-of-service. At the same time, it is no attribute to the illegitimate user.
[66, p. 18], [92], [41], and [96, p. 6] relate availability to the general definition without setting it in a security context. [76] targets availability of information towards authorized users.
An attack on availability introduces an external fault (malicious or non-malicious) that prevents a system service from operating correctly. Thereby, the resulting damage is the unavailability of the service to the user or the availability to unintended users. This definition of availability includes preventing information flow to the user and thereby confidentiality. Thereby, it emphasizes, again, the symmetry of both properties.

5.4.4. Secondary Objectives

Secondary objectives are specializations of the primary objectives [13, 15]. This specialization relates to the object the objective refers to, e.g., regarding the user or transactions [41].

Authorization - [41] relates authorization to users as "allowed or denied access to resources" [41]. In [69] authorization is defined from the dual viewpoint. If a user abuses his access rights without malicious actions, he violates authorization. By that, he changes his status from a user to an attacker [69].
[96, p. 4] specify that authorization applies to information and actions. Therefore, access rights to information and system functions fit the users' needs. This view emphasizes that authorization is usable as expedient to ensure the system objectives.

Definition 17 (Authorization). *Authorization is expedient to ensure the primary system objectives by access control to information and system functionality.*

Non-repudiation as a system property relates "to an action and an entity performing the action" [96, p. 5] . It means that "Users can't perform an action and later deny performing it" [41].
Therefore, non-repudiation is a composition of availability and integrity. If related to information, it is possible to distinguish between the non-repudiation of the source and the sink of the performing entities' identity [15, 13]. Nevertheless, an action engaged by a user is the exchange of information and has an origin and a receiving side. Therefore, it is legible to abstract non-repudiation to actions instead of further distinctions.
[96, p. 5] adds that non-repudiation may be a legal obligation or introduced for liability.

Definition 18 (Non-repudiation). *Non-repudiation is the availability and integrity of actions and the involved entities.*

Authentication defines [41] as the property of proving the user's identity. More detailed is the definition of [92] as "ensuring that inputs to, and outputs from systems, the state of the system and any associated processes, information or data are genuine and have not been tampered with or modified" [92]. Inputs, outputs, and processing relate to information and data change. This change results in a modified system state. Therefore, the definition of [92] relates to data authenticity. In [96, p. 4] define data authenticity concerning the true origin of data and the author. Like [15, 15] the idea is to add information like time and place of creation to the information. By that, the integrity of the information and its origin is provable, encompassing authenticity.

Definition 19 (Authenticity). *Authenticity implies the integrity of information and its origin.*

Privacy The traditional sources discussing dependability/security terms do not include privacy. Privacy is a topic that arose in the age of constantly rising connectivity and by that simultaneously to the late works for the *Laprie model* and [64]. Therefore, they do not discuss the term privacy. Nevertheless, current standards like, e.g., ISO/SAE 21434 [54] demand include privacy in the security analysis. Therefore, this work includes privacy as a secondary objective to security.
[3] defines privacy as the objective that persons have a complete view of the usage of their personal information by others. This definition is also included in the E-safety vehicle intrusion protected applications (EVITA) project [66, p. 18]. Another work product of the EVITA project further defines privacy as a property of "an entity and a set of information"[96, p. 5] related to confidentiality. Therefore, privacy is an objective of anonymity of the system user and confidentiality of its sensitive information.

Definition 20 (Privacy). *Privacy is the property of anonymity of the system user and confidentiality of its sensitive information.*

5.5. Security Mitigations

In security, mitigations typically relate to reducing risk [54, 85]. Transferring the traditional dependability means of fault prevention, removal, forecasting, and tolerance to security would substitute fault by risk. However, those mitigations do not remove risk but threats, which are the risk sources. Therefore, this work uses the term threat instead of risk.
Additionally, ISO/SAE 21434 includes risk retaining and sharing into the mitigations. Those mitigations are not subject to the security taxonomy. Although risk retaining and sharing is common in practice, it should be no primary way to treat system threats. Also, the extent to which risks are accepted is subject to development process planning. Therefore, it is not part of the general development process but determined initially, out of the scope of this work. For further information, [93, p. 44] compares different risk acceptance criteria.

5.5.1. Threat Prevention

In general, threat prevention is directly comparable to fault prevention. The aim is to prevent the introduction of weaknesses into the system.
Like in fault prevention, the way to achieve this is by policies and best practices. Those policies have different sources like organizational policies, normative sources, or external policies. Examples for normative sources in automotive industry are ISO/SAE 21434 [54] and UNECE No. R155 [85]. Also fault prevention techniques like MISRA-C [84] are applicable in threat prevention.
Threat prevention is subject to the complete development process.

5.5.2. Threat Removal

Threat removal has the same drawbacks and methods as fault removal. It is simply impossible to eliminate all system threats. Therefore, it is reasonable to include also threat reduction into this category
Approaches for threat removal include treatment and structural methods. Treatment methods are the subject of Chapter 10. Removing threats by structural methods also includes parts of treatment, but the simplest way is by excluding a system part from, e.g., communication or the complete system. During the system lifetime, updates are also a method of threat removal. Examples are updates from cryptographic libraries in order to remove insecure routines.
Therefore, threat removal is subject to the complete development process and system lifetime.

5.5.3. Threat Forecasting

The normative sources for automotive development demand specific analysis steps, e.g., security risk evaluation and risk assessment. Those are part of threat forecasting: system analysis regarding threats, likelihood, and impact on the system. The later chapters provide further information about the system analysis steps.
Threat forecasting is a recurrent task throughout system development.

5.5.4. Threat Tolerance

The term threat tolerance seems to be dual to the security analysis concept. Nevertheless, the number of security threats is constantly changing. Examples are newly revealed threats or successful attacks. Therefore, the exact number and nature of threats are never known, and removing all threats from the system is impossible. There is always the possibility of malicious or non-malicious attacks. Therefore, the system needs a concept to continue fulfilling its purpose in the presence of threats - called threat tolerance.
Threat tolerance treats this fact by approaches to detect and respond to system threats [73]. Examples are monitoring systems like intrusion detection systems but also threat databases.
Methods of threat tolerance are subject to the treatment step in system development (see 10).

5.5.5. Discussion

The similarities between threat mitigations and dependability means are apparent. Also, the interrelationship between the different mitigation techniques is the same [15].
Threat prevention reduces the number of introduced threats during development. Threat forecasting is necessary in order to reveal the remaining system threats. These methods enable threat removal to reduce the number of threats in the system. Since it is impossible to have a threat-free system, threat tolerance supports detecting and handling threats during run-time.

Everything should be made as simple as possible, but not simpler.

Albert Einstein

6 Terms and Definitions

The previous chapters defined the terms from the *Laprie model* and transferred the idea of Laprie to the security terms. Nevertheless, there are further terms needed throughout this work. Such terms' definitions define this chapter in alphabetical order.

Asset - In the most sources, an asset is related to something which has value for a stakeholder and assigned security objective(s) [54] [38, p. 15]. [96, p. 58] is more distinct and states that an asset's nature is that they are the goal of attacks. Both definitions target the same issue with a dual view. [66, p. 15] adds that assets are often related to data, but their scope is rather anything that has the potential for being attacked.

Definition 21 (Asset). *Assets are related to a system, fraud with security objectives, and can potentially be attacked.*

Attacker - Individuals or organizations which aim to violate a security objective maliciously are called attackers [38, p. 21]. They exploit a vulnerability by unauthorized actions [106] in order to start an attack [96, p. 58].

Definition 22 (Attacker). *Source of an external fault introduction during customer usage time to launch an attack.*

Customer usage time - The automotive system's run-time is when the vehicle is in use. In automotive systems, it is helpful to have a separate term for the time after the vehicle leaves the OEM and is in use by the customer. This time spans the complete helpful lifetime of the car, irrelevant if the vehicle is parking or driving.

Definition 23 (Customer Usage Time). *Time-span of the automotive system from going into use till decommissioning.*

Risk - There are two viewpoints to assessing the term risk. [91, p. 506] and [120, p. 11] target the term from the result viewpoint. The risk is the potential for the customer usage time or the system to be harmed.
Other sources target risk from the evaluation point of view. ISO/SAE 21434 defines risk as "effect of uncertainty on road vehicle cybersecurity expressed in terms of attack feasibility"[54]. Therefore, ISO/SAE 21434 misses the influence of the damage scenario impact. Nevertheless, commonly risk is the combination of attack feasibility and the damage potential [38, 85, p. 19], [102], [96, p. 61].

Definition 24 (Risk). *Risk is the probability of occurrence of an attack and its impact of it.*

Validation - One part of the development process is evaluating whether the resulting system accomplishes the stated goals [54]. The idea is to confirm that the development requirements were correctly understood and implemented.

Definition 25 (Validation). *Validation is the evaluation of the system's conformity with the development goals.*

Verification - The idea of verification is the evaluation of the requirements fulfillment [54]. Verification evaluates the systems correctness, completeness, and consistency [95, p. 17].

Definition 26 (Verification). *Verification is the system analysis regarding the fulfillment of the development requirements.*

Part III.

Security Development Process Design

ISO/SAE 21434 outlines the necessary development process steps, along with possible methods. The standard divides the development cycle into three steps relevant for this work: security relevance evaluation (SRE), security risk assessment (SRA) and risk treatment. Transferring this outline to a function-oriented development environment, the process steps on the V-Models left leg divide into four layers:

Planning: The security department defines the security development process [2].

Function: The security relevance and risk analysis on a functional level takes place.

Component: Composition of the functional risk analysis results based on the deployment, risk analysis on system level, and risk treatment.

System: Derivation of the requirement specifications on function and component level – evaluation of the overall architecture.

The idea of the security relevance evaluation (SRE) as the first evaluation step is the general relevance of a given item regarding cybersecurity. Due to the function orientation, the SRA step divides into two layers. First, each relevant vehicle function is subject to the subsequent function-oriented SRA. Following the final deployment, the system SRA targets threats to the functions resulting from the hardware structure. The identified risks are subject to risk treatment to mitigate the threats. This step assigns different defense methods to each attack path.
This thesis focuses on the function and component level and the transfer to the system level. Figure 6.1 illustrates the parts and relations of this work's focus.

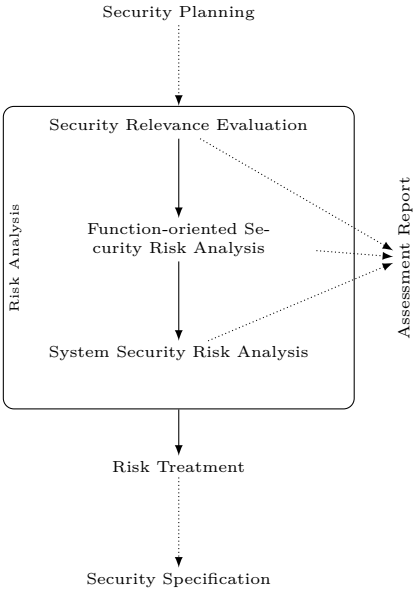

Figure 6.1.: Security development process parts and relations in the left leg of the V-Model. Dashed arrows depict relations to process steps out of the scope of this work.

The important thing is not to stop questioning.
Albert Einstein

7 Security Relevance Evaluation

The security relevance evaluation (SRE) is the first activity in the security analysis process at the functional level. It is an efficient method to determine the lower bound of the security criticality before proceeding to the SRA and is, therefore, part of the overall risk analysis process.

7.1. Process Description

ISO/SAE 21434 [54] defines the security relevance evaluation (SRE) as an item-related planning step.
UNECE No. R155 does not explicitly demand this process step. Nevertheless, the amendments of UNECE No. R155 suggest to use the SRE process step in the development process by referring to ISO/SAE 21434 [86].
Due to the nature of the SRE, no other related work is available.

7.1.1. Normative Background

ISO/SAE 21434 defines neither input nor work products but only describes the process step itself: "[...] the item or component shall be analyzed to determine: a) whether the item or component is cybersecurity relevant; [...] b) if the item or component is cybersecurity relevant, whether the item or component is a new development or a reuse; [...] "[54].
If the item is not security-relevant, the subsequent security process excludes this item. Nevertheless, the final security assessment report includes the items evaluation result. Therefore, a judgment is necessary regarding the acceptance or rejection of the items security [54].
For conducting the SRE, ISO/SAE 21434 suggests two different methods. The first method is to determine the security relevance "based on experience and multiple expert judgments" [54]. Security engineers have different experiences and backgrounds. Their valuable experience is a necessary prerequisite for a successful security development project. Nevertheless, in a distributed development environment, where a group of security engineers accomplishes the SRE an approach that depends on expert judgment may lead to different results. Therefore, such methods are efficient but tend to be less objective, hindering reproducibility.
The second possible method for SREs is using the question catalog provided in Annex D of ISO/SAE 21434 [54]. If one of the questions answered yes, the item may be security-relevant and proceeds further in the security process. Approaches based on such question catalogs enhance objectivity. Another advantage of this method is that it enables an automatized tracing of uncertainty and critical elements of the assessment.

Requirements - Evaluating the normative reference and the aim of this thesis leads to the following requirements:

- **R7.1** The method must incorporate the criticality evaluation comprehensively and straightforwardly.
- **R7.2** The SRE method must support the explicit designation of uncertainty.
- **R7.3** SREs must support the tracing of uncertainty.
- **R7.4** SREs must be realizable in a time efficient way.

7.1.2. Implementation in the Development Process

A straightforward implementation of the SRE is using a questionnaire and applying it to every function (R7.4).
The evaluated process uses a spreadsheet that extends the questions from ISO/SAE 21434. The extension targets OEM related questions and general information about the item and the current vehicle project. Like in ISO/SAE 21434, each question has two answering possibilities, yes or no resulting in dropping out of the security development process or not.
After the risk assessment, ISO/SAE 21434 demands a recommendation for each item's level of risk acceptance. This recommendation needs a notion of criticality (R7.1) for the item to provide an objective rating.

		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 7.1.: Risk Matrix taken from ISO/SAE 21434 [54]. For the SRE criticality, the attack feasibility rating would be the exposure of the function; the impact rating is the highest damage potential in the questionnaire.

The risk assessment in ISO/SAE 21434 [54] determines the risk for an item using the impact of the damage and the likelihood of an attack. Transferring this approach to SRE demands a rating of the questions regarding their impact. For this, the evaluated process includes several additional questions for damage scenarios rated by predefined answers. Those depict the damage potential of the item. For the attack likelihood, a question regarding network interfaces gives an initial idea about the item's exposure. Using the highest-rated damage level and the most critical attack surface, the risk matrix from ISO/SAE 21434 (see Table 7.1) allows for identifying the criticality.

7.1.3. Problem Description

The evaluated process extends the question catalog of ISO/SAE 21434 by questions related to the OEM. Additional questions support the determination of the item's impact and exposure and ultimately the item's criticality (R7.1). In parts, the question catalog and the damage catalog ask for the same information. Therefore, it is subject to evaluation and adjustment.
Especially in the early stages of the development process, information may not be available. This information is acquirable in a follow-up. By that, it is possible to verify that the criticality of an item does not change through that data. Existing information requires tracing to avoid missing any updates. Also, tracing enables refining and deriving of requirements in later process steps. The current version of the SRE questionnaire miss the possibility to make uncertainty explicit (R7.2) and use tracing information for later recaps (R7.3). This lack leads to situations in which expert judgment about the functionality and criticality of the item determine the damage potential. Especially in distributed development teams, this approach lacks objectivity and reproducibility.
Automotive development projects typically have hundreds of different software functionalities. Every functionality needs to be evaluated based on the question catalog.This procedure leads to a high effort, with comparatively few items proceeding further in the security process. Therefore, the SRE process step has much potential for enhancing efficiency (R7.4).

7.1.4. Approach

One idea for enhancing efficiency in the SREs (R7.4) is to automatize this process step. This automation would also directly enable the tracing of uncertainties and results (R7.3, R7.2). Nevertheless, this approach has several drawbacks: It demands integrated and usable models or databases. Early development stages tend to use a variety of models for the different types of functionalities. Also, automation depends on a strict administration of the input models and is vulnerable to tool changes in the input sources. Those drawbacks leave this approach inapplicable. Nevertheless, the automation idea is applicable for updating information and follow-ups.
Function-oriented development uses a catalog of possible functions in the early development stages. The serial development includes not all of them but leaves out specific variants. Also, the catalog of functions shows similarities between the functions usable for clustering those functions. This clustering reduces the number of single SREs and enhances efficiency (R7.4).
ISO/SAE 21434 demands a SRA for every relevant item. Therefore, an extension of the cluster approach would be to directly do SRAs for the identified clusters. ISO/SAE 21434 does not strictly demand the use of the questionnaire but allows to determine the relevance "based on experience and multiple expert judgments, e.g., involving safety experts and cybersecurity experts" [54]. Therefore, ISO/SAE 21434 justifies this approach. With that approach, SRE and SRA use the same method. By that, the incorporated persons have fewer tools to use, enabling higher acceptance. SRAs have a high effort compared to SREs (R7.4). On the other hand, the results may provide early information for basic security requirements in this cluster. Those allow tailoring of the vehicle architecture in early development stages. The development process would use the same interface throughout the analysis steps for tracing and refining results (R7.3). On the disadvantage side, this approach needs deep adjustment in the SRA process. Clustering should reduce the number of analyses in the SRE step as much as possible. Therefore, the items in the cluster still have much difference. The tool needs to be able to model those kinds of differences and similarities. The most important drawback is that SRAs needs much and detailed information about the item. That information is not available in the early development stages. Accomplishing that information can be impossible and is subject to many changes till serial development.
Using the questionnaire approach for cluster SREs, the analysis steps in the development process use different methods. This is a disadvantage regarding tool acceptance and training effort, but an advantage for tracing (R7.3). Both analysis methods produce different results, which are subject to tracing. Their difference makes it possible to formulate dependencies and directly identify chains of change in case of newly revealed information. In summary, this cluster analysis approach is more fruitful (R7.4). Therefore, this work dismisses the idea of cluster SRA instead of SRE and further evaluates the idea and needed adjustments for cluster SREs (see Section 7.2).

The existing extended questionnaire is subject to evaluation and tailoring in this work. The idea is to delete the initial questions for the SRE and incorporate missing information in the damage criteria and exposure section to enable efficient criticality evaluation (R7.4, R7.1). Also, the template gets easier to fill out by thinning out the predefined answers. Especially categories where information is missing in early development steps can be subject to recap. Including a particular answer for those uncertainties allows for tracing in the subsequent process (R7.2). Those deliberations result in a adjusted template for the SRE (see Section 7.3).
The template allows an automatized evaluation of the results (R7.4, R7.3). Evaluating uncertainties and distributing the results based on their need is possible. Modeling tools can serve as an interface to the SRE. The results are assimilable into the models, and uncertainties are resolvable by querying the model. Also, cluster-based results are dividable into groups of different criticality ratings (see Section 7.4).

7.2. Cluster SRE

The clustering of SREs reduces the effort for this process step by reducing the number of analyses (R7.4). For this, a clustering technique and changes in the questionnaire are necessary. While the clustering must be suitable to accomplish the improvement, the questionnaire needs to allow the notation of differences and different criticalities for a set of functions.

7.2.1. Criteria for the Clustering

For evaluating different clustering techniques, distinct evaluation criteria are necessary. Those are:

- Comparatively fewer SREs
- Low number of contact persons per cluster
- Similarity in the items
- Support of function-oriented development

The evaluation of the cluster techniques' efficiency uses the first criterion. Since the main objective of the cluster SRE is to reduce the effort for SREs, the goal is a comparatively low number of SREs while accomplishing the other criteria.
Completing the questionnaire is the development team's responsibility with the security department's support. Typically, one person is in contact for a few functions developed in the team. This person acquires the information from his teammates. Therefore, the more SRE this person needs to complete, the more often he needs to contact his teammates. Clustering this process step enables the contact person to acquire the necessary information in one step. The SRE may also necessitate a meeting between the contact person and the security engineer to clarify questions and get support for the process step. Clustering supports that fewer meetings are necessary and with fewer persons involved, required meetings are easier adjusted. The more contact persons are necessary for the cluster SRE, the longer the process takes.

The questionnaire needs an exchange between the different persons. This exchange is also error-prone and leads to different styles of commenting and fill-out. The expectancy is that a low number of contact persons raises the acceptance in the development teams. This expectation holds since fewer questionnaires need completion, so the development team has less deflection in daily work. Also, the SRE process is faster with few persons incorporated. Following from the described influences, this criterion is very important to acquire the demands of efficiency (R7.4).
Clustering raises efficiency only if the items have similarities. Also, there are possibly different variants of the same item. Examples are country-dependent variants or variants with different capabilities based on the configuration level. Those variants are targetable at once in a clustered SRE. The resulting SRE shows similar arguments and provides a straightforward overview.
The SRE is mostly done before serial development starts. At this development stage, typically, no integrated modeling tool is available. This point in the development process also means no deployment of software to hardware, as well as sensors and actuators, is wholly defined [95, p. 41]. Therefore, the development follows a function-oriented style. The clustering of the SRE needs to support this by the independence of the deployment in the architecture.

7.2.2. Possible Cluster Techniques

The automotive development process structure and the questionnaire enable different ways to cluster the SRE. Incorporating the experience from the first development cycle according to ISO/SAE 21434 leads to evaluating the different possibilities.

ECUs - The validation and verification steps in the development process incorporate the software deployment to the ECUs. An ECU-based clustering would lead to an easy possibility for tracking results through the complete development cycle. Also, it reveals inconsistencies in the security evaluation of items as early as possible.
Introducing COTS and high integration of software [81] leads to ECUs with a variety of software functionalities. Therefore, a cluster would incorporate many functions, leaving it unhandy. Also, those high integration ECUs are commonly subject to virtualization, which undermines an ECU-based clustering. This case would lead to sub-clusters based on the isolation unit, e.g., virtual machine/container. Modern cars contain around/up to 100 ECUs [29, 11]. This number does not contain sensors that do not count as ECU but may still need a SRE for their software. Sub-clusters for virtualization raise this number even more. Therefore, an ECU-based approach leads to over 100 SREs.
It is common to combine functions from different functional domains into ECUs, especially in the case of high integration. This combination leads to low similarities between the items and a high number of different contact persons.

Function-oriented development builds upon the independence of a concrete deployment. The allocation of functions to hardware finalizes at the start of serial development [95, p. 41]. ECU-based clustering undermines this independence and is impossible at this time in the development process. Also, in the case of function variants, it may be that functions are missing since they are not part of the initial deployment plan.

Automotive Domain - A common scheme to divide automotive functionalities is automotive domains. According to the literature, there are four domains: power train, chassis, vehicle body, and multimedia [16]. Due to the rising number of vehicle functionalities, OEMs tend to make subclasses of automotive domains, e.g., light and access control. These subclasses lead to around 20 domains.
The automotive domains are a coarse division of automotive functions. Due to that, clustering the SRE along the automotive domains supports function-oriented development. Nevertheless, dividing the software functionalities by 20 domains leads to a high number of functions in each SRE Cluster.
Development domains are independent of a concrete deployment and have certain similarities, e.g., light functions. While the overall functionality of the items has similarities from an abstract point of view, a more detailed look shows many differences. One difference lies in the development areas. There are different development teams inside the automotive domains, e.g., for electric and mechanics. Also, there are differences in the concept of the domain. In the case of light functions from an abstract point of view, they are similar, but there is a big difference between interior and exterior lights. Also, exterior light functions are different depending on the use case: driving or arriving/leaving concepts that present special light situations for the driver.
The differences in the functions inside one domain lead to many departments and contact persons for the cluster. Therefore, as a single criterion, the automotive domains are insufficient.

Team Clusters - Each OEM has a different type of clustering inside the company departments. Typically different departments again have different development teams. Those teams typically focus on a subset of an automotive domain, e.g., software-based light functions. Additionally, those teams take a particular viewpoint on this domain, e.g., standard light functions. Therefore, structuring the identified automotive domains into the development teams overcomes the drawbacks of pure automotive domain structuring.
Clustering along the development teams leads to functional groups of the items. Due to the teams' focus, the functions are similar by nature.
The teams incorporate all possible variants of the functions. Therefore, the cluster automatically includes function variants. This clustering also leads to the independence of a concrete deployment.

In the teams, there is typically a responsible person for security. Therefore, the development team clustering leads to the least possible amount of contact persons. Nevertheless, the clustering in development teams leads to a higher amount of SREs compared to the other solutions.

Feature Clusters - There are already precedential cases of cluster SRAs. Those clusters rest in certain areas where only exceptional cases are interesting. For example, only functions that allow feature-activation over the air are security-relevant in certain areas. This knowledge allows clustering also SRE.
This clustering technique directly leads to a low number of SRE since only functions with particular functionalities are part of the evaluation. Nevertheless, this is also a big drawback. Identifying the complete set of relevant features is error-prone. Other relevant features may be veiled.
Also, this clustering technique requires much prior knowledge about the function. At this development stage, no global model of the system architecture exists. Each function needs an evaluation of the set of interesting features to build up the clusters for the SRE. This additional step in the development process needs contact with the responsible persons in the development teams. By that, the acceptance of the security process lowers in the teams.
Another drawback is the number of contact persons in the cluster. A cluster with every function featuring feature-activation over the air incorporates many functions from different development areas. Therefore, there is a high number of contact persons, which is against the stated criteria.

7.2.3. Resulting Clustering

In conclusion, the presented clustering techniques lead to the evaluation in Table 7.2.
The ECU-based clustering is straightforward and leads to a low number of SRE. Nevertheless, in the case of high-integration ECUs it often undermines function-oriented development and shows low similarities between the functions. Also, it has a high rate of contact persons.
Better is the approach based on the automotive domains. It shows better results for the similarities and the number of contact persons. Also, it is independent of the deployment and leads to a low number of analyses.
The team-based approach has just the drawback of a higher amount of SRE compared to the abovementioned approaches.
A clustering based on security-relevant features is suitable in a function-oriented development and leads to a lower amount of SRE. Nevertheless, it is error-prone and undermines the similarity criteria and a low number of contact persons.
The most promising approach for clustering the SREs is to extend the clustering according to automotive domains to include the development teams. This approach has a significant advantage over pure filtering for the development team. Depending on the clustering results, different layers of the result are usable.

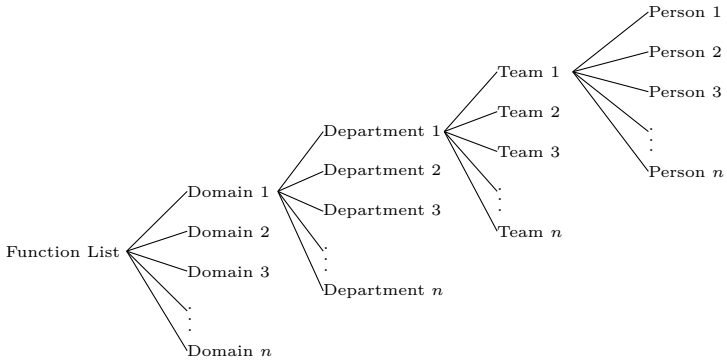

Figure 7.1.: Resulting clustering for the SRE. First automotive domain, then department, development team, and contact persons.

For example, in cases where the cluster for the department is relatively homogeneous, no further filter for the development team is necessary.
The information for such clustering is easily acquirable by analyzing item lists. Automotive development relies on lists that incorporate all possible functions and their variants. Such lists give rise to a unique id, name, automotive domain, development team, and typically, the contact person in this team. Clustering this list into automotive domains, the department, responsible development team, and persons also guarantee that the SRE step misses no items (see Figure 7.1).

		Criteria			
		Contact Persons	Similarities	Function Oriented	Low Number
Cluster Techniques	ECUs	-	-	-	+
	Domain	∼	∼	+	+
	Department	+	+	+	∼
	Feature	-	-	+	∼

Table 7.2.: Results of evaluating different clustering techniques. (+) for good results, (∼) for better than original, (-) bad results

7.3. Questionnaire

The evaluated process involves an extended version of the ISO/SAE 21434s question catalog and additional questions for determining the items' criticality (R7.1). The template is subject to NDA, but analysis revealed that the questions for relevance evaluation are a subset of the questions for determining the criticality.

Therefore, the relevance questions are deletable from the catalog (R7.4). The remaining section concentrates on the criticality questions and dismisses the basic relevance questions.
Evaluating the question catalog of ISO/SAE 21434 allows evaluating whether the existing template aligns with ISO/SAE 21434. For the notion of criticality, the evaluated process determines damage potentials and the exposure based on pre-defined answers. Evaluating and adjusting those answers allows for tracing uncertainties (R7.2). Subsequently, adjusting the template to incorporate the proposed changes completes this section.

