Visual Modeling and Simulation of AUTOSAR Application Layer Models Using Modelica

1st Peihao Yang — *Faculty of Computing*, *Harbin Institute of Technology*, Harbin, China, yangpeihao@stu.hit.edu.cn

2nd Tiantian Wang — *Faculty of Computing*, *Harbin Institute of Technology*, Harbin, China, wangtiantian@hit.edu.cn

3rd Ming Yang — *Faculty of Computing*, *Harbin Institute of Technology*, Harbin, China, 23S136282@stu.hit.edu.cn

4th Xiaohong Su — *Faculty of Computing*, *Harbin Institute of Technology*, Harbin, China, sxh@hit.edu.cn

***Abstract*—As automotive electronic architectures grow increasingly complex and software development costs escalate, AUTOSAR plays a critical role in standardizing and enhancing the reusability of automotive controllers. However, existing AUTOSAR application layer modeling tools, such as the Simulink AUTOSAR Blockset, primarily adopt causal modeling paradigms, which constrain flexibility in capturing intricate system interactions. Additionally, their proprietary nature limits model accessibility, hindering cross-platform collaboration and multi-domain integration. Modelica, an object-oriented, equation-based modeling language, is particularly well-suited for multi-domain simulations due to its acausal modeling capabilities and strong support for component reuse.**

This paper proposes a Modelica-based visual modeling approach for AUTOSAR application layer models. Specifically, it establishes encapsulation rule for representing AUTOSAR constructs in Modelica and develops an open-source AUTOSAR model library, facilitating industry collaboration and accelerating rapid prototyping. A formal mathematical representation of AUTOSAR models is introduced to enhance both expressiveness and verifiability. Furthermore, a structured visual modeling methodology is presented to lower the development barrier for AUTOSAR application layer modeling. Comparative analysis with Simulink's AUTOSAR Blockset demonstrates that the proposed approach successfully integrates Modelica's multi-domain modeling capabilities into the AUTOSAR workflow while ensuring simulation consistency with Simulink.

To the best of our knowledge, this work represents the first integration of AUTOSAR within Modelica's multi-domain simulation framework. Compared to Simulink, Modelica's acausal modeling paradigm enables more flexible system representations, its open ecosystem supports cross-platform collaboration, and its multi-domain integration enhances interoperability. Beyond the automotive domain, the proposed approach can also be applied to controller design in other industries, further demonstrating its potential for cross-disciplinary adoption.

***Index Terms*—Embedded systems, Modeling methodologies, Design languages, Simulation languages, Model-driven software engineering**

I. Introduction

As automotive technology advances rapidly, vehicle electronic and electrical architectures are increasingly shifting toward intelligence, connectivity, and electrification [1]. Automotive electronics have become a crucial enabler, expanding vehicle functionalities and enhancing performance. As this architecture evolves, its complexity grows correspondingly. Software, now the "brain" of the vehicle, is becoming more deeply integrated into automotive systems. Consequently, software costs are rising steadily, posing a significant challenge to the industry [2, 3]. To address this challenge, the AUTOSAR (Automotive Open System Architecture) standard was introduced [4]. Developed through collaboration among global automakers, suppliers, and research institutions, AUTOSAR aims to standardize automotive electronic systems by decoupling software from hardware in ECUs. This reduces software development costs while enhancing both efficiency and reliability [5]. At the heart of the AUTOSAR application layer are Software Components (SWCs), which encapsulate functional behaviors and communicate via standardized interfaces. These SWCs promote modularity, reusability, and cross-platform integration, making them essential components of modern automotive software development.

The AUTOSAR standard is essential to the automotive industry, yet its complex framework presents challenges in design, configuration, and simulation [6]. These issues are particularly apparent in multi-domain collaborative modeling, where existing tools, such as the Simulink AUTOSAR Blockset, are limited by their reliance on causal modeling and proprietary ecosystems, which hinder flexibility in capturing complex system behaviors. Furthermore, these tools natively do not support multi-domain modeling. Their closed-source nature also restricts model accessibility, hindering cross-platform collaboration. Overcoming these limitations requires adopting acausal modeling and creating an open model exchange ecosystem. Modelica, an object-oriented, equation-based language, is well-suited for multi-domain simulations, offering acausal modeling and strong component reuse support.

This paper introduces a visual modeling and simulation approach for AUTOSAR application layer models using Modelica, aimed at simplifying the complexity of traditional AUTOSAR model development. By evaluating practical cases, the proposed method leverages Modelica's multi-domain simulation capabilities to visually represent AUTOSAR models, ensuring consistency with simulation results obtained from the Simulink AUTOSAR Blockset. This work addresses the gap in Modelica's support for the AUTOSAR standard and is the first to integrate AUTOSAR into a multi-domain simulation framework, enabling new opportunities for collaborative multi-domain modeling.

The contributions of our work are summarized below.

- A proposal for a Modelica-based encapsulation rule for the AUTOSAR model library and the implementation of a publicly accessible model library [1], which facilitates industry collaboration and accelerates rapid prototyping.
- A formal mathematical representation of AUTOSAR models, along with structured visual modeling rules for AUTOSAR application layer modeling, enhancing both expressiveness and verifiability while ensuring correctness in model construction.
- A comparative case study that evaluates the feasibility of using Modelica for AUTOSAR modeling, demonstrating simulation consistency with Simulink's AUTOSAR Blockset and providing insights for Modelica-based controller modeling in other industries.

The remainder of this paper is organized as follows: Section II provides a review of related work, including an overview of the AUTOSAR standard, existing tools for AUTOSAR modeling and simulation, and the application of Modelica in modeling complex systems. Section III introduces the proposed Modelica-based AUTOSAR modeling framework, detailing the encapsulation of the AUTOSAR model library, the formal representation of AUTOSAR models, and the visual modeling rules for the development of application-layer AUTOSAR models. Section IV describes the implementation of the framework, including the development environment, code composition of the AUTOSAR model library, and the visual modeling process. Section V presents a case study and evaluation. First, a comparative analysis is performed between the proposed Modelica-based approach and Simulink's AUTOSAR Blockset, focusing on modeling efficiency and simulation consistency. Then, the model used in the comparison is applied in a multi-domain simulation scenario to demonstrate its effectiveness in controlling the automotive headlamp flashing function. Section VI discusses the strengths and limitations of the proposed approach and explores potential directions for future research. Section VII concludes the paper by summarizing the key contributions and findings.

II. Related Works

A. AUTOSAR Standard

AUTOSAR is a global standard designed to address the growing complexity of automotive electronic systems by defining a unified software architecture [7]. The standard is developed through collaboration among global automotive manufacturers, component suppliers, and research institutions, establishing an open software architecture for ECUs. It encompasses three main components: a layered architecture, methodology, and application interfaces. By implementing this standard, the decoupling of hardware and software is achieved, supporting modular design, effectively reducing software development costs, and ensuring software quality. It aims to realize the goals of "standardization, decentralized implementation, and centralized configuration."

[1]https://github.com/YangPeihao1203/Modelica4AUTOSAR

The AUTOSAR layered architecture [8] structures automotive software into four layers: the Application Software Layer (ASW), Runtime Environment (RTE), Basic Software Layer (BSW), and Microcontroller. This modular design enhances scalability and interoperability by ensuring independence between layers through well-defined interfaces. At the core of the ASW are Software Components (SWCs), which encapsulate specific functionalities and interact with other components or external systems via standardized interfaces. SWCs are hardware-independent, allowing flexible deployment across different ECUs, while the RTE abstracts hardware dependencies to facilitate seamless integration. To standardize automotive software development, the AUTOSAR methodology [9] defines best practices for project management, requirements engineering, system design, software development, verification, and testing. Additionally, AUTOSAR specifies standard application interfaces [10], including AUTOSAR interfaces, standard AUTOSAR interfaces, and standardized interfaces, to ensure seamless integration of software components from different suppliers. This paper focuses on visual modeling of the ASW, particularly in representing SWCs, while also incorporating a Modelica-based library that integrates lower layers, providing a comprehensive and flexible modeling framework.

