2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

# A SysML v2-Based Framework for Multi-Disciplinary System Virtual Integration

Tianxiao Xu, Néjib Moalla, Mohand-Lounes Bentaha, Hazal Aktekin , Giuseppe Cereda and Claudia Agostinelli

***Abstract*— In the automotive industry, as market demands continue to grow, the complexity of vehicle systems has been increasing accordingly. Vehicle manufacturers are thus confronted with the challenge of offering more design options. Model-Based Systems Engineering (MBSE) has been introduced to manage the complexity of product development. Meanwhile, its integration with Multidisciplinary Design Analysis and Optimization (MDAO) provides a harmonized methodology for the design and analysis of complex systems, offering a system-level virtual integration environment to support decision-making processes. However, current MBSE modeling languages such as SysML v1.x are extensions of the graphical modeling language UML and are considered semi-formal. Their limited capability to extend and integrate with modeling languages across different disciplines significantly restricts their effectiveness. SysML v2, built upon the formal modeling language KerML and supporting standardized APIs, enhances model interoperability. By utilizing SysML v2 to develop a domain-specific modeling language tailored for multidisciplinary engineering, it becomes possible to effectively maintain data consistency between system design and analysis processes, establish a virtual integration platform, and ensure interoperability with other environments. This paper presents a SysML v2-based multidisciplinary system virtual integration framework, implemented by constructing a coupled multidisciplinary problem model in the context of automotive systems.**

## I. Introduction

The increasing complexity of automotive systems necessitates efficient system design and validation processes [1]. Implementing early-stage verification and validation (V&V) effectively assesses the quality of system design and helps avoid the discovery of defects during the implementation phase, which reduces development costs [2]. Model-Based Systems Engineering (MBSE) has been proposed as an efficient approach to manage system complexity [3], and to drive multidisciplinary system design, analysis, and optimization, providing support for informed decision-making in the selection of design solutions [4].

Tianxiao Xu is with Université Lumière Lyon 2, DISP laboratory, 69007 Lyon, France and IVECO France, 69200 Vénissieux, France (corresponding author, phone: +33 782012389; e-mail: tianxiao.xu@univ-lyon2.fr).

Néjib Moalla is with Université Lumière Lyon 2, DISP laboratory, 69007 Lyon, France (e-mail: nejib.moalla@univ-lyon2.fr).

Mohand-Lounes Bentaha is with Université Lumière Lyon 2, DISP laboratory, 69007 Lyon, France (e-mail: mohand.bentaha@univ-lyon2.fr).

Hazal Aktekin is with IVECO France, 69200 Vénissieux, France (e-mail: hazal.aktekin@ivecogroup.com).

Giuseppe Cereda is with IVECO SPA, 10156 Torino, Italy (e-mail: giuseppe.cereda@ivecogroup.com).

Claudia Agostinelli is with IVECO SPA, 10156 Torino, Italy (e-mail: claudia.agostinelli@ivecogroup.com).

The practice of MBSE greatly benefits from system modeling languages such as Systems Modeling Language (SysML) [5]. SysML provides comprehensive semantics and powerful syntax, enabling effective representation of system functions, states, and logic. Through its extensibility mechanisms [6], Domain-Specific Modelling Language (DSML) can be packaged as a profile with in SysML. It enhances interoperability with external analysis tools, supporting a virtual integration platform that facilitates the evaluation and validation of design concepts before physical implementation [7]. By using MBSE to drive other engineering disciplines, certain MBSE capabilities, such as model checking [8] and generative AI [9], are also applied in the Multidisciplinary Design Analysis and Optimization (MDAO) process [10][11].

By establishing a system model, MBSE provides a consistent and reusable Single Source of Truth (SSoT) throughout the entire product lifecycle, supporting data continuity [12]. SysML v2 is the next generation version of SysML. It addresses several limitations of SysML v1 by introducing a more formal, expressive, and semantically precise modeling foundation [13]. In addition, its API offers a standard set of services for interacting with SysML v2 models, enabling effective management, scalability, and integration with other product information in the context of Digital Engineering [14]. The digital continuity between system design and analysis processes has been initially achieved using SysML v2 language features [15].