7.3.1. Relevance Criteria

The security relevance criteria are derivable from the SRA scope. This scope is the possible damages arising from the item. Those target the protective goods. Like in security analysis, security relevance depends not only on what is protected but also on who is protected. Therefore, the assets, as well as the stakeholder, are subject to identification. In automotive security, stakeholders are the OEM and the user. The protective goods are derivable from security objectives and secondary attributes like privacy. The assets are the main categories of damage scenarios to the stakeholder and the security objectives.
Another question is, what information is necessary to identify an asset in danger? But also the information source as well as the time of availability. Not all information is available at the beginning of the development process. The more specific the questionnaire design is, the more precise the relevance evaluation. Conversely, more information might not be available, leading to more uncertainty in the result. This uncertainty demands later recaps of the evaluation and, therefore, raises the effort. Therefore, it is necessary to find a balance for a suitable level for the relevance evaluation and leave more depth information for the subsequent SRA. Focusing on the main damage scenario categories and formulating questions at a suitable level acquires this criterion.
Each question should target only one asset. On the one hand, mixing the questions' scope leads to imprecise answers. The contact person might have different answers for the scopes and is uncertain how to rate the question. On the other hand, this division allows keywords for each question. Those keywords are usable for tracing results.

7.3.2. ISO/SAE 21434 Question catalog analysis

Annex D of ISO/SAE 21434 [54] provides a flow chart (see Figure 7.2) with four questions targeting a primary line of damage criteria and attack surfaces: Safety, privacy, and interfaces.
The questions of ISO/SAE 21434 start with the origin of the function: "Is the candidate an E/E item or component?" [54]. This question narrows the scope of the evaluation to E/E items. The question's relevance depends on the development structure and the input data for the SRE. It is unnecessary if the company's structure strictly divides between E/E and non-EE functions.

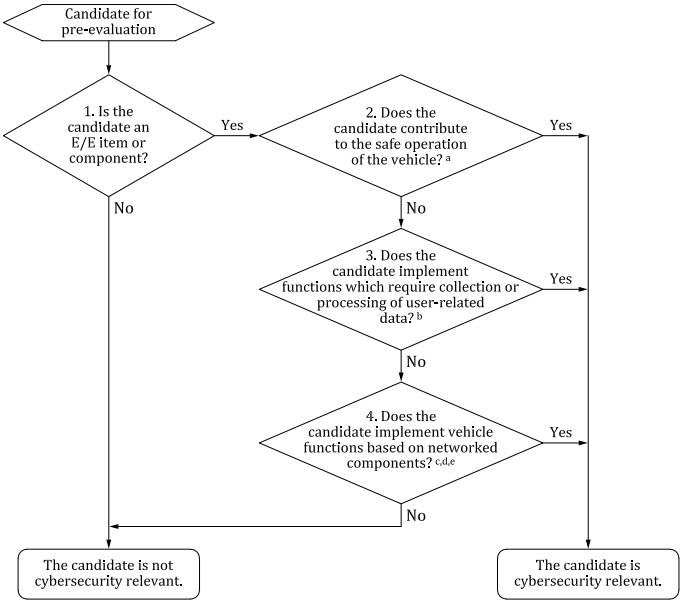

Figure 7.2.: Flow Chart for the SRE, taken from Annex D of [54].

Also, an annotated function list denoting whether a function is an E/E function makes this question useless.
Question two determines the damage potential of the item regarding safety. Safety and security interrelate and its cohabitation makes the safety-related evaluation of security relevance important. For example, safety degradation may negatively influence availability [115]. However, also security mechanisms affect safety mechanisms making a distinct consideration and evaluation necessary. Safety and security follow the same development process structure. Based on a relevance evaluation, the consecutive safety analysis takes place.
Nevertheless, the evaluation timing differs between safety and security. Therefore, it is better to incorporate this damage criterion in the questionnaire than to rely on getting the safety evaluation results from the safety department on time. The development team might have the results earlier.
The scope of question three is privacy which is crucial regarding the user information inside the automotive system. Especially since [32] privacy is a primary concern in each connected system. Information disclosure analysis depends on the data communicated and saved in the function. The details of such data reveal successively through the design process. Nevertheless, the general information, which type of data the function needs, should be known at the beginning of the development cycle. Otherwise, no algorithm design is possible. Therefore, privacy is an important asset, and the question is answerable, at least in general, at the time the SRE takes place. Nevertheless, the scope of the question is too narrow. The scope should include general confidentiality information like intellectual property of the OEM or car-related critical data, e.g., mileage. The former denotes critical information of the OEM. The latter prevents manipulation of the car, which also targets customer needs. Examples are car sale manipulations, e.g., turning the mileage indicator down to get a higher sales price.
The last question is regarding the system's attack surface, asking for the network capabilities of the item (internal and external interfaces and wirelessly connected sensors and actuators). This question targets the exposure to accomplish the damages defined in questions two, and three. While this question is essential to evaluate the items' vulnerability, the design of the question mixes all interfaces. Therefore, when the item has any interface, it is security-relevant independent of the interface type. This view is again too narrow. A comprehensive evaluation demands dividing this question into several questions or levels of answers.
In conclusion, the assets determining security relevance are relatively narrow. Especially the possible damage criteria in the financial and legal area, also OEM specific, should be more comprehensive. An extension provides a better understanding of the security impact of the item. Additionally, interpreting the attack surface question as exposure and an extension to several levels gives a comprehensive picture of the item. It allows the objective identification of an initial criticality.

7.3.3. Damage Criteria

[96, pp. 1 sq.] states that four damage criteria are relevant for future automotive systems. The financial area targets transactions like unauthorized feature activation and vehicle theft. Operational criteria affect the performance of the vehicle functions. Main damage scenarios involve interference with vehicle communications. Privacy and safety relate to the already mentioned factors: The driver's identity data and the vehicle's safe operation. Those descriptions follow the demands of ISO/SAE 21434 for the risk analysis. [96, pp. 1 sq.] adds to privacy, vehicle identification data, and system design. This approach goes along with the demand of this work to incorporate the OEMs intellectual property and car-related data.
Nevertheless, the discussed sources miss regulation-related criteria. Besides UNECE No. R155 and ISO/SAE 21434, other regulations demand specific security-related actions in the development process. Those are relevant for type approval, in general, or country-dependent. Omitting these criteria from the questionnaire may leave essential functions unattended in the security process.
The existing SRE template includes the following damage criteria:

Safety:	Rating of A-SIL Levels or appropriate
Finance:	Impact on production, product recall, financial loss to OEM or road user
Legal/Regulations:	Theft protection, type approval, privacy, manipulations
Quality decline:	Product quality

The criteria integrate the ISO/SAE 21434s question regarding privacy in the legal and regulations category. The other categories target the described automotive systems' most significant damage criteria.
In conclusion, the existing template is extensive enough to target the stakeholders' needs. The detailed question analysis is subject to the template section.

7.3.4. Exposure

One element in the question catalog of ISO/SAE 21434 is regarding the interfaces of the item. This question asks for external and internal network capabilities, wireless sensors, and actuators.
The items' communication capabilities are interpretable as exposure, like in the risk analysis. Therefore, splitting this question to target each attack surface is reasonable. This extension allows a comprehensive picture of the attack surfaces the item has.
There are items where an attack is only possible through the ECU or software. For them, a correlating option is necessary. The exposure question already uses a split between the different surfaces in the current process. This question has possible answers which are too deeply nested.

The proposed answers are as follows:

- External wireless interfaces
- External wired interfaces
- In-vehicle interfaces
- No ECU-external interfaces

7.3.5. Criticality

The evaluated version of the template provides pre-defined answers. Each answer leads to a damage potential level from one to five (very low to very high), extending the suggestion of ISO/SAE 21434. This work suggests using levels one to four (very low to high) like in ISO/SAE 21434 see Table 7.1. An extension to five levels has no additional value in the SRE.
The template provides several answers leading to the same damage potential. Also, sometimes the answers are very detailed. Depending on the possible level of information detail before serial development, it may be necessary to formulate fewer thresholds. It is easier for the responsible persons to find a suitable answer. Otherwise, it may be the case that the question is left open or answered incorrectly due to the lack of information.
Reducing the provided answers lowers the level of detail in the resulting initial criticality. Nevertheless, the initial criticality is still rated objectively over the evaluated items. Security planning uses criticality to determine the next steps. Minor detail in the evaluation in this step is better than missing damage scenarios.
Also, there is the possibility that a damage scenario is impossible for this item. Therefore, a negative answer is necessary. This answer level is zero to exclude it from the criticality rating.
Additionally to the depicted levels, another possible answer for each damage scenario is *unknown*. Unknown explicitly states the uncertainty or lack of information. The resulting criticality is, for the moment, not thoroughly evaluated but still allows to proceed in the process without delay. Such uncertainty is typical for early development stages. Including it explicitly makes the analysis more comprehensive and honest. Tracing this uncertainty through the development process allows follow-up on those items and a later recap of necessary changes.
An example question in the finance category is as follows:

Is a product recall necessary in case of an attack?

4 Yes, a global product recall is possibly needed

2 Yes, a product recall is possibly needed

0 No recall necessary

-1 I cannot answer this question

For the exposure, levels one to four also apply. Since an item without ECU-external interfaces is attackable through the ECU, the lowest level is one. An item can have several interfaces. If so, the responsible person marks all suitable answers. The criticality determination uses the highest possible exposure level. These deliberations lead to the following rating of the exposure:

4 External wireless interfaces

3 External wired interfaces

2 In-vehicle interfaces

1 No ECU-external interfaces

In order to use the described ratings, the criticality determination follows from the damage scenarios and the attack surface. The evaluation combines the highest possible damage potential over all damage scenarios and the highest exposure level (see Equation 7.1. In the case where all damage potentials rating is zero (no damage potential), the item evaluates to not relevant for security. This evaluation aligns with ISO/SAE 21434, where items are irrelevant or relevant. The additional criticality levels allow a detailed evaluation of the risk assessment report.

$$Criticality = max(Damage\ Potential) \times max(Exposure) \tag{7.1}$$

ISO/SAE 21434 allows three levels for risk recommendation in the assessment report: accepting, conditionally accepting, or rejecting the risk [54]. Transferring this to SRE leads to different ways to proceed in consecutive security development process. In cases where all damage scenarios answer is no, the item is subject to risk rejection. A level below a defined threshold leads to risk acceptance. Cases with conditional risk acceptance are subject to a subsequent SRA.

7.3.6. Template

The described contents incorporate a template for accomplishing the SRE. This work adjusts the existing spreadsheet-based template for the SRE to include the cluster idea, the adjusted damage scenario levels, and the adjusted exposure answers. This template has three sections: the general information, the damage scenarios, and the exposure. Each input field must incorporate single items or item lists for the cluster approach. Therefore, the following inputs are valid:

- single item id, e.g., 42
- list of items, e.g., $\{21, 42\}$
- none
- all

General information comprise the information needed to identify the SRE:

- Department / Team incorporated in the SRE
- Name of the cluster/function
- Ids of the function(s)

- Platform or vehicle project
- Which other function(s) use / are used by this function(s)
- If known, the deployment of the function
- If this function(s) was/were subject to a SRA

Damage scenarios incorporate the different questions out of the four damage criteria sections. Each question has up to four qualified answers. Additionally, each question has one answer for dismissing this question as inappropriate and not available information (unknown) (R7.2).
The damage scenarios include the safety relevance of the item, covering the user and the environment as protective goods. Others are questions regarding the users' privacy and the OEMs intellectual property. Those questions demand confidentiality, as well as integrity of information.
Other questions target the integrity of the vehicle's functionality. Those evaluate possibilities for tuning and activating functions and relate to the OEMs income of purchases.
Prevention of theft and therefore protecting the system and user is the goal of questions regarding physical access to the vehicle.

Exposure describes the four possible types of interfaces. This part of the template relates to which attack surfaces the item has. By that, the exposure of the damage scenarios to the attacker is evaluable.

7.4. SRE Analysis

The improved SRE enables different analysis steps. A distinction between single and cluster SRE is necessary. Single SRE are directly analyzable while cluster SRE allow for possibilities. Also, the follow-up analysis differs in the development timeline - before and after the start of serial production.

7.4.1. Tool Support

In general, tracking the SRE procedure, e.g., via a ticket system, is necessary. This ticket system should allow to include the clusters and link to the single items. By that, it is easy to recap the relation of the SRE process. This structure reveals uncertainties that need clarification before the SRA step.
The SRE results are subject to versioning systems. This system is necessary to fulfill the demand of ISO/SAE 21434 and UNECE No. R155 to track changes. It needs to be impossible to change any results without the possibility of reconstructing the original result and recap the changes.

7.4.2. Criticality of Single SRE

Single SREs are analyzable like before the presented changes. The damage scenario with the highest damage potential and the highest exposure level determines the criticality of the item. Nevertheless, answers which depict uncertainty (unknown) are subject to follow-up.

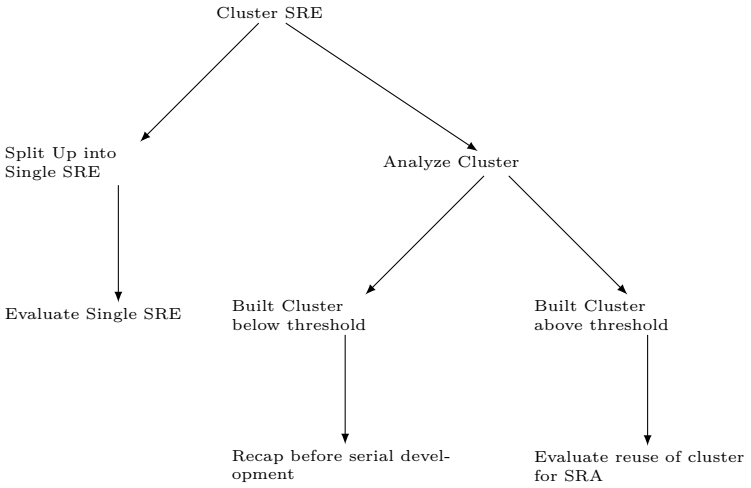

Figure 7.3.: Different possibilities for analyzing cluster SRE

This recap is necessary if the criticality of the item is less than the threshold for subsequent analysis. In this case, the item must be marked in the ticket system and evaluated later.

7.4.3. Criticality of Cluster SRE

Cluster SREs allow different types of analysis (see Figure 7.3). First, the single items are evaluable by splitting up the cluster. The resulting SREs follows the described process for single SRE.
The cluster itself is usable for different evaluations. The most practical possibility is to evaluate the different criticality levels to build new clusters. One cluster depicts items whether criticality is lower than the threshold and which drop out of the subsequent process. The other items build the cluster, which might be reusable for the SRA. For that reuse, the security engineer evaluates similarities and differences in the SRE results and decides whether this cluster is suitable for a cluster SRA in Chapter 8.

7.4.4. Automatic Analysis

The questionnaires spreadsheet structure allows for automatic analysis (R7.4) and tracking (R7.3) of the results. For this analysis, a script can parse the template and build the data structure of the results. This analysis script is adjustable to different tools that use the results as input format.
Such a script is adjustable to several other use cases. One idea is to design the output in the format suitable to stream it into the ticket system used for tracking the SRE system. Also, it is possible to insert the results into the versioning system directly. Another idea is to fill ToDo Lists with the item IDs having uncertainties. Those ideas are directly realizable by adjusting the output formats of the analysis script to the formats needed for the other tools.
In the end, the script is linkable to a script that parses the integrated architectural model for changes before the start of serial development. This additional analysis is especially relevant for those items which drop out of the security development process. Changes in those items might lead to re-include them in the process.

7.4.5. Trace Uncertainties

The presented SRE structure explicitly states uncertainties by the answer possibility of "unknown" (R7.2). The analysis marks those answers for a later recap on this item. This recap is necessary whenever the item drops out of the subsequent process. If the criticality of the item is above the defined threshold, the item is subject to a SRA, which makes a recap on the SRE unnecessary.

Before the start of serial development there is no holistic architectural model. Therefore, at his time, the recap of SRE is impossible to automatize. Depending on the timeline of the security process, it may be necessary to do recaps manually. This approach is helpful in cases where a cluster SRA could incorporate an item that is uncertain to be security-relevant. Combining the recap with coordinating dates and needed information for the cluster SRA is practical in those cases.

After the start of serial development the tracing of uncertainties is automatable by evaluating architectural modeling tools. For that, the model serves as a database. The different questionnaire contents serve as variables for querying the model.
An integrated model should also include safety. Therefore, the question regarding an Automotive Safety Integrity Level (ASIL) classification is easy to verify. More complex is the questions regarding sensitive data, personal data, and influence on the vehicle. These questions demand an appropriate tagging of the model.
The recommendation is to design a catalog with essential data categories and reuse it with every design cycle for tagging the model. This catalog contains data known to be related to the damage categories. Data where the developer is unsure whether it is relevant is subject to a recap with the security department. In the case of a security-relevant function, the SRA does this recap automatically. The developer directly contacts the security department to discuss the issue in case of irrelevant functions.

Unauthorized activation is only relevant for tagging in the model if the function is an add-on. All mandatory functions for the vehicle that are not configurable during the purchase are irrelevant for unauthorized activation.
Typically, functions that allow vehicle access are subject to the same development department. Therefore, it is easy to identify such functions at first glance during the SRE. Nevertheless, an open window also allows physical access, which is not subject to the same development department. Such information is therefore difficult to identify. Automation is intricate since there are too many possibilities based on the function. In case of uncertain answers to this question, the security engineers have to contact the responsible developers to clarify this.
The exposure is easy to automatize using an integrated model. The model incorporates the interfaces by nature. Therefore, uncertainties in this part of the SRE are resolvable automatically.

There is nothing more practical than a good theory.

Kurt Lewin

8 Function-oriented Security Risk Analysis

The SRA is the next analysis step after the SRE. The function-oriented SRA uses the SRE results - the relevant functions - and determine their security risks. In function-oriented development, the SRA scope is at first the function's intended behavior and possible negative influences through incorrect data.
In general, security risk analysis (sometimes called risk assessment) aims to forecast actions an attacker might take to introduce external faults into the system and launch an attack. Therefore, risk analysis identifies asset damage scenarios and derives possible attacks to realize these scenarios. The combination of the damage scenario impacts and the attack feasibilities lead to the risks to the system.
Some sources include risk treatment in risk analysis (e.g., [17]) and also in practice, both development steps sometimes intertwine. Also, for some defense methods like network segmentation, it is necessary to introduce them in earlier development steps. The SRA needs to consider their influence on the security risk. However, the overall risk treatment should always take place at least on a hardware-global basis, after the system SRA. Chapter 10 further elaborates on this topic.

8.1. Process Description

UNECE No. R155 relates risk analysis to vehicle types and refers to ISO/SAE 21434 for further information. Section 8.1.1 illustrates the risk analysis steps of ISO/SAE 21434, followed by a short introduction of the method used in the evaluated development process (Section 8.1.2). In the literature, several different approaches to risk analysis exist. While the approach's idea is always the same, the procedure differs depending on the subject of analysis. Related work (Section 8.1.3) depicts other approaches for risk analysis in the automotive sector. The subsequent Section 8.1.5 describes improvement potentials and their solution approach.

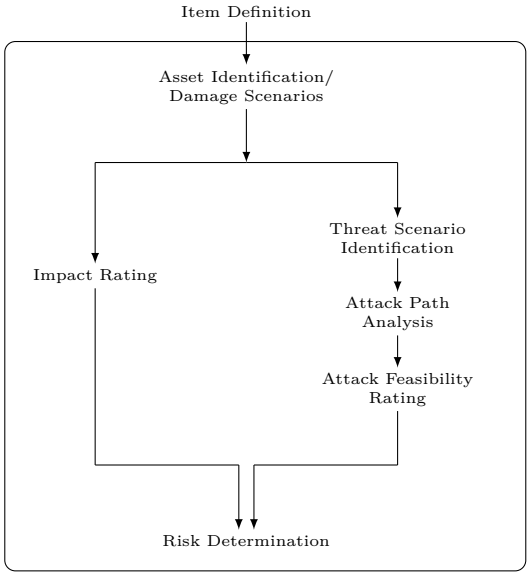

Figure 8.1.: Structure of the SRA according to ISO/SAE 21434. The item definition is a preliminary activity in the SRA not belonging to the core activities.

8.1.1. Normative Background

ISO/SAE 21434 [54] states the general risk analysis structure in Clause 9. The detailed description of the procedural steps is subject to Clause 15. Figure 8.1 provides an overview of the different activities in the SRA.
The preliminary step in risk analysis is the *item definition*. This work product is a model of the evaluation target consisting of the boundary, the functions, and assumptions about the environment. Those assumptions are expectations about the environment's behavior or specific facts, e.g., a certain level of encryption [38, p. 23]. This step is crucial in the analysis since it defines the system boundary of the current analysis [17] which is the point of fault introduction[65].
Asset identification includes the formulation of damage scenarios as a description of adverse consequences to an item element with potential harm and an asset, e.g., "disclosure of the personal information without the customer's consent resulting from the loss of confidentiality" [54]. Those assets and assigned damage scenarios give rise to the targeted security objectives [27].

Impact Rating assesses each damage scenario regarding its "magnitude of damage or physical harm from a damage scenario" [54] to the road user. The consecutive rating uses safety, financial, operational, and privacy categories, each rated as negligible, moderate, major, or severe.
Threat Scenarios define how the damage scenarios are realizable. Each asset is subject to evaluation regarding the compromised security objective and the possible causes (vulnerable states) of this compromise. This description may also include technical limitations or attack surfaces.
The *Attack Path Analysis* further evaluates the identified threat scenarios. It aims to annotate an attacker's steps to realize the threat scenarios. Therefore, bottom-up and top-down approaches are usable. The former derives the attack path from the threat scenarios, e.g., by attack trees. The latter builds the path from the identified vulnerable state.
To complete the attack evaluation, *attack feasibility rating* analyzes the likelihood of the attack paths. Each attack path gets a rating from very low to high based on either an attack potential-based, a Common Vulnerability Scoring System (CVSS)-based, or an attack vector-based approach. All three approaches target the attacker in behalves of his necessary capabilities and possibilities to attack the system. The Annex of ISO/SAE 21434 [54] introduces the different approaches so that the interested reader might use it as a reference.
The *Risk value determination* combines the results from the impact rating of the damage scenarios and the attack feasibility rating. Each threat scenario gets a rating from 1 to 5 (very low to very high) based on risk matrices or formulas.
UNECE No. R155 demands the identification of risks but leaves the execution specifics to the accomplishing company. Nevertheless, the UNECE No. R155 provides a threat catalog in the Annex, which must be part of the security risk analysis.

Requirements - The evaluation of the normative references, the SRE improvements, the risk treatment requirements, and the overall aim of the thesis lead to the following requirements:

- **R8.1** Function-oriented SRAs must be realizable in a time efficient way.
- **R8.2** The function-oriented SRA damage criteria catalog and the SRE questionnaire must be consistent.
- **R8.3** The function-oriented SRA implementation must maintain the differentiation between risks to the OEM and the road user in accordance with ISO/SAE 21434
- **R8.4** The used method must internally support tracing, and the embedding in the overall security development process must use the tracing capabilities.

8.1.2. Implementation in the Process

The evaluated implementation of the security development process uses the Modular Risk Assessment (MoRA) method (see Figure 8.2) [10, 9, 8, 30]. It relies on incorporating multiple stakeholders into the analysis procedure. Therefore, the method provides a modular and iterative approach, accompanied by several catalogs, e.g.,

for controls and assumptions. Those are adjustable, for example, by deriving them from the security policy at the beginning of the development cycle [30, 9, 10].
In the *item definition* step, MoRA models the item in four steps. At first, the functions are subject to identification and decomposition into components and interfaces. A mapping from the components to the interfaces enables the identification of relations between those. Assumptions limit the scope of the analysis, e.g., by assuming a secure backend. Functions produce or consume data while components are the units storing data or hosting functions [58]. The result is a model which depicts, starting from the functions, the relations to data, components, and data flows.
The next step determines the *assets* by assigning security objectives to the item definitions elements and rating the *impact* using the damage criteria catalog (R8.2). All combinations which result in a non-zero impact are assets [9]. Combining the assets with identical consequences results in the damage scenarios. Their impact results from the security objective with the highest impact [38, p. 60].
In contrast to ISO/SAE 21434, MoRA allows for assessing the impact on the road user and the OEM. As a result, the highest impact value sustains through the MoRA. This method is in line with the opening clause of ISO/SAE 21434 and the suggested extension of Chapter 7 for the SRE.
In the next step, the *threat analysis* takes place. This step identifies the trust boundaries of the item definition and maps potential threats to the security objectives. The threat analysis uses a previously defined threat catalog following the STRIDE approach [30]. Each threat in the catalog has an attack surface, the target layer, and the technology assigned, allowing easy identification of relevant threats for the item. Together with additional assumptions and preparational attacks, this enables the modeling of different *attack paths* for each threat. Thereby, assumptions leverage damage scenarios' potential by transforming or deleting them. Preparational attacks show different ways to start an attack [58].
The *feasibility* determination uses a predefined attacker model for each threat. The attacker model in MoRA follows a capability-centric approach [38, p. 22]. The maximum from each capability category depicts the feasibility of the attack path. The sum of the categories and a defined threshold table defines the required attack potential for the attack path.
MoRA defines a risk for each attack path that links to the attack feasibility. The attack paths connect to the functions, components, and interfaces and, through them, to the damage scenarios [30]. The risk estimation represents a combination of the damage scenarios' impact and the attack potential. A risk matrix, like Table 7.1 allows for identifying the risk. This risk number is a numerical ranking of the potential attack and the damage impact, like in failure mode and effects analysis (FMEA) (see [23, p. 44]).
In the end, the MoRA approach has the risk assessment as the resulting work product. This documents the identified damage scenarios and impacts, threats, attack paths, feasibility, and determined risks. Therefore, it allows to follow the references through the model and enables tracing inside the MoRA process (R8.4).

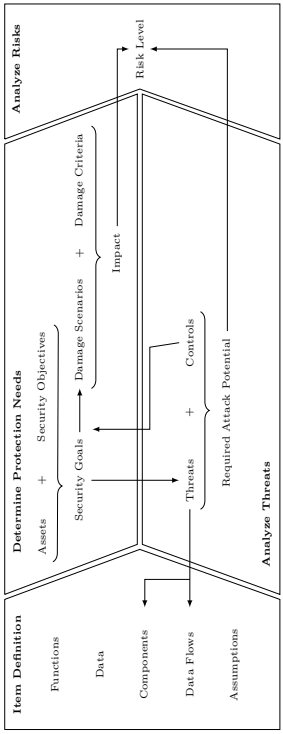

Figure 8.2.: Overview about the MoRA steps redrawn from [7].

Threat Class	Description	Security Goal
Spoofing	Attacker identifies as another one	Authentication
Tampering	Intentional modification	Integrity
Repudiation	Manipulation in the name of others	Non-repudiation
Information disclosure	Sharing of confidential material	Confidentiality
Denial of service	Disruption of Service	Availability
Elevation of privilege	Exploiting a design flaw to gain elevated access	Authorization

Table 8.1.: Threat classification according to the Microsoft STRIDE model [41]

Another result from the MoRA is the list of open issues, if applicable, which depicts necessary re-caps to complete the analysis.

8.1.3. Related Work

As [91, p. 508] states, it is beneficial to have risk analysis processes as a blueprint but it is necessary to adapt them to the system under consideration. ISO/SAE 21434 suggests four approaches to derive the threat scenarios and attack paths. Those are EVITA [96], Process for Attack Simulation and Threat Analysis (PASTA) [83], Threat, Vulnerability and Risk Assessment (TVRA) [31], and STRIDE [41].
TVRA [31] targets telecommunication and PASTA [83] focuses on the software development life-cycle and web applications. Both projects may serve as information sources for an automotive risk analysis method, but it is impossible to directly transfer them to the specialties of the automotive domain.

Microsofts' STRIDE model provides a threat model which classifies threats into six categories [41]. Thereby it is no integrated approach for the complete risk analysis process but supports the threat scenario definition. Also, it is useful to incorporate it in other approaches, like in the currently used MoRA approach.
STRIDE provides six threat classes, each combined with the respective security attribute. This categorization allows to focus the subsequent risk analysis steps on specific security attributes or identify the ratio of the security attributes. This combination leads to a classification of threats to the system like in Table 8.1.
While [41] suggests using the STRIDE model on the decomposed system, it is also possible to use the provided categories for a general threat model instantiated in the risk assessment step. Such a model incorporates known threats and attacks from former developments and threat databases as suggested by [38]. Thereby it is made sure that in a distributed and heterogeneous system like in the automotive industry, the risk analysis of the different system parts uses the same assumptions.

