Multi-layered model-based design approach towards system safety and security co-engineering

Megha Quamara, Gabriel Pedroza, Brahim Hamid

Multi-layered Model-based Design Approach towards System Safety and Security Co-engineering

Megha Quamara*, Gabriel Pedroza*, Brahim Hamid†
*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
Email: {megha.quamara, gabriel.pedroza}@cea.fr
†IRIT - University of Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
Email: brahim.hamid@irit.fr

***Abstract*—The integration of safety and security concerns in critical domains (e.g., Cyber-Physical Systems (CPSs)) is of utmost importance, and should be conducted in early design phases of system engineering process. Within a Model-Based System Engineering (MBSE) context, safety and security requirements cascade-down across models and views, thus contributing to the detailed missions, functions, and lastly, the architecture. Such enrichment process is often complex and lacks guidance to consistently breakdown high-level mission-centric system specifications into the detailed architecture. In particular, non-savvy safety and security engineers require support to facilitate integration and verification of stringent safety constraints and security exigencies. In this regard, we propose a multi-layered design approach that leverages existing techniques like Model-Driven Engineering (MDE) and formal methods, to facilitate integrated verification of high-level safety and security objectives that can be further specialized across different representations (i.e. mission, functional, and architectural) of the system. The overall approach is validated based upon a Connected Driving Vehicles (CDVs) case study, and using Eclipse Papyrus and Rodin as experimentation tools.**

***Index Terms*—safety, security, co-engineering, design, model-driven engineering, formal methods, connected driving vehicles**

I. Introduction

Current system development paradigms show a shift from traditional Industrial Control Systems (ICSs) to software-intensive systems (e.g., Cyber-Physical Systems, CPSs) what substantially increases their degree of inter-connectivity, the stringency of requirements, and, in consequence, the design complexity. Indeed, the impact of deploying systems with design flaws, which are not merely inadequate in terms of their operational capabilities, but also unsafe and vulnerable to security attacks, can be critical in regards to economical, business, and safety criteria what, in the end, can potentially jeopardize human lives [1]. In such cases, an effective identification and treatment of safety and security risks is crucial, which according to the *"correct-by-design"* principle, should be conducted at early design stages of the system development process [2].

In a typical Model-Based Systems Engineering (MBSE) approach, requirements are broken down across models and views, from high-level teleological representations of the target system up to detailed architecture models [3]. This enrichment process is often complex and lacks guidance for consistent transfer of knowledge pertaining to both safety and security disciplines, across different representations of the system under design (*issue* **P1**). The state-of-the-art reveals that the assurance of both dedicated safety and security properties[1] is imperative for improving system confidence to perform critical tasks [4]. However, conducting design-level properties' verification can be error-prone, mainly due to ambiguous properties' specifications or biases introduced by non-savvy engineer's interpretation (*issue* **P2**). Notwithstanding the need for incorporation, existing approaches for system engineering show that, in many cases, safety and security analyses are often conducted independently [5], [6]. As observed in several safety-critical domains (e.g., automotive), an entanglement exists between safety constraints (e.g., messages' latency) and security exigencies (e.g., encryption mechanisms' overhead), and their mutual assurance needs to be verified [7] (*issue* **P3**). However, to harmonize safety and security properties' specifications, a joint analysis is technically challenging in practice due to complexity in terms of model size, frameworks/languages involved, proofs' intricacy, etc. A lack of automated tool support for integrated validation and verification of properties is thus observed (*issue* **P4**).

A. Intended Contributions

To address the above-discussed issues (P1-P4), the contributions of this paper are multi-fold. First, we propose a multi-layered model approach to facilitate the incorporation of safety and security properties' specifications at mission, functional, and architectural layers (P1). To this end, we leverage existing modeling techniques [3], such as Unified Modeling Language (UML) [8] targeting both systems' and properties' modeling, and their further mapping into the formalism of solvers and model checkers for verification. The use of formal techniques, namely Event-B [9], even at early modeling phases, should ensure that, once deployed, the system's operation is reliable and in conformity with the rules originated by mathematical logic (P2). In addition to typical system's structural and behavioral concerns' analysis [10], we propose to impose a defined set of safety and security objectives at each layer (*i.e.* mission, functional, and architectural) for identifying whether the system design meets them, and avoiding potential conflicts (P3).

[1]Fundamental well-defined notions that are the building blocks upon which high-level requirements can be decomposed and characterized.

The proposed approach being technology-agnostic provides the flexibility with regards to the choice of specification/modeling languages and Verification and Validation (V&V) tool support. For demonstration purposes, the tool support developed is based on Papyrus [11], as modeling framework, and Rodin [12], as verification tool. This tool-chain is applied to evaluate the overall approach by analyzing a use case of Connected Driving Vehicles (CDVs), being both safety- and security-critical CPSs (P4). Overall, the main contribution lies in the multi-layered design approach to conduct an integrated safety-security properties' verification. Since this is a first step towards an effective co-engineering, this paper mainly covers the mission-centric layer. To sum up, our approach also aims to facilitate model re-usability to non-savvy engineers via a set of libraries containing safety and security signatures that are generic, but specialize-able and amenable for instantiation across different layers.

In summary, the intended contributions of this work are -

- **(C1)** The introduction of a multi-layered modeling framework to capture a system under design at different levels of granularity via a set of Domain-Specific Modeling Languages (DSMLs) corresponding to each layer.
- **(C2)** The instantiation of the approach through an integrated design framework, targeting high-level mission-centric system specification (*i.e., Mission layer*).
- **(C3)** The formal-based and rigorous specification of safety and security objectives of the modeled-system.
- **(C4)** A tool-chain support integrating Model-Driven Engineering (MDE) techniques, namely Eclipse Papyrus, and a formal-based tool, namely Rodin, to conduct verification, spot inconsistencies, and ensure design conformity with respect to the objectives.
- **(C5)** The application of the approach in the context of CDVs.

B. Outline

The remainder of this paper is organized as follows. Section II details the key aspects of the proposed approach. Section III presents an instantiation of the approach for the mission layer of system specification. Section IV exemplifies the approach via a CPS case study. Section V reviews related works and positions our contribution. Finally, Section VI concludes this paper, with perspective work directions.

II. Overall Approach

In this section, we discuss the key aspects of the proposed multi-layered design approach, including the modeling and formalization languages and techniques for its realization.

A. Need for Integrated Safety-Security Multi-layered Design

Assurance of high-level safety missions (e.g., collision avoidance) of the system (e.g., autonomous vehicles) relies on its underlying functions (e.g., obstacle detection) and architectural components (e.g., LIDAR). In such cases, safety properties' verification can be conducted with respect to the system, sub-systems, or architecture, depending on the availability of essential design details. Likewise, risk analysis to address high-level security concerns (e.g. unauthorized access) is often information-centric what entails cascading down to a detailed system view, comprising components and transmission channels. Such instances, in turn, call for a consistent transfer of system-related aspects and preservation of properties across different representations of the system with varying granularity. However, this enrichment process being complex, often lacks guidance in terms of language and automated tool support for non-savvy engineers, to integrate and conduct verification of safety and security properties' specifications across the system design. This endorses the development of the multi-layered design approach proposed in this work.