This study aims to harmonize the design and analysis processes in the development of complex systems, ensuring a closer collaboration across multiple engineering domains. By establishing a system-level model, it ensures data consistency among different engineering disciplines throughout the product lifecycle. Furthermore, the research is dedicated to developing a highly extensible and modular system architecture to enable interoperability among various design and analysis models, enhancing the scalability and applicability of the proposed approach in practical implementations.

The expected results of this study include the development of a semantically integrated framework to unify system design and analysis, enabling consistent representation of multidisciplinary relationships. A generalized modeling language will be established to support this integration, ensuring clear communication across domains and among stakeholders. In addition, mechanisms for interoperability between design and analysis models will be ensured, allowing the architectural model to serve as a centralized and authoritative data source, which improves overall system development efficiency.

978-1-6654-5784-2/26/$31.00 ©2026 IEEE

The motivation for this study stems from several critical issues identified in current practices. First, there is a lack of clear and standardized methods for semantic integration between MBSE and MDAO, which restricts collaboration between system-level design and domain-specific analysis. Second, challenges remain in achieving a unified and modular definition of system elements, which is essential for enabling reuse, scalability, and traceability within complex engineering processes. Lastly, as a semi-formal language based on UML, SysML v1 exhibits significant limitations in integrating with other tools and platforms, often requiring ad-hoc solutions that not only increase development effort but also complicate future upgrades. These issues collectively highlight the necessity of establishing a more formal, extensible, and interoperable modeling framework.

Therefore, the research problem addressed in this paper is how to formally integrate MBSE and MDAO using SysML v2 to enable a unified, modular, and interoperable modeling framework that supports multidisciplinary system design and analysis. Although the semantics and metamodels for system design and analysis have been developed and applied in various domains, they operate within different environments and face integration challenges. By introducing the novel formal modeling language SysML v2, this study realizes a multidisciplinary virtual integration framework that empowers the current capabilities of MBSE to support multidisciplinary design analysis and optimization.

This paper focuses on how to utilize SysML v2, a formal modeling language, to support and harmonize the design and analysis activities of complex systems, by defining the analysis architecture to facilitate the analysis of multidisciplinary problems. It aims to enhance problem-solving capabilities through the integration of MBSE and MDAO activities. However, since the processes for solving multidisciplinary problems and constructing discipline models have already been extensively studied, this paper will primarily demonstrate the definition and application of the proposed method.

## II. Related work

To achieve a harmonized design and analysis process, it is often necessary to integrate the SysML used in the MBSE with the modeling languages used in analysis such as MDAO [6][15]. This integration ensures digital continuity between the two environments and enables closed-loop information exchange[4][7].

### A. MDAO

*MDAO formulation:* Multidisciplinary Design Analysis and Optimization (MDAO) is a methodology used for designing complex engineering systems. It aims to enhance system performance, reduce costs, and shorten the design cycle through interdisciplinary collaborative analysis and optimization. Multidisciplinary Design Optimization (MDO) focuses on addressing design problems that involve coupled analysis models [16]. A formal definition of the consolidated statement of a multidisciplinary optimization problem has been defined in [17].

*MDAO structure:* The extended design structure matrix (XDSM) has been introduced to provide a standardized method for representing architectures that address multidisciplinary design optimization problems. This diagram enhances the conventional design structure matrix (DSM) by incorporating both data dependencies and process flows within a single unified representation [18].

*MDAO semantics:* By driving MDAO systems through MBSE, the deployment and operation of analysis systems are accelerated, enhancing collaboration across multidisciplinary domains [19]. To achieve integration between MBSE and MDAO and ensure data consistency, a semantic-level integration approach has been proposed. This approach enables the use of a unified modeling language to describe both the system architecture and the MDAO architecture [20]. The data model of this semantic integration method explicitly defines various MDAO metamodels within the MBSE environment and emphasizes the relationships among them. Digital artifacts in MDAO are linked to constraints and partial attributes within the system architecture, enabling MDAO simulation and analysis functionalities.