The EVITA project focuses on vehicle intrusion detection. Besides the idea of hardware security through a Hardware Security Module (HSM) it defines a complete risk analysis process. The threat scenario and attack path identification follow an attacker-centric approach by deriving threats from the attacker's motivation. This motivation spread from harming the user and passengers to mass terrorism [96].

The EVITA project uses attack trees to model the attack paths. Attack trees, like fault trees, are top-down modeling methods. Starting from the attack goal, the attack tree incorporates the attack objectives, methods, and attack steps in its layers. The attack path modeling uses boolean algebra to depict the relation of the attack steps [96, 103, 80].
The attack trees' attack objectives are the damage scenarios, rated against the categories from ISO/SAE 21434: Safety, financial, operational, and privacy. EVITA assigns an impact vector to each attack objective (damage scenario).
For the attack feasibility rating, EVITA uses, like MoRA and ISO/SAE 21434 a threshold-based approach. The categories of the attack model have four levels, like in ISO/SAE 21434 assigned to the attack steps in the tree. The combined attack potential for an attack path is calculated using a min/max approach for the boolean relationships of the attack steps and assigned to the attack method layer in the attack tree [96, pp. 86 sqq.].
The risk determination of EVITA differs from MoRA due to the vector representation of the impact. EVITA designates a risk value for each impact category. For safety-relevant risks, a controllability value is utilizable to leverage the risk based on ISO 26262:2018 evaluation results [96, pp. 90 sqq.].
Using severity and risk as a vector depicting values for each impact category has advantages. This view allows to rank the risks according to impact category priorities, e.g., prioritize safety risks. By that, it is easily possible to make trade-offs in the subsequent risk treatment procedure. This method has no overhead since the impact analysis evaluates all categories separately. Also, the attack trees support a distinct and traceable modeling and analysis of the threat scenarios. MoRA uses the item definition and threat models to derive the threat scenarios and attack paths. From that, it is impossible to identify the relationships of the attack path at one glance.
On the other hand, it is difficult to include the necessary documentation, e.g., comments, into the attack tree without a suitable tool. Those tools are typically subject to changes, and backward compatibility is difficult to maintain. This issue undermines the UNECE No. R155s demand to be able to reproduce work products at any time, also after finishing the development cycle.
Another drawback is the direct incorporation of hardware into the analysis. One of the constraints in this work is the function-oriented development methodology before serial development. Therefore, EVITA is not usable without major changes.

The HEAVENS project targets vulnerabilities in automotive systems and provides methods and tools to evaluate the security and the interrelationship of safety and security. HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) risk analysis method also starts by modeling the item. In this case, the chosen modeling technique is Data Flow Diagrams (DFD) consisting of the trust boundary, the functions, external entities, data storage, and data flow [74].

The DFD serves as input for deriving the threats using STRIDE and deriving damage scenarios from them. Like EVITA, HEAVENS uses attack trees for modeling attack paths.
In contrast to MoRA and EVITA, HEAVENS uses a logarithmic scale to rate the impact and emphasize the financial and safety category by decreasing the other factors by one magnitude. The overall impact derives from rating and normalizing the impact categories.
Attack feasibility rating in HEAVENS uses an attack potential-based approach and normalizes the ratings from the categories (expertise, knowledge, equipment, window of opportunity). Each category has four levels with values in a linear scale [51].
Using the impact and the feasibility rating, the risk value is determinable according to a risk table [74]. This step follows the approach of MoRA where the impact and the feasibility is a scalar used to derive the risk value for the attack path.
The HEAVENS model got an update to align with ISO/SAE 21434. With this update, the approach is a combination of MoRA and EVITA. The threat scenario derivation uses attack trees like in EVITA while the risk determination follows the straight-forward approach from MoRA.
The HEAVENS approach to introduce weights for the impact and the feasibility rating is interesting since it provides the possibility for several trade-offs and prioritization already in the SRA. Nevertheless, the weights should be defined at the beginning of the development process to keep them static over all SRAs. Otherwise, the security engineers can introduce different adjustments for similar scenarios and leverage the comparability between the analysis results.
While HEAVENS is an interesting approach for incorporating safety and security, it is not directly usable in the evaluated process design. In a function-oriented setting it is impossible to define the complete data flow in the function level of the development process. Transferring the risk analysis step into a later development time, would contradict to the idea of security-by-design. Therefore, HEAVENS is suitable for integrated development processes and in parts it is reusable in the system SRA method.

8.1.4. Problem Description

The evaluated process implementation uses two types of MoRA implementations: A spread-sheet-based template and the Yakindu security analyst[1]. The Yakindu security analysis is a proprietary tool that implements MoRA besides other methods. Both implementations have their advantages and disadvantages. The Yakindu security analyst uses a versioning system to track changes in the model, which is a significant advantage over a spread-sheet-based approach. On the other hand, proprietary tools are subject to license fees. Also, software-based SRAs have the problem that updates may destroy backward compatibility, making it difficult to re-open existing models. This problem prevents the reproducibility of results and

[1] https://www.itemis.com/de/yakindu/security-analyst/

the update of existing analysis, which must be possible until the end of the vehicle type's life cycle (R8.4). A template keeps its structure; backward compatibility is excellent in most spreadsheet programs. Therefore, a spreadsheet-based template might be preferable. It needs no solutions to keep backward compatibility or old implementations running. Especially, a mixture of both worlds is inconvenient since it raises the need to maintain the analysis foundations. The decision for one implementation and version should be part of the planning step and kept till the end of one development cycle. Nevertheless, those decisions lie with the security department and are out of the scope of this thesis. This thesis focuses on the spreadsheet-based implementation of MoRA since it is possible to modify and adjust this.
The SRA needs many preparations for the input data and from experience, at least three meetings between the security engineering department, the responsible person for the function, and the security analysts accomplishing the SRA. Therefore, efficiency (R8.1) is one issue with the SRA step.
Another point in the SRA step is the damage criteria scope compared to the SRE (R8.2). Both process steps evaluate an item's damage criteria and rate them according to certain thresholds. In the SRE steps, this is an explicit step through the question catalog and the pre-defined answers. The SRA incorporates this through the impact rating, which uses pre-defined impact categories and levels. Both process steps should match the scope of the damage criteria.
Using additional damage criteria in the MoRA has two main drawbacks. The contact person from the development team needs to know the differences. Otherwise, he might not prepare all the necessary information for the SRA (R8.1). Also, the other actors in the analysis need to be aware of those differences to prevent missing ratings.
On the other hand, less damage criteria in the SRE is adverse. If the criteria are essential, then the SRE should also target them. Otherwise, it might lead to functions rated irrelevant that are relevant according to the missing criteria.
Also, the security development process is subject to continuous audits. The auditors evaluate whether the process aligns with the UNECE No. R155 and whether the tools and methods are suitable to cover the demands. The auditors analyze different use cases according to completeness, traceability, and reproducibility. Different catalogs in the SRE and the SRA give rise to questions regarding completeness of those two analysis parts.
Therefore, for completeness reasons, the catalog should be consistent regarding their topics and, if possible, the levels (R8.2). Differences should be assessed and documented during the security planning step (R8.4). The differences through composing levels in the SRE are unproblematic since the evaluation targets the same criteria and levels but merges some for a more accessible overview. By that, the function-oriented SRA is a expansion and refinement of the previous SRE.
ISO/SAE 21434 demands to evaluate risks against the road user. While it is relevant for the OEM whether the function-oriented SRA also reveals risks against his interests, it is not necessary to align with ISO/SAE 21434. Therefore, to pro-

vide both information separately, it is reasonable to differentiate between those risk types (R8.3).
Tracing (R8.4) targets two directions in the case of the function-oriented SRA. Decisions need to be traceable through the development cycle. This case is already integral to the MoRA method. MoRA accomplishes the complete SRA step in one integrated tool and links the steps to the underlying item definition and between themselves. Comments provide the possibility to annotate further information. By that, it is possible to trace all decisions taken. The other direction of tracing is regarding the underlying catalogs. Those might be subject to changes during the development or life cycle. Such a change makes it necessary to re-evaluate the SRAs. This re-opening has to be implementable in a time-efficient way by tracing the necessary information throughout the complete life-cycle of the vehicle type.
The MoRA spreadsheet incorporates the explicit formulation of uncertainties by including a particular part for open issues. Nevertheless, those issues are part of a necessary recap subject to tracing.

8.1.5. Approach

Like in the SRE also the items of the function-oriented SRA show many similarities. Therefore, also for the SRA, clustering the items is a way to raise time efficiency in the security and development departments (R8.1). Clustering the SRAs reduces costs in this process steps, since the effort for the cluster SRA is less compared to single SRAs. The reduction results from fewer meetings necessary and less effort to introduce the items. Clustering also reduces inconsistencies between SRA results since all items in the cluster follow the same deliberations, e.g., about assumptions. On the other hand, single SRAs might be more comprehensible and complete. Clustering may lead to missing assumptions, damage scenarios, or threats. MoRA provides several catalogs which support the consistency between different SRAs and completeness. Together with the analysts' experience, this overcomes the issue. Therefore, Section 8.2 elaborates about strategies for clustering and needed changes in the MoRA template. The evaluation of necessary changes to the MoRA method is part of Section 8.4.
The biggest issue in risk analysis is tracing. This issue incorporates the consistency of damage criteria between the SRE and the SRA (R8.2), open issues remaining after completing a SRA, and the used catalogs of the MoRA method (R8.4). Section 8.3 discusses the different approaches and necessities to enable tracing.
One necessary adjustment in the MoRA template is due to requirement R8.3. ISO/SAE 21434 demands to evaluate the risk against the road user but not against the OEM. MoRA uses the highest impact for the risk determination. Therefore, it is unclear whether the resulting risks are crucial according to ISO/SAE 21434. This also influences the efficiency of subsequent process steps. Therefore, it is necessary to designate the risks according to the road user and the OEM separately. Section 8.3 targets this adjustment and others arising from the preceding sections.

8.2. Cluster SRA

Clustering the SRA process follows the same idea like with the SRE (see Section 7.2). Therefore, it has identical evaluation criteria for the clustering technique.
The straightforward approach for clustering the SRAs is to use the clusters with the relevant functions of the SRE (R8.1). Those show a low number of contact persons, which is essential since in the SRA process are typically at least 2-3 meetings necessary. Participants of those meetings are representatives from the security department, the development team contact person, and the analysts. Therefore, they have many participants from different departments and companies (in case of outsourcing), making it challenging to adjust meetings.
Also, the functions in the criticality cluster show high similarity. This criterion is necessary to prevent unnecessary complex results in the SRA. Clustering SRAs leads to more damage scenarios, threats, and risks. Similarities in the functions reduce this number since at least threats and damage scenarios should intersect.
Additional criteria for clustering SRA might be to limit the cluster to functions of the same component in the deployment. This limit reduces the number of functions in the cluster. However, it can also lead to a high number of single SRAs. High integration ECUs tend to manage many different functions, which would all be the target of a cluster SRA. Other ECUs typically have only one function deployed. In both cases, the ECU has a close bond to other related ECUs, e.g., from the light domain. In this case, it is unnecessary to split the SRA because the item definition includes the function's environment and, by that, would include the other functions of the cluster. It is not easy to find an all-purpose answer. Therefore, this criterion is left to the security engineer to decide if a criticality cluster is not subject to splitting.
The clustering scheme (see Figure 8.3) is a categorization the security engineer can use to validate the clusters before the SRA takes place. Also, the security engineer must verify that there are no functions below the criticality threshold for risk acceptance that have open uncertainties (R8.4). This information is visible in the SRE results and, if used, in the ticket system. In such cases, the security engineer must contact the responsible person of the development team and complete the questionnaire. Otherwise, it might be possible that relevant functions drop out of the development process.

8.3. Traceability of SRA

The MoRA method inherently supports tracing [30]. Therefore, analysis further tracing possibilities inside MoRA is out of scope of this work.
However, the MoRA results and the relation to the SRE and risk treatment are subject to evaluation. This evaluation incorporates the damage criteria catalog of both process steps, the open issues and assumptions of the SRA results, the MoRA catalogs, and the issues which trigger a reopening of completed SRAs (R8.4, R8.2).

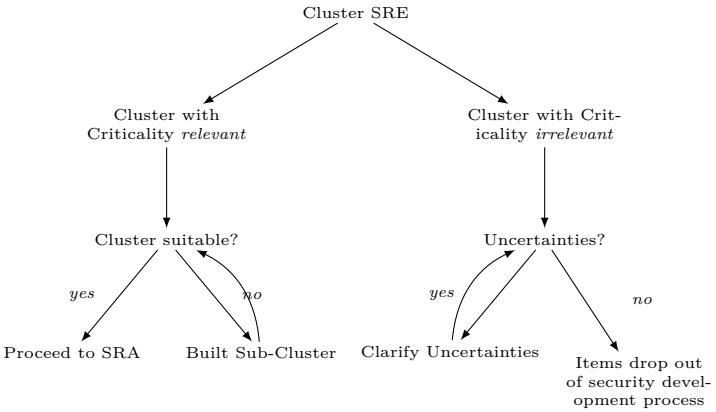

Figure 8.3.: Process from the cluster SRE to the cluster SRA depicting the clarifying of uncertainties and re-clustering.

8.3.1. Damage Criteria Alignment

The impact rating of damage scenarios in MoRA uses a damage criteria catalog. This catalog provides impact levels to the road user and the OEM in the same categories and with the same issues as the SRE questionnaire.
Comparing the unimproved SRE template and the MoRA template reveals some inconsistencies. While the criteria catalog in MoRA incorporates all categories and levels like in the basic SRE version, the catalog merges the levels in the categories. Since MoRA allows only one impact level per category, only the highest impact per category is noticeable. This notation diminishes other damage criteria (R8.3).
Also, sometimes the naming differs substantially, which is especially problematic in the safety category. The automotive development process commonly uses the naming of the severity levels used for determining the ASIL according to ISO 26262:2018 [53, Part III]. Also the SRE template uses this levels but the MoRA template not (R8.4). In the template, the safety levels are differentiated not in S0 to S3 but verbally and different to ISO 26262:2018, which is no mistake but an inconsistency with the common naming scheme. For example: In the SRE template, the lowest level is "(S0) No injuries" and in the MoRA it is "Potentially dangerous situation". For supporting the consistency (R8.2) between the damage criteria in SRE and SRA a comparison overview is usable in the security development process planning. The idea is to update the overview in case of tool changes and maintain consistency.

In contrast to the SRE, it is not necessary to have levels for uncertainty or irrelevant damage criteria in SRA. Uncertainties in the analysis lead to open issues separately declared in the analysis results. Since analysts accomplish the SRA, it is not necessary to explicitly target irrelevant damage criteria level. It is assumable that the analysts either completely target all damage criteria or declare open issues.

8.3.2. Open Issues

It is necessary to track the progress of the security development process (R8.4), which incorporates the tracking of open issues.
As proposed in the SRE analysis section (Section 7.4), it is reasonable to use a kind of ticket system to maintain the progress of the security development process. In this case, this system should be able to mark that SRAs which has open issues or even assign them as tickets.

8.3.3. MoRA Catalog Changes

The MoRA method provides several catalogs for the analysis: damage criteria, threat classes, attacker model, and assumptions. At the beginning of the development project, security planning defines the catalog contents. In order to maintain consistent analysis, the catalogs should be static for the complete development cycle. Nevertheless, the demand for static catalogs is unsustainable due to the constant change in the knowledge base. Regularly, new threats reveal, and with them also new attack paths. With rising computing power, assumptions about the strength of cryptographic algorithms break. Those may make changes in the MoRA catalogs necessary (R8.4).

Threat catalog - Newly revealed threats during the development process can impact already accomplished or open SRAs. Also, the threat catalog must be updated after the start of production and during the vehicle architecture's lifetime. Therefore, the security engineers include new threats according to the threat class in the catalog and assign an appropriate attack potential.
The other case is where the attack potential of an existing threat changes. This change might be due to published vulnerabilities or because the potential attack rating reveals to be wrong. Those changes led to a modified version of the existing threat catalog.
Also, the attacker model itself is subject to change by adding new attacker capabilities. However, those changes do not lead to a reevaluation of SRA but an updated attack model in the next development cycle.

Damage criteria - MoRA uses the damage criteria catalog to rate the impact of damage scenarios. Therefore, they do not introduce or change risks in the analysis. Nevertheless, it might be that the rating of a damage scenario class changes, e.g., due to precedential law cases. Those situations are scarce and are negligible in most cases. Also, the need for more damage criteria in the catalog may arise. However, it is unlikely that new criteria reveal which are that important that existing

development processes need an update. Therefore, the damage criteria catalog needs no additional tracing besides its link to the SRE.

Assumptions - The use of assumptions limits the scope of the SRA or changes the impact or feasibility of risks. MoRA provides several assumptions regarding the attacker model, scope, operational environment, and cryptography.
An example that limits the scope is that attacks from workshop employees are not in scope. This limit is reasonable since, if the workshop employees would attack vehicle security, they are subject to civil and employment law, and it is acceptable that they would not risk that. The security planning step defines those limits, which are static throughout the development and life-cycle.
Assumptions regarding cryptography target, for example, random number generators or hash functions. Those also limit the scope from a different perspective and are also static. Security planning defines the defense method catalog and limits the use of insecure or insufficient cryptography. If cryptographic functions or procedures break, the risk treatment is the change target, not the SRA.
The operational environment, e.g., excludes the backend from the analysis. This limit is reasonable since the backend is subject to IT security in the sense of ISO/IEC 27001 [56]. Therefore, the scope and procedure of the analysis differ significantly, and it is not advisable to include that in the SRA. Those assumptions do not change throughout the development and life cycle.
Nevertheless, there is also the possibility of limiting the scope of the analysis by self-defined assumptions. Those assume that specific parts of the analysis are targeted to the SRA of other functions since they are their responsibility. Such assumptions are subject to tracing. Before proceeding to risk treatment, a reconciliation is necessary to ensure these assumptions hold. Therefore, the recommendation is to include those assumptions in the open issues to clarify the uncertainty about the dissolution.

8.3.4. Re-Evaluating SRAs

The elements identified for tracing, as well as the item definition, might lead to substantial changes. In such cases, it might be necessary to re-open SRAs and evaluate if they lead to new risks or change risk determinations.

Newly revealed Threats - can impact already accomplished SRAs and make it necessary to re-evaluate them. Based on the information, the security planning updates the threat catalog, and the security engineers need to evaluate to which SRAs this update affects.
The threat catalog incorporates the following information:

Target Layer	describes the architecture part the threat relates to, e.g., data flow, component
Attack Surface	the interface or location the attacker uses to execute the threat
Technology	interfaces or technologies where the external fault introduction takes place, e.g., signal communication

Based on this information and the integrated architecture model, it is possible to identify affected system parts for new threats. The system engineers filter the model for all system parts which use the affected technology and provide the attack surface. This filtering provides the directly affected SRAs which may need reevaluation. The same filtering is also suitable for attack paths resulting from vulnerabilities in the component.
Newly revealed threats and attack paths may propagate through the vehicle network. Therefore, it is necessary to evaluate communication paths starting from affected functions. This evaluation takes place after the reevaluation of affected SRAs. Since those analyses provide insight if the threat that can propagate through the network.

Attack Potential - Changes in the attack potential of threat classes "occur due to updated ratings, new use cases, or processes that impose penalties" [38, p. 99]. Typical reasons are changes in the rating of COTS system parts [38, pp. 99 sq.]. In case of such changes, the threat catalog changes, and all affected SRAs need reevaluation. The determination of affected analysis has two dimensions.
If the change relates to specific system parts, e.g., COTS, the affected SRAs are determinable using the integrated architectural model. Otherwise, if the change is due to updated threat classes or specific threats, it is necessary to make a look-up which SRAs includes those threats. Depending on how elaborate the ticket system annotations are, this system is usable. Otherwise, an analysis script that parses the SRA results is suitable.

Input Data - After accomplishing the SRA, changes in the function might also change risk analysis results. Therefore, the developer must push those changes into the security department. It is suitable to have update cycles for the SRA to prevent inconsistencies by missing updates. After the start of serial development, this is automatable by using the integrated model usable at this time.
In case of updates, it is necessary to evaluate if these changes affect the SRA results. The primary concern of the SRA is evaluating communication and stored data. Therefore, additional communication or changing communication technology demands a reevaluation. Also, changed storage patterns or amounts make this step necessary.
Changes like timing patterns or resource usage, in general, might be irrelevant. Nevertheless, the recommendation is to reassure with the development team.

Open Issues and Assumptions - Missing information during the SRA and assumptions regarding the analysis responsibility between different functions lead to open issues. The security department needs to re-cap them before proceeding to the risk treatment step. In cases where the assumptions break or the open issues reveal necessary changes, it is necessary to re-open the SRA and dissolve those issues.

8.4. Template Adjustments

For the clustering approach (R8.1), it is necessary to relate each part of MoRA to the function ID. This idea is the same as in the cluster SRE. Each entry in the SRA gets an additional annotation for the function ID(s). If an entry relates to more than one function, the analyst assigns the list of function IDs.
As mentioned in the process description section (Section 8.1), one approach is to split the risks to the OEM and the road user (R8.3). MoRA derives the risk from the attack feasibility and the damage scenario impact. While the former relates to the attacker model, independent of road user and OEM, the latter derives from the damage potential catalog, which has different levels for the road user and the OEM impact. Therefore, the damage scenarios need an adjustment to designate them separately. The most comprehensive way is to triple each damage scenario and rate it against the road user, the OEM and like as of yet to both. This approach has another advantage: How the combined impact composes is clear. Splitting the damage scenario makes the implicit maximization explicit.
The identification of risks follows the damage scenario example. The analysis result triples each risk, and the resulting evaluation is the risk level with the impact against the road user, the OEM and the maximum of both. With this, it is obvious which risks have a high rating against the road user and are thereby crucial according to the prioritization of ISO/SAE 21434.

Anyone who has never made a mistake has never tried anything new.

Albert Einstein

9 Security Risk Analysis on System Level

Typically, the function-oriented SRA takes place before the final deployment of the functions to the hardware. Therefore, it has incomplete information about the deployment and the used hardware. Also, it has a different scope: It concentrates on the functions' behavior, communication and environment. Nevertheless, also the hardware poses threats to the system. Ignoring them leads to threat propagation from the hardware development into the resulting system (see Section 5.3.5). Therefore, a complete and consistent security development process needs to supplement the function-oriented SRA results with a system view.
The system SRA completes the risk analysis cycle: The criticality evaluation of the SRE followed by the function-oriented SRA concentrating on the specifics of the function until the system view, introducing the threats resulting from the used hardware.

9.1. Process Description

To the authors' knowledge, the division between system and function-oriented SRA is unique to the evaluated development process. Therefore, no specific related work exists. Also, the evaluated security development process does currently not have a specific system SRA step, which is why no description and experience exist.
The normative reference section depicts the differences between the system and function-oriented SRA. The problem and approach sections describe the system SRA idea of whether development is subject to the remaining chapter.

9.1.1. Normative Background

ISO/SAE 21434 elaborates not separately about system SRA. The only difference lies in the attack path analysis.
In the sense of ISO/SAE 21434, a component is a "part that is logically and technically separable" [54]. This separation possibility does not necessarily relate to components in the sense of ECUs like in this thesis. Nevertheless, as defined beforehand, the items in a function-oriented development security analysis are the functions, e.g., park assistant, deployed to components (hardware) and system software, e.g., operating systems. Therefore, this hardware and system software combination is the technically and logically separable unit in the development process - the component.
The function-oriented SRA attack path analysis relies on the item definition and threat scenarios. For components, ISO/SAE 21434 suggests using the cybersecurity specifications and the threat scenarios [54].
Cybersecurity specifications regarding ISO/SAE 21434 are specifications on different architectural abstraction layers. They include the results from the security analysis: the item definition with the interfaces between sub-components, security requirements, configuration, and calibration parameters and, if applicable, already assigned defense methods (see Chapter 10).
Therefore, the attack path analysis of the system SRA uses the function-oriented SRA results. The system itself does not introduce separate damage scenarios and impacts. However, it introduces new attack path steps influencing the functions' feasibility ratings and risks.

Requirements - Based on the normative background and the evaluation of the function SRAs scope, the following requirements for the system SRA arise:

- **R9.1** The system SRA must extend the attack paths to include system influences.
- **R9.2** The system SRA must re-evaluate the risks from the function-oriented SRA
- **R9.3** The method should efficiently re-use the results from the function-oriented SRA.
- **R9.4** The method must support tracing by connecting the item definition and inputs.
- **R9.5** The method must maintain consistency between the different system SRA.
- **R9.6** The method must align with the normative references.

9.1.2. Problem Description

The change of the analysis perspective prevents a direct transfer of the function-oriented SRA method. In contrast to the function-oriented SRA, the component analysis aims to identify and rate the hardware influence on the attack paths and thereby risks (R9.1, R9.2). Therefore, also the starting point differs between both analysis steps. The system SRA relies upon two types of input data: The capabili-

ties and structure of the system and the function-oriented SRA results of assigned functions.
Modeling the system includes the hardware structure, system software, and capabilities. Therefore, it is necessary to define the amount and representation of information. One example is the incorporation of separate processors or virtualization.
In order to use the function-oriented SRA results (R9.3), it is necessary to compose those into an integrated representation.
Also, the system SRA must maintain traceability (R9.4). The normative references demand the reproducibility of accomplished process steps. Besides, it is necessary to be able to backtrack from the implementation throughout the analysis steps till the beginning of the security development process.
The security planning step defines the basic templates and catalogs used throughout the security development process. Therefore, it is reasonable to ground also the system SRA on a distinct basis. This approach maintains conciseness between different system SRA (R9.5).

9.1.3. Approach

In order to design a methodology for system SRA it is possible to evaluate the general SRA steps from the normative references (R9.6). The aim is to define the method based on the necessary information for a system viewpoint. This approach enables the derivation of the input data structure and necessary input data (see Section 9.2).
The item definition of the system SRA is twofold. One part is the function-oriented SRA results providing the function information, and the other depicts the hardware and system software information. The latter results from the deployment information, including the function placement. The idea is to reduce the modeling effort to a few compatible model elements (see Section 9.3). This approach allows varying abstraction levels for the analysis.
For the function-oriented SRA results, it is necessary to compose them in a data structure suitable to support the system SRA. Therefore, the analysis results need a separation according to their deployment (R9.3). This step is necessary since cluster SRA can include functions from several different components. Therefore, annotating the hardware deployment in the function-oriented SRA is necessary (R9.4). The MoRA method allows this annotation already. Nevertheless, in most situations, the deployment is not fixed during the function-oriented SRA. In such cases, it is necessary to complete this information. Alternatively, it is possible to backtrack the results based on the components item definition. Therefore, this information is also used to divide the function-oriented SRA results in order to use them in the system SRA item definition.
The central part of the system SRA is the identification of threats to the hardware and system software in order to supplement the attack paths and update the attack feasibility rating (R9.1). Therefore, it is necessary to have a concise data basis for the threats and the attacker model (R9.5). Following the MoRA approach

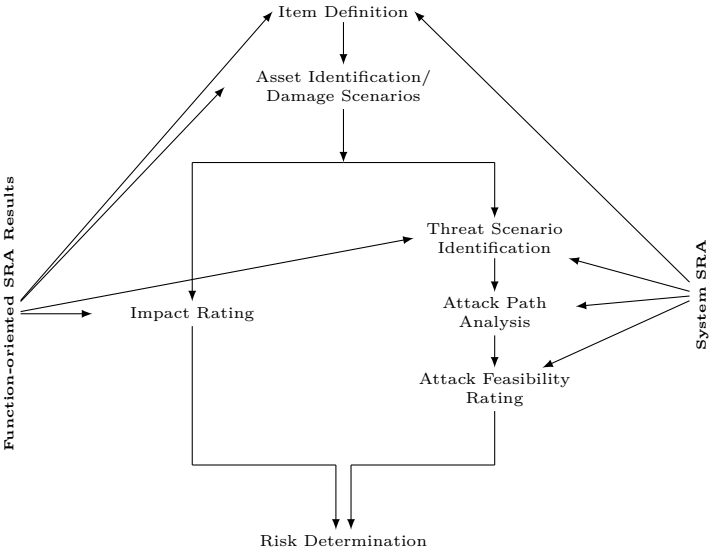

Figure 9.1.: Influences from the function SRA results on the system SRA and steps necessary to accomplish in the system SRA.

of providing catalogs for those analysis parts, Section 9.4 provides an overview of threat and attacker models and examples for basic threat classes. This threat model is usable for the system SRA analysis (see Section 9.5), which aims to identify the additional attack path steps and update the attack feasibility rating and, therefore, the risks.

