Proceedings of the Annual General Donald R. Keith Memorial Conference
West Point, New York, USA
April 24, 2025
A Regional Conference of the Society for Industrial and Systems Engineering

Modernizing Air Defense: Applying Model-Based Systems Engineering to the Notional Air Defense System

Alec Vaccaro, David Adarkwah, Brooke Tyquiengco, Richard Mcdermott, Vasileios Katsigiannis, and James Enos

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

Corresponding author's Email: alecvac09@gmail.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 James Enos at james.enos@westpoint.edu.

Abstract: This paper examines the transition from document-based systems engineering to Model-Based Systems Engineering (MBSE) in the context of application to the Notional Air Defense System (NADS). The development of both structural and behavioral architecture explores how MBSE can effectively display the relationships between different components of a system, develop a sense of architectural traceability, and model system behaviors. This research addresses the practicality, benefits, and potential uses of this form of modeling in the scope of modern engineering. The application of these principles to the NADS demonstrates both the effectiveness and potential limitations of MBSE in the context of providing theatre-level air defense.

Keywords: Model-Based Systems Engineering (MBSE), Notional Air Defense System (NADS), Block Definition Diagram (BDD), Internal Block Diagram (IBD), Activity Diagram, Sequence Diagram, State Machine Diagram, and Systems Modeling Language (SysML)

1. Introduction

Model-Based Systems Engineering is an approach growing in popularity that many systems engineers are finding effective when managing large, complex systems. The transition from a document-based approach to model-focused designs allows engineers to better articulate the traceability and dynamic aspects of systems architecture. Using modeling techniques improves communication during and after the design process between each part of the system across its entire lifecycle. MBSE enables quick prototyping and system simulation. The dynamic properties of Model-Based Systems Engineering allow systems engineers to easily manipulate models. This adaptability allows engineers to meet their stakeholders' wants and needs, while also creating a visualization that increases the probability of foresing potential issues and design flaws before they arise.

The Notional Air Defense System (NADS) is an Air Defense Artillery (ADA) asset that acts as a defensive capability against short, medium, and intermediate-range ballistic missile threats. Air Defense Artillery holds the responsibility of protecting military forces, allies, and assets from aerial attack, missile attack, and surveillance. The NADS can intercept targets both within and outside the atmosphere, as well as provide early warning and detection necessary to maintain successful theatre-level air defense when integrated with other ADA systems. This paper explores Model-Based Systems Engineering as it pertains to the NADS. The introduction summarizes the current relevance and potential of MBSE, as well as introduces its applicability to the Notional Air Defense System. The literature review describes the different kinds of modeling architecture currently used in the field and highlights their advantages over a document-based approach. The methodology and results section explore how MBSE was applied to both the physical and behavioral architecture of NADS. Overall, this research demonstrates the effectiveness and potential of this emerging domain of engineering with a focus on its pertinency to air defense.

2. Literature Review

Model-Based Systems Engineering is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout the system's later life cycle (Baker, 2022). The most used language in the realm of MBSE is the Systems Modeling Language (SysML). SysML consists of five components, one being diagram-types, which are "used to model different concepts, stories, and architectures" (Baker, 2022). Within this component are nine different types of diagrams, each serving distinct roles in creating an overall graphical model of the system. MBSE diagrams are designed to either contribute to the structural or behavioral

architecture of a system. Structural architecture helps to define the overall structure of a system's design. Behavioral architecture speaks to how the structure will behave under different conditions. Both types of architecture complement each other to give engineers a holistic understanding of the system both before and after a physical product is constructed.

2.1 Structural Architecture

Structural Architecture allows systems engineers to exhibit information about the structure of a system by displaying various model elements and the relationships between them. In MBSE, the two main diagrams used to convey a system's structural architecture are Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD). Block Definition Diagrams provide a visual representation of the composition hierarchy of a system or component (Friedenthal, 2015). The most basic unit of structure on a BDD is a block. Blocks can represent any level of the system hierarchy. The relationships between the blocks in a BDD are crucial to the system of interest. Three types of relationships can exist between blocks: associations, generalizations, and dependencies (Delligatti, 2014). Associations assist system engineers in modeling relationships between blocks. They serve as an alternative notion to convey the different kinds of structural relationships within a system (Delligatti, 2014). There are two types of associations that systems engineers create between blocks on BDDs, reference associations and composite associations.

