JOT
JOURNAL OF
OBJECT TECHNOLOGY

A Unified Model-Driven Framework for Co-Designing System Architecture and Safety Analysis in Cyber-Physical Systems

Muhammad Asim Minhas*, **Erik Burger***, **Ralf Reussner***, **and Tianhai Liu***
*Karlsruhe Institute of Technology, Germany

ABSTRACT Designing safety-critical Cyber-Physical Systems (CPS), such as self-driving cars, is inherently complex. A fundamental challenge arises from the separation between architecture modeling and safety analysis, which typically rely on different languages, tools, and modeling perspectives. As a result, engineering teams must maintain multiple, partially overlapping models of the same system. Keeping these models synchronized and consistent is labor-intensive, repetitive, and error-prone, leading to increased development cost and a higher risk of inconsistencies. We propose a unified modeling framework built on a deeply integrated metamodel foundation. Unlike Systems Modeling Language (SysML) profiles, which provide syntactic integration, our approach semantically unifies two EMF-based metamodels: a custom SysML implementation for system structure and a manually transformed Risk Analysis and Assessment Modeling Language (RAAML) metamodel for safety analysis. This integration enables a cohesive modeling environment in which architectural models and safety analyses are developed as synchronized views over a single, shared underlying system representation. As a result, consistency between system architecture and safety analysis is maintained implicitly by construction. We demonstrate that this approach provides a more robust, analyzable, and less error-prone foundation for model-based safety engineering than approaches based on dynamically applied profiles.

KEYWORDS Model-based Safety Engineering (MBSE), SysML, RAAML, Safety Analysis, Cyber-Physical Systems (CPS), Metamodel Integration

1. Introduction

Model-based Systems Engineering (MBSE) aims to support the specification, design, analysis, and validation of complex systems by systematically using formal models. By capturing requirements, structure, behavior, interfaces, properties, and parameters in explicit modeling artifacts, MBSE helps engineers reason about complex systems and manage their development across multiple abstraction levels and lifecycle phases. Standardized modeling languages play a central role in this process, most notably the Systems Modeling Language (SysML) (Object Management Group 2024), an extension of the Unified Modeling Language (UML), and the Risk Analysis and Assessment Modeling Language (RAAML) (Object Management Group 2021) for safety analysis.

However, addressing heterogeneous engineering concerns remains a significant challenge for MBSE. Different stakeholders and analysis activities require specialized viewpoints, languages, and tools, resulting in multiple models that describe overlapping aspects of the same system. MBSE, therefore, relies on the notion of views, often realized through SysML diagram types such as Block Definition Diagrams and Activity Diagrams, to accommodate these diverse concerns. Since these views are developed using different formalisms and tools, their underlying models are interdependent but weakly integrated. Safety analyses, such as Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA), exemplify this challenge: they

JOT reference format:
Muhammad Asim Minhas, Erik Burger, Ralf Reussner, and Tianhai Liu. *A Unified Model-Driven Framework for Co-Designing System Architecture and Safety Analysis in Cyber-Physical Systems*. Journal of Object Technology. Vol. 25, No. 3, 2026.
http://dx.doi.org/10.5381/jot.2026.25.3.a17

are frequently performed separately by safety engineers using dedicated tools and are introduced late in the design process. As a consequence, safety-driven design decisions are deferred, making late-stage changes costly and challenging to incorporate (Lai et al. 2021).

Another major challenge arises from the gap between system models and safety analyses. In current practice, safety engineers must first extract relevant information from the system model before performing the safety analysis. Because system models evolve continuously throughout development, safety models derived from them quickly become outdated (Mhenni et al. 2013). Maintaining consistency, therefore, requires the explicit creation and management of trace links or manual synchronization between models defined over distinct semantic domains. This manual alignment is complex, labor-intensive, and costly, and it significantly increases the risk of inconsistencies between architectural and safety-related artifacts.

In summary, our main **research question** is: *What is the effectiveness of semantically integrating architecture and safety metamodels in consistency management of cyber-physical systems?* To answer this question, we propose three contributions. **C1**: We have created a model-driven framework that unifies the semantic foundations of system architecture and safety analysis. In contrast to conventional methods, which rely on loosely coupled profiles or tool integrations, our framework provides a single integrated metamodel that integrates the fundamental concepts from both SysML architecture models and RAAML safety analysis models. **C2**: We present a Sirius-based[1] modeling workbench in which multiple views are projected from a unified model that serves as a single source of truth. This enables automated synchronization between architecture and safety analysis views. This eliminates the need for manual model synchronization and ensures coherence throughout the development lifecycle. **C3**: Traceability views are automatically generated and updated once any model element is created or updated in any of the architecture, safety, or FMEA analysis views. The other benefits include streamlined model management and reduced semantic ambiguity, enabling multidisciplinary teams to collaborate more effectively and to accelerate the validation cycle for safety-critical engineering projects. This results in greater confidence in both design and safety integrity, as well as simplifying the regulatory compliance. The proposed approach provides a simplified co-design framework offering consistent traceability from hazards to components and behavior. The integration overhead across tools and teams can be reduced.

The remainder of the paper is structured as follows. We introduce a motivating running example in section 2. In section 3, we present the proposed framework and the architecture of the unified metamodel. The evaluation based on an Automotive Emergency Brake System case study is reported in section 4. The results and their implications are discussed in section 5. We then review existing approaches and related work in section 6, and conclude the paper with directions for future work in section 7.

[1] https://eclipse.dev/sirius/, retrieved 27/02/2026

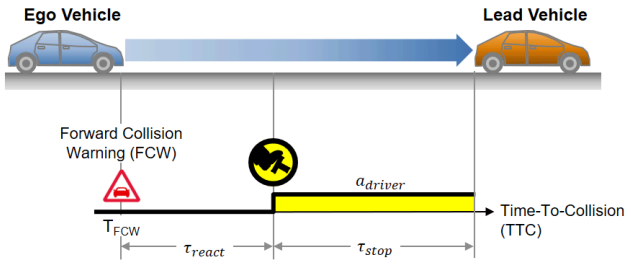

Figure 1 Autonomous Emergency Braking system architecture (adapted from (MathWorks 2025)).

2. Motivating Use Case

Modern vehicles rely on *Advanced Driver Assistance Systems (ADAS)* to provide road-safety features and reduce collision risk. Among these, *Autonomous Emergency Braking (AEB)* has emerged as a critical function, designed to prevent or mitigate frontal crashes when drivers fail to react in time (Figure 1). The AEB system utilizes on-board sensors (e.g., cameras, radar, and lidar) to perceive traffic conditions and evaluate the risk of collisions with pedestrians, remote vehicles, or other obstacles ahead. To avoid collision, the system automatically triggers the actuator to apply the necessary braking. The operational procedure of AEB can be subdivided into three phases: First, the AEB system (Yang et al. 2022) does not intervene in the regular operation of the vehicles as long as there is a potential chance of collision, which can be treated as an idle state for AEB. Second, the AEB generates an early warning to the driver immediately through audio alarms, visual warning signs, or tightening the safety belt. Third, the AEB, depending on the situation, either applies maximum braking or gradually enhances the braking pressure to avoid the potential collision. System-wise, AEB is considered safety-critical, as catastrophic consequences could arise if the system is unable to perform its intended functionality. Secondly, it is a hard real-time system with stringent timing requirements, meaning that any results produced late will be considered wrong. The other aspect is its importance as a life-saving component in today's automotive domain. Literature shows that AEB systems can reduce the number of traffic accidents and rear-end collisions by 25 % to 50 % (Cicchino 2017)

(Sharma et al. 2019) presents a safety and security analysis of AEB using *System Theoretic Process Analysis (STPA)* and defines some assumptions, accidents, and hazards associated with AEB. In particular, system-level hazards for AEB have been proposed. To validate our proposed unified approach, we use the AEB system along with the system hazards defined in (Sharma et al. 2019). We implement the architectural modeling and safety analysis views to showcase the benefits of the proposed unification between SysML and RAAML. We use STPA as a source to derive safety hazards, enabling us to perform architectural and safety analyses accordingly. We define the following safety hazards for AEB. **AH1:** Minimum Safe Distance (MSD) violation with Forward Mobile Objects. **AH2:** Entry into Prohibited Areas. **AH3:** Unsafe G-forces on occupants. **AH4:** Cybersecurity threat (sensor spoofing)

3. Proposed Framework

3.1. Overview and Core Philosophy

In traditional safety analysis, all safety artifacts are separately modeled and maintained from the architecture models, which leads to traceability challenges and often inconsistencies. To address these challenges and establish a semantic unification, we propose a unified safety metamodel. This metamodel provides a single, formal, and tool-implementable semantic foundation that addresses the gaps between system architecture, failure behavior, and safety analysis.

To understand the core philosophy behind the unified approach, consider the definition of safety from ISO 26262: *Safety is freedom from unacceptable risk of physical injury or damage to health.* (ISO 26262 2018) For example, a statement such as *There is a failure mode with severity 8* is meaningless on its own because it lacks the relevant context. Specific questions, such as which component exhibits this failure, which system function it affects, and under what conditions it becomes hazardous, remain unanswered in this statement. Instead, if we say: *The brake actuator has a failure mode* `Excessive_Braking_Force` *with severity 8* because it can harm the occupants, it would be meaningful and actionable. Our key insight is that safety is not an absolute entity, but a relational property that must be semantically bound to the architecture elements from which it emerged. Traditional approaches model architecture and safety concerns separately:

- SysML: Architecture only.
- FMEA Excel: Safety analysis only.
- Manual traceability: "Manually observing sensor block in SysML model."

