METAMODEL FOR SAFETY AND SECURITY INTEGRATED SYSTEM ARCHITECTURE MODELING

Kharatyan, Aschot;
Tekaat, Julian;
Japs, Sergej;
Anacker, Harald;
Dumitrescu, Roman

Fraunhofer Research Institute for Mechatronic Systems Design IEM

ABSTRACT

As digitization progresses, the integration of information and communication technologies in technical systems is constantly increasing. Fascinating value potentials are emerging (e.g. autonomous driving), but also challenges in the system development. The constantly increasing product complexity and degree of networking require a systemic development, which is fulfilled by established approaches of Model-Based Systems Engineering (MBSE). To ensure the reliability of tomorrow's systems, an integrative and early consideration of security and safety is additionally required. In order to show the possibility and consequences of failures and attacks, the paper develops a modeling language that links established and partly isolated security and safety approaches within a consistent metamodel. The developer is enabled to synthesize system architectures transparently on an interdisciplinary level and to analyze attack and failure propagation integratively. The approach uncovers synergetic and especially contrasting goals and effects of architectural designs in terms of safety and security in order to make adequate architectural decisions based on trade-off analyses.

Keywords: Systems Engineering (SE), Product architecture, Early design phases, Security and safety by design

Contact:
Kharatyan, Aschot
Fraunhofer Research Institute for Mechatronic Systems Design IEM
Product Engineering
Germany
aschot.kharatyan@iem.fraunhofer.de

Cite this article: Kharatyan, A., Tekaat, J., Japs, S., Anacker, H., Dumitrescu, R. (2021) 'Metamodel for Safety and Security Integrated System Architecture Modeling', in *Proceedings of the International Conference on Engineering Design (ICED21),* Gothenburg, Sweden, 16-20 August 2021. DOI:10.1017/pds.2021.464

1 INTRODUCTION

The increasing digitization and networking of mechatronic systems is leading to a growing integration of information and communication technologies. In the field of mobility in particular, this is leading to enormous leaps in innovation and fascinating potential benefits of the vehicles of the future (Dumitrescu et al., 2018). The change from classic technical systems to intelligent systems involves, in addition to the new functionalities and their benefits, increasing challenges for system development. A promising approach to the development of complex interdisciplinary systems is offered by Systems Engineering (SE) and the consistent continuation by Model Based Systems Engineering (Walden et al., 2015). Particularly in the early phase of development, the consistent use of models creates a great potential to master complexity and, for example, to detect and eliminate possible errors at an early stage.

Studies show that the constantly increasing degree of connectivity, automation and autonomy, especially in the automotive industry, results in a high vulnerability of technical systems (Miller and Valasek, 2013). Ensuring system reliability (in particular functional safety and cyber security) thus represents a further major challenge for today's developers (Nigam et al., 2019). The question arises as to how developers can be supported in the early stages of product development within the MBSE in order to develop future products in a safe and secure manner. In order to synthesize system architecture models and to be able to analyze the challenges of safety and security integratively across individual domains, a modeling language is required that addresses security and safety-relevant aspects integratively.

The goal of this paper is to create a basis for a safety and security-oriented system modeling by specifying a modeling language for the system architecture. The developer is enabled to identify errors and threats such as error propagation and attack vectors in an integrative way at an early development stage. In the first chapter, the problem analysis is adressed, in which the system architecture modeling, the challenges of safety and security as well as the necessity of an integrative integration into the MBSE modeling language are presented. The second chapter evaluates the state of the art. The third chapter presents the solution approach in the form of a metamodel, while the fourth chapter includes a validation. The fifth chapter concludes with a summary and an outlook on further fields of research.

2 PROBLEM ANALYSIS

2.1 Architecture modeling in Systems Engineering

Systems engineering allows the holistic development of complex and networked technical systems (Gausemeier et al., 2014). The development and consideration of a design on different levels of abstraction above the individual technical domains helps to make the complexity of the overall system controllable for humans. Abstraction is used to disclose only relevant information. The system architecture is such a level or view that shows which functionalities are fulfilled by which systems (Haberfellner et al., 2019). The system architecture is addressed in this paper because of its high relevance to the early and holistic protection of a system with respect to safety and security (Nigam et al., 2019). A distinction is made between the synthesis phase, in which the system architecture is created, and the analysis phase, in which existing architecture constructs are validated.

Model-Based Systems Engineering (MBSE): According to the International Council on Systems Engineering (INCOSE), the future of systems engineering will be model-based. A model-based design approach can take advantage of a holistic system model to integrate multiple domains in a more precise, consistent and reusable format - in contrast to the document-oriented process (Walden et al., 2015).

The three main parts to describe a system model are a method, a tool and a **modeling language**. The modeling method is the guideline, which describes in which way the language elements have to be modeled. The modeling language, however, is only a means of expression and specifies the individual language elements that are available for system modeling. Finally, the modeling tool can be used to perform the modeling. Only a coordinated combination of the three parts leads to a successful system modeling. Especially the use of graphical modeling languages provides advantages in the areas of perception, maintenance, processing and communication between the departments. For this purpose, INCOSE and the Object Management Group (OMG) have developed the standardized Systems Model Language (SysML) for mapping and modeling the system model (Weilkiens, 2014).

