Integrating Attack and Fault Tree to Support Safety and Security Co-Analysis in the Automotive Domain

1st Victor Luiz Grechi
Universidade de São Paulo
São Carlos, Brazil
victor.grechi@usp.br

2nd André Luiz de Oliveira
Universidade Federal de Juiz de Fora
Juiz de Fora, Brazil
andre.oliveira@ufjf.br

3rd Barbara Gallina
Mälardalen University
Västerås, Sweden
barbara.gallina@mdu.se

4th Leonardo Montecchi
Norwegian University of Science and Technology
Trondheim, Norway
leonardo.montecchi@ntnu.no

5th Rosana T. Vaccare Braga
Universidade de São Paulo
São Carlos, Brazil
rtvb@icmc.usp.br

***Abstract*—Integrating safety and security in automotive cyber-physical systems (CPS) domains (e.g. autonomous vehicles), is challenging for two main reasons. First, it is still difficult to represent the potential consequences of system failures or malicious attacks. Secondly, these systems must ensure safety and security despite unknowns and uncertainties. A Digital Dependability Identity (DDI) can facilitate this by encapsulating all dependability characteristics (e.g., design, requirements, safety, and security analysis models) of CPS's components. The Open Dependability Exchange (ODE) metamodel is an implementation of the DDI concept, but has limitations in the interplay between safety and security. ODE is aligned with with ISO 26262 but lacks certain security concepts to be aligned with ISO 21434. Also, ODE supports modeling fault trees, but modeling attack trees and attack-fault trees still not. This paper proposes an extension to the ODE metamodel, aiming to increase coverage of ISO 21434 concepts and allowing the modeling of attack-fault trees. We built these metamodel extensions based on an analysis of the ODE metamodel, industry standards, Microsoft STRIDE model, and HEAVENS security analysis methodologies. We evaluated the proposed extensions in an illustrative example of an autonomous vehicle.**

***Index Terms*—Safety and Security Co-analysis, Attack-fault Trees, Threat Analysis and Risk Assessment, Open Dependability Exchange (ODE).**

I. Introduction

Safety-related systems in cyber-physical system (CPS) domains, such as autonomous vehicles [1] and aviation [2], are becoming increasingly critical due to the integration of components from various manufacturers and their responsibility to perform safety-critical functions that, if compromised, can harm people or disrupt critical infrastructures with catastrophic consequences to people and society [3], [4].

Ensuring correct CPS operation is essential for the safety of passengers and other road users. Thus, dependable CPS design must prevent failures caused by software/hardware design faults or malicious attacks. Safety engineering addresses random and systematic faults, while malicious attacks and vulnerabilities are managed during security analysis. However, interdependencies between safety and security can lead to conflicting requirements. For example, security measures such as cryptography can introduce time delays, conflicting with safety requirements such as worst-case execution time [5].

Safety and security engineering teams often work independently, with limited interaction. They follow different standards (e.g. ISO 26262 [6] for safety and ISO 21434 [7] for cybersecurity in the automotive domain), which involve different processes, terminology, and artifacts. Despite these differences, both disciplines rely on a common system architecture and tree-based representations to model information flow [8], [9]. The Digital Dependability Identity (DDI) helps bridge these differences by encapsulating design, safety, and security analysis models of a CPS component [10].

The Open Dependability Exchange (ODE) metamodel [3], [11] implements the DDI concept, enabling the exchange of design, requirements, domain, safety, and security information between CPS components. The Executable Digital Dependable Identity (EDDI) is built on DDI, aiming to be executable at runtime. It provides a model-based, data-driven solution for real-time dependability assurance in multi-robot systems [2]. ODE leverages both DDI and EDDI to support dependable information exchange across design-time and runtime contexts.

ODE supports ISO 26262 by providing the *ODE:: Dependability::HARA* and *ODE:: Dependability::FailureLogic* packages. The HARA package includes elements to assess severity, likelihood (exposure), and controllability, which help classify risks under a given Automotive Safety Integrity Level (ASIL) [8]. The *FailureLogic* package includes sub-packages for common safety analysis techniques, such as Fault-Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), and Markov Chains. However, ODE lacks support for security concepts essential for safety and security co-analysis. Those security concepts include damage and threat scenarios from

ISO 21434 [7], the CIA triad (Confidentiality, Integrity, Availability) [12], [13], and the STRIDE model (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege) [14]. These gaps limit the modeling of fault, attack, and attack-fault trees, and consequently restrict the use of DDI concepts within the ODE specification.

This paper introduces ISO 21434-aligned extensions to the ODE metamodel. The proposed extensions aim to assist the research community and industry in exploring the full potential of ODE in supporting safety and security co-analysis of automotive CPSs using interconnected fault, attack, and attack-fault trees. Our extension includes the addition of new metamodel elements and associations to the ODE metamodel in accordance with industry standards. We built this metamodel extension based on an analysis of the ODE metamodel, ISO 26262, ISO 21434, STRIDE, and HEAVENS security analysis methodologies [13] to bridge the gap between safety and security concepts.

The remainder of this paper is organized as follows: Section II provides background information on safety and security analysis and the ODE metamodel. Section III discusses the limitations of ODE addressed in this paper. Section IV presents the proposed modifications to the *ODE::Dependability::TARA* package. Section V applies our extension to an illustrative CPS example in the automotive domain, showing how to create an attack-fault tree using ODE. Section VI reviews related work on metamodels for safety and security co-analysis. Finally, Section VII presents conclusions and future work.

II. Background

In this section, we provide an overview of tree-based safety and security analysis models, security concepts, and the ODE metamodel required for the reader to understand the proposed extension to ODE within the scope of this paper.

A. Tree models for safety, security and their interplay

The Fault Tree (FT) model used in the Fault-Tree Analysis (FTA) technique allows the representation of the correlations between a higher-level primary event (system failures, i.e., hazards) and the basic events, which can either be independent (basic events) or dependent (intermediate events) on other events [15]. Kaiser et al. [16] state that the usage of FTA proves beneficial in helping engineers identify potential causes and impactful factors for a hypothesized failure by iteratively tracing back the causal chain. As for the analysis part, qualitative analysis can be carried out by identifying sets of basic events leading up to the top event, while quantitative analysis helps calculate the top event probability based on the tree structure. The application of FTA is recommended or even required by safety standards such as ISO 26262 [16].

The attack tree (AT) model used in the Attack Tree Analysis (ATA) is similar to FT by employing nodes to represent attacks and describes pathways of attacks through the system [17]. However, while FTA is centered around safety properties, ATA considers the skills, resources, and risk appetite of an attacker [18]. The domain ontology (i.e., safety and security goals for an application domain) of FTs and ATs is what sets them apart as formalisms. These models differ on the attributes of the leaves (e.g., mean time to failure in FTs and vulnerabilities for ATs) and types of gates (e.g., SPARE gates in FTs and SAND gates in ATs) [19].

