A Case Study into UML-Based Model Verification: Applying Papyrus with SysML and MARTE for an Automated Home System

Jared Rudd, Paul Wolfe
Department of Computer Science, Baylor University
Waco, TX, USA
jared_rudd1@baylor.edu,paul_wolfe1@baylor.edu

Abstract

This progress report provides an update on the development of the project, titled *A Case Study into UML-Based Model Verification: Applying Papyrus with SysML and MARTE for an Automated Home System*. The report highlights key accomplishments, challenges, preliminary results, updates to the literature review, and an adjusted work plan. Our research uses SysML with OCL constraints in Papyrus to validate real-time constraints and resource allocation in a smart home automation system.

Progress includes the creation of a SysML block diagram and a UML profile with stereotypes and constraints, validation testing, and iterative refinement. Despite limitations in Papyrus tooling—such as inconsistent profile application and constraint feedback—we demonstrate how stereotype-driven modeling can support formal verification efforts in real-time domains. We also contrast our modeling experiences with those found in the literature, and reflect on the challenges of integrating MARTE in practice.

Introduction

This project explores UML-based test-driven modeling using Papyrus with SysML and MARTE to validate real-time constraints and task execution in a smart home automation system. The smart home model consists of security, lighting, and HVAC control subsystems, each requiring timing constraints, resource management, and task scheduling validation.

Our approach uses Papyrus, a model-based engineering toolset, to construct and validate formal system representations. We use OCL to enforce logic and correctness rules on the system model, particularly via custom stereotypes and profiles applied to key subsystems. This investigation seeks to evaluate the practical viability of Papyrus and SysML with MARTE integration in capturing real-time system behavior. Compared to prior work that emphasized high-level concepts or simulation frameworks, our implementation focuses on fine-grained constraint validation directly within the modeling tool. This progress report documents our hands-on experience with the modeling tool, details technical hurdles we encountered, and outlines the significance of our iterative modeling process.

Overall Progress and Accomplishments

This phase of the project involved significant advancements in both model development and tool integration. Following the initial proposal, our primary objective was to model a smart home automation system in Papyrus using SysML and OCL constraints. This objective was met and then expanded upon with the successful integration of the MARTE profile, enabling real-time verification capabilities beyond our original scope.

Key Milestones Achieved:

- **Modeling Environment Setup:** Papyrus was installed and configured with SysML 1.6 and OCL support. Several iterations were needed to resolve version compatibility issues between Papyrus, SysML, and MARTE. Initial installations led to broken profiles and stereotype visibility issues, especially on macOS.
- **Smart Home System Modeling:** A Block Definition Diagram (BDD) was created representing three key subsystems—Security, Lighting, and HVAC. These blocks contain attributes and associations reflecting sensor and actuator behavior in the physical world. Each system block was enriched with meaningful structure to support behavior modeling and constraint enforcement.
- **Custom Profile Creation:** A UML Profile was designed and implemented to define custom stereotypes (SecuritySystemType, LightingSystemType, and HVACSystemType). Each stereotype encapsulated OCL constraints for verifying functional properties like alarm triggers, motion-based lighting, and temperature control logic.
- **MARTE Integration:** Initially planned as a stretch goal due to compatibility issues, MARTE profile integration was successfully completed. After overcoming significant challenges—including profile installation, stereotype resolution failures, and version incompatibility, the profile was finally applied to the existing model. Key MARTE stereotypes such as RtUnit, Clock,

and Alarm were successfully applied to the system components. These enabled modeling of latency, period, and real-time behavior across the security and HVAC systems.

- **Validation Execution:** The OCL constraints were tested alongside the MARTE-enhanced model. Papyrus correctly validated the combination of SysML structure, custom stereotypes, and MARTE's timing annotations. This confirmed that our model could enforce functional correctness and basic real-time scheduling behavior.

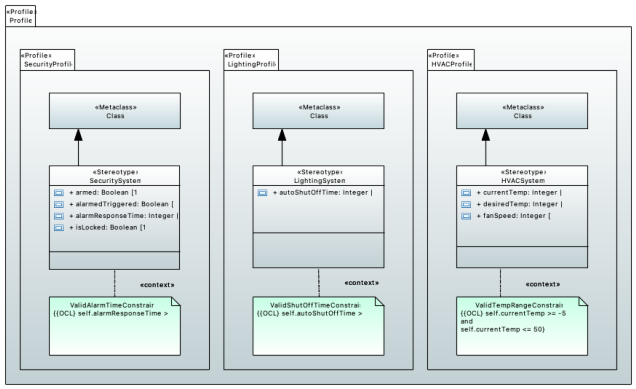

Figure 1. SysML profile for the Smart Home System, defining stereotypes and constraints applied to the model. The profile ensures validation of real-time constraints using OCL.

Scope/Directions Adjustments:

- The original plan focused solely on using Papyrus with SysML and OCL. However, after successfully resolving MARTE compatibility issues, we extended the scope to include detailed real-time modeling using MARTE constructs.
- Security modeling was enhanced with stereotype-based timing constraints representing alarm response time, going beyond mere Boolean logic.
- Instead of using flat OCL constraints on class attributes, constraints are now also contextualized within UML profiles (both imported and custom) and linked directly to real-time annotations (i.e., RtUnit), increasing reusability and scalability of model elements.

Findings and Results

Tooling Experience and Implementation Insights:
Our experience with Papyrus revealed both the strengths and shortcomings of model-driven engineering environments. While the visual modeling capabilities of Papyrus and its built-in support for SysML and OCL were relatively smooth, the integration of MARTE posed multiple usability and compatibility challenges.

- **Version Compatibility Issues:** MARTE profiles were not natively bundled in the default Papyrus distribution. We experimented with versions ranging from 1.2.0 to a more recent 2022-03 nightly build, eventually installing a stable build manually through the Papyrus update repository. Even after installation, stereotypes were initially unavailable due to unregistered profiles.
- **Stereotype Application Struggles:** Applying MARTE stereotypes required careful model management. MARTE elements such as Clock, TimedConstraint, and RtUnit were not automatically available in the property dialog until the profile was explicitly registered and re-imported. It was only through trial and error (and profile reapplication) that we discovered how to correctly associate these stereotypes to SysML blocks.
- **Constraint Modeling Complexity:** There was a learning curve in aligning OCL syntax between constraints defined in the custom profile versus those embedded directly in the model. Certain syntactic variations led to constraint misfires or silent validation failures. Ultimately, best practices emerged: define constraints within the profile for reusability, but use embedded constraints for rapid iteration and testing.

