# Formal analysis approach for multi-layered system safety and security co-engineering

Megha Quamara, Gabriel Pedroza, Brahim Hamid

# Formal Analysis Approach for Multi-layered System Safety and Security Co-engineering

Megha Quamara[1,2][0000−0001−9380−6916], Gabriel Pedroza[1][0000−0002−7889−2892], and Brahim Hamid[2][0000−0002−2199−3916]

[1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
{megha.quamara, gabriel.pedroza}@cea.fr
[2] IRIT - University of Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
brahim.hamid@irit.fr

**Abstract.** Critical domains (like Cyber-Physical Systems-CPSs) have witnessed increased demand for considering both safety and security concerns during the early phases of the System Engineering (SE) process. Particularly, in the design phase, safety and security requirements should cascade down across different system views till the architectural design. However, such an enrichment process is often complex and lacks guidance to precisely specify the corresponding properties and consistently break down high-level system specifications into intricate architecture for a rigorous analysis. To this end, we propose a formal approach pursuing joint analysis of safety and security objectives, specialize-able across different system views. In particular, the approach strives for a multi-layered system representation, integrating mission, functional and component views, and libraries of pre-defined safety and security properties, instantiate-able at each layer. We rely upon the meta-modeling and formal techniques for the specification, conceptual modeling, formal interpretation, and verification of the system w.r.t. the allocated properties. The overall approach is validated using Rodin as an instance of a formal-based tool for properties' verification.

**Keywords:** Safety · Security · Co-engineering · Design · Analysis · Model-Driven Engineering · Formal Methods.

## 1 Introduction

Modern engineered systems, like Cyber-Physical Systems (CPSs), are becoming increasingly complex due to the integration of a variety of technology, highly networked and with heterogeneous usages and contexts. Despite their enormous potential, deploying such systems in critical applications entails the integration of safety and security concerns in light of their mutual influence. Nevertheless, a joint analysis towards harmonizing safety and security expertise is technically challenging. Particularly, in the design phase, the requirements are broken down from the high-level teleological representations to the detailed technical architecture of the System Under Design (SUD). However, this enrichment process is often complex and lacks guidance for consistent semantic transfer and integration of safety and security concerns/requirements. Besides,

conducting design-level properties'[3] verification to increase design trustworthiness, can be error-prone due to ambiguous properties' specifications or biases introduced by non-savvy engineer's interpretation. Moreover, existing System Engineering (SE) approaches exhibit standalone safety and security analyses in many cases [7,5]. As evident 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 [15]. In addition, we observe a lack of automated tool support for integrated analysis of the system, safety, and security properties; the disciplines of software architecture, security, and safety engineering are still to be better interfaced regarding their methods and frameworks.

To address the problematics above, we propose a joint design and analysis approach for three-layered system safety and security co-engineering. The work in this paper mainly focuses on (1) the vertical integration of notions for safety and security interplay across different modeling views of the system and (2) emphasizing the formal aspects for properties' representation, verification, and conflict identification. The approach relies on conceptual modeling using existing modeling languages (e.g., Unified Modeling Language (UML) [9]) to describe high-level safety and security objectives, i.e., the positive features or properties, specialize-able across different system views in the context of a three-layered system representation: mission, functional, and component. Besides the typical system's structural and behavioral concerns' analysis [3], we propose to impose a defined set of safety and security objectives' signatures at each layer to check for design conformity and avoid potential conflicts. As a prerequisite, we define interpretation rules for mapping the modeling concepts to their formal-based counterparts relying upon mathematical logic, namely First-Order Logic (FOL) and Modal Logic [4]. Moreover, we use Event-B [1], to obtain a more concrete specification of the system and property conceptual model and the accompanying formal-based tool, namely Rodin [16], to mechanize properties verification and spot inconsistencies at early modeling phases. The approach is illustrated via a Connected Driving Vehicles (CDVs) use case.

The rest of the paper is organized as follows. Section 2 describes the conceptual modeling of the system and safety and security properties. Section 3 presents the formalization of the model using FOL and Modal Logic. Section 4 describes the interpretation of the formalized model into Event-B for supporting properties analysis. Section 5 reviews related works. Finally, Section 6 concludes this paper, with perspective work directions.

## 2 Multi-layered System Conceptual Model

A conceptual model of a multi-layered system should capture the main concepts and relationships for describing the system in the context of different standards and domain-specific practices. We use UML Class diagrams to describe the conceptual model. Thus, concepts are represented by Classes, concept attributes by Class attributes, and relationships among concepts by links (e.g., association). The Package notation is used to make

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

groupings of the concepts. An excerpt of the graphical representation of the concepts and relationships is given in Fig. 1. In the rest of this section, we outline the different packages. Special attention is given to the concepts that show the essential features of this work. To facilitate readability and comprehension, the attributes of the different concepts and some of the links among the concepts are also described.