In our approach, safety properties are attributes or relationships of architectural components and not separate artifacts about them.

3.2. Proposed Metamodel Structure and Semantics

An excerpt of the proposed metamodel is displayed in Figure 2. For unification, we bring together two standard OMG languages. i.e., architectural concepts from SysML and the safety concepts from RAAML. In our previous work (Minhas et al. 2025), we identified a novel technique for transforming SysML profiles into EMF. Therefore, we assume that ECore representations of both SysML and RAAML exist. The metamodel is implemented as an Ecore metamodel using EMF. The architecture of the metamodel provides an interconnected conceptualization of system architecture, failure behavior, and safety analysis as a single, consistent source of truth. The design eliminates gaps between architecture and the safety domain, providing automated, bidirectional traceability and consistency by design.

3.2.1. Foundational Core: Unified Element

All entities within the metamodel inherit from the abstract UnifiedElement class, which provides shared properties id and name. This common concept provides a uniform foundation for identification, traceability, and management across the system. (The element is not displayed in Figure 2 for reasons of compactness.)

3.2.2. Architectural Foundation: System Structure

The physical and logical architectures are captured through the *SystemBlock* hierarchy. A SystemBlock represents a modular part of the system, starting from the entire product down to its smallest sub-assembly. It also supports decomposition via the parentBlock and subBlocks references to enable a natural part-of representation of the system. Blocks interact through two primary types of relationships:

- BlockConnection is used to model dynamic flows between blocks such as data, control, signal, or power (connectionType), specifying a source fromBlock and a target (toBlock). This provides definitions for modeling functional operation and potential failure propagation.
- BlockAssociation is used to capture static and structural relationships like dependency, aggregation, or composition (associationType) between a sourceBlock and a targetBlock

The root container UnifiedSystemModel composes all instances of these architectural elements, providing the complete system context.

3.2.3. Safety Requirements and Safety Goals

To incorporate safety requirements and safety goals, the metamodel has been extended with Requirement class, which provides shared attributes such as requirementID, requirementText, status, verificationMethod, and rationale. It has references for hierarchical decomposition derivedFrom and architectural realization satisfiedBy. SafetyGoal represents top-level safety objectives derived from hazard analysis. Its attributes are asilLevel and safeState. Goals are decomposed into FunctionalSafetyRequirements, which represent function-level obligations, and into TechnicalSafetyRequirements that represent design-level obligations through allocatedASIL and a link validatedBy to the corresponding FMEAItem. Finally, SafetyMechanism represents countermeasures, such as watchdogs, lockstep execution which implements the TechnicalSafetyRequirement, resides within a SafetyCriticalBlock and is validatedBy an FMEAItem.

3.2.4. Safety-Critical Specialization: Embedding Failure Logic

The fundamental unification is achieved by specializing architectural blocks to address safety concerns. A SafetyCriticalBlock EClass extends a SystemBlock to inherit all of its structural properties to add safety-specific semantics.

- It is assigned a safetyCriticality level (from LOW to CRITICAL) in order to prioritize analysis efforts.
- Most importantly, it owns one or more BlockFailureMode instances. A BlockFailureMode represents a specific way a component can fail (e.g, "open circuit", "value drift"), and has a bidirectional link to its affectedBlock.

This design implements the principle that failure behavior is not an external annotation but an inherent property of a safety-critical component. Hence, failure modes are now first-class citizens of the unified metamodel.

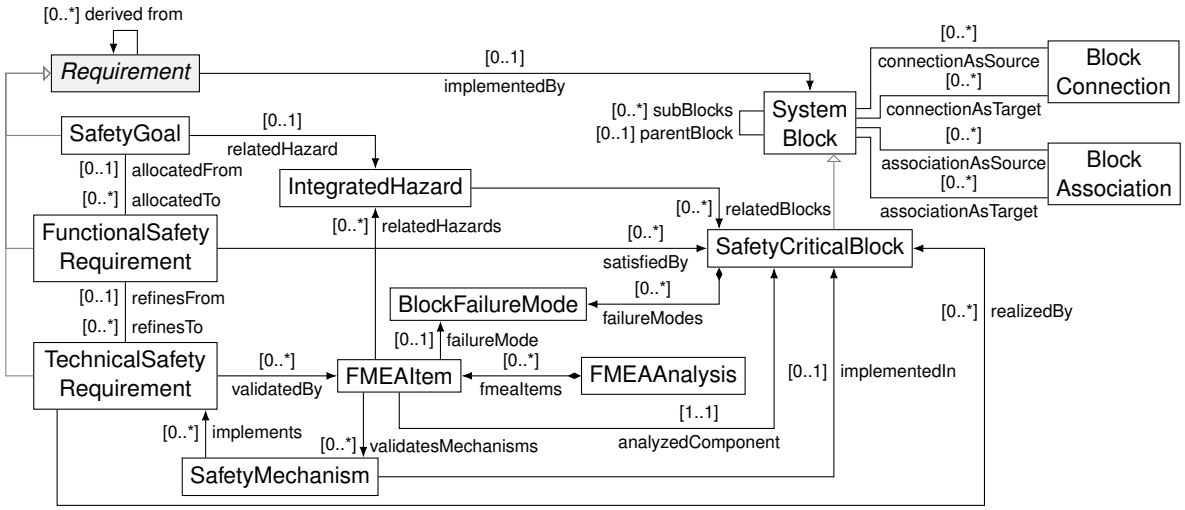

Figure 2 A unified metamodel for integration of SysML and RAAML concepts. (Attributes, enumerations, and the common superclass UnifiedElement are not shown. A complete diagram is available in the Github repository (Minhas 2026).)

3.2.5. System-Wide Risk: Integrated Hazards System-level risks are modeled as IntegratedHazard objects, which represent dangerous conditions arising from the interaction of multiple component failures. Each hazard is quantifiable using a riskLevel (from NEGLIGIBLE to CATASTROPHIC), and it is linked through relatedBlocks to the specific SafetyCriticalBlock that provides the context and manifestation for the hazard. This creates a direct, navigable link from a high-level system risk to the architectural elements.

3.2.6. Safety Analysis Integration: FMEA Workflow The metamodel integrates the safety analysis workflow, offering dynamic process tracking rather than a static description.

- An FMEAAnalysis container, contains a set of FMEAItem. Each FMEAItem provides a concrete analysis record for a specific BlockFailureMode of a particular analyzedComponent (SafetyCriticalBlock).
- It captures the classical FMEA metrics (severity, occurrence, detection), their derived Risk Priority Number RPN and textual effects (localEffects, systemEffect).
- It also models the mitigation lifecycle: a recommendedAction and its actionStatus (OPEN, IN_PROGRESS, VERIFIED, COMPLETED). This provides integration of the safety engineering process and its current status directly into the model.

3.2.7. Enabling Semantic Unification The effectiveness of the unified safety metamodel lies in its network of semantically precise references.

- From Hazard to Component: IntegratedHazard → relatedBlocks → SafetyCriticalBlock.
- From Analysis to Architecture: FMEAItem → (analyzedComponent + failureMode) → SafetyCriticalBlock and BlockFailureMode.
- From Failure to Function: BlockFailureMode → affectedBlock → (BlockConnection/BlockAssociation →) SystemBlocks.
- From Hazard to Goal: IntegratedHazard ← relatedHazard ← SafetyGoal.
- From Goal to Requirement: SafetyGoal → allocatedTo → FunctionalSafetyRequirement → refinedTo → TechnicalSafetyRequirement.
- From Requirement to Architecture: FunctionalSafetyRequirement → implementedBy → SafetyCriticalBlock; TechnicalSafetyRequirement → realizedBy → SafetyCriticalBlock.
- From Requirement to Mechanism: TechnicalSafetyRequirement ← implements ← SafetyMechanism → implementedIn → SafetyCriticalBlock.
- From Analysis to Mechanism: FMEAItem ↔ validatesMechanisms / validatedBy ↔ SafetyMechanism; TechnicalSafetyRequirement → verifiedBy → FMEAItem.

The interconnection ensures that changes in one view, such as deleting a component or updating a failure mode's severity, are automatically reflected in the related safety artifacts. These changes trigger semantic validation rules defined in the metamodel to ensure that architectural and safety elements remain consistent. In this way, inconsistencies among architectural elements, failure modes, hazards, and other safety artifacts can be detected early. The unified metamodel thus not only provides a unified vocabulary but also an analyzable network of engineering knowledge. This enables the creation of a semantic foundation necessary for automated analysis and synthesis of safety artifacts and also the generation of compliance evidence.

3.3. Foundational Design Principles for Unification

During the process of defining and refining the proposed metamodel, we identified generalized design principles that can serve as a unification criterion for semantically integrating multiple correlated domains.