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

As already explained, in order to describe a system model-based, a **modeling language** (e.g. SysML) is required. As shown in Figure 1, the modeling language is defined by the syntax and semantics (Dorociak, 2015). In detail, a modeling language consists of an abstract syntax, one or more concrete syntaxes and a semantics. The abstract syntax defines the different model elements together with their attributes and relationships to other model elements. The abstract syntax (also called **metamodel**) also contains the static semantics, which is mainly used to define constraints of the model elements and relations. The graphical representation is recorded in the concrete syntax, also called notation. For example, it can be defined that a system element should be represented by a blue rectangle. In semantics, the meanings of model elements are defined in a concrete model environment.
A great variety of domains exists, making the development of a modeling language that covers all existing aspects practically impossible. To meet this limitation, extension mechanisms (so-called profiles) are used to add new concepts and notations (Weilkiens, 2014). For this purpose, metamodels can be created and integrated into the existing modeling languages.

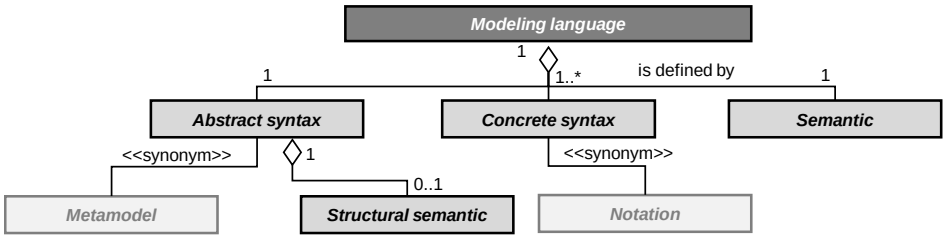

Figure 1. Structure of a modeling language according to Rodrigues da Silva (2015)

2.2 Safety and security in the development process

Safety and security are approached with different ways of thinking. While safety is about controlling catastrophic events caused, for example, by system malfunctions, security is about preventing malicious entities from attacking the system. The ISO standard 26262 (Road Vehicles - Functional safety) defines the term functional safety as "*absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems*" (ISO 26262, 2011). Safety is thus understood as the freedom from unacceptable risks in relation to dangers resulting from functional failure of the system. The term security is understood as the minimization of the weak points of assets and resources. Thus, the term security is not only understood as the protection against known hacker attacks, but also all possible external interventions from outside, which can damage the system. The counterpart to ISO 26262 is ISO 21434, which focuses on the cyber security of road vehicles. It uses the term Road Vehicle Cyber Security and defines it as "*Condition in which assets are sufficiently protected against threat scenarios to electrical or electronic components of road vehicles and their functions"* (ISO/SAE 21434, 2020).
The different ways in which safety and security are addressed in the development process are clearly reflected in the different methods (Nigam et al., 2019). **Safety methods** are used to safeguard against dangers, which arise, for example, from functional failures or unforeseen situations. Established methods include, for example: Fault tree analysis (FTA), component fault tree (CFT), dynamic fault tree (DFT) and failure mode and effects analysis (FMEA) (Dorociak, 2015). The **security methods** focus on the classification and prognosis of attacks, especially in the context of cyber security. Exemplary methods for identifying threats and their damage potential are the Threat Assessment and Remediation Analysis (TARA), Attack Tree (AT) or STRIDE (ISO/SAE 21434, 2020).

2.3 Need for action in safety and security integrated modeling languages

Due to the constant functional integration and networking of modern technical systems and their components, communication and network interfaces in particular are rapidly increasing. This results in a high system complexity, which in the development process can only be made manageable by approaches of Model-Based Systems Engineering. The importance of an early safety analysis in the development process is growing steadily (Pierre and Shawky, 2010). A purely safety- or security-oriented design and protection will no longer meet the increasing requirements. For example, security gaps can accumulate in the form of weak points and malfunctions that lead to errors in the smallest subsystems and result in unforeseeable effects in the overall system (Jäger, 2016). In order to anticipate these challenges in early phases of development and to be able to compensate for them with suitable measures, a modeling language for system architectures is required that models both safety and security-specific aspects (e.g.

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

error or attack paths) across disciplines. The modelling language is the basis for an integrative or combinatorial analysis and protection of security and safety and allows to detect synergies as well as conflicting effects or goals and to address them with suitable means at an early stage.

3 RELATED WORK

The analysed state of the art can be divided into basic modelling languages and profiles, as described in section 2.1. While basic modeling languages provide the fundamental possibility to represent system architectures and other aspects (requirements etc.), profiles allow the representation of specific concepts by consistently extending the basic modeling language.