B. Proposed Multi-layered System Modeling Framework

The proposed multi-layered system modeling framework **(C1)** depicted in Fig. 1, decomposes system representation into following three layers[2] - 1) **Mission layer** (*Layer 1*) defines the set of missions to be achieved by the system under design that are envisioned to be accomplished via application of operations. In other terms, it concentrates on the formulation of high-level strategic concerns of the system that are derived from the operational, safety, and security requirements. 2) **Functional layer** (*Layer 2*) defines the system decomposed into functional paths that represent the flow of information between a set of functions performed by different sub-systems, together with their functional interfaces. 3) **Architectural layer** (*Layer 3*) defines the system as a composition of a set of components that represent self-contained computational/-communication elements or physical entities. In other terms, it concentrates on low-level technical details of the system.

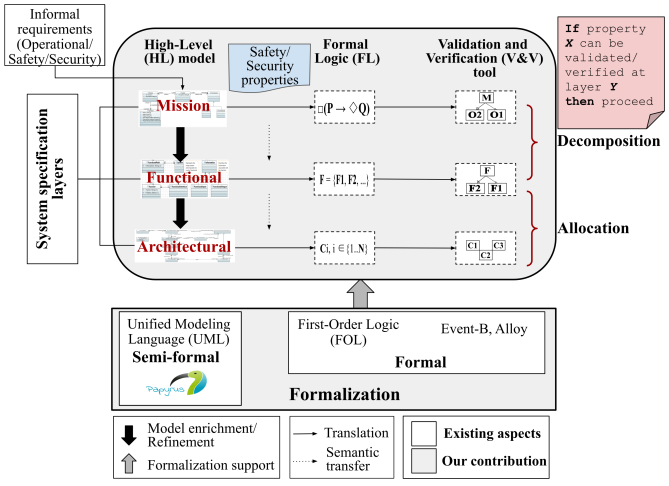

Fig. 1. Multi-layered system specification framework.

Before going further, it is worth to situate the choice behind the specific layered representations in the context of this work. *Layer 1* offers a teleological view to understand the

[2]Inclusion of the relevant representations of the system to capture its key aspects is the choice of the system designer.

overall purpose of complex engineered systems, where an early safety-security analysis can still be conducted despite the absence of detailed architectural choices. Likewise, *Layer 2* represents a classic functional decomposition of the system based on consensus in the literature [13]. Finally, *Layer 3* specification provides the technical details to decide how the system functions distribute over the components lying within the system. The conceptual models corresponding to the aforementioned layers comprise fundamental notions, along with their attributes and potential relationships, for specifying the structural and behavioral aspects of the system at different levels (*Mission* at highest) of granularity. The introduced layers stand for keeping separation between modeling purposes in the aim to facilitate incorporation and treatment of safety and security properties, rather than keeping them in a single encapsulated description [13]. Each layered representation is independently defined with its own set of semantics to facilitate integrated specification and analysis of the properties. To address their intertwined semantics, the framework is amenable for both top-down or bottom-up approaches to be applied. Specifically, the former would allow the incorporation of greater details during the system specification process, as described below -

- *Mission* captures WHAT is needed to be achieved by the system.
- *Function* captures HOW to achieve WHAT is needed to be achieved by the system.
- *Architecture* captures WHICH elements can finally realize the WHAT and HOW.

On the other hand, the bottom-up approach shall allow to analyze the propagation of malfunctioning at lowest layer upwards up to the highest one. For instance, from safety perspective, a faulty component (e.g., sensor) may eventually trigger a hazardous situation for the whole system (e.g., autonomous vehicle) through the transmission of erroneous information. The structural and semantic linking among these layers concentrate on ensuring the consistency between corresponding representations and preservation of properties via traceability.

C. Integrated Design Approach for Safety and Security

The aforementioned conceptual models lay the foundation for the proposed integrated design approach, which combines - 1) *modeling* of detailed system aspects, and safety and security properties, 2) *formalization* of both system and properties, 3) *integration* of formalized model elements, and 4) *verification* of properties via transformation into a delegated formal tool. The proposed integration allows to encompass the *"safe- and secure-by-design"* principle, the disambiguation of properties' specification, and the early detection of potential conflicts between properties. In addition, the formalized properties' specifications will be generic enough to be accommodated as reusable libraries to be instantiated at the three model layers.

D. Languages and Techniques for Modeling and Formalization

The specification language for the proposed design approach is a multi-paradigm language, comprising -

- *Semi-formal* constructs for graphical representation and modeling aspects supported by the approach. Accordingly, we use UML to create system specification meta-models and profiles (*i.e.* the DSMLs) [8]. Being a general-purpose modeling language, UML can be extended according to the specificities of the target application domain, whereby system architects and developers can understand the diverse aspects associated with the system and collaborate during the engineering process.
- *Formal* constructs for specification, reasoning, and verification of properties. Accordingly, we use First-Order Logic (FOL) to specify the aspects related to the system and safety and security properties in a generic (reduces state explosion during model checking), technology-independent (prevents technology-specific assumptions during use case analysis), and unambiguous fashion [14]. Other formal logic-based alternatives include Z notion [15] and pi-calculus [16] - which are beyond the scope of this work. Besides, the application of V&V analysis on the design model helps to identify discrepancies between verified properties (refer to Section III-F).

III. Mission Layer Integrated Design Approach

In this section, we present an instantiation of the integrated design approach proposed in Section II-C, for the *Mission* layer (i.e., *Layer 1*) of multi-layered system specification framework, as depicted in Fig. 2 **(C2)**.

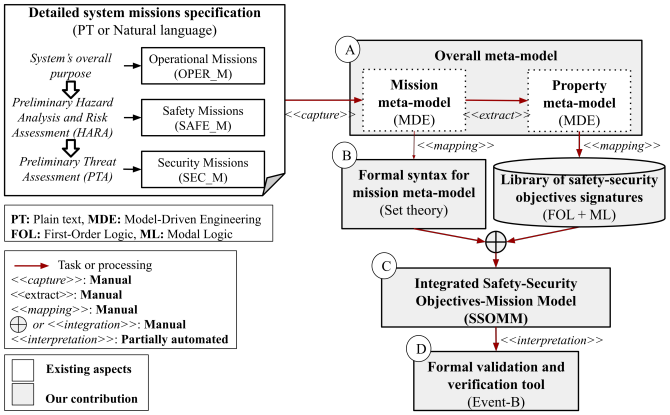

Fig. 2. Mission-layer (*Layer 1*) integrated design approach.