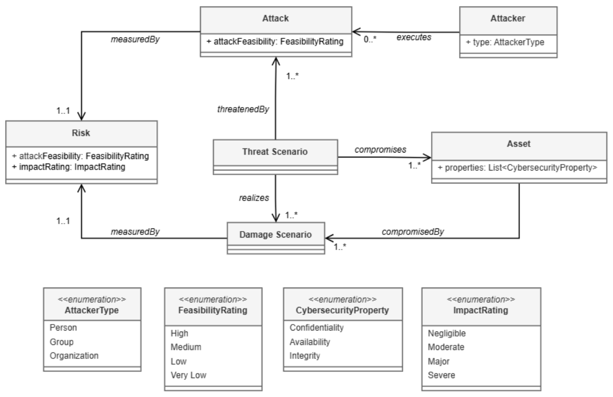


Fig. 1. Illustrative model of ISO/SAE 21434.

Attack-Fault Trees (AFT) merge the safety features of FT with security aspects from AT, making it possible to break down a top-level goal into smaller ones [9]. AFT can be modeled by incorporating FTs and ATs. An AT model can be incorporated into a FT model using an OR gate if the top event of the AT (i.e., its goal) is the same as an FT event [15]. The ODE metamodel supports the specification of fault trees, but has limitations concerning AT and AFT modeling. Those limitations relate to how an attack (security concept) is associated with a failure (safety concept).

B. Security concepts

Here, we highlight the security concepts used throughout this paper, starting with threat and damage scenarios. A threat scenario is a potential result when an attacker exploits vulnerabilities that lead to risks to the system's assets [13]. According to Lautenbach et al. [20], *"threat scenarios describe a set of actions that lead to one or more damage scenarios,"* where damage scenarios specify the result of an attack. In addition, their proposed HEAVENS 2.0 methodology has a step called "Threat scenario identification", which complies with the industry standard ISO 21434 [7]. We identify a chain of relations here: one or more attacks can represent a threat scenario to a system, and one or more threat scenarios can lead the system to the same damage scenario.

According to ISO/SAE 21434 [7], a damage scenario is an adverse consequence involving a vehicle or vehicle function affecting a road user. The estimation of damage or physical harm is named "impact", which is a required parameter, along with "attack feasibility", for the estimation of "risk". Figure 1 presents an illustrative model of the ISO/SAE 21434 concepts used in this work.

The STRIDE classification is a threat model defined in the Microsoft security development lifecycle. STRIDE categories help security engineers to identify threats based on what the

TABLE I
RELATIONSHIP BETWEEN STRIDE CLASSIFICATIONS AND IAS OCTAVE ATTRIBUTES.

Threat	Attribute
Spoofing	Authenticity
Tampering	Integrity
Repudiation	Non-repudiation, Accountability and Auditability
Information Disclousure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

attacker is trying to achieve instead of thinking about endless possibilities of attacks and attack techniques. Thus, this categorization avoids scenario explosion [13]. The STRIDE acronym stands for: *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege* [21]. Each STRIDE threat category compromises a given cybersecurity property of one or more assets, for example, spoofing (threat) of a vehicle sensor (asset) violates authenticity. A cybersecurity property is an attribute of an asset that can be worth protecting, e.g. Confidentiality, Integrity, and Availability (CIA). The CIA triad is often referred to as primary security attributes. However, in 2013, the Information Assurance & Security (IAS) Octave was proposed as an extension to the CIA triad, and five new attributes were included: authenticity, authorization, non-repudiation, accountability, and auditability [13].

Harm, impact, and risk are somehow present in the ODE metamodel, but adjustments must be made to ensure compliance with the ISO 21434 cybersecurity standard. In addition, STRIDE and IAS Octave are important classifications that can lead to a more robust analysis. Table I highlights the relationships between the threat categories of STRIDE and the cybersecurity properties of IAS Octave.

C. Open Dependability Exchange metamodel

In open and adaptive systems, systems and components need to exchange dependability (e.g., safety and security) information with other systems they are connecting to (see Section I). However, according to Zeller [3], the review, extraction, and relation of safety/security information between suppliers are usually manual, highly time-consuming, and error-prone. Safety and security information should be expressed in a common and interoperable language.

The Open Dependability Exchange (ODE) metamodel [11] provides the basis for representing and exchanging safety and security information between open-adaptive systems. With its metamodel elements, engineers can describe the system design, requirements, domain, safety and security information in the format of assurance cases.

An assurance case is the heart of the DDI, a structured, modular, and hierarchical model of system dependability properties [2]. The DDI, created from the ODE metamodel, carries all this information within the system or component so that dependability can be assured at design time and runtime (i.e., when the DDI is extended to be executable).

1) ODE Packages: The first version of ODE is organized into packages that provide means to express, connect, and communicate safety information, such as: architectural modeling (sufficient to express safety information) through the *ODE::Design* package; failure logic modeling (e.g., using FTA, FMEA, and Markov Chain techniques) through the *ODE::FailureLogic* package; hazard and risk analysis through the *ODE::Dependability::HARA* package (based on ISO 26262 [6]); domain standards and assurance levels through the *ODE::Dependability::Domain* package; and requirements through the *ODE::Dependability:: Requirements* package [3]. In Table II, we present an illustrative selection of metamodel elements from each package.

In the remainder of this paper, we will use the italic font style to represent ODE packages and the bold font style to represent metamodel elements.

2) ODE Profile V2.: The second version of ODE introduces security concepts primarily found in the *ODE::Dependability::TARA* package (referred to as the *TARA* package in the remainder of this paper). This new package contains elements for conducting security analysis, which are detailed in the following. The **Asset** element and its specialization **VulnerableItem** are valuable objects potentially targeted by attackers (i.e., **ThreatAgent**), who can exploit weaknesses of the system (i.e., **Vulnerability**) through an **Attack** [7]. The ODE metamodel ODE Profile V2 also has the **AttackerGoal** element, even though this concept is not presented in ISO 21434. The **AttackerGoal** element represents a **SecurityRisk** to the system and is addressed by **SecurityCapabilities**, which **SecurityControls** implement. These last two elements are high-level and low-level countermeasures, respectively. In this version, the **Failure** element from *ODE::FailureLogic* package receives a specialization named **SecurityViolation**, caused by one or more **Attacks**. This last addition aims to enable users to model the adverse consequences of security **Attacks** on the system.

Although the ODE metamodel completeness around security concepts, there are still limitations that must be overcome to make this metamodel ISO 21434-compliant. In the next section, we discuss each limitation of the *TARA* package around those concepts and the FT specification.

III. ODE LIMITATIONS

As mentioned in Section II-C, the ODE metamodel can be beneficial when used to support safety and security co-analysis due to its wide range of modeling concepts and package structure, along with the exchange capability due to the DDI concept it is built upon. However, we identified limitations in the *TARA* package concerning security concepts needed to support safety and security co-analysis. In the following sections, we highlight those limitations and the rationale behind them.