Internal Block Diagrams (IBD) are used to specify the internal structure of a single block and act as static view of the system or one of its parts (Delligatti, 2014). Rather than displaying a block, an IBD displays the usage of blocks. IBDs are useful when the system engineer wants a more in-depth view of the BDD they have created. They focus on interaction by utilizing connectors and ports rather than associations to represent connections between system elements. These connectors indicate how blocks communicate and ensure mutual accessibility within the system. The types of connections established through ports in IBDs often specify the type of matter, energy, or data that flows between components, as well as the services each component provides or requires (White, 2019). This connectivity highlights the functional interactions of a system and how different structures collaborate to fulfill operational goals.

A BDD allows the user to define a block and its properties, while an IBD allows them to display a valid configuration of that block and a specific set of connections among its properties (Delligatti, 2014). Both diagrams are essential to structural architecture and are often created simultaneously at different points for the system's life cycle.

2.2 Messages and Signals

A message represents either an invocation or request for service from a sending lifeline to a receiving lifeline. Messages are shown on sequence diagrams as a line with different arrow heads and annotations depending on the kind of message. In Model-Based Systems Engineering, messages are primarily essential for communication and interactions between different components and subsystems within the overall system. Messages in MBSE carry important information such as data packets, commands, and signals. Messages also have the responsibility of defining interactions in system diagrams such as sequence diagrams and activity diagrams.

A signal is a function of one or more variables that indicates some phenomenon. Signal servers carry information between communications. Unlike messages, signals represent data flow and are often sent as the action of a state (Pericherla, 2013). Signals can also represent a notification or event. These representations are used to trigger specific actions or interactions between system components. Signals are one-way communications that convey an occurrence. Unlike the broader concept of messages, signals do not have responsibility for delivering data, instructions, or responses. A signal represents a given event by notifying a system that a certain change or action needs attention. Signals tend to be present in sequence, activity, and/or state machine diagrams, where they capture and document how sub-systems react to events.

2.3 Activity Diagrams

Activity diagrams, also known as behavior diagrams, model three types of behaviors that are separated into three categories: Functional, Operational, and System Behavior. Functional Behavior is described as the way in which a "thing or process" functions or operates (Baker, 2022). Functional behavior is used to define the functionality of a system using system threads (Baker, 2022). Operational Behavior is derived from mission and mission threads. These behaviors take in stakeholder needs, goals, and missions as well as performance structure and interfaces to generate user requirements. Operational domain behavior originates with user stakeholder needs, goals, and missions by taking the individual aspects of a system that need to work. These considerations take a big picture idea and break it down into many small aspects that must be met for the big picture to work.

System behavior is derived from defining system functions to satisfy previously derived System Requirements. The user requirements help define System Structure, Interfaces and System Data to help ensure accurate System Requirements are

created. System Requirements trace back to high level user requirements (Baker, 2022). The role of Activity Diagrams fits into the SysML language with three other diagrams: Use Case Diagrams, Sequence Diagrams, and State Machine Diagrams. All these diagrams interact together to provide a complete picture and understanding of how the system works. The combination of these diagrams in SysML will provide a model of where the system is currently at and where it will end up. SysML Activity Diagrams are used to define the behavior of a system by using information such as time ordered sequence of actions, and specified control flow logic to model activity flow-based behavior. Activity Diagrams share similarities with functional flow block definition diagrams as they both can be used to model system behavior.

2.4 Sequence Diagrams

The purpose of a sequence diagram is to present a dynamic view of a system that expresses the sequencing of behaviors and event occurrences over time (Delligatti, 2014). This dynamic view captures the flow of information and control within a system, ensuring that engineers can trace the paths of interactions with precision. It provides an easy articulation of how each part of the system interacts with other parts via operation calls and asynchronous signals (Delligatti, 2014). This behavior between parts at the most fundamental level is called an interaction. Sequence diagrams convey three requisite pieces of information: the order in which behaviors are performed, which structure performs each behavior, and which structure invokes each behavior (Delligatti, 2014). This detailed conveyance is particularly important in complex systems, where understanding the sequence of events can help avoid errors in implementation.

Additionally, sequence events are particularly useful in capturing detailed interactions in complex systems like the NADS launcher (Visual Paradigm, n.d.). Since the process of launching a missile is so precise, Systems Engineers tend to use sequence diagrams as a graphical test case specification that displays only a single scenario of a larger use case behavior. By focusing on specific interactions, these diagrams allow engineers to test and validate system behaviors in a controlled manner. Sequence Diagrams are similar to behavior and activity diagrams in how they present a dynamic view of the system. These diagrams are essential for visualizing interactions among various components, allowing for a clearer understanding of the system's behavior (Hause, 2006). This visualization is crucial, as it enables engineers to foresee potential issues and design flaws before they occur.