9.2. Method Design

The system SRA uses the same normative background like the function-oriented SRA. Nevertheless, it has a different scope and approach. This section outlines the method for the system SRA, starting with the analysis scope. The remaining part of the section transfers the SRA steps to the system viewpoint and develops the idea of the method.

9.2.1. Scope

The start of each analysis method design is its scope. In function-oriented development, the functions define the systems' value and harm. The function-oriented SRA targets a function's internal structure and the communicated data to evaluate

risks to the system. For the system SRA, the scope changes onto the data handling in terms of data storage and flow.

9.2.2. System Viewpoint SRA Steps

The system viewpoint adds attack path steps, reassessing the feasibility of the attacks and, therefore, the risk levels. This approach requires including the supporting SRA steps and leaving others aside.

The item definition aims to model the system under consideration, its boundaries, the relationship between the systems, and its environment [17]. In the case of a system SRA, these are the hardware elements and system software: the ECUs, high integration platforms[1], sensors, actuators, and according network interfaces, operating system and middle-ware.
The system SRA focuses on data storage and flow making it necessary to model the system with its deployed functions. Therefore, the item definition includes the results from the function SRAs of the deployed (sub-) functions (R9.3).

The Asset Identification/Damage Scenarios includes identifying the assets in the item model and the relation between the item parts and adverse consequences. Assigning security objectives to the assets leads to the damage scenarios [54].
Security risk analysis targets the risks against the vehicle functionality. Therefore, system SRA does not add damage scenarios but takes over the damage scenarios result from the function-oriented SRA.

Impact Rating assesses the damage scenarios against adverse consequences regarding safety, financial, operational, and privacy [54]. Therefore, each damage scenario gets an impact level (negligible to severe) for the different categories.
The system SRA adds no additional damage scenarios. Therefore, the results from the function-oriented SRA need no update.

Threat Scenario Identification evaluates which threats can realize a damage scenario. It is possible to use information besides the damage scenarios: “technical interdependencies between assets, attackers, methods, tools, and attack surfaces” [54].
Threat modeling approaches support deriving the threat scenarios. MoRA uses a threat catalog following the STRIDE model [30]. This attempt is reasonable and maintains a high degree of confidence in completeness. Also, a threat catalog provides standardized attack potentials, which maintains consistency between the different system SRA.
Incorporating a system threat catalog helps to identify threats arising from the system and maintains conciseness (R9.5). Section 9.4 covers the details of the threat catalog for the system SRA.

[1] Regarding conciseness, the remaining chapter incorporates high integration platforms into the term ECU if it is not necessary to emphasize the difference.

Attack Path Analysis and feasibility rating evaluates the threat scenarios regarding the necessary steps to accomplish the threats and rates their feasibility. Since the system adds steps to the functions' attack paths, it is necessary to update also the feasibility ratings.
The attack path analysis and feasibility update are part of the analysis section (see Section 9.5).

Risk Value Determination combines the attack path feasibility and the associated damage scenario impact.
Following the approach of MoRA is reasonable to maintain consistency. Therefore, the method reuses the existing damage scenario impacts from the function-oriented SRA and updated attack paths from the preceding steps (R9.3).
The function-oriented SRA chapter (Chapter 8) suggested designating the risk against the road user and the OEM separately. The system SRA follows this suggestion. Using the evaluated attack feasibility ratings allows us to re-evaluate also the functions' risks. The result of the risk value determination is an updated version of the composed function-oriented SRA risks. The system influence either raises or lowers the function-oriented SRA risks.

9.3. Item Definition for System SRAs

Each risk analysis begins with the item definition. In the case of the system SRA, the data basis is available through the ECU information in the vehicle project and should contain the following items:

- ID of the ECU
- Hardware resources, e.g., virtualization, network interfaces, HSM, storage
- Operating systems and middle-ware
- Deployed (sub-) functions
- Connections to other ECUs

The ID of the ECU is necessary for tracing the input and output of the analysis and throughout the remaining process. System resources provide the functional environment for the functions and are therefore primary part of the data flow - the scope of the analysis. Storage resources handle data and provide access through shared or multiplied interfaces [1]. Network interfaces give rise to aspects of data handling and security threats on the platform [2]. Virtualization techniques lead to an extended hardware modeling to incorporate the different isolation units. Already assigned defense mechanisms are part of the ECU technologies. They secure the system but pose threats to it, e.g., secure key storage in a HSM vs. insecure interfaces to the HSM.
The system software incorporates the operating system and middle-ware(s). These provide the run-time environment for the deployed functions and, thereby, the information for determining the data flow in the system.
The (sub-) functions of the system depict the functionality and give rise to the damage scenarios for the system. They access data through interfaces of the holding

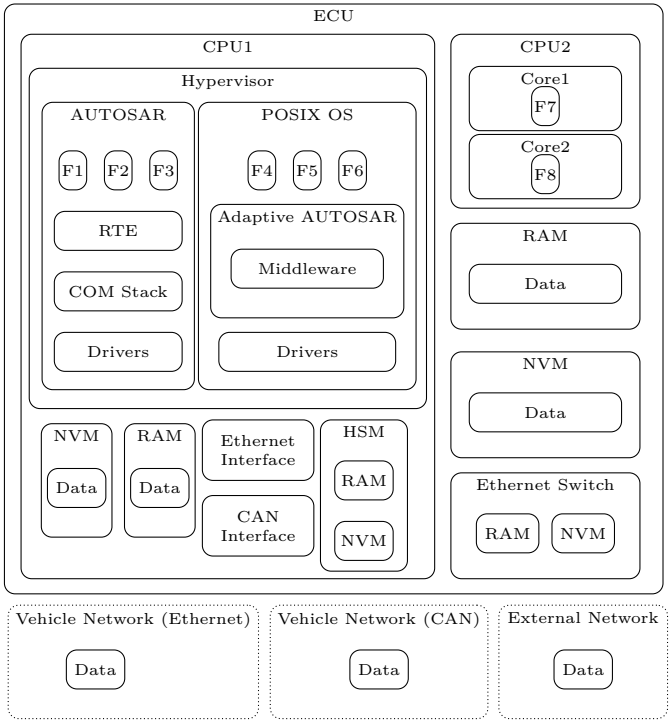

Figure 9.2.: Example architecture for a high-integration platform with two CPUs and different network interfaces.

resource. Also, the list of deployed (sub-) functions is the criterion for the function-oriented SRA result composition.
The item definition consists of the component model and the function-oriented SRA results. Since these are static inhabitants, it is reasonable to include the operating system and middle-ware into the resource part of the item definition. This approach divides the analysis parts of the system SRA between the already covered function-oriented SRA results and the new analysis subjects, the hardware, and system software (operating system and middle-ware).

9.3.1. Hardware Architecture Hierarchy

The basis for the item model is the system architecture providing the run-time environment for the functions. Figure 9.2 provides an example for a high-integration platform with virtualization, and different operating systems.
This architecture provides a hierarchy that depicts the nesting of the hardware structure, the system software, and the deployed (sub-) functions:

Functions access data via interfaces to the providing resource. For the system SRA the autonomously deployed sub-functions are the subject of the analysis. Therefore, they are functions in the sense of this analysis part.

System Software provides the run-time environment for the deployed functions. Thereby, it can be hierarchically structured and encapsulated. For example, a CPU hosts an operating system with an assigned middle-ware and a function in the system hierarchy. The operating system and the middle-ware encapsulate the function from direct CPU access.

Interfaces allow different ECU inhabitants to access resources. They establish a connection between the requester and data. Resources can directly provide interfaces (e.g., a memory module) or indirectly through nesting (e.g., by using a middle-ware).
External interfaces depict connected sensors/actuators and different network interfaces. They are accessible through interfaces to send or receive data.
Network interfaces are the connections to other ECUs providing insights about the overall system architecture.
Therefore, a system architecture consists of the hardware, the system software, and the provided software functionalities decomposed into autonomous functions. The example ECU in Figure 9.2 is one component out of the complete system architecture. It consists of two CPUs, different storage resources, network technologies (CAN and Ethernet), and an embedded Ethernet Switch. In automotive systems, Ethernet switches tend to be embedded into an ECU and often have firewall capabilities. Storage resources can be either encapsulated in a CPU or the ECU providing unrestricted access to all CPUs.

9.3.2. Structural Model

The structural model represents the system architecture as a connected graph G_H. The graph takes the viewpoint of the function, where every hardware element and system software is a resource. Therefore, the graph depicts the system hierarchy of hardware with assigned system software and the functions as “leaf” nodes.

$$
\begin{aligned}
&G_H = (P, R, X, Y) \\
&\quad P = \bigcup_i P_i : \text{ partition of F where F is the set of all functions} \\
&\quad R : \text{set of resources} \\
&\quad X \subseteq \left\{ \{p, r\} \;\middle|\; p \in P, r \in R \right\}, \forall p \in P : \exists! \, \{p, r\} \in X \\
&\quad Y \subseteq \left\{ \{r_1, r_2\} \;\middle|\; r_1, r_2 \in R \wedge r_1 \neq r_2 \right\}
\end{aligned}
$$

The set of functions F is partitioned into sets P_i, representing the functions assignment to units of isolation deployed to a resource element. Resource elements R are the different CPUs, storage technologies, networking interfaces, and system software. Those resource elements depict the system's hierarchical architecture. Interfaces X and Y connect resources and requester. Therefore, the model representation depicts a relation between both as edges. This relation is possibly hierarchical in the case of indirect access through nesting. Those cases lead to an access path through different levels until resource access.
For the presented example architecture excerpt, Figure 9.3 shows the resulting architecture graph. The graph consists of all functions deployed to the two CPUs, their corresponding encapsulation, and their connection to networking technologies. In reality, the architecture graph represents the complete system architecture.

9.3.3. Communication Flow Model

The system SRA aims to add the influence of data flow to the attack paths of the functions. A communication flow graph illustrates the communication links between the collaborating functions. The composed function-oriented SRA results provide the necessary information, i.e., if they have a proper abstraction level. Otherwise, they only provide the sender and receiver of messages, and other models are necessary to complete the information.
For the composition of the function SRAs, the deployment information of the functions is the main composition criterion. Combining the function-oriented SRA results concerning their deployment to the units of isolation in the architecture graph leads to the set of risks for each unit of isolation.
The easiest way to accomplish the composition is to parse the MoRA results and copy them into a new template for each isolation unit. It is also possible to use one integrated template per component and annotate the isolation units for each entry. The result is the necessary input for the system SRA communication flow model.

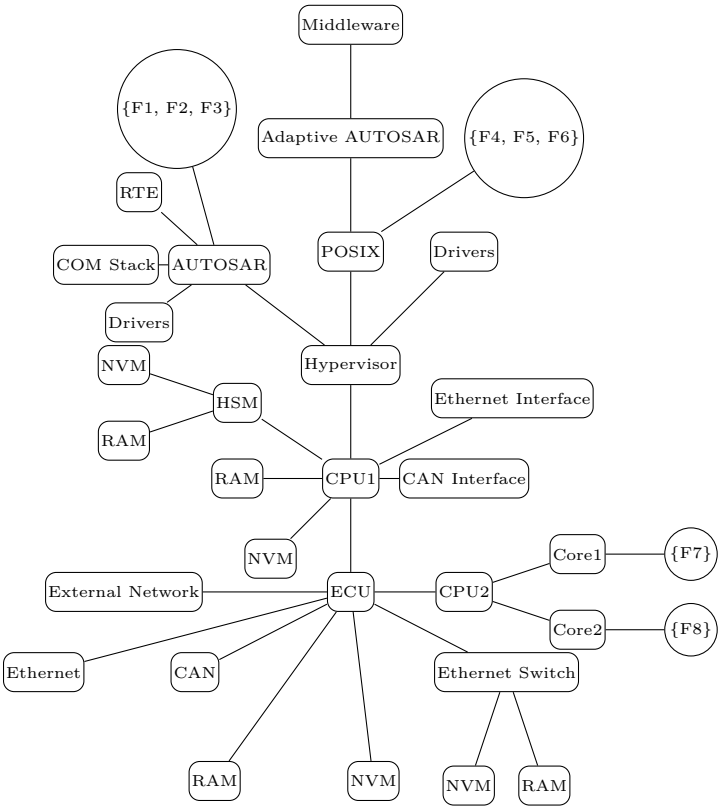

Figure 9.3.: Architecture graph for the example architecture of Figure 9.2. Circles represent functions. Rounded rectangles represent structural elements in hardware or software.

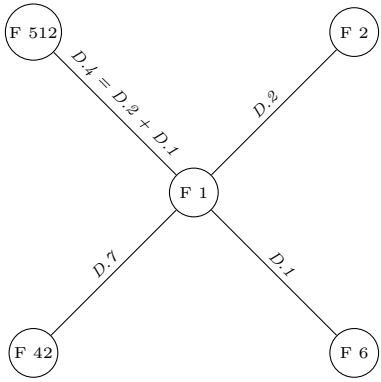

Figure 9.4.: Example of a communication flow graph of function $F1$.

$$
\begin{aligned}
G_D &= (F, C, W, D) \\
F &= \text{set of all functions} \\
C &\subseteq \left\{ \{f_S, f_R\} \in F \times F \right\} \\
D &= \text{set of all data items} \\
W &: C \rightarrow D
\end{aligned}
$$

A communication flow graph is a connected graph G_D including the functions F and communication links C (see Figure 9.4). The item modeling builds such a graph for each function in the system. Each communication link gets an annotation W of the communicated data item $D.i$. If this data item is an aggregation, e.g., $D.4 = D.1 + D.2$, the aggregations serves as the annotation.
The presented graph is undirected, meaning all information can flow in both directions. While this is true in many cases, it leads to an overestimation in the other situations. Nevertheless, it reduces the model overhead, leading to higher efficiency.

9.4. Threat Catalog for System SRAs

As proposed in the approach, it is reasonable to use a threat catalog for analyzing attack paths and feasibility. Following the literature, a threat catalog incorporates the attacker and possible threats to the system [106]. The former defines possible attackers with their motivation, equipment, and possibilities. The latter part describes threats to the system.

The threat catalog in MoRA includes several threat classes and example threats whether feasibility is rated with an attacker model. This section develops a threat class catalog for system SRA. At first, the section elaborates on threat models and attacker models before presenting example threat classes for the system SRA.

9.4.1. Threat Model

A threat model narrows the threat identification step of the risk analysis process. To prevent a too-narrow or too-wide risk determination, it is necessary to define a distinct threat model for each system type [58]. The security planning step defines the threat model for the complete development cycle. Therefore, the system SRA of the different system parts use the same assumptions. If there is a newly detected threat to the system, the threat model needs an adjustment leading to possible reevaluations of the completed risk analysis. For that, the deliberations of Section 8.3 apply.
While the content of threat models can be different, the basics they should define are the same: the threat surface, the threat type, the threatened security objective, and the interfaces or technologies.

Attack Surface is the interface or location the attacker uses to execute the threat [120, p. 25]. The distinct formulation of the attack surface leads to structured threat analysis. Thereby, the attack surface definition can have different levels of detail. An abstract attack surface definition for automotive systems differentiates two surfaces attackers may use, independent of their motivation:

Local attack: The local attacker has physical access to the vehicle, its ECUs, the vehicle network, and all its interfaces. It is unnecessary to distinguish whether or not the attacker has legitimate access to the vehicle.

Remote attack: A remote attacker has no physical access to the vehicle but attacks the termination points of the vehicle's external interfaces.

For the subsequent analysis steps, it is notable that the attack surface influences the assumptions on attack probabilities. A local attack does not scale since it has to be performed manually on each vehicle. Also, the local attacker can access all vehicle interfaces and perform remote attacks but prefers easier local attacks. On the other hand, remote attacks potentially scale to the entire fleet. Since remote attacks use external interfaces of the vehicle as entry points, it is assumable that a remote attacker compromises the first termination points of online communication (wireless local area network (WLAN), Bluetooth). Therefore, also communication on connected buses may be compromised.
The attack surface classification allows distinguishing the threats according to local and remote threats. It is also possible to exclude threats based on stated assumptions, explicitly accompanied by an appropriate reason. An example is threats that are achievable easier than by threatening cyber security, e.g., by mechanical manipulation.

Threats denote the goal of the attack [106]. Microsofts' STRIDE model classifies threats into six categories (see Table 8.1) [41]. While [41] suggests using the STRIDE model on the decomposed system, it is also possible to use the provided categories for establishing a general threat model, later usable in the risk assessment process. This model incorporates known threats and attacks from former development and threat databases suggested by [38]. Thereby it is made sure that in a distributed and heterogeneous system like in the automotive industry, the risk analysis of the different system parts uses the same assumptions.
The combination of threats with the respective security attribute allows to focus the subsequent security process steps on specific security attributes or identify the ratio of the security objectives. This combination leads to a classification of threats to the system like in Table 8.1.

Technology allows to identify of attack patterns [41]. Those are preconditions for executing the threat. On the other hand, technologies introduce new threats, making the threat model more comprehensive.
For that, it is necessary to enumerate the interfaces and technologies which concretize the threat classes regarding the (access) type for each system class [17] – defining and combining the nature of the interfaces and technologies with the threat classes narrows analysis objectives, and the setting in which the security analysis takes place.
An example would be internal communication interfaces not exposed to the vehicle user. Those are distinguishable into signal communication and service communication combined with the attack surface and security attributes, e.g. local extraction of data via signal communication, where the target layer is data, the technology is signal communication, and the affected security attribute is confidentiality.

9.4.2. Attacker Model

Attacker models are dual to threat models. They define the classification of attacker capabilities to execute a threat. Later in the risk assessment process, the feasibility rating of threats and attack paths instantiates this attacker model.
[38, p. 22] distinguishes attacker models into attacker-centric and capability-centric ones. Attacker-centric models focus on the attacker's type and capabilities, e.g., if he is a system user or an external attacker. They define the type of attacker who generally may attack the system. Capability-centric attacker models focus on the access, the time, expertise, knowledge, and equipment needed to execute the threat [17, 38, 120], e.g., hacker vs. vehicle owner performing chip-tuning.. Therefore they relate to certain system threats and rate-defined threats. Since the attacker model is usable as a schema to rate defined threats and attack paths of the system, this work follows the idea of MoRA. It uses the idea of capability-centric attacker models.
Table A.1 provides an overview of the categories incorporated in the attacker model. Each category has several levels with assigned values based on a discrete scale [38, p. 57]. Those values provide the rating for the minimum effort in this category [96,

Threshold	RAP	Number
0	Very low	1
14	Low	2
20	Moderate	3
25	High	4
35	Beyond high	5

Table 9.1.: Thresholds for estimating the required attack potential

p. 86]. Comparing the sum of these values with the threshold matrix (see Table 9.1) provides the required attack potential - the capability and effort an attacker needs to possess to successfully realizes the attack [38, p. 57].
Interested readers may have a look into Appendix A for further information about the attacker model categories.

9.4.3. Threat Classes

[102] states that each item element is attackable directly by modifying the entity or indirectly through the data flow. This view leads to the following general threat classes for data flow and data storage, taken from [102]:

- **Data flow:**

  Tampering of the data flow, e.g., man-in-the-middle attacks.

  Information disclosure on a communication channel, e.g., recording of unencrypted radio signals.

  Denial of Service of a data flow, e.g., GSM jammer.

- **Data storage (processes, external entities, storage):**

  Tampering of data storage leads to modified behavior, e.g., attacks by altering the ECU firmware.

  Denial of Service of data storage leads to timing, omission faults, or crash faults, e.g., attacks by uncontrolled resource acquisition

  Information Disclosure includes not only intellectual property but also cryptographic keys or passwords, e.g., attacks by reverse engineering or side-channel attacks.

Transferring those general threat classes onto the hierarchical hardware model (see Section 9.3) leads to threat classes for the system SRA. As already stated, the data flow uses interfaces between the requester and the data. Therefore, for each attack, it is necessary to target an interface that makes them the threat classes' primary concern.
An elaborated threat catalog is necessary for usage in the system SRA. Building such a catalog is out of the scope of this work. Table B.1 provides an illustrative example for basic threat classes. This example does not include the technology and surfaces. Each threat class is possible for a local and a remote attacker and different

technologies. For usage in real use cases, the threat classes need an extension for the different surfaces and technologies. Also, the provided attack potential has no claim for correctness. In order to complete and adjust the threat classes, it is recommendable to include the information from CAPEC [82] which is an attack pattern catalog maintained by MITRE.

9.5. System SRA Analysis

The analysis step aims to extend the function-oriented SRA attack paths to include the systems' influence. To accomplish this, it is necessary to annotate the attack potential to the provided item model. Subsequently, the graphs are transformable to evaluate attack paths from the system view and re-evaluate the risks by propagating the required attack potential elements. Thus, not the Required Attack Potential (RAP) but its underlying category levels are the exciting information. The remaining section uses the term RAP level for these category levels.

9.5.1. Annotating the Attack Potential

In order to complete the architecture graph and the communication flow graph, it is necessary to annotate the attack feasibility of the interfaces and communication links. The function-oriented SRA results provide parts of them in the threat definitions. The threats relate to the functions, the components, and the data flows. Also, they give rise to the used technology, thereby providing the information that interfaces the data flow.
By annotating the graph relations, the threats may provide different RAP information for the same model element. The higher the RAP is, the more the effort for an attacker to accomplish this step, and better defense methods are necessary. Therefore, in the case of contradictory RAP information, it is reasonable to use the minimum of the different elements.
Missing relations are subject to completion by using the threat classes for the system SRA. Each edge in the architecture graph and communication flow graph has the RAP levels as an assigned vector.
Therefore, the complete definition of the architecture graph is as follows:

$$
\begin{aligned}
G_H &= (P, R, X, Y, W_x, W_y, RAP) \\
P &= \bigcup_i P_i : \text{ partition of F where F is the set of all functions} \\
T &= \text{set of resources} \\
X &\subseteq \left\{ \{p, r\} \;\middle|\; p \in P, r \in R \right\}, \forall p \in P : \exists! \{p, r\} \in X \\
Y &\subseteq \left\{ \{r_1, r_2\} \;\middle|\; r_1, r_2 \in R \wedge r_1 \neq r_2 \right\} \\
RAP &= \text{The set of the required attack potentials as vectors} \\
W_x &: X \to RAP \\
W_y &: Y \to RAP
\end{aligned}
$$

For the communication flow graph, the weight function changes to include the RAP:

$$
\begin{aligned}
G_D &= (F, C, W, D, RAP) \\
F &= \text{set of all functions} \\
C &\subseteq \left\{ \{f_S, f_R\} \in F \times F \right\} \\
D &= \text{set of all data items} \\
RAP &= \text{The set of the required attack potentials as vectors} \\
W &: C \to D \times RAP
\end{aligned}
$$

9.5.2. Graph Embedding

The evaluation of attack paths demands a graph embedding of the communication flow graph into the architecture graph. This approach allows for evaluating chains of actions in the data flow. The embedding is a composition of the communication flow graph for each function and the architecture graph of the system architecture. For each link in the data flow graph, the path in the architecture graph is evaluated (see Algorithm 1). The result is a tuple representing the data flow through the involved resources. Thereby, in case of an aggregated weight at the communication link (e.g., $D.4 = D.1 + D.2$), the pathfinding includes the sub-parts (e.g., $D.1$ and $D.2$), too. The path's first and last elements are the requester and provider of the information, thereby leaving the architecture graph. The path from the incorporated functions in the architecture graph conserves the order of the functions and resources along this path. The tuple is merged into a set of tuples representing the union of all paths for each function.
Over these paths, it is possible to derive the attack potentials of the functions by propagating the minimum RAP levels through the paths (see Algorithm 2). The result is a vector allowing to estimate the RAP of the data flow (see Algorithm 2).

Algorithm 1 Estimate Paths for the functions

Require:
$G_H = (P, R, X, Y)$
$G_D = (F, C, W, D)$
visited = C ▷ Set of all edges in G_D
paths = $\emptyset$
Ensure: |paths| = C ▷ Each edge is part of a path

while c $\in$ visited **do**
    paths = paths $\cup$ PATH(c, G_D) ▷ Determines the path for $\{x_0, x_1\} \in C$ in the graph G_D
    visited = visited $\setminus$ c
end while

Algorithm 2 Evaluate RAPs over the paths

Require:
paths ▷ Input from the path estimation algorithm
data ▷ Set of data items for annotating the RAP per item
Ensure: paths = $\emptyset$ ▷ every path is handled

while p $\in$ paths **do**
    min = empty vector

    while i $\in$ p **do** ELEM_MIN(min, i) ▷ Determines the element-wise minimum of the path elements RAP and the current minimum
        p = $\setminus$ i
    end while
    UPDATE_DATA(data,min)n ▷ Annotates the resulting RAP
    paths = paths $\setminus$ p
end while

Algorithm 3 Re-evaluation of the risk levels from the function-oriented SRA

```
Require:
 1: risks as array of ID × Data_items
 2: data                                          ▷ Set of data items with annotated RAP

 3: while d ∈ data do
 4:     r ⊆ risks : d ∈ Data_items
 5:     while i ∈ r do
 6:         Assigns result of element-wise minimum to the RAP of the risk
 7:
 8:         r = \ i
 9:     end while
10:
11:     data = \ d
12: end while
13: UPDATE_RISKS(risks)          ▷ Re-evaluates all risk levels based on the changed RAP values
```

This re-evaluated RAP allows to update the risks from the function-oriented SRA (see Algorithm 3). Therefore, each item in the data item set leads to an update of the risks and whether the attack path incorporates them. Ultimately, all risks are re-evaluated based on the changed RAPs.

Premature optimization is the root of all evil.
Donald Knuth

10 Risk Treatment

After the risk analysis, risk treatment leverages the risks by assigning defense methods. Trade-offs and cost analysis help to optimize the treatment solution [91, p. 507] still ensuring a suitable level of protection [38, p. 2].

10.1. Process Description

While it is possible, and often done in practice, to include risk treatment into the risk analysis process, this work implements it as a separate process step (see Chapter 8 for rational). This section evaluates the normative background and current process implementation before it proceeds to related work, the problem description, and the approach.

10.1.1. Normative Background

In the case of risk treatment, ISO/SAE 21434 and UNECE No. R155 have specific demands for the security development process and the system design.

UNECE No. R155 demands to protect the vehicle type and to implement "all mitigations [...] which are relevant for the risks identified"[85].
Besides the general demand for implementing defense methods, the UNECE No. R155 states three concrete demands:

- Intrusion-Detection for the vehicles
- A central monitoring facility for new threats and vulnerabilities
- The use of up-to-date cryptographic modules

Annex 5 of [85] provides a list of possible risks and appropriate defense methods. The list in the Annex are not concrete implementations, but rather categories of defense methods, e.g., "The vehicle shall verify the authenticity and integrity of messages it receives" [85]. The OEM may differ from the provided list of defense methods if the list is insufficient to mitigate a specific risk.
In conclusion, the UNECE No. R155 demands risk treatment according to the provided categories in the Annex. The general demands are only OEM-related, one regarding the monitoring facility, and two vehicle-related demands. There is no suggestion regarding threat mitigation techniques or methods.

ISO/SAE 21434 demands using the item definition, the identified attack paths, and the risk values resulting from the risk analysis. Optional inputs are cybersecurity specifications, previous risk treatment decisions, damage scenarios with impact ratings, and attack paths with feasibility ratings.
Thereby, ISO/SAE 21434 requires the risk treatment for all identified risks by using one or more treatment options. Those options are the classical ones (see Section 5.5): threat prevention, threat reduction, and the security planning options risk-sharing or retaining. The process documentation must record the decision to retain or share risk. Therefore, ISO/SAE 21434 explicitly allows risk acceptance up to a certain level, as long as the assessment report documents this threshold and the retained risks. Like the UNECE No. R155, the ISO/SAE 21434 does not provide possible methods for risk treatment.

Requirements - Following the normative references and the aim of this work, the following requirements apply:

R10.1 The risk treatment method must support different levels of defense.
R10.2 Risk treatment must align with the normative references and policies.
R10.3 The risk treatment process must support different trade-offs.
R10.4 Risk treatment must be realizable in an efficient way.

10.1.2. Implementation in the Process

Currently, the function-oriented SRA includes risk treatment semi-automatic (R10.4). The MoRA template assigns defense methods to the attack paths based on the threat classes and the security objective [2]. To enable this semi-automatic assignment, MoRA provides a defense method catalog.
The defense method catalog results from the security planning step and defines four different groups of defense methods:

Perimeter security targets wired and wireless external interfaces.

Domain separation reduces the remote attack surface by separating the external interfaces from the vehicle network.