A. Security Concepts

First, as its name ”Threat Analysis and Risk Assessment” suggests, we should be able to use the metamodel when

TABLE II
METAMODEL ELEMENTS FROM EACH PACKAGE (ILLUSTRATIVE SELECTION).

Package	Elements
ODE::Base	BaseElement and TimeUnit
ODE::Design	System, Context, Port, Function, and Signal
ODE::FailureLogic	Failure, SecurityViolation, and FailureModel
ODE::FailureLogic::FTA	FaultTree, Cause, and Gate
ODE::FailureLogic::FMEA	FMEA and FMEAEntry
ODE::FailureLogic::Markov	MarkovChain, Transition and State
ODE::Dependability	Measure and MaintenanceProcedure
ODE::Dependability::HARA	Hazard, RiskParameter, and RiskAssessment
ODE::Dependability::Domain	Standard and AssuranceLevel
ODE::Dependability::Requirements	RequirementSource and DependabilityRequirement

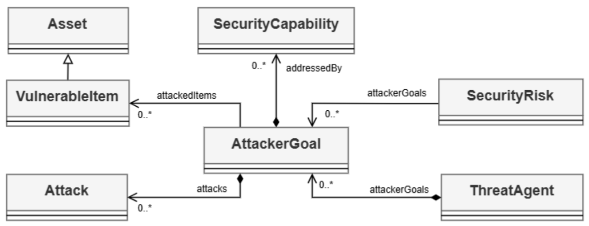


Fig. 2. Current modeling of *ODE::TARA* package.

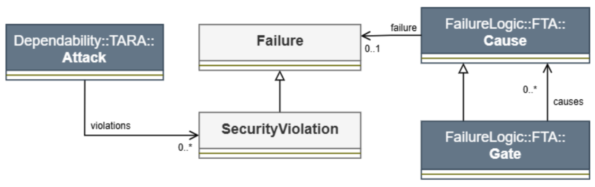


Fig. 3. Current modeling of *ODE::FailureLogic* package.

analyzing threats and assessing risks. However, we do not have a representation for ”Threat”. We understand that the positioning of ”Threat” in the ODE metamodel is now occupied by the **AttackerGoal** element, due to associations with **VulnerableItem** (**Asset**), **SecurityCapability**, **SecurityRisk**, **ThreatAgent**, and **Attack**, as presented in Fig.2. Moreover, a threat can be enhanced with Microsoft STRIDE classification to help engineers not think about endless possible threats that hinder the analysis (see Section II-B).

The IAS Octave security attributes are other important but missing classifications (see Section II-B). These attributes represent security properties that a security requirement must protect. The **SecurityRequirement** is represented in ODE, but it does not relate to any security attribute. Along with identified threat categories, these attributes can be beneficial when defining security requirements.

In security, harm and risk assessment can only be expressed if we use ODE to model the chain of relationship composed of **Hazards**, **Failures**, **SecurityViolations**, **Attacks**, **AttackerGoals**, and **SecurityRisk**. Although it is possible and reasonable in safety and security co-analysis, this current modeling restricts security analysis to be conducted in isolation.

B. Tree Models

Fig.3 presents the current ODE modeling that supports the tree model specification. All nodes of the tree model are **Causes**. The **Cause** element can be linked to at most one **Failure**, which allows us to represent **Failure** with a node of the tree model. For a **Cause** to be linked to multiple **Causes**, it has to be specialized as a **Gate**. As a result, the possible nodes of the tree model include both **Causes** and **Gates**. Since the **Cause** becomes a **Gate**, it is not associated with **Failure**, which justifies the association with zero **Failures**.

According to ODE Profile V2 [11], the **SecurityViolation** element enables the modeling of the direct effect a security **Attack** has on the system by inheriting from **Failure**. However, this element is absent in the ISO 21434 glossary [7]. Avizienis et al [12] classify faults as internal or external, which is supported by the ODE metamodel. The authors also state that, in security terms, an attack is a malicious external fault, but the ODE metamodel does not model failures and attacks as parent and child elements. This connection is an association when it should be an inheritance. Furthermore, this current modeling does not allow the specification of **Attacks** as basic events in tree models. We highlight this as a crucial limitation since it hinders engineers when conducting safety and security co-analysis through the correct modeling of FT.

As stated by Andre et al. [18], in an FT, the nodes of the tree model could be either basic component failures or basic attack steps. Using ODE, we should be able to represent a **Failure** and an **Attack** as nodes of the tree, but that is not the case. Also, according to Hayakawa et al. [22], security attacks are performed by exploiting system vulnerability, leading to a failure, connecting an attack tree to a fault tree. We should be able to replace a **Failure** with an attack tree, making the **Failure** the top event of the attack tree. In traditional attack trees, the top event results from the attack, and the nodes refine this result [23]. Considering the current version of the ODE metamodel, **Attacks** can not be linked with **Gate** elements, which weakens the propagation analysis that users could achieve with the FTA technique. A suitable representation of an FT is illustrated in Fig.4.

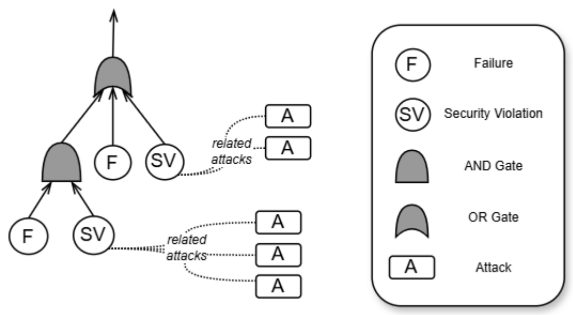


Fig. 4. A representation of an AFT achievable with current ODE.

IV. Proposed Extension

This study does not propose a visual notation for tree-based dependability analysis models. Since the ODE metamodel provides the abstract syntax of system design, safety, reliability, and security analysis models, we focus on extending the *TARA* package to support safety and security analysis aligned with ISO 21434 standard. The proposed metamodel extension may guide the development of tooling to support safety and security co-analysis in compliance with dependability standards. The concrete syntax in the specification of an AFT used in this study is based on the FT notation and was used for illustrative purposes. The proposal of a visual notation for AFT is another potential contribution subject of further work.

Our extension to the ODE metamodel is restricted to the *TARA* package. We will present all modeling modifications, shown in Fig.5, using the letter M followed by a sequence number (e.g., M1). Each modification was designed and built by taking into account the background (see Section II) and ODE limitations (see Section III) presented in this paper.

In **M1**, a **ThreatScenario** element is created and set as a substitute for the **AttackerGoal** element. As pointed out in Section II-C1, an **AttackerGoal** is not a concept presented in ISO 21434 but is equivalent to a **ThreatScenario**. This new element has the "category" attribute, typed as a **ThreatCategory**, an enumeration created based on the Microsoft STRIDE threat analysis model (see Section II-B). We kept previous relationships with **ThreatAgent**, **Attack**, **VulnerableItem** (a specialization of **Asset**), and **SecurityCapability**. This last element is related to **SecurityRequirement**, which received the attribute cybersecurityProperty, typed as a **CybersecurityProperty**, in our **M2** modification. All possible values for **ThreatCategory** and **CybersecurityProperty** are presented on the left side of Fig.5.