A. Modeling System Missions and Safety-Security Objectives

As mentioned in Section II-B, the *Mission* system specification model concentrates on the high-level strategic concerns of the system (i.e., missions), without any low-level technical details (i.e., functions and architecture). This conceptual model captures two different views - *mission specification* and *property specification*, where the latter extends the former. Some

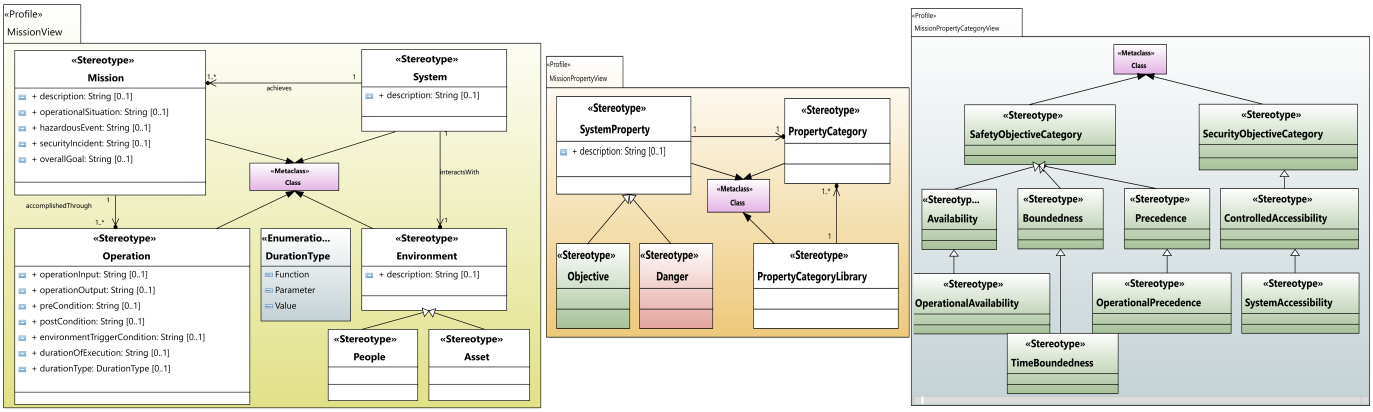

Fig. 3. DSML profiles defined for modeling Missions (MissionView) and Properties (MissionPropertyView, MissionPropertyCategoryView).

of the representative elements constituting this model are semantically inherited from the state-of-the-art artifacts proposed for rigorously defining the missions [17]–[21] and properties [22]. The meta-models corresponding to the aforementioned views are discussed in the following sub-sections, where both are implemented as UML profiles to provide a standardized modeling environment.

1) Mission DSML: Fig. 3 depicts the UML profile capturing the *"mission specification view"*. It aims to represent the system in terms of its missions, operations, environment, and their interactions. It also provides the basis to incorporate new concepts that are necessary to capture safety and security properties. The principal classes constituting the elements of the corresponding meta-model as UML notations, are defined as follows -

- **System:** Represents the system under design as an atomic unit, sharing a physical and logical border with its environment.
- **Mission:** Represents a high-level purpose of the system in the form of an atomic finality related to some behavior, constraint, or feature. The mission statement reflects an obligation comprising modal verbs (e.g., *must*, *will*, *should*). *overallGoal* captures the intended *operationalSituation* of the system, derived from the *hazardousEvent* or *securityIncident*.
- **Operation:** Represents the strategy or implementation steps performed by the system to accomplish the mission. Application of the operation leads to an elementary state transition with respect to the system that may or may not be aligned with the mission statement. *operationInput* and *operationOutput* represent the operation-specific stimuli and response, respectively, crossing the physical and logical border between the system and its environment. *pre/postCondition* and *environmentTriggerCondition* capture the descriptive and prescriptive conditions, respectively, associated with the application of the operation. *durationOfExecution* captures the interval typed as **DurationType** (*Function*, *Parameter*, or *Value*), for the operation to get triggered and produce an output.
- **Environment:** Represents the physical environment with which the system interacts. It may comprise - 1) **People** (e.g., system operators, users, or observers) having cooperative or even malicious intent, and 2) **Asset** (commercial, personal, or civic) that exist outside the system.

2) Safety and Security Properties DSML: Fig. 3 depicts the profile capturing the *"property specification view"*. It represents the reusable model libraries typed as **PropertyCategoryLibrary**, defining high-level properties associated with the system (**SystemProperty**). **PropertyCategory** represents the classification of safety and security objectives associated with the system in a given context. For example, in a safety context, *operational availability* is defined by the International Standard Organization (ISO) 26262 as the "capability of a product (e.g., system, hardware, software, etc.) to provide a stated function if demanded, under given conditions over its defined lifetime" [20]. Readiness of the system in terms of completion of the operations assigned to it, is of significant importance to guarantee distinct missions in safety-critical systems, e.g., obstacle detection in autonomous vehicles to avoid crash. Likewise, in security context, *system accessibility* is defined as "a qualitative feature that represents the ability to limit the access of the system and retrieval of information by the entities" [23]. These libraries are subsequently used as external models for capturing the safety and security objectives of the system as signatures, as described in the following sub-sections.

B. Formalization of the Modeling Languages

In this section, we describe the formal syntax associated to the DSMLs corresponding to the system missions and safety and security objectives mentioned in the previous Section III-A.

1) Formal Syntax for Mission DSML: The DSML illustrated in Fig. 3, is attached with a syntax that preserves the associations and types in the UML profile. Once the system missions are modelled in this language, their corresponding formal syntax can be obtained by following the mapping described in Table I.

2) Formal Syntax and Logic for Properties DSML: For the sake of simplicity, we only explain one safety objective

TABLE I
MAPPING: MISSION VIEW PROFILE ↦ FORMAL SYNTAX

DSML element	Formal syntax
System	$S := (\{M_i\}, Environment), i \in \{1 \dots n\}$
Mission	$M_i := (\{O_j\}, operationalSituation, hazardousEvent, securityIncident, overallGoal), j \in \{1 \dots k\}$
Operation	$O_j := (operationInput, operationOutput, preCondition, postCondition, environmentTriggerCondition, durationOfExecution, durationType), durationType \in \{FUNCTION, PARAMETER, VALUE\}$
Environment	$Environment \in \{People, Asset\}$

category (i.e., *Availability*) and one security objective category (i.e., *Controlled Accessibility*) that represent the pragmatical choices inspired by literature concerning safety- and security-critical systems [20], [23]. Notice that the work developed so far includes a library of other representative safety and security objectives' signatures (see Fig. 3), which is nonetheless not included in this paper, due to lack of space. The FOL-based formalism of the safety and security objectives belonging to the defined categories is extended with Modal Logic (ModL) [24] to capture their temporal aspects. The formal syntax and logic defining each safety and security objective signature is introduced in Table II, and discussed below **(C3)** -

- **Operational availability:** The operational availability objective is defined as follows - "a system S in a certain operational situation should allow for the realization of safety-critical operation(s) O_j, whenever a hazardous event is detected". The execution of these operations should be in alignment with the accomplishment of the safety mission.
- **System accessibility:** The system accessibility objective is defined as follows - "a system S must allow and limit the access of security-critical operation(s) O_j to only authorized entities". In other words, entities belonging to the environment, for instance people, can get access to perform certain operation(s) O_j over the system S if and only if they have the required privileges.