B. Tools for AUTOSAR

Various tools have been developed to support AUTOSAR-compliant systems, categorized into commercial and open-source solutions. Commercial tools are prevalent in the industry due to their comprehensive feature sets. Vector's DaVinci Developer and Configurator provide complete toolchains for configuring Basic Software and Application Software, including RTE generation. Vector's CANoe and CANalyzer are essential for communication analysis and debugging in AUTOSAR systems. ETAS's ISOLAR tool family offers application modeling, software configuration, and virtual ECU simulation, particularly through ISOLAR-A and ISOLAR-EVE. Mentor Graphics' Capital VSTAR supports end-to-end development from application software to basic software. Electrobit's EB tresos Studio is commonly used for configuring AUTOSAR BSW and meeting ISO 26262 [11] requirements. dSPACE's SystemDesk and VEOS provide powerful modeling, code generation, and simulation platforms. MathWorks' Simulink AUTOSAR Blockset supports both Classic and Adaptive Platforms, enhancing model-based design in the Simulink environment. Additionally, IBM's Rational Rhapsody supports AUTOSAR modeling [12] through UML/SysML-based design and automatic code generation, while Dassault Systèmes' SCADE Suite and SCADE Architect provide comprehensive tools for model-based design, verification, and integration within the AUTOSAR framework.

Open-source tools, although less widely adopted, play a significant role in research and preliminary development. ARTop [13], an Eclipse-based framework, facilitates custom AUTOSAR tool development. Arctic Core provides foundational software modules and runtime environment configuration for embedded systems. openAUTOSAR, derived from

Arctic Core, offers improved module support and closer adherence to AUTOSAR standards.

Existing tools primarily focus on single-domain simulations and lack support for efficient multi-domain modeling and simulation. This highlights the need for a Modelica-based approach. By enabling visual modeling and simulation of AUTOSAR application layer models, Modelica offers greater flexibility and efficiency, improving simulation accuracy and model-based design for complex automotive systems.

C. Modelica

The Modelica specification, maintained by the Modelica Association, consists of the Modelica Language Specification and the Modelica Standard Library, both freely available under the Modelica License Agreement [14]. As of version 3.6, Modelica has been widely adopted for system modeling and simulation across various engineering domains [15]. As an open, multi-domain modeling language, it enables the description of dynamic behaviors in multidisciplinary systems, including mechanical [16], electrical [17], hydraulic [18], and thermodynamic [19] domains. Unlike traditional causal modeling, Modelica employs an equation-based, non-causal approach that supports hybrid continuous-discrete modeling, allowing for more intuitive and flexible representations of physical systems. By leveraging graph theory and general equations, it facilitates the development of modular, reusable, and scalable models, enhancing the efficiency of multi-domain system simulation and analysis.

In recent years, Modelica has become a key standard in unified modeling and a driving force in industrial digital transformation. Widely adopted by industrial sectors and software vendors, it has fostered a collaborative innovation ecosystem [20]. Its modeling capabilities align closely with the needs of Cyber-Physical Systems (CPS) [21, 22], which integrate computational and physical processes. By enabling the simulation of both domains, Modelica supports CPS design and optimization, enhancing system performance, safety, and reliability while advancing next-generation intelligent systems.

III. Proposed Framework

The proposed framework for visually modeling and simulating AUTOSAR application layer models with Modelica, as shown in Fig. 1, integrates Modelica's multi-domain modeling capabilities into the AUTOSAR modeling process. It begins by analyzing the AUTOSAR standard and applying the encapsulation rules from Section III-A to construct a Modelica-based AUTOSAR library. This process covers library structuring, data type mapping, static structure encapsulation, visual representation, and dynamic behavior modeling, ensuring compliance with AUTOSAR specifications and accurate simulation behavior. Section III-B then introduces the formal representation of AUTOSAR model elements within Modelica. Based on the visual modeling rules in Section III-C, AUTOSAR application layer models are constructed, incorporating dynamic behavior design and parameter configuration. Iterative optimization is performed based on simulation results, ultimately yielding a complete visual representation of AUTOSAR application layer models in Modelica.

A. Encapsulation of the AUTOSAR Model Library.

Encapsulating the AUTOSAR model library in Modelica aims to represent core AUTOSAR concepts while ensuring model reuse and enabling users to construct visual models that comply with AUTOSAR standards. This paper focuses on encapsulating the AUTOSAR library for the Classic platform [23], though the approach is also applicable to the Adaptive platform. The encapsulation rules are detailed in Table I.

The model library structure is crucial to the overall system design, as it organizes the AUTOSAR model library and defines the package paths for various model elements. The design must adhere to principles of data abstraction, interface separation, and modularization, while complying with AUTOSAR standard specifications to ensure consistency. The proposed encapsulation structure of the AUTOSAR model library, as shown in Fig. 2 is as follows:

- MSR Package: Contains class definitions for elements in the Manufacturer Supplier Relationship (MSR) [24] standard, providing auxiliary modeling information such as document font format and version metadata.
- AUTOSAR Templates Package: Includes core modeling elements, such as GenericStructure, SWComponentTemplate, and CommonStructure, essential for AUTOSAR system modeling and indirectly involved in the graphical modeling process.
- Utilities Package: A tool library that supports AUTOSAR model development, addressing issues like attribute multiplicity not covered by the standard.
- VisibleElements Package: Contains class definitions for all visual elements, directly involved in the graphical visualization process in Modelica, ensuring compliance with AUTOSAR standards through inheritance from the AUTOSAR Templates package.
- Examples Package: Provides a set of visualization examples to help users quickly understand the AUTOSAR model library and its application.

In the AUTOSAR standard, the concept of data types is explicitly defined. Therefore, when encapsulating the model library, it is essential to design the mapping between AUTOSAR data types and Modelica data types. For basic data types, Modelica supports built-in types such as `Real`, `Integer`, `Boolean`, and `String`, or new basic data types can be defined using the `type` keyword. For structured data types, Modelica defines them using the `record` keyword, and for enumeration data types, Modelica provides support through the `enumeration` keyword. This design enables more flexible data type definitions and handling, offering greater flexibility in the modeling and simulation of AUTOSAR models.