Modification **M3** introduces the **DamageScenario** as a concrete situation raised by a **ThreatScenario**. The estimation of risk for the **DamageScenario** is present in the **SecurityRisk** element, which has properties for likelihood and impact on assets, individuals, and businesses. Since a DamageScenario can cause harm, we established an inheritance with **Hazard** element from the *ODE::Dependability::HARA*.

As presented in Section III, the current ODE metamodel prevents specifying **Attacks** as basic events in an attack or attack-fault trees. We aim to solve this issue by proposing modifications **M4** and **M5**. To be aligned with the ISO 21434 cybersecurity standard, in **M4**, the **SecurityViolation** element was removed along with its inheritance from **Failure**. This modeling is refactored with **M5**, which adds the direct inheritance from **Failure** to **Attack**. The **Attack** element is a **Failure** with the attribute *originType* set as *INPUT*. Also, the modeling of the **Attack** element already has important attributes, such as *financialCost*, *timeRequired*, *difficulty*, *detectability*, and *feasabilityRating*, which makes this element conceptually more robust than just an external fault.

Finally, modification **M6** adds to the **Attack** element a relationship to itself, named "attackPaths". This modification aims to enable engineers to model attack trees to the desired granularity, refining attacks into attack steps until no more refinement is needed or desired [18].

V. Illustrative example: HAD Vehicle

This section presents the usage of our metamodel in supporting safety and security co-analysis of an automotive Highly Automated Driven (HAD) vehicle involved in a rear collision hazard. The architecture and scenarios were inspired by the work of Gallina et al. [24], Kruck et al. [8] and Sabaliauskaite et al. [25]. HAD is an example of an automotive cyber-physical system that demands dependability assurance since it is exposed to events critical for both safety (failures - e.g., braking system fault) and security (attack - e.g., jammed communication with external entities) due to its operation in uncertain environments.

A. Architecture

We present the architecture of this illustrative example in Fig.6. The Communication Unit (CU) component enables vehicle openness by allowing it to connect with the Central Gateway to receive over-the-air updates for the Vehicle Computer (VC) component. HAD allows users to update firmware via USB Stick (US), which requires physical access to the vehicle. Updated packages input from USB data or CU data are loaded by the On-Board Tester (OBT), which is responsible for flashing the provided software on the VC.

The vehicle also contains a LiDAR (Light Detection and Ranging) responsible for sensing the vehicle's immediate environment (e.g., distances to neighboring vehicles, road traffic conditions, and traffic signs) as well as its dynamics (e.g., location and speed). LiDAR sensors use laser pulses that bounce off nearby objects and reflect to the sensor, allowing object identification and recognition. This HAD Vehicle is also equipped with Cameras in four directions (front, rear, left, and right). Cameras seek to imitate human vision as closely as possible, which is used as a complement to LiDAR sensors. With environmental data, the VC can decide and manipulate the Brakes (B) braking force, the Powertrain (PT) torque, and the Steering (S) angle to adapt the vehicle to different

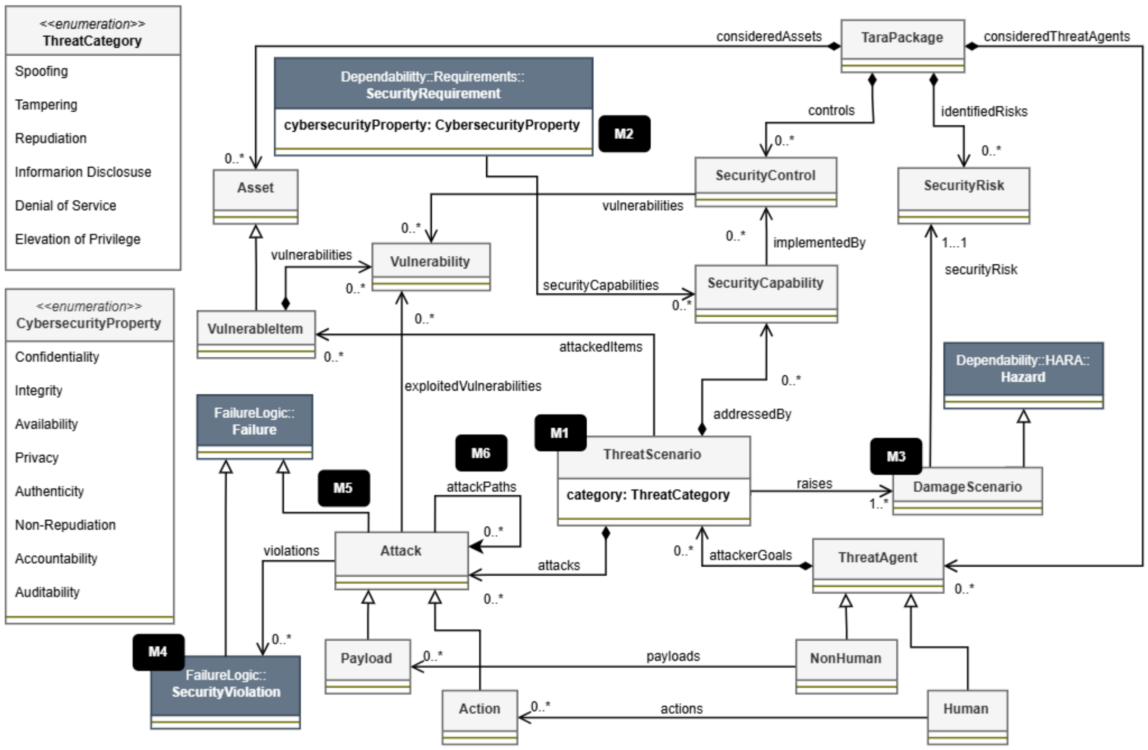


Fig. 5. Modeling modifications in the *TARA* package.

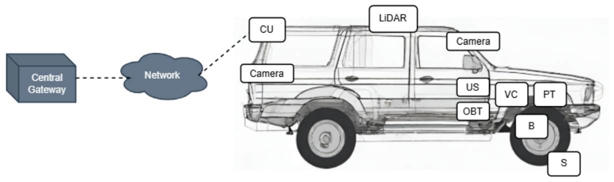


Fig. 6. Architecture containing the HAD Vehicle and related infrastructure.

situations. Fig.7 presents an internal block diagram of the HAD Vehicle.

B. Scenarios

This example focuses on three scenarios leading to a rear collision hazard, chosen to represent safety, security, and cybersecurity events. We aim to conduct a safety and security co-analysis using the proposed extensions. *Scenario 1*, ”blurring the camera image”, which represents a safety event due to an internal failure. In *Scenario 2*, the security event involves a ”malicious software update using the USB port”, requiring the attacker's physical presence. Lastly, *Scenario 3* represents a cybersecurity event with a ”malicious over-the-air update through network using spoofing techniques”.

