



# Model-Based Systems Engineering De-Mystified

Dr. Warren K. Vaneman, ESEP

Copyright 2025 Warren K. Vaneman

Model-Based Systems Engineering (MBSE) is a mysterious concept that means many things to many different people. It was envisioned to manage the increasing complexity within systems and System of Systems (SoS). This monograph defines MBSE as the formalized application of modeling (static and dynamic) to support system design and analysis, throughout all phases of the system lifecycle, and through the collection of modeling languages, structures, model-based processes, and presentation frameworks used to support the discipline of systems engineering in a model-based or model-driven context.

Despite the almost two-decade emphasis on MBSE adoption and implementation, there are still misperceptions which prevent MBSE from achieving its full potential. This monograph seeks to “de-mystify” MBSE by exploring its definition, the application of modeling-languages, model structure, model-based processes, and presentation frameworks, and the roll of MBSE tools. Understanding MBSE is fundamental to advancing the system engineering and program management disciplines into a future environment where complexity is going to need to be controlled to field effective systems.

This monograph treats MBSE from a high-level, agnostic perspective with respect to modeling languages, mode structure, modeling processes, presentation frameworks, and modeling tools perspective. This document is not designed to supplant any books MBSE, but to seeks to look at MBSE from a holistic perspective. The organization of this monograph is shown below.

- Chapter 1 – Introduction. Provides the background for MBSE.
- Chapter 2 – The Essence of MBSE. Suggests a revised definition for MBSE and the background on the systems thinking approach where the model is a virtual representation of the system.
- Chapter 3 – Modeling Languages. Addresses the role of modeling languages in MBSE. While two modeling languages (Systems Modeling (SysML) and Lifecycle Modeling Language (LML)) are briefly discussed.

- Chapter 4 – Model Structure. Model structure is a topic that is rarely discussed but is an important topic if the model is going to be a virtual representation of the system.
- Chapter 5 – Modeling Processes. Modeling of systems engineering processes is often restricted to those early architectural and analysis issues. This chapter examines additional modeling considerations across the system’s lifecycle.
- Chapter 6 – Presentation Frameworks. Architecture frameworks have existed since the Zachman Framework was introduced almost 40 years ago. However, these frameworks only show architectural data. This chapter discusses why the current architectural frameworks need to be expanded to include visualization across the systems lifecycle.
- Chapter 7 – Modeling Tools Selection. “Modeling tool wars” have existed since long before MBSE was popularized. Often the winner of the tool wars is the “political” favorite, and not the best solution for the issues that need to be modeled. This chapter provides a “non-political” process for evaluating and selecting tools based on the needs of the of the problem being modeled.
- Chapter 8 – Epilogue. Provides some concluding thoughts on MBSE to include the evolution of MBSE to digital engineering.

Hopefully, this monograph will provide new insights that will allow for the advancement of MBSE for systems-based disciplines.

**Warren K. Vaneman**  
*Sebastian, FL*  
*January 1<sup>st</sup>, 2025*

# CHAPTER 1

## INTRODUCTION

---

*“Advancements in computing, modeling, data management, and analytical capabilities offer great opportunities for engineering practice. Applying these tools and methods, we are shifting toward a dynamic digital engineering ecosystem. This digital engineering transformation is necessary to meet new threats, maintain overmatch, and leverage technology advancements.”*

Kristin Baldwin (2018)

Models<sup>1</sup> have been used in engineering since ancient times to communicate with stakeholders, and to gain insights to increase confidence in the design and reduce risk and costs. The earliest models were physical models that included scale models and prototypes to visualize and validate design concepts prior to full-scale production. Mathematical models emerged to describe a system's behavior and predict outcomes using mathematical equations and relationships. This led to simulation models that allowed systems to be evaluated through a time sequence. More recently digital models have emerged for domain-specific models using Computer Aided Design (CAD), and visualization models such as 3D and virtual reality models.

Systems engineering has always been a discipline based on models. In the early days, those systems engineering models took the form of diagrams, documents, and spreadsheets. For almost two decades, the systems engineering has been undergoing a renaissance to transform from a document-based to a model-based approach since the International Council on Systems Engineering (INCOSE) defined and popularized the term “Model-Based Systems Engineering” (MBSE) in 2007 (INCOSE 2007). This sea-change was prompted by a need to address the increasing system complexity<sup>2</sup> where the models can appropriately be tailored to changing conditions and program needs, re-used, and observed from both static and dynamic perspectives.

---

<sup>1</sup> A **model** is an abstraction of a system, aimed at understanding, communicating, explaining, and designing aspects of the system of interest (SoI).

<sup>2</sup> **System complexity** refers to the degree of difficulty in understanding, predicting, or managing the behavior of a system due to the interactions and interdependencies among its components. This complexity arises from several factors:

- Number of Components - Systems with many parts or elements tend to be more complex.
- Interactions - The ways in which these parts interact can add layers of complexity, especially if the interactions are non-linear or dynamic.

This MBSE transformation means more than using model-based tools<sup>3</sup> and processes to create hard-copy text-based documents, drawings, and diagrams. Data in a MBSE ecosystem is ideally maintained within a single repository and has a singular definition for any model element and allows for the static and dynamic representations of a system from several different perspectives and levels of decomposition.

The enduring challenge for program management, engineering, and acquisition is how to deal with this increased systems complexity, while ensuring a comprehensive and high-quality design. Model-Based Systems Engineering will modernize systems engineering to better support the delivery of capability to meet mission needs, and have the following objectives:

- Increased schedule efficiency and cost saving;
- Improved insights and understanding of complexity within the system;
- Better requirements development and management;
- Encourages re-use within and among models;
- Facilitates more informed decision-making;
- Improved collaboration and communication among stakeholder groups;
- Rapid development of systems and insertion of new technology;
- Increased understanding of system, and system of systems, interoperability.