In encapsulating static structures, the AUTOSAR standard's object-oriented [25] approach aligns well with Modelica's object-oriented nature, enabling efficient mapping of AUTOSAR-defined structures. Consistency between type names, class relationships, attribute names, and attribute

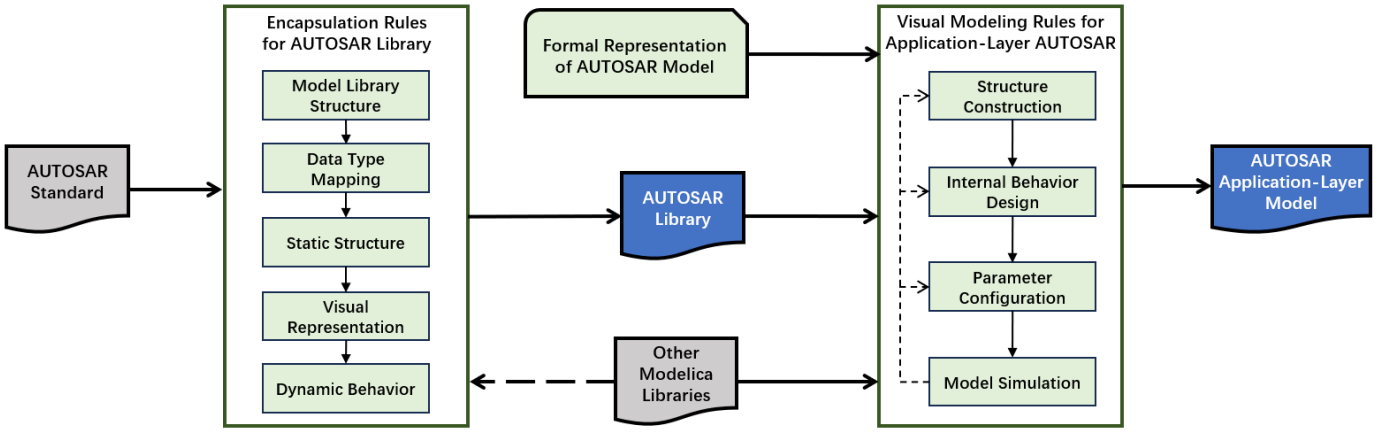


Fig. 1: Proposed Framework for Visual Modeling and Simulation of AUTOSAR Application Layer Models Using Modelica.

TABLE I: Encapsulation Rules for AUTOSAR Library

Encapsulation Item	AUTOSAR Standard	Encapsulation Rule
Model Library Structure	Organizes conceptual elements into distinct structures	Packages include: MSR, AUTOSAR Templates, Utilities, VisibleElements, and Examples, each containing relevant sub-packages.
Data Type Mapping	Primitive data types	Uses `built-in data types` or custom definitions through the `type` mechanism.
	Structured data types	Supported using the `record` mechanism.
	Enumeration data types	Supported via the `enumeration` mechanism.
Static Structure	Class names	Classes are named to match AUTOSAR definitions; Abstract classes are represented using the `partial` mechanism.
	Class relationships	Supports single inheritance with transitivity.
	Attribute names	Class members are named to match AUTOSAR definitions.
	Attribute types	Class members are typed according to AUTOSAR definitions.
	Attribute multiplicity	Fixed multiplicity is represented by explicit cardinality; Variable multiplicity is represented by an unbounded array.
	Attribute aggregation	Direct declaration via class member variables; Reference types are represented using the `inner/outer` mechanism.
Visualization Representation	Visualization of common modeling elements	Inherits constraints from static structure classes and is enhanced with graphical annotations.
Dynamic Behavior	Interoperability and simulation of AUTOSAR elements	Non-causal modeling for inter-component connections; Causal modeling for detailed component behavior; Data flow is controlled by causal modeling assignment order.

types must be maintained. Modelica defines abstract classes using the `partial` keyword and employs single inheritance with transitivity to resolve multiple inheritance issues in the AUTOSAR standard. For attribute multiplicity, Modelica uses fixed quantities when multiplicity is explicitly defined by AUTOSAR. When unspecified, it employs arrays with undetermined dimensions. The `parameter` mechanism parameterizes array sizes (defaulting to 0) and, combined with the annotation mechanism (e.g., `Dialog(connectorSizing=true)`), supports dynamic dimension adjustment during modeling. As Modelica lacks explicit aggregation mechanisms, member variables are directly declared within classes. The `inner/outer` mechanism ensures proper reference resolution within nested classes, maintaining model integrity and consistency.

To enable intuitive construction of AUTOSAR application layer models through drag-and-drop functionality in Modelica, visual modeling elements are encapsulated by inheriting definitions and constraints from static structure classes. Following AUTOSAR's visualization specifications, elements such as SWC, Runnable Entity(Runnable), Inter-Runnable Variables(IRV), RTE Events, Required Port(RPort), and Privided Port(PPort) are graphically represented. Descriptions of these elements are provided in Table II.

To ensure accurate simulation of the encapsulated AUTOSAR library, dynamic behavior encapsulation is required. Non-causal connectors (`equation`) are used to model inter-component relationships, while causal connectors (`algorithm`) describe the internal dynamics of individual components. For instance, `equation` links the Runnable to the IRV, while `algorithm` defines the internal behavior of the Runnable. In ECUs, signal flow is unidirectional, from input to output ports. Therefore, connections in RPort and PPort must use causal `algorithm` connectors, not non-causal `equation`, to avoid inconsistencies between the solved equation order and the Modelica connection sequence.

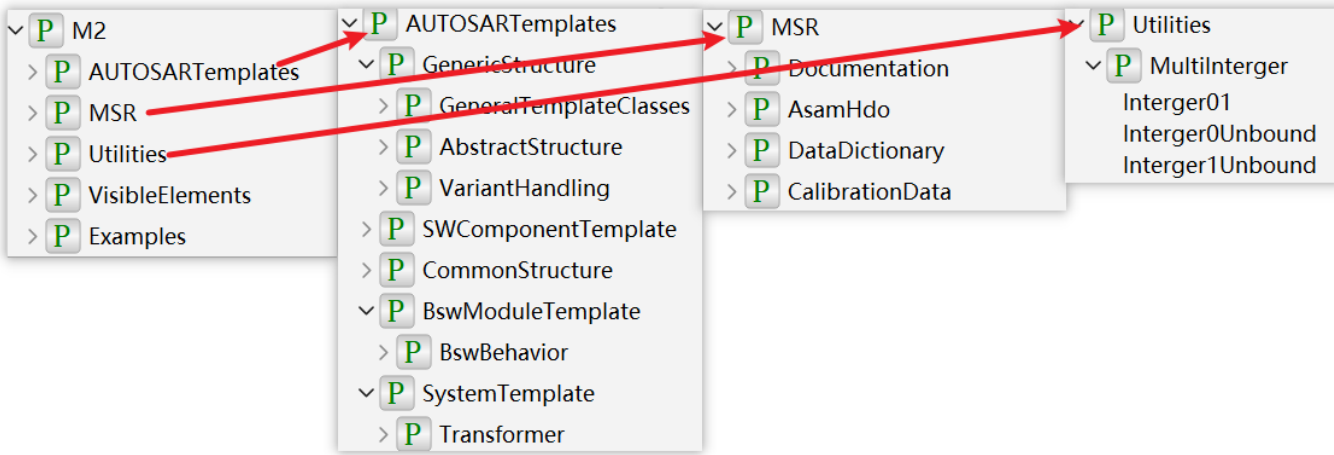


Fig. 2: The AUTOSAR Model Library Structure Based on Modelica Designed and Encapsulated in This Paper

TABLE II: Description of AUTOSAR Elements for Application Layer

Element	Description
SWC	A basic unit providing functionalities through interfaces.
Runnable	The smallest schedulable unit of code, triggered by events or periodic timing.
IRV	Variables used to transfer data between Runnables, enabling data sharing and system module decoupling.
RTE Events	Events that trigger the execution of Runnables via the RTE.
RPort	A receiving communication port that consumes data or service requests.
PPort	A providing communication port that supplies data or services to other components.

