Proceedings of the Annual General Donald R. Keith Memorial Conference
West Point, New York, USA
May 2, 2024
A Regional Conference of the Society for Industrial and Systems Engineering

Notional Air Defense System: A Transition to Model-Based Systems Engineering

Brandon Gibson, Pierce Guthrie, Juan Rodriguez, Michael Trangredi, Mario Treviño, Mark Styles, and James Enos

Department of Systems Engineering,
United States Military Academy,
West Point, New York 10996

Corresponding author's Email: lorenzordz@live.com

Author Note: We are grateful for the support and funding provided by the Systems Engineering Department at the United States Military Academy and our project sponsor, which enabled the completion of this project. For inquiries or requests regarding this publication, please contact Juan Rodriguez at lorenzordz@live.com.

Abstract: This paper explores the transition from document-based systems engineering to Model-Based Systems Engineering (MBSE), with a specific focus on developing methodologies for creating Use Cases, Block Definition Diagrams (BDD), and Internal Block Diagrams (IBD). This research addresses the challenges and opportunities inherent in embracing the movement towards MBSE. Through a combination of theoretical frameworks and practical applications, the paper provides a comprehensive examination of the benefits, implementation strategies, and best practices associated with adopting MBSE principles in modern systems engineering contexts. These principles are vastly applied to a Notional Air Defense System (NADS). As companies try to evolve from document-based approaches, the processes used in this paper will become more relevant with the transition to MBSE. The client found easier traceability to components within the system. This enables ease in identifying errors and costs.

Keywords: Model-Based Systems Engineering (MBSE), Use Case Methodology, Block Definition Diagram (BDD) and Internal Block Diagram (IBD)

1. Introduction

Model-Based Systems Engineering (MBSE) transforms the traditional document-based approach to engineering by employing models as the primary means of understanding. MBSE is a systematic approach to the engineering of systems that uses models to represent various facets of the system life cycle, encompassing requirements, design, analysis, verification, and validation. It represents a modern methodology that enhances communication and collaboration between stakeholders and engineers while facilitating early validation and verification. This results in minimized risks and costs, elevated system quality and reliability, and ease of reuse and maintainability. There are four foundational principles to MBSE: models are the authoritative source of truth, models are executable, models are integrated, and models are reusable. These principles highlight the essence of MBSE and the transformative impact on moder systems engineering practices.

The Notional Air Defense System (NADS) described in this paper is a mobile anti-ballistic missile system designed to acquire, track, and shoot down short, medium, and intermediate-range ballistic missiles through a hit-to-kill approach. The NADS enables Air Defense Artillery (ADA), which is a branch of the military that specializes in protecting assets from aerial attacks, to conduct theater missile defense, offering a layer of defense for strategic assets and regional security. This enables greater efficiency within multi-domain assets because the NADS offers early warning indication and protects critical infrastructure, increasing battle space awareness and decision making within multi-domain assets.

The paper outlines the application of MBSE principles to a notional air defense system as part of the larger ADA system of systems. The introduction provides an overview of MBSE, describing the significance of the modern engineering practices. The literature review explores the transition from document-centric to model-centric engineering, emphasizing the benefits of MBSE. The methodology and results section explains the systematic breakdown of the NADS, detailing its physical and logical architecture using SysML. This offers the foundation for future research in this advancing field of engineering.

2. Literature Review

The transition from document-centric to model-centric engineering is being integrated slowly in engineering. Documents are being replaced as the primary product for system engineering processes by these models. MBSE is the application of modeling to support system requirements, analysis, verification and validation, and life cycle phases of a system (Friedenthal et al., 2009). It encompasses everything from the functional, performance, structural, and other analysis models. The process for MBSE starts with the beginning phases of problem introduction, continues with testing and validation, and ends with upgrades and retirement of the system. Digital tools in MBSE have made it easier and faster to generate, maintain, and utilize models in a collaborative setting ("Model-Based Systems Engineering (MBSE) - SEBoK"). This evolution is affecting areas outside of traditional systems engineering by allowing earlier identification of issues through the simulation of the model. MBSE, and the shift towards interoperability of digital tools, enables a centralized dimension to design, facilitating adaptation of products to meet customers' evolving needs, whether the product is in the design phase or deployed for years.