- 
- Emergence - Complex systems often exhibit emergent properties, where the whole system behaves in ways that are not predictable from the behavior of individual parts.
  - Adaptability - Systems that can adapt or evolve in response to changes in their environment add another layer of complexity.
  - Uncertainty- The presence of uncertainty or randomness in the system's behavior can make it more complex to analyze and predict (SEBoK, n.d.).

<sup>3</sup> A **model-based tool** is a software application or system used to develop models that represent, design, analyze, and manage complex systems. These tools serve as the primary means of information exchange throughout the system's lifecycle and helps visualize and simulate various aspects of a system, such as its requirements, behavior, and structure, making it easier to understand and manage complex interactions and dependencies.

## CHAPTER 2

# THE ESSENCE OF MBSE

---

*“MBSE is fundamentally a thought process. It provides the framework to allow the systems engineering team to be effective and consistent right from the start of any project. At the same time, it is flexible enough to allow the ‘thought’ process to adapt to special constraints of circumstances present in the problem.”* – David Long and Zane Scott (2011)

The fundamental objective of systems engineering is to facilitate a process that consistently leads to the development of successful systems. A system is an integrated set of elements, designed to function together to achieve some defined objective. The key words of this definition are:

- **Integrated** – A series of system elements that when combined work as one;
- **Design** – The deliberate planning and arranging of system elements;
- **Function** – The way in which the integrated system elements work together through rules and procedures that were established during the design;
- **Objective** – The purpose, or goal, the system was designed to satisfy.

It is clear from this definition that the system is more than the physical components but includes all aspects<sup>4</sup> of the system of interest (SoI). As sub-systems are added to systems, and systems are

---

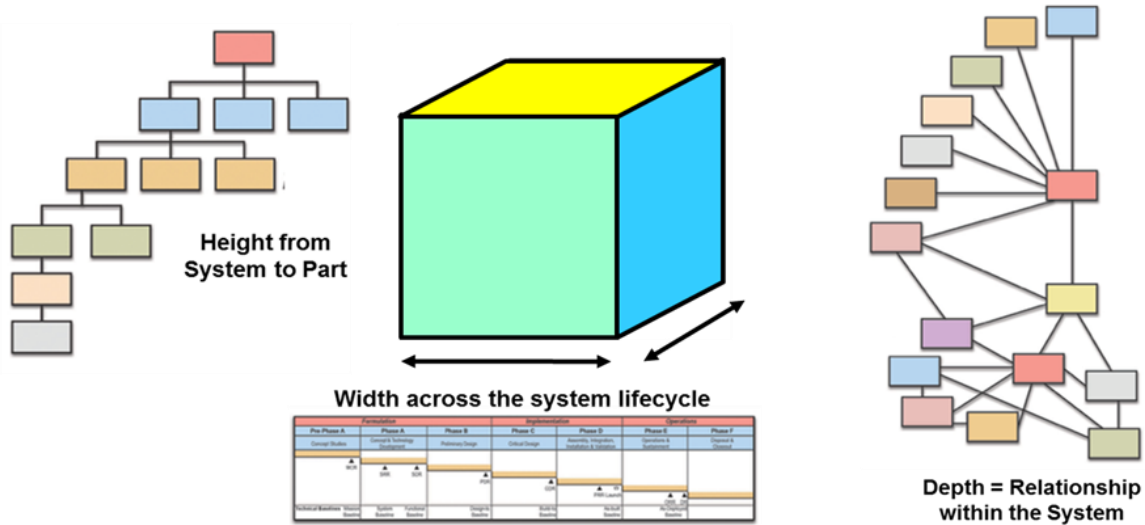
<sup>4</sup> A system can be understood through several key aspects, each contributing to its overall functionality and behavior:

- Components - These are the individual parts or elements that make up the system. Each component has a specific role and function within the system.
- Interactions - The ways in which the components of a system interact with each other. These interactions can be physical, informational, or functional.
- Boundaries - The limits that define what is inside and outside the system. Boundaries help to distinguish the system from its environment.
- Environment - Everything outside the system's boundaries that can affect or be affected by the system. The environment provides inputs to the system and receives outputs from it.
- Inputs and Outputs - Inputs are resources, information, or energy that enter the system, while outputs are the results or products that leave the system.
- Processes - The activities or operations that transform inputs into outputs. Processes are the mechanisms through which the system achieves its goals.

added to System of Systems (SoS), interfaces grow nonlinearly. Interfaces and interactions are often difficult to comprehend, with a cascading effect leading to an uncertain and incomplete architecture that fails to account for emergence within the system. As a result, complexity has emerged as an enduring, and most significant, challenge of systems engineering.

To address the “complexity” challenge, systems engineering is adopting a model-based approach. The system model serves as a virtual representation of SoI through a set of entities and relationships that represent the system’s elements, functions, objectives, and every other aspect of the system. Ideally, each entity is represented in the model as many times it is represented as an element in the actual system – only once. To illustrate this concept the “dimensions” of a system project must be considered. Assume that the cube in Fig. 1 (Vaneman *et al.*, 2019) is a SoI. The system has “width” that provides insight across the entire system lifecycle from the definition of need to system disposal. The system has “height” which provides for the decomposition from the holistic view of the system at the highest level to the components, and eventually the parts, at the lowest levels. The system also has “depth,” which includes the complex relationships between systems, functions, requirements, analysis, risk, costs, schedule, etc. (Vaneman, 2016).