C. Threat Analysis and Risk Assessment

This section presents how to instantiate the extended ODE metamodel as a UML Object Diagram for threat analysis and risk assessment, using ISO 21434 concepts within the extended *TARA* package. We do not intend to conduct the ISO 21434 TARA process in completeness.

The threat of unauthorized service modification is related to two different assets (i.e., LiDAR and CU), represented by *Scenarios 2 and 3*. Due to space limitations, we present only *Scenario 3* in this section, but it is complete enough to show the ODE metamodel in practice with our extension.

In *Scenario 3*, LiDAR sensors are spoofed (**ThreatCategory**) via an over-the-air update attack. Fig.8 presents this scenario in a UML object diagram. Due to the CU's lack of authentication (**Vulnerability**), it makes it possible for the attacker to send a malicious software update through the network and cause impact on the Vehicle (**VulnerableItem / Asset**). Once the LiDAR sensors are spoofed, the attacker can simulate an obstacle in the front of the car at high speed, leading to the violation of LiDAR sensors' data integrity (**DamageScenario**). This simulation causes the VC to brake abruptly to stop longitudinal movement, which can lead to a rear collision.

Risk estimation (**SecurityRisk**) considers the likelihood and the impact as parameters. Likelihood can be estimated with calculations over the attribute *attackFeasability* of all **Attacks** that compose this scenario. Impact is classified based on how it affects the asset, individuals, and business. We modeled those impacts as *high* since a rear collision can cause great damage to vehicles, passengers, and brand reputation. It is worth mentioning that this impact can be mitigated by implementing protection measures (presented in the following), and the

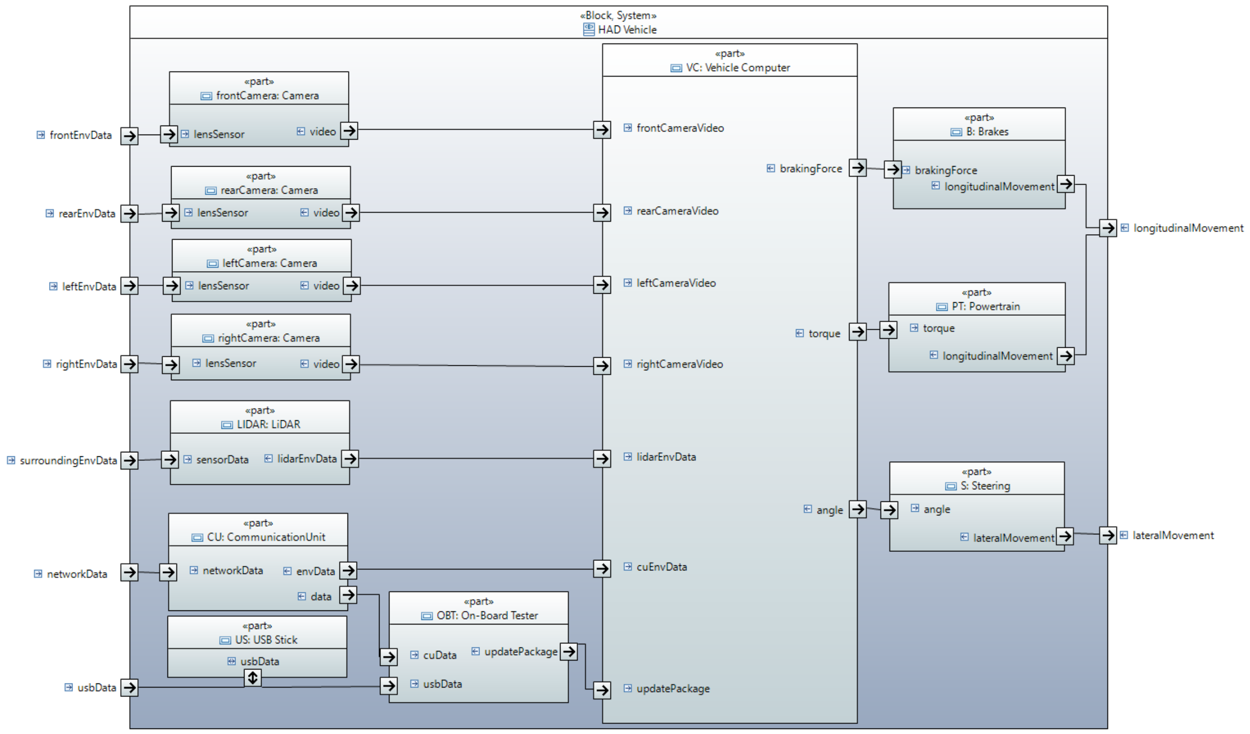


Fig. 7. HAD Vehicle components and communications.

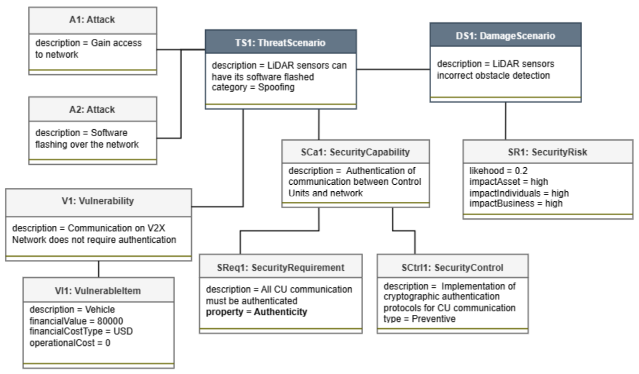

A1: Attack
description = Gain access to network
A2: Attack
description = Software flashing over the network
TS1: ThreatScenario
description = LiDAR sensors can have its software flashed
category = Spoofing
DS1: DamageScenario
description = LiDAR sensors incorrect obstacle detection
SCa1: SecurityCapability
description = Authentication of communication between Control Units and network
SR1: SecurityRisk
likehood = 0.2
impactAsset = high
impactIndividuals = high
impactBusiness = high
V1: Vulnerability
description = Communication on V2X Network does not require authentication
VI1: VulnerableItem
description = Vehicle
financialValue = 80000
financialCostType = USD
operationalCost = 0
SReq1: SecurityRequirement
description = All CU communication must be authenticated
property = Authenticity
SCtrl1: SecurityControl
description = Implementation of cryptographic authentication protocols for CU communication
type = Preventive

Fig. 8. Object diagram for the spoofing on LiDAR sensors threat scenario. Elements and attributes proposed by our contribution are highlighted in darker colors and bold font style, respectively.

vehicle still has four cameras that can act as a redundancy for LiDAR (which will be more evident in Section V-D, during Attack-Fault Tree Analysis).

This threat scenario can be avoided by assuring authentication (**SecurityRequirement**), which is implemented by implementing a cryptographic authentication mechanism (**SecurityControl**) for CU communication.