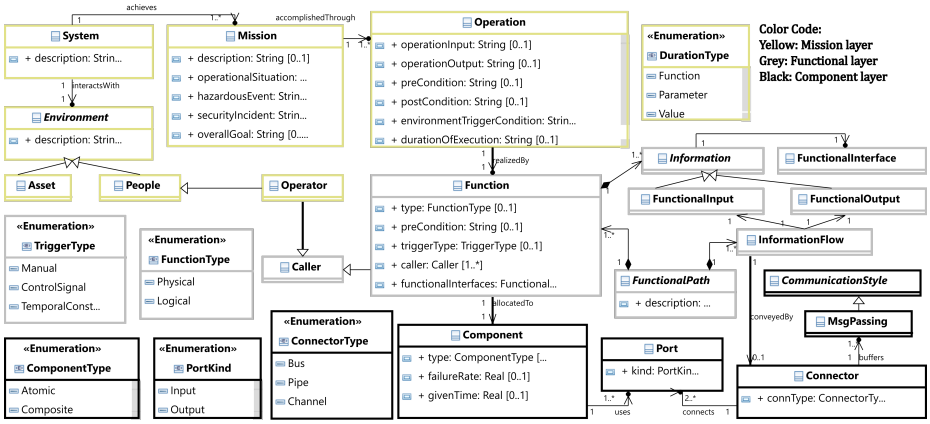


**Fig. 1.** Excerpt of the multi-layered system specification meta-model.

The proposed conceptual model is divided into four packages: (1) **Mission**, for concepts related to mission engineering; (2) **Functional**, for concepts related to functional engineering; (3) **Component**, for describing component-based engineering; and (4) **Property**, for safety and security aspects capturing and analysis. The *Mission* package contains concepts for offering system's teleological view, such as the *System* (e.g., CDV), *Mission* (system's high-level strategic concerns, e.g., collision avoidance [SAFE_M], ensure authorized actions [SEC_M]), *Operation* (for mission accomplishment, e.g., braking, access provisioning), *Environment* (comprising *People* or *Assets*), etc. The *Functional* package contains all the concepts correlated with the system's functionality, such as the *Function* (system's elementary tasks or services, e.g., perception- or actuation-related), *FunctionalPath* (sequence of functions to realize the operation via *InformationFlows*), etc. The *Component* package contains all the concepts for the system's detailed technical representation, such as the *Component* (self-contained computational elements/physical entities, e.g., sensor), *Port*s (for data exchange), *Connector*s (channels for establishing communication), etc.

Finally, the *Property* package (see Fig. 2) contains all the concepts for safety and security properties capturing and analysis, such as the *PropertyCategoryLibrary* (reusable model libraries for defining high-level properties of the system, function, component, etc.) and *PropertyCategory* (classification of safety and security objectives in a given context). The libraries are subsequently used as external models for capturing the objectives as signatures. We extend some properties (e.g., *Precedence* [8] and *Equivalence* [2]) from the literature by adapting them to the context of our approach. We call these

properties the basic ones since they play an elementary role in defining specific safety and security objectives (e.g., *Functional Integrity*) associated with the target SUD.

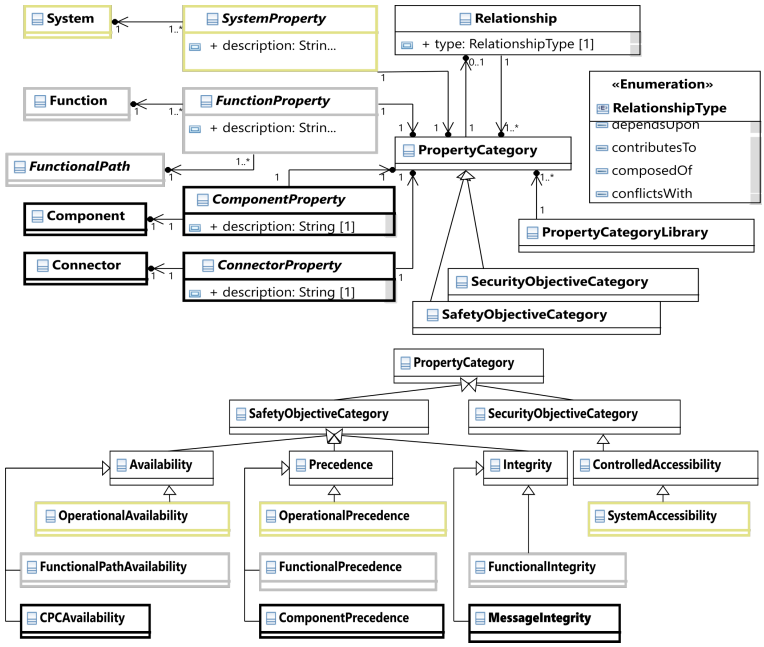


**Fig. 2.** Safety and security properties' specification meta-model.

Likewise, potential inter- or intra-relationships between safety and security objectives may arise within or across the layers. Thus, we aim to establish an alignment via links between objectives to support further analysis, including safety and security interplay. We define the association (e.g., *dependsUpon*, *conflictsWith*) between these objectives. For example, operational availability (mission layer) *depends upon* functional path availability (functional layer), system accessibility (mission layer) *conflicts with* operational availability (mission layer), etc.