- 
- Feedback - Information about the system's performance that is used to make adjustments and improvements. Feedback can be positive (reinforcing) or negative (corrective).
  - Goals and Objective- The purposes or desired outcomes that the system is designed to achieve. Goals guide the system's operations and development.
  - Structure- The arrangement and organization of components within the system. Structure determines how components are connected and interact.
  - Emergent Properties - Characteristics of the system that arise from the interactions of its components, which cannot be predicted by examining the components individually



**Figure 1:** Dimensions of a Systems Engineering Project (Vaneman, *et al.*, 2019)

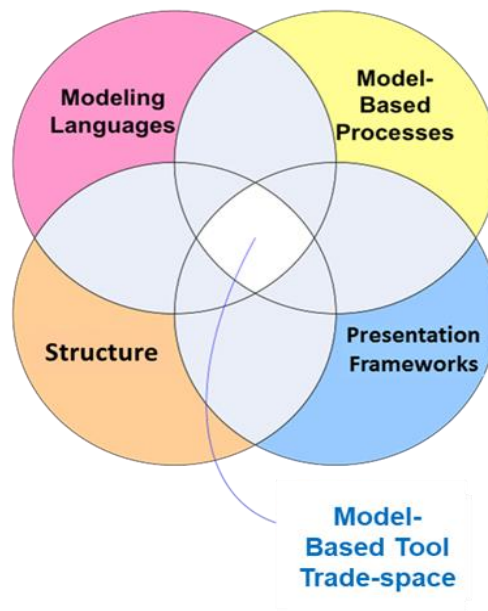
Model-Based Systems Engineering was envisioned to transform systems engineering’s reliance on document-based work products to engineering environment based on models. INCOSE (2007) defines MBSE as “the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” While this definition captures the lifecycle perspective, it does not give any indication how MBSE is different than traditional systems engineering given that “models” have always been a cornerstone of the systems engineering discipline.

An enhanced, and more sufficient, definition of MBSE is “the formalized application of modeling (static and dynamic) to support system design and analysis, throughout all phases of the system lifecycle, through the collection of modeling languages, structures, model-based processes, and presentation frameworks used to support the discipline of systems engineering in a ‘model-based’ or ‘model-driven’ context” (Vaneman, 2016). The four components of MBSE are depicted in Fig. 2, and are defined as:

- **Modeling Languages** – Serves as the basis of tools, and enables the development of system models. Modeling languages are based on a logical construct (visual representation) and/or an ontology. An ontology is a collection of standardized, defined terms and concepts and the relationships among the terms and concepts (Dam, 2019).



- **Structure** – Defines the relationships between the system’s entities. It is these structures that allow for the emergence of system behaviors and performance characterizations within the model.
- **Model-Based Processes** – Provides the analytical framework to conduct the analysis of the system virtually defined in the model. The model-based processes may be traditional systems engineering processes such as requirements management, risk management, or analytical methods such as discrete event simulation, systems dynamics modeling, and dynamic programming.
- **Presentation Frameworks** - Provides the framework for the logical constructs of the system data in visualization models that are appropriate for the given stakeholders. These visualization models take the form of traditional systems engineering models. These individual models are often grouped into frameworks that provide the standard views<sup>5</sup> and descriptions of the models, and the standard data structure of architecture models.



**Figure 2.** Components of MBSE.

Maximum MBSE effectiveness occurs at the convergence of the four components, therefore MBSE tools strive to not only be within this convergence, but to capitalize on various aspects to

---

<sup>5</sup> A “view” is a representation of a related set of information using formats and representations of data in any understandable format that conveys the meaning of the data.

give them a competitive market advantage. Model-Based Systems Engineering tools are general purpose software products that use modeling languages, and support the specification, design, analysis, validation, and verification of complex system representations. These tools serve as the basis of the MBSE ecosystem.

In an MBSE ecosystem, each entity is represented as data, only once, with all necessary attributes and relationships of that entity being portrayed. This data representation then allows for the entity to be explored from the various engineering and programmatic viewpoints. A viewpoint describes data drawn from one or more thematic perspectives and organized in a particular way useful to management decision-making. The compilation of viewpoints (e.g. capability, operational, system, programmatic viewpoints, etc.) represents the entire system, where the system can be explored as a whole, or from a single perspective.

Systems have structures that consist of “building blocks” and their relationships to each other that allows them to come together in a designed form that satisfies the desired system capabilities and functionality. These structures are governed by the property of concordance. Concordance is the ability to represent a single entity, such that data in one view, or level of abstraction, matches the data in another view, or level of abstraction, when talking about the exact same thing. This allows for complexity to be managed more efficiently because each entity is ideally represented in the model only once, essentially creating a virtual representation of the system in the model. Systems engineering views are generated from the data (Vaneman, 2016).

Often the MBSE ecosystem is contained in a single tool with its own data repository. While desirable, it may not be feasible due to the size and scope of the SoI model. Regardless, if the ecosystem is a single tool and data repository, or is composed of multiple tools, and possibly an integrated data repository, the four components must be present and followed for the MBSE ecosystem to be fully effective.

One of the challenges continually faced by program managers and chief engineers is defining the value of MBSE. Many promises have been made about MBSE saving time and rework through the efficient use of data, and early identification of issues that can be detected in the model versus a physical prototype, but there are no suitable MBSE metrics that can be compared to traditional systems engineering processes to determine if any of the claims made about MBSE are valid.

Table 1 identifies seven qualitative MBSE effectiveness measures (Friedenthal and Burkhart, 2015) that will be used to provide an assessment and forecast of the four components of MBSE.