Common Identification of Related Concepts: If there is a specific concept that represents the same logical or physical entity, then all of these concepts must share a common identity. The

benefit is that the traceability overhead of maintaining consistent identities is eliminated. For example, system engineers and safety experts may refer to the same elements, but with different identifiers; with the proposed approach, this traceability gap is addressed. Example of this principle in our proposed metamodel is UnifiedElement, which provides universal id and name attributes.

Properties should be contained: The properties should be contained inside their associated concept and should not exist separately. For example, safety properties should be embedded within or contained by architectural elements and not modeled independently. For example, SafetyCriticalBlock contains failureModes. The core advantage is that the gap between what the system does and its failure behavior is eliminated in this way.

Bidirectional Navigability of References: References that are navigable in both ways are helpful when impact analysis needs to be computed from both directions. For example, in the metamodel, there is a bidirectional reference SafetyCriticalBlock ↔ BlockFailureMode, which means that from every failure mode, it is possible to navigate to the component that exhibits this mode, and to determine all failure modes of a component conveniently. While bidirectional navigation should only be allowed where necessary to reduce dependencies, missing navigability leads to workarounds, such as helper functions or OCL queries, which are more expensive to compute. Navigability is also vital for performing various analyses, such as *Failure Propagation*, which computes the set of affected elements in case of component failure.

Separation between Architecture and Analysis: To keep separation of concerns, there should be an explicit way to distinguish between which elements are safety-critical and which are not. This is enforced by providing the concepts SystemBlock (without safety-critical behavior) and SafetyCriticalBlock, which extend the same idea to include safety-critical behavior. This allowed us to maintain the separation of concerns at the semantic level.

Integration of Analysis: Different analysis methods or types, such as FMEA and STPA, should reference the same unified architectural model. In our metamodel, FMEAAnalysis operates on SafetyCriticalBlock and IntegratedHazard represents STPA system-level hazards. Both analysis types refer to the same SafetyCriticalBlock, thereby avoiding the need for safety teams to maintain separate, inconsistent models.

3.4. Prototypical Implementation

We have implemented the proposed unified framework in Eclipse Sirius to provide multiview modeling capabilities and demonstrate the advantages of the adopted approach, as shown in Figure 3. We used Sirius because it provides relatively simple ways to develop creation, edition, and navigation tools on top of the EMF metamodels. The developed modeling workbench can be downloaded and used via the update site repository hosted at GitHub (Minhas 2026).

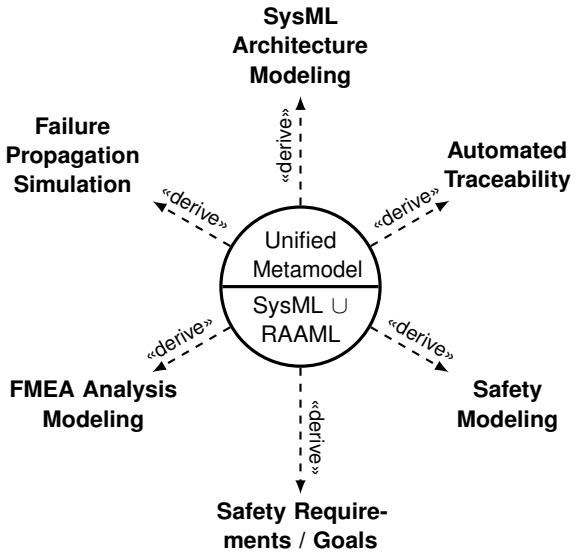

Figure 3 Unified framework implemented as a minimum viable prototype in Eclipse Sirius

3.4.1. Design decisions We have utilized the semantic foundations defined in our unified metamodel and, within a single viewpoint UnifiedModeling, created four distinct diagram types. ArchitectureDiagram, SafetyDiagram, FMEADiagram, and TraceabilityDiagram. Each diagram type held its own layer. For example, ArchitectureDiagram contains ArchitectureLayer, and so on. We have implemented all diagram types according to their definitions and the relationships defined in the unified metamodel. Different visual elements and styles were defined to represent each type of EClass, and each type of associations and relationships. We have implemented tool creation pallets for each diagram type to allow users to create elements of their choice via drag-and-drop in the diagram's user interface.

We have defined essential semantic consistency rules to enforce the validation and integrity of the semantics. For example, in our tool, we defined a semantic validation rule (Listing 1) that each FMEAItem must have severity, occurrence, and detection values defined.

```
aql: self.severity > 0 and self.occurrence > 0
    and self.detection > 0
```

Listing 1 AQL rule ensuring severity, occurrence, and detection are positive

3.4.2. Modeling Workflow The **system engineer** opens an *Architecture Diagram* to model the system structure by creating SystemBlocks (non-critical) and SafetyCriticalBlocks (critical components). They define BlockConnections (e.g., dataflow or controlflow) and BlockAssociations (dependency and aggregation), and then perform the validation that all blocks are properly named, and the connection types are properly defined.

The **safety expert** opens a *Safety Diagram* to perform hazard analysis. They create IntegratedHazards with risk levels, and then link IntegratedHazards to SafetyCriticalBlocks using the threatens relationship, add FailureModes to each SafetyCriticalBlock, and set safety-criticality levels. To perform FMEA

analysis, they open the *FMEA Diagram*, and all safety-critical blocks with their predefined failure modes, technical safety requirements, and safety mechanisms are already available. They create an FMEA item and link it to the related safety-critical block. The relationship between the technical safety requirement and the safety mechanism, and the FMEAItem, can be modeled using the tool palette. When creating the requirement diagram, a system engineer begins by instantiating SafetyGoals derived from the hazard analysis and linking each to its originating IntegratedHazard. These goals are then progressively decomposed into FunctionalSafetyRequirements and further into TechnicalSafetyRequirements, with each level maintaining bidirectional traceability to its parent. Finally, SafetyMechanisms are created and linked to the technical requirements they implement and the SafetyCriticalBlocks that host them, completing a fully traceable chain from hazard to architectural realization They set severity, occurrence, and detection values. The RPN is calculated automatically. Failure propagation simulation context model is created with a single click as shown in Figure 8. From the diagram, a user selects any block, including SafetyCriticalBlocks, with their embedded failure modes and linked hazards and triggers forward or backward propagation. The diagram immediately highlights which downstream (or upstream) blocks are affected and which connections carry the fault, giving a visual read of the blast radius of any single-component failure. Both roles mentioned above use the *Traceability Diagram*, which is automatically generated and updated for the elements in other views. Traceability is fully automated without any additional effort. Semantic validation rules enforce the completeness of the traceability view. It also supports multi-view consistency, with changes in one diagram/view automatically reflected in the others.

4. Evaluation

The proposed approach has been validated using two case studies. The automotive domain AEB case study and the medical infusion pump. Due to space limitations, we include only the models for the AEB case study; the medical infusion pump case study models are available in the GitHub repository (Minhas 2026). Ananda Adhikari et al. have performed a functional safety analysis of the AEB system based on ISO 26262 (Ananda Adhikari et al. 2023), e.g., Hazard and Risk Analysis (HARA), FMEA, or FTA. They also presented the item architecture (block definition) of the AEB system. Their models indicate that they had to manually link architectural elements to their corresponding safety artifacts. There is no bidirectional traceability between the presented models, meaning that both the architecture and the safety models evolve independently. For architectural modeling in our case study, we adapted the item architecture and offered a more refined block definition for AEB in our developed modeling workbench.

We model the structural elements of AEB as shown in Table 1. In the following subsections, we will discuss the unified modeling approach implemented on AEB using our developed modeling workbench.

ID	Component	Responsibility
	Safety-Critical Blocks	
SCB-01	Front_Radar_Sensor	Detects forward objects and measures range
SCB-02	AEB_Control_ECU	Computes braking decision, and issues Brake Force Commands (BFC)
SCB-03	Vehicle_Speed_Sensor	Measures vehicle/wheel speed for the braking decision
SCB-04	Sensor_Fusion_ECU	Fuses radar, camera, performs state estimation and Time-To-Collision (TTC)
	Supporting Blocks	
SB-01	CAN_Bus_Gateway	Routing messages between components
SB-02	Power_Supply_Unit	Supplies electrical power to system components
SB-03	Driver_HMI	Warns the driver and displays system status
SB-04	Non_Volatile_Memory	Persistent storage for calibration data and event/fault logging

Table 1 AEB System Architecture Components.