## 3 Logical Specification

This section presents the formalization of the multi-layered system conceptual model defined in the previous section for rigorous specification and analysis of safety and security objectives. We use FOL and Modal Logic [4] as technology-independent formalisms to incorporate formal syntaxes for the system model and properties belonging to the defined categories. They play as a pivot language that aims to facilitate model interpretation and improve flexibility, *e.g.,* to delegate the properties' analysis to other tooled-formal languages and frameworks. In our context, the analysis has been mechanized, thanks to the Rodin framework, as presented in the next section.

### 3.1 Formal Syntax for Multi-layered System Specification and Properties

The meta-model in Fig. 1 is attached with a syntax, preserving the associations and types in the UML models (see Table 1). The formal syntax and logic defining some properties' signatures across the layered models are introduced in Table 2 and are discussed below:

- **Operational Availability:** A system $S$ in a certain operational situation should allow for the realization of safety-critical operation(s) $\mathrm{Op}_j$, whenever a hazardous event is detected.
- **Functional Path Availability:** A functional path FP := ($\mathrm{F}_1$,..., $\mathrm{F}_k$), $k \in \mathbb{N}$ satisfies functional path availability denoted by ***FunctionalPathAvailability***(FP) iff:
  - There is a preservation of the information flow between orderly execution of functions constituting the functional path, i.e., ***FunctionalIntegrity***(FP), and
  - The timeliness of the execution of the functional path is ensured for a predefined threshold $\Theta \in \mathbb{R}^+$, i.e., ***ExecutionTimeliness***(FP, $\Theta$).
- **Component-Port-Connector (CPC) Availability:** Given two components $\mathrm{C}_i$ and $\mathrm{C}_j$ (with failure rates $\lambda_i$ and $\lambda_j$, respectively), where $i$, $j \in \mathbb{N}$, a connector $Conn$ (with failure rate $\lambda_{Conn}$) connecting $\mathrm{C}_i$ and $\mathrm{C}_j$, time instant $t$, a message $m$, and a predefined threshold $\Delta \in \mathbb{R}^+$, the CPC availability objective is satisfied iff:
  - Availability of components is ensured, i.e., ***ComponentAvailability***($\mathrm{C}_i$, $\lambda_i$, $t$),
  - Availability of connector is ensured, i.e., ***ConnectorAvailability***(Conn, $\lambda_{Conn}$, $t$),
  - Delivery of critical messages between the components is ensured, i.e., ***EventualMessageDelivery***($\mathrm{C}_i$, $\mathrm{C}_j$, $m$) (or ***BoundedMessageDelivery***($\mathrm{C}_i$, $\mathrm{C}_j$, $m$, $\Delta$)), and
  - There exists logical conformity of the component architecture (i.e., configuration) with functional architecture.
- **System Accessibility:** A system $S$ must allow and limit the access of security-critical operation(s) $\mathrm{Op}_j$ to only authorized entities (e.g., Operator).

Here, we integrate the formal syntaxes presented in Tables 1 and 2 to define formal semantics for the layered elements. The FOL-based formalism of safety and security objectives' specifications are extended using a range of modalities, including $\circ$ (*next*), $\Diamond$ (*eventually*), $\Diamond_{\leq\Theta}$ (*bounded eventually*, where $\Theta$ denotes the threshold or bounded gap between two events/occurrences/actions etc.), and $\Box$ (*always*) for capturing the notion of future [4]. Likewise, $\bullet$, $\blacklozenge$, $\blacklozenge_{\leq\Theta}$, and $\blacksquare$, respectively denote their past counterparts. These are defined on top of standard FOL operators, including $\wedge$ (*conjunction*), $\vee$ (*disjunction*), $\neg$ (*negation*), $\rightarrow$ (*implication*), and $\leftrightarrow$ (*equivalence*). For example, predicates of the form $P \Rightarrow Q$ can be interpreted as $\Box(P \rightarrow \Diamond Q)$. Similarly, predicates of the form $P \Leftrightarrow Q$ can be interpreted as $\Box(P \leftrightarrow Q)$. Here, $\Rightarrow$ means *strongly implies* and $\Leftrightarrow$ means *strongly equivalent*.