### B. MBSE

*MBSE application:* Model-Based Systems Engineering (MBSE) is a holistic, SE approach centered on the evolving system model, which serves as the Single Source of Truth (SSoT) about the system [21]. In the application of MBSE, there are many common methodologies, such as Object-Oriented Systems Engineering Method (OOSEM) [22] and Requirements-Functional-Logical-Physical (RFLP) [23]. These methodologies are able to be tailored to fit specific business processes. For example, Dassault Systèmes' MagicGrid [24] has four pillars that align with RFLP, but it allows for scalability across different system levels.

*MBSE Language:* SysML is a standardized modeling language specifically designed for SE, based on Unified Modeling Language (UML) but extended and modified to better handle the unique needs of complex systems beyond software [6]. The SysML language includes nine diagram types that describe various aspects of complex systems, such as requirements, behavior, and structure, and effectively integrate with the modeling process. As a general modeling language, SysML offers an extension mechanism that enables different domains to encapsulate their DSML as profiles [7].

### C. SysML V2

*SysML V2 Language:* SysML v2 is the next generation systems modeling language to address many of the limitations that were identified with SysML v1 by introducing a more formal, expressive, and semantically precise modeling foundation [13]. SysML v2 is built upon the Kernel Modeling Language (KerML) and supports both graphical and textual modeling representations, enhancing the precision and consistency of system modeling and enabling a more formal expression of system semantics [25].

*Modelling API:* One of the most significant advancements of SysML v2 is its standardized and open Application Programming Interface (API), which enables external tools to programmatically interact with system models. This interface

supports model querying, updating, validation, and transformation, facilitating interoperability with analysis tools, simulation environments, and other engineering services [26]. The SysML v2 API helps build a model-driven digital thread across product lifecycle to enable data exchange and to reduce reliance on ad-hoc solutions. In addition, SysML v2 adopts a modular and extensible architecture that supports extensions through domain-specific languages, thereby promoting its scalable application in multidisciplinary environments [14].

### D. Research gap

Although many DSML implementations have been developed by importing profiles into SysML, there remains a significant research gap in feasible approaches for integrating design and analysis processes, as well as achieving subsequent model interoperability. Most DSML extensions are based on SysML v1. Due to several limitations of SysML v1, such as the lack of formal semantics and standardized APIs, the integration of model environments highly relies on ad-hoc solutions or tool-specific implementations. Under the growing needs for V&V in multidisciplinary system integration, this limits the reusability of digital artifacts, the scalability of virtual integration platforms, and upgrade of future verification technologies. SysML v2 has been introduced as the next-generation systems modeling language to address the shortcomings of its predecessor. However, there is currently no practical case demonstrating semantic integration between MBSE and MDAO using SysML v2. To address this research gap, this paper proposes a SysML v2-based framework for multidisciplinary system virtual integration by a case study of a vehicle system.

## III. Methodology

The methodology presented in this research focuses on integrating SysML v2 with a MDAO DSML. Converting system architecture models into SysML v2 models enables the virtual integration of models from different disciplines. This framework consists of 3 components: metadata definition, part definition, and part usage, as shown in Fig. 1. After defining the semantic elements in MDAO as metadata, modular and reusable definitions are created to describe artifacts. Finally, by utilizing these modules through part usage to establish relationships, different alternative solutions are generated.