3.1 Basic modelling languages

The graphical, model-based representation of a cross-disciplinary system model is realized by well-known modeling languages such as UML, SysML and CONSENS. Through the formally defined elements, system architectures can be represented in addition to aspects such as use cases or requirements. The Unified Modeling Language (UML) offers fourteen different diagram types to represent the structure and behavior of a system. The first seven diagrams can be used for the structural description of models, while the other diagrams are used for behavioral modeling (Rupp and Queins, 2012). The UML is used as the basis for the Systems Modeling Language (SysML). With regard to a standardized extension, elements were added and modified completely new. Elements of the UML that are irrelevant for Systems Engineering were explicitly excluded.
The specification technique CONSENS - "CONceptual design Specification technique for the Engineering of complex Systems" is used in the MBSE area for the development of highly complex systems. CONSENS has both a modeling language and a modeling method for the creation of the system model. The goal of the modeling language is to improve communication and cooperation between the various disciplines in the area of product design, as well as a holistic description of the system model. For this purpose, eight different views are addressed in CONSENS, one of which comprises the system architecture in the form of a so-called active structure (Gausemeier et al., 2014).
The basic modeling languages offer standardized language constructs with which system architectures can be specified across disciplines. A direct mapping of safety and/or security-specific aspects is not aimed at and can therefore only be realized by metamodel extensions in the sense of profiles.

3.2 Profiles / Extensions of modeling languages

The UML profile according to Fockel (2018) offers a model-based implementation of component fault trees (CFT) and thus a particularly good basis for safety-relevant analyses in the system architecture. The dedicated specification of failure types offers a better traceability of possible hazards and at the same time the possibility to evaluate effects on the system more transparently. Based on UML and SysML, the Electronics Architecture and Software Technology - Architecture Description Language (EAST- ADL) was developed for the automotive sector. Regarding functional safety according to ISO 26262 and reliability, different elements are predefined in the Dependability Package (EAST-ADL, 2013; Hillenbrand, 2012). EAST-ADL enables the modeling of system behavior for safety analysis (e.g. FTA) and the possibility to point out the errors/failures. For error propagation, errors can be detected on the ports, which are linked with propagation links (Chen et al., 2011). The CONSENS profile by Dorociak (2015) is, analogous to the EAST-ADL, well suited for the architecture analysis of safety-relevant aspects, but does not offer a specification towards automotive. Another approach is the SysML profile by Biggs et al. (2018) by introducing safety-relevant stereotypes for safety aspects. However, the focus of the profile is not on architecture modeling, but rather on requirements engineering.
In the context of security, UMLsec comprises, according to Schmidt and Jürjens (2011), a UML extension that focuses on secure software development. Different stereotypes are defined, which allow e.g. the representation and classification of different communication channels and the assignment of criticality. SysMLsec according to Roudier and Apvrille (2015) is an extension of SysML and allows an integrated modeling of an attack tree. The goal is to identify threats more transparently at an early stage in order to design suitable countermeasures. The SysML profile according to (Oates et al., 2013) integrates a security perspective. The core of the profile is the Threat Agent Model. It describes how threats from the threat agents can follow and provide information about the vulnerabilities.

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

The analyzed state of the art shows the lack of a metamodel that allows an combinatorial application of safety and security analysis methods in early system architecture modeling. The addressing of safety or security-specific aspects, which are used in the established methods presented in section 2.2, is done exclusively in isolation. Consistent integration in the form of a holistic metamodel is required.

4 METAMODEL FOR SAFETY AND SAFETY INTEGRATED SYSTEM ARCHITECTURE MODELING

As already explained, a metamodel is required for the early protection of systems to be developed, which addresses safety and security integratively and thus allows a combinatorial applicability. In order to specify the metamodel, the OMG's Meta Object Facility (MOF) metamodeling language is used in the following. This language uses associations, data types, classes and packages and allows the formal structure of a metamodel analogous to UML. As shown in Figure 2, the metamodel (SystemArchitectureModel) developed in the context of this paper comprises a synthesis model (SynthesisModel) and an analysis model (AnalysisModel). The **synthesis model** contains architectural elements (ArchitectureElement) and architectural connections (ArchitectureConnection), which are required for the synthesis of the system architecture. The basis for deriving these class groups is provided by established approaches and basic modeling languages like SysML (Weilkiens, 2014). The **analysis model** consists of the trigger model and the error and attack propagation model. With the TriggerModel all classes are addressed, which represent possible disturbances in and on the system. The ErrorAttackPropagation, on the other hand, enables the modeling of the effects in the architecture. Thus, possible error and attack propagations in the complex architecture can be reproduced integratively. Several approaches were used for this purpose, which will be explained in the following sections.

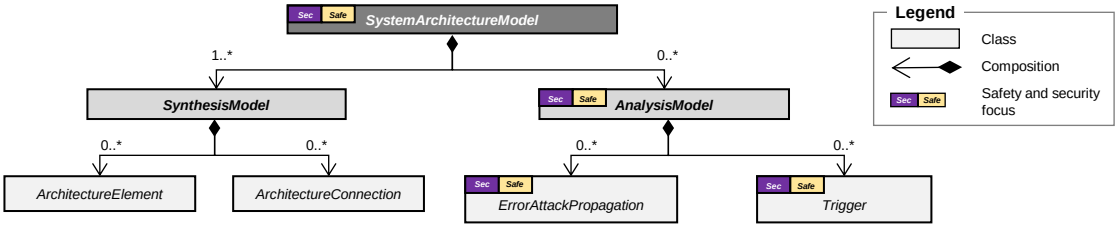

Figure 2. Overview of the metamodel

4.1 SynthesisModel

The synthesis model presented in detail in the following allows the modeling of the system architecture. Architectural modeling requires various architectural elements (ArchitecturElement), but also relationships (ArchitectureConnection) to represent the interactions between the individual elements (see Figure 3). **Architecture elements** are differentiated into **system elements** (SystemElement), **ports** (Port) or **port specifications** (PortSpecification). The system element represents all components of the system down to the atomic subsystem (e.g. brake caliper). A system element can contain any number of system elements as subsystems.