ECU security targets the interfaces to prevent manipulation at rest and during run-time.

Function security aims to secure the system's functions higher than other categories.

In the semi-automatic assignment, the process compares the defense methods with the attack paths regarding the following properties: Technology, attack surface, supported security objectives, and dependencies. The first properties are straightforward. Technology describes the underlying technical implementation, e.g., Ethernet. The attack surface provides information against which kind of attack the defense method is usable - local or remote. In order to apply suitable defense methods, security objectives information is necessary. The dependencies target the reliance of the defense method on other defense methods, e.g., a cryptographic protocol might rely on a HSM for secure key storage.
The defense method catalog provides the standard defense methods used in the development process. Nevertheless, it is not exhaustive, meaning that there might be cases in which the catalog is not sufficient. In such cases, it is necessary to include other defense methods manually in the security development process.

10.1.3. Related Work

In [66] the authors define several levels for the selection of defense methods. Their categorization is very exhaustive, e.g., to the circuit level. In practice, the risk analysis for components grounds on a selected hardware platform. Therefore, at this point in the development process it is complicated to change certain specifics, making it most of the times unnecessary to take such detailed levels for hardware into account. Therefore, staying on more abstract and less detailed control categories is sufficient. Their approach to risk treatment is a rich and detailed data model. The data model allows a very comprehensive analysis of dependencies between defense methods. On the other hand, this also demands the use of their entire framework for the security process. Otherwise, the demanded input data model may be impossible to acquire.
[91] describes a categorization of defense methods. The main categories are encryption, software, hardware, and physical. Typically, encryption is no stand-alone method but a feature of another defense method, e.g., secure boot or Transport Layer Security (TLS). Also, the authors integrate defense methods implemented as separate functions like intrusion detection systems and password checkers into software and hardware levels. In contrast, this work differentiates between the level a control has its impact on and the technical and functional control categories. Those include all applicable controls the authors mention, just in a different categorization.

Also [24] describes the categorization of security requirements into different layers. Those are the points where the information situates: internally, externally or both. Additionally, the author differentiates between the development stage and the methods' scope. In contrast, this work concentrates only on the product development and the customer usage time of the system, leaving process demands aside. Procedural methods are not assignable to the system in the risk treatment step but accompany the complete development process. Therefore, they are subject to security planning and are out of the scope of this work.
The literature review of [5] results in a thematic taxonomy regarding the different layers of software-defined networks. The categorization is very comprehensive but not easily transferable to other system domains.
[4] presents another taxonomy based on literature research. The scope of this work is regarding software integrity protection techniques only. Different system views build the basis for the categorization: system view, defense view, and attack view. They evaluate related work, map it onto the views, and evaluate correlations. Nevertheless, besides the different scopes, the granularity of the taxonomy is not helpful for the approach in this work. The author aims for a flexible approach to clarify the terminology needed for the defense method assignment.

10.1.4. Problem Description

The vast amount of different terminology is the first problem when looking for demands and approaches for component-global risk treatment. Unclear terminology makes it difficult to understand demands and compare approaches. An example is UNECE No. R155 [85]. While its main section uses the term mitigation, the Annex uses security control, measure, and mitigation without proper definition. The same applies to other sources in the literature. Security requirement demands from system external sources, like normative or organizational policies, are often mixed with requirements related to specifics of the target of evaluation (R10.1). After risk treatment, applied trade-offs must adhere to a system's external security demands. Otherwise, the system might be cost-effective but infeasible from a legal point of view (R10.2).
Risk treatment assigns defense methods to the system elements. Those defense methods rely on specific system capabilities, e.g., a HSM. In the case of function level risk treatment, it may be that assigned defense methods are impossible on the available hardware platform. Also, function level risk treatment may lead to different defense methods with the same or similar influence. Such results lead to higher resource demands. Besides, this assignment method is less efficient than assigning defense methods on a global component level, using a by-catch test for other functions (R10.4). This approach makes it possible to introduce defense methods on different abstraction levels and with different trade-offs (R10.3) [24, p. 198].
Security always relates to the system environment [40]. Regarding risk treatment, this relation is twofold. On the one hand, assumptions in the function-oriented SRA limit the analysis scope and make statements about the security of other system elements. Verifying these assumptions is part of the subsequent system validation

and verification. On the other hand, risk treatment introduces defense methods to interfaces and communication channels that might interfere with the capabilities on the receiver side.

10.1.5. Approach

There is a need to find a distinct definition of the terms used for the risk treatment in automotive systems. Such a structured data basis enables efficient risk treatment (R10.1, R10.2, R10.4). Introducing a division between the origin of the requirement also supports verification of the results regarding the normative demands. Therefore, Section 10.2 defines a simple structured taxonomy based on a distinct terminology to support the division between requirements origins, types, and methods. The derived defense method catalog allows for efficient risk treatment.
As already stated, defense methods may influence several risks in the system. Therefore, it is recommendable to accomplish risk treatment on a high abstraction level to support a by-catch-test on other risks (R10.4). On the other hand, this approach may lead to state space explosion. This issue makes it necessary to find a suitable level of abstraction. Section 10.3 discusses this issue and provides a method for risk treatment assignment on a component global basis.
Risk treatment aims to minimize system threats by reducing the impact or the feasibility. Prioritizing the defense method assignment increases this process step's efficiency (see Section R10.4). Besides, those priorities allow trade-offs between different impact categories or attacker model categories (R10.3). The presented approach (see Section 10.3) aims for a flexible prioritization of the risk treatment procedure and the risk impact categories to allow different trade-off points.
Risk analysis determines threats to the system based on attack paths. Those have different configurations. Single threats (one-element attack paths) are either comparable to single point of failure (SPOF) in reliability or are preparation attacks for other paths. Prioritizing them in the treatment process mitigates crucial threats (SPOF) and preparation attacks. The latter directly cut attack paths, reducing the effort of treating those. Updating the risks by determining the impact of the defense method on other risks raises the by-catch and reduces the overall effort of implementing defense methods (R10.4).
Another problem is the maintenance of assumption tracing and dependencies between hardware items. Without further tool support, validating those automatically in a component-level view is impossible. Therefore, this is left open for future work, making it necessary for the security engineers to validate the dependencies and assumptions manually.

10.2. Security Defense Methods

Security defense methods have a variety of names, definitions, and granularity [38, p. 66]. This section aims to overcome this problem by describing a structure and classification for a common understanding of the risk treatment methodology. The resulting taxonomy completes the overall security taxonomy (see Section 5) by providing the development process-related defense methods for threat mitigations.

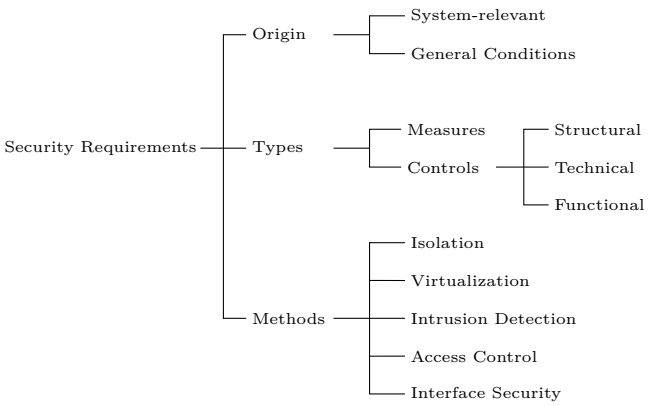

Figure 10.1.: Taxonomy of Security Defense Methods.

The taxonomy (see Figure 10.1) has three categories: the origin, the type, and the defense methods. The objective of the taxonomy is the use in the risk treatment step. Therefore, the scope is on those requirements that apply to the vehicle and directly influence its behavior. Process requirements (e.g., audit, testing) and supporting processes (e.g., documentation) relate to the ecosystem in the development process and are therefore out of scope.
Including related work and the presented methods for threat mitigations (see Section 5.5) aims to achieve completeness of the taxonomy. The use in the risk treatment process and the possibility to combine it with the security taxonomy enables usefulness and completeness of taxonomies [50].

10.2.1. Security Requirements

A taxonomy of security requirements aims to support the security planning process. It helps to define the threat mitigation processes and thereby categorize the influences to risk treatment. Threat mitigation procedures incorporate organizational and regulatory requirements (R10.2) and state-of-the-art requirements for the system type [105, p. 11].
The resulting set of requirements is a set of "security-motivated constraints" to the system [15]. Therefore, risk treatment assigns defense methods to leverage the identified risks and supplements the system constraints.

10.2.2. Origin

As already stated, security requirements arise from different sources: General conditions from normative references and organizational policies and system-related requirements from the items security analysis.

General conditions, or extrinsic security requirements [38, p. 26], arise from external sources. They may not directly support accomplishing a security objective, but non-fulfillment provokes a failure of the development.
Regulatory organizations are the most critical source for those requirements (R10.2). Norms like UNECE No. R155 [85] and ISO/SAE 21434 [54], but also country-specific regulations like GB/T 40856-2021 [107] demand certain types of security requirements in a varying level of detail. Examples are the intrusion detection demand of UNECE No. R155. Those regulatory requirements are relevant for the type approval of the vehicle project. Therefore, a non-accomplishment endangers the possibility of selling the vehicle type, at least in certain countries or regions, e.g. Europe.
Security-in-depth and security-by-design demand to define certain basic security standards on varying organizational level [105]. Those are also a source for general conditions. A non-accomplishment of those policies does not lead to the loss of type approval but endangers the company's internal vehicle audit, e.g., during quality assurance. Also, company policies may have a varying level of detail from distinct methods in a certain variation to general demands.
The nature of general conditions is that they have varying levels of detail. They may not explicitly project to specific defense methods but categories of those. Therefore, they may not directly support accomplishing a security objective (e.g., Confidentiality) to a certain degree but demand a defense method that supports this goal.

System-related Requirements, or intrinsic requirements [38, p. 26] arise from the security risk analysis.
The system-related requirements are distinct since they base on specific risks according to threats against security objectives and impacts regarding their damage and attack potential. Therefore, system-related requirements can be refined and transformed until it is possible to map a defined variation of specific defense method to leverage the risk.

10.2.3. Types of Security Requirements

The type of security requirements defines the nature of the security requirements. The taxonomy categorizes them into those directly recognizable in the resulting system or not.

Measures are those requirements that are not directly recognizable in the system. They instead depict threat prevention methods that can be non-technical, procedural, or logical methods against violating a security objective [38, p. 23], [17]. Examples are preventing specific information flows or removing unused software libraries instead of applying expensive defense methods. Measures are typically

general conditions or system-related requirements. Therefore, they are not directly used in the risk treatment step but are rules to verify after risk treatment.

Controls are directly or indirectly recognizable requirements in the system. They accomplish the means of threat reduction and threat tolerance through risk treatment [54, 17], [38, p. 23], [91, p. 7], [105, p. 2]: reduce or detect risks. Risk treatment refines those requirements till they depict specific defense methods in certain variations. Examples are safeguards on all system levels like access control, intrusion detection system (IDS) or secure communication protocols.
The different layers of the controls represent the defense-in-depth onion model: structural, technical, and functional controls (R10.1). Structural controls relate to the overall system structure, e.g., network segmentation. Technical controls defend threats from the internal processing or component side. Functional controls complete the onion model by providing defense on the functional level. Those controls define the behavior of connections to and within the system, e.g., with communication protocols and access controls.

10.2.4. Method categories

The last part of the security requirement taxonomy is the defense method categories. Those categories depend on the development project and are adjustable for every vehicle type in automotive development projects. This adjustment ensures that the categories are up-to-date and complete.
In principle, the categories depict classes of defense methods and allow a more straightforward assignment in the risk treatment process (R10.1, R10.3, R10.4). One option is to align the classes with classical general conditions, e.g., intrusion detection, isolation, and segmentation. Examples of typical classes for automotive development projects show Figure 10.1 without a claim for completeness.
Another possibility is to align the categories on the threat mitigations: prevention, detection, and response [113, p. 235]. Nevertheless, this classification is contrary to the origin of the requirements. Those typically demand categories like intrusion detection or segmentation. Also, this classification is not distinct since defense methods may fall into more than one category, e.g., correcting codes that detect and respond.
The presented taxonomy allows a cross-product of types and method categories. by assigning concrete defense methods (see Table 10.1). This assignment clusters the set of available defense methods into the categories from the taxonomy (R10.2) and annotates them with the type of control. Demands from general conditions either map to the categories or limit the scope of the available methods. The UNECE No. R155 demand for intrusion detection leads during risk treatment to the use of this category. Policies demanding, e.g., only whitelist firewalls, limit the category (in this case, isolation).
Defense methods may be configurable (R10.1). Those variations can be directly included in the catalog or hidden as variants in the method's properties. Encryption

	Implementations		Control Types S	T	F
Isolation	NW Segmentation				
		VLAN	x		x
		Physical	x		
		Firewall	x	x	x
	Host Segmentation				
		CPU		x	x
	Virtualization				
		SOA Domain			
		Hypervisor			
		Sandboxing		x	x
Intrusion Detection	Run-time				
		IDS	x	x	x
		Logging			x
		Runtime Protection		x	x
	Startup				
		Secure Boot		x	x
		Authenticated Boot		x	x
Interface Security	Communication				
		TLS			x
		IPsec			x
		SecOC			x
		ViWi			

Table 10.1.: Classes and Implementations of Defense methods. Columns are the types of controls (S=Structural, T=Technical, F=Functional). Rows indicate the class hierarchy and examples for concrete implementations. "x" indicates that this implementation is usable for this control type.

types are not direct defense methods but are usable by defense methods, e.g., secure communication protocols. They are, therefore, variants of the using method.
Defense methods may have no direct impact on the security objectives. The impact may be emergent only in combination with other controls, e.g., a HSM is only helpful in combination with a defense method that uses the cryptographic algorithms and the secure key storage provided by the HSM. Therefore, those methods are not included in the catalog but are variants of the benefiting defense method. An example would be TLS combined with a present HSM [118]. In this case, there would be at least two variants of TLS, with and without an HSM.
The catalog of defense methods is also subject to change. The catalog must be updated and refined in every development cycle, e.g., every vehicle type. Arising new threats may lead to changes during a development cycle. In this case, the responsible security engineers must be informed about deleted methods and possible substitutions.

10.2.5. Properties of Defense Methods

Defense methods mitigate threats to security objectives. Thereby, they have several different properties, and as stated before, they may have variations [50] e.g., use different encryption mechanisms. Those variations configure the level of restriction a method poses to the system. On the other hand, the system SRA model restricts the applicability of defense methods on behalf of their dependencies. Those variations configure the level of restriction a method poses to the system [66, p. 1]. A structured list of defense method properties enables the building of a defense method catalog, suitable for semi-automatic assignment [38, p. 77].
The property categories incorporate the dependencies onto the system SRA model, the impact on the security objectives, the effect on the RAP and the damage potential, the surface they target, and the implementation overhead and costs. The remaining section briefly describes those categories, while Annex C.1 gives an overview of them, accompanied by example properties without a claim of completeness.

Dependencies limit the applicability of defense methods. They relate to supporting processes needed from the technical side for this method (variant) to work, or the limited scope of a defense method, e.g., virtual local area network (VLAN) technology is only applicable for Ethernet Local Area Network (LAN). They characterize the supporting environmental properties of the ECU and the intended data flow [49].

Overhead and Cost annotations support the requirement assignment and enable an efficient risk treatment solution (R10.4). The overhead represents the resource usage in the case of implementing this method. Examples would be processor clocks per byte [50]. [118] provides different ways to derive the cryptographic algorithms' costs and examples. Cost definitions allow to include risk treatment directly into resource scheduling procedures [50] which is out of scope in this work.

Impact information is necessary to assign defense methods suitable for the damage scenarios security objective. [91, p. 522] suggests a range from *-2* to *+2* for each security objectives. The positive part depicts the advantages of the defense method on security objectives, while the negative part illustrates that defense methods may negatively influence certain security objectives. A rating with *+2* means that the defense method is a primary candidate for mitigating threats against this security objective. Rating a security objective with *1* marks it as a secondary candidate since they do not provide the best possible mitigation. The negative ratings are the dual of the positive ones.
Different evaluation techniques are possible for them, e.g., the strength of the cryptographic algorithm or the quality of an authentication mechanism [49].
Assigning such ratings for each security objective provides a qualitative rating of the advantages and disadvantages of the defense methods.

The Surface limits the applicability regarding the attack surface, which this defense method mitigates. This property is necessary since some defense methods mitigate only local attacks, e.g., physically disabling a JTAG interface.

Effect denotes a defense method's influence on the attack potential of specific threats. This step reuses the method from the risk analysis, for the attack's potential impact may include the influence on the needed time, knowledge, tools, expertise, and access of the attacker.

10.2.6. Defense Method Catalog

During security planning, one step is preparing the defense method catalog for risk treatment. This catalog is a cross-product of the defense method categories and properties to build an instantiable defense method catalog. Therefore, the security engineers assign the functional variants and properties to each defense method.
Table 10.2 provides an example for securing JTAG. One variant physically disables JTAG, e.g., through blowing fuses. This variant is the most secure one but highly influences the system's availability since it removes the interface. Another variant by utilizing cryptography, e.g., Secure Sockets Layer (SSL) [75] which has a compared high effort since it needs TLS support in the system and has a higher resource usage.

10.3. Assignment of Security Requirements

The goal of risk treatment is twofold: Minimizing the risk, which means reducing the costs of non-implementation of defense methods, and minimizing the effort by reducing the costs of implementing defense methods.
General conditions are relevant for the type of approval or necessary to fulfill OEM demands. Therefore, their costs for not-implementation are infinite. On the other hand, the costs for implementing general conditions are most of the time variable. They are typically related to categories of defense methods. Therefore, the implementation costs depend on the chosen method of the respective category. For system-related requirements, it is necessary to trade the costs of not-implementation against the implementation costs R10.3.

	Property	Variants	
		Physical (Blow fuses)	Cryptographic (Typically available procedures)
	Technology	JTAG	JTAG
	Costs	0	?
Impact	Confidentiality	+2	+1
Impact	Integrity	+2	+1
Impact	Availability	-2	-1
	Surface	Local	Local
	Dependencies	-	SSL
	Effect	RAP: ?; Impact: ?	RAP: ?; Impact: ?

Table 10.2.: Example of defense method properties for securing the JTAG port without claim for completeness. The values for cost, impact, and effect are variables to assign based on the given setup.

Also, if risk treatment cannot consider implementation costs directly, it needs to consider those cost-related problems, at least indirectly. This observance is possible by reducing the overall number of defense methods through structured method assignment, e.g., by first cutting attack paths before treating the risk. Another possibility is to assign methods that impact several security objectives at once instead of using a method per security objective.
The remaining section proceeds through the risk treatment process. Because of the already discussed problems with acquiring the costs of defense methods, this work counts for efficiency (R10.4) through a structured method assignment. It illustrates possible trade-offs and optimization points (R10.3).

10.3.1. Prerequisites

According to ISO/SAE 21434 [54] and UNECE No. R155 [85] risk treatment relates to identified risks and attack paths of the SRA. Concrete, those risks whose impact is against the road user. While it is possible to consider other impact categories, this work concentrates on those risks (R10.4). The treatment of other risks is subject to by-catch of assigned methods and trade-offs (R10.3).

10.3.2. Strategy

In general, risk treatment should be done holistically over the complete system. A global approach demands formal constraints for all defense methods and a distinct format for the input data. The latter is a big problem, especially in distributed development environments like the automotive industry. Also, tools for risk analysis allow different levels of abstraction and typically use natural language to formulate the damage scenarios and threats. Therefore, the bigger the input space for the risk treatment, the more significant the divergence between the input data. This divergence is also because the automotive functionalities and components are diverse. The vehicle combines service-oriented functionalities with hard real-time

classic development strands. This problem is minor on a lower level of abstraction, e.g., on the component level. Single components typically combine functions from similar development strands and complexity. Therefore, their analysis results are less diverse. Nevertheless, a global brute force approach would lead to a state-space explosion due to the nature of such NP problems.
Another strategy could be to assign all possible defense methods on the component level to minimize the impact of the threats. On the one hand, less can depend more on the defense methods [41]. Assigning more defense methods leads to higher costs and might lead to new risks to the system. Also, automotive systems are embedded systems with limited resources. Therefore, an overestimation of defense methods might leave the system infeasible.
An even lower level of abstraction would be assigning the defense methods in the decomposed system to each function individually. This strategy has the lowest probability of a state-space explosion and is accomplishable by brute force leading to an optimal solution for the different functions. On the other hand, a complete individual risk treatment might lead to various defense methods deployed to one component and its functionalities. The result is high costs for implementing defense methods and high resource usage, which might leave the system infeasible.
A good strategy for efficient and cost-effective risk treatment is a component-level heuristic approach (R10.1) – the defense methods assignments basis is prioritizing the threats on the system. A by-catch test reveals a positive impact of the defense method on other components or functions.

Possible Heuristics for risk treatment provides related work. [96] presents the straightforward idea of the highest risk first. Risk is a combination of the damage potential and the required attack potential of the attack path and allows a global ranking of the relative priority. Therefore, a risk-based heuristic is coarse-grained and does not allow to, e.g., prefer certain types of damage potentials.
In [105, p. 12] the suggestion is a prioritization according to the highest impact or the highest likelihood first. Such a prioritization enables a ranking of risks according to the impact regarding their damage or the attack potential. Depending on the level of detail of the input model, this approach allows even heuristics with several layers, e.g., highest safety impact first.
A prominent approach for risk treatment are the defense-in-depth layers [105, p. 12]. The idea is to assign defense methods from different types onto the system to have an onion-like security defense. While this approach leads to the stated cost and state-space problems when done globally, the idea is usable component-globally and heuristically by iterating through the defense method types.
Another idea is to group security requirements according to viewpoints [118], e.g., according to the user's view on the system. This approach is highly dependent on the input data model. Therefore, it must be part of a method suite where the risk analysis produces the required output. If so, the approach allows, in principle, the same heuristics as highest impact/likelihood first.

The sources do not prioritize one element attacks paths. That *single risks* are either single points of failure (using the reliability language) or preparation attacks. The first case is essential to solving since these attack paths have only one step to accomplish the attack. For the latter case, prioritizing single risks mitigate parts of several attack paths. This prioritization reduces the overall effort for risk treatment.

Heuristic Layers that prioritizes single risks before tackling the highest impact is the approach used in this work. Also, the approach uses the defense-in-depth idea to assign methods from as many control types as possible to result in a layered security defense model.
The prioritization allows the extension to varying levels of detail (R10.3). Possible additional layers include specific impacts, e.g., safety, likelihood impacts, certain attack surfaces, time, or knowledge level.

10.3.3. Input Data Composition

Risk treatment aims to secure the risks analyzed before. Therefore, the primary input for the risk treatment procedure is the resulting attack paths and assigned risks from the system SRA.
According to ISO/SAE 21434, risk treatment excludes those risks whether the risk is below a defined threshold. This approach reflects that critical assets must have the highest possible security. However, items where the implementation costs for defense methods are above the unlikely impact can stay unsecured [38, p. 77]. On the other hand, it is reasonable to secure all risks with the assumption that someone might also launch unlikely attacks with a low impact.
In practice, there is typically a threshold for the risk level below which the functions drop out of the risk treatment process. Nevertheless, defense method assignment on the complete set of risks enables tracing the by-catch of excluded risks and evaluating their resulting risk level. This approach might reveal that mitigating even those risks onto a deficient level or entirely is possible and by that follow the rule that someone will carry out every threat [96]. Therefore, the method assignment in our approach takes place on the relevant risks, while a final by-catch test allows a complete overview of the remaining risk levels.
Due to the timeline of the development process, there may be already assigned defense methods, especially on a structural level. This information is critical for the risk treatment process. Otherwise, the semi-automatic assignment might deploy already-used defense methods.

10.3.4. Approach

Following the presented taxonomy, the approach is twofold. The first step is regarding the general conditions and therefore demands necessary to fulfill (R10.2). The second step accomplishes the system-related requirements. The remaining section elaborates on the two assignment steps. In order to support the comprehension of the system-related requirements assignment, this part provides the sketch for an

Algorithm 4 Handle risks against road users with system-related requirements

Require:
1: risks **as array** of threat × attack_path × Item_element × applied_means ▷ at the start, for all risks: applied_methods can be = ∅
2: criticality_threshold
3: means ▷ all available means to possibly increase security
Ensure: $\mathbb{R}_{rel} = \emptyset$ ▷ relevant risks

4: $\mathbb{R}_{rel}$ ← all elements r of risk with CRITICALITY(r)>criticality_threshold
5: **while** risk∈ $\mathbb{R}_{rel}$ with empty attack path exists **do**
6: ASSIGN($\mathbb{R}_{rel}$,current_risk, criticality_threshold, means)
7: $\mathbb{R}_{rel}$ ← all elements r of risk with CRITICALITY(r)>criticality_threshold
8: **end while**
9: **repeat**
10: current_risk ← element of $\mathbb{R}_{rel}$ with maximal DAMAGE_POTENTIAL(current_risk)
11: ASSIGN($\mathbb{R}_{rel}$,current_risk, criticality_threshold, means)
12: $\mathbb{R}_{rel}$ ← all elements r of risk with CRITICALITY(r)>criticality_threshold
13: **until** $R_{rel} = \emptyset$

assignment algorithm. This algorithm does not include every step in detail but provides an overview of the idea.

General Conditions are security requirements that apply to all security-relevant items. Therefore, they need no heuristic approach. They can be applied by hand or by formally defining the rules for assignment, e.g., in a logical language.
General conditions may lead to assigning defense method categories (e.g., network segmentation, intrusion detection) or detailed methods (e.g., whitelist firewall on each Ethernet node). Typically, the general conditions resulting from normative references match with organizational policies. Therefore it is possible to define rules and concrete defense methods to assign to fulfill the general conditions demands.

System-related requirements - For the system-related defense methods, algorithm 4 uses a set of *risks* as input. Each risk (line 1) is a set consisting of the *threat*, the *attack path*, the relation to the *item element* and applied defense methods (*applied_means*). The set *applied_means* may not be empty at the beginning reflecting already applied defense methods, e.g., structural network segmentation.
Risk value determination provides function CRITICALITY(risk) (Algorithm 6) which combines the impact functions $DAMAGE\_POTENTIAL(risk)$ and $REQUIRED\_ATTACK\_POTENTIAL(risk)$. The two functions provide the impact level using the risk matrix (see 7.1) and the feasibility test for the defense methods. The feasibility test uses the defense method properties to test if this defense method is usable for this risk under consideration. Please note that the algorithm does not define those functions in detail.
The idea of Algorithm 4 is to assign defense methods to the set of relevant risks $\mathbb{R}_{rel}$ as long as their risk value is above the defined threshold *criticality_threshold* (e.g., low).

Algorithm 5 Assign defense methods to the currently processed risk.

```
1: procedure ASSIGN(ref ℝ, ref current_risk, criticality_threshold, means)
2:     const means_category ← [structural, technical, functional]
3:     i ← 0
4:     fail ← 0
5:     while CRITICALITY(current_risk) ≥ criticality_threshold & fail ≠ 3 do
6:         highest_impact ← 0
7:         highest_impact_mean ← none
8:         for all dm ∈ means do                          ▷ find mean with highest impact
9:             if CATEGORY(dm)=means_category[i] & dm ∉ current_risk.applied_means then
10:                if CRITICALITY(current_risk with dm applied) > highest_impact then
11:                    highest_impact_mean ← dm
12:                    highest_impact ← CRITICALITY(current_risk with dm applied)
13:                end if
14:            end if
15:        end for
16:        if highest_impact_mean ≠ none then
17:            Apply highest_impact_mean to current_risk
18:            Apply highest_impact_mean to all elements of ℝ where it can be applied
19:        else
20:            fail ← fail +1
21:        end if
22:        i ← (i + 1) mod 3
23:    end while
24:    if fail = 3 & CRITICALITY(current_risk) ≥ criticality_threshold then
25:        Deal otherwise with current_risk
26:        ℝ ← ℝ − current_risk
27:    end if
28: end procedure
```

Algorithm 6 Determine updated risk from damage potential and required attack potential.

```
1: function CRITICALITY(risk)
2:     return DAMAGE_POTENTIAL(risk)·REQUIRED_ATTACK_POTENTIAL(risk)
3: end function
```

The first loop (lines 5-8) evaluates all risks without an assigned attack path. Those are SPOF or preparation attacks. After their mitigation, the second loop evaluates the remaining risks in descending order of their impact (lines 10-13).
Function *ASSIGN* (algorithm 5) shows the assignment of defense methods. As long as the risk under consideration is not below the threshold, it tries to find a defense method dm and assigns it to the risk (loop line 5-23). Through the impact functions (lines 10-13), the algorithm tests whether the feasibility and properties of the method fit the risk. Therefore, it also takes dependencies and technology into account. This step assigns methods with the highest impact on the risk value. Other heuristic layers are easy to integrate and lead to new iteration conditions.
To find a defense-in-depth solution, which means using all control categories, the algorithm iterates over the structural, technical, and functional defense methods (lines 5, 20, 22) until the risk is mitigated or no possible method is available ($fail = 3$). Other possibilities would be to use as many methods as possible from each category or until the impact is below a defined threshold. Both possibilities have a high probability of one-sided methods, which is against the defense-in-depth onion model.
If a risk is impossible to mitigate, the threat remains in the resulting risk value (line 24). Depending on the type of damage (e.g., Safety vs. Financial), other possibilities for mitigation are necessary, e.g., change deployment, or add isolation. Therefore, the algorithm leaves this risk for dealing otherwise with it and excludes the risk from the set $\mathbb{R}_{rel}$. This problem should be a corner case since the defense method catalogs provide a variety of possibilities. Nevertheless, such situations are also possible in non-automatic risk treatment procedures.
After each successful assignment, the function tries to assign the defense method to other risks (lines 16-18). The assignment test in the impact functions adheres (depending on the defense method) to the components' units of isolation. For example, a run-time protection mechanism on a virtual machine applies only to functions on the virtual machine. This component global assignment applies the by-catch test not only on attack paths but also on the component level.
Before mitigating the subsequent risk, the algorithm updates the relevant risks $\mathbb{R}_{rel}$. This update excludes the currently mitigated risk and those mitigated with the by-catch-test from the set.