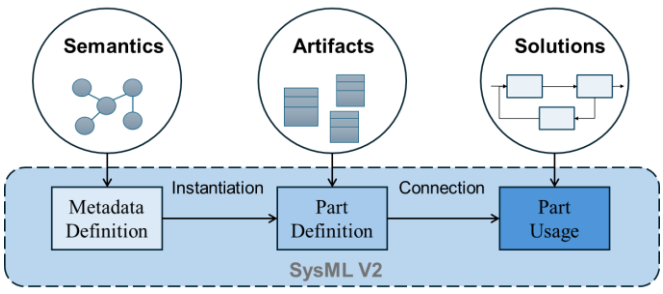


Figure 1. SysML v2 metadata definition process

### A. Metadata def

The integration begins with the formal definition of metadata, which establishes the fundamental elements and semantics required to represent key MDAO concepts within the SysML v2 framework. The metadata definition ensures that domain-specific entities such as optimizers, analyzers, and disciplines are precisely characterized with their attributes and relationships, enabling consistent interpretation across different models. The defined metadata is derived through the specialization of SysML v2's semantic elements. The extended semantics are expressed using the same syntax. This ensures that both the system and analysis architecture are interpreted within the same environment, including analysis models accessed and interoperated through the API.

### B. Part Def

Different design and analysis elements are defined through imported domain-specific metadata. These components serve as fundamental building blocks for constructing complex multidisciplinary system models. Domain-specific algorithms or optimization processes have been encapsulated with defined parameters and expressions. This modular definition approach promotes reuse and standardization of domain-specific models. On the one hand, different disciplines or system components are developed based on a unified modeling standard. On the other hand, standardized definitions reduce the complexity of model integration and verification. The definition and updating of domain models are performed independently. By using the defined metadata, the established definitions are annotated, which facilitates the classification of elements and guides their subsequent usage.

### C. Part Usage

By invoking previously defined multidisciplinary element definitions through usage constructs, relationships among different elements are established to realize the MDAO analysis architecture. In the integrated MBSE environment, the input and output parameters required by analysis elements are bound to the value properties of system elements. As a result, before the analysis begins, parameters such as constants and initial values required by the analysis architecture are synchronized from the corresponding value properties in the system architecture. After the analysis is completed, feasible alternatives are preserved as instantiated configurations. These instances include both simulation results and assigned values of system architecture properties. This usage approach promotes the reuse of modular definitions. For example, in constructing architectures for different multidisciplinary analysis processes, the same fundamental units may be reused to form different relationships or generate different instances from the same fundamental modules.

### D. Methodology verification

The MDAO metadata is defined and embedded within the SysML v2 model to ensure that the design and analysis architectures of complex systems are defined within a unified environment. Interoperability is tested by exchanging the model with external tools via SysML's standardized APIs, ensuring that no semantic information is lost during data exchange. This approach directly integrates MDAO metadata into SysML v2 as a unified architecture within MBSE. Unlike previous semi-formal integration methods, the MDAO DSML maintains the same interoperability mechanisms as SysML v2, which ensures the feasibility of integrated verification.

## IV. IMPLEMENTATION

This case study focuses on the design of a vehicle system and its 2 subsystems: the powertrain and the chassis, as illustrated in Fig. 2. The system-level value properties comprise autonomy, passenger density, and battery energy. For the powertrain, the value properties include battery weight, cabin temperature, battery technology, and payload weight. The chassis is characterized by value properties such as the number of seats and seat configuration.