4.1. AEB Architecture Modeling

The architecture diagram is provided for the system engineer role. As shown in Figure 4, the architecture engineer opens the tool, then creates the safety-critical and supporting blocks according to the scheme in Table 1. Using the drag-and-drop tools, the engineer also defines the block connection by setting the appropriate connectionType properties, i.e., DATA_FLOW, CONTROL_FLOW, POWER_FLOW, SIGNAL_FLOW, or MECHANICAL. In the next step, the engineer finalizes the design by defining the associations among the previously described blocks. For example, in AEB architecture, AEB Control ECU depends on Path_Planning_Module, so an association edge is created from the source block(dependent), AEB_Controller, to the target block(dependee) Sensor Fusion ECU. The final step

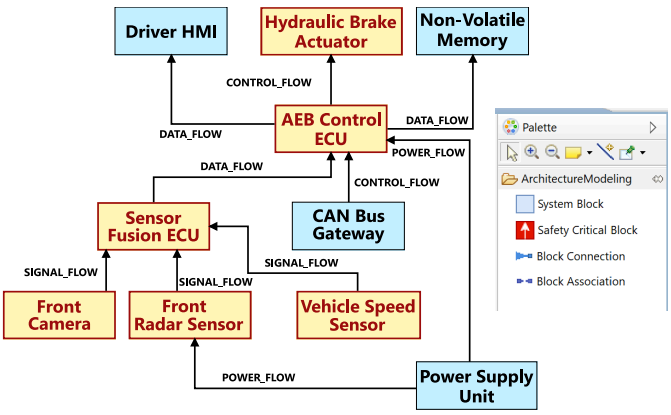

Figure 4 AEB Architecture Modeling in MVP

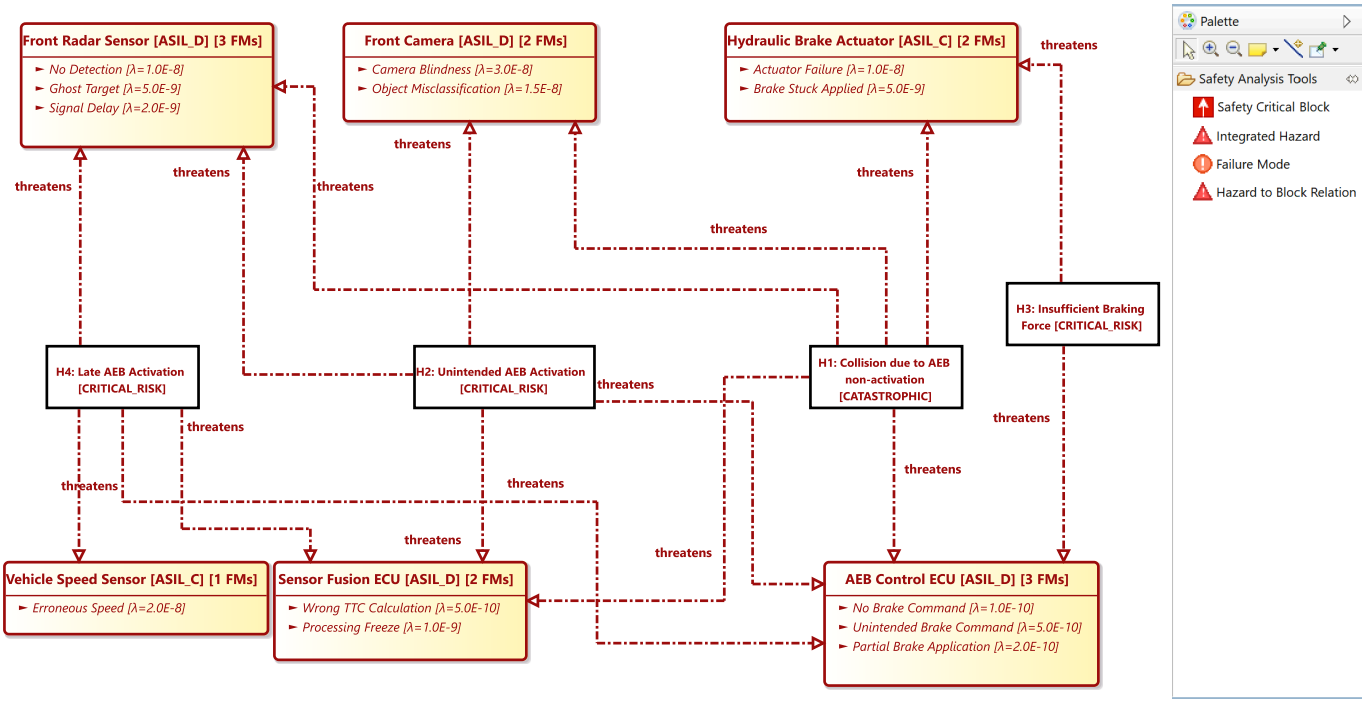

Figure 5 AEB Safety Modeling in MVP

is to perform validation, which includes implicit validation of the EMF and customized semantic validation rules that check the current model for any semantic violations. They are reported to the end user as warnings or errors. Tool-based validation is an effective means of enforcing explicit correctness checks.

4.2. AEB Safety Modeling

The safety diagram provides safety modeling tools to the safety experts. For AEB safety modeling, the moment the safety expert creates the safety diagram, their view automatically updates to include the safety-critical blocks that the architect engineers have previously added. In our proposed framework, we envision a collaborative workflow in which system engineers/architects work closely with safety experts. If additional SafetyCriticalBlock are required, the safety engineer can do it using the tool palette. First, a Failure Mode is defined for each SafetyCriticalBlock. As failure modes represent local failures, they are modeled within the corresponding SafetyCriticalBlock container. Next, system-level hazards are specified by defining IntegratedHazard. Each IntegratedHazard may be associated with multiple SafetyCriticalBlocks via HazardToBlockRelation edges, as illustrated in Figure 5.

4.3. AEB Safety Goals and Requirements Modeling

The Requirements Modeling diagram allows safety experts to define safety goals and how these goals address specific integrated hazards. These safety goals are allocated to Functional Safety Requirements (FSRs) and refined to Technical Safety Requirements (TSRs), finally connecting TSRs to Safety Mechanisms that implement them as shown in the Figure 6.

4.4. AEB Safety Analysis

For safety analysis, an FMEADiagram is created, as shown in Figure 7. Immediately after creation, all SafetyCriticalBlock elements that were previously created, along with their associated failure modes, are automatically created for FMEAAnalysis. To perform FMEA analysis, there are two options: either manually create the FMEAItem through the tool palette or just right-click on the diagram and select Auto-Generate FMEAItems. The system assigns default values to S/O/D and also sets the "OPEN" action status. The FMEAItem's label indicates the selected component, failure mode, computed RPN ($SXOXD$), and a visual risk category (low/medium/high). In the next step, an FMEAItem can be connected to a component ("analyzes") and, optionally, linked to Technical Safety Requirements for verification and to SafetyMechanisms for validation as depicted in Figure 7. The diagram includes an additional layer for filtering and focusing the view. For example, displaying all components, only high-risk items (RPN > 100), or only open actions. The semantic validation rules trigger warnings in case the FMEA input data (valid S/O/D ranges, component assignment, mitigation for high RPN) is missing or inconsistent.

4.5. AEB Failure Propagation Simulation

The Failure Propagation Simulation diagram provides an interactive way to explore how a failure could spread through the architecture using the model's existing blocks and block connections, as shown in Figure 8. The simulation can be triggered through any selected block as a failure source and any propagated/affected blocks are visually distinguishable as the simulation runs. Specific failure modes can also be selected as the source of failure propagation. The diagram provides additional layers