**Table 1.** MBSE Effectiveness Measures and Definitions.

Effectiveness Measure	Definition
Expressiveness	The ability to express system concepts to include at a minimum capabilities, functional, system, standards, and programmatic concepts.
Precise	System representation is unambiguous and concise as needed at various levels of data abstraction.
Presentation/ Communication	Ability to effectively communicate with diverse stakeholders. With standard systems engineering and fit-for-purpose views, answering the questions who, what, when, where, why, and how.
Model Construction/ Manageability	Ability to efficiently and intuitively construct and manage models, to include normal model construction and model extensions for special or domain-specific concepts and terminology.
Interoperable/ Logical Consistency	Ability to exchange and transform data with other models and structured data to include various numeric and non-numeric analysis tasks.
Usable	The ability for stakeholders to efficiently and intuitively create, maintain, and use the model.
Concordance/ Referential Integrity	The ability to represent singularly entity data such that data in one view, or level of abstraction, matches the data in another view, or level of abstraction, when talking about the exact same thing.

## CHAPTER 3

# MODELING LANGUAGES

---

*“Languages shapes the way we think and determines what we can think about.”* - Unknown

Written and spoken languages use different words, syntax<sup>6</sup>, and semantics<sup>7</sup> to describe the same thing. Similarly, despite their differences modeling languages all describe the same thing – the SoI. Modeling languages serve as the basis of tools and enable the development of system models. These languages are based on a graphical representations and/or a data schema. While the languages serve as the foundation for MBSE tool development, tool vendors often interpret the languages to enable the best implementation for their tools. Thus, while a common language is used, the MBSE tools can be very different. For example, MBSE tools that are based on SysML support a common set of graphical models, but typically have unique data schemas (Vaneman, 2016).

The foundation of the MBSE ecosystem is the modeling languages that enable the tools. While these languages have achieved prominence with the System Modeling Language (SysML) during the past decade, the origins can be found in the Structured Analysis and Structure Design (SA/SD) diagrammatic approach that was popular in the 1970s, and the Booch object-oriented software development method in the mid-1990s. Both approaches represented the logical and structural aspects of a SoI with a set of diagrams (Wikipedia, n.d.).

Following the spirit of SA/SD and the Booch object-oriented software approach, the Unified Modeling Language (UML) was developed in the late 1990s as a general-purpose object-oriented graphical modeling languages are intended to provide a standard representation of SoI, primarily for software development (Booch, *et al.* 2005 p. 496). The UML contains 14 diagrams within two pillars – structure and behavior diagrams. (OMG, 2014).

---

<sup>6</sup> Syntax in written language is the way words into phrases, clauses, and sentences to create meaning and convey impact.

<sup>7</sup> Semantics in written language is the study of meaning and interpretation in words, symbols, and sentence structure.

Given the successes of UML, INCOSE and the Object Management Group (OMG) developed SysML as an object-oriented, general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities (Friedenthal, *et al.* 2015). As a profile of the UML metamodel, SysML uses seven of the 14 diagrams (views) from UML, plus two new models based on the needs of the systems engineering discipline. The nine views are categorized into four “pillars” (viewpoints) (behavior, structure, requirements, and parametric), and support requirements specification, analysis, design, validation, and verification, for systems that include hardware, software, information, process, and people. The four SysML pillars, and their associated views and descriptions are shown in Fig. 3-5 (OMG, 2012).