D. Attack-Fault Tree Analysis

After identifying damage scenarios, threat scenarios, and related attacks during the analysis using ODE with our extension, we now have the hierarchical relationships between safety-related failure and security-related attacks needed to specify a complete tree model.

First, let us clarify how to specify a tree model according to the ODE metamodel presented in Section III-B. Fig.9 presents five steps for specifying a tree model using ODE elements. All tree nodes must be **Cause** elements (*STEP 1*). The **Cause** element can not be associated with other causes unless it is specialized as a **Gate** element. In *STEP 2*, we represented the OR and AND gates, but ODE also supports the XOR, NOT, VOTE, PAND, POR, and SAND gates [9], [19], [26]. In the following steps, the **Cause** element remains as a circle, with a dashed line for **Failure** (*STEP 3*), and a dotted line for **Attack** (*STEP 4*). In *STEP 5*, modification **M6** facilitates developing an **Attack** into minor **Attacks**, as described in Section IV.

The model corresponding to our example is presented in Fig.10. The presented AFT illustrates an rear collision hazard as the top event. We included all scenarios presented in Section V-B. Representing safety events of *Scenario 1*, we have F3 (Front camera blurry image) and F4 (Front camera recognition system fault) connected to an OR gate, resulting in F2 (Front camera incorrect video stream value). For the representation of security events (i.e., attacks conducted in person) of *Scenario 2*, we have A5 (Gain physical access to the vehicle) and A6 (Software flashing using a USB Stick) connected to an AND gate, resulting in A2 (LiDAR software spoofed physically). The attack A5 can be developed into A7 (force door open) and A8 (Clone car key signal) connected to an OR gate. Finally, representing cybersecurity events (i.e., attacks conducted remotely) of *Scenario 3*, we have A3 (Gain access to network)

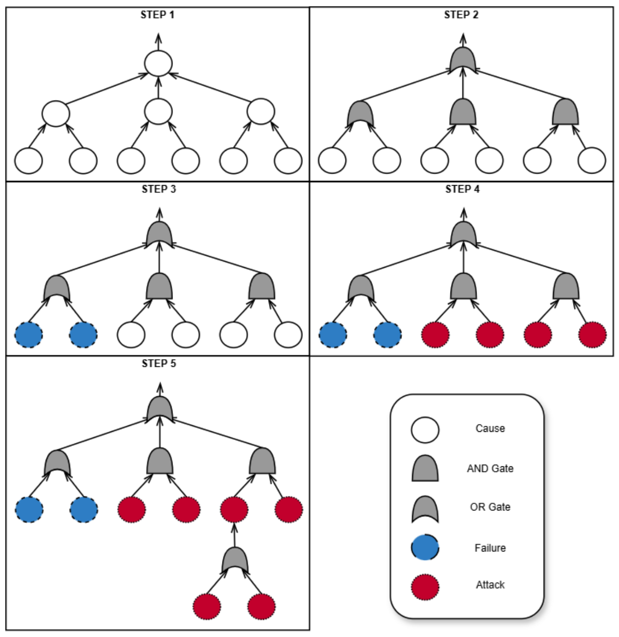


Fig. 9. A step-by-step process to specify AFT using the ODE metamodel. STEP 1: Specify tree model composed of only Cause elements. STEP 2: Replace Cause elements with its child Gate element where it applies. STEP 3 and 4: Associate Failure and Attack elements where they apply. STEP 5: Expand Attack elements if needed.

and A4 (Software flashing over the network) connected to an AND gate, resulting in A1 (VC software spoofed remotely).

The hazard H1 (Rear collision) is achieved when F1 (Unintentional Braking) occurs, and this can be due to the occurrence of F1, A1, or A2. A1 violates the VC software remotely, A2 violates the LiDAR sensor's data integrity (scenario explored in Section V-C), and failure F2 compromises the front camera video. Once the autonomous vehicle concludes that there is an obstacle in the front of the vehicle due to any of these events, unintentional braking is triggered and can cause a rear collision in many situations. As mentioned in Section V-C, cameras and LiDAR sensors can act as redundancy to each other in order to mitigate this kind of risk. However, we did not model these countermeasures due to space limitations.

VI. Related Work

As related work, we considered contributions that designed or extended a metamodel that relates concepts of safety and security, and contributions that integrate fault and attack trees. Most of the presented studies in this section were extracted from a systematic literature review on model-driven safety and security co-analysis [27].

Even though we conducted a safety and security co-analysis in Section V, this paper's goal is to extend the ODE metamodel in a conceptual contribution. We recognize the existence of other works that propose new methodologies, approaches, processes, or frameworks related to safety and security co-analysis. However, we did not mention them in this section as related work, since their contribution is not focused on metamodels and tree models as ours.

A. Metamodels

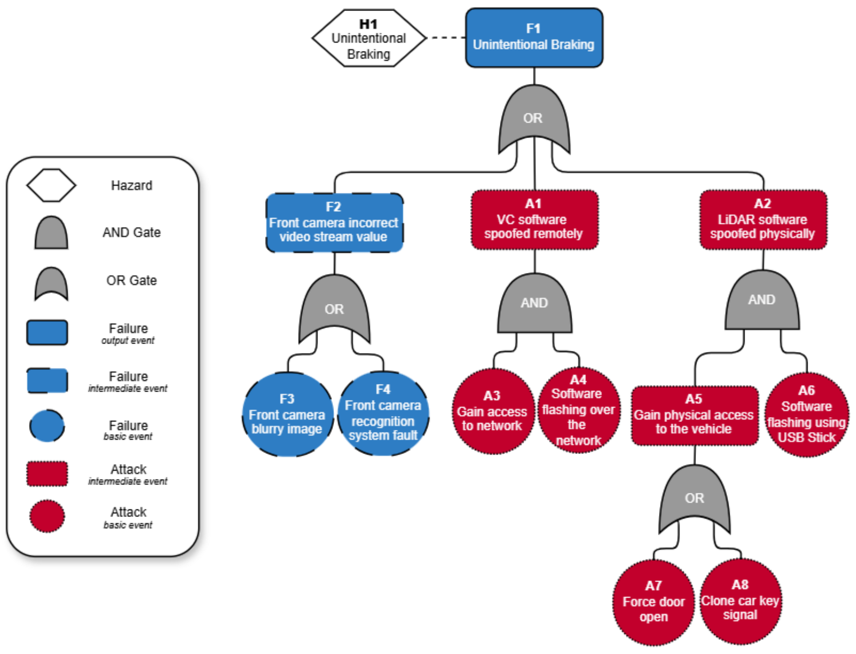
Bakirtzis et al. [28] propose a metamodel that addresses safety, security, and resilience. Their metamodel is built upon STAMP and mission-aware cybersecurity to create general connections between safety and security concepts. Sharing the same motivation as ODE, they were concerned about exchanging dependability information between systems and reusing models between modeling tools.