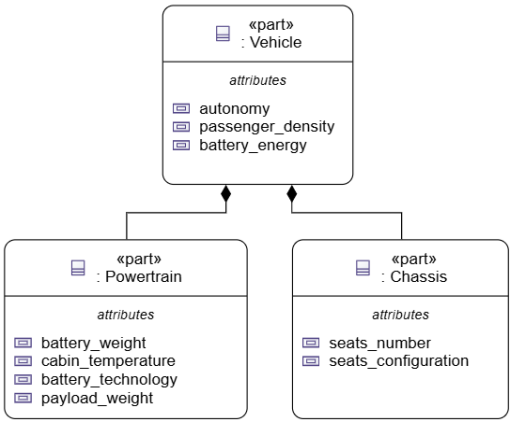


Figure 2. System architecture

In order to illustrate the development of an analysis architecture within an MBSE environment, a multidisciplinary problem statement is defined, incorporating two disciplinary analysis models: Passenger Payload and Energy Consumption. The objective is to optimize the vehicle's autonomy, subject to the constraint that passenger density does not exceed its rated value. The design variable considered is battery energy. Value properties defined at the subsystem level serve as default inputs and identify parameters to be computed.

```
package 'MDAO DSML' {
        import Metaobjects::SemanticMetadata;

        metadata def <opt> optimizer specializes SemanticMetadata {
            redefines baseType = optimizers meta SysML::ConstraintDefinition;
            subsets annotatedElement : SysML::ConstraintDefinition;
        }

        metadata def <disc> discipline specializes SemanticMetadata {
            redefines baseType = disciplines meta SysML::ConstraintDefinition;
            subsets annotatedElement : SysML::ConstraintDefinition;
        }

        metadata def <an> analyzer specializes SemanticMetadata {
            redefines baseType = analyzers meta SysML::ConstraintDefinition;
            subsets annotatedElement : SysML::ConstraintDefinition;
        }
}
```

Figure 3. MDAO DSML package

### A. Metadata package

The metadata package contains the semantic definitions required by the MDAO DSML in this case study, as shown in Fig. 3. It includes three types of metadata: optimizer, analyzer, and discipline, all of which inherit from a common base semantic metadata. These metadata elements redefine their base types as subsets of SysML's Constraint Definition, enabling precise annotation and classification of MDAO-related concepts within the SysML model.

### B. MDAO elements definition

In the MDAO architecture package, the elements required to resolve the multidisciplinary problem have been defined. As shown in Fig. 4, the MDAO DSML package needs to be imported so that the relevant elements are properly annotated. In this part of the model, an optimizer and a system analyzer have been defined. The optimizer is responsible for solving for the optimal autonomy and passenger density, ensuring constraint satisfaction, and providing the design space for battery energy from its initial value. The system analyzer handles the coupled parameters of the multidisciplinary problem, including battery weight, cabin temperature, number of seats, and passenger density, and requires battery energy as an input. The calculation formulas for each parameter, such as the definition of the design space, are specified using the semantic type of calculation, which can be further expanded to define the analysis algorithms.

```
package 'MDAO Architecture' {
        import MDAO DSML::*;

        constraint def sysOptimizer {
            @optimizer
            in battery_energy0 : IntegerValue;
            in autonomy : IntegerValue;
            in passenger_density : IntegerValue;

            battery_energy == DesignSpace(battery_energy0, autonomy,
            passenger_density) and
            autonomy_opt == Optimisation(autonomy) and
            passenger_density_opt == SubjectTo(passenger_density) ;
        }

        constraint def sysAnalyzer {
            @analyzer
            in battery_weight_in : IntegerValue;
            in cabin_temperature_in : IntegerValue;
            in seats_number_in : IntegerValue;
            in passenger_density_in : IntegerValue;
            in battery_energy : IntegerValue;

            battery_weight_out == Calc_obs(battery_weight_in) and
            cabin_temperature_out == Calc_obs(cabin_temperature_in)
            and
            seats_number_out == Calc_obs(seats_number_in) and
            passenger_density_out == Calc_obs(passenger_density_in);
        }
```

Figure 4. MDAO system definition

2 disciplinary analysis modules, Passenger Payload and Energy Consumption, are established. The Passenger Payload module is used to compute the number of seats, passenger density, and payload capacity. The Energy Consumption module is used to compute battery weight, cabin temperature, and autonomy. These disciplinary models perform internal calculations and trade-off analyses based on their respective inputs, as shown in Fig. 5. Each expression used to compute

