engineering proceedings

CITESCORE 1.8

Proceeding Paper

HADES—A Framework for Hierarchical Architecture Design for Engineering Systems

Marcel Mischke, Durga Sri Sharan Katabathula and Leonardo Francelino Melico

MDPI

https://doi.org/10.3390/engproc2026133155

engineering proceedings

MDPI

Proceeding Paper

HADES—A Framework for Hierarchical Architecture Design for Engineering Systems †

Marcel Mischke *, Durga Sri Sharan Katabathula and Leonardo Francelino Melico

German Aerospace Center (DLR), Institute of Electrified Aero Engines, 03046 Cottbus, Germany; durga.katabathula@dlr.de (D.S.S.K.); leonardo.melico@dlr.de (L.F.M.)

* Correspondence: marcel.mischke@dlr.de

† Presented at the 15th EASN International Conference, Madrid, Spain, 14–17 October 2025.

Abstract

The increasing complexity of technical aeronautical systems requires novel and consistent methods to manage a wide range of dependencies and disciplines. Model-based systems engineering addresses this need by providing a systematic, model-based approach to support development, analysis, and validation within a systemic context. This paper presents the strategy behind HADES, an abstract modelling framework that enables a well-structured and generic description of complex systems. The system information is clearly defined in a hierarchical structure and supported by the use of various viewpoints that consistently integrate different systemic aspects. HADES extends the classic RFLP viewpoints with an operational view. In addition, system safety is methodically embedded in the model-based development process from the initial stages as a key aspect. HADES thus supports a traceable development strategy for aeronautical systems according to SAE ARP4754B.

Keywords: architecture; framework; MBSA; MBSE; RFLP; safety; system; viewpoint

1. Introduction and State of the Art

The development of complex technical systems is a constantly growing challenge due to increasing systemic complexity [1] (p. 4). To address this, traditional document-based systems engineering (SE) has been adapted to a model-based systems engineering (MBSE) approach. MBSE represents an extension of the SE process, enabling the representation, analysis, and management of complex systems through abstract models [2]. This approach offers clear communication, reduced errors, and optimized consistency of information across different disciplines. A standardized and rule-based modelling language, such as the Systems Modelling Language (SysML), is required for implementation.

SysML is a notation-based and graphical language published in 2007 by the Object Management Group [3] (p. xvii, p. 13). The basic idea is to define information as specific elements and place them in associated diagrams where their relationships are determined. A common example of a modelling approach is the SysMod methodology, which offers a pragmatic approach to modelling complex systems on the basis of SysML [4]. Other approaches, such as SPES [1] and MagicGrid [5], provide a clear focus on cross-domain interfaces and relationships of abstract system information. These approaches, including SysML, are widely supported by commercial and open-source MBSE (2024) software solutions, such as the CAMEO Systems Modeler 2024 (CSM) from Dassault Systemes.

Academic Editors: Spiros Pantelakis, Andreas Strohmayer and Gustavo Alonso

Published: 19 May 2026

The general boundary conditions of a potential standardized framework for the methodology of abstract system modelling and its fundamental content are defined by ISO/IEC/IEEE 24641 [6]. Based on the given guidelines, individual modelling approaches can be developed and used as an independent or supplementary method. The user-oriented application of the individual methods can be carried out using a wide range of commercial and open-source MBSE software solutions.

The development process defined in this work focuses on a complete modelling approach to satisfy the development process of aeronautical systems in accordance with SAE ARP4754B [7] as fully traceable abstract system model. The decomposition of complex systemic relationships is carried out using a level- and viewpoint-based approach supported by artificial intelligence (AI). The resulting abstract views of the holistic system are implemented using a specially developed metamodel that is strongly inspired by the basic features of SysML. The direct application takes place via a modelling framework for software-supported **H**ierarchical **A**rchitecture **D**esign for **E**ngineering **S**ystems (HADES). This is designed as a plug-in for the commercial MBSE tool, CSM.