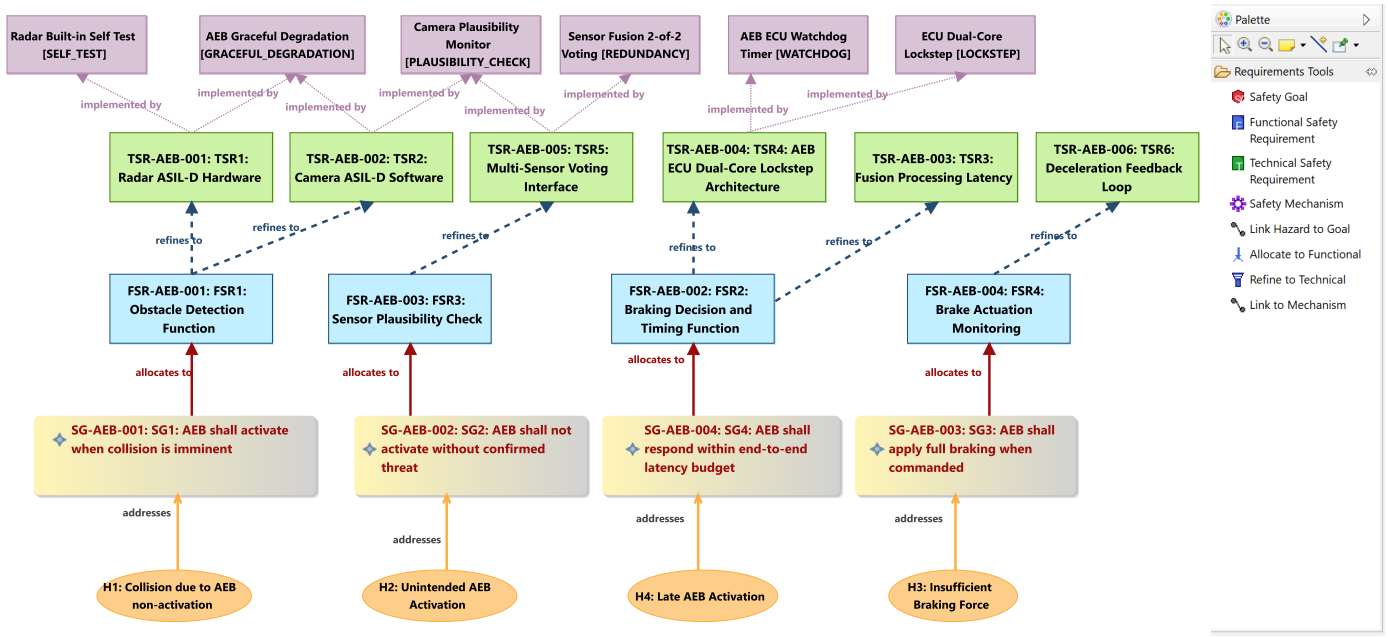

Figure 6 AEB Safety Goals and Requirements Modeling in MVP

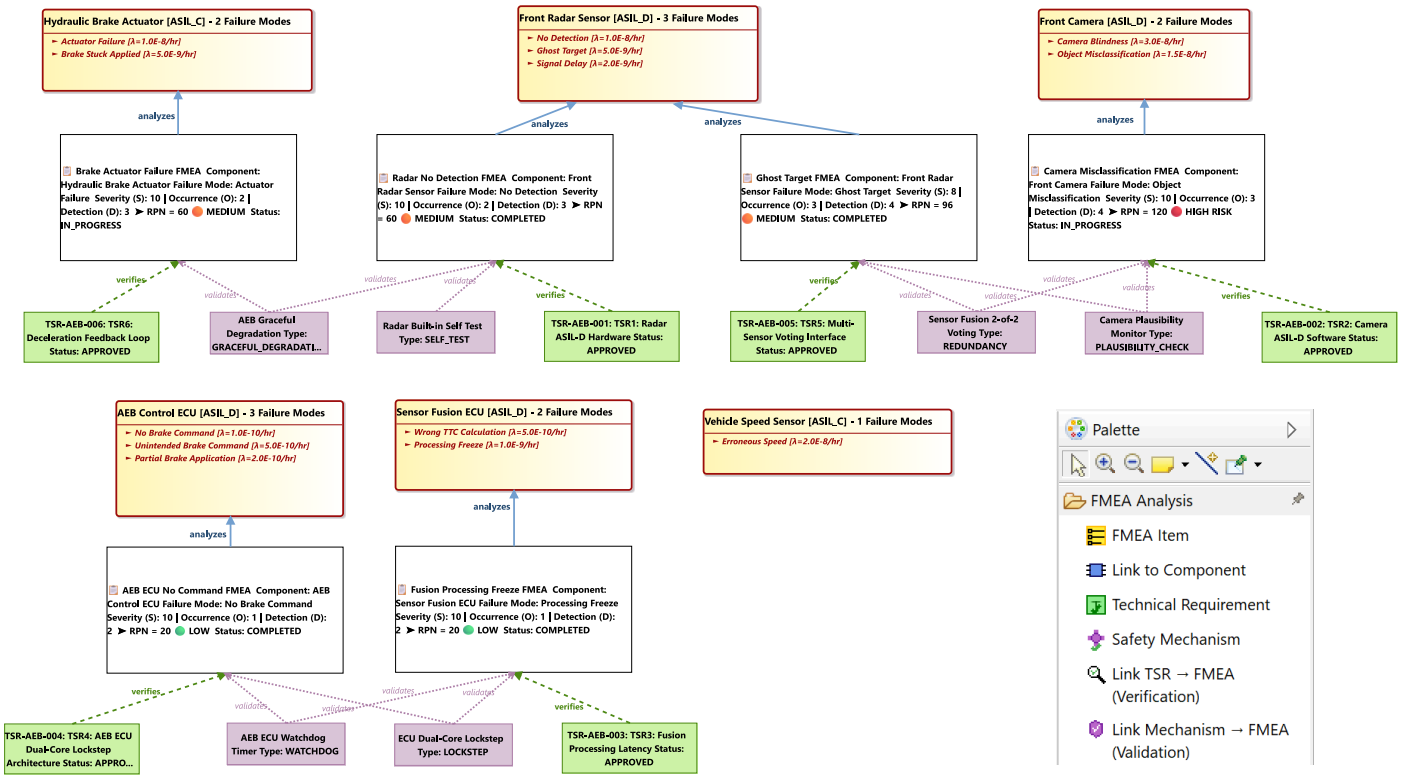

Figure 7 AEB FMEA Analysis Modeling in MVP

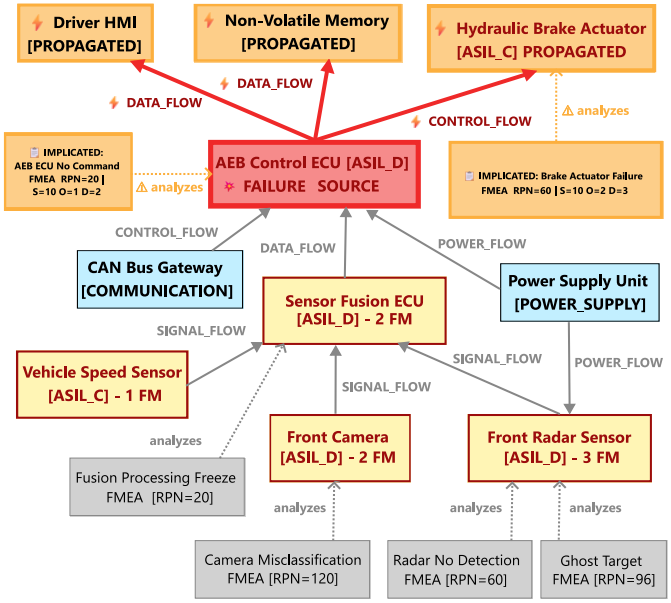

Figure 8 AEB Failure Propagation Simulation in MVP

to conditionally display and highlight the activated hazards and implicated FMEA Items, making it easier to see which hazards become relevant and which FMEA entries are affected by the simulated scenario. The simulation enables early exploration of the most critical impact paths.

4.6. Automatic Traceability for ISO 26262 Compliance

The Automated Traceability Diagram provides an end-to-end navigable view of how different artifacts are linked across the model which helps to confirm that the safety and design intent are consistently connected. In this diagram shown in Figure 9, Sirius automatically populates nodes for architecture elements(System Blocks and Safety-Critical Blocks) and safety/process artifacts (Hazards, FMEA Items, Safety goals, Functional Safety Requirements, Technical Safety Requirements, and Safety Mechanisms), then it draws the typed relationships such as threatens (Hazard $\rightarrow$ Block), addressedby (Goal $\rightarrow$ Hazard), allocates (Goal $\rightarrow$ FSR), refines (FSR $\rightarrow$ TSR), realized by (TSR/FSR $\rightarrow$ Block), and verified by (TSR/Mechanism $\leftrightarrow$ FMEA). The view helps identify missing links and drive impact analysis by following a specific forward or backward-tracing chain. This provides a strong foundation for conformance with ISO 26262 (Clause 9) requirements for safety-oriented analysis and with traceability requirements (Clause 8). The unified framework covers major lifecycle artifacts by providing end-to-end traceability across them.

4.7. Sirius-based Modeling Workbench

We implemented the unified metamodel using Eclipse Sirius to demonstrate the workflow and effectiveness of the unified approach. While Sirius is widely adopted in academic and industrial modeling contexts, we emphasize that the *conceptual unified metamodel* is tool-agnostic; its Ecore-based implementation can, in principle, be ported to other modeling frameworks.(e.g. Xtext for textual syntax, web-based diagramming libraries). To evaluate the approach across different safety-critical domains, we developed two case studies: an Autonomous Emergency Braking (AEB) system and a Medical Infusion Pump. For each system, architecture modeling was performed using block definition diagrams, during which SafetyCriticalBlocks were also identified. The workbench provides all necessary types of associations and connections to ensure consistent system modeling.

The tool supports requirements modeling, including safety goals, functional and technical safety requirements and specifying safety mechanisms. These requirements can be linked to the system components and safety artifacts. This enables end-to-end traceability between architecture, requirements, and analysis models.

The next phase was to use safety modeling, in which 14 failure modes were defined and mapped to their respective SafetyCriticalBlocks for AEB and to an analogous set for the infusion pump. Then, four system-level hazards (IntegratedHazards) were defined and linked to the affected blocks as an individual system-level hazard could affect multiple SafetyCriticalBlocks. Safety modeling not only specifies failure modes and system-level hazards but also sets the stage for automated FMEA safety analysis. In the subsequent FMEA analysis diagram, FMEAItems are defined and then linked to their respective SafetyCriticalBlock and FailureModes. The end user is expected to provide severity, occurrence, and detection, and the Risk Priority Number (RPN) is calculated automatically. Finally, the traceability diagram is automatically generated with a single click, which is particularly helpful for standard compliance. Using the same architectural model, the tool enables the creation of a failure-propagation simulation context in which a safety-critical node serves as the source of propagation, and its effects on other components can be simulated in both forward and backward directions. Overall, the implementation across both automotive and medical domains shows that the proposed platform effectively supports integrated architectural modeling, safety analysis, requirements engineering, and compliance-oriented traceability. These results demonstrate that the approach is practical for co-designing complex CPS in representative industrial case studies.

5. Discussion

To discuss the results of the evaluation, we have formulated the following questions:

- **Q1:** To what extent does the proposed framework provide intrinsic consistency between architectural models and safety artifacts?
- **Q2:** How effectively does the framework integrate safety analysis and provide bidirectional traceability from system hazards to the failure modes of SafetyCriticalBlocks?
- **Q3:** Can the framework automatically generate traceability diagrams between all modeling elements related to architecture and safety?
- **Q4:** How practical is the proposed platform for co-designing architecture and safety concerns for developers of cyber-physical systems?