Further Trade-offs (R10.3) - Currently, the algorithm does not include cost information. Those are especially helpful if defense methods with equal impact can assign [50]. The algorithm currently uses the first method found and assigns it to the threat. Without considering costs [91, p. 507], other defense methods could be annotated as possible substitutions, leaving the option for the security engineer to swap them.

Defense methods may have several variants with different impacts. The algorithm uses the variant with the highest impact. Therefore, one trade-off point is to mark those variant points and annotate the different impacts. The security engineer validates the results and adjusts the method variant to the preferable one. This work follows the approach to have a more secure variant than less instead.

10.3.5. Open Issues

The costs for the introduced defense methods should be less than the damage scenarios impact. Otherwise, the effort is not reasonable [38, p. 26]. The presented method and the taxonomy incorporate cost factors. Also, in the algorithm, cost information is easily includable. Nevertheless, as already stated in the taxonomy, the derivation of such information is not easily accomplishable, and there is no information for the presented defense methods available. Therefore, this is an open issue.

Another issue is the possibility of misjudgment. Every (semi-) automatic assignment has [38, p. 75]. In order to prevent misjudgment by the algorithm, an experienced security engineer must review the algorithm results.

The re-opening of risks is currently left out of the algorithm if the algorithm assigns a defense method with a negative influence to a security objective. The same counts for assignments that negatively influence the systems functionality [38, p. 75], e.g., through excessive resource usage.

As already included in the algorithm break statements, there might be a case where no suitable defense methods are assignable [118, p. 23]. Examples are, e.g., protecting damage scenarios regarding availability against local attackers. Those attackers, especially car owners, have unlimited access to the vehicle and can physically break the communication. In such cases, the security engineer might be able to find creative solutions. Otherwise, other threat mitigation techniques or risk acceptance are the only possibilities to leverage such risks.

Part IV.

Use Cases and Evaluation

Measure what is measurable, and make measurable what is not so.

Galileo Galilei

11 Evaluation Criteria

Every development process needs an evaluation regarding defined criteria. The basic criteria for evaluating the security development process presented in the previous chapters derive from the capability maturity model.

11.1. Automotive SPICE

Typically, the evaluation of development processes uses capability maturity models. Those models can identify improvement activities and measure the process alignment to the defined process model [95, p. 12].

Automotive development uses Automotive Software Process Improvement and Capability Determination (Automotive SPICE) as capability maturity model. The model divides the process alignment into six levels [112]:

0: Incomplete - The process is not implemented or working.

1: Performed - The process is implemented and functioning.

2: Managed - The process is implemented and managed.

3: Established - The process is working in the desired fashion.

4: Predictable - The process allows quantitative evaluation and alignment.

5: Innovating - The process aligns to improvements and change.

The development process design should at least fulfill the demands of level 2 of the process assessment model. This demand leads to requirements for the definition and implementation of the process [95, p. 10].
A key aspect of Automotive SPICE is continuous process monitoring and improvement. The idea is to analyze the development process to identify issues continually. Those are subject to identifying improvement goals and process changes. The next step implements and evaluates the changes [112]. This work identified several issues and suggested process adjustments. Following the Automotive SPICE process, those are subject to evaluation according to improvement goals.

11.2. Improvement Goals

Automotive SPICE does not clearly define how improvement goals and resulting evaluation criteria for process improvement look like. During accompanying the security development process several issues revealed. Those lead to the following criteria for evaluating the suggested changes:

EC1 Time efficiency in the security department

EC2 Acceptance in the development departments
- Time efficiency
- Low repetition of similar process steps
- Reduction of inquiries for clarification
- Provision of information about necessary inputs for the process steps

EC3 Process implementation
- Tracing decisions
- Explicit formulation of uncertainties
- Alignment to ISO/SAE 21434 and UNECE No. R155

In the first development cycle, according to the evaluated security development process, the most significant drawback was the high effort for the development process steps in the security department. Independent of the effort needed for each process step, the process steps themselves have the potential for more time efficiency. Therefore, each process improvement needs to lower the time and effort spent [27]. Also, accepting the security development process in the development departments is subject to improvement. Each development department is responsible for various

items, and each item participates in the security development process to a certain extent. The main work area of the development departments is the development of the next vehicle. Processes for accomplishing meta-functional aspects like safety and security are often seen as extraordinary charges since they are no selling argument. Therefore, the more time the responsible persons spend to accomplish meta-functional analysis, the less acceptance they have for those process steps.
Currently, the development teams and the security department repeat the same analysis with just a little difference between the targeted items. This approach is time inefficient, annoying, and lowers the acceptance. Therefore, improvements in lowering the repetition rate support the time efficiency in the security department and the acceptance in the development departments.
A clear structuring and distinct formulation of the process templates helps avoid inquiries for clarification. Providing information about necessary information, e.g., input documents for each process step, lower the additional effort in the development departments. That information enables the development departments to prepare for the security development process steps leading to a higher support [100]. This information lowers the rate of additional effort for accomplishing this information.
The existing process structure is suitable to cover the security development process. Nevertheless, existing drawbacks reveal evaluation criteria for the suggested improvements of the process and tools. In the evaluated form, the process requires many manual steps. These manual steps are error-prone for the tracing of decisions and changes. Therefore, the process improvements need to support automation for data exchange between the tools [27].
Parts of the security development process take place before serial development. At this time, there is a lack for detail information about the items. The security development process must adhere to this and should be able to explicitly state uncertainties. Making uncertainties explicit supports the tracing of decisions and changes. Additionally, it enables the security development process for structured recaps of analysis steps as early as possible.
ISO/SAE 21434 and UNECE No. R155 define the security development process steps. Therefore, the goal of each improvement in the process is to keep or raise the alignment with the standards. This goal adheres that the security development process is relatively new. It literally is fledgling. Therefore, improvements may raise the alignment with the standards. Other improvements need to keep the alignment, meaning they do not contravene to requirements of the normative references.
The stated improvement goals are qualitative. Therefore, the use case and evaluation chapters qualitatively show the goals' fulfillment. Also, some process steps are challenging to accomplish without all the input information. Suggested improvements may make it necessary to repeat the same analysis with the same item to show the improvement rate. This repetition is impossible for some steps. The following chapters try to overcome these issues as far as possible.

We can easily forgive a child who is afraid of the dark; the real tragedy of life is when men are afraid of the light.

Plato

12 Use Case: Security Relevance Evaluation

Chapter 7 discussed several issues and improvements of the SRE. This chapter presents a use case showing the SRE method in the adjusted version. Based on the estimated effort for accomplishing SRE in the last development cycle, the evaluation section discusses how the improvements accomplish the stated evaluation criteria. The evaluation uses light functions. Other use cases for the *occupant and pedestrian domain* and *advanced driver assistant functions* showed similar results.

12.1. Light Functions

The 49 light functions of the use case vary, e.g., between interior and exterior light. Those functions situate in one department, making using the filter for the development teams necessary. This filtering results in 2 clusters. The smaller one has 13 light functions with one contact person. The bigger cluster shows 36 light functions divided into six contact persons. Those contact persons are responsible for one, up to 24 light functions. Figure 12.1 shows the division onto the clustering. Table 12.1 shows the resulting criticalities for the biggest cluster of 24 items. In summary, four items have no damage potential and are therefore not security-relevant. Nine items have a very low security relevance. Depending on the company's risk appetite, they are subject to risk retaining and dropping out of the security development process. The other 11 items have a low security relevance. They are likely subject to the subsequent security process and proceed into the SRA.

Item	Safety	Finance	L & R	Quality	Exposure	Criticality	
1332	0	0	0	0	1	0	None
1334	0	0	0	0	1	0	None
1336	0	0	0	0	1	0	None
1355	0	0	3	0	1	1	Very Low
1358	0	0	3	0	1	1	Very Low
1360	0	0	3	0	1	1	Very Low
1384	2	0	3	0	2	2	Low
1386	2	0	0	0	1	1	Very Low
1388	0	0	3	0	1	1	Very Low
1392	2	0	2	0	2	2	Low
1394	2	0	2	0	2	2	Low
1396	2	0	2	0	2	2	Low
1398	0	0	2	0	2	2	Low
1400	0	0	3	0	1	1	Very Low
1404	0	0	0	0	1	0	None
1406	0	0	2	0	2	2	Low
1408	0	0	2	0	2	2	Low
1410	0	0	3	0	1	1	Very Low
1412	0	0	3	0	1	1	Very Low
1414	0	0	3	0	2	2	Low
1416	1	0	2	0	2	2	Low
1418	0	0	3	0	1	1	Very Low
1420	0	0	2	0	2	2	Low
1422	0	0	2	0	2	2	Low

Table 12.1.: Results for the cluster SRE of the standard light functions.

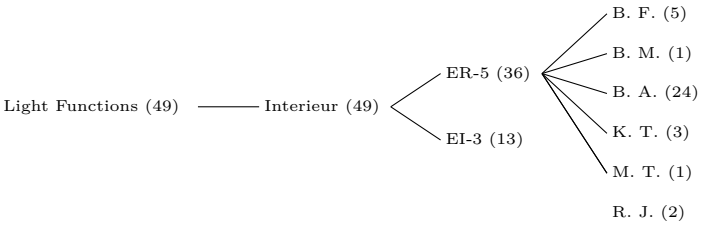

Figure 12.1.: Resulting clustering for the SRE Use Case in the light domain. This domain situates in the interior department. Two development teams have light functions, one with six reference persons.

12.2. Improvement Evaluation

The presented use case already shows good results for the intended improvements. The cluster technique reduces the number of SREs from 49 in the light domain to seven. In the first run, it took about one hour for the security department to accomplish a SRE together with the development department. These are 49 hours for the light domain. Due to the clustering, the effort reduces to seven hours. This is approximately 85 % *less time effort*. Other use cases show similar results. For example, the time effort in the occupant and pedestrian domain reduces by 63.83 % (EC1, EC2).

Without evaluation of all domains and possible clusters, it is assumable that the time effort reduces significantly. In the previous run, more than 400 SRE took place. The presented improvements reduce the effort at least by 50 %, which is more than 200 hours. Also, the template enables the responsible persons to act autonomously in the SRE. In combination with the evaluation script, this drastically reduces the time and effort in the security department (EC1).

For the development departments, the rise in the *time efficiency* is the same (EC2). Deleting the relevance questions from the template enhances the time effort for irrelevant functions. For relevant functions, the effort lowers. Due to the clustering, it is unlikely that a complete cluster is irrelevant. Only in this case does the effort for the SRE raise. In all other cases, the deletion of the duplicate questions lowers the effort.

Also, the clustering reduces the *repetition rate of similar process steps* (EC2). The responsible person must fill in only one template instead of 24 in the use case example. This approach also reduces the *effort in case of clarification needs*. The responsible person prepares the SRE for all items in a row as far as possible. Arising questions are clarifiable at once. The presented template is used for autonomous filling by the responsible persons. Therefore, they can acquire all *necessary input data* before they start to accomplish the SRE. Overall, the presented improvements are suitable to raise the acceptance in the development teams.

Within the clusters, the results show similarities. In the light cluster, three different results arise. Those arrange in a non-relevant cluster, one with low criticality and one with moderate criticality. The new template allows to identify those clusters directly and reuse them in the SRA (EC3). This identification makes *decisions* for tailoring subsequent process steps more obvious. The automatic evaluation possibilities allow automatizing also the feedback into the tools for modeling and *tracing* the development process (EC3). This method is less error-prone than manual maintenance of the data basis.
The template gives the possibility to mark a damage potential question as irrelevant (EC3). By that, items may be non-relevant for security. Therefore, not only the questions in the template but also the results *align with ISO/SAE 21434 and UNECE No. R155*. The question's answer regarding lack of knowledge about the correct answer allows the explicit formulation of *uncertainties*. Combined with suitable tools, the follow-up process is automatable and, therefore, less error-prone. Table 12.2 provides an overview of the achieved improvements.

Requirement	**Goal**	**Current Process**	**Improvement**
EC1/EC2	Time Efficiency	49 h = 100%	7 h = -85%
EC2	Development Department		
	Repetition Rate	24 SREs	1 SRE
	Inquiries	up to 24 meetings	approx. up to 1 meeting
	Input information	Questionnaire with duplicates and deeply nested levels	Clear questionnaire, autonomous fill-out possible
EC3	Process Implementation		
	Tracing decisions	-	Automatic evaluation
	Formulation of uncertainties	not possible	explicitly possible
	Alignment with Norms	All items are relevant	Not relevant is possible

Table 12.2.: Overview about the achievements regarding the improvement goals in the use case.

Only a fool doesn't experiment
Charles Darwin

13 Use Case: Function-oriented Security Risk Analysis

The SRA chapter discusses the MoRA method and its embedding in the security development process. From the process evaluation, several improvements arise, which are subject to evaluation in this chapter. Therefore, this chapter continues the standard light use case from Chapter 12 followed by an evaluation of the improvement goals.

13.1. Light Functions

In the pre-evaluation (SRE) 11 functions resulted in a moderate criticality. Those proceed in the security development process and are subject to the SRA.
Starting with the item definition, the SRA incorporates all functions whereby the "emergency braking lights" function has a relation to the "braking lights" function. Therefore, if the *braking lights* function sustains damage due to security issues, the *emergency braking lights* function is likely to have similar issues. The 11 functions communicate 16 data items, resulting in 61 data flows (different receivers and transmission paths). The standard procedure already considers the deployment if it is known. In the case of the standard light functions, they incorporate 21 hardware components consisting of standard ECUs, high integration ECUs but also buttons (e.g., emergency light) and the power supply themselves. As for the assumptions, two limit the scope according to the environment, and two relate to the item.

The next step in the SRA is to evaluate the assets according to security objectives and damage potential. The use cases template is an early version that defines no damage scenarios but only security goals. Therefore, the asset and the security objective are subject to impact rating.
This use case does not provide the differentiation between damage criteria against the road user and the OEM as suggested in Section 8.4. Therefore, the author did this differentiation manually, and the results intention is not to be exhaustive. The safety damage always relates to the road user (RU). Therefore, in this case, the differentiation is unproblematic. Also, the use case does not designate damages in the law category. For the financial and the quality category, the differentiation has more issues. In the case of a very low impact for one stakeholder, the other cannot have a higher impact level. This result is due to the maximum approach for the impact on the stakeholders. If the impact is higher rated, it might be that the other stakeholder also has a lower impact in this category. In this case, the use case loses an impact rating. Nevertheless, this is not crucial for evaluating the improvements.
MoRA supports the analyst by providing a security goal template. This template is a permutation of security objectives, functions, and data. Thereby, it is likely that the template incorporates irrelevant security goal suggestions. In this use case, no relevant security goal with an impact on confidentiality exists. Therefore, this security objective is not relevant for standard lights. The overall result of the 143 security goals is as follows:

- 46 according to confidentiality which is irrelevant
- 32 integrity damage scenarios where 11 are irrelevant
- 48 for availability where 11 are irrelevant
- the rest are authenticity damage scenarios which are no primary goal and therefore not in the scope of this thesis

The threat scenario evaluation reveals 25 threats from nine threat classes with an attack potential: low (5), moderate (6), high (13), and beyond high (1). This version of MoRA does not depict preparation attacks as separate risks but only complete attack paths. Therefore, the resulting risk assessment has 18 attack paths with associated risk evaluations. Those attack paths evaluate low (4) and moderate (14) risks. The remaining seven threats are preparation attacks.

13.2. Improvement Evaluation

The cluster SRA for the standard light functions needed four meetings between the security department, the development team, and the analysts. This number is one more than the basic structure: one inaugural meeting, one intermediate meeting, and one final presentation meeting. Nevertheless, the experience reveals that often one more meeting is necessary. Therefore, the meeting effort in the development team and the security department does not change in the cluster approach but is much lower than with single SRAs (EC1, EC2). If the 11 functions had single SRAs 33 up to 44 meetings would have been necessary.
The item definition gets bigger through the clustering since more ECUs are in the analysis scope. On the other hand, high integration ECUs tends to have functions

playing a central coordination role, for example, for maintaining the front or rear lights. Those would be in the scope of all functional SRAs connected to this central function. Therefore, the item definition grows, but the effort for repetition lowers (EC2). The same is true also for the other evaluated use cases.
The damage scenarios and security goals grow primarily because they incorporate the assignment of security objectives, leading to many similar entries. This issue is subject to further evaluation if it is suitable to divide the damage scenarios and attack goals and assign the item ID. The damage scenarios would be abstract but linked to the ID instead of directly incorporating the item name. This adjustment also simplifies tracing through the MoRA and parsing of results into other tools (EC3). The current version leads to similar damage scenarios that differentiate because of the item name. Nevertheless, this needs an adjustment and evaluation of the MoRA method, which is left for future work and discussion with the developers.
In contrast to the damage scenarios, the threat scenarios do not grow much, resulting from the different template structures. The threat scenario formulation is rather abstract and links to the security goals. By that, it is possible to describe a threat related to security goals from several functions – the same counts for the risks.
Overall, the effort for accomplishing the cluster SRA is not much different than for a single SRA. Since there are no comparable numbers, the estimate in Table 13.1 is one quarter higher to incorporate the preparation of information for the cluster items. The remaining time effort stays the same, since there are no extra meetings for the cluster. Besides, the effort of the security analyst is not of primary concern since the analysis is subject to outsourcing.
In the development departments, the time efficiency rises with the clustering (EC2). Also, the repetition rate lowers due to the lower number of analyses. The core information necessary for the process steps differs between the functions. Nevertheless, due to the central coordinator functions and the aim for high integration, much information is similar between the single SRA. This similarity reduces the effort to provide the information. Especially the general information and introduction for the overview of the functions do not repeat, which raises the acceptance in the development departments and lowers their time.
The evaluated cluster SRA does not include an extra identifier for the different cluster items. Therefore, tracing is only indirectly possible through backtracking from the risks to the function in the item definition (EC3). The MoRA template provides a separate overview of the risks per security goal. This overview is usable to identify the risks for each function. Nevertheless, it provides only the risk numbers and the resulting risk but not the other relevant information. Therefore, including the suggested id in the analysis to trace the original item of the damage scenarios, threats and risks make the analysis more comprehensive.
Table 12.2 provides an overview of the achieved improvements.

Requirement	Goal	Current Process	Improvement
EC1/EC2	Time Efficiency	approx. 16 h per function	approx. 20 h per cluster
EC2	Development Department		
	Repetition Rate	11 function SRAs	1 function-oriented SRA
	Inquiries	approx. 3-4 meetings per function	approx. up to 4 meetings per cluster
EC3	Process Implementation		
	Tracing decisions	only inside MoRA	
	Formulation of uncertainties	through open issues	through open issues
	Alignment with Norms	overestimation of risks	division of risks between road user and OEM

Table 13.1.: Overview about the achievements regarding the improvement goals in the use case.

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I've found it!), but 'That's funny...'

Isaac Asimov

14 Use Case: System Security Risk Analysis

The chapter about the system SRA step (Chapter 9) differed from the preceding ones since this is a development step currently not implemented in the process under consideration.
Therefore, also the use case chapter differs slightly. There is no primary use case that is adjustable to improvements. Instead, it is necessary to build it from the ground up. Due to NDA and information source reasons, it is impossible to provide the complete content and results. Nevertheless, the aim is to provide sufficient insights into the use case to evaluate the achievements.

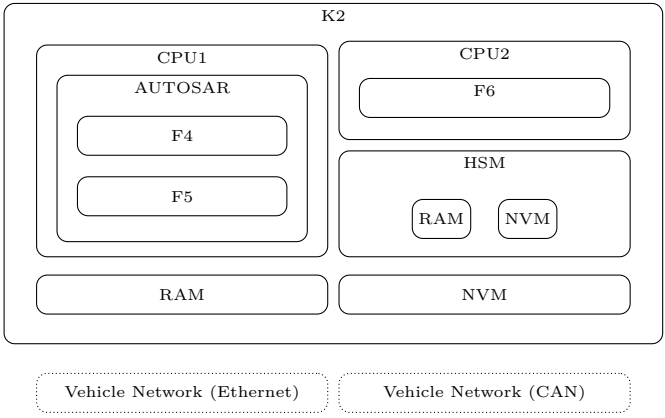

Figure 14.1.: Example architecture for the use case: A high-integration ECU with two CPUs and an Ethernet and CAN interface.

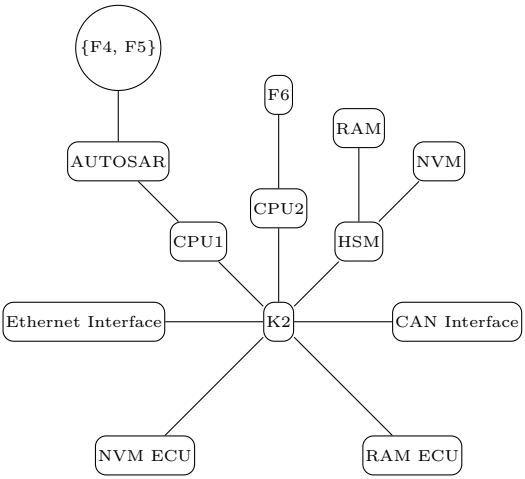

Figure 14.2.: Architecture graph for the example architecture of Figure 9.2. Circles represent functions and system software. Rounded rectangles represent hardware structural elements.

14.1. Light Functions

The evaluated function-oriented cluster SRA of the light functions serves as a basis for the system SRA use case. In order to keep the use case comprehensive, the system SRA focuses on one component with deployed light functions, keeping other deployed functions and other components aside. The presented models are much bigger in reality since they incorporate the overall system architecture and every communication path.
From the light use case, the deployment assigns three of the functions to the same ECU. Figure 14.1 shows the hardware structure of this ECU consisting of two CPUs. One of them incorporates an AUTomotive Open System ARchitecture (AUTOSAR) operating system with two assigned functions. The third function runs on the other CPU natively. Furthermore, the CPU has an HSM with its RAM and NVM, an Ethernet and a CAN interface, and storage facilities (NVM, RAM). Figure 14.2 provides the according architecture graph for the hardware structure.
In order to further narrow the use case, only one function is in scope ($F6$), which runs natively on the second CPU. This function has four communication partners. Two are the functions deployed to $CPU1$ ($F4$, $F5$). The other run on other ECUs. Therefore, it is necessary to communicate via the CAN interface to realize those communication links. Figure 14.3 provides the according to communication flow graph, which has labels for the communication interfaces for more comprehensiveness.
Analyzing the system SRA models requires a graph embedding to derive the architecture's communication paths for each risk. The next step of the analysis demands the propagation of the RAP category levels through the path. For this, in case of contradicting assignments, the propagation uses the minimum levels. The following environment provides the communication paths with the resulting RAP level and the evaluation.

$$
\begin{aligned}
D.15, D.16 &: F6 - CPU2 - K2 - CPU1 - AUTOSAR - F4 \rightarrow \text{very low} \\
D.1, D.2, D.8, D.10 &: F6 - CPU2 - K2 - CPU1 - AUTOSAR - F5 \rightarrow \text{very low} \\
D.5 &: F6 - CPU2 - K2 - CAN(F14) \rightarrow \text{low} \\
D.7 &: F6 - CPU2 - K2 - CAN(F17) \rightarrow \text{very low}
\end{aligned}
$$

In the end, the analysis updates the according risks. This means that the risks for $F6, F4, F14$ and $F17$ gets the minimum of the resulting RAP values. For $F6$ and $F5$ this would be $31043 = 1 =$ very low. For $F4$ and $F17$ it is $31143 = 1 =$ very low and for $F14$: $61143 = 2 =$ low. This change triggers a re-evaluation of the risk values for these functions. The result is an updated risk value that incorporates the functional and the system influence.

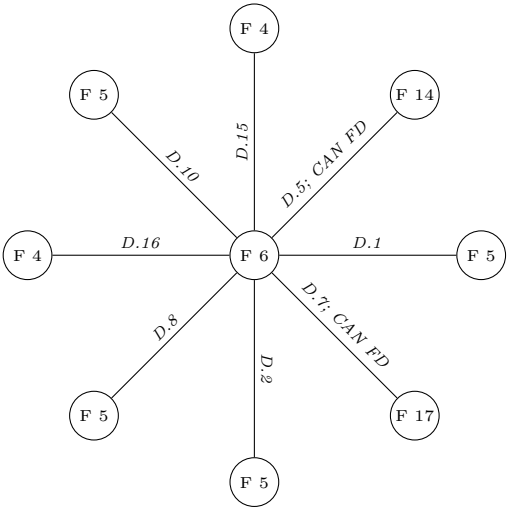

Figure 14.3.: Communication flow graph for the use case architecture.

14.2. Method Evaluation

The evaluation criteria in Chapter 11 focus on process improvements. In the case of the system SRA, some of them are inapplicable. This is especially true for those who target time efficiency and acceptance raising. Those need a comparison between the former process and the improvements. Nevertheless, this section tries to target all evaluation criteria as far as possible, incorporating the general requirements to the process step (see Section 9.1).

The developed system SRA method includes the function-oriented SRA results and supplements them by extending the attack paths (R9.1). Due to the spreadsheet character of the function-oriented SRA it is possible to parse the results automatically into the system SRA (R9.3). This approach raises time efficiency and the repetition rate for information acquisition (EC1, EC2).

During the analysis, the graph embedding method is efficient regarding the analysis of attack paths. Also the minimum computation of the RAP supports an efficient reevaluation of the RAP and therefore the risks (R9.2). The results from the analysis are usable for tracing decisions (EC3). During the analysis, the method evaluates the communication paths with the according RAP. Supporting this approach by a tools allows varying outputs, usable in versioning systems and ticket systems to monitor the decision process.

On the other hand, the method is able to use inputs from architectural modeling tools. Therefore, it is possible to directly include those information and maintain tracing of input data changes (R9.4).
The system SRA method aligns with the normative references since it re-uses the results from the function-oriented SRA and supplements them if necessary (R9.6). These extensions include the attack paths, the attack feasibility, and, therefore also, the risk levels.
In order to maintain consistency between the system SRA (R9.5), Section 9.4 provides example threat classes adjustable to the specific settings. This threat catalog also raises time efficiency since it narrows the time needed to evaluate possible attack targets according to RAP information (EC1).
The method needs some manual adjustments to the input data. Examples are annotating the deployment information and clarifying uncertainties in the function-oriented SRA results. Nevertheless, these are necessary steps independent of the system SRA method.

But we do not think there is much sense in trying to deal with the devil in the detail right away - and ignoring his grandparents, who may be hidden in an inappropriate, mistaken, or non-existent overall concept.

[37, p. 28]

15 Use Case: Risk Treatment

This work proposed to divide the risk analysis and risk treatment steps. By that, it is possible to include the system SRA results and accomplish a component-global risk treatment. Therefore, it is (again) impossible to base the evaluation on an existing use case adjusted to the improvements, but building the use case with narrow data is necessary.