TABLE II
MAPPING: PROPERTIES PROFILE ↦ FOL AND MODL

DSML element	FOL and ModL
System	$S := \{M_i\}, i \in \{1 \dots n\}$
Mission	M_i
Operation	$O_j, j \in \{1 \dots k\}$
OperationalAvailability	$(operationalSituation[M_i] \wedge hazardousSituation[M_i]) \Rightarrow accomplishedThrough[M_i, O_j]$
ControlledAccessibility	$privilege[People, S, O_j] \Rightarrow access[People, O_j]$

C. Integrated Properties-Mission Syntax

In this section, we integrate the formal syntaxes presented in Tables I and II to have a single formal specification. Accordingly, a formal semantics is defined and associated with the system missions and operations. The resulting integrated model is called Safety-Security Objective-Mission Model (SSOMM), and its construction is described in the following paragraphs.

In this model, we assume the local operationalization of the system missions based on their description. In simpler terms, system operations are derived from the mission descriptions. Inference rules can be applied for identifying the correct set of pre/post-conditions and environment trigger conditions corresponding to the operations for mission accomplishment. An operation can be performed if and only if the *environmentTriggerCondition* holds in the event that the *preCondition* also holds. This ensures the consistency with respect to the application of the operations, and traceability between the operations and their underlying missions. In this process, we rely upon *generative* semantics that disallow all the behavioral changes, except for the ones that are explicitly required by the mission description [25]. In this case, the operations are observed as restrictive executions on the state transitions of the system. Here, only those system attributes variables that are declared in the *operationOutput* clause will be affected by the application of the operation.

The informal description associated with the mission is decomposed into *operationalSituation*, *hazardousEvent/securityIncident*, and *overallGoal* for the formal representation. Missions and operations can be formally specified using a range of modalities (ModL), including $\circ$ (next), $\Diamond$ (eventually), $\Diamond_{\leq} d$ (bounded eventually, where d denotes the gap between two successive occurrences), and $\Box$ (always) for capturing the notion of future. These are defined on top of standard FOL operators, including $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence). Particularly, we specify missions and operations in the form of the following predicate -

$$PreCondition(P) \Rightarrow PostCondition(Q)$$

Predicates of the form $P \Rightarrow Q$ interpret as $\Box(P \rightarrow \Diamond Q)$. Similarly, predicates of the form $P \Leftrightarrow Q$ mean $\Box(P \leftrightarrow Q)$. Here, $\Rightarrow$ means *strongly implies* and $\Leftrightarrow$ means *strongly equivalent*. Depending on the priority of the mission or operation with respect to safety and security aspects, $\circ$ (highest priority), $\Diamond_{\leq} d$, or $\Diamond$ (lowest prority) can be used in the formal specification. A mission allocation pattern can be defined for the allocation of the mission $M_i \in M$, where $M = \{M_1, M_2, ..., M_i\}$ is a set of independent missions, to one or more relevant operations $O_j \in O$, where $O = \{O_1, O_2, ..., O_j\}$ is a set of independent or dependent operations, for the accomplishment of M_i, as follows -

$$\begin{aligned}((&operationalSituation[M_i] \wedge hazardousSituation[M_i]) \\ \Leftrightarrow &(preCondition[O_j] \wedge requiredTriggerCondition[O_j])) \\ &\Rightarrow accomplishedThrough[M_i, O_j] \quad (1)\end{aligned}$$

Upon execution of the operation O_j to which the mission M_i is allocated, the *overallGoal* of M_i becomes equivalent to the *postCondition* of O_j. Formally speaking,

$$\begin{aligned}accomplishedThrough[M_i, O_j] \Rightarrow (&overallGoal[M_i] \Leftrightarrow \\ &postCondition[O_j]) \quad (2)\end{aligned}$$

D. Properties Conflict Identification

Conflicts between objectives can be identified by verifying whether after mission accomplishment, there are states in

which all post-conditions are not satisfied simultaneously -

$$accomplishedThrough[M_i, O_{j_1}, O_{j_2}] \Rightarrow (overallGoal[M_i] \Leftrightarrow (\neg postCondition[O_{j_1}] \vee \neg postCondition[O_{j_2}])) \quad (3)$$

The interpretation of previous formula is given by -

$$\Box(accomplishedThrough[M_i, O_{j_1}, O_{j_2}] \rightarrow \Diamond(overallGoal[M_i] \Leftrightarrow (\neg postCondition[O_{j_1}] \vee \neg postCondition[O_{j_2}]))) \quad (4)$$

No simultaneous fulfillment of post-conditions after mission accomplishment shall indicate, in particular, a conflict between the safety and security objectives of the system under design.

E. Model Interpretation and Formal Verification

In this section, we present the formalization of the *Mission* system specification DSML, along with safety and security objectives into Event-B. The interpretation of the *Mission* and *Property* profiles presented in Fig. 3 into Event-B is provided in Tables III and IV, respectively, along with the rationale/semantics regarding the choice made with respect to the elements and their relationships.

TABLE III
INTERPRETATION: UML MISSION DSML ↦ EVENT-B

UML Element (TYPE)	Event-B Element (TYPE)	Rationale
«Stereotype» System + description: String [0..1]	System (SET) *description* dropped*	*Required only for informal specification
«Stereotype» Mission + description: String [0..1] + operationalSituation: String [0..1] + hazardousEvent: String [0..1] + securityIncident: String [0..1] + overallGoal: String [0..1]	Mission (SET) *description* dropped* operationalSituation, hazardousEvent, securityIncident, overallGoal (VARIABLE)** operationalSituation ∈ Mission → BOOL (INVARIANT) hazardousEvent ∈ Mission → BOOL (INVARIANT) securityIncident ∈ Mission → BOOL (INVARIANT) overallGoal ∈ Mission → BOOL (INVARIANT)	*Required only for informal specification **Split the Mission description as per the form: *PreCondition* ⇒ *PostCondition*, for formal analysis
«Stereotype» Operation + operationInput: String [0..1] + operationOutput: String [0..1] + preCondition: String [0..1] + postCondition: String [0..1] + environmentTriggerCondition: String [0..1] + durationOfExecution: String [0..1] + durationType: DurationType [0..1]	Operation (SET) *operationInput*, *operationOutput* dropped* preCondition, postCondition, environmentTriggerCondition, counter** (VARIABLE) preCondition ∈ Operation → BOOL (INVARIANT) postCondition ∈ Operation → BOOL (INVARIANT) environmentTriggerCondition ∈ Operation → BOOL (INVARIANT) counter ∈ ℤ ∧ (0 ≤ counter) ∧ (counter ≤ n)*** (INVARIANT)	*Required only for informal specification **durationOfExecution is represented with counter of integral type ***n = number of events
1..* achieves 1	achieves (VARIABLE) achieves ∈ System → Mission (INVARIANT) achieves~ ∈ Mission ⇸ System (INVARIANT)	A System can achieve one or more Missions
1 interactsWith 1	interactsWith (VARIABLE) interactsWith ∈ System → Environment (INVARIANT)	A System interacts with an Environment as a whole
1 accomplishedThrough 1..*	accomplishedThrough (VARIABLE) accomplishedThrough ∈ Mission → Operation (INVARIANT) accomplishedThrough~ ∈ Operation ⇸ Mission (INVARIANT)	A Mission can be accomplished by one or more Operations
«Stereotype» Environment + description: String [0..1] «Stereotype» People «Stereotype» Asset	Environment (SET) People, Asset (CONSTANT) People ∈ ℙ (Environment) (AXIOM) Asset ∈ ℙ (Environment) (AXIOM) partition(Environment, People, Asset) (AXIOM)	People and Asset partition the Environment; i.e., (Environment = People ∪ Asset) ∧ (People ∩ Asset = ϕ)