output parameters is also defined in a modular way by using the calculation construct, enabling reuse.

```
constraint def Passenger_Payload{
    @discipline
    in battery_weight : IntegerValue;
    in cabin_temperature : IntegerValue;
    in seats_configuration : IntegerValue;

    seats_number == Calc_ seats(battery_weight,
    cabin_temperature, seats_configuration) and
    payload_weight == Calc_ payload(battery_weight,
    cabin_temperature, seats_configuration) and
    passenger_density == Calc_ density(battery_weight,
    cabin_temperature, seats_configuration) ;
}

constraint def Energy_Consumption{
    @discipline
    in seats_number : IntegerValue;
    in passenger_density : IntegerValue;
    in battery_energy : IntegerValue;
    in battery_technology : IntegerValue;

    battery_weight == Calc_weight(seats_number,
    passenger_density, battery_energy, battery_technology) and
    cabin_temperature == Calc_temperature(seats_number,
    passenger_density, battery_energy, battery_technology) and
    autonomy == Calc_autonomy(seats_number,
    passenger_density, battery_energy, battery_technology) ;
}
```

Figure 5. Discipline definition

```
part MDAO system {
    ref vehicle : Vehicle;
    attribute obs_battery_weight : IntegerValue;
    attribute obs_ cabin_temperature: IntegerValue;
    attribute obs_seats_number : IntegerValue;
    attribute obs_passenger_density : IntegerValue;
    attribute cp_autonomy : IntegerValue;
    attribute cp_battery_energy : IntegerValue;

    assert constraint opt1 : sysOptimizer {
        in battery_energy0 = vehicle.battery_energy;
        in autonomy = cp_autonomy;
        in passenger_density = obs_passenger_density;
        out battery_energy = cp_battery_energy;
        out autonomy_opt = vehicle.autonomy;
        out passenger_density_opt = vehicle.passenger_density;
    }

    assert constraint an1 : sysAnalyzer {
        in battery_weight_in = obs_battery_weight;
        in cabin_temperature_in = obs_ cabin_temperature;
        in seats_number_in = obs_seats_number;
        in passenger_density_in = obs_passenger_density;
        in battery_energy = cp_battery_energy;
        out battery_weight_out = obs_battery_weight;
        out cabin_temperature_out = obs_ cabin_temperature;
        out seats_number_out = obs_seats_number;
        out passenger_density_out = obs_passenger_density;
    }
```

Figure 6. Constraint assertions

## C. *MDAO architecture*

In SysML v2, constraint assertions are used to define the usage of constraint definitions. In the MDAO system model, an analysis model equivalent to a parametric diagram has been established, as shown in Fig. 6. This analysis architecture needs to refer to the definitions from the system architecture. In this model, 6 status variables have been defined firstly, such as battery weight and cabin temperature, to store intermediate values during the computation process. Then, constraint assertions have been performed, to instantiate the optimizer and system analyzer. The parameters of the constraints are bound to the corresponding status variables or system value properties. For example, in the system optimizer, the battery energy parameter is provided through the vehicle value property, while the autonomy parameter comes from the observable variable. The analyzer is asserted in the same way. Subsequently, the two disciplines, Passenger Payload and Energy Consumption, are asserted, as illustrated in Fig. 7. Once the computation is complete, the value properties within the alternatives are populated accordingly.

```
    assert constraint disc1 : Passenger_Payload {
        in battery_weight = obs_battery_weight;
        in cabin_temperature = obs_ cabin_temperature;
        in seats_configuration = vehicle.chassis.seats_configuration;
        out seats_number = obs_seats_number;
        out seats_number = vehicle.chassis.seats_number;
        out payload_weight = vehicle.powertrain.payload_weight;
        out passenger_density = obs_passenger_density;
    }

    assert constraint disc2 : Energy_Consumption {
        in seats_number = obs_seats_number;
        in passenger_density = obs_passenger_density;
        in battery_energy = cp_battery_energy;
        in battery_technology = vehicle.powertrain.battery_technology;
        out battery_weight = obs_battery_weight;
        out battery_weight = vehicle.powertrain.battery_weight;
        out cabin_temperature = obs_ cabin_temperature;
        out cabin_temperature = vehicle.powertrain. cabin_temperature;
        out autonomy = cp_autonomy;
    }
}
```