15.1. Light Functions

Throughout the use case and evaluation, the standard light cluster served as the basic use case. In the system SRA the scope laid on one ECU and to make it more comprehensive only on function $F6$. The risk treatment method aims to use the component global risk analysis results. Therefore, it also concentrates on $F6$ of the example ECU. First, it assigns the general condition requirements before prioritizing the single risks, before proceeding to those with an assigned attack path in descending order of their impact.
The available data narrows to the information gathered from the function-oriented SRA and fictive component information. Therefore, applying the algorithm to the complete component information is impossible. However, using the available information to illustrate the procedure should be suitable. The advantage is that the results are more comprehensive than a complete component.
Also, the suggested method is not yet in place, and the proposed taxonomy was not subject to a security planning step, which leaves it incomplete. Therefore, this evaluation has no claim for completeness. Using the proposed taxonomy and method, the result would be different. A component-global risk treatment would prioritize the highest impact risks on overall functions. Other assignments may mitigate certain risks of $F6$ in the by-catch test resulting in a different defense method assignment.

15.1.1. General Conditions

General conditions arise from company policies or normative references. According to the attack paths, and UNECE No. R155 it is necessary to implement an IDS method.

15.1.2. System-related Requirements

For the application of system-related requirements, the method first targets single risks before proceeding with the highest impact first. Thereby, the method takes risks against the road user for the application. Risks against the OEM are only subject to the by-catch test (R10.3, R10.4).
$F6$ has no risk without an assigned attack path (no single risk). Therefore, the first assignment round has no input, and the algorithm proceeds directly to the highest impact first loop.
For the system-related requirements, the assumption is that the risk acceptance threshold is at “very low” meaning that every risk above this level is subject to risk treatment. Relevant for the function $F6$ are nine risks arising from 13 damage scenarios. Those risks are six with a moderate and three with a low risk level. The damage scenarios have a low impact, and all have a safety impact. Therefore, the algorithm starts with the first of six moderate risks assigning a suitable defense method.
Overall, the algorithm applies three defense methods, one of which is a demand from normative references. Through the deployment, the function $F6$ is directly hardware

isolated on *CPU2*. Therefore, no additional technical isolation method is necessary. Another technical defense method is secure boot, which secures the storage integrity and thereby the integrity of the functions running on it. Communication over the CAN Bus is in focus for the functional defense methods. Therefore, applying SecOC is a good result.
On a real component-global basis, the by-catch test would provide the following result: The IDS system (general condition) is host-based. Therefore it targets only *CPU2* where the function is running. Secure boot is a component-global method that can secure all software running on the ECU. On the other hand, the SecOC protocol restraints data flows. Since the ECU supports SecOC, the other data flows may also use it. Secure boot and SecOC demands the existence of a trust anchor. Secure boot for general usability, SecOC depending on the configuration and the security level to be reachable (R10.1). However, since the component natively has an HSM deployed, this demand is no problem. In the result, the algorithm applies three defense methods with a high by-catch usage. There are no open risks necessary to treat in another way.

15.2. Method Evaluation

The function risk analysis step in the current process implementation also incorporates risk treatment. The introduction of the system SRA and efficiency issues (R10.4) made it necessary to divide risk treatment from the function-oriented SRA. The described risk treatment method supports variations of defense methods. Thereby, it supports different levels of defense (R10.1). Those different levels of defense also provide trade-offs with respect to resource usage and implementation costs (EC3, R10.3).
Currently, the algorithm prefers methods with the highest impact on all threatened security objectives. This priority led to the assignment of defense methods that impact more than one security objective before only one. Typically it is cheaper to implement one costly defense method, which impacts more security objectives than to implement several defense methods which target fewer security objectives. Nevertheless, this preference might lead to higher costs in some cases. Including the implementation costs into the assignment leads to a higher state space and a more optimal solution regarding the costs (R10.3).
On the other hand, it is costly and challenging to achieve the implementation costs of the defense methods. Since, for some defense methods, the costs are hardware and implementation-dependent, the costs change for each hardware type. Average costs might prevent this problem while still optimizing the assignment. At least for each vehicle project, the costs have to be newly evaluated during the defense method catalog update.
The approach incorporates different origins of defense methods. Normative and organizational security demands must be adhered to in the development (R10.2). Otherwise, the system is infeasible from those points of view. Therefore, the method differentiates between those security demands and risks revealed in the risk analysis step (EC3). On the one hand, this distinction allows tracing the source of assigned

defense methods (EC3). On the other hand, this enables trade-offs in the assignment of system-related defense methods (EC3, R10.3).
Efficient risk treatment (EC3, R10.4) demands a particular structure in the assignment process. Therefore, the approach mitigates threats without attack paths first. Single threats are either single-points-of failure or preparation attacks: The former needs mitigations for a high-security defense. The latter has a high by-catch rate since they directly cut attack paths. After that, we prioritize the attack path with the highest impact against the road user.
Automatic assignments of defense methods always have the probability of misjudgment [38]. This limitation also accounts for the current version of the presented approach. Therefore, the result needs manual validation. Nevertheless, this semi-automatic assignment is more efficient than a function-local or complete manual assignment (EC1).
ISO/SAE 21434 clearly defines impacts only to the road user. Therefore, the current algorithm version mitigates only threats impacting the road user by raising efficiency and lowering costs (EC3). Typically, threats have an impact on both stakeholders to a varying degree. Only a few threats remain untreated by targeting the threats with impact against the road user. Those are only regarding the OEM, or the road user impact is below the threshold while the OEM impact is above the threshold. Untreated threats might benefit from assigned defense methods. A global by-catch test on the complete list of threats reveals such situations. Nevertheless, this is a possible point to optimize the system setup. A second threshold for the OEM impact allows a second run of the algorithm over those threats. Combined with a cost limit, it is possible to mitigate OEM-related threats to a certain degree (EC1).
Besides the advantages for the system implementation, the presented approach provides benefits for the security and development departments. A direct evaluation against the current process implementation is complex since the risk treatment step is not separate and does not include the component risks. However, a component-global and semi-automatic risk treatment provides a high time efficiency for the security department (EC1). The security department needs to accomplish another separate process step. However, this step directly targets a complete component leading to fewer contradictions. Applying defense methods in the function-oriented SRA, also based on a narrow catalog, may lead to contradictions between applied methods. Also, it may be that the component does not support the defense method, e.g., due to a missing HSM. This issue arises from the early point in time of the function-oriented SRA. At this point, the deployment is unfinished, leading to insufficient information about the hardware and system software structure.
For the development departments, a component-global defense method approach is beneficial. The development departments have less different defense method demands for the implementation and therefore lowering implementation costs (EC1, EC2). Also, this reduction lowers hardware resource demands, which is again beneficial in terms of run-time efficiency and costs.

Part V.

Closing

Science is a wonderful thing if one does not have to earn one's living at it.

Albert Einstein

16 Discussion

The preceding chapters discussed several improvements to the evaluated development process. For the SRE and the function-oriented SRA only minor changes already raise efficiency and traceability. The system SRA and explicit risk treatment are new steps with the same aim. This chapter discusses other ideas for improving the security development process.

16.1. Security development planning

The core of defense-in-depth core is a continuous refinement of the company's development policies and procedures. This approach reduces the number of "unwritten" policies. Therefore, decisions are traceable and results are reproducible [105, p. 12]. Planning is the first step in the security development process and was out of the scope of this work. The evaluated development process focuses mainly on one vehicle project at a time. Combined with the function-oriented development approach, this leads to a long development time where the component view is out of scope. Parallel to that, component development starts, and the security department assigns defense methods to the components based on experience. This approach may lead to overestimation because it does not ground on analysis results.
An idea to reduce the possibility of overestimation is to run two development strands in parallel. The project with the next start of production starts with the functional development layer. Then it proceeds to the system view and risk treatment. While it is in this progress, the next project starts in the functional view.
This idea might seem impossible at first glance. At the start of serial production, several functions drop out of the development process. Typically, most of them are subject to the subsequent development project. Therefore, it is reasonable to directly push them into the next iteration of the security development process. Other functions are part of future vehicle projects. Those are transferable and are then part of a re-use analysis in function-oriented development.

16.2. Emergency

The nature of meta-functional aspects is that their complete influence on the system gets known only in a holistic view. Examples are interactions that lead to former immune functions being vulnerable in the composed system [41]. In such cases, the system composition invalidates assumptions taken on the subsystem level.
The evaluated development process only verifies meta-functional aspects by testing at the end of the development time. Chapter 18 shows ideas for a holistic development process that allows verification of meta-functional aspects during the complete development cycle.

16.3. Safety and Security

Automotive systems are risk-prone to safety and security. Therefore, both meta-functional aspects co-habitat in vehicles leading to continuous interference. Safety targets protecting the vehicle environment from harm (Definition 7). Security, however, targets the system's resistance against introducing malicious external faults from the vehicle's environment. Both aspects target the system from dual viewpoints.

Those differences result in ways to reduce a systems risk [76]. The introduced safety measures and security defense methods can either support or impede each other. Including both aspects in a holistic development process enables a complete overview of the system's risks. This idea is subject to the idea of architectural verification in Chapter 18.

16.4. Formal Methods

The development teams work with different modeling and analysis tools in complex distributed development environments. With the start of serial production, the companies work with an integrated architectural model parallel to other development tools. The resulting requirements are part of Doors sheets [19, p. 328].
This vast amount of different tools and models makes the development process unclear and can lead to inconsistencies. The use of formal methods starting with the development process makes it possible to reveal development flaws at any time [122]. Also, it directly supports the analysis of meta-functional aspects. Formal methods typically allow the import with different formats, making it possible to include the models from the development department tools.
Chapter 18 introduces the idea of development contracts to cope with these issues.

Embedded software, the field in which I've spent most of my career, faces an existential crisis.

Marilyn Wolf

17 Conclusion

This work aimed to design a complete, consistent, and efficient security development process for the automotive industry. The design of the resulting process used two sources of criteria: The methodology from Chapter 3 and the evaluation criteria of Chapter 11. Those served as guidelines to answer the research questions stated in the motivation (see Section 1.1):

RQ 1: What is a suitable terminology for the security development process? This work targets readers from academia and industry. Chapter 4 incorporates them by introducing the general concept of dependability using the *Laprie model*. The presented definitions use related work from normative references, the *Laprie model*, research, and industry publications. This chapter also serves as a baseline to transfer the *Laprie model* into a security taxonomy.
Since automotive security is a relatively new topic, Chapter 5 introduces a security taxonomy to present a distinct terminology. This taxonomy aligns with the structure of the *Laprie model* and subdivides it into security objectives, threats, and mitigations. The presented definitions have the same sources as in the dependability chapter. A security threat and propagation chain emphasize the correlations between the defined terms. Thereby, the threat chain targets the course within the security terminology. The propagation chain also incorporates dependability aspects and presents the course within the systems development and customer usage time.
Chapter 10 completes the terminology by providing a taxonomy of defense methods. This taxonomy targets structuring the terminology and sources of security defense methods. The resulting structure is easily adjustable in every development cycle and other industry areas. Also, the structure allows for deriving a defense method catalog usable during risk treatment.

RQ 2: What are the demands of the normative references? and RQ 3: What are the necessary analysis and treatment steps? Evaluating the normative references reveals that the security development process under consideration covers the development steps demanded in the normative references. Nevertheless, it only indirectly incorporates threats introduced by the system level. Therefore, Chapter 9 provides a methodology for a system SRA. This is no new risk analysis step but supplements the results from the function-oriented SRA. The aim is to derive additional steps in the attack paths which arise from hardware and system software. A suitable analysis method updates the attack feasibilities and risks from the function-oriented SRA.

RQ 4: What are the possibilities to raise efficiency in the security development process? Chapter 7 and Chapter 8 suggest clustering as one solution to raise efficiency. The security relevance evaluation defines clusters of functions based on a presented clustering scheme. This cluster is subject to a simultaneous evaluation. A new analysis scheme provides efficiency and acceptance by dividing the results into single functions and providing information about resulting criticality clusters. This clustering is re-usable in the function-oriented SRA, enhancing efficiency both in the development and the security teams. The automatized process also raises acceptance in the development teams.
While the normative references do not define whether risk treatment and SRA need different development steps, it is reasonable to divide them. The evaluated development process includes risk treatment in the function-oriented SRA. This abstraction level is too narrow for efficient risk treatment because a higher abstraction leaves room for trade-offs and lowers the effort. Also, introducing a system SRA as part of the risk analysis step makes a division necessary.
Chapter 10 suggests a method for heuristic risk treatment that allows for different prioritization and trade-offs. This method prioritizes at first single risks, comparable to SPOF or preparation attacks. In both cases, it is recommendable to treat them first to prevent single entry points or to make risk treatment more efficient by cutting attack paths. The second step of the method prioritizes risks with the highest impact since they have a high damage potential. Besides, this method aligns with ISO/SAE 21434 by first treating risks to the road user. Risks to the OEM are subject to the by-catch test. This step tries to apply assigned defense methods to other risks to leverage them without further effort.

RQ 5: Where is tracing necessary? How can tracing be introduced? It is impossible to have negative or uncertain answers in the SRE, as defined in the normative references and the evaluated process description. Therefore, every function is relevant to some extent, and if the necessary information is currently missing, it is impossible to complete the SRE without assuming data. Chapter 7 suggests an extension of the questionnaire to overcome this issues. The analysis script evaluates answers which result in no damage potential with 0 and uncertainty with -1. The illustrated result depicts uncertainties in supporting tracing. Before proceeding to

the function-oriented SRA, those uncertainties are subject to re-cap if the function would be irrelevant without this answer.
Chapter 8 proposes to align the damage potential catalog with the SRE questionnaire. This alignment enables the tracing of information and makes the overall development process comprehensive. Besides, this chapter elaborates on the necessity to trace open issues remaining from the analysis and catalog changes primarily arising from the threat model.
The future work (Chapter 18) links to these issues by providing the idea of contract-based development to establish a holistic and formally supported development process. This approach allows the identification of necessary reevaluations in case of changes in the underlying policies, e.g., the threat model.

Dissertations are not finished; they are abandoned.

Fred Brooks

18 Future Work

The presented work evaluated a currently implemented security development process limited to the V-Models left leg. The evaluated security development process is successfully audited. In order to not endanger the audit certification, only minor process adjustments were possible. Nevertheless, the different process methodologies and tools make overall traceability hard to achieve. Especially the transfer onto the V-Models right leg is difficult.
Relinquishing the constraints of this work enables a different view of the development process. The idea to overcome this is to accompany the security development process with a holistic modeling methodology that allows incorporating the analysis results from the functional and meta-functional development strands to overcome this issue. Nevertheless, this approach demands profound adjustments in those development strands.
Therefore, future work aims to develop the idea of a holistic development process using development contracts. Such a development process enables fast reactions to changes in the system, during the development time and also during customer usage time. Transferred to meta-functional aspects, it enables safety- and security-by-design from the beginning in the development process [48].
In the following chapter, this idea is subject to evaluation. It starts by introducing the idea of development contracts as an overall formal method. In order to stay comprehensive, the subsequent description of the transfer onto the automotive development process limits the security development process. However, the same procedure also counts for the functional and other meta-functional system properties.

18.1. Holistic View on the Development Process

Currently, the development process, also in the security department, uses different separate methods and tools to accomplish the necessary development modeling and analysis. Verification and validation are complicated process steps without a good companionship with a framework. This difficulty prevents a fast reaction to system changes.
As described by [34] and [57] modern automotive systems demand updates, also during run-time. The concept of dynamic composition supports those frequent system changes with the need for continuous validation and verification.
The idea of dynamic composition is to support the development process by models and define constraints that guarantee a correct system composition. Thereby it is possible to compose the system parts on differing levels of abstraction continuously. This continuous composition follows the idea of correctness-by-construction.
The concept of dynamic composition uses formal models as a basis and formal constraints for composition and refinement. The remaining chapter presents the idea of contract-based development as a formal basis for composition and refinement during the development process.

18.2. Assumption Guarantee Contracts

Assumption-guarantee contracts specify the allowed design contexts and characteristics on functional and meta-functional level [26]. The contract defines items' behavioral properties, where an item is the unit of design [20]. Therefore, those contracts support system decomposition and development methodologies, e.g., V-Model, spiral model.
Each contract consists of assumptions and guarantees. The guarantee defines the design artifacts [26] or the guaranteed behavior under correct usage. Therefore, each guarantee specifies the required properties of the system [19]. Those guarantees directly support the systems engineering idea of different requirement levels. During decomposition, each abstraction level refines the guarantees, and validation and verification steps evaluate if the system fulfills the guarantees.
Contract assumptions specify the design context [26] or constraints the item has for correct behavior. Those assumptions are not under the control of the item, meaning they relate to the contract's environment [20]. Therefore, assumptions allow specifying the usage constraints of items. Assumptions limit responsibilities in distributed development environments. Also, they provide information for validation and verification, e.g., for test case generation.
Since contracts support decomposition and composition operators, they can represent the system decomposition and composition steps throughout the development process. It is also possible to divide the functional and meta-functional parts into different contracts and examine their compatibility during composition.
The structure of assumption-guarantee contracts also supports the development process's verification and validation phase. Contracts support virtual integration testing by allowing different system viewpoints [26].

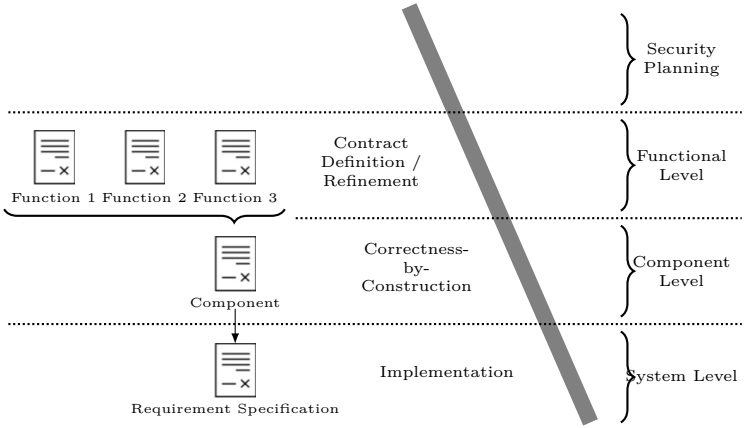

Figure 18.1.: Idea of contract-based development and correctness-by-construction in the security V-Model

18.3. Contract based Security Development

Automotive systems incorporate many different software functions whether correct behavior demands close cooperation between each other [16]. Therefore, the development process has several different viewpoints and analysis steps. Tracing the analysis results, requirements derivation, and refinement is possible with assumption-guarantee contracts. Therefore, accompanying the development process with contracts is a suitable method. This idea uses different contracts for the systems engineering abstraction layers and system viewpoints, e.g., safety, security, and function. The evolution of the development process uses contract refinement and composition to supplement the security contracts.

The preceding chapters of this work described the security development process steps along with the current implementation. Accompanying this process steps by security contracts leads to the layout in Figure 18.1.

In the function-oriented part, decomposing the system contract into one contract per function and viewpoint describes the function's behavior in a functional and meta-functional manner. Each process step refines the contracts: The input to the process step refines the assumptions; the work products refine the guarantees.

For the component level, composing the function-based contracts per component guarantees correctness-by-construction. Extending this composed contract by hardware and system software contracts leads to a holistic view.

The resulting contracts are usable as requirement specifications for the implementation. Also, the composition of the contracts fulfills the system contract and bridges

the gap back to the initial requirements. The remaining section highlights the procedure and advantages of the security development process. Those are transferable to the other viewpoints, e.g., safety.

18.3.1. Risk Analysis

The risk analysis according to ISO/SAE 21434 comprises three parts: The SRE, function, and system SRA. Contract-based security development has several advantages for this process step.

SRE - Contract decomposition and refinement leads to high automation possibilities for the SRE. The functional contract includes most of the necessary information to evaluate the security relevance of the function. Therefore, the security contract refines to the "answers" from the template. Through the formal nature of the contract, it is possible to include uncertainty and trace changes directly.

SRA - The SRA part of the risk analysis step aims to identify assets, damage scenarios, and threats and rate them according to their impact and attack potential. Contract-based development supports this step by providing most of the necessary input information from the applicable contracts. The SRA then refine the security contract by introducing the mentioned parts. Also, the applicable contracts support the derivation of information for attack tree modeling. This approach makes the attack path estimation more comprehensive and reproducible. Also, it reduces the informality of the SRA leading to an easier tracing of influences in case of changes.

18.3.2. Risk Treatment

The current approach for risk treatment excludes compatibility validation of the assigned defense methods for interface security between different communication partners. This limitation is due to the state space explosion problem in system-global risk treatment and can lead to inconsistencies between the components. Using contract-based development overcomes this issue. The risk treatment step extends the security contract of the functions and components. This extension enables the inclusion of composition operators in the risk treatment procedure, which directly ensures compatibility through correctness-by-construction.

18.3.3. Requirement Specification

The derivation of requirement specifications was out of the scope of this work's process evaluation. Currently, most requirement specifications are informal documents, like Doors sheets or Writer documents enhanced by (semi-) formal information. Those documents lack comprehensiveness and the possibility of tracing.
Contract-based development allows deriving requirement specifications from the composed contracts directly. Therefore, the responsible person links the assumptions and guarantees to the requirement specification. This approach ensures completeness and traceability of requirements.

18.3.4. Validation and Verification

Contract-based development directly supports the V-Models right leg. The security validation and verification steps aim to ensure the implementation of the requirements defined in the V-Models left leg. Typically, these steps use test case generation, e.g., for penetration testing.
This approach is subject to the main drawback of testing: There is no possibility to ensure the absence of mistakes, but only the existence - 100 % test coverage is impossible in an efficient way. Also, testing reveals inconsistencies and interference too late in the development process. A holistic design process through contract-based development allows virtual integration testing in the early design phases.

18.3.5. Other Advantages

A holistic development process has several advantages in terms of architectural verification. Besides the challenges mentioned in the authors' work ([57, 58, 61]), two critical open issues remain after this work. Those are answerable through a holistic approach:

Confidentiality - Including the architectural view and global information flow analysis into the analysis and risk treatment steps enables efficient isolation and encryption techniques. Through the global and distinct view of the system, it is possible to evaluate the borders where it is necessary to introduce defense methods.

Availability - The property of being accessible and usable is a property related to the underlying logical and physical communication architecture. Therefore, it needs architectural analysis regarding the availability of the hardware, as well as the logical architecture utilizing the hardware. The analysis of availability demands the transfer of the demanded communication structure onto the physical-logical architecture regarding mechanisms preventing the information flow, e.g., Firewalls, Routers. This analysis is impossible using scattered analysis procedures and various data sources but needs a holistic view of the system.

Bibliography

[1] Audi AG. “AUDI: ECU Security Architecture Model”. internal. 2021.

[2] Audi AG. “AUDI: Vehicle Security Architecture Model”. internal. 2021.

[3] Dakshi Agrawal and Doğan Kesdoğan. “Measuring Anonymity: The Disclosure Attack”. In: *IEEE Security & Privacy* 1.6 (Nov. 2003), pp. 27–34. DOI: 10.1109/MSECP.2003.1253565.

[4] Mohsen Ahmadvand, Alexander Pretschner, and Florian Kelbert. “A Taxonomy of Software Integrity Protection Techniques”. In: vol. 112. Advances in Computers. Elsevier, 2019, pp. 413–486. DOI: 10.1016/bs.adcom.2017.12.007.

[5] Adnan Akhunzada et al. “Securing Software Defined Networks: Taxonomy, Requirements, and Open Issues”. In: *IEEE Communications Magazine* 53.4 (Apr. 2015), pp. 36–44. DOI: 10.1109/MCOM.2015.7081073.

[6] Jason Andress. *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. 2nd ed. Waltham, MA, United States: Syngress, 2014. ISBN: 978-0-12-800744-0.

[7] Daniel Angermeier. “SECURITY RISK ASSESSMENTS: Seminar on Method and Tooling”. internal. 2021.

[8] Daniel Angermeier and Jörn Eichler. “Risk-Driven Security Engineering in the Automotive Domain”. In: Embedded Security in Cars (Escar USA) (Detroit, United States, June 1–2, 2016). 2016.

[9] Daniel Angermeier, Alexander Nieding, and Jörn Eichler. “Supporting Risk Assessment with the Systematic Identification, Merging, and Validation of Security Goals”. In: *Risk Assessment and Risk-Driven Quality Assurance*. Vol. 10224. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 82–95. DOI: 10.1007/978-3-319-57858-3_7.

[10] Daniel Angermeier et al. *Modeling Security Risk Assessments*. 2019. URL: https://www.researchgate.net/profile/Joern-Eichler/publication/336413410_Modeling_Security_Risk_Assessments/links/5da04e63a6fdcc8fc346ee2f/Modeling-Security-Risk-Assessments.pdf (visited on 07/05/2022).

[11] AUTOCRYPT. *The Changing Automotive E/E Architecture and What It Means for the Supply Chain*. 2021. URL: https://autocrypt.io/changing-automotive-e-e-architecture/ (visited on 05/27/2022).

[12] Algirdas Avižienis and Jean-Claude Laprie. “Dependable Computing: From Concepts to Design Diversity”. In: *Proceedings of the IEEE* 74.5 (May 1986), pp. 629–638. DOI: 10.1109/PROC.1986.13527.

[13] Algirdas Avižienis, Jean-Claude Laprie, and Brian Randell. "Dependability and Its Threats: A Taxonomy". In: *Building the Information Society*. Vol. 156. IFIP Advances in Information and Communication Technology. Boston, MA, United States: Springer, 2004, pp. 91–120. DOI: 10.1007/978-1-4020-8157-6_13.

[14] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell, et al. *Fundamental Concepts of Dependability*. Tech. rep. 739. University of Newcastle upon Tyne, Computing Science, 2001.

[15] Algirdas Avižienis et al. "Basic Concepts and Taxonomy of Dependable and Secure Computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Oct. 2004), pp. 11–33. DOI: 10.1109/TDSC.2004.2.

[16] Thomas Beck, Clemens Reichmann, and Jörg Schäuffele. "E/E-Entwicklung Modellbasierter Ansatz vom Architekturentwurf bis zur Serienreife". In: *ATZ elektronik* 11.6 (2016), pp. 58–63. DOI: 10.1007/s35658-016-0094-7.

[17] Lotfi Ben Othmane et al. "Incorporating Attacker Capabilities in Risk Estimation and Mitigation". In: *Computers & Security* 51 (May 2015), pp. 41–61. DOI: 10.1016/j.cose.2015.03.001.

[18] Albert Benveniste and Gérard Berry. "The Synchronous Approach to Reactive and Real-Time Systems". In: *Proceedings of the IEEE* 79.9 (Sept. 1991), pp. 1270–1282. DOI: 10.1109/5.97297.

[19] Albert Benveniste et al. "Contracts for System Design". In: *Foundations and Trends in Electronic Design Automation* 12.2-3 (Mar. 2018), pp. 124–400. DOI: 10.1561/1000000053.

[20] Albert Benveniste et al. "Multiple Viewpoint Contract-Based Specification and Design". In: *Formal Methods for Components and Objects*. Ed. by Frank S. de Boer et al. Vol. 5382. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 200–225. DOI: 10.1007/978-3-540-92188-2_9.

[21] Norm Bridge and Corinne Miller. "Orthogonal Defect Classification Using Defect Data to Improve Software Development". In: *Software Quality* 3.1 (1998), pp. 1–8. DOI: 10.1.1.41.7873.

[22] Manfred Broy, Günter Reichart, and Lutz Rothhardt. "Architekturen softwarebasierter Funktionen im Fahrzeug: von den Anforderungen zur Umsetzung". In: *Informatik-Spektrum* 34.1 (Jan. 2011), pp. 42–59. DOI: 10.1007/s00287-010-0507-6.

[23] Carl Carlson. *Effective FMEAs: Achieving Safe, Reliable, and Economical Products and Processes Using Failure Mode and Effects Analysis*. 1st ed. John Wiley & Sons, 2012. ISBN: 978-1118007433.

[24] Lawrence Chung et al. *Non-Functional Requirements in Software Engineering*. 1st ed. Vol. 5. The Kluwer International Series in Software Engineering. Boston: Kluwer Academic, 1999. ISBN: 978-0-7923-8666-7.

[25] Barbara J. Czerny. "System Security and System Safety Engineering: Differences and Similarities and a System Security Engineering Process Based on the ISO 26262 Process Framework". In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 6.1 (Apr. 2013), pp. 349–359. DOI: 10.4271/2013-01-1419.

[26] Werner Damm et al. "Using Contract-Based Component Specifications for Virtual Integration Testing and Architecture Design". In: *2011 Design, Automation & Test in Europe*. 2011 Design, Automation & Test in Europe (Grenoble, France, Mar. 14–18, 2011). IEEE, Mar. 2011, pp. 1–6. DOI: 10.1109/DATE.2011.5763167.