2. The HADES Process

2.1. A Concept of System Decomposing

The methodological basis for HADES is the international standard ISO/IEC/IEEE 15288 [8], which defines the general system life cycle process of technical systems, as well as SAE ARP4754B [7], the development guideline of civil aeronautical systems. In its current stage of development, HADES focuses on the initial system conception and the iterative detailing of the System of Interest (SoI) to be developed. For this purpose, a corresponding approach of continuous hierarchical abstraction is established by inductive elaboration of subordinate system levels with increasing information content.

The resulting system architecture offers a high degree of consistency and traceability of information for the holistic system model. In addition to the hierarchical system structure, HADES works with a targeted categorization, into specific views and viewpoints in accordance with ISO/IEC/IEEE 42010 [9] (pp. 9–12), for advanced structuring of the information. The viewpoints describe which aspect or which particular part of the overall system architecture is focused on. Depending on this, the view for a specific system within the hierarchical structure provides all the systemic information contained in this intersection. A schematic representation of the views and viewpoints for a generic system hierarchy is shown in Figure 1.

The classic RFLP approach is extended by HADES so that the following viewpoints are provided for structuring the information:

- Operations Viewpoint (OpsVP; **O**);
- Requirements Viewpoint (ReqVP; **R**);
- Functional Viewpoint (FctVP; **F**);
- Logical Viewpoint (LogVP; **L**);
- Physical Viewpoint (PhyVP; **P**);
- Safety Viewpoint (SafVP; **S**).

The extended ORFLPS approach of HADES was developed to transfer the process defined in SAE ARP4754B [7] (p. 21) for the development of aeronautical systems into a structured MBSE method. What distinguishes this approach is its comprehensive mapping of the entire ARP4754 process—from requirements definition and system development to system safety assessments—within a consistent and traceable model framework. Operational aspects of the system are taken into account from the outset for each abstracted system level and are adapted individually. In addition, system safety is not treated as a

secondary systemic requirement, but is integrated holistically into the modelling strategy and analysis from the earliest stages as a fundamental point of view.

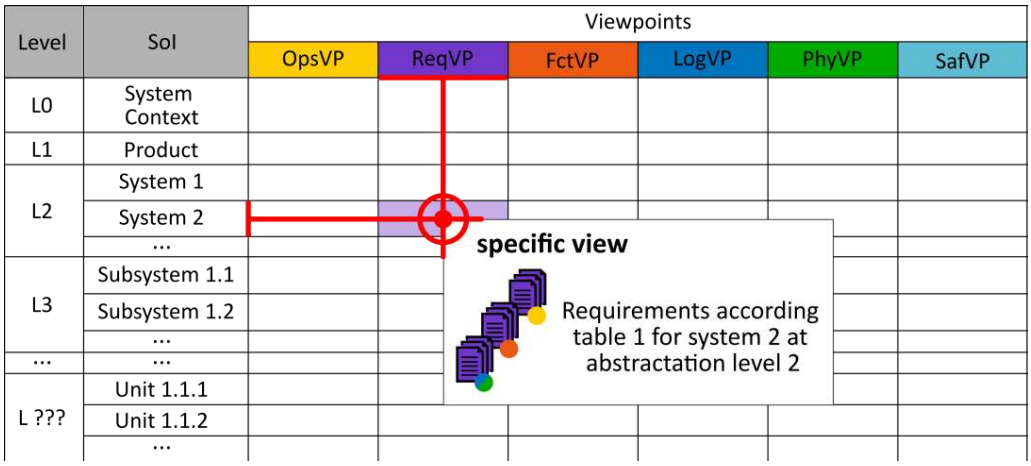

Figure 1. HADES-Matrix to structure the decomposed system information according the hierarchical system structure based on [8] (pp. 11–14) and a viewpoint categorization as per [9] (pp. 9–12).

2.2. Methodology of ORFLPS

To develop the SoI, the ORFLPS viewpoints are passed through sequentially, starting with the OpsVP. The information obtained in the specific view requires an extended scope of information for the surrounding views of the considered systemic hierarchy level. Furthermore, the detailed procedure for the level-specific abstraction of systemic information is defined in detail for each viewpoint and its interrelations are explained.