B. Formal Representation of AUTOSAR Model

To address the research needs of AUTOSAR application layer modeling, this paper formalizes a subset of AUTOSAR elements in Modelica, adhering to the principles of minimization and descriptiveness. The model consists of the previously encapsulated AUTOSAR library, supplemented by elements from other Modelica libraries, including those from the official Modelica standard library. Additionally, components related to controllers, such as signal processing, logical elements, mathematical operations, and control systems (e.g., PID controllers), are integrated to enhance the model. The formalized definitions are as follows:

Definition 1. Alphabet and Symbol Definition

- $AR_MODELICA = \{AR, V, C, CON, SUB, PARAM, INPUT, OUTPUT, \prec\}$, where uppercase letters denote sets and lowercase letters denote elements of those sets.
- AR represents the set of AUTOSAR elements modeled in Modelica, defined as $AR = \{V, C\}$.
- $V \subseteq AR$ is the set of visual modeling elements from the *VisibleElements* package in the AUTOSAR library, including *RPORT*, *PPORT*, *Runnable(RE)*, and *IRV*, which together form the set $V = \{RPORT, PPORT, RE, IRV\}$.
- $\prec$ denotes a partial order relation. If $e_1 \prec e_2$, this indicates that element e_1 is modeled within element e_2.
- $C \subseteq AR$ represents the set of Modelica elements for controller modeling, categorized into atomic and composite types. $C=\{INPUT,OUTPUT,...\}$, $C \prec RE$.
- $INPUT \subseteq C$, representing the set of input interfaces derived from `Modelica.Blocks.Interfaces`.
- $OUTPUT \subseteq C$, representing the set of output interfaces derived from `Modelica.Blocks.Interfaces`.
- $PARAM$ is the param set, where $param(ar)[key] = value$, enabling independent configuration of parameters within the Modelica model.

Definition 2. Formal Representation of AUTOSAR Model

- $c \in C$, $re \in RE$, then $c \prec re$, represents c is modeled within re.
- $CON \subseteq AR \times AR$ represents the set of connections between model elements, where $con(AR_s, AR_t)$ or (AR_s,AR_t) indicates a connection from AR_s to AR_t.
- $SUB \subseteq C$, where $SUB = C \times CON \times C$, is the set of subsystems, each consisting of nested Modelica elements and their interconnections, similar to a controller model.

C. Visual Modeling Rules for Application-Layer AUTOSAR

To facilitate description, the symbol "$\rightarrow$" is used to represent the refinement relationship in the AUTOSAR model expressed in Modelica. The application of Rules 1, 2, and 3 focuses on the structural construction of the AUTOSAR application-layer model, specifically referring to SWC. In this process, visual modeling elements from the AUTOSAR model library are first employed to construct the framework, performing basic refinement as described in Rule 1. Then, connections between AUTOSAR elements are incrementally established according to Rule 3. Throughout this process, valid connections are allowed and enable communication, while connections that do not meet the defined criteria are rejected. The specific set for valid connections are detailed in Rule 2.

Rule 1. Refinement Rule for AUTOSAR SWC

$$\frac{V' = V \cup \{v_0, v_1, \ldots, v_k\},\ \forall n \neq m,\ v_n, v_m \in V',\ N(v_n) \neq N(v_m)}{V \rightarrow V',\ v \rightarrow \{v_0, v_1, \ldots, v_k\}} \quad (1)$$

Rule 1 describes the process of extending the original set of AUTOSAR SWC elements V by adding multiple AUTOSAR visual elements $v_0, v_1, \ldots, v_k$. Through this refinement operation, the original set V is expanded to a new set V'. Here, $N(ar)$ denotes the name of element ar. This rule ensures that the refined model set does not contain duplicate elements, thereby maintaining uniqueness of element names.

Rule 2. Valid Connection Set for AUTOSAR SWC

$$\mathcal{A} = \begin{Bmatrix} \text{con}(RPORT, RE_{\text{in}}), \\ \text{con}(RE_{\text{out}}, PPORT), \\ \text{con}(IRV_{\text{out}}, RE_{\text{in}}), \\ \text{con}(RE_{\text{out}}, IRV_{\text{in}}) \end{Bmatrix} \tag{2}$$

This rule defines the valid connection set between elements in the AUTOSAR SWC framework. Any connection not included in this set is considered an invalid connection.

Rule 3. Refinement of Connections in AUTOSAR SWC

$$\frac{CON' = CON \cup \{con'\},\ con' \in \mathcal{A}}{CON \to CON',\ con' = (v_s, v_t)} \tag{3}$$

Rule 3 defines the addition of new connections $con' = (v_s, v_t)$ to the existing connection set CON, where v_s represents the source port and v_t represents the target port. CON' represents the refined set of connections. It is important to note that not all elements in the AUTOSAR SWC framework are capable of establishing connections, as connections must satisfy the valid connection set defined in Rule 2.

Next, the internal behavior of AUTOSAR application-layer components will be designed, with a focus on expanding the design of the Runnable. To achieve this, four rules are defined in this paper. Rule 4 ensures consistency between the ports in the internal behavior design of the Runnable and the ports of the Runnable in the SWC framework. Rule 5 outlines the refinement rules for adding simple elements in the internal behavior design of the Runnable. Rule 6 specifies the refinement rules for establishing connections between controller elements within the Runnable. Finally, Rule 7 describes the decomposition refinement of the Runnable, supporting the design and implementation of subsystems.

Rule 4: Port Constraints in Runnable Entity

$$\begin{gathered} \forall re \in RE\ (\exists re_{\text{in}} \Rightarrow input \in INPUT \wedge input \prec re) \\ \text{and} \quad \forall re^1_{\text{in}}, re^2_{\text{in}} \in RE,\ \left(re^1_{\text{in}} \neq re^2_{\text{in}} \Rightarrow input_1 \neq input_2\right) \end{gathered} \tag{4}$$

$$\begin{gathered} \forall re \in RE\ (\exists re_{\text{out}} \Rightarrow output \in OUTPUT \wedge output \prec re) \\ \text{and} \quad \forall re^1_{\text{out}}, re^2_{\text{out}} \in RE,\ \left(re^1_{\text{out}} \neq re^2_{\text{out}} \Rightarrow output_1 \neq output_2\right) \end{gathered} \tag{5}$$

Rule 4 specifies that for any *Runnable* in the AUTOSAR SWC, if it has an input port (re_{in}), there must be a corresponding "*input*" in the Modelica-represented Runnable model. Each re_{in} must be mapped to a unique "*input*", and similarly, for output ports, each re_{out} must be associated with a distinct "*output*". This relationship is expressed by the logical implication($\Rightarrow$).

Rule 5. Refinement Rule for Adding Controller Elements within a Runnable Entity

$$\frac{C' = C \cup \{c_0, c_1, \ldots, c_k\},\ \forall a \neq b,\ c_a, c_b \in C',\ N(c_a) \neq N(c_b)}{C \to C',\ c \to \{c_0, c_1, \ldots, c_k\}} \tag{6}$$

Rule 5 defines the addition of multiple controller elements, represented by Modelica within a Runnable, with the requirement that the names of the newly added elements are unique.

Rule 6. Refinement of Connections in Runnable Entity

$$\frac{CON' = CON \cup \{con'\}}{CON \to CON',\ con' = (c_s, c_t)} \tag{7}$$

Rule 6 represents the refinement of connections between controller elements inside the Runnable.

Rule 7. Subsystem Decomposition Rule in Runnable Entity

$$\frac{C' = C \setminus \{c_i\} \cup SUB'}{C \to C',\ c_i \to SUB'} \tag{8}$$

Rule 7 refines the controller element c_i within the Runnable into a subsystem *SUB'*. It enables recursive nesting of subsystems, allowing the creation of complex control logic.