### 3.2 Safety and Security Interplay

*Interplay within Mission Layer.* Consider a use case scenario where collision avoidance is the high-level mission of the CDV. The manual triggering of brakes to avoid the collision situation may hinder the denial of manual operation access to the operator in cases where safety is prioritized over security.

**Table 1.** Excerpt of Mapping: Three-layered conceptual model ↦ Formal syntax.

| Conceptual element | Formal syntax |
|---|---|
| System | S := ({$M_i$}, Environment), $i \in \{1 \dots \mathbb{N}\}$, Environment ∈ {People, Asset} |
| Mission | $M_i$ := ({$Op_j$}, $operationalSituation$, $hazardousEvent$, $securityIncident$, $overallGoal$), $i, j \in \{1 \dots \mathbb{N}\}$ |
| Operation | $Op_j$ := ($operationInput$, $operationOutput$, $preCondition$, $postCondition$, $environmentTriggerCondition$, $durationOfExecution$, $durationType$), $j \in \{1 \dots \mathbb{N}\}$, $durationType \in \{Function, Parameter, Value\}$ |
| Function | $F_k$ := ($type$, $preCondition$, $triggerType$, $caller$, $functionalInterfaces$, Information), $type \in \{Physical, Logical\}$, $triggerType \in \{Manual, ControlSignal, TemporalConstraint\}$, TemporalConstraint ∈ {Duration, TriggerTime}, $caller \in$ {Operator, $F_l$, TemporalConstraint}, $k, l \in \{1 \dots \mathbb{N}\}$ |
| Information | Information ∈ {FunctionalInput (or $I_k$), FunctionalOutput (or $O_k$)}, $k \in \{1 \dots \mathbb{N}\}$ |
| InformationFlow | InformationFlow := ({$(O_i, I_j) \stackrel{\Delta}{=} (F_i, F_j)$} \| FunctionalOutput[$F_i$] = $O_i$, FunctionalInput[$F_j$] = $I_j$), $i, j \in \{1 \dots \mathbb{N}\}$ |
| FunctionalPath | FP := ($F_1$,...,$F_k$) \| $\forall(F_k, F_{k+1})$, $\exists (O_k, I_{k+1}) \in \{(O_i, I_j)\}$, $i, j, k \in \{1 \dots \mathbb{N}\}$ |
| Component | $C_m$:= ({$P_n$}, $type$, $failureRate$, $givenTime$), $m, n \in \{1 \dots \mathbb{N}\}$, $type \in \{Atomic, Composite\}$ |
| Port | $P_n$:= ($kind$), $n \in \{1 \dots \mathbb{N}\}$, $kind \in \{Input, Output\}$ |
| Connector | Conn:= ($connType$, $P_n$, $P_o$, CommunicationStyle), $connType \in \{Bus, Pipe, Channel\}$, $n, o \in \{1 \dots \mathbb{N}\}$ |

In such cases, conflicts between safety (namely operational availability) and security (namely system accessibility) objectives can be identified by analyzing the predicates. More concretely, for any states in which all post-conditions are not satisfied simultaneously. For example, consider the following predicate at the mission layer specification, where $Op_1$:= braking and $Op_2$:= access provisioning, respectively denote the safety and security-critical operations performed by the CDV for the accomplishment of the mission $M_1$: collision avoidance:

$$accomplishedThrough[M_1, Op_1, Op_2] \Rightarrow (overallGoal[M_1] \Leftrightarrow (\neg postCondition[Op_1] \vee \neg postCondition[Op_2]))$$

Using the semantics defined in Section 3.1, the above predicate is specified as:

$$\Box(accomplishedThrough[M_1, Op_1, Op_2] \rightarrow \Diamond(overallGoal[M_1] \leftrightarrow (\neg postCondition[Op_1] \vee \neg postCondition[Op_2])))$$

Thus, no simultaneous fulfillment of post-conditions belonging to $Op_1$ and $Op_2$ after mission accomplishment shall indicate, in particular, a conflict between the safety and security objectives of the SUD.

*Interplay across Mission and Functional Layers.* Consider a situation when the trigger condition of an operation becomes true within the duration of another operation's execution. For example, the premature deployment of the vehicle's airbag during braking, which is still in progress without effective crash condition. In such cases, the conflict may occur at the level of high-level mission specification since the two missions' overall goals may have conflicting requirements.