The **Operations Viewpoint** (OpsVP; O) takes an operational view of the SoI. The starting point is the fundamental expectations of stakeholders regarding the system to be developed. According to [10] (p. 49), these are recorded in the form of needs, which define the basic mission statements. By working out specific expectations of the SoI, the needs are further detailed and several goals are derived. By expanding the goals with quantitative information and entry conditions, specific objectives are defined for the previously unknown system. The relations between the items of initial knowledge acquisition at the corresponding abstraction level (system level) of the SoI are shown in Figure 2.

Based on these initial stakeholder expectations, potential actors of the SoI are worked out. The known stakeholders can usually be defined directly as actors themselves or in abstracted instances. Other actors can usually be found indirectly in the form of operationally relevant environmental influences, externally interacting third-party systems, and their software interfaces or binding regulations. A level-specific system context is defined from the combination of all essential actors.

In addition, specific sets of use cases are derived from the combination of potential stakeholders and the developed goals. The use cases define the expected application of the SoI as a primary system service for one or a combination of several actors. With the help of quantitative information and specific operating conditions (objectives), individual scenarios are derived from each use case, which further represent specific static operating states of the SoI. Finally, the previously defined objectives are converted into specific stakeholder or top-level requirements. For this purpose, a specific system is assigned to the objective and corresponding obligations (e.g., "should/shall" statement) are added.

The **Requirements Viewpoint** (ReqVP; R) is used for the structured storage of all requirements related to the SoI. Each abstraction level or hierarchical link from the overall system (product) to subsystems to component or unit level has its own ReqVP in accordance with the HADES-Matrix depicted in Figure 1. The stakeholder requirements of the SoI, as

well as the system's internal functional interactions and interfaces with the logical or physical subsystems, are categorized into main and sub types based on the recommendations of SAE ARP4754B [7] (pp. 42–44). Table 1 provides an overview of the main and sub types for requirements contained in HADES.

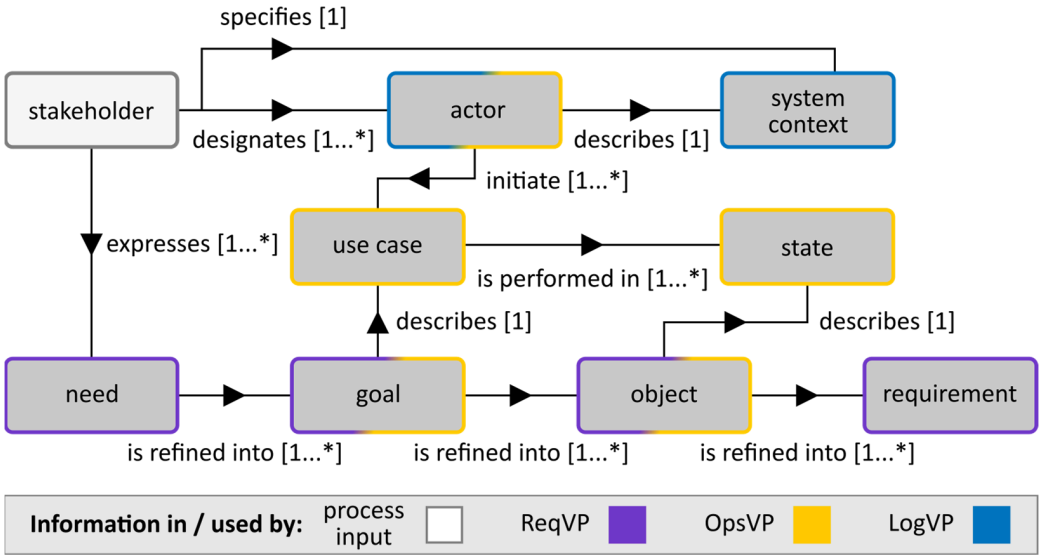

Figure 2. Process deliverables, interactions, and functional relationships of the OpsVP.

Table 1. Requirements categorization of HADES according to SAE ARP4754B [7].