Next, the visualized AUTOSAR application-layer model will be parameterized to further refine its dynamic simulation behavior, catering to the needs of different simulation scenarios. This paper primarily focuses on configuring the sampling time parameter for the RPort, i.e., *param*(*rport*)[*sample_time*]. Once the structure construction, internal behavior design, and parameters are configured, the AUTOSAR application-layer model can be simulated, and the design can be adjusted and optimized based on the simulation results.

IV. Implementation of AUTOSAR Library and Visual Modeling

A. Development Environment

The implementation of the Modelica-based AUTOSAR model library and its visual modeling are conducted using MWORKS Sysplorer[2], a modeling and simulation tool that supports both graphical (drag-and-drop) and textual Modelica modeling. This enables a flexible and efficient approach to AUTOSAR model development. However, the proposed framework is IDE-agnostic and can be implemented in any Modelica-compliant environment, such as OpenModelica, Dymola, or Wolfram System Modeler.

B. AUTOSAR Library

Based on the encapsulation rules, we constructs a publicly accessible Modelica-based AUTOSAR library[1]. The library structure is shown in Fig. 2, with code statistics summarized in III, excluding blank lines and comments for accuracy.

The library consists of five parts. The overall library structure includes "package.order" and "package.mo" files, defining package dependencies and hierarchy. The data type mapping comprises "Datatype.mo" and "ImplementationDataTypes.mo" files, providing data type definitions essential for application layer interaction. Visualization representation involves the `Annotation` keyword, specifying `Icon` and

[2]https://www.tongyuan.cc/product/MWorksSysplorer

`Diagram` content for graphical visualization. Dynamic behavior is described by code lines under `equation` and `algorithm`, supporting both acausal modeling and causal logic. The static structure includes all remaining code, defining components' static attributes and interfaces.

TABLE III: Code Count of AUTOSAR Model Library

Encapsulation Item	Code Count
Model Library Structure	100+
Data Type Mapping	150+
Static Structure	3000+
Visualization Representation	100+
Dynamic Behavior	50+

C. Visual Modeling Implementation

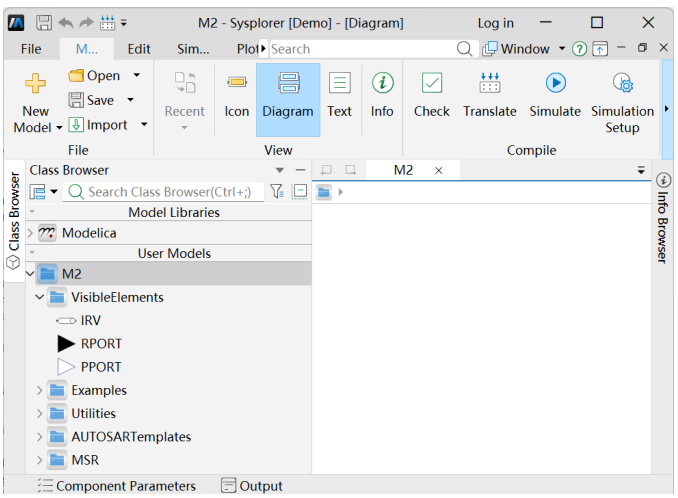


Fig. 3: AUTOSAR Visual Modeling Interface in MWORKS Sysplorer

The visual modeling interface for AUTOSAR application layer models in MWORKS Sysplorer is shown in Fig. 3. The modeling process begins by manually loading the AUTOSAR library. Elements from the M2/VisibleElements package, such as IRV, RPORT, and PPORT, are placed using a drag-and-drop approach. For SWC and Runnable, which may contain nested subsystem-level designs, new models are created, and their internal structures are defined within these models.

MWORKS Sysplorer offers multiple view modes—"Icon," "Diagram," and "Text"—enabling modeling at different levels of abstraction. Simulation parameters can be configured through the "Simulation Setup" option, and the model can be executed by selecting "Simulate".

V. Case Study and Evaluation

This section presents a comparative modeling and simulation study of a multi-rate sampling system using the proposed method and the Simulink/AUTOSAR Blockset to demonstrate the effectiveness of the proposed approach. To demonstrate the applicability of the proposed method beyond traditional AUTOSAR modeling, we extend the study to a multi-domain simulation scenario. Specifically, a Modelica-based multi-domain model is developed for an automotive headlamp flashing control system, showcasing its capability for seamless integration into complex, cross-domain applications.

The case study, adapted from the official AUTOSAR Blockset documentation, models a multi-rate sampling system within the SWC in the AUTOSAR application layer, where signals are sampled at different intervals and Runnables are periodically activated[3]. This mechanism is crucial for ECUs, as automotive systems often integrate sensors and actuators that require updates at varying rates.

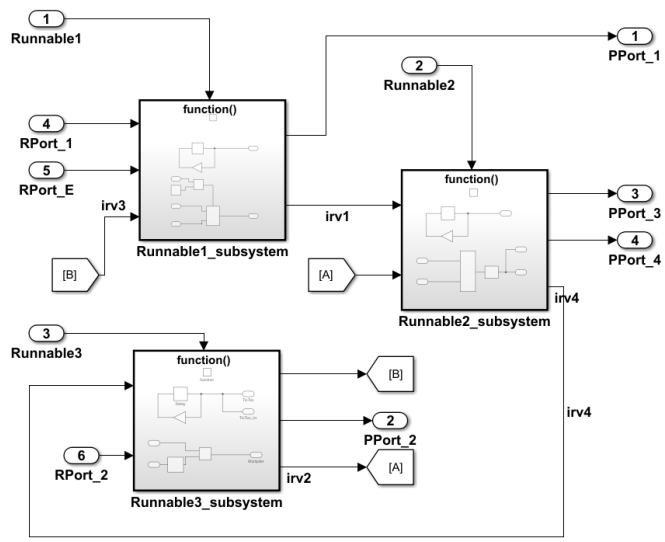


Fig. 4: AUTOSAR Application Layer SWC Established Using Simulink AUTOSAR Blockset

The multi-rate sampling system in the AUTOSAR application layer is modeled as an SWC using the Simulink AUTOSAR Blockset toolbox, as shown in Fig.4. The same system is then modeled with the proposed Modelica-based AUTOSAR framework, following encapsulation rules (Rule1 to Rule3), with results illustrated in Fig.5.

The AUTOSAR application layer model's SWC comprises three RPorts: RPort_E, RPort_1, and RPort_2. RPort_E is used for error state validation, set to 0 to indicate no errors. RPort_1 handles gain control, applying an additive gain (rather than multiplicative) with a 1-second sampling period. The gain is applied to the control output PPort_4 every 1-second cycle, with the gain set to 0 in this model, meaning no additional gain is applied. RPort_2 represents an attenuation factor with a 10-second sampling period. It adjusts PPort_4's output by reducing the previous cycle's value and thus maintains smoother output. In this model, the attenuation factor is set to 0, resulting in no attenuation.