**Table 2.** Excerpt of Mapping: Properties' metamodel ↦ Formal syntax.

| Conceptual element | Formal syntax |
|---|---|
| OperationalAvailability | $(operationalSituation[\mathrm{M}_i] \wedge hazardousEvent[\mathrm{M}_i]) \Rightarrow accomplishedThrough[\mathrm{M}_i, \mathrm{Op}_j]$ |
| FunctionalPathAvailability | a) ***FunctionalIntegrity***(FP):<br>***FunctionalPrecedence***$(\mathrm{F}_k, \mathrm{F}_{k+1}), \forall k \in \{1...\mathbb{N}\text{-}1\}: \mathrm{F}_j \Rightarrow \mathrm{F}_i$<br>***InformationEquivalence***$(\mathrm{O}_k, \mathrm{I}_{k+1}), \forall k \in \{1...\mathbb{N}\text{-}1\}: \mathrm{I}_j \Leftrightarrow \mathrm{O}_i$<br>b) ***ExecutionTimeliness***(FP, Θ): $\mathcal{Occ}(\mathrm{F}_1, \mathrm{F}_1, t_1, \mathrm{I}_1) \Rightarrow_{\Theta} \mathcal{Occ}(\mathrm{F}_{k-1}, \mathrm{F}_k, t_{2k}, \mathrm{O}_k)$ |
| CPCAvailability | a) ***ComponentAvailability***$(\mathrm{C}_i, \lambda_i, t)$ b) ***ConnectorAvailability***$(\mathrm{Conn}, \lambda_{Conn}, t)$<br>c) ***EventualMessageDelivery***$(\mathrm{C}_i, \mathrm{C}_j, m)$: $send(\mathrm{C}_i, m) \Rightarrow receive(\mathrm{C}_j, m)$<br>(or ***BoundedMessageDelivery***$(\mathrm{C}_i, \mathrm{C}_j, m, \Delta)$): $send(\mathrm{C}_i, m) \Rightarrow_{\Delta} receive(\mathrm{C}_j, m)$<br>d) Logical conformity of component architecture with functional architecture |
| SystemAccessibility | $privilege[\mathrm{Operator}, \mathrm{S}, \mathrm{Op}_j] \Rightarrow access[\mathrm{Operator}, \mathrm{Op}_j]$ |

In such cases, conflicts between objectives can be identified during the operations' (i.e., $\mathrm{Op}_j$) realization to accomplish different missions. Hence, for the overall system model at the mission layer, the following predicate is violated:

$$environmentTriggerCondition[\mathrm{Op}_j] \Rightarrow preCondition[\mathrm{Op}_j]$$

Using the semantics provided in our context, this predicate is specified as:

$$\Box(environmentTriggerCondition[\mathrm{Op}_j] \rightarrow \blacklozenge(preCondition[\mathrm{Op}_j]))$$

To illustrate the interplay with the functional layer, we consider a functional path comprising a set of functions (i.e., $\mathrm{F}_k$) and the link *realizedBy* between the operations and the functions. Herein, the assurance of functional integrity for the functional path realizing a sequence of operations is assumed. However, the delay introduced by any constituent function during execution may violate functional path freshness, influencing the timely realization of the operations. Hence, the above mission-layer predicate can be analyzed with the details offered by the functional layer using the following predicate:

$$(triggerTime[\mathrm{F}_1] > 0 \wedge trigger[\mathrm{F}_2]) \Rightarrow (triggerTime[\mathrm{F}_1] + duration < triggerTime[\mathrm{F}_2])$$

Using the semantics provided in our context, this predicate can be specified as:

$$\Box((triggerTime[\mathrm{F}_1] > 0 \wedge trigger[\mathrm{F}_2]) \rightarrow (triggerTime[\mathrm{F}_1] + duration < triggerTime[\mathrm{F}_2]))$$

*Interplay within Component Layer.* Consider a scenario comprising three system components, viz. $\mathrm{C}_1$:= processing unit, $\mathrm{C}_2$:= multi-function control unit, and $\mathrm{C}_3$:= brake actuator, engaged in transmitting a message $m$:= braking command, from $\mathrm{C}_1$ to $\mathrm{C}_3$ via $\mathrm{C}_2$, i.e., $\mathrm{C}_1 \longrightarrow_m \mathrm{C}_2$ and $\mathrm{C}_2 \longrightarrow_m \mathrm{C}_3$. Herein, we consider eventual message delivery, influencing CPC Availability, and Non-duplication, influencing message freshness, as the safety and security objectives to be respectively satisfied. We assume that all these

components are legitimate. However, if $C_2$ misbehaves and becomes faulty, it may tamper with $m$ to $m'$, leading to the following flow: $C_1 \longrightarrow_m C_2$ and $C_2 \longrightarrow_{m'} C_3$.

Thus, even after tampering, the non-duplication objective is satisfied as none of the recipients (viz. $C_2$ and $C_3$) undergo the repeated transmission of the message. However, the eventual message delivery for $m$ is not satisfied as $C_3$, the intended recipient, does not receive the original message $m$. Hence, in such scenarios, the safety and security analysis should not be conducted as standalone but integrated for the assurance of corresponding objectives, especially across the full chain of transmission.