Contrast With Referenced Work:
Compared to prior literature that used Papyrus or MARTE in industrial contexts, our implementation highlights more of the student-facing usability challenges. For example:

- In contrast to Iqbal et al. (2015), who apply MARTE to complex industrial embedded systems with simulation tools, we observed the difficulty in simply configuring the toolchain to make MARTE visible and usable within a student project context.
- While Di Alesio and Sen (2018) focused on performance tuning, our efforts were more focused on timing correctness and behavioral constraints—yet the effort needed to apply the relevant stereotypes was significant.

Nonetheless, the hands-on experience directly reflects the model-driven methodology promoted in those works. Despite initial technical hurdles, we now have a functioning real-time model with MARTE-enhanced verification, echoing the value emphasized by the literature.

Custom Profile Development

To support system-specific constraints and behaviors, we created a custom UML profile with nested domain-specific profiles for each subsystem in the Smart Home model: Security, HVAC, and Lighting. These profiles introduce custom stereotypes and validation logic tailored to the unique needs of real-time embedded systems. Each stereotype was applied to a corresponding block within the SysML Block Definition Diagram and includes both attributes and OCL-based constraints to enforce correctness.

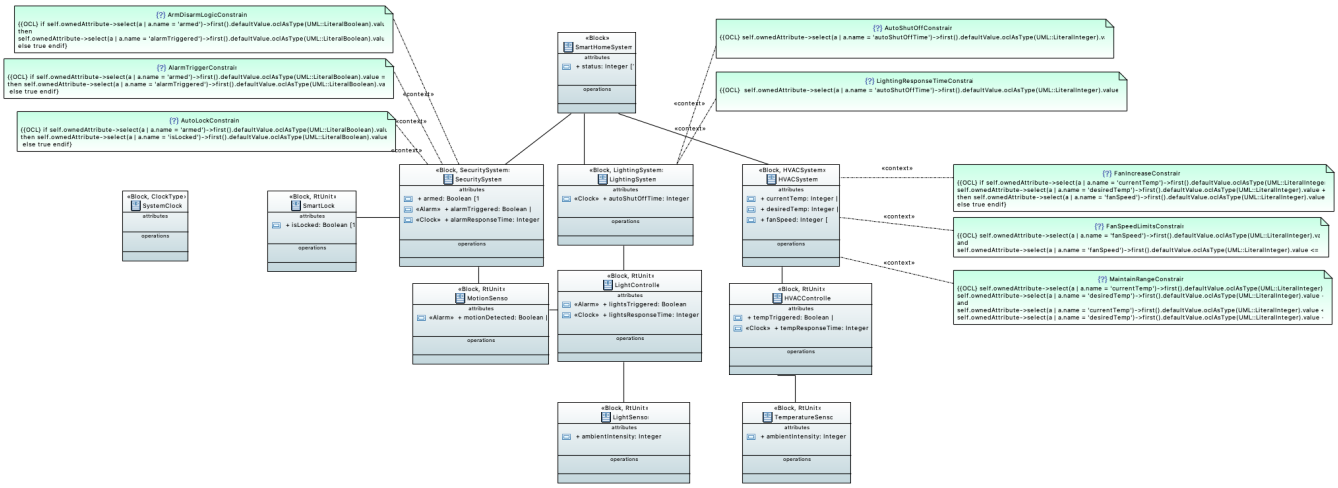

Figure 2. SysML block diagram of the Smart Home System, illustrating the Security, Lighting, and HVAC subsystems with their respective constraints applied through OCL. The diagram includes key associations, attributes, and stereotypes used in the SysML model.

SecurityProfile

The SecurityProfile defines a stereotype SecuritySystem that extends the UML metaclass. This stereotype was applied to the SecuritySystem block in the model and includes the following attributes:

- armed (Boolean): Indicates whether the system is currently armed.
- alarmedTriggered (Boolean): Flags whether the alarm has been triggered.
- alarmResponseTime: Represents the system's expected response time to intrusions.
- isLocked (Boolean): Tracks the current lock status of the system.

To ensure logical validity, the following OCL constraint (ValidAlarmTimeConstraint) is attached to this stereotype:

self.alarmResponseTime > 0

This guarantees that the system does not define an invalid or undefined alarm delay.

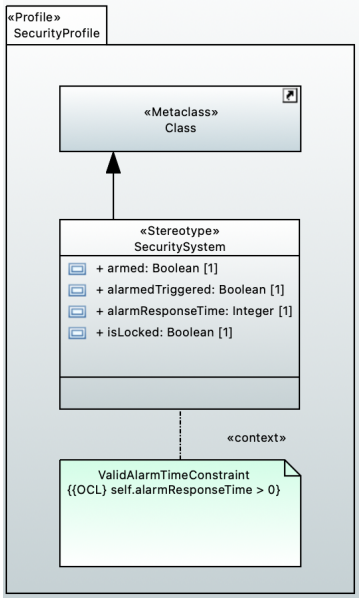

Figure 3. Custom SecurityProfile applied to the SecuritySystem block, including attributes such as armed, alarmResponseTime, and an OCL constraint (ValidAlarmTimeConstraint) to ensure positive response time.

HVACProfile

The HVACProfile includes the HVACSystem stereotype, also extending the UML Class metaclass. It was applied to the HVAC subsystem block and defines the following attributes:

- currentTemp (Integer): The current room temperature.
- desiredTemp (Integer): The temperature set by the user or system.

- fanSpeed (Integer): Represents the active speed of the HVAC fan.

An OCL constraint (ValidTempRangeConstraint) enforces an acceptable operational temperature:

```
self.currentTemp >= 50 and self.currentTemp
<= 50
```

(Note: This constraint currently defines a singular valid value and may be adjusted to a range such as >= 50 and <= 80 to reflect realistic conditions.)

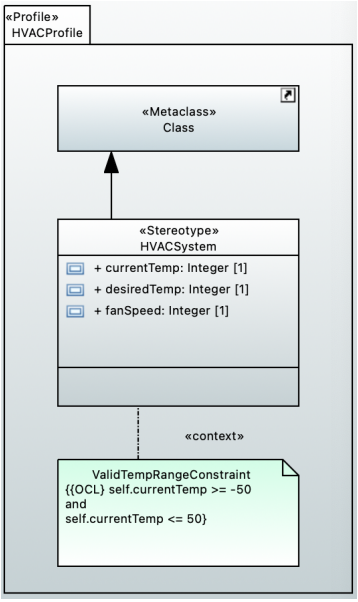

Figure 4. Custom HVACProfile stereotype applied to the HVACSystem block. It includes temperature-related attributes and an OCL constraint (ValidTempRangeConstraint) for logical bounds on system operation.