Main Type	Operational	Functional	Physical & Installation
Sub Type	• Maintenance • Customer • Certification • System Safety • Derived	• Performance • Interface • Customer • Certification • System Safety • Derived	• Load & Strength • Design Properties • Interface • Maintenance • Customer • Certification • System Safety • Derived

The information contained in the ReqVP is derived from the analyses and processes of the other VPs or serves as input for them to enable further decomposition of the SoI. To facilitate the allocation and traceability of the requirements with the system-specific information objects of other VPs, HADES provides various tools for targeted allocation of requirements with use cases, functions, and system blocks.

The internal functionality of the system is shown in the **Functional Viewpoint** (FctVP; F). Basic information for the functional analysis is provided by the previously defined use cases of the OpsVP. The information of the FctVP thus describes which internal system functions and processes are required to enable the externally oriented service (use case) of the system for its actors. For a more precise description of the function and its performance, the associated requirements of the ReqVP are used, which must be satisfied by the function to be defined. It is necessary to derive additional requirements from the resulting functional aspects (Table 1) and interrelations of the system model. This ensures a meaningful deduction of the system into a deeper level of abstraction.

The **Logical Viewpoint** (LogVP; L) is used for the logical–abstract description of the SoI. As shown in Figure 3, purely systemic relationships are defined in the form of interface

information between the system objects. The information required for this is provided by the ReqVP and FctVP. Furthermore, from this step in the system modelling, the SoI can be named for the first time and no longer exists as a black box. Based on the information already obtained from the SoI, the functions defined in the FctVP can be allocated directly to a system element. This also enables the indirect satisfaction of the use cases and subdivided operating scenarios of the OpsVP by an associated system within the defined system context. Additional requirements of the type "Physical and Installation" (Table 1) can be derived if necessary.

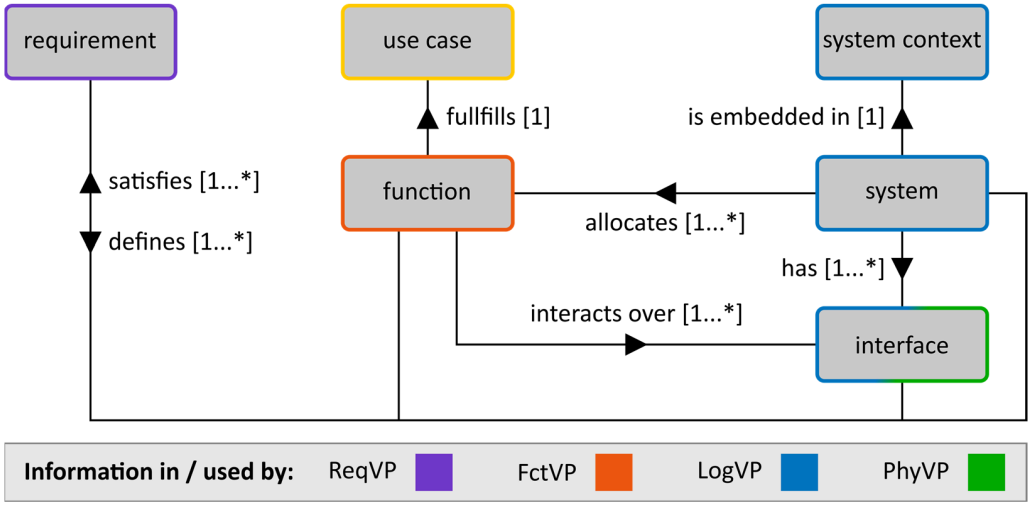

Figure 3. Process deliverables, interactions, and functional relationships of the LogVP.

The **Physical Viewpoint** (PhyVP; P) serves to refine the abstract system model from the purely logical view of the LogVP into a physical, technically realizable solution. The previously defined logical architecture of the SoI thus provides the basis for the development of the physical view. This is supported by additional requirements that describe the topological and morphological shape of the logical system object in more detail and define interfaces by means of precise input and output values.