Our model in Event-B contains three contexts, viz. *C0MissionView*, *C1MissionPropertyView*, and *C2MissionUtility*, for the formal declaration of mission and property elements, along with the utility constants for a generic representation of the elements, respectively. Furthermore, the initial abstract specification *M0MissionView* corresponding to the mission DSML, formally captures the desired behavioral aspects of the system as model invariants that are verified in the later refinements. We define four events (viz. INITIALISATION, EVENT1, EVENT2, and EVENT3) to capture the state transitions associated with

TABLE IV
INTERPRETATION: UML PROPERTY DSML ↦ EVENT-B

UML Element (TYPE)	Event-B Element (TYPE)
«Stereotype» SystemProperty + description: String [0..1] «Stereotype» Objective «Stereotype» Danger	SystemProperty (SET) *description* dropped[a] Objective, Danger (CONSTANT) Objective ∈ ℙ (SystemProperty) (AXIOM) Danger ∈ ℙ (SystemProperty) (AXIOM) partition(SystemProperty, Objective, Danger) (AXIOM)
«Stereotype» PropertyCategory 1..* 1 «Stereotype» PropertyCategoryLibrary	PropertyCategory, PropertyCategoryLibrary (SET) propertyCategoryLibrary (CONSTANT) propertyCategoryLibrary ∈ PropertyCategoryLibrary ⇸ PropertyCategory (AXIOM) propertyCategoryLibrary~ ∈ propertyCategory ⇸ PropertyCategoryLibrary (AXIOM)
«Stereotype» System 1..* «Stereotype» SystemProperty	systemProperty (CONSTANT) systemProperty ∈ SystemProperty ⇸ System (AXIOM)
«Stereotype» SystemProperty 1 1 «Stereotype» PropertyCategory	propertyCategory (CONSTANT) propertyCategory ∈ SystemProperty ↠ PropertyCategory (AXIOM)
«Stereotype» PropertyCategory «Stereotype» SafetyObjectiveCategory «Stereotype» SecurityObjectiveCategory	partition(SafetyObjectiveCategory, SecurityObjectiveCategory) (AXIOM)
«Stereotype» OperationalAvailability	OperationalAvailability (CONSTANT) OperationalAvailability ∈ SafetyObjective (AXIOM) ∀m·∃o·m ∈ Mission ∧ o ∈ Operation ∧ (operationalSituation[m] = {TRUE} ∧ hazardousEvent[m] = {TRUE}) ⇒ accomplishedThrough[m] = {o} (INVARIANT)
«Stereotype» SystemAccessibility	SystemAccessibility (CONSTANT) SystemAccessibility ∈ SecurityObjective (AXIOM) ∀p·∀s·∀o·p ∈ People ∧ s ∈ System ∧ o ∈ Operation ∧ p ↦ s ↦ o ∉ privilege ⇒ access[p ↦ o] = {FALSE} (INVARIANT)

[a]Required only for informal specification.

the system missions and operations. We also define the safety and security objectives presented in Section III-B2 in Event-B in the form of model invariants to verify that the defined events preserve the same during model verification. The guards corresponding to these events restrict the values of the variables as enabling conditions for the events. The correctness of the safety and security proof carried out on this machine is dependent on fifty-three Proof-Obligations (POs)[3] that are related to mission accomplishment through the execution of operation-centric events.

F. Tool-chain Support

The implementation of tool support for the *Mission* layer mainly targets system, safety, and security missions modeling and their translation into their formal counterparts, for further analysis and verification of safety and security objectives. This process involves several phases that must be backed up by system modeling (involving meta-modeling and profiling), model-to-model interpretation, and formal verification tool support. In this section, we introduce the two background platforms, namely Eclipse Papyrus [11] and Rodin [12], used in this work to make a proof-of-concept **(C4)** -

1) Eclipse Papyrus as Modeling Framework: For graphical representation, modeling, and profiling of the aspects involved in our approach, we rely upon Papyrus, a UML and SysML modeler developed as an open source Eclipse project [11]. Salient features offered by it include - full UML 2.0 support,

[3]Distribution of POs: Variable initialization (22), System specification (26), Safety and security invariants (5).

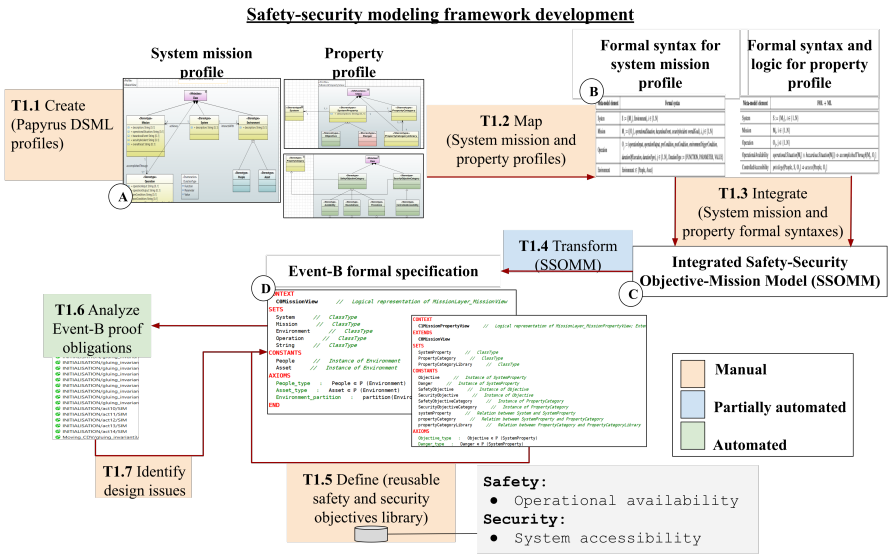

Fig. 4. Tool support architecture and related approach artifacts.

model validation, and code generation. The consistency between layers (DSMLs) is ensured by the extension mechanisms inherited from UML. Models correctness can be checked via extensions of Papyrus validation features.