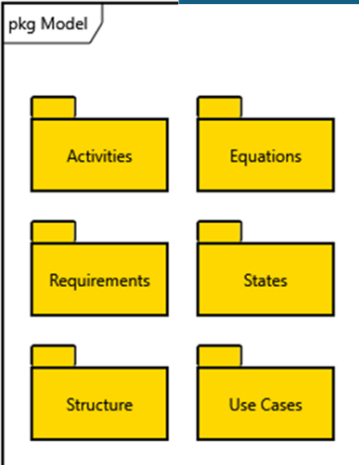
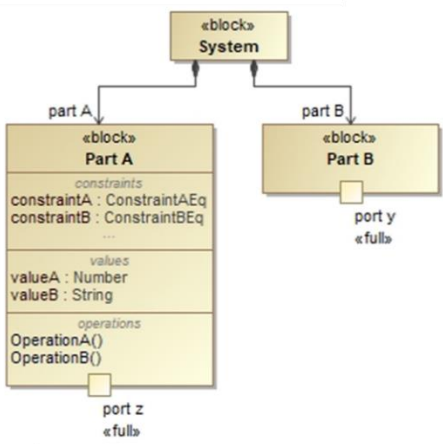
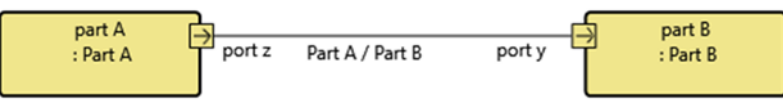
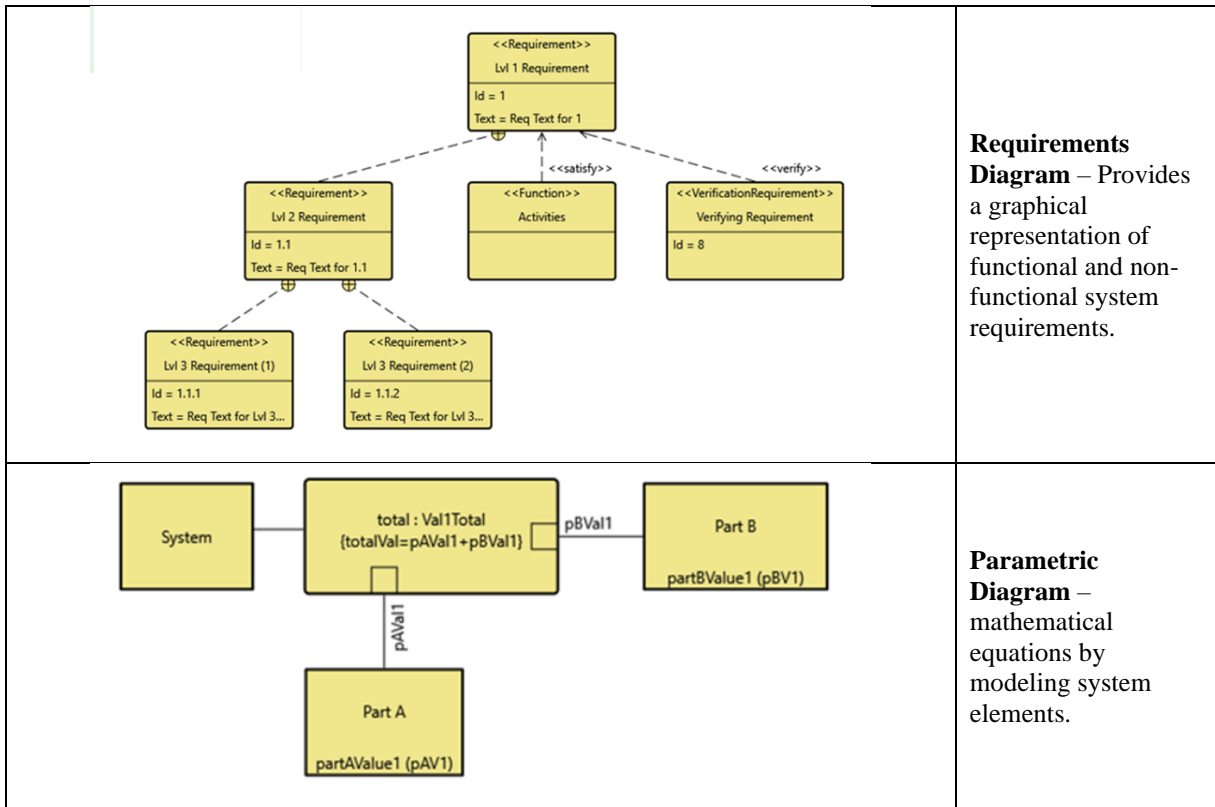
	<p><b>Package Diagram</b> – Describes how a system is divided into logical groupings by showing the dependencies among the groupings.</p>
	<p><b>Block Definition Diagram</b> – Depicts the principal parts of a system as a series of blocks, with interconnections to represent the relationships.</p>
	<p><b>Internal Block Diagram</b> – Describes the internal structure of a system in terms of component properties, and the relationship of the constituent parts.</p>

Fig. 3. SysML Structure Pillar Views.

<p>A Use Case Diagram for a system. Actor A is connected to Use Case A, Use Case D, and Use Case C. Use Case B is connected to Actor B. Use Case C has extension points. Use Case A includes Use Case D. Use Case A extends Use Case C at an extension point.</p>	<p><b>Use Case Diagram</b> - Depicts the system's functionality in terms of actors, dependencies between use cases, and use case goals.</p>
<p>An Activity Diagram showing a sequence of actions: Action 1 leads to Item 1, which leads to Action 3. Action 3 leads to a Loop, which leads to Action 2. Action 2 leads to a final state. There are merge and split nodes throughout the flow.</p>	<p><b>Activity Diagram</b> – Describes the inputs, outputs, and controls of the system activities.</p>
<p>A Sequence Diagram showing two systems, System A and System B, interacting. System A sends Action 1 to System B, and System B sends Action 2 back to System A.</p>	<p><b>Sequence Diagram</b> – visual representation of how objects in a system interact over time.</p>
<p>A State Machine Diagram showing four states: State A, State B, State C, and a final state. Transitions are labeled with guards: [Guard 1], [Guard 2], [Guard 3], [Guard 4], and [Guard 5].</p>	<p><b>State Machine Diagram</b> – Describes the states and the state transitions of the system.</p>

**Fig. 4.** SysML Behavior Pillar Views.

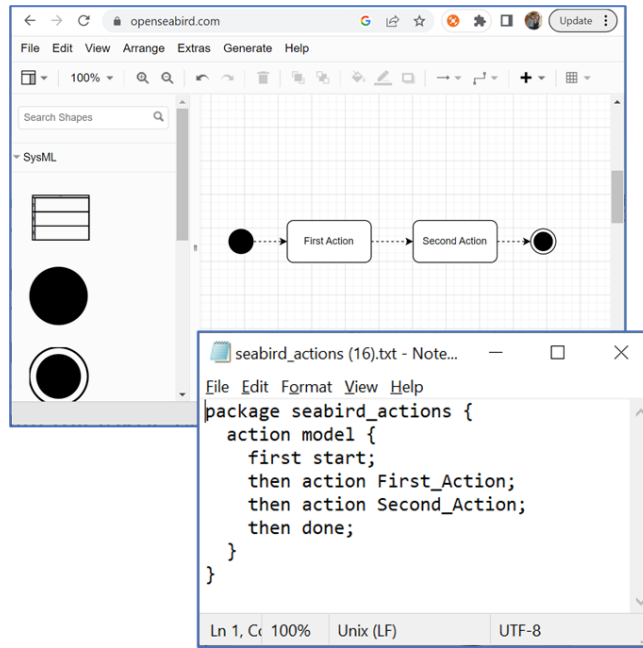


**Fig. 5.** SysML Requirements and Parametric Pillar.

SysML 2.0 is currently being introduced and being implemented in MBSE tools. Unlike SysML 1.0, SysML is not a profile of the UML metamodel. In addition to graphical representations, SysML 2.0 includes a modeling language based on the Kernel Modeling Language (KerML)<sup>8</sup> (OMG, 2023), and an application programming interface (API). KerML provides a textual notation that allows for the expression to precisely represent a SoI, the elements, and the environment. The API provides a standard set of exchange services that can interact with SysML 2.0 to exchange model data among tools (Bajaj, 2022). Fig. 6 is an example of a SysML 2.0 activity diagram and associated KerML code.