The system elements possess connection constructs in the form of ports to interact with other elements. The ports have a direction, which is set either in or out. A more detailed description of the ports is given in the port specifications. The Type attribute allows ports to be typified in the same way as technical relationships. The **technical relationships** comprise the various flows that flow between the individual system elements in order to represent technical interdependencies. These are subdivided into material, energy and information flows and can only represent directional connections. As a special feature, mechanical connections can be modeled in a dedicated way, which are of high relevance especially in terms of safety. For example, the destruction of a mechanical connection between two elements - intentionally or unintentionally - can have drastic effects on the entire system safety. The **allocation relation** (AllocateRelation) is possible without using the ports. Allocations comprise the "satisfied", "realized" and "implemented" relationship between system elements and other aspects in the system model. For example, the Satisfy relationship can be used to assign security goals (safety and security goals) to responsible elements. The Realizes relationship links the functions that are fulfilled by the elements. By means of the Implement relationship, the system elements are

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

assigned to the physical solution elements (e.g. concrete control units). The assignment of hazards and threats is represented by the Trace relationship.

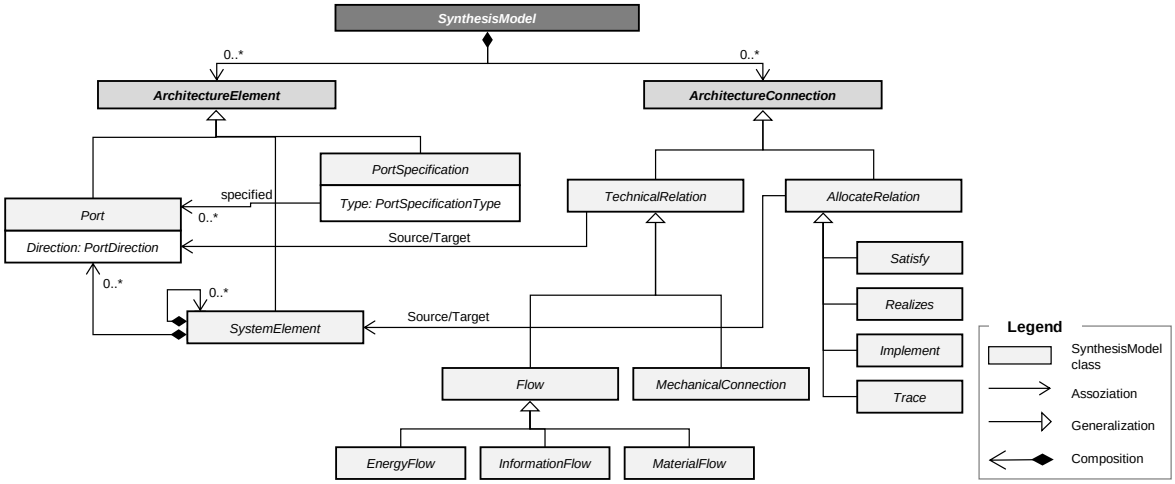

Figure 3. Overview of the SynthesisModel

4.2 AnalysisModel

Based on or integrative to the architecture synthesis, the architecture analysis is performed, in which synthesized constructs are examined, for example, with respect to vulnerability or the effects of fault propagation. The submodels that are necessary for the analysis are presented separately below.

4.2.1 TriggerModel

The TriggerModel includes the classes that allow the modeling of fault and attack possibilities on the system architecture. Within a system element **internal faults** (InternalFault) can be modeled. In the safety context, the InternalFault is strongly based on the error model of the EAST-ADL (2013) and the CONSENS profile according to Dorociak (2015). In order to also address security, further aspects have been added which are explained below.

The InternalFault represents the internal state of the respective system element. For this purpose, the attribute FaultOccurence is used to specify the random failure probability of the element. Furthermore the attribute HackImpact represents the effect of a hack attack on the random failure probability of a system element. The change of the random failure probability depends on the type of attack. Thus the different effects of a HackType on the target element of the architecture can be represented. For example, a DoS (Denial of Service) attack on a target element can lead to an internal failure due to the increased load and thus possibly trigger a safety-critical situation which the developer might not have considered from this point of view in advance.

Another essential attribute is the IndirectHackImpact, which in contrast to the general HackImpact addresses the indirect influences of an attack. The attack on a concrete target system is usually carried out via networked neighbouring systems. For example, a successful hack attack on the target element D is only carried out by the access path via elements A, B and C. This access and the continuation of the attack path has real effects on the indirectly affected systems. In case of information processing systems (e.g. electronic control unit) the access can lead to an increased probability of a system failure due to an unexpected resource demand. In terms of safety, this can result in an uncontrolled error propagation and finally in an overall system failure.