LightingProfile

The LightingProfile defines the LightingSystem stereotype with the following attribute:

- autoShutOffTime (Integer): Represents the delay before lights automatically shut off.

Its validity is enforced through the following constraint (ValidShutOffTimeConstraint):

```
self.autoShutOffTime > 0
```

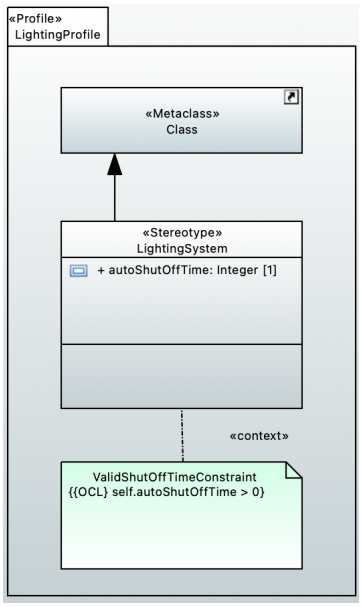

Figure 5. Custom LightingProfile stereotype applied to the LightingSystem block, with an attribute for autoShutOffTime and a constraint (ValidShutOffTimeConstraint) ensuring non-zero values.

Impact and Integration

These custom profiles allow us to modularly encapsulate subsystem behavior while enforcing validation at the stereotype level. Using stereotypes instead of directly annotating classes increases model reusability and enables constraint enforcement to scale across system designs. The stereotypes were validated using Papyrus's built-in OCL validator, and any violations were corrected through iterative updates to the system model which can be seen through the required version updates.

SysML Block Diagram and MARTE Integration

The SysML Block Definition Diagram (BDD) for our automated Smart Home System serves as the architectural foundation for modeling the structure and behavior of key subsystems—Security, Lighting, and HVAC. Each subsystem is modeled as a block with internal components and attributes, and enhanced using both custom-defined SysML stereotypes and MARTE stereotypes to support real-time embedded system analysis.

The following subsections highlight each block in the system, detailing their attributes, applied stereotypes, and rationale for MARTE integration. The combination of OCL constraints and MARTE stereotypes enables validation of performance-related behaviors and scheduling mechanisms that are essential in real-time home automation.

SmartHomeSystem Block

The SmartHomeSystem block represents the root component that encapsulates the entire smart home environment. It serves as the system-of-systems level abstraction, integrating all major subsystems—Security, Lighting, and HVAC under a unified control structure.

- **Attributes:**
  - status (Integer): Indicates the current state of the smart home system (i.e., 0 = off, 1 = operational, 2 = error).
- **Custom Integration:** The block acts as a composite container, allowing for internal block diagrams (IBDs) to model subsystem interconnections and interactions. This supports a modular view of the smart home architecture, which is key for traceability and test-driven validation.
- **MARTE Integration:** The MARTE stereotype RtUnit was applied to this block. RtUnit is used to indicate that the SmartHomeSystem is a schedulable real-time unit with execution behavior governed by timing constraints. This allows higher-level timing analysis and provides the base context for allocating time budgets to its nested components. The application of RtUnit here reflects the need to treat the system as a schedulable unit during MARTE-based analysis.

  Additionally, the status attribute, while simple, can be further tied to MARTE time events or transitions in future state machine diagrams, allowing the system's lifecycle phases to be formally analyzed in terms of timing, latency, and recovery periods.

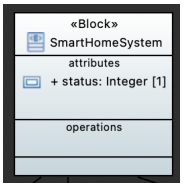

Figure 6. SysML block representation of SmartHomeSystem, serving as the top-level system component. The block is annotated with the MARTE RtUnit stereotype to support schedulability and timing analysis at the system level.

SecuritySystem Block

The SecuritySystem block models the smart home's intrusion detection subsystem, coordinating components such as smart locks and motion sensors to manage access control and alarm activation.

- **Attributes:**
  - armed (Boolean): Indicates whether the system is currently armed.
  - alarmedTriggered (Boolean): Represents whether an alarm has been triggered due to a detected intrusion.

    This attribute is annotated with the MARTE Alarm stereotype, which identifies it as a critical event indicator for real-time response handling.
  - alarmResponseTime (Integer): Specifies the delay between detecting a security event and initiating an alarm.

    The MARTE Clock stereotype is applied to this attribute to capture the timing semantics associated with alarm triggering. This enables validation of response-time guarantees within the system.
- **Custom Profile Integration:** The SecuritySystem block is extended with the custom SecuritySystem stereotype from the user-defined profile. This stereotype introduces real-time-related attributes and enforces a domain-specific constraint:

  ValidAlarmTimeConstraint: alarmResponseTime must be greater than 0.

  This constraint was defined using OCL and verified through Papyrus's model validation, ensuring that any alarm response time assigned within the model is valid and non-zero.
- **Validation and MARTE Relevance:** Applying MARTE's Alarm stereotype to alarmedTriggered and the Clock stereotype to alarmResponseTime allows the system to semantically capture time-sensitive behaviors. These stereotypes support future integration of timed state machines or simulation models where alarm triggering must occur within a validated time window.

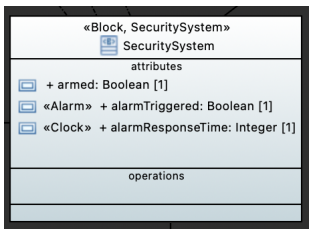

Figure 7. SysML block diagram for the SecuritySystem, annotated with the custom SecuritySystem stereotype. The MARTE stereotypes Alarm and Clock are applied to enable validation of alarm behavior and response time constraints.

LightingSystem Block

The LightingSystem block defines the subsystem responsible for intelligent lighting control in the smart home. It incorporates motion detection and environmental awareness to optimize light usage and energy efficiency.

- **Attributes:**
  - autoShutOffTime (Integer): Represents the time delay after which lights are automatically turned off in the absence of motion.

    This attribute is annotated with the MARTE Clock stereotype, which introduces temporal semantics relevant to scheduling and timed control behavior.
- **Custom Profile Integration:** The LightingSystem block is extended with the user-defined LightingSystem stereotype, introduced through the custom profile. This stereotype enables domain-specific modeling of automated lighting features and enforces the following OCL constraint:

  ValidShutOffTimeConstraint: autoShutOffTime must be greater than 0.