2.5 State Machine Diagrams

A state machine diagram “represents the states of a structure element and how it transitions between states over its lifetime” (Baker, 2022). A state “represents a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event” (SysML.org, n.d.). Properly illustrating and understanding these diagrams allow for the model to communicate dynamic factors of a system, conveying a more accurate representation of what is being modeled, as well as the functions that the system undergoes throughout its lifecycle. State machine diagrams are useful in modeling real-time and event-driven behavior due to both of their dynamic natures (IBM, 2023).

States, transitions, and psudo-states are all possible components of a state machine Diagram. These elements are what allow for the most complex behaviors and system transitions to be visualized and modeled. Each symbol belonging to state machine diagrams have sub-levels allowing for significant detail to be achieved in the modeling process. Transitions are what indicate within the model that a state change is taking place. Transitions can be made up of triggers, guards, and effects. Triggers are the causes of a transition occurring and get classified as one of four types of events: signal, time, change, and call. As model-based systems engineering has grown in popularity across a variety of industries, new applications of state machine diagrams are appearing every day. Whether it be managing protocols and signal flows in telecommunication networks, modeling aerospace and defense assets, or visualizing user interactions and workflows in software, there is no shortage of real-world relevance.

3. Methodology and Results

When applying the MBSE concepts explored within the literature review to the Notional Air Defense System, the Systems Modeling Language (SysML) acted as the primary instrument for developing design architecture. The results of these modeling efforts produced highly detailed physical and behavioral framework for the NADS, which can be easily navigated and decomposed. This methodology provides engineers with a sense of traceability, visualizing a hierarchical breakdown of the infrastructural aspects of the system, while creating greater flexibility in identifying and correcting design flaws.

In addition to having greater insights into the system’s physical architecture, the behavioral modeling products cultivate a logical connection between how the subsystems of the NADS communicate between itself and the operator through

messages and signals. The overall results create a comprehensive view of the system, allowing for engineers to conduct analyses and simulations at different points in the system's lifecycle. These capabilities further enable engineers to be more accurate and efficient when verifying and validating the system. The most detailed modeling efforts in this research explored the behavioral state transition process when firing a missile based on commands from the Terminal Fire Control Center (TFCC).

3.1 Model Organization

The primary method of organizing the model was a package diagram. A package diagram allows for easy visual understanding of model arrangement for both conceptual and navigational purposes. Figure 1 displays the package diagram for the NADS architecture and contains both the top-level and sub-packages for the system. Physical architecture acted as the top-level package for the system's launcher, with structure, interfaces, flow elements, parametrics, and behavior being its sub-packages for more detailed aspects of the model. The structure package contains all the structural architecture. Interfaces contain how subsystems and components of the NADS interact with each other. Flow elements specify the flow of functions and decisions conducted in the behavioral architecture of the system. Parametrics offers a collection of the incorporations of mathematical rules which created specific constraints within the model. The behavior sub-package contains all the behavioral architecture created during modeling efforts.