Kruck et al. [8] also proposed a metamodel relating safety and security concepts and a method to use it. They got inspiration from HARA described in ISO 26262 [6] for safety, building their metamodel over the fault-tree analysis and component fault trees. For security, they based their approach on the Modular Risk Assessment (MoRA) method, adapting the terminology to comply with ISO 21434 [7]. Their approach was implemented in a modeling tool where the system model is analyzed, and suggestions are presented to the user based on the metamodel relationships between safety and security concepts. Their contribution is a model-driven approach for safety and security co-analysis, but the proposed metamodel cannot be exchanged since it is coupled to a specific tool.

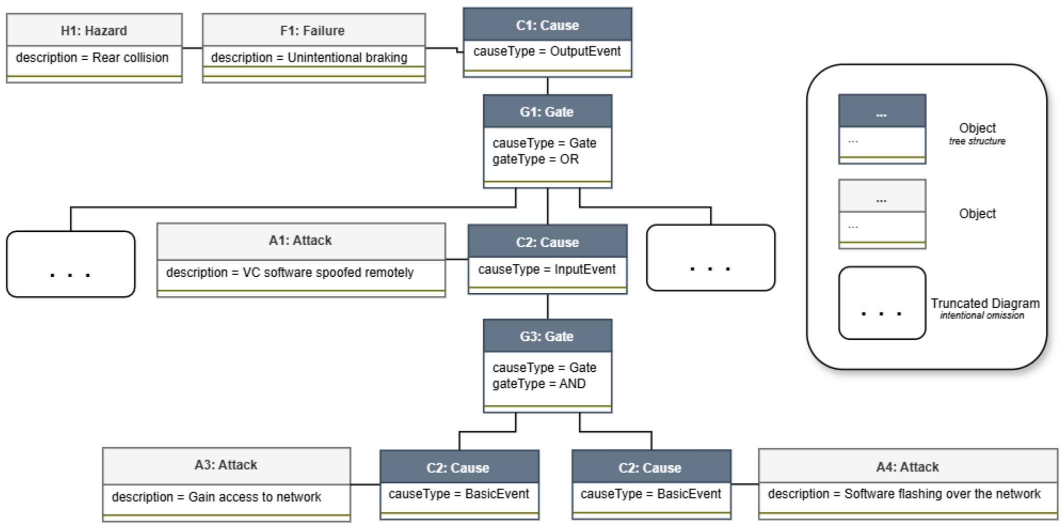
Gallina and Haider [29] extended the SafeConcert metamodel in the CHESS modeling tool for multi-concern modeling, focusing on alignment with ECSS (European Cooperation for Space Standardization) standards. Subsequently, Debiasi et al. [30] expanded the tool to support system-level safety and security analysis. The CHESS toolset uses the CHESSML modeling language profile based on OMG standard languages such as UML, SysML, and MARTE. Due to previous contributions, such as Failure Logic Analysis [31] and State-Based Quantitative Dependability Analysis [32], CHESS already supported safety analysis. With the contributions from Gallina, Haider, and Debiasi, CHESS now includes security modeling, but still lacks mechanisms to exchange and fully integrate dependability artifacts.

B. Integration of safety and security tree models

An enhancement of FTA with security concerns is proposed by Steiner and Liggesmeyer [33]. They use the STRIDE classification to highlight basic safety events that can be achieved by threatening security properties. Then, this safety event becomes the top event of an attack tree, allowing the integration of models. Kumar and Stoelinga [9] equipped AFTs with stochastic model checking techniques in a model-driven approach that enables both qualitative and quantitative analysis. André et al. [26] contribute with a framework that translates AFTs into parametric weighted timed automata within the ATTop tool and IMITATOR model-checker. This model-driven approach enables the analysis of the most feasible failure and attack scenarios. Other relevant studies connect FTs with ATs to form AFTs [15], [34]–[36]. However, none of these works presented in this section leverages the architectural modeling of the system as the ODE metamodel does, and their


H1 Unintentional Braking
F1 Unintentional Braking
OR
F2 Front camera incorrect video stream value
A1 VC software spoofed remotely
A2 LiDAR software spoofed physically
OR
AND
AND
F3 Front camera blurry image
F4 Front camera recognition system fault
A3 Gain access to network
A4 Software flashing over the network
A5 Gain physical access to the vehicle
A6 Software flashing using USB Stick
OR
A7 Force door open
A8 Clone car key signal
Hazard
AND Gate
OR Gate
Failure output event
Failure intermediate event
Failure basic event
Attack intermediate event
Attack basic event

(a) Tree model.



(b) Object diagram. Only one branch of the tree (the central one) is presented, while the left and right branches are omitted due to space limitations.

Fig. 10. AFT for the rear collision hazard using the proposed metamodel. AFT as an tree model is presented in fig. 10a and the metamodel concepts used to model it are presented in fig. 10b as an object diagram.

focus is not on achieving compliance with industry standards (e.g., ISO 26262 and ISO 21434) like ours.

VII. Conclusion

In this paper, we address the importance of integrating safety and security in cyber-physical systems, especially in critical domains such as the automotive sector, which demands high levels of safety and protection against cyberattacks. Through a detailed analysis of the Open Dependability Exchange (ODE) metamodel, we identified significant limitations in its ability to perform safety and security co-analysis. To overcome these limitations, we proposed an extension to the ODE metamodel, which introduces new elements and relationships to model attack and failure scenarios more effectively, in line with the ISO-21434 and ISO-26262 standards.

Our contribution was evaluated in an illustrative example of an autonomous vehicle, where we presented the coverage increase of ISO 21434 concepts in our extended metamodel for safety and security co-analysis. By modeling attack-fault trees, we showed how cyberattacks and hardware faults could interact, culminating in risky situations such as unintentional braking, and highlighted the importance of mitigation measures to prevent such scenarios.

We believe that the proposed extensions improve ODE's co-analysis capabilities, making it a more robust tool for engineering dependable systems. For future work, we propose the creation of a concrete syntax for the metamodel and the development of a model-oriented methodology that facilitates integrated safety and fault analysis. We also intend to align our safety and security co-analysis metamodel with the new ISO/TS 5083 automated-driven system safety standard. Additional studies could explore the applicability of our approach in other domains, such as aviation or industrial systems, to validate the feasibility of our proposal in other domains.

Acknowledgment

We thank CAPES Brazilian research funding agency for the financial support.

References

[1] Y. Dajsuren and M. van den Brand, *Automotive Systems and Software Engineering*. Cham: Springer International Publishing, 2019.

[2] K. Aslansefat *et al.*, "Safedrones: Real-time reliability evaluation of uavs using executable digital dependable identities," in *Int. Symposium on Model-Based Safety and Assessment*, (Cham), pp. 252–266, Springer International Publishing, 2022.

[3] M. Zeller *et al.*, "Open dependability exchange metamodel: A format to exchange safety information," in *2023 Annual Reliability and Maintainability Symposium (RAMS)*, (Piscataway, NJ), pp. 1–7, IEEE, 2023.