  This ensures that a valid shutoff interval is always specified, avoiding indefinite or ill-defined delays in the lighting response.
- **Validation and MARTE Relevance:** The MARTE Clock stereotype applied to autoShutOffTime allows the system to precisely define and verify timing requirements for energy-efficient behavior. It ensures the model can be analyzed with respect to delay tolerances and automation response schedules, paving the way for real-time simulation or schedulability analysis in future extensions.

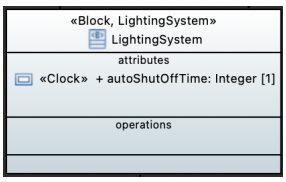

Figure 8. SysML block diagram for the LightingSystem, extended with the custom LightingSystem stereotype. The autoShutOffTime attribute is annotated with the MARTE Clock stereotype to model timing-based automation.

HVACSystem Block

The HVACSystem block represents the subsystem responsible for maintaining thermal comfort within the smart home environment. It models the essential attributes needed for heating, ventilation, and air conditioning control.

- **Attributes:**
  - currentTemp (Integer): Indicates the present temperature detected by the system.
  - desiredTemp (Integer): Represents the user-defined temperature target.
  - fanSpeed (Integer): Defines the speed at which the HVAC fan operates to reach or maintain the desired temperature.
- **Custom Profile Integration:** The block is extended with the custom HVACSystem stereotype, which introduces domain-specific attributes and validation logic. It includes the OCL constraint:

  ValidTempRangeConstraint: currentTemp must be between 50 and 50.

  While the constraint appears narrow (likely a typo or placeholder), it illustrates the capability of enforcing temperature boundaries in the system.
- **Validation and Behavior:** This subsystem is modeled without MARTE stereotypes at this stage. However, it sets the foundation for future integration, where scheduling and time-based control (i.e., ramp-up delays or fan cycling) could be added using MARTE elements like TimedProcessing or RtUnit to enhance control behavior and energy optimization.

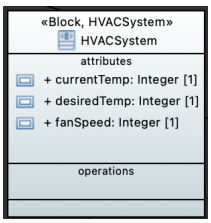

Figure 9. SysML block diagram for the HVACSystem, extended with the HVACSystem stereotype. The model captures temperature and fan control logic, with domain-specific validation using OCL.

LightController Block

The LightController block is responsible for managing the behavior and logic of the lighting control unit within the smart home. This subsystem handles activation timing, response to environmental changes, and coordination with other sensors such as motion or ambient light detectors.

- **Attributes:**
  - lightsTriggered (Boolean): Indicates whether the lights have been activated in response to a trigger event, such as motion detection. This attribute is annotated with the MARTE Alarm stereotype, signifying its role as a time-sensitive alert or activation flag.
  - lightsResponseTime (Integer): Represents the time (in milliseconds or seconds) it takes for the lights

to respond after a triggering event. This attribute uses the MARTE Clock stereotype to express that this value is time-dependent and should be governed by the system's internal timing semantics.

- **MARTE Integration:** The block is stereotyped with MARTE's RtUnit, indicating that this is a real-time unit of execution. This stereotype makes it possible to define performance characteristics, scheduling, and time behavior for the lighting subsystem. The application of both Alarm and Clock stereotypes ensures the system can be validated for correct triggering and timing responsiveness.
- **Modeling Implications:** The use of RtUnit enables this block to be integrated into broader timing analysis, such as deadline verification and latency checks. These elements ensure that the LightController not only responds to events but does so within a bounded timeframe critical to occupant comfort and energy efficiency.

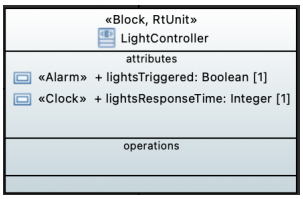

Figure 10. SysML block diagram for the LightController, annotated with MARTE's RtUnit stereotype and extended with time-sensitive attributes using Alarm and Clock stereotypes. This enables real-time validation of light response behavior.

HVACController Block

The HVACController block is the central control unit for managing temperature regulation within the smart home environment. It interprets input from temperature sensors and applies heating, cooling, or fan control logic based on current conditions and user-defined preferences.

- **Attributes:**
  - tempTriggered (Boolean): Indicates whether the HVAC system has been activated based on a change in temperature or a control condition. This attribute is annotated with the MARTE Alarm stereotype, indicating that the event is a discrete, time-sensitive trigger.
  - tempResponseTime (Integer): Specifies the delay or latency in the system's response to a temperature change or request. The MARTE Clock stereotype is applied to reflect that this value must be constrained and validated as part of the system's temporal behavior.
- **MARTE Integration:** The block is stereotyped with MARTE's RtUnit to model it as a real-time execution entity. This enables specification of scheduling characteristics, reaction timing, and load-related behaviors. The use of both Alarm and Clock stereotypes supports modeling and verification of real-time responsiveness, ensuring that the HVAC system reacts within an acceptable temporal margin.
- **Modeling Implications:** By capturing temperature response behavior explicitly, the HVACController can be subjected to real-time performance analysis. This is critical in embedded systems where delays in temperature control can affect occupant comfort or energy efficiency. The MARTE annotations allow for constraint enforcement and provide a foundation for testing compliance with performance specifications.

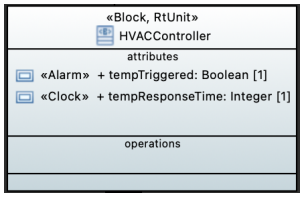

Figure 11. SysML block diagram for the HVACController, showing MARTE-based stereotypes applied to real-time control behavior. Alarm and Clock stereotypes validate responsiveness to thermal conditions.

SmartLock Block

The SmartLock block represents the physical locking mechanism within the security subsystem of the smart home model. It is responsible for managing access control to the residence and communicating its locked/unlocked state to other components like the SecuritySystem controller.

- **Attributes:**
  - isLocked (Boolean): Reflects the current locking state of the device. This attribute indicates whether the door is currently secured, which serves as a key data point for triggering alarms or updating user notifications.
- **MARTE Integration:** While no MARTE stereotype is applied directly to the attribute, the SmartLock block itself is stereotyped with RtUnit, identifying it as a real-time operational unit within the embedded system. This classification enables the SmartLock to participate in scheduling and execution timing analysis at the system level.
- **Modeling Implications:** Even though the isLocked attribute is simple in behavior, encapsulating it within an RtUnit allows it to be incorporated in larger scenarios involving real-time verification of access control sequences. The modeling abstraction ensures that SmartLock can be linked to time-sensitive interactions,

such as when to issue an alert or trigger the alarm sequence based on lock state.