Hence, in our context, for the overall system model targeting the component layer details, the aforementioned aspect can be analyzed via the following predicate:

$$send(C_i, m) \Rightarrow receive(C_j, m) \wedge \neg(receive(C_i, m))$$

$$send(C_i, m) \Rightarrow receive(C_j, m)$$

Using the semantics provided before, the predicate above can be specified as follows:

$$\Box(send(C_i, m) \rightarrow \Diamond(receive(C_j, m)) \wedge \neg(\blacklozenge(receive(C_i, m))))$$

$$\Box(send(C_i, m) \rightarrow \Diamond(receive(C_j, m)) \wedge \neg(\blacklozenge(receive(C_j, m))))$$

The first predicate is not expected to be satisfied; otherwise, the message sent by $C_2$ to $C_3$ is not the same as the one sent by $C_1$ to $C_2$. The second predicate should be satisfied whenever the uniqueness of the message is expected to occur.

## 4 Formal Specification and Analysis in Event-B

In this section, we consider interpreting the system and properties' meta-models and logical specification in the previous section into Event-B [1].

*Structural Elements.* The multi-layered system model is represented in Event-B via a set of *contexts* and *machines* refined within each layer. The *contexts* are used for the formal declaration of conceptual model elements, both structural and properties, along with the utility constants for generic representation of the elements. Herein, the key elements like Mission, Function, Component, and enumerations are defined as Event-B *sets*, while their attributes are represented as *constants*. Likewise, *axioms* capture the relationship between the elements and their attributes. To model the relation between a set $\mathbb{S}$ (e.g., Environment) and its sub-sets $s_1, s_2, \ldots, s_n$ represented as constants (e.g., People, Asset), we use the *partition* operator, i.e., $partition(\mathbb{S}, s_1, s_2, \ldots, s_n)$. An excerpt of this interpretation is depicted in Listing 1.1.

```
CONTEXT C1MissionView, C2FunctionView, C3ComponentView // for each layer
SETS System, Mission, Environment, Operation, Function, Information,
     TriggerType, Component, Port, Connector, CommunicationStyle,
     ConnectorType
CONSTANTS People, Asset, Manual, ControlSignal, TemporalConstraint,
     FunctionalPath, funcInfo, MsgPassing, Bus, Pipe, Channel, uses, connects
AXIOMS
partition(Environment, People, Asset) // Mission layer
```

```
partition(TriggerType, Manual, ControlSignal, TemporalConstraint) //
    Functional layer
FunctionalPath ∈ Function → Function
funcInfo ∈ Function ↔ Information
funcInfo~ ∈ Information ↠ Function
partition(CommunicationStyle, MsgPassing) // Component layer
partition(ConnectorType, Bus, Pipe, Channel)
uses ∈ Component → Port
uses~ ∈ Port ↠ Component
connects ∈ Connector → Port
connects~ ∈ Port ↠ Connector
```

**Listing 1.1.** Interpretation excerpt: Three-layered system model in Event-B.

Furthermore, *machine* specifications formally capture the desired behavioral aspects of the system as model *invariants*. The abstract machine specification provides an initial setup for the Event-B model that can be refined with concrete details. A machine *sees* a context to use its axioms in conjunction as hypotheses in the mathematical proofs. We define *events* in the machines targeting each layer to capture the state transitions associated with the application of operations, function execution, and CPC-based message transmission. Listing 1.2 depicts an excerpt of this interpretation for the mission layer.

```
MACHINE M1MissionView
SEES C4MissionUtility // Context for utility constants, e.g., s1, m1, o1
VARIABLES operationalSituation, hazardousEvent, overallGoal,
    environmentTriggerCondition, achieves, counter, accomplishedThrough,
    interactsWith
EVENTS HazardousEventPresence
WHEN
    operationalSituation = {m1 ↦ TRUE}
    hazardousEvent = {m1 ↦ FALSE}
    overallGoal = {m1 ↦ FALSE}
    environmentTriggerCondition = {o1 ↦ FALSE}
    achieves = {s1 ↦ m1}
    counter > 0 // An integer counter for capturing timeout
THEN
    hazardousEvent ≔ {m1 ↦ TRUE}
    environmentTriggerCondition ≔ {o1 ↦ TRUE}
    accomplishedThrough ≔ {m1 ↦ o1}
    interactsWith ≔ {s1 ↦ a1}
    counter ≔ counter − 1
```