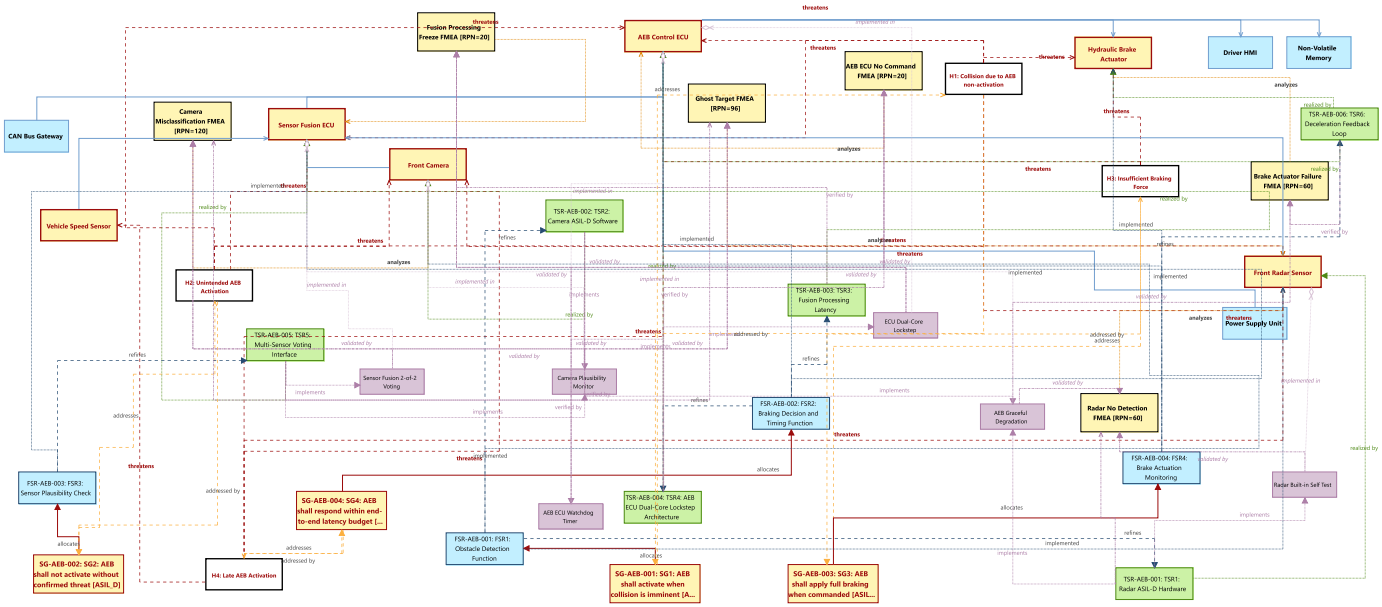

Figure 9 Automatically Generated Traceability Model in MVP

5.1. Advantages of Proposed Approach

The evaluation that we have performed is qualitative and does not provide any metrics on improvements over the state of the art. We can, however, give answers to the research questions Q1–Q4 based on the qualitative results:

The evaluation of the AEB case study introduced in section 2 demonstrates that, in contrast to existing SysML profile-based extensions, our proposed unification of architecture and safety offers several benefits: Safety is considered an inherent property of the system elements rather than a separately specified concern. This semantic unification has eliminated the gaps in architecture and safety modeling. The unified framework provides a common platform for both system architects and safety engineers to collaborate. As shown in the case study description, these roles complement each other. For example, a system architect creates a SafetyCriticalBlock, which is automatically available to a safety engineer as a starting point for modeling safety concerns in a safety diagram. This offers consistency by design (Q1), as semantic validation rules can be defined and enforced during design and safety analysis. For example, input to the analysis can be validated before the safety analysis itself. The other aspect is enabling impact analysis, which means understanding how safety models or concerns affect architectural design decisions, and vice versa. As a result, design-related anomalies and safety-analysis or mitigation-related issues can be detected earlier in the design and modeling phase, reducing rework in later stages of development.

The other core advantage is automated traceability among all model elements (Q3), enabled by a semantically unified approach. This is an implicit traceability, an essential requirement of many standards such as ISO 26262, which requires that all artifacts, including safety goals and requirements, design models, analysis-related artifacts, and assurance cases, should be bidirectionally traceable. The proposed approach supports bidirectional traceability (Q2) without any additional effort, making conformance to relevant standards easier and less costly. It also shifts the standard conformance concept from document-centric to tool-aided. The unified approach integrates the safety analysis itself, eliminating the need to rely on external tools, such as in (Clegg et al. 2019a). Safety analysis inputs are implicitly available, and the analysis is integrated into the framework without relying on external tools. The failure propagation simulation capability is a direct benefit of the unified metamodel: since architectural blocks, failure modes, hazards, and FMEA items all coexist in a single model, the tool can automatically construct the full simulation context without any manual setup (Q4). For the technical realization, an investigation of the scalability of Eclipse Sirius is subject to future work; the authors of Sirius Web state that the framework is supposed to accomodate models with millions of elements and views with thousands of elements (Giraudet et al. 2024).

In total, the proposed approach provides benefits for development, and conformity to industry standards. Rather than relying on model transformations and manual synchronization, our approach maintains architectural and safety artifacts within a single, semantically unified model instance. Semantic unification reduces fragmentation between architecture and safety engineering activities. It offers automated traceability across system components, hazards, and failure modes, reducing manual effort to synchronize models and minimize inconsistencies. Early detection of design issues and effective safety hazard mitigation strategies become possible, as both architecture and safety concerns can be analyzed collaboratively by system architects and safety experts. Automated traceability accelerates the adoption ISO 26262. This framework enables strong collaboration between safety engineers and system architects. This facilitates a co-design process in which safety properties directly influence architectural decisions and vice versa.

5.2. Limitations

The scope of this work was to demonstrate and validate the feasibility of the unified metamodel and its associated modeling workflow using two representative industrial safety-critical case studies. Although, these examples cover distinct domains (automotive and medical), broader domain-specific evaluations would further strengthen the approach's generalizability. Similarly, the semantic validation rules implemented in the prototype focus on the core constraints needed to support the workflow presented in this paper. Future work will extend the current validation rules toward a full industrial compliance set. Especially for questions Q2 and Q4, future evaluations will provide quantitative measures to assess integration effectiveness, such as analyzing the completeness and correctness of integrating representative models.

Finally, although the prototype was implemented in Eclipse Sirius for rapid exploration and demonstration, the proposed metamodel is tool-agnostic and independent of any specific tooling platform. In future the proposed unified framework will be assessed across alternative modeling environments to investigate quantitative measures, such as modeling effort or defect reduction, to complement the qualitative observations reported in this study.

6. Related Work

6.1. Model-driven Approaches for Architecture and Safety Co-design:

Cyber-Physical Systems (CPS) require compliance with stringent safety standards (ISO 26262 2018; RTCA DO-178C/ED-12C 2011). To meet these standards, rigorous safety analysis activities must be performed at each phase of development. In the ISO 26262 V-Model, every development activity is associated with a corresponding safety-focused analysis activity, spanning from the definition of system requirements through integration and deployment. Demonstrating the system's safety properties further requires constructing a safety argument substantiated by concrete evidence and safety artifacts (Walden et al. 2023). These activities are often time-consuming, costly, and difficult to execute effectively.

Model-Based Systems Engineering (MBSE) employs SysML-based modeling to capture requirements, structure, behavior, and parameters (Walden et al. 2023). SysML models act as a backbone for communication across heterogeneous domains, supporting traceability from high-level system goals down to detailed architectural decisions. Several approaches exist that try to integrate architectural concerns with safety analysis to bridge the gap between architectural design and safety assessment (Brüggemann et al. 2020; Krueger et al. 2016). SysML profile-based extensions dominate integrated system-safety modeling. (Clegg et al. 2019a) discussed their experience of integrating fault trees to SysML safety profile with the assumption that analysis execution itself is external to the modeling environment. The purpose of this profile was to enable the export of the fault logic to external tools by capturing only the minimum information required. (Hu et al. 2019) extended SysML to define a profile representing fault behavior and events and mapping rules between AltaRica event modeling language and SysML models to support Model-Based Safety Analysis (MBSA). First, the SysML model is converted to AltaRica, and then the safety analysis module flattens the generated model, generates fault tree analysis, and performs step-wise simulation.

(Roudier & Apvrille 2015) introduce a model-driven framework that enables system designers and security experts to work together at the earliest stages of development. It addresses the challenge that adding security mechanisms might compromise safety. The proposed approach is based on extended SysML diagrams to jointly model safety and security concerns, including requirements, attacks, and software-hardware partitioning. The approach was implemented as a tool called *TTool*, equipped with its own model checker. Still, it also supports external provers for formal verification of both safety and security properties. (Denney et al. 2017) presents a model-driven approach for developing safety architectures for complex safety-critical systems, demonstrating it on NASA's unmanned aircraft operations. The proposed approach was implemented on the *AdvoCATE* toolset, providing instrumentation to obtain regulatory compliance for beyond-visual-line-of-sight drone flights. (Mhenni et al. 2018) proposes a SysML-based methodology that embeds safety analysis into systems engineering workflows. SysML is extended with safety-specific concepts and information to generate Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) from these models. This ensures the consistency between design models and safety artifacts. Early integration helps to reduce late-stage errors. The methodology was demonstrated through a case study on electromechanical actuators. (Clegg et al. 2019b) introduced another SysML profile to integrate FTA elements into SysML models, thereby linking failure modes to system components and functions, thereby improving traceability. The framework enhances consistency and allows automated fault tree generation. The approach also supports external analysis tools.