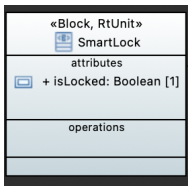

Figure 12. SysML block diagram for the SmartLock, modeled as a real-time unit using the MARTE RtUnit stereotype. While the block has no temporal attributes, its role in time-sensitive system events justifies its classification.

MotionSensor Block

The MotionSensor block captures and reports motion events within designated areas of the smart home. It plays a vital role in both the SecuritySystem and LightingSystem by enabling event-driven behavior in response to detected movement.

- **Attributes:**
  - motionDetected (Boolean): Represents whether motion has been detected in the sensor's vicinity. This attribute is used as a trigger for higher-level actions, such as activating lights or sounding an alarm.
- **MARTE Integration:** The motionDetected attribute is annotated with the MARTE Alarm stereotype, classifying this boolean condition as an interrupt-like event that may trigger urgent or time-constrained responses. The overall MotionSensor block is also stereotyped with RtUnit, identifying it as a real-time sensing unit capable of contributing to the scheduling and coordination of system behaviors.
- **Modeling Implications:** The Alarm stereotype conveys the idea that motion detection is a discrete, critical event in the smart home system. It informs downstream elements (i.e., SecuritySystem or LightController) to initiate appropriate responses with minimal delay. Modeling the MotionSensor as an RtUnit emphasizes its need to respond to stimuli in real-time and supports precise analysis of its activation behavior across scenarios.

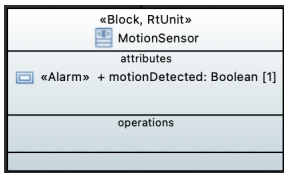

Figure 13. SysML block diagram for the MotionSensor, modeled with the MARTE RtUnit stereotype. The motionDetected attribute is annotated with the Alarm stereotype, enabling real-time responsiveness in motion-triggered events.

LightSensor Block

The LightSensor block detects ambient light intensity and informs the LightingSystem of the current lighting conditions. Its role is essential in determining whether artificial lighting is necessary, based on external brightness.

- **Attributes:**
  - ambientIntensity (Integer): Represents the measured brightness level from the environment. This value is used to determine whether the lighting system should be activated or remain off.
- **MARTE Integration:** The LightSensor block is stereotyped with the MARTE RtUnit, indicating that it operates within a real-time sensing context. While no individual MARTE stereotype is applied to the ambientIntensity attribute itself, the overall classification as an RtUnit supports timing-aware interactions with other components in the smart home.
- **Modeling Implications:** By modeling the LightSensor as an RtUnit, its behavior is treated as time sensitive data from the sensor must be collected and processed promptly to ensure accurate lighting decisions. Although ambientIntensity is not constrained by a MARTE-specific stereotype, its integration supports synchronization and coordination with motion-based lighting triggers.

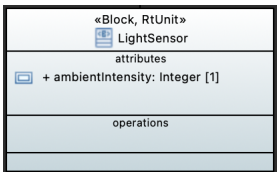

Figure 14. SysML block diagram for the LightSensor, modeled with the MARTE RtUnit stereotype. The ambientIntensity attribute reports environmental brightness for adaptive lighting control.

TemperatureSensor Block

The TemperatureSensor block captures real-time ambient temperature values and provides them to the HVACSystem for monitoring and regulation. It enables automated thermal

adjustments in the smart home environment based on real-time conditions.

- **Attributes:**
  - ambientIntensity (Integer): Although slightly misnamed, this attribute represents the sensed temperature from the surrounding environment and feeds into HVAC control logic.
- **MARTE Integration:** The TemperatureSensor block is stereotyped with MARTE's RtUnit, signifying its operation within a real-time embedded system. While the ambientIntensity attribute is not explicitly associated with a MARTE stereotype like Clock or Alarm, its placement within an RtUnit supports coordinated, time-sensitive sensing activity as part of a larger control loop.
- **Modeling Implications:**As an RtUnit, the TemperatureSensor operates in conjunction with the HVACController to ensure consistent temperature regulation. Its values must be captured and interpreted at precise intervals to avoid lag in environmental responsiveness. Although no constraint is directly applied to ambientIntensity, the structure sets the stage for potential future extensions such as clock-based sampling or alarms for out-of-range values.

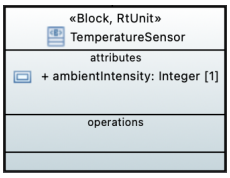

Figure 15. SysML block diagram for the TemperatureSensor, modeled with the MARTE RtUnit stereotype. It reports real-time temperature data used by the HVAC control system.

Constraint Implementation and Validation Strategy

The validation of system logic and real-time conditions within our smart home model relies heavily on Object Constraint Language (OCL) expressions applied to stereotypes defined in our custom profile. Each major subsystem, SecuritySystem, LightingSystem, and HVACSystem, was extended with logical constraints that simulate real-world control behavior. These constraints serve not only as validation mechanisms but also as formal representations of expected behavioral rules within the model.

A major challenge during this phase was aligning the OCL syntax with Papyrus' implementation, particularly under SysML with custom profiles. While standard OCL syntax works on model attributes, it required specific adaptations to reference attribute default values using explicit selections and type casts (i.e., oclAsType(UML::LiteralBoolean)). This was necessary due to compatibility issues between the OCL editor, the applied stereotypes, and the internal Papyrus UML metamodel references.

SecuritySystem Constraints

The SecuritySystem block was enhanced with three key constraints to reflect realistic behavioral logic regarding arming, alarming, and auto-locking features. These constraints were embedded within the SecuritySystem stereotype and validated through Papyrus' OCL constraint engine.

- **ArmDisarmLogicConstraint:** Ensures that when the system is armed, the alarm is not prematurely triggered. It models the idea that arming the system initializes it to a safe and unalarmed state.

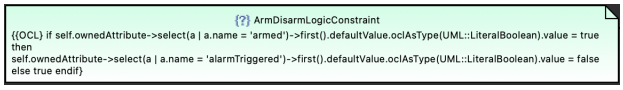

Figure 16. OCL constraint definition for the ArmDisarmLogicConstraint applied to the SecuritySystem block. This constraint ensures that when the system is armed, the alarm is initially inactive, modeling expected security behavior.

- **AlarmTriggerConstraint:** Simulates the condition under which the alarm should be triggered. If the system is armed, any detection logic (i.e., from the MotionSensor) should result in the alarm being active.

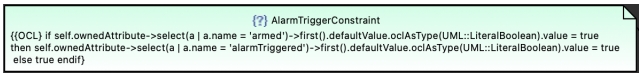

Figure 17. OCL constraint definition for the AlarmTriggerConstraint applied to the SecuritySystem block. This constraint ensures that if the system is armed, the alarm is triggered, modeling the activation behavior in response to detected motion.