**Listing 1.2.** Excerpt of an Event-B machine at the mission layer specification.

*Properties Specification.* We define the safety and security objectives presented in Section 3.1 in Event-B as model *invariants* to verify that the defined events satisfy them during system model verification at different layers. Herein, the invariants are represented as the combination of state variables (i.e., the concept attributes) and logic symbols, i.e., propositions (e.g., ⇒, ⇔) and predicates (e.g., ∀, ∃) between them. The *guards* restrict the values of the variables as enabling conditions for the events. An excerpt of the Event-B interpretation for the *Operational Availability* objective is given in Listing 1.3.

```
INVARIANTS
∀m.∃o.m∈ Mission ∧ o ∈ Operation ∧ (operationalSituation[{m}] = {TRUE} ∧
    hazardousEvent[{m}] = {TRUE}) ⇒ accomplishedThrough[{m}] = {o}
```

**Listing 1.3.** Operational availability objective as Event-B invariant.

Of particular interest here, objectives like *Functional Path Availability* are defined as extensions of basic properties like *Functional Precedence*, *Information Equivalence*, and *Execution Timeliness*. Then they are captured in Event-B via invariant decomposition, where each basic property is associated individually with an invariant.

*Properties Analysis.* To reason and verify the invariants, we rely upon mathematical proofs comprising a set of rules. These rules are based upon the convention of Proof-Obligations (POs) supported by the Rodin platform. A PO is generated for every invariant that can be affected by an event, i.e., the invariant contains variables that an event can change. The hypothesis for these POs relies upon the satisfaction of all the invariants (including behavioral, safety and security objectives, and gluing) and the validity of the guards restricting the values of the variables before the triggering of events in every reachable state of the system. The correctness of the instantiated model is dependent on ninety-one POs[4] that are related to mission accomplishment through the execution of the operation, function, and CPC-centric events.

To illustrate the analysis, we present the extraction of the PO for the satisfaction of the operational availability objective as an invariant, relying upon two invariants INV1 and INV2, and a theorem THM, as follows:

1. **MissionAllocation (INV1):** $\forall$m $\in$ Mission, $\exists$o $\in$ Operation | ((operationalSituation[m] $\wedge$ hazardousEvent[m]) $\Leftrightarrow$ (preCondition[o] $\wedge$ environmentTriggerCondition[o])) $\Rightarrow$ accomplishedThrough[m] = o; holds for all the events.
2. **MissionConsistency (INV2):** $\forall$m $\in$ Mission, (operationalSituation[m] $\wedge\neg$ hazardousEvent[m]) $\Rightarrow$ $\neg$ overallGoal[m]; holds for all the events.
3. **MissionAccomplishment (THM):** $\forall$m $\in$ Mission, o $\in$ Operation | (accomplishedThrough[m] = o) $\Rightarrow$ (postCondition[o] $\Leftrightarrow$ overallGoal[m]); a derived axiom that relies upon INV1 and INV2.

Likewise, the properties' conflict identification predicate presented in Section 3.2, for instance, at the mission layer, is presented in the Listing 1.4 as an Event-B invariant.

```
INVARIANTS
∀m.∃o1.∃o2.m ∈ Mission ∧ o1 ∈ Operation ∧ o2 ∈ Operation ∧ accomplishedThrough
    [{m}] = {o1, o2} ⇒ (overallGoal[{m}] = {TRUE} ∧ (¬postCondition[{o1}] =
    {TRUE} ∨¬postCondition[{o2}] = {TRUE}))
```

**Listing 1.4.** Mission layer properties conflict identification predicate as Event-B invariant.

For the safety and security missions SAFE_M and SEC_M, respectively, in the use case (see Section 2), the PO corresponding to the invariant in Listing 1.4 is discharged, depicting a potential conflict between braking and access provisioning operations. The reason can be attributed to the lack of privilege to the operator to realize the braking operation manually (i.e., Operator $\mapsto$ CDV1 $\mapsto$ Braking $\notin$ *privilege*).

## 5 Related Work and Positioning

So far, numerous solutions have been proposed in the literature to address safety and security concerns during the SE process (e.g., [17]). However, the challenge is reconciling

[4] Distribution of POs: Variable initialization (32), System specification (48), Safety and security invariants (11).

interdisciplinary knowledge, domain-oriented goals, and standalone practices towards effective co-engineering. This section dwells on some related works and positions our approach, emphasizing system conceptual modeling frameworks, design and analysis methods, and tooled-formal methods.

The mission is defined as a context-oriented and multi-paradigmatic concept in [13] (e.g., goal and object-oriented) that facilitates entanglement of system functionality with architectural design process. In light of this, the authors in [6] presented a mission-based process for wave development cycle, thus resulting in less generic than our approach. Nonetheless, these works cover only the requirement engineering phase. To address high-level functional specification and application of safety-oriented practices during critical systems' design, Design Space Exploration (DSE) capabilities are used in [18]. Component-based design techniques and frameworks are proposed concerning safety [14] and security [5] requirements in automotive CPSs applications by modeling and analyzing real-time component interactions. 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).