2) Rodin for Formal Verification: Rodin is an Eclipse-based Integrated Development Environment (IDE) and an open tool supporting Event-B method, which facilitates mathematical proofs and refinement on discrete state transition-based system models [12]. In order to carry out safety and security analysis on the system model, we rely on theorem proving supported by Event-B via axioms and derivation rules. Indeed this process is iterative and compatible with the refinement process that allows model instantiation.

3) Implemented Tool-chain Architecture: Fig. 4 depicts the tool-chain support architecture implemented to support the safety-security co-engineering framework. Overall, it supports seven activities that are marked with the alphabets corresponding to the different phases of the proposed approach (see Fig. 2) - **(T1.1)** Creation of DSML meta-models and profiles corresponding to system missions and safety-security objectives in Papyrus. **(T1.2)** Mapping of DSML profiles to their corresponding formal syntaxes involving the use of FOL and ModL. **(T1.3)** Merging of formal syntaxes leading to an integrated SSOMM. **(T1.4)** Transformation of the integrated SSOMM to Event-B to formally capture the static and dynamic aspects of the system, along with the safety and security objectives' semantics. This constitutes indeed the formal models library corresponding to the *Mission* layer. **(T1.5)** Defining the reusable safety and security objectives' libraries at formal model level in the form of signatures. **(T1.6)** Generation of POs by Rodin corresponding to the safety and security objective invariants. **(T1.7)** Analysis of undischarged POs for identifying and fixing the underlying design issues.

IV. Case Study: Connected Driving Vehicles

In this section, we present the use case of autonomous Connecting Driving Vehicles (CDVs), as a domain-specific CPSs application towards technical demonstration and evaluation of the proposed approach **(C5)**. CDVs are both safety- and security-critical, as their malfunctioning behavior with respect to the design intent is hazardous towards the system and its environment [26]. Moreover, their high-networked architecture offers exposure of their functionalities to potentially hostile players. Notwithstanding the extensive ongoing engineering effort in the field, further work is still needed regarding methodological guidance for integrated safety and security verification to increase CDVs design trustworthiness.

A. Use Case Scenario

In the scope of this work, we assume the applicability of CDVs in limited perimeter areas; e.g., as shuttle buses in airport terminals, parks, and playgrounds [27]. Furthermore, we consider a realistic scenario of a converging road plan inspired by [28], as shown in Fig. 5. In this scenario, there are first two vehicles, denoted by CDV1 and CDV2, both in driving mode and approaching each other in different but converging roads with non-negligible speed. Another third vehicle CDV3 is in the same lane and direction as CDV1, but just behind it. In order to stay on the road lane, these vehicles follow virtual guides placed over the road. For the sake of simplicity, the aspects related to the road guides detection and the communication at the system level are disentangled. We also assume that CDVs in the scenario are in self-driving mode and with no possibility of any operational take over by passengers or the operator [29]. In the following sub-sections, we present the CDV mission model, and its formalization through instantiation in Event-B.

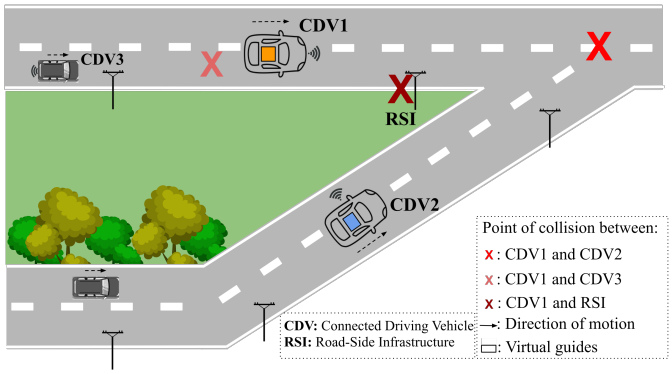

Fig. 5. Use case scenario: Converging road plan.

B. Mission Model of Connected Driving Vehicles

We consider a typical CDV as an atomic system intended to achieve a set of operational, safety, and security missions that are specified, as follows -

1) Operational Mission Specification: The specification of a high-level operational mission (OPER_M) of a CDV system, that follows the generic language syntax proposed in Section III-A1 and provided as input in natural language format to the overall approach, is as follows -

[OPER_M] A *CDV should allow transportation of the passengers from one place to another.*

2) Identification of Safety Hazards and Security Threats: We follow a preliminary approach based on ISO 26262 Hazards Analysis and Risks Assessment (HARA) [20] and threat analysis [6] to identify the safety hazards and security threats, respectively, with respect to OPER_M. For each identified hazardous scenario, we determine the Automotive Safety Integrity Levels (ASIL) by evaluating the three risk impact factors (i.e., Exposure, Controllability, and Severity). Likewise, we type each of the identified threats based on STRIDE threat model. An excerpt of our analysis results is presented in Table V. These are considered as input for the specification of safety and security missions at design time.

TABLE V
RESULT OF PRELIMINARY HAZARD AND THREAT ANALYSIS

Safety hazard	Description	Accident	ASIL
SH1: Unintended acceleration	Two CDVs approaching each other on converging roads with non-negligible speed	A CDV is side-collided leading to rollover	D (E4, C3, S3)

Security threat	Description	Attack	STRIDE category
ST1: Unauthorized operations	Hostile party performing unauthorized actions within on-board CDV architecture	Introduction of malware in CDV's computing facilities	Elevation of privilege

3) Safety and Security Missions Specification and Modeling: Relying on the syntax presented in Section III-A1, an excerpt of the safety and security missions specifications, in correspondence to SH1 and ST1 in Table V, is provided -

- **[SAFE_M1]** *A moving CDV should stop whenever an obstacle is detected.*
- **[SEC_M1]** *An operational CDV should only allow access to authorized actions being performed by its operator.*

We consider the above mission specifications and the semantics presented in Section III-C to instantiate the DSMLs presented in Section III-A1 according to the CDV scenario. For the sake of brevity, only an excerpt of this instantiation for SAFE_M1 is presented in Fig. 6.

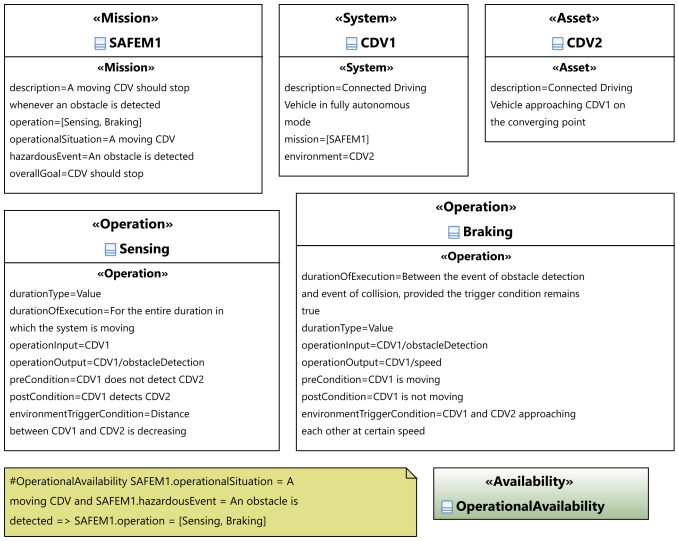

Fig. 6. UML profile instantiation of the safety mission SAFE_M1.