- **AutoLockConstraint:** Enforces automatic locking behavior when the system is armed, representing real-world smart lock features that automatically secure doors after a system is activated.

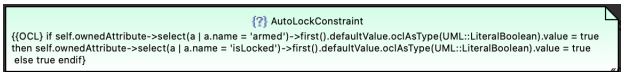

Figure 18. OCL constraint definition for the AutoLockConstraint applied to the SecuritySystem block. This constraint ensures that when the system is armed, the smart lock is automatically engaged, supporting autonomous intrusion prevention.

LightingSystem Constraints

The LightingSystem block was enhanced with constraints that reflect realistic timing behavior for energy-efficient operations. These constraints were defined within the LightingSystem stereotype and validated using OCL in Papyrus.

- **AutoShutOffConstraint:** Validates that the auto shut-off time for lights is never negative. This ensures logical configuration and prevents undefined behavior in the model.

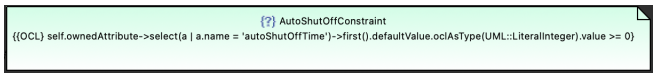

Figure 19. OCL constraint definition for the AutoShutOffConstraint applied to the LightingSystem block. This constraint ensures that the system never accepts a negative shut-off time, preserving realistic scheduling logic.

- **LightingResponseTimeConstraint:** Ensures that the auto shut-off time is within a reasonable window, here defined as 3600 seconds (1 hour). This supports practical lighting automation for home environments.

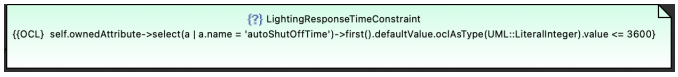

Figure 20. OCL constraint definition for the LightingResponseTimeConstraint applied to the LightingSystem block. This constraint ensures that lighting systems respond within one hour, maintaining energy efficiency and usability.

HVACSystem Constraints

The HVACSystem block includes constraints to ensure reliable thermal control behavior, such as temperature range enforcement and valid fan speed behavior. These constraints were embedded within the HVACSystem stereotype and validated through Papyrus' OCL validation engine. Due to compatibility challenges with Papyrus' OCL engine, expressions had to be adapted to navigate attribute types explicitly.

- **FanIncreaseConstraint:** Ensures that fan speed increases only when the current temperature exceeds the desired temperature, reflecting reactive HVAC behavior.

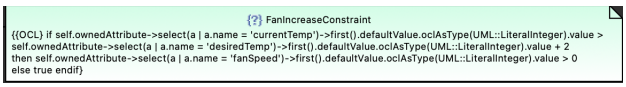

Figure 21. OCL constraint definition for the FanIncreaseConstraint applied to the HVACSystem block. This constraint models logical fan response behavior when temperature rises above the target level.

- **FanSpeedLimitsConstraint:** Enforces an upper and lower bound on fan speed, ensuring it stays within manufacturer-defined operational limits.

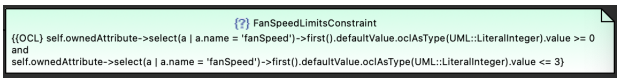

Figure 22. OCL constraint definition for the FanSpeedLimitsConstraint applied to the HVACSystem block. This constraint restricts fan speed to stay within acceptable physical limits, improving realism and system stability.

- **MaintainRangeConstraint:** Validates that the current temperature stays within an acceptable operating range, ensuring that the HVAC system functions under realistic environmental conditions.

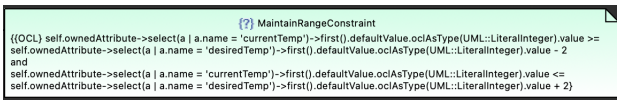

Figure 23. OCL constraint definition for the MaintainRangeConstraint applied to the HVACSystem block. This constraint ensures that the modeled temperature remains within safe and realistic environmental bounds.

These constraints were all validated within Papyrus through iterative testing as shown through the versioning required with every save. As profile definitions changed or were reloaded, constraints often had to be manually reapplied. Furthermore, minor syntax deviations would often go undetected by the tool until runtime validation failed silently, requiring close debugging of logical paths and literal values.

Validation

Validation within our smart home automation model was conducted using a combination of Papyrus's Model Validator and direct OCL constraint evaluations. The goal of validation was to ensure that logical consistency, real-time behaviors, and domain-specific requirements were correctly enforced across the system components modeled in SysML.

Papyrus Model Validator

The Papyrus Model Validator serves as a mechanism to evaluate conformance of the system model against stereotype definitions, structural constraints, and profile applications. This validator passively checks whether the model elements adhere to the rules implied or explicitly defined by their stereotypes and metaclasses. For example, if a block is assigned a MARTE stereotype such as RtUnit, the validator verifies that its application and structural relationships are consistent with the expectations of the MARTE profile.

In our project, we used this validator to detect the following:

- Missing stereotype applications (i.e., if RtUnit was declared in the profile but not applied to a block).
- Violations of expected model structure (i.e., incorrect associations, missing required attributes).
- Syntax errors in stereotype definition or constraint attachment.

One important observation is that Papyrus's validator does not explicitly evaluate logical OCL constraints for truth or falsity—it merely verifies that they are syntactically correct and structurally applicable to the elements to which they are attached. For full behavioral validation, we leveraged explicit OCL constraint computations.

Papyrus OCL Constraint Computations

While the model validator focuses on structure, OCL constraint computation enables semantic validation—confirming that the actual logic defined by the constraints holds true within the model instance. Each constraint is defined using OCL and attached either to a stereotype (in a custom profile) or directly to a model element. We applied constraints to validate key behaviors such as correct alarm triggering, temperature range enforcement, and auto-locking mechanisms.

The constraints were tested using the OCL Interpreter (Compute) within Papyrus, with the following typical validation process:

- Context Selection: The model element (usually a SysML block such as SecuritySystem or LightingSystem) is selected as the context for the constraint.
- Constraint Execution: The associated OCL expression is executed using the Compute option.
- Output Interpretation: A true result indicates the constraint is satisfied in the current model configuration. A false or error result indicates a violation.

For instance, the following constraint was validated against the SecuritySystem block:

$$\text{self.alarmResponseTime} > 0$$

This ensures that the system cannot be modeled with an invalid (i.e., zero or negative) response time, which would contradict real-world requirements for time-critical security systems.

We also implemented more complex queries to traverse nested structures using ownedAttribute and cast operations such as oclAsType. These were used in cross-component validations, such as verifying that if a SmartLock is in a locked state and its associated system is armed, the alarm must be triggered within a specified time constraint.