The system also includes four PPorts: PPort_1, PPort_2, PPort_3, and PPort_4. PPort_1 and PPort_3 output periodic signals at 1-second intervals (alternating between -1 and 1), with a 1-second delay. PPort_2 outputs periodic signals at 10-second intervals (alternating between -1 and 1), also with a 1-second delay. The output of PPort_4 is determined by specific

[3]https://ww2.mathworks.cn/help/autosar/ug/example-model-autosar-runnables-exported.html

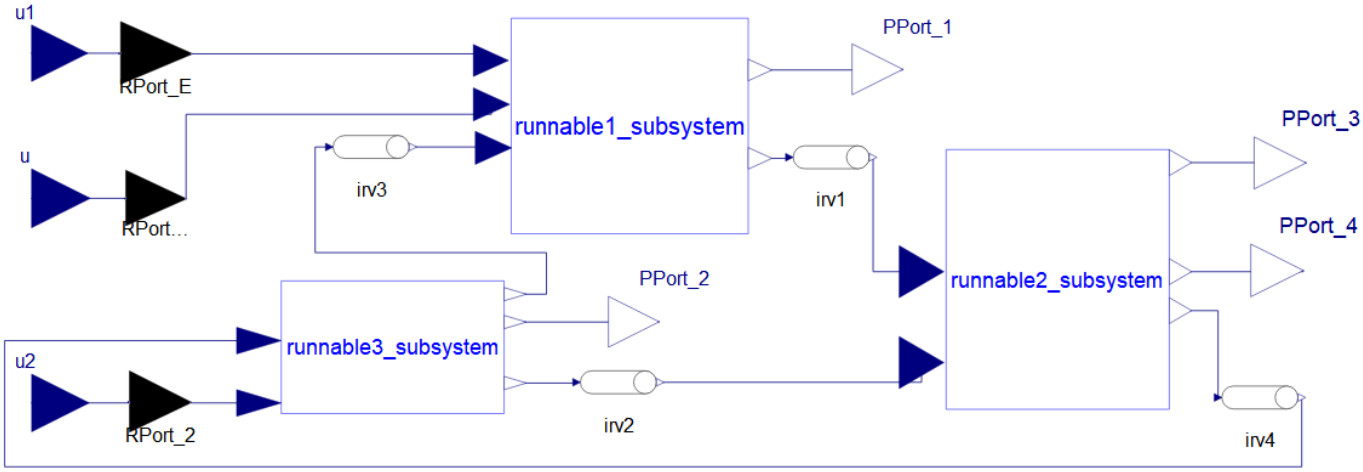
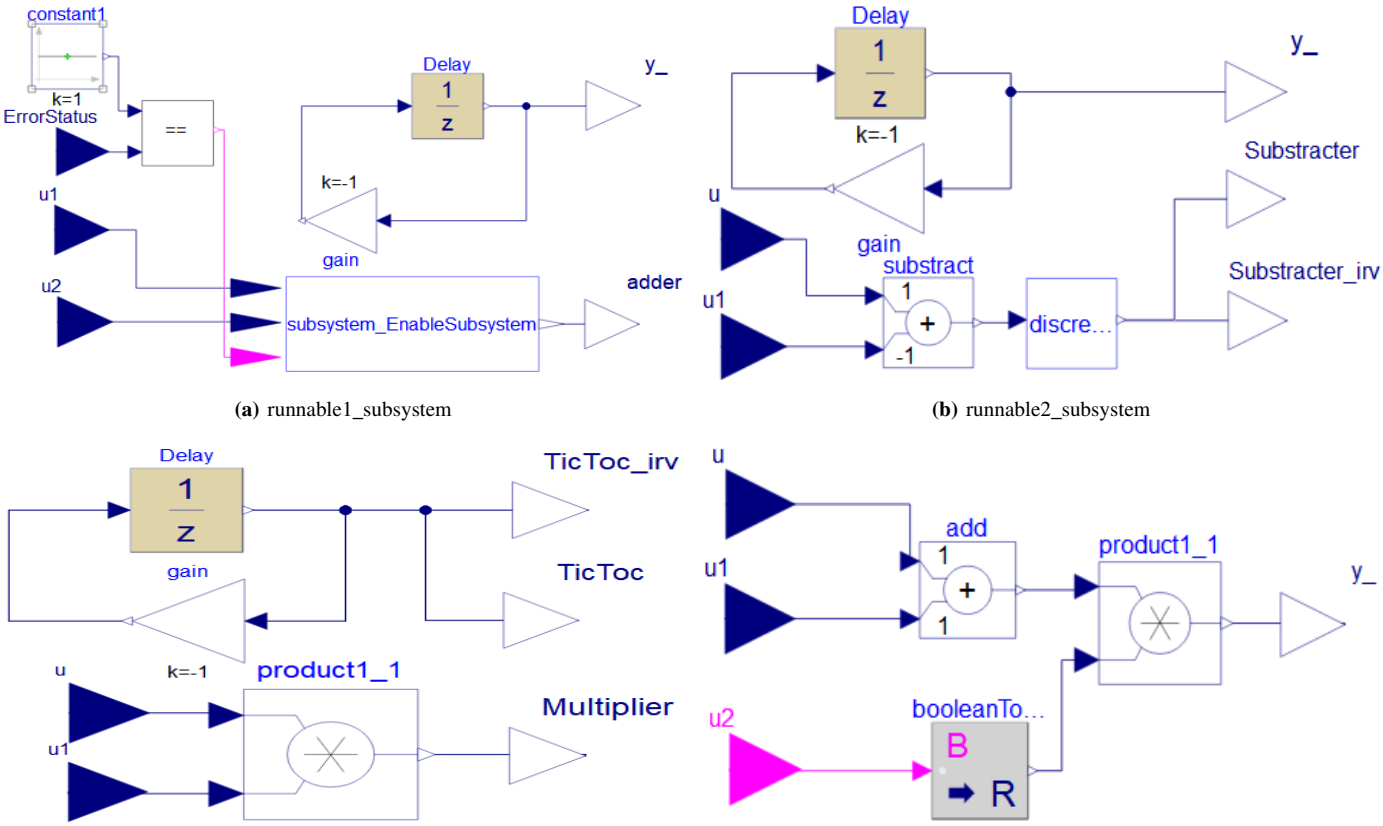


Fig. 5: AUTOSAR Application Layer SWC Established Using Modelica



(a) runnable1_subsystem

(b) runnable2_subsystem

(c) runnable3_subsystem

(d) runnable1_subsystem/subsystem_EnableSubsystem

Fig. 6: Internal Behavior Design of Runnable Entities in the AUTOSAR Application Layer SWC Using Modelica

internal behaviors, which dictate the signal value based on the Modelica model's operations.

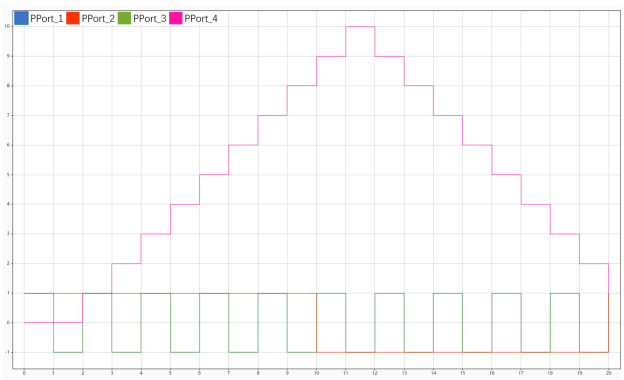
The system includes three Runnables:

- The first and second runnables, connected to RPort_1 and RPort_E, have 1-second sample times.
- The third runnable entity, connected to RPort_2, has a sample time of 10 seconds.
- The entities communicate through IRV.