SysML categorizes systems into three main types: physical architecture, logical architecture, and system requirements. The physical architecture, depicted through block definition diagrams (BDDs), provides a structural representation of the system. Each block within the diagram is decomposed into subsystems and parts, which are further detailed in internal block diagrams (IBDs) to illustrate interconnections and behaviors (Fernandez & Hernandez, 2019). Interfaces are the points of interaction, full ports and proxy ports, where the behavioral information is then associated with the physical components. Logical Architecture defines the system's functionalities and interactions by modeling the connection between the physical components. This creates a clear connection between information flows through components.

2.1 Physical

BDDs display very detailed information on the system. They define blocks, actors, value, types, constraint blocks, flow specifications, and interfaces (Delligatti, 2013). The relationships between associations, generalizations, and dependencies are important elements in BDDs. These definitions convey the decomposition and classification of the blocks (Delligatti, 2013). BDDs encompass various types of diagrams such as package, model, modelLibrary, view, block, and constraintBlock. These diagrams are organized within namespaces, typically represented by packages, which contain elements appearing in BDDs. Blocks, fundamental units in SysML, encapsulate structural and behavioral features. Optional compartments in blocks include parts, references, values, constraints, operations, receptions, and various types of ports to include full ports and proxy ports (Delligatti, 2013).

Structural features, or properties, within blocks can represent part properties, reference properties, value properties, and constraint properties. The part properties represent the structure that is internal to that specific block. Values are used to provide detail such as weight and units (Delligatti, 2013). Multiplicity is an important aspect as it contains the number of a certain part a system or block has. Reference properties represent a structure that is external to the block. It does not describe ownership, rather is describes a "needs" relationship. Value properties are listed under the compartments and represent a quantity. These values are parameters of the specific performance or structural design. The constraint properties are represented by mathematical relationships that are imposed on a set of value properties like memory sufficiency (Delligatti, 2013).

2.2 Interfaces

The primary way that modeling tools utilizing SysML to depict interfaces is through the use of ports. A port is a kind of property that represents a distinct interaction point at the boundary of a structure through which external entities can interact with that structure – either to provide or request a service or to exchange matter, energy, or data. (Delligatti, 2013). SysML identifies two kinds of ports, one that exposes features of the owning block or its internal parts (proxy ports), and another that supports its own features (full ports) (OMG, 2018).

A full port is equivalent to a part on the boundary of the parent block that is made available as an access point to and from the block. A full port is typed by a block and can have nested parts and behaviors, and can modify incoming and outgoing flows like any other part. A full port can represent a physical part such as an electrical connector or a mechanical interface assembly, and therefore is a part in the system parts tree. The other kind of port is a proxy port. By contrast, a proxy port does not constitute a part of its parent block, but instead provides external access to and from the features of its parent block or the block's parts without modifying its inputs or outputs. A proxy port is typed by an interface block that specifies the features that can be accessed via the port (Friedenthal et al., 2015). Interfaces can also simply be depicted using an interface block. Interface blocks are a specialized form of block that does not contain any internal structure or behavior (Friedenthal et al., 2015). A single interface block, using a conjugate port, can be reused rather than creating two separate specifications for the proxy ports on interacting blocks (Friedenthal et al., 2015).

2.3 Logical

Logical architecture in MBSE is a solution-independent model of the problem domain. It focuses on understanding what needs to be done to achieve the desired system behavior, without specifying how it will be implemented. It defines the essential functions and interactions of the system without being bound by specific technologies of physical components (Hause et al., 2022). Logical architecture and use cases are two essential concepts in MBSE. Logical architecture provides a high-level view of a system's components and how they interact, while use cases describe how actors use the system to achieve specific goals. By modeling both logical architecture and use cases, engineers can gain a deeper understanding of the system and design it to meet the needs of its users (Shevchenko, 2020).