The **Safety Viewpoint** (SafVP; S) is used to perform an integrated MBSA (model-based safety assessment) approach to analyze the system safety as a result of the model-based system architectures developed in the previous VPs. To enable this, the HADES-extension called OSIRIS (**O**perational **S**afety and **I**ntegrated **RI**sk analysis; [11]) was developed, which can support various safety methods as well as the complex integration and interaction of the ORFLPS-VPs. The basis is formed by the methods for system safety analysis provided by ARP4761A [12] (pp. 18–19) and in combination with the application process in the context of system development in accordance with ARP4754B [7] (p. 32). At the time of publication, the implementation in the MBSE/MBSA environment of HADES and OSIRIS supports the Functional Hazard Assessment (FHA) [13] (pp. 37–47) as described in detail by [11]. This provides information on potential failure cases of the SoI, as well as their effects on various higher-level actors (e.g., aircraft, crew, occupants in case of aeronautical systems). The required input for the FHA is provided by the functions of the FctVP, the operating scenarios of the OpsVP, and the associated requirements of the ReqVP for the respective self-contained abstraction level. Alternatively, HADES can incorporate additional information on the associated system object of the LogVP as well as logical and functional interface information on neighbouring function and system objects into the safety analysis. The inputs are analyzed and individual failure cases of the FHA are created using AI support, as shown in Figure 4.

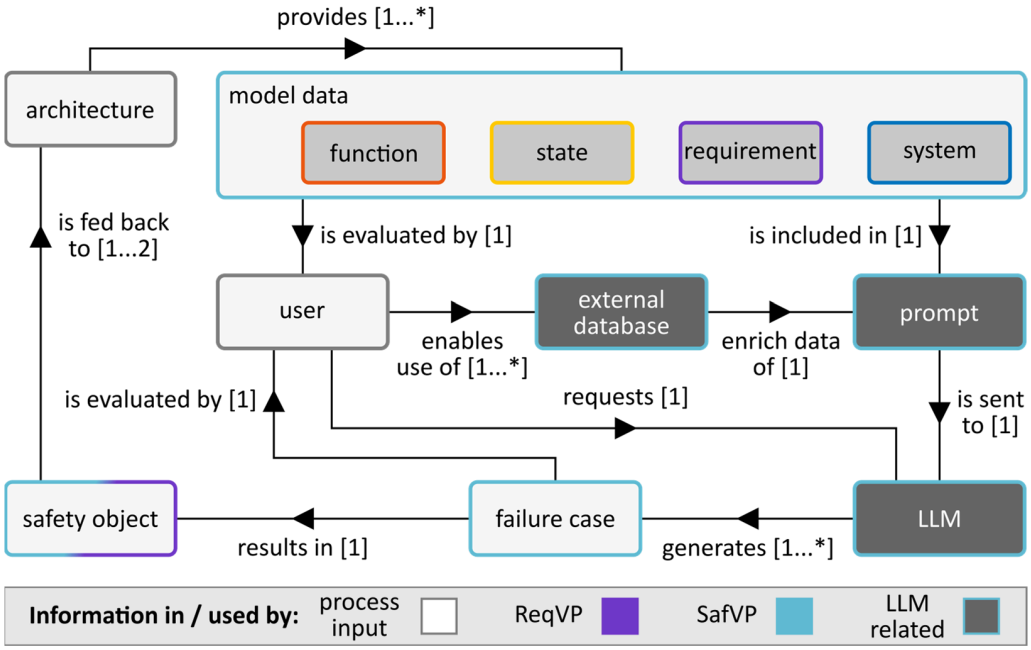

Figure 4. Process deliverables, interactions, and functional relationships of the SafVP including LLM support according to [11].

The data is interpreted by a Large Language Model (LLM) and the results are generated in the form of several potential failure conditions, including their classification in accordance with CS-25.1309 [14] (p. 866). The user has the option to evaluate the generated results, incorporating changes and adopting selected failure conditions as a new safety object in the SafVP of the system model. The linking and traceability of the failure conditions, with the information from the other VPs, is provided.