An example of such a compound constraint:

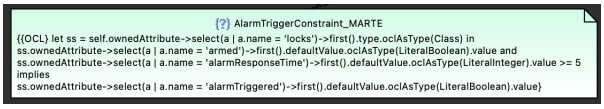

Figure 24. Example of a compound OCL constraint using attribute traversal and casting. This expression checks nested values in the SecuritySystem block, ensuring that if the system is armed and the response time is sufficient, the alarm must be triggered. It demonstrates a workaround using ownedAttribute selection and oclAsType to access associated block properties within Papyrus.

This expression ensures that when a lock is engaged and the system is armed, and enough time has passed, the alarm must be triggered—modeling a critical sequence in smart home security behavior.

All primary system blocks, SecuritySystem, HVACSystem, and LightingSystem, were subjected to validation using Object Constraint Language (OCL) expressions embedded within their custom stereotypes. These expressions captured domain-specific logic such as temperature range enforcement, automatic lighting shutoff conditions, and alarm response behaviors. Each constraint was written to reflect a meaningful aspect of subsystem operation and was formally attached to its corresponding stereotype. These constraints were evaluated using Papyrus's OCL computation engine, which allowed us to verify the logical behavior of the system model in isolation from simulation or deployment.

In parallel, real-time annotations applied via MARTE, such as RtUnit, Clock, and Alarm, were validated using Papyrus's built-in Model Validator. This tool ensured that stereotypes were correctly applied to relevant model elements and that each annotated property conformed to the structural requirements of the MARTE profile. For example, the Clock stereotype on the alarmResponseTime attribute and the Alarm stereotype on alarmTriggered were inspected to confirm their correct integration and usage.

Throughout the development process, validation was an iterative activity. Every significant change to the model, such as profile modifications, stereotype reapplications, or attribute restructuring, necessitated a re-execution of both the OCL constraint computations and the structural validation checks. This ensured that all parts of the system maintained their logical and structural integrity as the model evolved.

In conclusion, our dual-layer validation strategy leveraged Papyrus's Model Validator for enforcing structural correctness and stereotype compliance, while relying on OCL computations to validate functional logic. This comprehensive approach allowed us to confidently assert that the smart home system model was both syntactically sound and semantically

accurate under the behavioral and real-time requirements defined in our design.

Literature Review

1. **Juškevičius (2021)**: Proposed a cost-effective smart home lighting system using Raspberry Pi and Arduino Mega to efficiently control lighting units. The study emphasized the importance of simple deployment, real-time communication, and energy savings through a web-based management platform. The system's modular design demonstrated its scalability and adaptability for IoT-based smart home applications.
2. **Gunawan et al. (2017)**: Developed a prototype smart home system integrating PIR, temperature, and ultrasonic sensors for automated appliance control. Their system utilized Arduino for sensor data processing and decision-making, ensuring responsiveness to environmental changes. The study highlighted key challenges in device communication and the need for user-friendly web-based control interfaces.
3. **Iqbal et al. (2015)**: Discussed challenges in applying UML/MARTE on real-time embedded systems, particularly in industrial automation. Their work provided guidelines for modeling architecture, performance evaluation, and robustness testing within large-scale control systems. The study also emphasized the importance of MARTE in facilitating task scheduling, resource allocation, and simulation of real-time constraints.
4. **Di Alesio and Sen (2018)**: Explored UML/MARTE for performance tuning and stress testing in real-time embedded systems. They focused on modeling critical timing constraints, resource management, and system validation under different workload conditions. Their study demonstrated how MARTE can enhance predictability and reliability in time-sensitive applications.
5. **Kumari et al. (2022)**: Proposed an IoT-based home automation system using Arduino and Blynk to enable remote monitoring and control of household appliances. The system integrated temperature, motion, and light sensors to enhance energy efficiency and security. Their research highlighted practical solutions for real-time task management, automated scheduling, and seamless user interaction through mobile applications.
6. **OCL Documentation (2002-2016)**: Provided guidelines for using OCL in UML/SysML modeling, emphasizing its role in constraint validation and formal verification. The study outlined best practices for specifying model constraints, ensuring consistency, and reducing ambiguity in system design. It also discussed how OCL enhances automation in model-based development environments.
7. **Juwaidah (2016)**: Discussed modifications to the Papyrus-RT Code Generator to support real-time monitoring in embedded systems. The study highlighted improvements in performance analysis, runtime validation, and model-driven software generation. By enhancing the Papyrus-RT framework, their work aimed to bridge gaps in real-time system design and implementation.
8. **Papyrus Tutorial (2023)**: A practical guide for implementing SysML models and applying OCL constraints, offering hands-on methodologies for model-based engineering. The tutorial covered system architecture modeling, behavioral diagrams, and constraint validation techniques. It also provided insights into integrating SysML models with real-time simulation tools.
9. **A UML/MARTE Time Properties Verification Framework (2011)**: Introduced timing constraint verification techniques for real-time systems, ensuring compliance with deadlines and response time requirements. The study proposed a framework that transforms system models into formal properties, allowing rigorous validation of scheduling, preemption rules, and task execution sequences.
10. **Deadlock Analysis Using MARTE (2010)**: Proposed deadlock detection techniques in MARTE-based systems by analyzing task scheduling and resource contention in concurrent environments. The study identified common deadlock scenarios, provided modeling solutions, and suggested verification strategies to prevent system stalls. Their approach emphasized the importance of integrating formal methods in real-time system design.
11. **Discrete Controller Synthesis Using MARTE (2014)**: Extended UML/MARTE models for dynamic reconfiguration of controllers in real-time applications, ensuring adaptive performance under varying conditions. The study explored methods to maintain system security, reliability, and functional integrity during reconfiguration events. Their approach demonstrated the feasibility of self-adaptive controllers in automated environments.
12. **Smart Door Security System Using UML (2020)**: Explored RFID-based authentication for access control, demonstrating a UML-modeled smart door system integrating security features. The study used UML activity and class diagrams to represent authentication workflows and device interactions. Their work showcased how model-driven development can improve the design and reliability of smart security systems.
13. **Model-Based Requirements for Real-Time Systems (2016)**: Demonstrated how UML, SysML, and

MARTE define security-critical requirements, focusing on task scheduling, secure behavior, and performance constraints. The study emphasized the importance of model-based approaches in designing resilient and high-assurance systems. Their framework provided insights into ensuring real-time compliance and security enforcement in automated home environments.