Activity diagrams, also known as behavior diagrams, depict dynamic aspects of systems by illustrating the flow between activities. Components include nodes such as action, object, and control nodes, with control nodes further categorized into fork, join, decision, merge, initial, activity final, and flow final nodes (Jarraya et al., 2009). Decision nodes bifurcate activities based on Boolean conditions, while fork and join nodes manage parallel and sequential flows (Jarraya et al., 2009).

Logical architecture defines system functionalities and interactions abstractly, independent of specific solutions. Use cases capture system requirements through abstract concepts, which can later be mapped to technologies and physical elements. Actors, system boundaries, and use case relationships delineate user interactions and system behavior (Delligatti, 2013). Use cases are elaborated with context diagrams, sequence diagrams, activity diagrams, and state machine diagrams to specify system functions and behaviors (Friedenthal et al., 2015). These diagrams help translate abstract requirements into physical components, driving the design of the system's physical architecture.

3. Methodology and Results

The client had paper engineering documents for the existing system. In the modeling process of the legacy system, the sequence of architectural representation deviated from the conventional logical to physical construction. Instead, this initial focus was on the physical architecture due to the availability of the documentation of components and connections. By delineating the system's components, interfaces, and relationships, the report facilitates a comprehensive understanding of how the system functions and how its various elements interoperate. This breakdown enables engineers to analyze and optimize both the physical infrastructure and the underlying logic, ensuring that the system design aligns with its intended functionality and performance requirements. Through the integration of SysML and a structured approach to architectural decomposition, this methodology offers a robust foundation for effective system development and refinement. Furthermore, by representing the system in both physical and logical architectures, the report provides a holistic view that accounts for both the tangible elements of the system, such as hardware components and interfaces, as well as the abstract concepts, such as data flows and control logic. This representation enables engineers to capture and analyze the system's behavior from multiple perspectives, facilitating more accurate modeling, simulation, and validation of system designs.

3.1. Physical

In delineating the physical architecture of the NADS, this report employs block definition diagrams within the SysML framework to systematically represent its parts. Specifically, the focus centers on the NADS Fire Control Center (FCC), a critical element of the system's architecture. Within this center, several key subsystems are identified, each serving distinct functions crucial to the overall operation of the system. These components include the Shelter, Network, Data Processing, and Voice Sub-Systems. By defining these components and internal components through block definition diagrams, the paper provides a clear and structured representation of the NADS Fire Control Center's physical architecture. Figure 1 presents the hierarchical depiction of the system with the associated subsystems, i.e. Part 1.1.1 Shelter Sub-System. This allowed for the reuse of parts in other parts of the model. This integration of parts is important for the component level of the system.