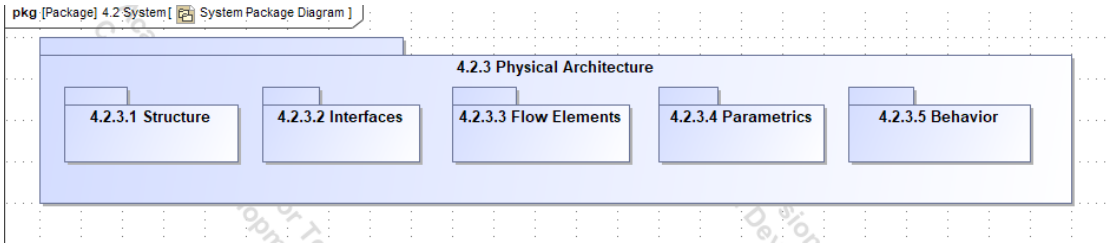

Figure 1. NADS Package Diagram

3.2 Physical Architecture

Physical modeling efforts were highly focused on creating an overall BDD for the NADS and developing detailed architecture on the Carrier Electronics Model (CEM) subsystem. The CEM is a main subsystem of the overall system which primarily deals with interfacing the electronic signals and communications to the missile round pallet. This integration allows for the system to perform its overall function of launching a missile to intercept a projectile. Figure 2 displays the BDD for the CEM, visualizing the hierarchical relationship between the subsystem and its components. The CEM is broken into six blocks each representing a component that contributes to its function based on the context given from previous document-based designs. The Ancillary Hardware component represents the hardware required for the launcher to perform its operation. The Launcher Computer Control Unit has the responsibility of central control. This component receives commands from the operator and data from other subfunctions. The Power component supplies and regulates power to the system. The Communications and Navigation component manages communication data and interfaces with navigation technology. The Missile Interface Component facilitates communication between launcher and missile. The Operator Interface acts as the human systems integration component of the launcher. This part of the subsystem models the connection between the operator and the launch process.

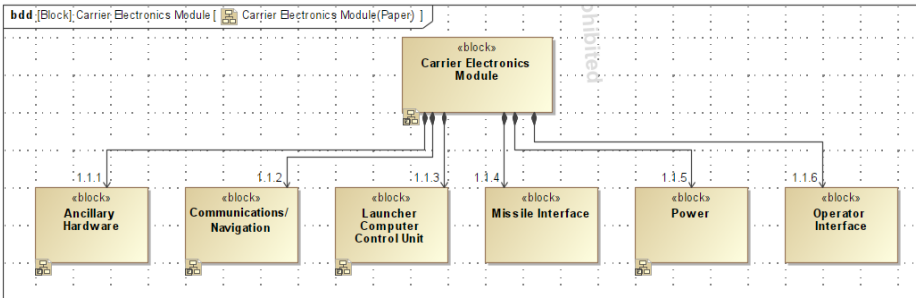

Figure 2. NADS Carrier Electronics Model Block Definition Diagram

3.3 Behavioral Architecture

The aim of modeling behavioral architecture for NADS was to create an executable representation of the process the system undergoes before, during and after performing its main function of launching a missile. Six behavioral states and five signals initiated by the TFCC were identified using previously documented designs. Figure 3 displays these pieces of architecture organized into a state-machine diagram to accurately convey the NADS's behavior being modeled. Both these states and functions demonstrate in further detail the procedure taken by the operator when preparing to fire the air defense system. "Out of Action" is the initial and end state. In this state, the power subsystem is shut down and all other subsystems are deactivated. During "Pending Standby" the system is powered but not fully activated, standing ready for further transition toward combat readiness pending the operator's commands.

If the operator signals "safe" the system will move into its "Safe" state, where the launcher is fully disarmed, and prepared to be completely powered down to go out of action. Without a prompt to move the launcher into "Safe", the system transitions to "Standby". During this state, the subsystems are now fully activated, and the missile interface is ready to receive launch commands but is not armed. Upon the "combat ready" command from the TFCC, the system moves into the "Combat Ready" state, where the launcher is armed, and the system is prepared to conduct its launch sequence upon receiving further commands. From this point, a missile is either launched or the launch sequence is aborted. At the end of the launch or cancellation of the launch sequence, the operator can use signals to either return the system to standby or safe. Before the NADS is set entirely to safe, it goes through one more transition to "Pending Safe". This state is purely transitional, as it is designated to ensure missiles are safely disarmed and power is drawn out of non-essential components.