<sup>8</sup> KerML is an application-independent modeling language with defined semantics for modeling SoI and includes a general syntax for developing and structuring model relationships, annotations and namespaces (OMG, 2023).





**Fig. 6.** SysML 2.0 Activity Diagram and Associated Code.

The Lifecycle Modeling Language (LML) was introduced as an attempt to provide a simpler language for planning, specifying, designing, analyzing, building, and maintaining modern complex systems. The language takes the principles of MBSE beyond development and production and into the conceptual, utilization, support and retirement stages. It provides a robust, easy to understand ontology<sup>9</sup> that allows you to model complex interrelationships between system components and programmatic artifacts, as well as express system information using easy to understand diagrams.

LML was designed to integrate all lifecycle disciplines, such as program management, systems engineering, testing, deployment and maintenance, into a single framework. As a result, LML can be used throughout the lifecycle. LML uses common, everyday language to define its modeling elements such as entity, attribute, schedule, cost, and relationship. Its primary modeling constructs are the box (which represents any part of the system that is necessary) and the directed arrow

---

<sup>9</sup> An ontology is a collection of standardized, defined terms and relationships between the terms to capture the information that describes the physical, functional, performance, and programmatic aspects of a system (LML Steering Committee, 2022).

(which depicts a relationship between modeled elements such as “consists of,” “derived from,” or “costs”) (LML Steering Committee, 2025 (pending)).

LML combines logical constructs with a corresponding ontology, thus expressing a wide range of entity classes, relationships, and attributes to capture engineering and programmatic information (Vaneman, 2016). The language uses a simplified ontology<sup>10</sup> of 12 primary and eight secondary entity classes to capture the system characteristics, relationships, and interactions. Table 2 shows the LML entities and their definitions (LML Steering Committee, 2022).

---

<sup>10</sup> Common ways of describing such ontologies is entity classes, relationship, and attribute (ERA). ERA is often used to define database schemas. LML uses the ERA approach but extends it by adding attributes to relationships. The extension reduces the number of relationships needed, just as attributes reduce the number of entities needed (Lifecycle Modeling Language Steering Committee, 2025 (pending)).

**Table 2.** LML Entity Classes and Definitions.

Entity Class	Definitions
<b>Action</b>	An <b>Action</b> entity specifies the mechanism by which inputs are transformed into outputs.
<b>Artifact</b>	An <b>Artifact</b> entity specifies a document or other source of information that is referenced by or generated in the knowledgebase.
<b>Asset</b>	An <b>Asset</b> entity specifies an object, person, or organization that performs Actions, such as a system, sub-system, component, person, or element.
<b>Resource (Asset)</b>	A <b>Resource</b> is a child entity of Asset that specifies a consumable or producible Asset
<b>Port (Asset)</b>	A <b>Port</b> entity is a child entity of Asset that represents an interaction point of a block, specifying the input and output flow. <i>Port is included so the MBSE tools based-on LML can be SysML compliant.</i>
<b>Characteristic</b>	A <b>Characteristic</b> entity specifies properties of an entity.
<b>Measure (Characteristic)</b>	A <b>Measure</b> specifies properties of measurement and measuring methodologies.
<b>Connection</b>	A <b>Connection</b> entity specifies the means for relating Asset instances to each other.
<b>Conduit (Connection)</b>	A <b>Conduit</b> is a child entity to Connection that provides a means for physically transporting Input/Output entities between Asset entities. Conduits are constrained by the attributes of capability and latency.
<b>Logical (Connection)</b>	A <b>Logical</b> entity is a child entity to Connection that represents the abstraction of the relationship between any two entities.
<b>Cost</b>	A <b>Cost</b> entity specifies the outlay of expenditure made to achieve an objective associated with another entity.
<b>Decision</b>	A <b>Decision</b> entity specifies a challenge and its resolution.
<b>Input/Output</b>	An <b>Input/Output</b> entity specifies the information, data, or object input to, trigger, or output from an Action.
<b>Location</b>	A <b>Location</b> entity specifies where an entity resides.
<b>Physical (Location)</b>	A <b>Physical</b> entity is a child entity of Location that specifies the location on, above, or below the surface.
<b>Orbital (Location)</b>	An <b>Orbital</b> entity is a child entity of Location that specifies a location along an orbit around a celestial body.
<b>Virtual (Location)</b>	A <b>Virtual</b> entity is a child entity of Location that specifies a location within cyber space or a physical network.
<b>Risk</b>	A <b>Risk</b> entity specifies the combined probability and consequence in achieving objectives
<b>Statement</b>	A <b>Statement</b> entity specifies text referenced by the knowledgebase and is usually contained in an Artifact.
<b>Requirement (Statement)</b>	A <i>Requirement</i> is a child entity to Statement that identifies a capability, characteristic, or quality factor of a system that must exist for the system to have value and utility to the system or user
<b>Time</b>	A <b>Time</b> entity specifies a point or period when something occurs or during which an action, asset, process, or condition exists or continues
<b>Equation</b>	An <b>Equation</b> entity specifies a mathematical or logical equation that can be used to describe a portion of the model. <i>Equation is included so the MBSE tools based-on LML can be SysML compliant.</i>

A fundamental tenet of LML is that each entity class has at least one corresponding visualization. The entities and their graphical models are represented in four areas: functional models; physical models; documentation entities; and parametric and program entities. Many LML models are equivalent to the familiar models that have been developed over time by UML, SysML, Business Process Modeling Notation (BPMN), and other engineering disciplines such as electrical engineering. These common visualizations should be as simple as possible to reduce the complexity of the language and make it more understandable to stakeholders. Other visualizations are encouraged to be used with LML as they aid in expressing information, which is the real goal of any language visualizations (LML Steering Committee, 2025).