ISBN: 97819384962-4-0

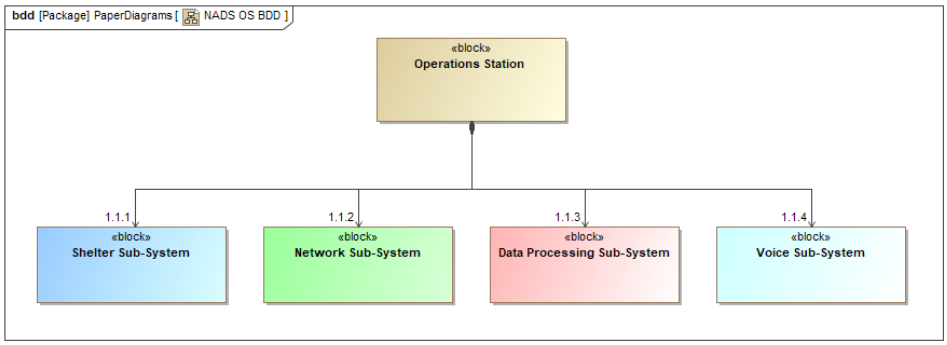

Figure 1: NADS FCC BDD

3.2. Interfaces and Message Interactions

In developing the IBDs using SysML for the NADS, meticulous attention is paid to the intricate interactions among its various components. Employing SysML's constructs such as full and proxy ports, the IBDs comprehensively depict the flow of information, signals, and control commands within and between system elements. Each component of the NADS, including the Fire Control Center, Launcher, Radar and Interceptor is represented as internal blocks interconnected via proxy ports, illustrating their direct communication pathways. These ports enable seamless data exchange, facilitating real-time monitoring, decision-making, and coordination functionalities crucial for the system's operation. Specifically, the Out Launcher Interceptor port serves as an interface for the NADs Launcher and the NADs Interceptor as messages are being passed between the launcher and interceptor.

Moreover, SysML provides mechanisms for translating messages and data between system components, a critical aspect in ensuring seamless communication and interoperability. IBDs serve as a pivotal tool for representing the internal structure of system components, including their ports through which messages and data are exchanged. Within the NADS, distinct IBDs are employed to model the internal architecture and interactions of subsystems such as the Operation Station (OS). By delineating the interfaces and connectivity between these components, SysML facilitates the seamless flow of information, enabling efficient coordination and collaboration among subsystems. This is especially useful when creating different configurations of the model and identifying which messages are necessary to complete a certain activity, as seen in Figure 2.