Safety-critical weak points can be specified in more detail for the ports of the system elements via **FailureModes** and **FailureSpecification**. FailureSpecification allows each input and output port to be assigned multiple failure modes including the allocation of threats/hazards. This class has been added following the UML profile according to Fockel (2018), which specifies five failure types: Omission (O), Commission (C), Crash (Cr), Value (V) and Timing Earlier (TE)/ Timing Late (TL). Security-critical triggers are modeled in the form of a **threat agent** (ThreatAgent) or using **environmental elements** (EnvironmentalElement). The ThreatAgent is modeled according to Steiner (2016) as a separate environmental element and specifies the attacker component. Steiner has developed an approach for the combined consideration of (component) fault trees with attack trees. The goal was to

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

enrich the safety analysis with suitable security aspects. Thereby, new safety-critical events are uncovered, which could be caused by a malicious attacker. According to Jones and Vidalis (2005), ThreatAgents include not only hacker attacks, but also natural disasters and malicious programs.

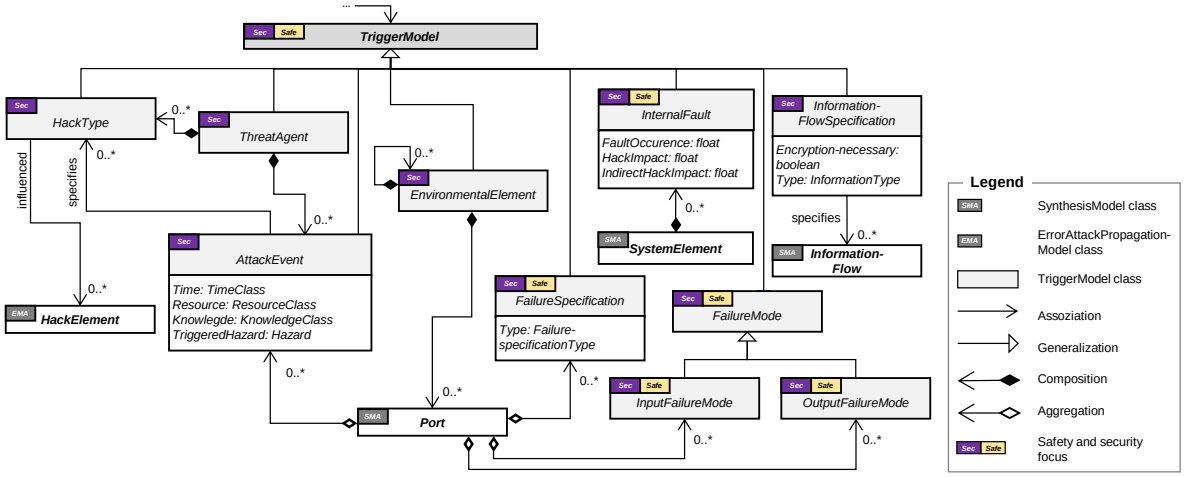

Figure 4. Overview of the TriggerModel

The ThreatAgent attacks are specified using the included **hack types** (HackType). The different hacker attacks can be differentiated into passive, active and physical. Various passive and active hacker attacks are classified in detail in the approach according to Hoppe (2014). The HackType can be further detailed via an attack event. This includes the required resources (Resource), the time needed (Time) and the required knowledge for an attack. The attributes are based on the Attack Feasibility Rating (8.7) of ISO/SAE 21434 Road vehicles – Cybersecurity engineering.

The **environmental element** (EnvironmentalElement) allows the modeling of all elements that lie outside the system context, with the exception of the attacker component. It possesses similar properties as the system element and allows the representation of attack path attacks through its relationship to other environmental elements (see section 4.2.2).

Finally, the information flow can be detailed by an **information flow specification** (InformationFlowSpecification) in the TriggerModel. Following UMLsec according to Schmidt and Jürjens (2011), it is possible to differentiate the type into LAN, WLAN, Bluetooth and Wire. Additionally, a boolean value can be used to model the necessity of information encryption.

4.2.2 ErrorPropagationModel

The last part of the overall metamodel covers error and attack propagation (ErrorAttackPropagation). The goal is to represent the effects of different attacks and errors on the system architecture. The metamodel is strongly based on the CONSENS profile according to Dorociak (2015) and the UML profile according to Fockel (2018). Both safety approaches address a component failure propagation (CFA). Regarding the security specific aspects (modeling of attack trees) the SysML-Sec according to Apvrille and Roudier (2014) was used as an essential reference.

The ports of a system element can have several **port states** (PortState), which can represent incoming and outgoing failures (see figure 5). The attribute State can be used to map the state “ok”, “partlyok” or “notok”. The port states can be both source and target of a “**can imply link**” (CanImplyLink). Both **gates** (Gate) and **hack elements** (HackElement) can be modeled in one system element. By means of gates and CanImplyLinks the different propagations through the architecture can be represented. The hack element is used to represent the access probability of the respective hack type. The connection between the attack event and the hack element is modeled by the **attack path** (AttackPath). In addition, an attack path between environment elements (EnvironmentalElement) and hacking element or attack event and environment element can also be represented, since threats can also occur via environment elements. The attack path was included analogous to the attack path analysis (8.6) of ISO/SAE 21434 (2020) and is of high relevance especially for the development of highly networked complex systems. In order to enable a risk classification already at system level, the system elements are extended by the concept of criticality. Criticality is determined by the ISO 26262 (ASIL) and the SAHARA method (SecL). By means of the integrative analysis and risk classification, the developer becomes aware which elements require special attention in the further detailing.