Finally, corresponding safety requirements for the ReqVP are derived based on the identified safety risks and their criticality. These can influence operational, functional, and logical, as well as physical systemic aspects, as shown in Table 1. As a result, the SoI at this abstraction level is iterated again with the existing information basis in all VPs. This iteration process is continued until the results of the SafVP of the current iteration do not require any further adjustment of the SoI.

3. A Generic System Concept

The SoI is refined to a deeper level of abstraction once the iterative elaboration of all VPs at the associated abstraction level has been completed (Figure 1). For a deeper level of abstraction, the VPs are sequentially worked through again. To redefine the system context of the OpsVP, the logical systems of the higher abstraction level become the actors in the lower system context. The addition of further actors is possible in consultation with the level-specific stakeholders. The use cases of the lower-level result from the functions of the higher abstraction level. This is due to the fact that the previously considered SoI now acts as an actor for its potential subsystem. Previously defined internal functions now serve as a direct service function (use case) of the subsystem to its superordinate system due to the gradation of the system context.

The higher-level SoI function is divided into sub-functions, enabling implementation of the higher-level function. This is defined as a white-box function, with sub-functions as new black-box functions (Figure 5). An analogous procedure can be carried out for subdividing the logical architecture in the LogVP, and an allocation between the new black-box functions and black-box systems, as well as the elaboration of the remaining VPs, is required in accordance with Section 2.1.

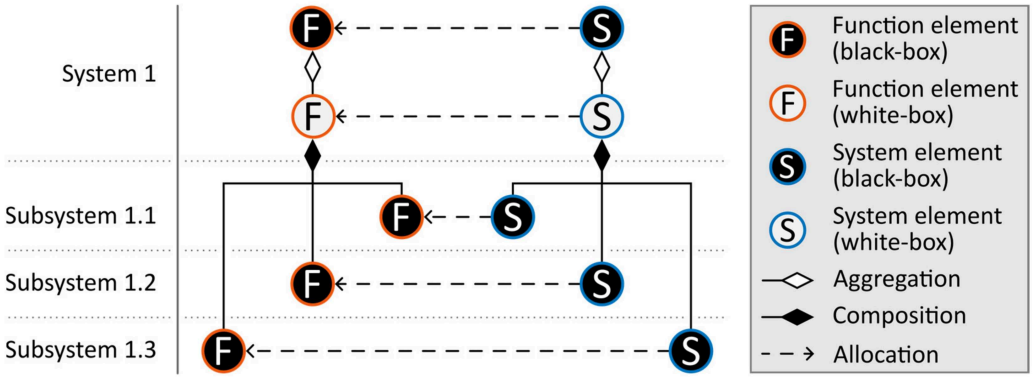

Figure 5. Schematic relationships between the black-/white-box elements of the FctVP and LogVP across two levels of abstraction.

This enables the creation of a database of potential systems, allowing for individual system compositions with specific top-level SoI. The effort required to create a database of generically interchangeable system modules with an integrated VP structure is time-consuming, but offers a considerable advantage for subsequent product developments.

4. Methodological Enhancements and Future Work on HADES

The acquisition of information in the OpsVP is largely based on analyzing and interacting with the stakeholders of the SoI. This process is supported by HADES through corresponding information objects for data structuring, but is not actively guided. A stronger process guidance of the user, which is supported by templates and an adaptive graphical user interface, is conceivable.

The support of system safety analyses by LLMs for text-based data interpretation and result generation is already promising in the current expansion stage of HADES. An extension of the concept for targeted LLM support for architecture decisions in FctVP and LogVP will be integrated in future versions. Various methods of optimizing the knowledge base of the utilized LLM are being sought for reliable and targeted application.

Another major application case of HADES is the simulation capability of the abstract system models, taking into account the information of all stored VPs, as well as a variable scope of abstraction levels. This allows for early-stage dynamic behaviour checks and malfunction detection, reducing risks and substantiating decisions. The integration of simulation capability also extends the scope of methods for (semi-)automatic system safety analyses according to OSIRIS. To realize this, the implementation of state machines based on SysML are planned for later expansion stages, requiring metamodel adjustments.