As written and spoken languages can be translated, so can modeling languages. The common denominator among the modeling languages are the diagrams. (Vaneman, *et al.*, 2023). Table 3 shows the translation between LML entities, related LML views, and SysML views (Vaneman, 2018).

**Table 3.** LML Entity Classes and Translation between LML to SysML.

Entity Class	LML Graphical Representations	SysML Graphical Representations
Action	Action Diagram, Sequence Diagram	Activity Diagram, Sequence Diagram, Use Case Diagram
Artifact	Photo, Diagram, etc.	
Asset	Asset Diagram	Block Definition Diagram, Internal Block Definition Diagram
Resource (Asset)	Asset Diagram	Block Definition Diagram, Internal Block Definition Diagram, Package Diagram
<i>Port (Asset)</i>	Asset Diagram	Internal Block Definition Diagram
Characteristic	State-Machine, Entity-Relationship, and Class Diagrams	State-Machine Diagram
Measure (Characteristic)	Hierarchy, Spider, and Radar Charts	Parametric Diagram
Connection	Asset Diagram	Block Definition Diagram, Internal Block Definition Diagram
Conduit (Connection)	Asset Diagram	Internal Block Definition Diagram
Logical (Connection)	Entity-Relationship Diagram	Internal Block Definition Diagram
Cost	Pie/Bar/Line Charts	N/A
Decision	Spider Diagram, Hierarchy Diagram, Tree Diagram	Package Diagram
Input/Output	State-Machine Diagram	State-Machine Diagram
Location	Map	N/A
Physical (Location)	Geographic Maps	N/A
Orbital (Location)	Orbital Charts	N/A
Virtual (Location)	Network Maps	N/A
Risk	Risk Matrix	N/A
Statement	Hierarchy Diagram, Requirements	Requirements Diagram
Requirement (Statement)	Hierarchy Diagram, Requirements	Requirements Diagram
Time	Gantt Chart, Timeline Diagram	N/A

Modeling languages such as LML, UML, SysML, SysML 2.0 (in the future) support a the specification, design, analysis, verification, and validation of complex SoI, there are other languages that should be considered for addressing problems and processes through the system’s lifecycle. These include languages used for numerical analysis – a class of problems often overlooked by the systems engineering community. Table 4 shows a sampling of several popular modeling languages for the modeling and analysis of systems.

**Table 4.** Popular Languages for the Modeling and Analysis of Systems.

Modeling Language	Description
<b>Architecture Analysis and Design Language (AADL)</b>	Language used for modeling and analyzing the architecture of embedded systems, particularly in avionics and automotive industries. It helps in performance analysis and ensures system reliability.
<b>Business Process Model and Notation (BPMN)</b>	A graphical representation for specifying business processes in a business process model. It provides a standard notation that is understandable by all business stakeholders, including business analysts, technical developers, and business managers
<b>Lifecycle Modeling Language (LML)</b>	An open-standard modeling language designed for systems engineering. It support the entire system’s lifecycle. (LML Steering Committee, 2022).
<b>MATrix LABoratory (MATLAB)</b>	A high-level programming language and numerical computing environment developed by MathWorks. It is based-on matrix and array mathematics, which support matrix algebra and numerical computations.
<b>Modelica</b>	An object-oriented language for modeling complex physical systems. It is used for simulation and analysis in various engineering domains, including mechanical, electrical, and control systems.
<b>Simulink</b>	A graphical programming environment for modeling, simulating, and analyzing multidomain dynamical systems.
<b>System Definition Language (SDL)</b>	A structured modeling language based on an Entity, Relationship, Attribute (ERA), to model a system’s architecture. SDL is used within ViTech’s systems engineering tools.
<b>System Modeling Language (SysML)</b>	A graphical systems engineering language to support the specification, design, verification and validation of systems
<b>System Modeling Language (SysML) 2.0</b>	The next-generation Systems Modeling Language, developed to improve the precision, expressiveness, and usability of the original SysML (Systems Modeling Language). It aims to enhance MBSE practices by addressing limitations and incorporating lessons learned from SysML 1.0
<b>Unified Modeling Language (UML)</b>	A general-purpose, graphical modeling language primarily used for software engineering but also applicable to systems engineering

Modeling languages serve as the foundation for MBSE tool development, vendors often interpret the languages to enable the best implementation for the goals of their tools. While a common language is used, the differences in the implementation of tool data schema often limits the ability to effectively exchange data. For example, MBSE tools that are based on SysML support a common set of visual models, but typically have unique data schemas making exchanging data between different tools difficult, if not impossible (Vaneman, 2016). Table 5 shows the assessment and forecast of modeling languages to the MBSE effectiveness measures.