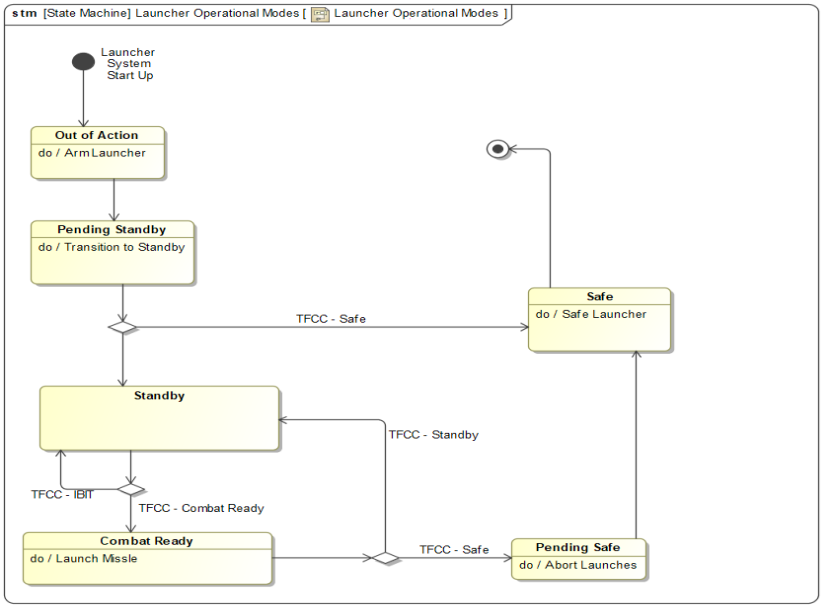

Figure 3. State Machine Diagram of the NADS firing sequence

3.4 Results

The results of these modeling efforts bridge the gap between hardware and software personnel involved with the development of the NADS. The physical aspects of the model give greater insight to those implementing software to existing or future configurations of the system. With a better understanding of the NADS's infrastructure, software engineers can test use cases or optimize the NADS before it is physically built. Furthermore, behavioral architecture better informs the hardware personnel on the physical requirements needed for the system to operate effectively. Understanding how the system will behave under different simulated conditions allows for effective resource allocation during the construction of the system. The integration of the structural and behavioral models sets conditions for the development of a digital twin for engineered systems. Overall, the model creates a common ground between the different disciplines required to develop a complex system. Utilizing this tool offers greater communication and cost efficiency in the design phase, as any discrepancies in the system design can be identified and corrected before resources are irreversibly committed to building an expensive physical prototype or first edition of a product.

ISBN: 97819384969-9-8

4. Conclusion

The creation of architecture for the NADS utilized a unique approach as system behaviors and physical architecture were modeled simultaneously. In a traditional approach to modeling a system using MBSE, the structure of the system is often established first, followed by how the physical components interact with each other using behavioral modeling techniques. Because of access to prior document-based designs, a basic understanding of the system's architecture allowed for the physical structure to be modernized while behaviors were modeled together.

This style of modeling drove our model organization, creating a clear separation between structural and behavioral architecture as opposed to embedding them within each other. Though extremely efficient, this method is limited in the sense that if there were any changes at the most detailed levels of physical architecture, it could force some changes on the behavioral side of the model to maintain continuity and correctness. This tactic would also not be recommended when modeling a system with no previous document-based records.

The comprehensive modeling efforts for the Notional Air Defense System throughout the execution of this project demonstrates the effectiveness of applying MBSE to legacy systems. From modeling physical architecture, engineers working on the system can have a better understanding of the NADS infrastructure, allowing for much easier refinement or evolution in the system's design. From designing the system's behavior, greater insight is achieved on how the system operates and synchronizes signals and commands with its tangible components.

For future work, use cases can be applied to the already established architecture. Variations in design can test the effectiveness of different versions of the system (i.e. a mobile versus stationary system or designs for newly released ballistic threats). Utilizing this approach in the future modernization of defense systems creates a cost-effective avenue to increase theatre-level air defense capabilities.

5. References

Baker, L. (2022, November 7-9). *Model-Based Engineering with SysML: Practical Experience Using CATIA Magic* [Microsoft PowerPoint Model-Based Engineering with SysML: Practical Experience Using CATIA Magic].

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

Gibson, B., Guthrie, P., Rodriguez, J., Trangredi, M., Treviño, M., Styles, M., & Enos, J. (2023). *Notional air defense system: A transition to model-based systems engineering*. Department of Systems Engineering, United States Military Academy.

Hause, M. (2006, September 20). *The SysML Modelling Language.* Retrieved from Fifth European Systems Engineering Conference.: researchgate.net

Holt, Jon, and Tim Weilkiens. *Systems Engineering Demystified: Apply Modern, Model-Based Systems Engineering Techniques to Build Complex Systems*. Second edition. Birmingham, England: Packt Publishing Ltd., 2023.

IBM. (2022). *Engineering Lifecycle Management Suite.* Armonk: IBM Corporation.

Pericherla, Suryateja. (2013, October 4). *Events and Signals in UML*. UML Tutorial for Beginners. https://www.startertutorials.com/uml/events-signals.html

SysML. (n.d.). *SysML FAQ*. Retrieved from SysML FAQ: What is a Sequence diagram (SD) and how is it used?: SysML.org

Visual Paradigm. (n.d.). *What is Sequence Diagram?* Retrieved from Visual Paradigm : visual-paradigm.com

White, P. (2019). *Welcome to SysML, the Language of MBSE,* INCOSE. Welcome to SysML, the Language of MBSE (incose.org)