ISBN: 97819384962-4-0

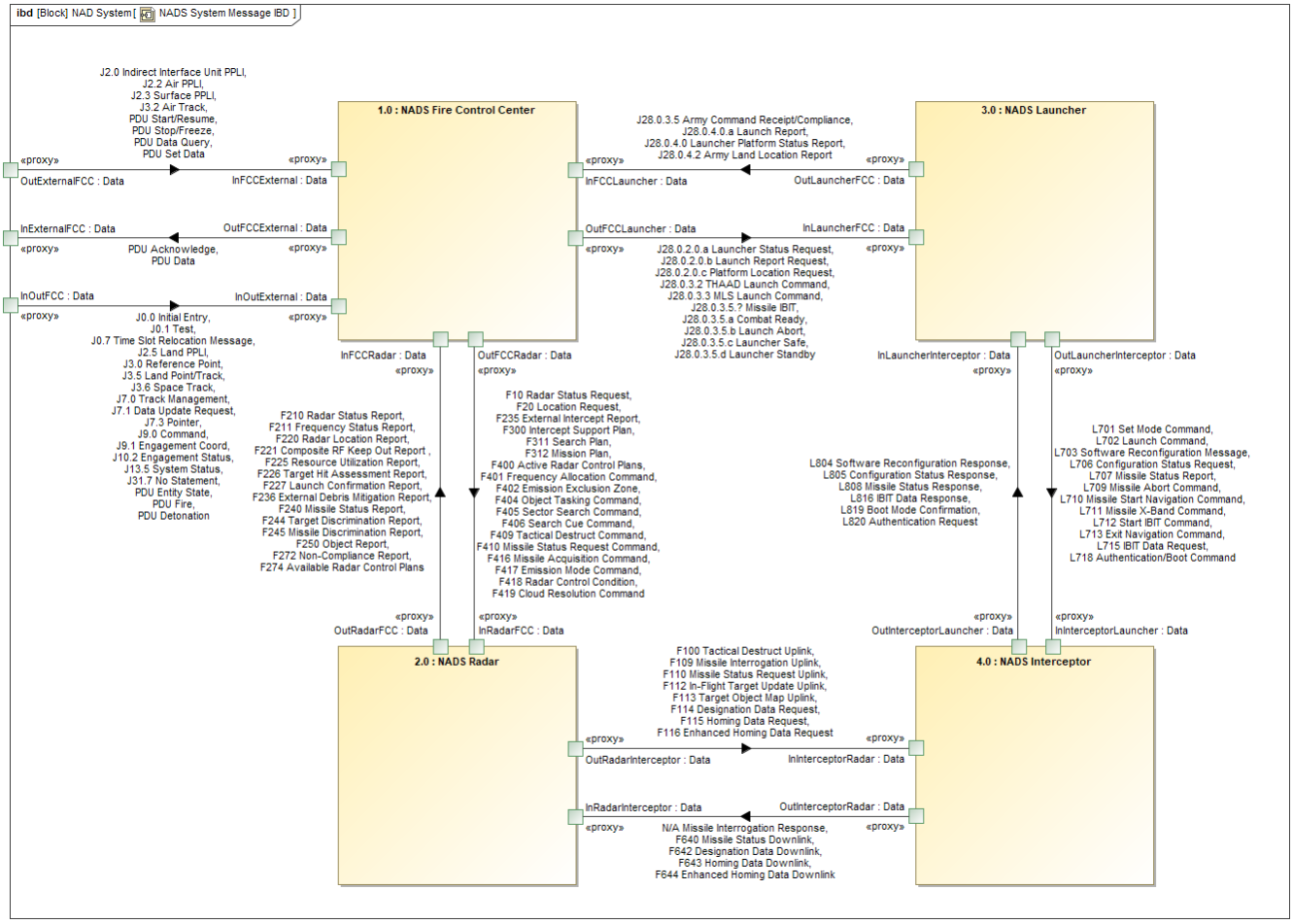

Figure 2: Messages and Connections within the NADS

Within the IBD of the OS, various components such as mice, headsets, monitors, and keyboards are interconnected through the Zero Client Clear Tube. The interfaces with the Zero Client Clear Tube serves as a hub for data processing within the OS. These parts, situated withing the OS Data Processing Sub-System, acts as a channel for transferring and receiving data from the other subsystems in the OS. This is established by a link to the Network Switch in the OS Network Subsystem. Adaptors are employed within the system to facilitate compatibility of differing wires and interfaces, such as RJ45 and DB9, shown in Figure 4. This network of interconnected components enhances efficiency of the system.