6.2. Integration Approaches: Syntactic vs Semantic

The approaches discussed in subsection 6.1 are mostly based on SysML profiles, which offer weak, syntactic integration. Profiles primarily add stereotypes and tags without changing the underlying language semantics, leading to *ambiguous or imprecise semantics*: SysML profiles inherit the complexity and ambiguity of UML, making constraint interpretation and consistency checking difficult (Wang et al. 2019). SysML is not a high-level formal language suitable for automatic analysis. For example, there is no way to verify that a SysML diagram satisfies a given property. (Quamara et al. 2022a) addressed early co-designing of safety and security by introducing a multi-layered modeling approach to capture functional, safety, and security concerns across architecture layers. The framework supports formal verification techniques to check properties at different abstraction levels, ensuring consistency and standards compliance. Extending the same techniques, the authors in (Quamara et al. 2022b) introduced a formal model-driven method for co-engineering safety and security in CPS. They also proposed reusable libraries of properties defined in formal specifications across multiple layers and used domain-specific modeling to ensure consistent

propagation of constraints. The approach was validated using Rodin. (Bakirtzis et al. 2021) introduced an ontological metamodel to unify concepts of security, safety, and resilience for CPS. These domains are formalized within a shared ontology to enable consistent reasoning across heterogeneous models and to support traceability of risks and mitigation strategies. The proposed metamodel provides a foundation for automated analysis and assurance case generation. A model transformation between SysML and AltaRica was presented to address the gap between system design and formal safety analysis (Nguyen et al. 2020). As AltaRica supports quantitative dependability, transformation between the two languages ensures semantic consistency and automatically generates safety models from SysML diagrams. The paper (Pennington et al. 2024) integrates System Theoretic Process Analysis Extended for Coordination (STPA-Coord) into SysML using RAAML while applying a profile-based extension mechanism. The technique enabled model-based safety analysis of complex systems-of-systems. The demonstration is presented via a case study from the military systems domain. The proposed approach emphasizes the benefits of a model-based framework that leverages RAAML and STPA over document-based methods.

6.3. Application Domains and Standards Compliance

(Tietz et al. 2021) address certification of domain-specific modeling tools. The authors identified six foundations: metalanguage, deterministic transformations, runtime, qualification artifact generation, visualizations, and modular component interaction. Their study demonstrates how metamodeling can support tool qualification workflows and regulated development lifecycles. In (Kharatyan et al. 2021a), a metamodel is proposed to integrate safety and security domains with system architecture modeling. It supports traceability among threats, hazards, and architectural elements. The approach was applied to a case study in the automotive domain, aligned with ISO 26262 and ISO 21434 for cybersecurity compliance. (De Miguel et al. 2008) introduce a Safety Modelling Framework (SMF) that comprises metamodels, profiles, model transformations, and evaluators of metrics to support analysis and development of safety-aware UML architectures. Safety analysis metamodels such as MECA (failure modes, effects, and criticality) analysis, FTA, and safety modeling framework (SMF) interfaces support SMF. Modeling transformations generate automated safety analysis models. The framework is validated through a case study in air traffic control within an RTCA DO-178B-compliant environment. An integrated safety and security metamodel has been presented in (Kharatyan et al. 2021b) to enable co-engineering and trade-off analysis of attack and failure propagation. It offers safety by design, which is ideal for ISO 26262 standard-regulated development environments. (Mokos et al. 2010) is an ontology-based MDE approach to automate safety verification. To enable reasoning and consistency checking, safety semantics and system models are combined using ontological inference. The work is implemented in the safety-critical systems domain, using IEC 61508 principles to achieve functional safety assurance.

6.4. Identified Research Gap and Contribution

The reviews in section 6 demonstrate that there is progress in integrating systems engineering with safety analysis using model-based techniques. SysML profiles provide improved traceability and consistency to a certain degree, and enable the linkage of safety analysis artifacts like FTA and FMEA to architectural models (Clegg et al. 2019b; Mhenni et al. 2018). However, syntactic integration via SysML is not well-suited to support analysis. Semantic integration approaches based on metamodeling and ontologies offer better analyzable unification of concepts, and also support automated reasoning and transformation. (Bakirtzis et al. 2021; Quamara et al. 2022b; Nguyen et al. 2020). While there has been significant progress in the unification of architecture and safety analysis, a critical gap remains at the intersection of theoretical unification and practical, validated tool support. SysML profiles offer pure syntactic integration, which provides the benefit of sharing a notation but without having deep formal semantics. These formal semantics are essential to perform automated and reliable analysis. (Wang et al. 2019). The other aspect is the gap between analysis modeling and analysis execution, which is often external to the modeling environment, creating additional challenges for consistency and traceability. However, advanced semantic approaches that create unified metamodels or ontologies often remain theoretical or are otherwise validated through limited-case transformations (e.g., SysML to AltaRica (Nguyen et al. 2020)). SysML to AltaRica (Nguyen et al. 2020)). Enforce unification end-to-end throughout the development lifecycle.

Unlike prior work, which emphasizes the formal analysis and automation, our proposed framework provides model integration and viewpoint consistency within an Eclipse Modeling Framework (EMF) environment. By leveraging Sirius (Pérez et al. 2015), we enable customizable diagrammatic representations for SysML and RAAML concepts. This provides a strong foundation for future extensions, such as the generation of automated artifacts and the application of incremental consistency checks. The proposed approach delivers immediate value to CPS design and safety teams by improving traceability and enabling structured co-modeling.

7. Conclusion and Future Work

In this paper, we propose a novel semantic unification approach for system architecture and safety modeling and present a unified metamodel. Unlike existing approaches, which either rely on model transformations, external tools, or SysML profile extensions that provide only weak syntactic integration, our approach is based on the semantic integration of relevant concepts, with intrinsic links between architecture and safety. We conceptualize safety as a property of the system and define it as part of the system architecture. To eliminate the gap between architecture and safety, we combined architectural modeling elements from SysML with the safety modeling elements from RAAML to provide a coherent modeling framework in which both system architects and safety experts can work together. This framework provides a foundation for architecture and safety co-design. We developed a Sirius-based modeling workbench to demonstrate

the applicability and effectiveness of the proposed framework. The evaluation through two case studies, the AEB case study from the Automotive domain and the medical infusion pump from the safety-critical medical domain, demonstrated that the proposed approach successfully addresses the existing fragmentation and gaps between architecture and safety models. It ensures consistency by construction and provides bidirectional automated traceability between all artifacts. The framework provides modeling support of safety goals, functional and technical safety requirements as well as safety mechanisms to mitigate hazards. Failure-propagation simulation helps prioritize safety mitigations and testing by identifying which downstream components, hazards, and FMEA items are most affected by a given failure scenario. Automated safety analysis integration and execution become possible due to this semantic unification as demonstrated through FMEA analysis of the AEB case study. As a result, conformance with relevant standards, such as ISO 26262, becomes easier and more streamlined.

We envision extending the unified framework to include additional analysis types, such as security, performance, probabilistic failure propagation, and automatic ASIL decomposition. Furthermore, AI-assisted FMEA generation can be employed to suggest S/O/D ratings. Formal verification engines such as theorem provers and constraint solvers can be integrated to support safety property proofs. These extensions would enable the unified framework to provide a more complete and formally rigorous foundation for ISO 26262-compliant safety engineering. It will support full lifecycle from hazard identification and requirement derivation through verification, validation and the generation of certification evidence.

8. Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC 1608 – 501798263.

References

Ananda Adhikari, Aspak Saban, Shirish Bohora, & Viranchi Shastri. (2023, August). Functional Safety for Automatic Emergency Braking based on ISO 26262: Paper No.: 2023-GI-04. *ARAI Journal of Mobility Technology*, *3*(3), 666–685. doi: 10.37285/ajmt.3.3.4

Bakirtzis, G., Sherburne, T., Adams, S., Horowitz, B. M., Beling, P. A., & Fleming, C. H. (2021, June). An ontological metamodel for cyber-physical system safety, security, and resilience coengineering. *Software and Systems Modeling*, *21*(1), 113–137. doi: 10.1007/s10270-021-00892-z

Brüggemann, O., Vogel-Heuser, B., & Törngren, M. (2020). Model-based systems engineering and safety analysis of cyber-physical systems: A survey of integration approaches. *Journal of Systems and Software*.