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

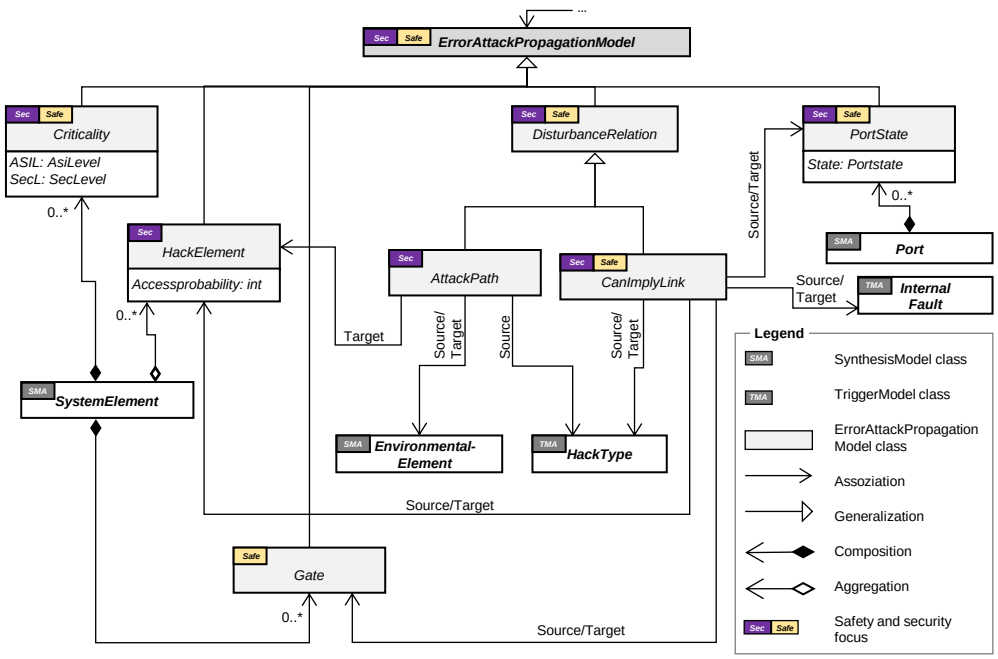

Figure 5. Overview of the ErrorAttackPropagationModel

5 VALIDATION

For the prototypical validation a highly simplified vehicle system is used, which is shown in figure 6. The following use case describes a security-critical intervention from outside, as well as the possible consequences in the vehicle system. The attacker component is specified as a criminal who wants to gain access to the detection sub-system (e.g. for environmental detection) of Vehicle1 in order to influence the braking subsystem in a safety-critical manner. Based on the architecture analysis, it becomes apparent that an attack vector exists via the WLAN of the infotainment subsystem.