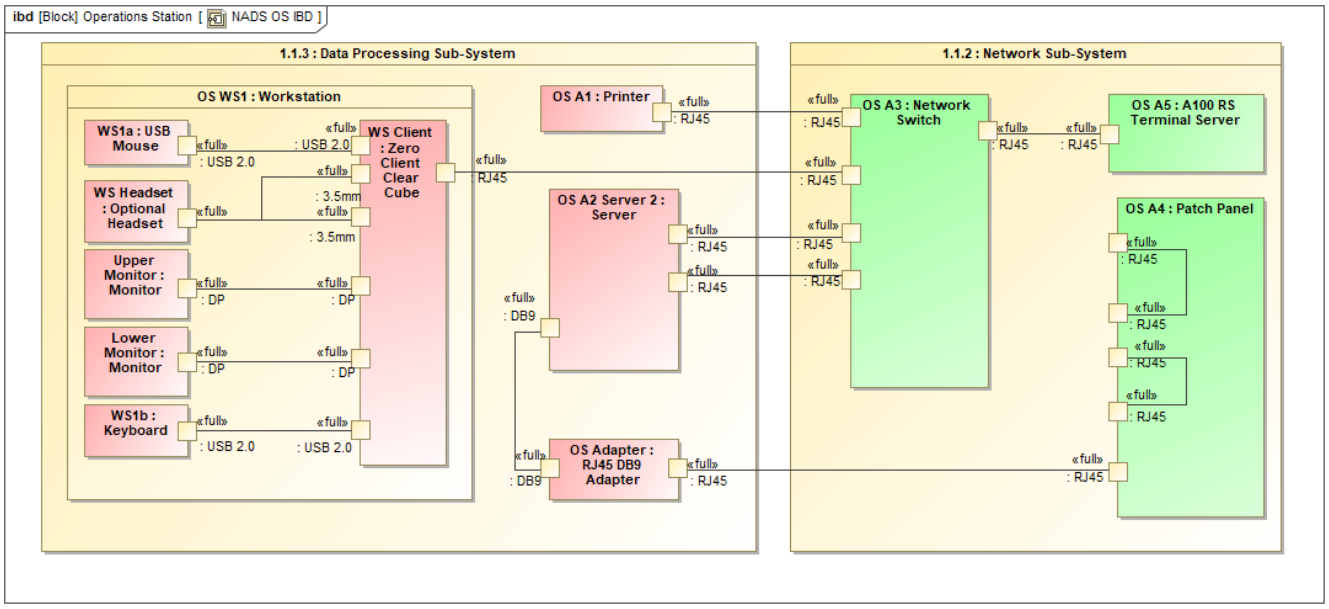

Figure 3: Operation Station IBD

The OS depicts RJ45, DB9, DP, and USB 2.0 ports and connectors. They are commonly used to connect one internet-enabled device to another network. These are depicted as full ports within the IBD, displaying the interaction between the various subsystems that consist of internal and external connections with other subsystems within the model, as seen in Figure 4. The RJ45 port, known as the Ethernet port, is utilized for connecting devices to local area networks or the internet. This port enables high-speed data transfer and connectivity, allowing the OS to communicate with other devices such as servers and workstations. The DB9 port, used for serial communication, provides the means for connecting serial devices and modems to the OS. This enables the transfer of data in a serial format used for industrial applications. It has a wide range of compatibility typically used for adaptors. The DP, or display port, serves to connect monitors and display devices to the OS. These allow for video output within the OS sub-system. USB 2.0 ports and connectors are used as an interface for connecting keyboards and mice. This port provides functionality of plug-and-play. By labeling the interfaces between components in the OS architecture, it highlights the importance of seamless communication and flow of data.

3.3 Logical

The NADS block depicts the environment in which the use cases operate within. The high-level view of the system's primary functionality and process is captured with a use case, as seen in Figure 5. The diagram includes five major use cases for the NADS: Employment, Engagement Operations, Maintenance and Operations, Force Operations, and Training. This enables enhanced communication and understanding among stakeholders by providing a shared language for discussions on the NADS requirements and functionalities. Additionally, individual use cases provide a basis for Activity Diagrams. Figure 6 presents an activity diagram based on the Engagement Operations Use Case.

Activity diagrams, on the other hand, delve deeper into the internal workings of the system, detailing the sequence of actions, decision points, and control flows involved in executing each use case. By visualizing these processes, Activity diagrams facilitate the identification of potential bottlenecks, inefficiencies, or conflicts within the system, enabling engineers to refine and optimize its design. Figure 6 presents the NADS engagement operations as an activity diagram, which helps visualize the sequence and process of the system from being launched to intercepting the missile. This includes several sub-activities that include acquisition, track management, threat assessment, battlespace, engagement planning and scheduling, conduct engagement, monitor engagement, and post engagement processing. Through this diagram, the logical architecture is captured in the block itself. It helps solidify the NADS requirements that are captured with the use cases and turns them into specific functions.