Tooled-formal methods, when used in the engineering process of safety- and security-critical systems, increase confidence in the aspects, e.g., properties, defined by the respective standards [11,12]. Several works demonstrate the use of the Event-B formal method for rigorous analysis of safety [10] and security [17] concerns. However, these works are constrained by the granularity level and concepts chosen for modeling what imposes requirements to be specified at the same level. The independent formal interpretation of the layers facilitates engineers to use them in standalone or coupled mode, adequate for granularity and analysis needs. Being technology-agnostic, the proposed approach provides further flexibility regarding the choice of specification/modeling languages and Validation and Verification (V&V) tool support.

## 6 Conclusion and Perspectives

We have proposed a joint design and analysis formal approach for integrated specification, modeling, and verification of safety and security properties. In this work, we mainly focused on the formalization aspects to address the ambiguities associated with the properties' specification and their interpretation w.r.t. the system design to conduct provable, sound analyses. To this end, the approach leveraged multi-layered system conceptual modeling, targeting high-level mission, functional, and detailed CPC-related concepts and their relationships. Indeed, the incorporated notions include properties' categories and signatures, a basis for joint safety and security analyses. Subsequently, the logical specification of the conceptual model was described using set theories, FOL, and Modal Logic as technology-independent formalisms. Based on this formalization, a model interpretation into Event-B was defined to conduct automated verification of safety and security objectives' signatures. However, being aware of the possibility of more complex configurations, like use cases calling upon a large set of objectives (e.g., those within the *Freshness* category), we plan to explore and analyze them to increase the approach's confidence. We are also interested in investigating the negative view of the properties where fault/failure/hazard and threat models are introduced.

## References


1. Abrial, J.R., et al.: Rodin: an open toolset for modelling and reasoning in Event-B. International journal on software tools for technology transfer **12**(6), 447–466 (2010)
2. Babel, K., Cheval, V., Kremer, S.: On the semantics of communications when verifying equivalence properties. Journal of Computer Security **28**(1), 71–127 (2020)
3. Bau, J., Mitchell, J.C.: Security modeling and analysis. IEEE Security & Privacy **9**(3), 18–25 (2011)
4. Bull, R., Segerberg, K.: Basic modal logic. In: Handbook of philosophical logic, pp. 1–88. Springer (1984)
5. Chattopadhyay, A., Lam, K.Y., Tavva, Y.: Autonomous vehicle: Security by design. IEEE Transactions on Intelligent Transportation Systems (2020)
6. Cherfa, I., Belloir, N., Sadou, S., Fleurquin, R., Bennouar, D.: Systems of systems: From mission definition to architecture description. Systems Engineering **22**(6), 437–454 (2019)
7. De Miguel, M.A., Briones, J.F., Silva, J.P., Alonso, A.: Integration of safety analysis in model-driven software development. IET software **2**(3), 260–280 (2008)
8. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st international conference on Software engineering. pp. 411–420 (1999)
9. Fuentes-Fernández, L., Vallecillo-Moreno, A.: An introduction to UML profiles. UML and Model Engineering **2**(6-13), 72 (2004)
10. Hong, Z., Lili, X.: Application of software safety analysis using event-b. In: 2013 IEEE Seventh International Conference on Software Security and Reliability Companion. pp. 137–144. IEEE (2013)
11. ISO 26262-1:2018 Road vehicles — Functional safety. `https://www.iso.org/standard/43464.html` (2018)
12. ISO/IEC 27000:2018 Information technology — Security techniques — Information security management systems. `https://www.iso.org/standard/73906.html` (2018)
13. Letier, E., Van Lamsweerde, A.: Deriving operational software specifications from system goals. ACM SIGSOFT Software Engineering Notes **27**(6), 119–128 (2002)
14. Masrur, A., Kit, M., Matěna, V., Bureš, T., Hardt, W.: Component-based design of cyber-physical applications with safety-critical requirements. Microprocessors and Microsystems **42**, 70–86 (2016)
15. Pedroza, G., Apvrille, L., Knorreck, D.: 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. pp. 1–10. IEEE (2011)
16. Rodin: Rodin Platform (2021), https://wiki.event-b.org/
17. Vistbakka, I., Troubitsyna, E.: Towards a formal approach to analysing security of safety-critical systems. In: 2018 14th European Dependable Computing Conference (EDCC). pp. 182–189. IEEE (2018)
18. Wan, J., Canedo, A., Al Faruque, M.A.: Cyber–physical codesign at the functional level for multidomain automotive systems. IEEE Systems Journal **11**(4), 2949–2959 (2015)