[27] Yuri Gill Dantas, Vivek Nigam, and Harald Rueß. *Security Engineering for ISO 21434*. München: fortiss GmbH, 2021. URL: https://www.fortiss.org/fileadmin/user_upload/05_Veroeffentlichungen/Informationsmaterialien/fortiss_whitepaper_security_engineering_ISO_web.pdf (visited on 08/18/2021).

[28] Delloite. *Cyber Security Management – Neue Dimensionen Automobiler Sicherheit*. Mar. 2021. URL: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/risk/Trusted-Software-POV-UNECE-R-155.pdf (visited on 06/17/2022).

[29] Luc van Dijk. *Future Vehicle Networks and ECUs*. Whitepaper FVNECUA4WP. 2017. URL: https://www.nxp.com/docs/en/white-paper/FVNECUA4WP.pdf (visited on 06/29/2022).

[30] Jörn Eichler and Daniel Angermeier. "Modular Risk Assessment for the Development of Secure Automotive Systems". In: *Proceedings of the 31st VDI/VW Joint Conference Automotive Security*. 31st VDI/VW Joint Conference Automotive Security (Wolfsburg, Germany). VDI-Berichte. VDI, 2015, p. 10.

[31] ETSI. *CYBER; Methods and Protocols; Part 1: Method and pro Forma for Threat, Vulnerability, Risk Analysis (TVRA), Version 5.2.3*. 0018. 2017. URL: https://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/05.02.03_60/ts_10216501v050203p.pdf (visited on 06/29/2022).

[32] EUROPEAN PARLIAMENT AND COUNCIL. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)*. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=DE (visited on 05/29/2022).

[33] Felix C. Freiling. “Introduction to Security Metrics”. In: *Dependability Metrics*. Vol. 4909. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 129–132. DOI: 10.1007/978-3-540-68947-8_11.

[34] Axel Freiwald and Gunn Hwang. “Safe and Secure Software Updates Over The Air for Electronic Brake Control Systems”. In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 10.1 (Sept. 2016), pp. 71–82. DOI: 10.4271/2016-01-1948.

[35] Benjamin Glas et al. “Automotive Safety and Security Integration Challenges”. In: *Automotive-Safety & Security 2014*. Bonn, Germany: Gesellschaft für Informatik e.V., 2015, pp. 13–28. ISBN: 978-3-88579-634-3.

[36] Iris Graessler and Julian Hentze. “The new V-Model of VDI 2206 and its validation”. In: *at - Automatisierungstechnik* 68.5 (2020), pp. 312–324. DOI: 10.1515/auto-2020-0015.

[37] Reinhard Haberfellner et al. *Systems Engineering: Fundamentals and Applications*. 1st ed. Cham: Springer International Publishing, 2019. DOI: 10.1007/978-3-030-13431-0.

[38] Gerhard Hansch. “Automating Security Risk and Requirements Management for Cyber-Physical Systems”. Georg-August-Universität Göttingen, 2020. DOI: 10.24406/AISEC-N-608669.

[39] Gerhard Hansch, Peter Schneider, and Gerd Stefan Brost. “Deriving Impact-Driven Security Requirements and Monitoring Measures for Industrial IoT”. In: *Proceedings of the 5th Cyber-Physical System Security Workshop - CPSS '19*. The 5th Cyber-Physical System Security Workshop (Auckland, New Zealand, July 8, 2019). New York, NY, United States: ACM Press, 2019, pp. 37–45. DOI: 10.1145/3327961.3329528.

[40] Artur Hecker. “On System Security Metrics and the Definition Approaches”. In: *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. International Conference on Emerging Security Information, Systems, and Technologies (SecureWare) (Cap Esterel, France, Aug. 25–31, 2008). IEEE, Aug. 2008, pp. 412–419. DOI: 10.1109/SECURWARE.2008.37.

[41] Shawn Hernan et al. “Uncover Security Design Flaws Using The STRIDE Approach”. In: *MSDN Magazine* (2006). URL: https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach (visited on 12/17/2021).

[42] Martin Hillenbrand. *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen*. Steinbuch Series on Advances in Information Technology 4. Karlsruhe: KIT Scientific Publishing, 2012. ISBN: 978-3-86644-803-2.

[43] Chris Hobbs. *Embedded Software Development for Safety-Critical Systems*. 1st ed. New York, United States: Auerbach Publications, Sept. 2016. DOI: 10.1201/b18965.

[44] John Douglas Howard. "An Analysis of Security Incidents on the Internet 1989-1995". Carnegie Mellon University, 1997.

[45] IEC. *IEC 61508:2001 Functional safety of electrical/electronic/programmable electronic safety-related systems (German version EN 61508:2001)*. Norm. 2002.

[46] IEC. *IEC Guide 116:2010 Guidelines for safety related risk assessment and risk reduction for low voltage equipment*. Norm. 2010.

[47] IEEE. *Standard Glossary of Software Engineering Terminology*. Norm. Sept. 1990.

[48] *Trends in Next-Generation Automotive Safety and Security*. Tech. rep. 1. Intel Corporation, 2017. URL: car%20of%20the%20future%20trends%20in%20next-generation%20automotive%20safety%20and%20secu%E2%80%A6 (visited on 02/06/2018).

[49] Cynthia Irvine and Timothy Levin. "Quality of Security Service". In: *Proceedings of the 2000 workshop on New security paradigms*. NSPW00: New Security Paradigms 2000 (Ballycotton, County Cork Ireland, Sept. 18–21, 2000). New York, United States: ACM Press, 2000, pp. 91–99. DOI: 10.1145/366173.366195.

[50] Cynthia Irvine and Timothy E. Levin. "Toward a Taxonomy and Costing Method for Security Services". In: *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*. 15th Annual Computer Security Applications Conference (Phoenix, AZ, USA, Dec. 6–10, 1999). IEEE, 1999, p. 6. DOI: 10.1109/CSAC.1999.816026.

[51] Mafijul Md. Islam et al. "A Risk Assessment Framework for Automotive Embedded Systems". In: *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*. ASIA CCS '16: ACM Asia Conference on Computer and Communications Security (Xi'an China, May 30, 2016). New York, NY, United States: ACM, 2016, pp. 3–14. DOI: 10.1145/2899015.2899018.

[52] ISO. *ISO 21448:2022 Road vehicles — Safety of the intended functionality*. Norm. 2022.

[53] ISO. *ISO 26262:2018 Road vehicles – Functional safety*. Norm. 2018.

[54] ISO. *ISO/SAE 21434:2021 Road vehicles – Cybersecurity engineering*. Norm. 2021.

[55] ISO/IEC. *ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary*. Norm. 2018.

[56] ISO/IEC. *ISO/IEC 27001:2013 Information Technology — Security Techniques — Information Security Management Systems — Requirements*. Norm. 2013.

[57] Christine Jakobs, Matthias Werner, and Peter Tröger. "Dynamic Composition of Cyber-Physical Systems". In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences 2019 (Grand Wailea, Maui, Jan. 8–11, 2019). 2019, pp. 7232–7241. ISBN: 978-0-9981331-2-6.

[58] Christine Jakobs et al. "Following the White Rabbit: Integrity Verification Based on Risk Analysis Results". In: *Computer Science in Cars Symposium*. CSCS '21: Computer Science in Cars Symposium (Online, Ingolstadt, Germany, Nov. 30, 2021). CSCS '21. New York, NY, USA: ACM, 2021. DOI: 10.1145/3488904.3493377.

[59] Christine Jakobs et al. "Heuristic Risk Treatment for ISO/SAE 21434 Development Projects". In: 3rd International Forum of Cyber Security, Privacy, and Trust (NEMESIS'22) (Hybrid, Sofia, Bulgaria, Sept. 4–7, 2022). to appear. 2022.

[60] Christine Jakobs et al. "Streamlining Security Relevance Analysis According to ISO 21434". In: *Proceedings of the 5th International Conference on Networking, Information Systems & Security*. The 5th International Conference on Networking, Information Systems & Security. (Mar. 30–31, 2022). to appear. Virtual, 2022.

[61] Christine Jakobs et al. "Verification of Integrity in Vehicle Architectures". In: *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*. The 3rd International Conference on Networking, Information Systems & Security. (Virtual, Mar. 31–Apr. 2, 2020). NISS 2020. New York, NY, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3386723.3387883.

[62] Erland Jonsson. "An Integrated Framework for Security and Dependability". In: *Proceedings of the 1998 Workshop on New Security Paradigms - NSPW '98*. NSPW98: New Security Paradigms Workshop '98 (Charlottesville, Virginia, United States, Sept. 22–26, 1998). New York, United States: ACM Press, Jan. 1998, pp. 22–29. DOI: 10.1145/310889.310903.

[63] Erland Jonsson. "Towards an Integrated Conceptual Model of Security and Dependability". In: *First International Conference on Availability, Reliability and Security (ARES'06)*. First International Conference on Availability, Reliability and Security (ARES'06) (Vienna, Austria, Apr. 20–22, 2006). IEEE, 2006, p. 8. DOI: 10.1109/ARES.2006.138.

[64] Erland Jonsson and Laleh Pirzadeh. “Identifying Suitable Attributes for Security and Dependability Metrication”. In: SECURWARE 2013 - The 7th International Conference on Emerging Security Information, Systems and Technologies (Barcelona, Spain, Aug. 25–31, 2013). IARIA, 2013. ISBN: 978-1-61208-298-1.

[65] Erland Jonsson, Lars Strömberg, and Stefan Lindskog. “On the Functional Relation between Security and Dependability Impairments”. In: *Proceedings of the 1999 Workshop on New Security Paradigms*. NSPW99: 1999 New Security Paradigms Workshop (Caledon Hills Ontario Canada, Sept. 22–24, 1999). New York, United States: ACM, Sept. 1999, pp. 104–111. DOI: 10.1145/335169.335204.

[66] Christophe Jouvray et al. *EVITA Deliverable D3.1: Security and trust model*. Project Report 3.1. 2009.

[67] Joseph Eli Kasser. *Systems Engineering: A Systemic and Systematic Methodology for Solving Complex Problems*. 1st ed. Boca Raton: CRC Press, Taylor & Francis, Oct. 2019. ISBN: 978-1-138-38793-5.

[68] John C. Knight. “Safety Critical Systems: Challenges and Directions”. In: *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*. 24th International Conference on Software Engineering. ICSE 2002 (Orlando, FL, USA, May 19–25, 2002). IEEE, May 2002, pp. 547–550. ISBN: 1-58113-472-X.

[69] Jean-Claude Laprie. “Dependability: Basic Concepts and Terminology”. In: ed. by Jean-Claude Laprie. 1st ed. Vol. 5. Dependable Computing and Fault-Tolerant Systems. Vienna: Springer Vienna, 1992. DOI: 10.1007/978-3-7091-9170-5.

[70] Jean-Claude Laprie. “Dependable Computing and Fault-Tolerance”. In: *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'*. FTCS (Pasadena, CA, USA, USA, June 27–30, 1995). Vol. III. Reprinted from FTCS-15, 1985. IEEE, 1995, pp. 2–11. DOI: 10.1109/FTCSH.1995.532603.

[71] Jean-Claude Laprie. “Dependable Computing: Concepts, Limits, Challenges”. In: *Special Issue of the 25th International Symposium on Fault-Tolerant Computing*. Pasadena, California, USA, 1995.

[72] Jean-Claude Laprie and Karama Kanoun. “Software Reliability and System Reliability”. In: *Handbook for Software Reliability Engineering*. McGraw-Hill, 1996. ISBN: 0-07-039400-8.

[73] Ulf E. Larson, Dennis K. Nilsson, and Erland Jonsson. “An Approach to Specification-Based Attack Detection for in-Vehicle Networks”. In: *2008 IEEE Intelligent Vehicles Symposium*. 2008 IEEE Intelligent Vehicles Symposium (IV) (Eindhoven, Netherlands, June 4–6, 2008). IEEE, 2008, pp. 220–225. DOI: 10.1109/IVS.2008.4621263.

[74] Aljoscha Lautenbach, Magnus Almgren, and Tomas Olovsson. "Proposing HEAVENS 2.0 – an Automotive Risk Assessment Model". In: *Computer Science in Cars Symposium*. CSCS '21: Computer Science in Cars Symposium (Online, Ingolstadt, Germany, Nov. 30, 2021). New York, NY, United States: ACM, 2021, pp. 1–12. DOI: 10.1145/3488904.3493378.

[75] Kyungroul Lee et al. "A Brief Review on JTAG Security". In: *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS) (Fukuoka, Japan, July 6–8, 2016). July 2016, pp. 486–490. DOI: 10.1109/IMIS.2016.102.

[76] Maria B. Line et al. "Safety vs. Security?" In: *Proceedings of the Eighth International Conference on Probabilistic Safety Assessment & Management*. Eighth International Conference on Probabilistic Safety Assessment & Management (New Orleans, Louisiana, United States, May 14–18, 2006). New York, United States: ASME Press, 2006, p. 9. DOI: 10.1115/1.802442.paper151.

[77] Dahai Liu. *Systems Engineering*. 1st ed. Boca Raton: CRC Press, Taylor & Francis, 2017. ISBN: 978-1-315-27386-0.

[78] Pietre-Cambacedes Ludovic and Chaudet Claude. "Disentangling the Relations between Safety and Security". In: *AIC'09: Proceedings of the 9th WSEAS international conference on Applied informatics and communications*. 9th WSEAS international conference on Applied informatics and communications (AIC '09') (Moscow, Russia, Aug. 20–22, 2009). Stevens Point, Wisconsin, United States: World Scientific and Engineering Academy and Society (WSEAS), Aug. 2009, pp. 156–161. ISBN: 978-960-474-107-6.

[79] Peter Maier Christoph. *Integriertes Modell zur Entwicklung von funktional sicheren Produkten in der Automobilbranche*. 1st ed. Vol. 23. Stuttgarter Beiträge zur Produktionsforschung. Stuttgart: Fraunhofer Verlag, 2013. ISBN: 978-3-8396-0644-5.

[80] Sjouke Mauw and Martijn Oostdijk. "Foundations of Attack Trees". In: *Information Security and Cryptology - ICISC 2005*. Vol. 3935. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 186–198. DOI: 10.1007/11734727_17.

[81] McAfee. *Automotive Security Best Practices*. Whitepaper. 2016. URL: https://www.mcafee.com/hk/resources/white-papers/wp-automotive-security.pdf (visited on 02/06/2018).

[82] MITRE. *Common Attack Pattern Enumeration and Classification*. URL: https://capec.mitre.org/index.html (visited on 07/26/2022).

[83] Marco M. Morana and Tony UcedaVélez. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. 1st ed. Hoboken, New Jersey: Wiley, May 2015. ISBN: 978-1-118-98835-0.

[84] Motor Industry Software Reliability Association, ed. *Development Guidelines for Vehicle Based Software*. Leicester: RS Print Ltd, 1998. ISBN: 0-9524156-0-7.

[85] United Nations. *Proposal for a New UN Regulation on Uniform Provisions Concerning the Approval of Vehicles with Regards to Cyber Security and Cyber Security Management System (UN Regulation No. 155)*. Norm. 2020.

[86] United Nations. *Proposal for Amendments to the Proposed Interpretation Document for the Regulation on Uniform Provisions Concerning the Approval of Vehicles with Regards to Cyber Security and Cyber Security Management System (UN Regulation No. 155)*. Norm. 2020.

[87] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. "Model-Based Evaluation: From Dependability to Security". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), pp. 48–65. DOI: 10.1109/TDSC.2004.11.

[88] Oxford University Press. *absence*. In: *LEXICO - Powered by Oxford*. 2021. URL: https://www.lexico.com/definition/absence (visited on 05/19/2022).

[89] Oxford University Press. *avoidance*. In: *LEXICO - Powered by Oxford*. 2021. URL: https://www.lexico.com/definition/avoidance (visited on 05/19/2022).

[90] Oxford University Press. *occurrence*. In: *LEXICO - Powered by Oxford*. 2021. URL: https://www.lexico.com/definition/occurrence (visited on 05/19/2022).

[91] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. 3rd ed. Prentice Hall PTR, 2003. ISBN: 978-0-13-035548-5.

[92] R.S.H. Piggin and H.A. Boyes. "Safety and Security - a Story of Interdependence". In: *10th IET System Safety and Cyber-Security Conference 2015*. 10th IET System Safety and Cyber-Security Conference 2015 (Bristol, UK, Oct. 21–22, 2015). Institution of Engineering and Technology, 2015. DOI: 10.1049/cp.2015.0292.

[93] Marvin Rausand and Arnljot Høyland. *System Reliability Theory: Models, Statistical Methods and Applications*. 2nd ed. Wiley-Interscience, 2004. ISBN: 978-0-471-47133-2.

[94] Konrad Reif. "Sicherheitsaspekte und funktionale Sicherheit". In: *Automobilelektronik: Eine Einführung für Ingenieure*. 6th ed. to appear; own chapter of Christine Jakobs: Safety. Wiesbaden: Springer Fachmedien Wiesbaden, 2022.

[95] Thomas Reiß. *Ein Referenzmodell für die Serienentwicklung mechatronischer Systeme in der Automobilindustrie*. 1st ed. Audi-Dissertationsreihe 86. Göttingen: Cuvillier, 2014. ISBN: 978-3-95404-610-2.

[96] Alastair Ruddle et al. *EVITA Deliverable D2.3: Security Requirements for Automotive on-Board Networks Based on Dark-Side Scenarios*. Project Report 2.3. 2009.

[97] John Rushby. "Critical System Properties: Survey and Taxonomy". In: *Reliability Engineering & System Safety*. Special Issue on Software Safety 43.2 (Jan. 1994), pp. 189–219. DOI: 10.1016/0951-8320(94)90065-5.

[98] SAE International. *SAE J3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*. Norm. 2016.

[99] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems". In: *European Journal of Control* 18.3 (2012), pp. 217–238. DOI: 10.3166/ejc.18.217-238.

[100] Dieter Scheithauer. "Managing Concurrency in Systems Engineering". In: *INCOSE International Symposium* 22.1 (Nov. 2012), pp. 2016–2030. DOI: 10.1002/j.2334-5837.2012.tb01453.x.

[101] Dieter Scheithauer and Kevin Forsberg. "V-Model Views". In: *INCOSE International Symposium* 23.1 (Nov. 2013), pp. 502–516. DOI: 10.1002/j.2334-5837.2013.tb03035.x.

[102] Karsten Schmidt et al. "Adapted Development Process for Security in Networked Automotive Systems". In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 7.2 (Apr. 2014), pp. 516–526. DOI: 10.4271/2014-01-0334.

[103] Bruce Schneier. *Academic: Attack Trees - Schneier on Security*. 1999. URL: https://www.schneier.com/academic/archives/1999/12/attack_trees.html (visited on 05/03/2022).

[104] Torsten Schütze. "Automotive Security: Cryptography for Car2X Communication". In: *Embedded World Conference*. Workshop on Cryptography and Embedded Security. Nürnberg, Mar. 1, 2011, p. 16. URL: http://www.torsten-schuetze.de/reports/ieee1609-2_security.pdf (visited on 03/25/2020).

[105] Homeland Security. *Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies*. Sept. 2016. URL: https://www.cisa.gov/uscert/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf (visited on 04/14/2022).

[106] Nadine Sinner, Daniel Zelle, and Rasmus Robrahn. *Angreifermodell Für Selbstdatenschutz Im Vernetzten Fahrzeug*. Project Report D1.5. Aug. 2017.

[107] State Administration for Market Regulation; Standardization Administration of the People's Republic of China. *Technical requirements and test methods for cybersecurity of on-board information interactive system (GB/T 40856-2021)*. Norm. 2021.

[108] Benno Stützel et al. *Systems Engineering in Deutschland*. Studie. 2018. URL: http://www.prozesswerk.eu/site/assets/files/1179/se_studie_prozesswerk_doppelseiten.pdf (visited on 10/19/2021).

[109] Peter Tröger. *Unsicherheit und Uneindeutigkeit in Verlässlichkeitsmodellen*. 1st ed. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-23341-9.

[110] Peter Tröger, Lena Feinbube, and Matthias Werner. "WAP: What Activates a Bug? A Refinement of the Laprie Terminology Model". In: *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE) (Gaithersbury, MD, USA, Nov. 2–5, 2015). IEEE, Nov. 2015, pp. 106–111. DOI: 10.1109/ISSRE.2015.7381804.

[111] United Nations. *Uniform Provisions Concerning the Approval of Vehicles with Regards to Software Update and Software Updates Management System (UN Regulation No. 156)*. Norm. 2021.

[112] VDA QMC Working Group 13 / Automotive SIG. *Automotive SPICE Process Assessment / Reference Model - Version 3.1*. Nov. 2017. URL: https://www.automotivespice.com/fileadmin/software-download/AutomotiveSPICE_PAM_31.pdf (visited on 06/08/2022).

[113] David D. Walden et al., eds. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th ed. Hoboken, New Jersey: Wiley, 2015. ISBN: 978-1-118-99940-0.

[114] Andreas Warkentin, Jürgen Gausemeier, and Joachim Herbst. "Function Orientation beyond Development – Use Cases in the Late Phases of the Product Life Cycle". In: *Proceedings of the 19th CIRP Design Conference – Competitive Design*. 19th CIRP Design Conference – Competitive Design, 2009 (Cranfield, UK, Mar. 20–31, 2009). Cranfield, UK: Cranfield University Press, 2009, pp. 420–427. ISBN: 978-0-9557436-4-1.

[115] Jana Karina von Wedel and Paul Arndt. "Safe and Secure Development: Challenges and Opportunities". In: *SAE Technical Paper*. WCX World Congress Experience (Detroit Michigan, United States, Apr. 10–12, 2018). SAE International, 2018, p. 9. DOI: 10.4271/2018-01-0020.

[116] Tim Weilkiens et al. "B: The V-Model". In: *Model-Based System Architecture*. Hoboken, NJ, USA: John Wiley & Sons Inc, 2015. DOI: 10.1002/9781119051930.app2.

[117] Stephen R. Welke, Barry W. Johnson, and James H. Aylor. "Reliability Modeling of Hardware/Software Systems". In: *IEEE Transactions on Reliability* 44.3 (Sept. 1995), pp. 413–418. DOI: 10.1109/24.406575.

[118] Benjamin Weyl et al. *EVITA Deliverable D3.2: Secure On-board Architecture Specification*. Project Report 3.2. 2011.

[119] Marilyn Wolf. "Embedded software in crisis". In: *Computer* 49.1 (2016), pp. 88–90. DOI: 10.1109/MC.2016.18.

[120] Marilyn Wolf and Dimitrios Nikolaou Serpanos. *Safe and Secure Cyber-Physical Systems and Internet-of-Things Systems*. 1st ed. Springer Cham, 2020. DOI: 10.1007/978-3-030-25808-5.

[121] David C. Wynn and P. John Clarkson. "Process Models in Design and Development". In: *Research in Engineering Design* 29.2 (2018), pp. 161–202. DOI: 10.1007/s00163-017-0262-7.

[122] Saad Zafar and R. Geoff Dromey. "Integrating Safety and Security Requirements into Design of an Embedded System". In: *Proceedings. 12th Asia-Pacific Software Engineering Conference (APSEC'05)*. 12th Asia-Pacific Software Engineering Conference (APSEC'05) (Taipei, Taiwan, Dec. 15–17, 2005). IEEE, 2005, pp. 629–636. DOI: 10.1109/APSEC.2005.75.

Appendices

A

Attacker Model Categories and Rating

The following paragraphs briefly describe the attacker model categories.

Category	Level	Description	Value
Expertise	Layman	Typical vehicle owner/driver	0
	Proficient	Technically interested driver/garage staff	3
	Expert	Garage staff with special expertise (<20 %), e.g., installation of corrupted smart cards	6
	Specialist	Highly qualified person (< 1%), e.g., side-channel attacks, cryptanalysis	8
Access	Remote	Remote access without the need for direct access to a vehicle	0
	Easy	Simple direct access to the vehicle	1
	Moderate	Complex access to vehicle parts required, e.g., access to flash memory inside an ECU	4
	Hard	Access on micro-electronic level, e.g., voltage or current measurement	10
Time Need	Hours		0
	Days		1
	Weeks		3
	Months		7
	Decades		35
Equipment	Standard	Common IT-equipment, e.g., notebook, freely available OBD diagnosis-tools	0
	Specialized	Professional garage-equipment, e.g., CAN-cards, diagnosis-equipment	4
	Bespoke		7
	Multiple bespoke	Multiple bespoke tools	9
Knowledge	Public	Public information	0
	Internal	Internal information	3
	Classified	Classified information: information leak can endanger product- or project-goals	7
	Confidential	Confidential information: critical to the enterprise	11

Table A.1.: Attack potential categories and levels.

Expertise - Each threat to a system needs a level of general knowledge, or external knowledge, from the attacker. A typical car owner cannot hack into the infotainment system without, e.g., a detailed attack blueprint. Therefore, the attacker model allows defining the needed capabilities for each threat [54], [96, p. 86].

Knowledge - The knowledge to acquire a threat is distinguishable from the expertise. The knowledge is regarding the internal information about the system to attack, while the expertise is general knowledge needed about the attack type [54], [96, p. 86]. The knowledge is differentiable between publicly available up to confidential information [25].

Needed Time - Another necessary information about the attacker is the needed time to identify the vulnerability and execute the attack [54], [96, p. 86]. Some attacks may take only minutes, while others, like breaking keys, may take months or even years. Rating threats according to the elapsed time helps to rate the importance of the attack. If the attack needs long periods to execute, an attacker will diminish this threat and look for an easier way. On the other hand, especially for cryptography, the elapsed time needs to be traceable throughout the whole life-cycle of the automotive system. If a vulnerability in a cryptographic algorithm reveals later, the elapsed time may change dramatically, changing the complete rating of a given class of threats.

Access - The access category of the attacker model depicts which type and duration of access to resources the attacker needs [54]. It is a more detailed rating of the threat model attack surface combined with the execution part of the elapsed time [96, p. 86]. On the one hand, for remote threats, also the access is remote. For local threats, the access is distinguishable from the difficulty of gaining access. This difference is no general answer given in the threat model but is continuously adjustable to the system under consideration.

Equipment - Depending on the system under consideration and the threat, different tools are necessary to accomplish the attack [54]. Those are distinguishable, e.g., between standard tools and those only available to OEMs. This difference gives rise to the probability that the threat is accomplishable. Threats with a high categorization in the equipment class are less attractive than those with standard tools.

B

Basic Threat Classes for System SRA

ID	Class	Ob	Lay	Ex	Ac	T	Eq	Kn	RAP
1	Tampering of data by using an interface	I	DF	P	Ea	D	St	In	Vl
2	Tampering of data by providing an interface	I	DS	E	Ea	H	Sp	Cl	Lo
3	Information disclosure of data by using an interface	C	DF	L	Ea	H	St	In	Vl
4	Information disclosure of data by providing an interface	C	DS	P	M	H	Sp	Cl	Lo
5	Information disclosure of data by tapping an interface	C	DS	E	M	W	Sp	Cl	Mod
6	Denial of service of data by interrupting an interface	A	DF/ DS	L	Ea	H	St	Pub	Vl
7	Tampering of resource behavior at rest	I	DS	S	M	M	Be	Cl	Hi
8	Tampering of resource behavior during run-time	I	DS	S	M	M	Be	In	Hi

Abbreviations:

Objective – **C**onfidentiality; **I**ntegrity; **A**vailability
Layer – **D**ata **F**low; **D**ata **S**torage
Expertise – **P**roficient; **L**ayman; **E**xpert; **S**pecialist
Time – **M**ths; **W**eeks; **D**ays; **H**ours
Access – **Ea**sy; **M**oderate
Equipment – **St**andard; **Sp**ecialized; **Be**spoke
Knowledge – **C**lassified; **In**ternal; **Pub**lic
RAP – **V**ery low; **Low**; **Mod**erate; **Hi**gh

Table B.1.: Basic threat classes for system SRA and example attack potential.

C

Categories of Defense Method Properties

	Dependencies	Overhead	Impact		Requirements	Surface	Effect
Interface	CAN	load (%)	Confid.	OS	Secure Timer	remote	Ex
	LIN	run-time peaks	Integrity		Watchdog	local	Ac
	Ethernet		Avail.		RNG		T
Protocol	TLS 1.2			Root of Trust	**HSM**		Eq
	SOME/IP				external		Kn
	VIWI				coupled at CPU		RAP
	MQTT				**Trust Zone**		
	SOCKS				**TEE**		
	MOSE			Software	FDS		
	SOA				VKMS		

Table C.1.: Categories of Properties and Examples. Bold indicates sub-categories, italic indicates variants.