C. Formal Verification of Safety and Security Objectives

Since the Event-B interpretation of the *Mission* layer of system specification is generic (see Section III-E), it can be instantiated to incorporate the specific details of a concrete system according to the application domain. In the light of the considered use case, we define two Event-B contexts, *C3MissionViewInstance* and *C4MissionPropertyViewInstance*, to instantiate the mission and property specification views presented in Sections III-A1 and III-A2, respectively. In these contexts, we define constants, along with axioms to type these constants and define their relationships with the carrier sets. An excerpt of the context *C3MissionViewInstance* is depicted in Listing 1.

```
CONTEXT C3MissionViewInstance
EXTENDS C2MissionUtility
CONSTANTS CDV1, SAFE_M1, SEC_M1, CDV2, Sensing,
    Braking, IdentityChecking, AccessProvisioning
AXIOMS
    CDV1 ∈ System
    SAFE_M1 ∈ Mission ∧ SEC_M1 ∈ Mission ∧ Mission =
        {SAFE_M1, SEC_M1}
    CDV2 ∈ Asset
    Sensing ∈ Operation ∧ Braking ∈ Operation ∧
        IdentityChecking ∈ Operation ∧
        AccessProvisioning ∈ Operation ∧ Operation =
         {Sensing, Braking, IdentityChecking,
        AccessProvisioning}
```

Listing 1. Excerpt of C3MissionViewInstance context.

We also define a concrete machine *M1MissionLayerInstance* to refine the abstract machine *M0MissionView* (refer to Section III-E) in the form of an instantiation specific to this use case scenario. We specify gluing invariants for maintaining the

consistency between the variables belonging to the abstract and concrete system. Specifically, the events Moving_CDV, Obstacle_Detection, and Ensure_Braking of the concrete machine refine the events EVENT1, EVENT2, and EVENT3, respectively, of the abstract machine. Listing 2 shows an excerpt of the event Obstacle_Detection in Event-B.

```
Obstacle_Detection REFINES EVENT2
WHEN
    operSit = {SAFE_M1 ↦ TRUE}
    hazEvent = {SAFE_M1 ↦ TRUE}
    preSensing = {Sensing ↦ TRUE}
    envTrigSensing = {Sensing ↦ TRUE}
    achieves = {CDV1 ↦ SAFE_M1}
    interactsWith = {CDV1 ↦ CDV2}
    accomplishedThrough = {SAFE_M1 ↦ Braking}
THEN
    preSensing ≔ {Sensing ↦ FALSE}
    postSensing ≔ {Sensing ↦ TRUE}
    envTrigBraking ≔ {Braking ↦ TRUE}
    accomplishedThrough ≔ {SAFE_M1 ↦ Braking}
    preBraking ≔ {Braking ↦ TRUE}
```

Listing 2. Excerpt of Obstacle_Detection event.

POs corresponding to the guard strengthening and validation of the gluing invariants, prove the correctness of the instantiated model. The safety (*operational availability*) and security (*system accessibility*) objectives are represented as following Event-B invariants -

(System availability) $\forall m \cdot \exists o \cdot m \in \text{Mission} \land o \in \text{Operation} \land (\text{operationalSituation}[m] = \{\text{TRUE}\} \land \text{hazardousEvent}[m] = \{\text{TRUE}\}) \Rightarrow \text{accomplishedThrough}[m] = \{o\}$

(Controlled accessibility) $\forall p \cdot \forall s \cdot \forall o \cdot p \in \text{People} \land s \in \text{System} \land o \in \text{Operation} \land p \mapsto s \mapsto o \notin \text{privilege} \Rightarrow \text{access}[p \mapsto o] = \{\text{FALSE}\}$

V. Related Work and Positioning

In this section, we highlight salient features of some related works and position our approach, emphasizing system modeling frameworks and DSMLs, design approaches, formal-based techniques, and tool support.

A light-weight pattern-based technique based on KAOS framework is proposed in [25], for deriving software's operational specifications from system's high-level goals. Similar to our work, it relies on a multi-paradigm specification language for incremental reasoning on partial models (i.e., goal, object, agent, and operation). Nonetheless, it only covers the requirement engineering phase. A variety of works, like [5], [30], rely on DSMLs, implemented as UML/SysML profiles, for modeling software systems together with desired safety or security aspects and detecting inconsistencies, often in a separate manner. Other approaches, like [31], incorporate both safety and security properties, but adopt a fixed modeling view not covering other layers (nor design steps, like allocation) as in our approach. The authors in [19] presented a mission-based process, strengthening the links between analysis and architecture design stages, and adapted it to the wave development cycle, thus resulting less generic than our approach.

A functional-layer fine-grained co-design methodology for automotive CPSs is proposed in [32] to capture the specification of high-level functions through Design Space Exploration (DSE). A component-based design technique is presented in [33] to address safety requirements in CPSs applications, using modeling and analysis of real-time component-layer interactions. Likewise, following the security-by-design principle, the authors in [6] proposed a framework for capturing security objectives in autonomous systems. Our work is aligned with these X-by-design approaches and aims, additionally, to provide generic and specializable means to integrate safety and security aspects irrespective of the design flow (top-down or bottom-up).

In some recent efforts, formal logics, like FOL and temporal logic, are used for rigorous specification of safety and security properties [22], [34], [35]. Some of these works (e.g., [34]) are nonetheless less oriented to cover integrated specification of high-level safety and security properties. As drawback, works in this category are often constrained either by the underlying formal language (not able to represent required properties) or by lack of guidance to apply them at different layers (or even impossibility), thus preventing outcomes reusability.

A main concern demanding tool support is the passage from semi-formal structured models to their formal specification. Some of the cited works address the use of Event-B as formal method and tool for the analysis of system's safety and security concerns [35] [36]. These works can be constrained by the granularity level and concepts chosen for modeling what imposes requirements to be specified at the same level. In case another layer is needed for modeling, further guidance is still required to interpret models into a formal logics.

The distinctive features of the approach herein proposed lay in three main axes and strive to reduce previous gaps. First, the multi-layered approach is amenable to cover different phases of critical systems development irrespective of the design flow (top-down or bottom-up) being thus more generic. Second, along with ensuring consistent design flow, the multilayered approach also allows engineers to select a single layer adequate to the modeling granularity and analysis needs, thus resulting flexible enough. Since each layer comes with its respective formal interpretation, it can be used in both standalone or coupled mode with other layers. Third, the overall approach is agnostic of the underlying tool-chain technology only developed for proof-of-concept purposes. Notwithstanding this choice, the tool-chain integrates mechanisms that facilitate approach extension and specialization. All these distinctive features are encompassed with the capabilities to specify and model -predefined- safety and security properties, and to verify them. The support for early identification and systematic analysis of their inter-dependencies and conflicts is a plus.