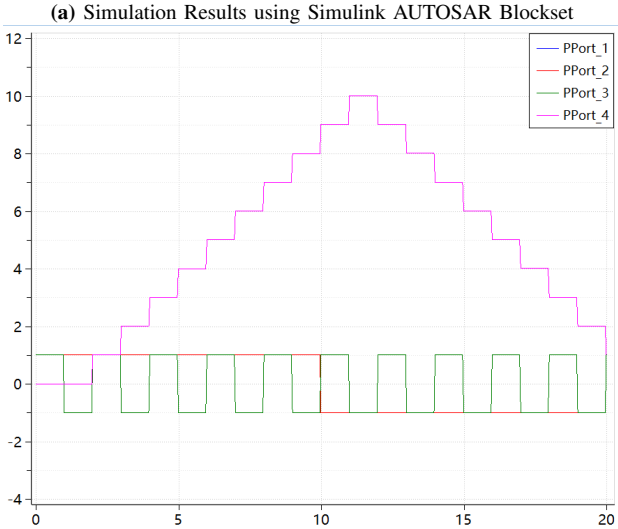
Accordingly, the first and second runnables operate on a 1-second cycle, while the third operates on a 10-second cycle. Following model construction, encapsulation rules (Rule4 to Rule7) are applied to specify the internal behaviors of the runnables. Fig. 6 illustrates the internal behavior design of the Runnables, presenting their detailed structure, including components, names, and connections.

After building the model, simulations are conducted to

validate the design, followed by adjustments based on the simulation results until the desired model is achieved. In this experiment, the simulation time is set to 20 seconds with a total of 500 simulation steps. The AUTOSAR application layer model is simulated using both the Simulink AUTOSAR Blockset toolbox and the proposed Modelica-based approach, with the results shown in Fig. 7. It illustrates the variation of the outputs from four PPorts over time. Notably, the outputs of PPort_1 and PPort_3 are perfectly aligned. The consistency between the simulation results obtained from the commercial Simulink AUTOSAR Blockset and the proposed Modelica-based method demonstrates the validity and correctness of the proposed approach. This case study, included in the "Examples" package of the encapsulated AUTOSAR model library, allows users to directly access and reproduce the results, further validating the proposed method's effectiveness.



(a) Simulation Results using Simulink AUTOSAR Blockset



(b) Simulation Results using the Modelica-based Approach

Fig. 7: Simulation Results of the AUTOSAR Application Layer Model

To demonstrate the advantages of the proposed method in multi-domain modeling, the encapsulated application-layer SWC multi-rate sampling system is integrated with an automotive headlamp for cross-domain co-simulation, as shown in Fig. 8. In this model, all three RPort inputs are set to 0. PPort_1 and PPort_3 are connected to the switch states of "headlamp_left" and "headlamp_right", respectively, indicating whether the headlights are on or off. PPort_4 represents the applied voltage and is linked to both "headlamp_left" and "headlamp_right" to simulate the electrical behavior of the lighting system. The HeadLamp model, shown in Listing 1, has two input ports: switch, which controls the on/off state, and voltageInput, which represents the applied voltage. The model computes the voltage based on the switch state, then applies Ohm's law to determine the current. The power is calculated as the product of voltage and current, and the brightness is derived by normalizing the power against its maximum value, ensuring it remains within [0,1].

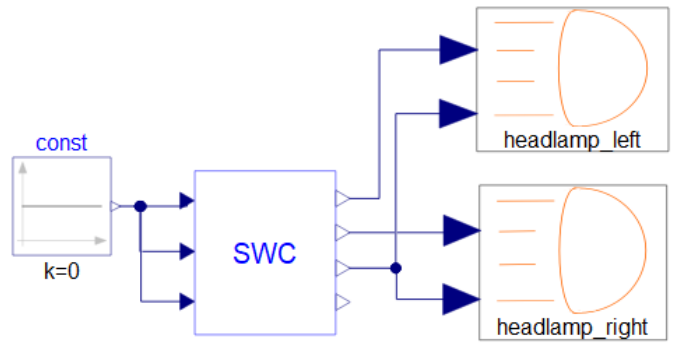


Fig. 8: Multi-Domain Modeling of SWC and Headlamp Using Modelica

Listing 1: Headlamp Electrical Model in Modelica

```
model HeadLamp
  // Import Modelica SI units library
  import Modelica.SIunits;
  // Voltage of the car light (V)
  SIunits.Voltage voltage;
  // Current of the car light (A)
  SIunits.Current current;
  // Power of the car light (W)
  SIunits.Power power;
  // Resistance of the car light (Ω)
  parameter SIunits.Resistance resistance =
      10;
  // Maximum power of the car light (W)
  parameter SIunits.Power P_max = 10;
  // Brightness of the car light (unitless,
      between 0 and 1)
  Real brightness(unit="1");
  // Input voltage signal
  Modelica.Blocks.Interfaces.RealInput
      voltageInput;
  // Switch signal (on/off)
  Modelica.Blocks.Interfaces.RealInput switch;
equation
  // Set voltage based on switch state
  voltage = if switch > 0 then voltageInput
      else 0;
  // Ohm's law: V = I * R
  current = voltage / resistance;
  // Power calculation: P = V * I
  power = voltage * current;
  // Brightness calculation: power / maximum
      power, ensuring 0 ≤ brightness ≤ 1
  brightness = min(max(power / P_max, 0), 1);
end HeadLamp;
```

In the model shown in Fig. 8, we use the same simulation configuration as the previous SWC model, setting the simulation duration to 20 seconds with 500 steps. The simulation results, shown in Fig. 9, demonstrate that the simulation trajectories of the left and right headlamps overlap completely, with the lamps alternately switching on and off every second. Meanwhile, the brightness follows a trend of increasing and then decreasing, creating a ”breathing light” effect, with a 1-second signal delay.

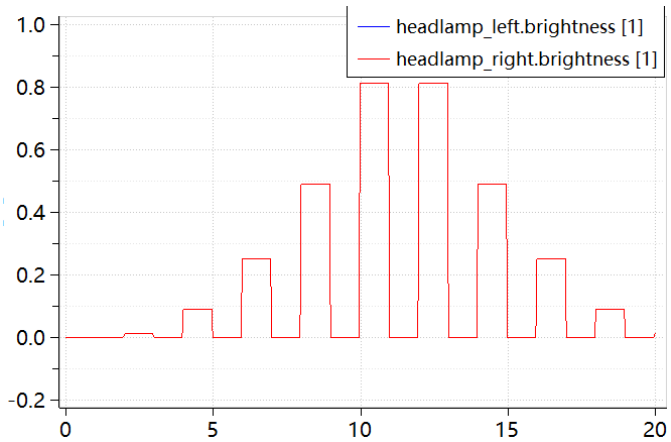


Fig. 9: Multi-Domain Modeling of SWC and Headlamp Using Modelica

The simulation results demonstrate that the model accurately replicates the expected headlamp flashing pattern. In contrast, implementing a similar cross-domain co-simulation in Simulink presents greater challenges, as it requires manually establishing signal dependencies based on causal relationships and may require additional toolboxes for modeling electrical systems. In comparison, the proposed non-causal modeling approach seamlessly captures the interactions between the AUTOSAR model and other physical domain models, simplifying the modeling process and enhancing the flexibility and convenience of multi-domain co-simulation. This highlights its feasibility and advantages for complex system modeling.

VI. Discussion

This paper presents a Modelica-based approach for the visual modeling and simulation of AUTOSAR application layer models. By leveraging Modelica’s multi-domain, equation-based modeling, this method facilitates seamless integration of AUTOSAR models with other domains while ensuring accurate ECU behavior representation. The proposed Modelica AUTOSAR model library simplifies model development, improving flexibility and reusability. Compared to Simulink’s causal, proprietary AUTOSAR Blockset, the proposed non-causal, open-source approach offers superior modeling flexibility, cross-platform compatibility, and multi-domain integration. As automotive systems become more complex, open and non-causal modeling methods like Modelica are increasingly seen as the future trend, enabling scalable and interoperable system design. However, this work is limited to application layer modeling for the Classic Platform and does not yet cover lower-level AUTOSAR modules. While the encapsulation rules simplify model construction, further expansion of the Modelica AUTOSAR library is necessary to support a broader range of AUTOSAR functionalities.