[4] M. Trapp, D. Schneider, and P. Liggesmeyer, "A safety roadmap to cyber-physical systems," in *Perspectives on the future of software engineering*, pp. 81–94, Cham: Springer, 2013.

[5] J. Martinez, J. Godot, A. Ruiz, A. Balbis, and R. Ruiz Nolasco, "Safety and security interference analysis in the design stage," in *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops* (A. Casimiro, F. Ortmeier, E. Schoitsch, F. Bitsch, and P. Ferreira, eds.), (Cham), pp. 54–68, Springer International Publishing, 2020.

[6] International Organization for Standardization, "ISO 26262 "Road Vehicles – Functional Safety"," 2011.

[7] ISO, "ISO/SAE 21434: Road vehicles — Cybersecurity engineering," 2021.

[8] B. Kruck, P. Munk, and D. Angermeier, "Safe and secure: Mutually supporting safety and security analyses with model-based suggestions," in *2021 IEEE Int. Symposium on Software Reliability Engineering Workshops (ISSREW)*, (Piscataway, NJ), pp. 172–181, IEEE, 2021.

[9] R. Kumar and M. Stoelinga, "Quantitative security and safety analysis with attack-fault trees," *Proc. of IEEE Int Symposium on High Assurance Systems Engineering*, vol. 1, no. September 2018, pp. 25–32, 2017.

[10] R. Wei *et al.*, "DEIS: dependability engineering innovation for cyber-physical systems," in *Software Technologies: Applications and Foundations - STAF 2017 Collocated Workshops, Marburg, Germany, July 17-21* (M. Seidl and S. Zschaler, eds.), vol. 10748 of *Lecture Notes in Computer Science*, pp. 409–416, Springer, 2017.

[11] "Dependability Engineering Innovation for CPS - DEIS." https://cordis.europa.eu/project/id/732242, 2016. Accessed: 2024-10-20.

[12] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[13] A. Lautenbach and M. Islam, "Heavens–healing vulnerabilities to enhance software security and safety," *The HEAVENS Consortium (Borås SE)*, 2016.

[14] Microsoft, "The STRIDE Threat Model," 2009.

[15] I. Nai Fovino, M. Masera, and A. De Cian, "Integrating cyber attacks within fault trees," *Reliability Engineering and System Safety*, vol. 94, no. 9, pp. 1394–1402, 2009.

[16] B. Kaiser *et al.*, "Advances in component fault trees," in *Safety and Reliability–Safe Societies in a Changing World*, pp. 815–823, Boca Raton, FL, USA: CRC Press, 2018.

[17] G. Sabaliauskaite and A. P. Mathur, "Aligning Cyber-Physical System Safety and Security," in *Complex Systems Design & Management Asia*, (Cham), pp. 41–53, Springer International Publishing, 2015.

[18] É. André *et al.*, "Parametric analyses of attack-fault trees," in *2019 19th Int. Conf. on Application of Concurrency to System Design (ACSD)*, (Piscataway, NJ), pp. 33–42, IEEE, 2019.

[19] C. E. Budde, C. Kolb, and M. Stoelinga, "Attack trees vs. fault trees: two sides of the same coin from different currencies," in *International Conference on Quantitative Evaluation of Systems*, pp. 457–467, Springer, 2021.

[20] A. Lautenbach, M. Almgren, and T. Olovsson, "Proposing heavens 2.0–an automotive risk assessment model," in *Proc. of the 5th ACM Computer Science in Cars Symposium*, (New York, NY, USA), pp. 1–12, ACM, 2021.

[21] N. P. de Souza *et al.*, "Extending STPA with STRIDE to identify cybersecurity loss scenarios," *Journal of Information Security and Applications*, vol. 55, no. October, 2020.

[22] T. Hayakawa *et al.*, "Proposal and application of security/safety evaluation method for medical device system that includes iot," in *Proc. of the 2018 VII Int. Conf. on Network, Communication and Computing*, (New York, NY, USA), pp. 157–164, ACM, 2018.

[23] N. Papakonstantinou *et al.*, "Early hybrid safety and security risk assessment based on interdisciplinary dependency models," in *2019 Annual Reliability and Maintainability Symposium (RAMS)*, (Piscataway, NJ), pp. 1–7, IEEE, 2019.

[24] B. Gallina, L. Montecchi, A. L. De Oliveira, and L. Bressan, "Multiconcern, dependability-centered assurance via a qualitative and quantitative coanalysis," *IEEE Software*, vol. 39, no. 4, pp. 39–47, 2022.

[25] G. Sabaliauskaite, J. Cui, L. S. Liew, and F. Zhou, "Integrated safety and cybersecurity risk analysis of cooperative intelligent transport systems," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, pp. 723–728, IEEE, 2018.

[26] É. André, D. Lime, M. Ramparison, and M. Stoelinga, "Parametric analyses of attack-fault trees," *Fundamenta Informaticae*, vol. 182, no. 1, pp. 69–94, 2021.

[27] V. L. Grechi, A. L. de Oliveira, and R. T. V. Braga, "Model-driven safety and security co-analysis: A systematic literature review," *Journal of Systems and Software*, p. 112251, 2024.

[28] G. Bakirtzis *et al.*, "An ontological metamodel for cyber-physical system safety, security, and resilience coengineering," *Software and Systems Modeling*, vol. 21, no. 1, pp. 113–137, 2022.

[29] B. Gallina and H. Zulqarnain, "Making safeconcert security-informed to enable multi-concern modelling," in *Proc. 30th Eur. Safety Reliability Conf.(ESREL)*, pp. 2049–2056, 2020.

[30] A. Debiasi, F. Ihirwe, P. Pierini, S. Mazzini, and S. Tonetta, "Model-based analysis support for dependable complex systems in chess," in *9th International Conference on Model-Driven Engineering and Software Development*, 2021.

[31] B. Gallina, M. A. Javed, F. U. Muram, and S. Punnekkat, "A model-driven dependability analysis method for component-based architectures," in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 233–240, IEEE, 2012.

[32] L. Montecchi, P. Lollini, and A. Bondavalli, "A reusable modular toolchain for automated dependability evaluation," in *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, pp. 298–303, 2013.

[33] M. Steiner and P. Liggesmeyer, "Combination of safety and security analysis-finding security problems that threaten the safety of a system," in *Computer Safety, Reliability, and Security. SAFECOMP 2013 Workshops*, 2013.

[34] A. Kondeva, V. Nigam, H. Ruess, and C. Carlan, "On computer-aided techniques for supporting safety and security co-engineering," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 346–353, IEEE, 2019.

[35] H. Abdo, M. Kaouk, J.-M. Flaus, and F. Masse, "A safety/security risk analysis approach of industrial control systems: A cyber bowtie–combining new version of attack tree with bowtie analysis," *Computers & security*, vol. 72, pp. 175–195, 2018.

[36] M. Jablonski, D. Wijesekera, and A. Singhal, "Generating cyber-physical system risk overlays for attack and fault trees using systems theory," in *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, pp. 13–20, 2022.