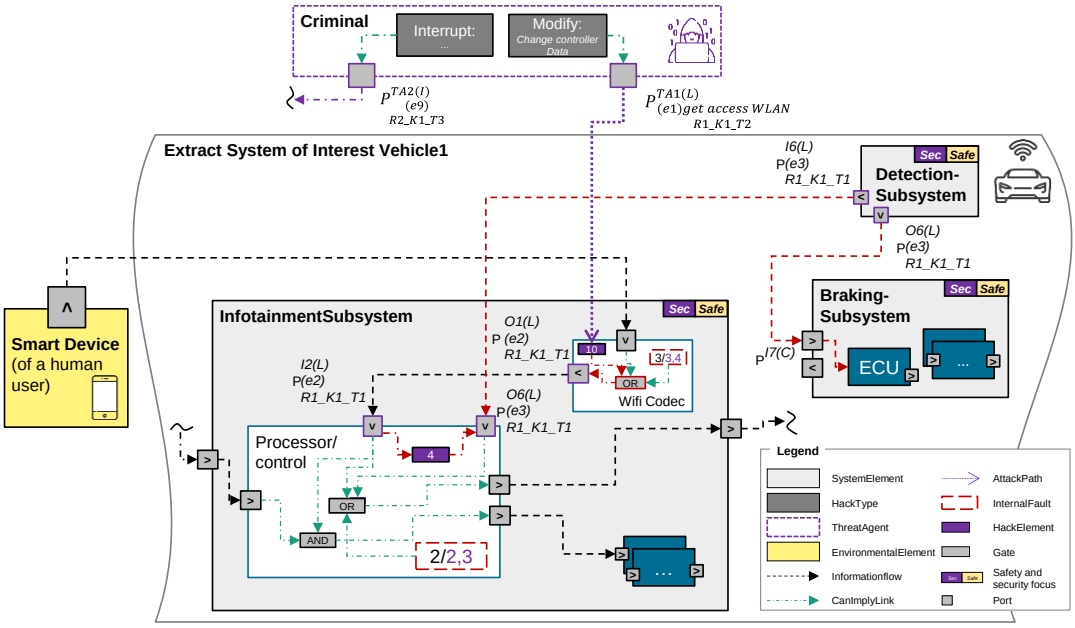

Figure 6. Validation example

The detected attack event can be specified in detail due to the required resources (R1), knowledge (K1) and the time required (T1). The first possible vulnerability lies in the elements of the infotainment subsystem that can communicate with the environment. The attack path leads via the Wifi codec, proccesor/control and the detection subsystem to the attacker's target. Each system element has a certain access and failure probability that can be calculated. To get a better understanding of the composition of the probabilities, a detailed overview is given in Figure 7. As explained in chapter 4, the internal fault of

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

each system element is composed of the three values: fault occurrence, hack impact and random hack impact. The fault occurrence specifies the random failure probability of a system element and is determined from past experiences (λ1). Before the failure probability can be determined by a direct hack, the access probability (hack element) to the system element must be determined at the beginning. For this purpose, the possibilities of a hack can be determined with reference to the attack feasibility rating of ISO 21434. Input for this analysis are the previously defined constructs threat agent, hack type and attack event. Alternatively, the fault of a system element can also be increased by an indirect hack attack (λ3), if an internal hack attack is performed on or via a connected neighboring system. In case of a successful access, the failure probability by a hack (λ2) is determined considering the hack type.

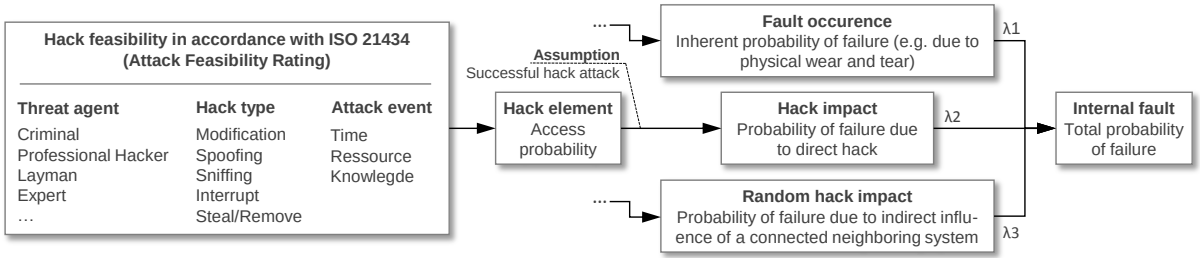

Figure 7. Calculation of probability of failure

If this event occurs successfully, the attacker succumbs to access to the control device in the infotainment sub-system, which enables him to gain access to the detection sub-system in the next step. Once this happens, he can manipulate the output data to modify the input data of the braking sub-system (Commission). This can result in safety-critical states that must be detected in early development stages in order to develop suitable countermeasures. It becomes obvious that the presented modelling language offers the possibility to represent much more complex attack and failure scenarios and to analyse them integratively regarding their security and safety consequences.

6 SUMMARY AND OUTLOOK

The technical systems of the future (e.g. autonomous vehicles) require a systemic development in which especially safety and security-compliant protection is integratively addressed in early development phases. In order to be able to detect and compensate for potential hazards and threats across domains, a modeling language for the system architecture is presented, which consistently links established safety and security approaches. Based on formally modeled constructs, the metamodel allows the transparent representation of complex interdependencies on an interdisciplinary, systemic abstraction level. In addition to supporting constructs for the synthesis of system architectures, attack and error propagation aspects including risk classification can be represented in the analysis phase. The integrative approach uncovers synergetic and, in particular, contrary goals and effects of architectural designs with regard to safety and security in order to make adequate architectural decisions based on trade-off analyses.

Further research is needed for the application of the developed modeling language. With regard to the definition of the system model, which forms the core of the MBSE, an appropriate method must be developed. Based on the identified classes, e.g. calculation algorithms for the determination of individual attributes have to be developed. Especially critical is the determination of the safety-critical failure probability, which is influenced by a potential external intervention (HackImpact). Furthermore, it is necessary to continuously examine further suitable safety and security analysis methods and to consistently address them in the metamodel. A key challenge of the application is the systematic identification of sources of attacks and errors. In order to enable the developer to effectively and efficiently model the architecture of secure and safe systems, it is advisable to structure solution patterns of attacks, failures and countermeasures in the form of databases in analogy to the metamodel. Furthermore, in order to uncover new forms of attacks and errors, it is necessary to develop an integrative method that promotes creativity in particular (e.g. design thinking). Finally, due to the current focus on the automotive sector, an application to other domains (e.g. cyber-enabled ships) is to be examined in further work by analyzing the corresponding challenges as well as established design methodologies.

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press

REFERENCES

Apvrille, L. and Roudier, Y. (2014), "Towards the Model-Driven Engineering of Secure yet Safe Embedded Systems", Electronic Proceedings in Theoretical Computer Science.