Figure 7. Discipline assertions

## D. *Validation results*

The developed case study demonstrates that, in the metadata definition phase, 3 metadata representing MDAO concepts were created, and 4 subsequently defined definitions were semantically annotated using these metadata. In the part definition phase, a system architecture related to the trade-off case study was established, consisting of 3 blocks and 9 value properties. Subsequently, 4 analysis elements were defined, including 1 optimizer, 1 system analyzer, and 2 disciplines, comprising a total of 15 input parameters and 20 output parameters. The expressions or algorithms for each output parameter were implemented through internal references to other part definitions of the calculation type. Since changes to the element definitions do not affect the internal structure of the calculation parts, this approach ensures modularity and supports reusability.

In the part usage phase, a solution was established to connect the value properties of the system architecture with the parameters of the analysis architecture, resulting in 6 coupling variables and 29 binding connections. The analysis architecture generated in this case study is formally described within the SysML v2 environment, and includes the same types and quantities of elements, binding connections, and coupling relationships as required in a typical MDAO execution environment. This validates the feasibility of the proposed methodology in building both system and analysis architectures within the same modeling environment.

## V. Conclusion

This paper presents a SysML v2-based framework for multi-disciplinary system virtual integration, addressing the MDAO problem-resolving for complex systems. By defining domain-specific metadata for MDAO within the SysML v2 environment, the framework enables semantic integration of MDAO processes directly into MBSE. This integration enhances traceability, data consistency, and interoperability among different disciplines. Early V&V is facilitated through virtual integration of analysis models, supporting decision-making in complex system design. The proposed approach demonstrates significant potential for improving collaborative development workflows across multiple engineering domains. Future work will aim to implement the proposed methodology through the implementation of case studies involving multidisciplinary system integrated simulation and continuous optimization across system lifecycle stages, utilizing the interoperability capabilities offered by SysML v2.

## Acknowledgment

This paper presents results that are developed in collaboration between the IVECO Group France company and the University Lumiere Lyon 2, DISP Lab. This research is established under a CIFRE contract (2024/0091). The content of this paper reflects an R&D initiative promoted by IVECO Group France. Responsibility for the information and views expressed in this paper lies entirely with the authors.

## References

[1] S. Kinay, U. Bolat, B. Y. Gökdemir, K. Babacan, N. Zengin, and E. Özkaya, "Advancing MBSE for ADAS/AD: Automated Scenario Generation," in *Proc. IEEE/SICE Int. Symp. on System Integration (SII)*, Jan. 2024, pp. 1583–1588.

[2] J. Cederbladh, A. Cicchetti, and J. Suryadevara, "Early validation and verification of system behaviour in model-based systems engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 3, pp. 1–67, 2024.

[3] A. W. Wymore, *Model-Based Systems Engineering*, Boca Raton, FL: CRC Press, 2018.

[4] O. Aïello *et al.*, "Populating MBSE Models from MDAO Analysis," in *Proc. 7th IEEE Int. Symp. on Systems Engineering (ISSE)*, 2021, doi: 10.1109/ISSE51541.2021.9582519.

[5] Object Management Group, *OMG Systems Modeling Language (OMG SysML™), Version 1.7*, 2024.

[6] Object Management Group, *SysML Extension for Physical Interaction and Signal Flow Simulation*, Version 1.1, OMG Document Number: formal/21-05-03, 2021.

[7] J. Zhao, H. Liang, X. Gao, T. Xu, and H. Yan, "Co-simulation architecture and platform establishment method for cloud-based predictive cruise control system," *Automotive Innovation*, vol. 7, no. 4, pp. 658–668, 2024.

[8] P. de Saqui-Sannes, L. Apvrille, and R. Vingerhoeds, "Checking SysML models against safety and security properties," *J. Aerosp. Inf. Syst.*, vol. 18, no. 12, pp. 906–918, 2021.

[9] M. Chami, C. Zoghbi, and J. M. Bruel, "A first step towards AI for MBSE: Generating a part of SysML models from text using AI," *A First Step towards AI*, 2019.