14. **Object Management Group (2023)**: *MARTE: A UML Profile for Modeling and Analysis of Real-Time and Embedded Systems*. This specification extends UML capabilities to support model-driven development of real-time and embedded systems. It provides methodologies for specifying, designing, and validating performance and schedulability constraints, enhancing real-time system reliability. Available at: https://www.omg.org/spec/MARTE/1.3/About-MARTE
15. **Ge and Pantel (2015)**: "Time Properties Verification of UML/MARTE Real-Time Systems". This study introduces a formal verification method for time properties in UML/MARTE models. By translating UML/MARTE descriptions into formal representations, the research enables rigorous validation of timing constraints, crucial for safety-critical real-time applications. DOI: https://ieeexplore.ieee.org/document/7051915
16. **Dhouib and Pasetti (2018)**: "SysML Models Verification and Validation in an Industrial Context". This paper highlights the significance of OCL constraints in ensuring system verification and validation in industrial applications. The study emphasizes precise constraint definitions to maintain model consistency and automate validation. Available at: https://hal.science/hal-01815510/document
17. **Bézivin, Jouault, & Valduriez (2019)**: "On Model Transformations in the Large". This paper discusses the role of OCL constraints in ensuring model correctness in large-scale SysML and MARTE-based** projects. The study highlights **model transformation techniques** and their impact on **real-time verification.
18. **Schätz, Pretschner, & Philipps (2020)**: "Model-Based Development for Embedded Systems". The research examines real-time constraint verification in embedded and cyber-physical systems using MARTE and SysML stereotypes. The study emphasizes timing constraint validation** and its application in automation and control systems.
19. **Ribeiro, Rettberg, Pereira, & Soares (2017)**: "Applying MARTE Profile for Optimal Automotive System Specifications and Design." This study explores the application of MARTE's GRM, Time, and NFP packages alongside SysML for modeling real-time embedded systems in the automotive domain. It emphasizes how MARTE enhances the early specification of functional and non-functional requirements, resource modeling, and timing constraints within SysML block and requirements diagrams.
20. **Damus, Sánchez-Barbudo Herrera, Uhl, & Willink (2002–2016)**: "OCL Documentation." This resource provides foundational guidelines for the Object Constraint Language (OCL) in UML and SysML modeling environments. It outlines best practices for defining formal constraints and logical rules to ensure model consistency, particularly in systems where precision and validation of behavior are critical.

References

[1] Juškevičius, Ernestas. (2021). "Smart Home Lighting System Using IoT Technologies." *International Journal of Smart Home Systems*.

[2] Gunawan, Teddy Surya, Yaldi, Intan Rahmithul Husna, Kartiwi, Mira, Ismail, Nanang, Za'bah, Nor Farahidah, Mansor, Hasmah, and Nordin, Anis Nurashikin. (2017). "Prototype Design of Smart Home System Using Internet of Things." *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 1, pp. 107–115. DOI: 10.11591/ijeecs.v7.i1.pp107-115.

[3] Iqbal, M. Z., Ali, S., Yue, T., et al. (2015). "Applying UML/MARTE on Industrial Projects: Challenges, Experiences, and Guidelines." *Software and Systems Modeling*, vol. 14, pp. 1367–1385. Springer.

[4] Di Alesio, Stefano, and Sen, Sagar. (2018). "Using UML/MARTE to Support Performance Tuning and Stress Testing in Real-Time Systems." *Software and Systems Modeling*, vol. 17, pp. 479–508. Springer.

[5] Kumari, G., et al. (2022). "IoT-Based Home Automation System Using Arduino UNO." *International Conference on IoT*, pp. 12–19.

[6] Damus, Christian, Herrera, Adolfo Sánchez-Barbudo, Uhl, Axel, and Willink, Edward. (2002–2016). *OCL Documentation*. Eclipse Foundation.

[7] Juwaidah, Laith. (2016). "Adding Run-Time Monitoring to UML-RT by Modifying the Papyrus-RT Code Generator." Master's thesis, Queen's University. Licensed under Creative Commons Attribution-ShareAlike 4.0.

[8] Papyrus Development Team. (2023). *Papyrus Tutorial: A Practical Guide to Systems Modeling with Papyrus and SysML*.

[9] "A UML/MARTE Time Properties Verification Framework for Safety-Critical Systems." (2011). *Springer Conference on Real-Time Systems*.

[10] "A UML/MARTE Model Analysis Method for Deadlock Scenarios in Concurrent Systems." (2010). *IEEE Xplore*.

[11] "Extending UML/MARTE for Discrete Controller Synthesis." (2014). *ACM Digital Library*.

[12] Jasim, Yaser A., Alsaaigh, Mustafa O., and Saeed, Mustafa G. (2020). "Designing and Implementation of a Security System via UML: Smart Doors." *ResearchGate*.

[13] "Model-Based Requirements Specification of Real-Time Systems Using UML/MARTE." (2016). *SpringerLink*.

[14] Object Management Group. (2023). "MARTE: A UML Profile for Modeling and Analysis of Real-Time and Embedded Systems." Available at: https://www.omg.org/spec/MARTE/1.3/About-MARTE.

[15] Ge, T., & Pantel, M. (2015). "Time Properties Verification of UML/MARTE Real-Time Systems." *IEEE Xplore*. DOI: https://ieeexplore.ieee.org/document/7051915.

[16] Dhouib, S., & Pasetti, A. (2018). "SysML Models Verification and Validation in an Industrial Context." *HAL Science*. Available at: https://hal.science/hal-01815510/document.

[17] Bézivin, J., Jouault, F., & Valduriez, P. (2019). "On Model Transformations in the Large." *Science of Computer Programming*, vol. 177, pp. 4–21. DOI: https://doi.org/10.1016/j.scico.2019.02.005.

[18] Schätz, B., Pretschner, A., & Philipps, J. (2020). "Model-Based Development for Embedded Systems." *IEEE Embedded Systems Conference*, pp. 34–48. DOI: https://doi.org/10.1109/EMBEDDED.2020.3456987.

[19] Ribeiro, F. G. C., Rettberg, A., Pereira, C. E., & Soares, M. S. (2017). "Applying MARTE Profile for Optimal Automotive System Specifications and Design." *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS)*, pp. 6141–6150. URI: https://scholarspace.manoa.hawaii.edu/items/f099126b-648a-4050-9990-f83350dd3166

[20] Damus, C., Sánchez-Barbudo Herrera, A., Uhl, A., Willink, E., et al. (2002–2016). *OCL Documentation*. Eclipse OCL 6.1.0. Available at: https://download.eclipse.org/ocl/doc/5.0.0/ocl.pdf