Biggs, G., Juknevicius, T., Armonas, A. and Post, K. (2018), "Integrating Safety and Reliability Analysis into MBSE: overview of the new proposed OMG standard", INCOSE International Symposium, Vol. 28 No. 1.

Chen, D., Johansson, R., Lönn, H., Blom, H., Walker, M., Papadopoulos, Y., Torchiaro, S., Tagliabo, F. and Sandberg, A. (2011), "Integrated safety and architecture modeling for automotive embedded systems", e & i Elektrotechnik und Informationstechnik, Vol. 128 No. 6.

Dorociak, R. (2015), "Systematik zur frühzeitigen Absicherung der Sicherheit und Zuverlässigkeit fortschrittlicher mechatronischer Systeme", Dissertation, Heinz Nixdorf Institut, Universität Paderborn, Paderborn.

Dumitrescu, R., Westermann, T. and Falkowski, T. (2018), "Autonome Systeme in der Produktion", Industrie 4.0 Management, Vol. 2018 No. 6.

EAST-ADL (2013), EAST-ADL Domain Model Specification, V2.1.12.

Fockel, M. (2018), "Safety Requirements Engineering for Early SIL Tailoring", Dissertation, Heinz Nixdorf Institut, Universität Paderborn, Paderborn.

Gausemeier, J., Ramming, F. J., Schäfer, W. (2014), *Design Methodology for Intelligent Technical Systems - Develop Intelligent Technical Systems of the Future*, Springer Berlin Heidelberg, Berlin, Heidelberg

Haberfellner, R., Weck, O.L. de and Fricke, E. (2019), Systems engineering: Fundamentals and applications, Birkhäuser, Switzerland.

Haskins, C. (2006), INCOSE - Systems Engineering Handbook - A Guide for System Life Cycle Processes and Activities, Version 3.

Hillenbrand, M. (2012), "Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen", Dissertation, Institut für Technik der Informationsverarbeitung, Karlsruher Institut für Technologie (KIT), Karlsruhe.

Hoppe, T. (2014), "Prävention, Detektion und Reaktion gegen drei Ausprägungsformen automotiver Malware - Eine methodische Analyse im Spektrum von Manipulationen und Schutzkonzepten", Dissertation, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Magdeburg.

ISO 26262 (2011), Road vehicles - Functional safety, ICS 01.040.43 No. 43.040.10 No., Beuth Verlag, Berlin.

ISO/SAE 21434 (2020), Road vehicles — Cybersecurity engineering, ICS: 43.040.15.

Jäger, T. (2016), "Safety und Security", available at: https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2015-gsem/jaeger/bericht.pdf (accessed 12 August 2020).

Jones, A. and VIdalis, S. (2005), Analyzing Threat Agents & Their Attributes, Information Security Consultant Geo-Bureau.

Macher, G., Armengaud, E., Brenner, E. and Kreiner, C. (2016), "Threat and Risk Assessment Methodologies in the Automotive Domain", Procedia Computer Science.

Miller, C. and Valasek, C. (2013), Adventures in automotive networks and control units, Def Con, Vol. 21, pp. 260–264.

Nigam, V., Pretschner, A. and Ruess, H. (2019), Model-Based Safety and Security Engineering.

Oates, R., Thom, F. and Herries, G. (2013), "Security-Aware, Model-Based Systems Engineering with SysML", in Janicke, H. (Ed.), 1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013), Leicester, UK, 16 - 17 September 2013, British Computer Soc, Swindon.

Pierre, D. and Shawky, M. (2010), "Supporting ISO 26262 with SysML, Benefits and Limits", Proceedings of European Safety and Reliability, ESREL 2010.

Rodrigues da Silva, A. (2015), "Model-driven engineering: A survey supported by the unified conceptual model", Computer Languages, Systems & Structures, Vol. 43.

Roudier, Y. and Apvrille, L. (2015), "SysML-Sec - A Model Driven Approach for Designing Safe and Secure Systems", in Hammoudi, S. (Eds.), Model-Driven Engineering and Software Development: Third International Conference, MODELSWARD 2015, Angers, France, February 9-11, 2015, Revised Selected Papers, Communications in Computer and Information Science, 1st ed. 2015, Springer International Publishing, Cham, s.l.

Rupp, C. and Queins, S. (2012), UML2 glasklar: Praxiswissen für die UML-Modellierung, 4., aktualisierte und erw. Aufl., Hanser, München.

Schmidt, H. and Jürjens, J. (2011), UMLsec4UML2 - Adopting UMLsec to Support UML2.

Steiner, M. (2016), "Integrating Security Concerns into Safety Analysis of Embedded Systems Using Component Fault Trees", Dissertation, Fachbereich Informatik, Technische Universität Kaiserslautern, Kaiserslautern.

Walden, D.D., Roedler, G.J., Forsberg, K., Hamelin, R.D. and Shortell, T.M. (Eds.) (2015), Systems engineering handbook: A guide for system life cycle processes and activities, INCOSE-TP-2003-002-04, 4. edition, Wiley, Hoboken, NJ.

Weilkiens, T. (2014), Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur, 3., überarb. und aktualisierte Aufl., dpunkt.verl., Heidelberg.

https://doi.org/10.1017/pds.2021.464 Published online by Cambridge University Press