Cicchino, J. B. (2017). Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates. *Accident Analysis & Prevention*, *99*, 142–152. doi: 10.1016/j.aap.2016.11.009

Clegg, K., Li, M., Stamp, D., Grigg, A., & McDermid, J. (2019a). Integrating Existing Safety Analyses into SysML. In Y. Papadopoulos, K. Aslansefat, P. Katsaros, & M. Bozzano (Eds.), *Model-Based Safety and Assessment* (Vol. 11842, pp. 63–77). Cham: Springer International Publishing. doi: 10.1007/978-3-030-32872-6_5

Clegg, K., Li, M., Stamp, D., Grigg, A., & McDermid, J. (2019b). A SysML Profile for Fault Trees—Linking Safety Models to System Design. In A. Romanovsky, E. Troubitsyna, & F. Bitsch (Eds.), *Computer Safety, Reliability, and Security* (Vol. 11698, pp. 85–93). Cham: Springer International Publishing. doi: 10.1007/978-3-030-26601-1_6

De Miguel, M., Briones, J., Silva, J., & Alonso, A. (2008, June). Integration of safety analysis in model-driven software development. *IET Software*, *2*(3), 260–280. doi: 10.1049/iet-sen:20070050

Denney, E., Pai, G., & Whiteside, I. (2017, September). Model-Driven Development of Safety Architectures. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (pp. 156–166). Austin, TX: IEEE. doi: 10.1109/MODELS.2017.27

Giraudet, T., Bats, M., Blouin, A., Combemale, B., & David, P.-C. (2024). Sirius Web: Insights in Language Workbenches - An Experience Report. *The Journal of Object Technology*, *23*(1), 1-20. doi: 10.5381/jot.2024.23.1.a6

Hu, J., Tang, H., Kang, J., & Wang, H. (2019, October). A Model Based Safety Analysis Framework for SysML and A Case Study. In *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)* (pp. 1846–1853). doi: 10.1109/EITCE47263.2019.9094927

International Organization for Standardization. (2018). *Road vehicles – Functional safety – ISO 26262-1:2018* (Tech. Rep.). Geneva, Switzerland.

Kharatyan, A., Tekaat, J., Japs, S., Anacker, H., & Dumitrescu, R. (2021a). Metamodel for safety and security integrated system architecture modeling. *Proceedings of the Design Society*, *1*, 2027–2036. doi: 10.1017/pds.2021.464

Kharatyan, A., Tekaat, J., Japs, S., Anacker, H., & Dumitrescu, R. (2021b, August). Metamodel for Safety and Security Integrated System Architecture Modeling. *Proceedings of the Design Society*, *1*, 2027–2036. doi: 10.1017/pds.2021.464

Krueger, P., Felderer, M., & Pretschner, A. (2016). Towards the integration of safety and architecture analysis. In *Computer safety, reliability, and security (safecomp)*. Springer.

Lai, K., Robert, T., Shindman, D., & Olechowski, A. (2021, July). Integrating Safety Analysis into Model-Based Systems Engineering for Aircraft Systems: A Literature Review and Methodology Proposal. *INCOSE International Symposium*, *31*(1), 988–1003. doi: 10.1002/j.2334-5837.2021.00882.x

MathWorks. (2025). *Autonomous emergency braking with sensor fusion.* MATLAB & Simulink Documentation. Retrieved from https://www.mathworks.com/help/driving/ug/autonomous-emergency-braking-with-sensor-fusion.html (Accessed: 2025-12-10)

Mhenni, F., Nguyen, N., & Choley, J.-Y. (2013). Towards the Integration of Safety Analysis in a Model-Based Sys-

tem Engineering Approach with SysML. In M. Haddar, L. Romdhane, J. Louati, & A. Ben Amara (Eds.), *Design and Modeling of Mechanical Systems* (pp. 61–68). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-37143-1_8

Mhenni, F., Nguyen, N., & Choley, J.-Y. (2018, March). SafeSysE: A Safety Analysis Integration in Systems Engineering Approach. *IEEE Systems Journal*, *12*(1), 161–172. doi: 10.1109/JSYST.2016.2547460

Minhas, M. A. (2026). *Archsafe unified modeler.* Retrieved from https://github.com/masimminhas/ArchSafeUnifiedModelerUpdateSite (GitHub repository)

Minhas, M. A., Lüntzel, V., & Burger, E. (2025). Towards a unified model-based engineering framework: Integrating UML profiles into EMF. In *ICMM 2025 – (In-)Consistency Management in Modeling @ STAF 2025.* CEUR Workshop Proceedings. Retrieved from https://ceur-ws.org/Vol-4122/paper7.pdf

Mokos, K., Meditskos, G., Katsaros, P., Bassiliades, N., & Vasiliades, V. (2010, September). Ontology-Based Model Driven Engineering for Safety Verification. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 47–54). Lille, TBD, France: IEEE. doi: 10.1109/SEAA.2010.60

Nguyen, N., Mhenni, F., & Choley, J.-Y. (2020). A Study on SysML and AltaRica Models Transformation. *2020 IEEE International Systems Conference (SysCon)*, 1-6. doi: 10.1109/SysCon47679.2020.9275868

Object Management Group. (2021). *Risk Analysis and Assessment Modeling Language Specification Version 1.0.* Retrieved 2025-12-05, from https://www.omg.org/spec/RAAML/1.0/About-RAAML

Object Management Group. (2024). *SysML® v1 Specification.* Retrieved 2025-12-09, from https://www.omg.org/sysml/sysmlv1/ (version 1.7)

Pennington, E., Johnson, K., Colombi, J., & Hobbs, K. (2024, July). Integrating STPA Extended for Coordination into SysML Using RAAML. *INCOSE International Symposium*, *34*(1), 749–762. doi: 10.1002/iis2.13175

Pérez, M., Coulon, S., Syriani, E., et al. (2015). Eclipse sirius demonstration. In *Proceedings of the models 2015 demonstrations track.* CEUR-WS. Retrieved from https://ceur-ws.org/Vol-1554/PD_MoDELS_2015_paper_4.pdf

Quamara, M., Pedroza, G., & Hamid, B. (2022a). Facilitating Safety and Security Co-design and Formal Analysis in Multi-layered System Modeling. In *2022 IEEE DASC/PiCom/CBDCom/CyberSciTech* (pp. 1–8). Falerna, Italy: IEEE. doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927773

Quamara, M., Pedroza, G., & Hamid, B. (2022b). Formal Analysis Approach for Multi-layered System Safety and Security Co-engineering. In S. Marrone et al. (Eds.), *Dependable Computing – EDCC 2022 Workshops* (Vol. 1656, pp. 18–31). Cham: Springer International Publishing. doi: 10.1007/978-3-031-16245-9_2

Roudier, Y., & Apvrille, L. (2015). SysML-Sec: A model driven approach for designing safe and secure systems. In *2015 3rd international conference on model-driven engineering and software development (modelsward)* (pp. 655–664). ESEO, Angers, Loire Valley, France: SCITEPRESS – Science and and Technology Publications. doi: 10.5220/0005402006550664

RTCA, Incorporated. (2011). *Software Considerations in Airborne Systems and Equipment Certification – DO-178C / ED-12C* (Tech. Rep.). Washington, DC.

Sharma, S., Flores, A., Hobbs, C., Stafford, J., & Fischmeister, S. (2019). Safety and Security Analysis of AEB for L4 Autonomous Vehicle Using STPA. *OASIcs, Volume 68, ASD 2019*, *68*, 5:1-5:13. doi: 10.4230/OASICS.ASD.2019.5

Tietz, V., Schoepf, J., Waldvogel, A., & Annighoefer, B. (2021). A Concept for a Qualifiable (Meta)-Modeling Framework Deployable in Systems and Tools of Safety-Critical and Cyber-Physical Environments. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (p. 163-169). doi: 10.1109/MODELS50736.2021.00025

Walden, D. D., et al. (Eds.). (2023). *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* (5th ed.). Hoboken, NJ: Wiley.

Wang, H., Zhong, D., Zhao, T., & Ren, F. (2019). Integrating Model Checking With SysML in Complex System Safety Analysis. *IEEE Access*, *7*, 16561–16571. doi: 10.1109/ACCESS.2019.2892745

Yang, L., Yang, Y., Wu, G., Zhao, X., Fang, S., Liao, X., … Zhang, M. (2022). A Systematic Review of Autonomous Emergency Braking System: Impact Factor, Technology, and Performance Evaluation. *Journal of Advanced Transportation*, *2022*(1), 1188089. doi: 10.1155/2022/1188089

About the authors

Muhammad Asim Minhas is a doctoral researcher at Karlsruhe Institute of Technology. His research interests are model-based systems engineering and quality analysis of cyber-physical systems. You can contact the author at muhammad.minhas@kit.edu or visit https://dsis.kastel.kit.edu/staff_muhammad_minhas.php.

Erik Burger is a postdoctoral researcher at Karlsruhe Institute of Technology. His research interests are view-based model-driven development and consistency management in cyber-physical systems modeling. You can contact the author at burger@kit.edu or visit https://dsis.kastel.kit.edu/staff_erik_burger.php.

Ralf Reussner is a computer science professor at Karlsruhe Institute of Technology (KIT). His research group works in the interplay of software architecture and predictable software quality as well as on view-based design methods for software-intensive technical systems.You can contact the author at ralf.reussner@kit.edu or visit https://dsis.kastel.kit.edu/staff_ralf_reussner.php.

Tianhai Liu is a postdoctoral researcher at Karlsruhe Institute of Technology. His research interests are formal methods and software verification in cyber-physical systems. You can contact the author at tianhai.liu@kit.edu or visit https://formal.kastel.kit.edu/liu2.