[10] T. Xu, N. Moalla, M. L. Bentaha, H. Aktekin, and C. Agostinelli, "AI-Integrated Framework for Enhancing High Level Architecture Design Across System Lifecycle Stages," in *Proc. 13th Int. Conf. on Model-Based Software and Systems Engineering*, Feb. 2025.

[11] T. Xu, N. Moalla, M. L. Bentaha, H. Aktekin, G. Cereda, and C. Agostinelli, "A feasible AI application identification approach for enhancing MDAO processes," in *Proc. IFIP 22nd Int. Conf. on Product Lifecycle Management*, Jul. 2025.

[12] S. Wu, J. Lu, Z. Hu, P. Yang, G. Wang, and D. Kiritsis, "Cognitive thread supports system of systems for complex system development," in *Proc. 16th Int. Conf. System of Systems Engineering (SoSE)*, Jun. 2021, pp. 82–87.

[13] S. Friedenthal, "Future Directions for MBSE with SysML v2," in *MODELSWARD*, Feb. 2023, pp. 5–9.

[14] M. Bajaj, S. Friedenthal, and E. Seidewitz, "Systems modeling language (SysML v2) support for digital engineering," *Insight*, vol. 25, no. 1, pp. 19–24, 2022.

[15] J. A. Zhang, B. Bagdatli, and D. N. Mavris, "Leveraging SysML V2 for integration of MBSE and multidisciplinary system development," in AIAA SCITECH 2023 Forum, 2023, p. 1895.

[16] A. Jeyaraj, N. Tabesh, and S. Liscouet-Hanke, "Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting – A case study within the AGILE4.0 project," in *AIAA AVIATION 2021 FORUM*, 2021, doi: 10.2514/6.2021-3077.

[17] J. Sobieszczanski-Sobieski, A. Morris, and M. Van Tooren, *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. Hoboken, NJ: John Wiley & Sons, 2015.

[18] A. B. Lambe and J. R. R. A. Martins, "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, 2012, doi: 10.1007/s00158-012-0763-y.

[19] P. D. Ciampa, G. La Rocca, and B. Nagel, "A MBSE approach to MDAO systems for the development of complex products," in *AIAA Aviation 2020 Forum*, 2020, p. 3150.

[20] T. Xu, N. Moalla, M. L. Bentaha, H. Aktekin, and C. Agostinelli, "A MBSE-enhanced Semantically Integration Method for populating MDAO design processes," in *2025 IEEE International Systems Conference (SysCon)*, 2025, pp. 1–8.

[21] A. M. Madni and M. Sievers, "Model-based systems engineering: Motivation, current status, and research opportunities," *Systems Engineering*, vol. 21, no. 3, pp. 172–190, 2018, doi: 10.1002/sys.21438.

[22] R. Fusaro, D. Ferretto, and N. Viola, "Model-Based Object-Oriented Systems Engineering Methodology for the Conceptual Design of a Hypersonic Transportation System," in *IEEE International Symposium on Systems Engineering (ISSE)*, 2016, pp. 1–8, doi: 10.1109/ISSE.2016.7753175.

[23] S. Kleiner and C. Kramer, "Model Based Design with Systems Engineering Based on RFLP Using," *Smart Product Engineering*, vol. 6, pp. 93–102, 2013, doi: 10.1007/978-3-642-30817-8_10.

[24] A. Aleksandraviciene and A. Morkevicius, *MagicGrid® Book of Knowledge: A Practical Guide to Systems Modeling Using MagicGrid from Dassault Systèmes*, 2nd ed. Kaunas: Vitae Litera, 2021.

[25] Object Management Group, *OMG Systems Modeling Language (SysML) v2 Specification* (beta/2025-04-05). [Online]. Available: https://www.omg.org/spec/SysML/2.0/

[26] Object Management Group, *SysML v2 API and Services Specification* (beta/2023-03-01). [Online]. Available: https://www.omg.org/spec/SysML-API/1.0/