5. Conclusions

The basics of abstract system modelling with the HADES modelling framework in accordance with common SE standards were presented. HADES focuses on a complete modelling approach to satisfy the development process of aeronautical systems in accordance with ARP4754B. For this purpose, the classic RFLP approach was taken up and expanded to include integrated methods for analyzing the systemic-operational aspects and suitable methods of system safety assessment. The subdivision of the SoI into more detailed subsystems and subfunctions is carried out in line with targeted system safety analyses to validate the abstract system model. Based on the potential system-level structure, HADES provides a generic approach that makes it possible to separate interchangeable system modules with an integrated, modelled VP structure. As a result, complex systems are visualized in a manageable way, responsibilities will be clearly structured and separated, and coherent relationships and interfaces can be explicitly defined.

https://doi.org/10.3390/engproc2026133155

Even in its current expansion stage, HADES has the potential to make development processes more efficient and transparent. The active use of HADES can save time and resources, increase productivity, and improve the resulting quality of the systems.

Author Contributions: Conceptualization, M.M.; methodology, M.M. and D.S.S.K.; software, D.S.S.K. and L.F.M.; validation, M.M., D.S.S.K. and L.F.M.; formal analysis, M.M.; investigation, M.M.; writing—original draft preparation, M.M.; writing—review and editing, M.M. and D.S.S.K.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pohl, K.; Broy, M.; Daembkes, H.; Hönninger, H. (Eds.) *Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology*; Springer International Publishing: Cham, Switzerland, 2016.
2. Wymore, A.W. *Model-Based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design*; CRC Press: Boca Raton, FL, USA, 1993.
3. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language*, 3rd ed.; Morgan Kaufmann: Burlington, MA, USA, 2015.
4. Weilkiens, T. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*, 1st ed.; Morgan Kaufmann OMG Press: Burlington, MA, USA, 2007.
5. Morkevicius, A.; Aleksandraviciene, A. *Magic Grid—Book of Knowledge: A Practical Guide to Systems Modeling Using MagicGrid from Dassault Systèmes*, 2nd ed.; Vitae Litera: Vilnius, Lithuania, 2021.
6. *ISO/IEC/IEEE 24641:2023*; Systems and Software Engineering: Methods and Tools for Model-Based Systems and Software Engineering. ISO: Geneva, Switzerland; IEC: Geneva, Switzerland; IEEE: Piscataway, NJ, USA, 2023.
7. SAE International. *Guidelines for Development of Civil Aircraft and Systems: ARP4754B*; SAE International: Warrendale, PA, USA, 2023.
8. *ISO/IEC/IEEE 15288:2023*; Systems and Software Engineering: System Life Cycle Process. ISO: Geneva, Switzerland; IEC: Geneva, Switzerland; IEEE: Piscataway, NJ, USA, 2023.
9. *ISO/IEC/IEEE 42010:2022*; Software, Systems and Enterprise: Architecture Description. ISO: Geneva, Switzerland; IEC: Geneva, Switzerland; IEEE: Piscataway, NJ, USA, 2022.
10. NASA. *NASA Systems Engineering Handbook*, 2nd ed.; NASA: Washington, DC, USA, 2016.
11. Katabathula, D.S.S.; Mischke, M.; Stoppa, S.; Frank, R.; Lachmund, M. OSIRIS—Generation of system-specific failure cases using artificial intelligence based on information from abstract system models. In Proceedings of the 15th EASN International Conference on "Innovation in Aviation & Space Towards Sustainability Today & Tomorrow", Madrid, Spain, 14–17 October 2025.
12. Society of Automotive Engineers. *Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment: ARP4761A*; SAE International: Warrendale, PA, USA, 2023.
13. Kritzinger, D. *Aircraft System Safety: Assessments for Initial Airworthiness Certification*; Woodhead Publishing: Duxford, UK, 2016.
14. EASA. *CS-25—Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes*; EASA: Cologne, Germany, 2023.