VI. Conclusion and Perspectives

In this paper, we propounded a multi-layered design approach for integrated specification and verification of safety and security properties. The approach is instantiated for mission-centric systems via modeling of their missions and

safety and security objectives. The design is conducted at two levels - 1) semi-formal; relying upon technology-agnostic, generic, and standardized modeling languages, and 2) formal; based upon formal syntaxes, rules, and semantics. A model transformation was defined for interpretation of *Mission* models into Event-B specifications in order to conduct automated verification of safety and security objectives' signatures. The feasibility of the approach was proven by applying it to a CDVs use case borrowed from the transportation domain. The approach contributes to the reusability of safety and security objectives' signatures across different design projects. Being generic and mostly automated, it facilitates model transformation towards other formal frameworks, thus alleviating the complexity of manually integrating formal analysis of safety and security aspects into the design phase.

Since the approach introduced in this paper is a work-in-progress, our next actions will focus on its instantiation with respect to the remaining *Functional* and *Architectural* layers of system specification (refer to Fig. 1). Furthermore, we plan to strengthen vertical integration between the layered DSMLs (specifically, *Mission-Functional* and *Functional-Architectural*) via identification of potential relationships and establishment of links between the corresponding elements. For that purpose, we must seek for solutions to ensure properties' preservation across different layers.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE, 2008, pp. 363–369.

[2] M. Wolf and D. Serpanos, "Safety and security in cyber-physical systems and internet-of-things systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 9–20, 2017.

[3] J. A. Estefan *et al.*, "Survey of model-based systems engineering (MBSE) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[4] S. Zafar and R. G. Dromey, "Integrating safety and security requirements into design of an embedded system," in *12th Asia-Pacific Software Engineering Conference (APSEC'05)*. IEEE, 2005, pp. 8–pp.

[5] M. A. De Miguel, J. F. Briones, J. P. Silva, and A. Alonso, "Integration of safety analysis in model-driven software development," *IET software*, vol. 2, no. 3, pp. 260–280, 2008.

[6] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[7] G. Pedroza, "Towards Safety and Security Co-engineering: Challenging Aspects for a Consistent Intertwining," in *Security and Safety Interplay of Intelligent Software Systems*. Springer, 2018, pp. 3–16.

[8] L. Fuentes-Fernández and A. Vallecillo-Moreno, "An introduction to UML profiles," *UML and Model Engineering*, vol. 2, no. 6-13, p. 72, 2004.

[9] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys (CSUR)*, vol. 28, no. 4, pp. 626–643, 1996.

[10] J. Bau and J. C. Mitchell, "Security modeling and analysis," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 18–25, 2011.

[11] CEA, "Eclipse Papyrus Modeling Environment," 2021, https://www.eclipse.org/papyrus/.

[12] J.-R. Abrial *et al.*, "Rodin: an open toolset for modelling and reasoning in Event-B," *International journal on software tools for technology transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[13] D. Sanderson, J. C. Chaplin, and S. Ratchev, "A Function-Behaviour-Structure design methodology for adaptive production systems," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 9, pp. 3731–3742, 2019.

[14] C. Heitmeyer, J. Kirby, B. Labaw, M. Archer, and R. Bharadwaj, "Using abstraction and model checking to detect safety violations in requirements specifications," *IEEE Transactions on software engineering*, vol. 24, no. 11, pp. 927–948, 1998.

[15] J. P. Bowen, "Z: A formal specification notation," in *Software specification methods*. Springer, 2001, pp. 3–19.

[16] R. Milner, *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.

[17] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of computer programming*, vol. 20, no. 1-2, pp. 3–50, 1993.

[18] D. G. Firesmith, "Common concepts underlying safety security and survivability engineering," Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep., 2003.

[19] I. Cherfa, N. Belloir, S. Sadou, R. Fleurquin, and D. Bennouar, "Systems of systems: From mission definition to architecture description," *Systems Engineering*, vol. 22, no. 6, pp. 437–454, 2019.

[20] "ISO 26262-1:2018 Road vehicles — Functional safety," https://www.iso.org/standard/43464.html, 2018.

[21] "ISO/IEC 27000:2018 Information technology — Security techniques — Information security management systems," https://www.iso.org/standard/73906.html, 2018.

[22] Q. Rouland, B. Hamid, and J. Jaskolka, "Specification, detection, and treatment of STRIDE threats for software components: Modeling, formal methods, and tool support," *Journal of Systems Architecture*, vol. 117, p. 102073, 2021.

[23] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, pp. 106–124, 2009.

[24] R. Bull and K. Segerberg, "Basic modal logic," in *Handbook of philosophical logic*. Springer, 1984, pp. 1–88.

[25] E. Letier and A. Van Lamsweerde, "Deriving operational software specifications from system goals," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 6, pp. 119–128, 2002.

[26] D. Dominic, S. Chhawri, R. M. Eustice, D. Ma, and A. Weimerskirch, "Risk assessment for cooperative automated driving," in *Proceedings of the 2nd ACM workshop on cyber-physical systems security and privacy*, 2016, pp. 47–58.

[27] S. Zhiwei *et al.*, "Map free lane following based on low-cost laser scanner for near future autonomous service vehicle," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 706–711.

[28] D. Firesmith, "Engineering safety-and security-related requirements for software-intensive systems," Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep., 2007.

[29] B. Smith, "Summary of levels of driving automation for on-road vehicles," *Center for Internet and Society, Stanford Law School*, vol. 1, 2013.

[30] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML environment for the design and development of secure embedded systems," *APCOSEC, Asia-Pacific Council on Systems Engineering*, pp. 8–11, 2013.

[31] G. Pedroza, L. Apvrille, and D. Knorreck, "AVATAR: A SysML environment for the formal verification of safety and security properties," in *2011 11th Annual International Conference on New Technologies of Distributed Systems*. IEEE, 2011, pp. 1–10.

[32] J. Wan, A. Canedo, and M. A. Al Faruque, "Cyber–physical codesign at the functional level for multidomain automotive systems," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2949–2959, 2015.

[33] A. Masrur, M. Kit, V. Matěna, T. Bureš, and W. Hardt, "Component-based design of cyber-physical applications with safety-critical requirements," *Microprocessors and Microsystems*, vol. 42, pp. 70–86, 2016.

[34] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, p. 103289, 2019.

[35] I. Vistbakka and E. Troubitsyna, "Towards a formal approach to analysing security of safety-critical systems," in *2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 182–189.

[36] Z. Hong and X. Lili, "Application of software safety analysis using event-b," in *2013 IEEE Seventh International Conference on Software Security and Reliability Companion*. IEEE, 2013, pp. 137–144.