ISBN: 97819384962-4-0

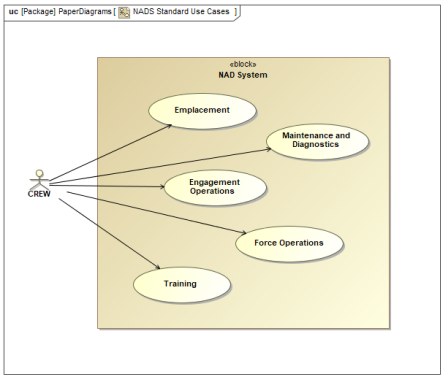

Figure 4: Standard Use Cases

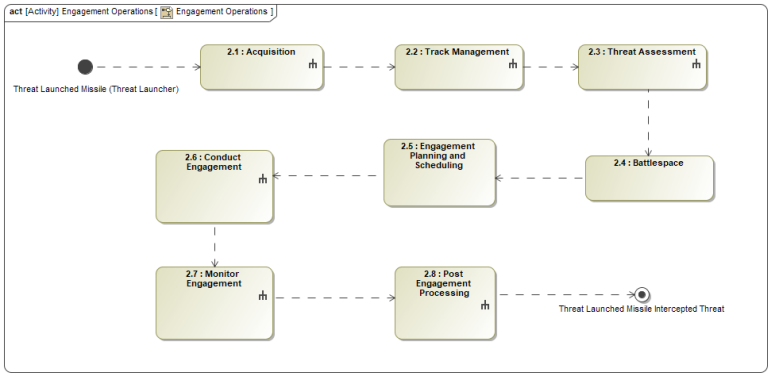

Figure 5: Activity Diagram for NADS Engagement Operations

5. Conclusion

The process of modeling the legacy system was unconventional compared to traditional MBSE modeling techniques as the sequence of architectural representation began with the physical architecture and then moved to the logical architecture. Traditionally, logical architecture is modeled first followed by the physical architecture; since the document-based components and connections were first provided, this unconventional MBSE modeling technique was required. A decision was made to define blocks for types of components used in multiple locations in the model. The model then uses these standard components as parts in the various subsystems, which enables IBDs to depict connections between the various parts. Additionally, this allows engineers to define specific variations of these parts without the need to change the structure and connection in the model as parts are upgraded. This reduces the number of defined blocks in the model and allows stakeholders to see where parts are used across the entire NADS model.

For future work, allocating physical components to the activity diagram through swim lanes will enable a comprehensive understanding and implementation of the system's intended functionality through coherent integration between the conceptual design and the concrete physical aspect (Delligatti, 2013). Furthermore, initiating development using different configurations enables development of the updated variants of NADS, along with the exploration of a stationary variation. These variations will drive a need for configuration exploration through a value-benefit analysis in order to tailor it to the stakeholder's strategic objectives and mission requirements.

5. References

Delligatti, L. (2013). SysML Distilled: A Brief Guide to the Systems Modeling Language. Addison-Wesley.

Friedenthal, S., Griego, R., & Sampson, M. (2009). INCOSE Model Based Systems Engineering (MBSE) Initiative. INCOSE.

Fernandez, J. L., & Hernandez, C. (2019). Practical model-based systems engineering. Artech House.

Friedenthal, S., Moore, A., & Steiner, R. (2015). A Practical Guide to SysML: The Systems Modeling Language. Elsevier Inc

Hause, Matthew, and Lars-Olof Kihlström. "You Can't Touch This! - Logical Architectures in MBSE and the UAF." INCOSE International Symposium, vol. 32, no. 1, 1 July 2022, pp. 434–452, https://doi.org/10.1002/iis2.12941. Accessed 5 Dec. 2023.

Jarraya, Y., Debbabi, M., & Bentahar, J. (2009). On the Meaning of SysML Activity Diagrams. San Francisco: IEEE.

"Model-Based Systems Engineering (MBSE) - SEBoK." Sebokwiki.org, sebokwiki.org/wiki/Model-Based_Systems_Engineering_(MBSE).

OMG Systems Modeling Language. (2018). OMG Systems Modeling Language (OMG SysML) Version 1.6. Retrieved from https://www.omg.org/spec/SysML/1.6/Beta1/PDF

Shevchenko, Nataliya. "An Introduction to Model-Based Systems Engineering (MBSE)." Carnegie Mellon University, Software Engineering Institute's Insights (blog). Carnegie Mellon's Software Engineering Institute, December 21, 2020. An Introduction to Model-Based Systems Engineering (MBSE) (cmu.edu)