Further research should extend support to the AUTOSAR Adaptive Platform and lower-level modules (e.g., BSW and RTE) to enhance engineering applicability. Additionally, integrating Modelica-based AUTOSAR models with automatic code generation would facilitate seamless deployment to AUTOSAR-compliant ECUs.

VII. Conclusion

This paper introduces a Modelica-based approach for the visual modeling and simulation of AUTOSAR application layer models. Specifically, it establishes encapsulation rules for representing AUTOSAR constructs in Modelica and develops an open-source AUTOSAR model library, fostering industry collaboration and accelerating rapid prototyping. A formal mathematical representation of AUTOSAR models is also provided to enhance both expressiveness and verifiability. Additionally, a structured visual modeling methodology is proposed to reduce the development complexity of AUTOSAR application layer modeling.

The feasibility of employing Modelica for representing AUTOSAR models is validated through the development of a publicly accessible AUTOSAR model library. Experimental results confirm that the Modelica-based AUTOSAR application layer models align with the simulation results obtained from Simulink’s AUTOSAR Blockset, thereby confirming the effectiveness of the proposed method. Although the current work focuses on application layer modeling for the AUTOSAR Classic platform, future research will extend the model library to encompass additional AUTOSAR elements, particularly those related to the Adaptive platform and complex functional modules. Moreover, code generation capabilities will be explored to facilitate the automatic translation of Modelica-based models into executable code, further enhancing the practical implementation of this approach in real-world applications.

References

[1] V. Vyasa and Z. Xu, “Maintenance in automotive and aerospace applications–an overview,” *International Journal of Advances in Science Transactions*, vol. 3, no. 2, pp. 349–361, 2024.

[2] V. Bandur, G. Selim, V. Pantelic, and M. Lawford, “Making the case for centralized automotive e/e architectures,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1230–1245, 2021.

[3] O. Burkacky, J. Deichmann, M. Guggenheimer, and M. Kellner, *Outlook on the automotive software and electronics market through 2030*. McKinsey, 2023.

[4] S. Fürst, J. Mössinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkämper, G. Kinkelin, K. Nishikawa, and K. Lange, “Autosar–a worldwide standard is on the road,” in *14th international VDI congress electronic systems for vehicles, Baden-Baden*, vol. 62, no. 5. Citeseer, 2009.

[5] S. Martínez-Fernández, C. P. Ayala, X. Franch, and E. Y. Nakagawa, "A survey on the benefits and drawbacks of autosar," in *Proceedings of the First International Workshop on Automotive Software Architecture*, ser. WASA '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 19–26. [Online]. Available: https://doi.org/10.1145/2752489.2752493

[6] M. Staron and D. Durisic, *AUTOSAR Standard*. Cham: Springer International Publishing, 2017, pp. 81–116. [Online]. Available: https://doi.org/10.1007/978-3-319-58610-6_4

[7] S. Dersten, J. Axelsson, and J. Froberg, "Effect analysis of the introduction of autosar: A systematic literature review," in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2011, pp. 239–246.

[8] S. Bunzel, "Autosar–the standardized software architecture," *Informatik-Spektrum*, vol. 34, no. 1, pp. 79–83, 2011.

[9] M. A. Shajahan, N. Richardson, N. Dhameliya, B. Patel, S. K. R. Anumandla, and V. K. Yarlagadda, "Autosar classic vs. autosar adaptive: A comparative analysis in stack development," *Eng. Int*, vol. 7, no. 2, pp. 161–178, 2019.

[10] H. Bo, D. Hui, W. Dafang, and Z. Guifan, "Basic concepts on autosar development," in *2010 International Conference on Intelligent Computation Technology and Automation*, vol. 1, 2010, pp. 871–873.

[11] R. Palin, D. Ward, I. Habli, and R. Rivett, *ISO 26262 safety cases: compliance and assurance*, p. B12. [Online]. Available: https://digital-library.theiet.org/doi/abs/10.1049/cp.2011.0251

[12] V. T and V. Hegde, "Design and development of automotive domain needs using rhapsody model manager," in *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2023, pp. 1–6.

[13] C. Knüchel, M. Rudorfer, S. Voget, S. Eberle, R. Sezestre, and A. Loyer, "Artop – an ecosystem approach for collaborative AUTOSAR tool development," in *ERTS 2010 proceedings*, Toulouse, France, May 2010. [Online]. Available: https://hal.science/hal-02267845

[14] A. Urquía Moraleda and C. Martín Villalba, *Modeling and simulation in engineering using Modelica*. Universidad Nacional de Educación a Distancia (España). Editorial, 2018.

[15] K. Qiu, J. Yang, Z. Gao, and F. Xu, "A review of modelica language in building and energy: Development, applications, and future prospect," *Energy and Buildings*, vol. 308, p. 113998, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778824001142

[16] F. Buse, A. Pignède, and S. Barthelmes, "A modelica library to add contact dynamics and terramechanics to multi-body mechanics," in *Modelica Conferences*, 2023, pp. 433–442.

[17] N. Fotias, R. Bao, H. Niu, M. Tiller, P. McGahan, and A. Ingleby, "A modelica library for modelling of electrified powertrain digital twins," in *Modelica Conferences*, 2021, pp. 249–261.

[18] H. Ding, W. Li, C. Duan, H. Xu, P. Ding, Y. Zhang, and G. H. and, "A thermal-hydraulic analysis model of printed circuit heat exchangers for system simulation using modelica," *Journal of Nuclear Science and Technology*, vol. 58, no. 12, pp. 1296–1307, 2021. [Online]. Available: https://doi.org/10.1080/00223131.2021.1939806

[19] Y. Zhou, L. Pan, X. Han, and L. Sun, "Dynamic modeling and thermodynamic analysis of lithium bromide absorption refrigeration system using modelica," *Applied Thermal Engineering*, vol. 225, p. 120106, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359431123001357

[20] D. T. Matt, M. Molinaro, G. Orzes, and G. Pedrini, "The role of innovation ecosystems in industry 4.0 adoption," *Journal of Manufacturing Technology Management*, vol. 32, no. 9, pp. 369–395, 2021.

[21] V. Sanz and A. Urquia, "Cyber–physical system modeling with modelica using message passing communication," *Simulation Modelling Practice and Theory*, vol. 117, p. 102501, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22000120

[22] T. Junjie, Z. Jianjun, D. Jianwan, C. Liping, X. Gang, G. Bin, and Y. Mengfei, "Cyber-physical systems modeling method based on modelica," in *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, 2012, pp. 188–191.

[23] P. Gai, M. Urbina, E. Guidieri, G. Serano, and N. Serreli, "Autosar university package classic platform," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4.

[24] O. Brandes, S. Brege, and P.-O. Brehmer, "The strategic importance of supplier relationships in the automotive industry," *International Journal of Engineering Business Management*, vol. 5, p. 17, 2013. [Online]. Available: https://doi.org/10.5772/56257

[25] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Connallen, and K. A. Houston, "Object-oriented analysis and design with applications, third edition," *SIGSOFT Softw. Eng. Notes*, vol. 33, no. 5, Aug. 2008. [Online]. Available: https://doi.org/10.1145/1402521.1413138