**Table 5. Modeling Languages Assessment and Forecast.**

Effectiveness Measure	Modeling Languages
Expressiveness	Recent developments in modeling languages allow MBSE to be more expressive. However, the strength of current modeling languages is for the early lifecycle phases. The languages need to be expanded to be expressive of the full system lifecycle with data represented by additional graphical representations.
Precise	Current languages are attempting to add precision, but these efforts are decentralized and uncoordinated. LML has a parsimonious ERA language that provides a manageable set of entities from which to develop models. KerML defines the syntax and semantics for SysML 2.0, which should add to precision. As modeling-languages are expanded to include the full system lifecycle, an equal amount of precision needs to be added to each lifecycle phase.
Presentation/ Communication	Presentation frameworks don't have any unique views, but renames and uses existing systems engineering graphical representations. Therefore, the architecture frameworks within MBSE tools are based on the modeling language that the tool is based on. As graphical representations in modeling languages are expanded, the views in presentation frameworks will also be expanded.
Model Construction/ Manageability	The ease of model construction is based on how tool vendors implement the modeling languages. LML and SysML 2.0 now can develop models by two methods. Both languages can develop the models graphically; LML can develop models from a data perspective; models in SysML 2.0 can be generated via the code based KerML.
Interoperable/ Logistical Consistency	Another major area for modeling language improvement is interoperability and logical consistency. SysML 2.0 has an API that should make sharing data between models more effective and efficient. However, exchanging models between different SysML 2.0 MBSE tools has yet to be proven. LML allows for data to be exchanged via an xmi interface, which works well between LML-based models, but not between other tools. While having multiple modeling languages is desired for the health of MBSE, a common interface exchange needs to be developed that will support existing and future modeling languages.
Usable	The usability of modeling languages is dependent on how it is implemented in MBSE tools.
Concordance/ Referential Integrity	All modeling languages can make relationships between entities. These relationships allow for concordance and emergent behavior.

## CHAPTER 4

# MODEL STRUCTURE

---

*“Creativity is the power to connect the seemingly unconnected.”*  
– William Plomer

Structure is probably the least understood aspect of MBSE. Like “creativity” in the William Plomer quote, model structure is the power to connect the seemingly unconnected. Systems consist of “building blocks” and their relationships to each other that allow them to come together in a designed form that satisfies the desired capabilities and functionality. Model structures establish concordance and define the relationships between the system entities that allows for the emergence of system behaviors and performance characterizations within the model. The conceptual data model describes the elements, attributes, and relationships that can be made within the model.

MBSE implementations often do not apply model structures efficiently, thereby causing data to be represented in the model more than once - which is tantamount to document-based artifacts. This often results in data maintenance errors, leading to different representations of the same data in different viewpoints, thereby failing to take advantage of a key benefit of MBSE.

All modeling languages have the ability to make relationships within the model. In graphical based languages, such as SyML, a relationship path can be formed by selecting an entity and navigating the relationship to another entity (Friedenthal, *et al.*, 2015). For example, in SysML, a function in an activity model can be allocated to a system entity in an internal block definition diagram (IBD). A value in an IBD can be assigned to an equation in a parametric diagram to represent a system measure. A value in a parametric diagram can verify a requirement in a requirements diagram. And a requirement can be related to a system entity in the IBD to be satisfied.

An ERA-based modeling language represents structure by defining the relationships between entities. The relationship between the 20 primary and secondary entities of LML use an economy of relationships approach, which is assisted by the bi-directional nature of relationships to the entities. Each entity has a *one-to-many*, or *many-to-many*, possible relationships within the matrix. Thus, for 20 defined entities, the total possible combinations is not  $20^2$  or 400 combinations but



approaches 20/ or  $2.43 \times 10^{18}$  combinations. This phenomenon allows the complexity of the SoI to be modeled while having a manageable entity and relationship “vocabulary,” at the atomic level. Fig. 7 shows an extraction of the LML relationship matrix (LML Steering Committee, 2022).

	Action	Artifact	Asset (Resource)	Characteristic (Measure)	Connection (Conduit, Logical)	Cost	Decision	Input/Output	Location (Orbital, Physical, Virtual)	Risk	Statement (Requirement)	Time
Action	decomposed by* related to*	references	(consumes) performs by (produces) (seizes)	specified by		incurs	enables results in	generates receives	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Artifact	referenced by	decomposed by* related to*	referenced by	referenced by specified by	defines protocol for referenced by	incurs referenced by	enables referenced by results in	referenced by	located at	causes mitigates referenced by	referenced by (satisfies) source of traced from (verifies)	occurs
Asset (Resource)	(consumes by) performs (produced by) (seized by)	references	decomposed by* related to*	specified by	connected by	incurs	enables made responds to results in		located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Characteristic (Measure)	specifies	references specifies	specifies	decomposed by* related to* specified by	specified by	incurs	enables results in		located at	causes mitigates	(satisfies) specifies	occurs
Connection (Conduit, Logical)		defined protocol by references	connects to	specified by	decomposed by* related to*							
Cost	incurred by	incurred by references	incurred by	incurred by specified by	incurred by							
Decision	enabled by result of	enabled by references result of	enabled by made by responded by result of	enabled by result of specified by	enabled by result of							
Input/Output	generated by received by	references		specified by	transferred by							
Location (Orbital, Physical, Logical)	locates	locates	located at	located at specified by	locates							
Risk	caused by mitigated by resolved by	caused by mitigated by resolved by	caused by mitigated by resolved by	caused by mitigated by resolved by	caused by mitigated by resolved by							
Statement (Requirement)	(satisfies) traced to (verifies)	(satisfies by) sourced by traced to (verifies)	(satisfies by) traced to (verifies)	(satisfies by) specified by traced to (verifies)	(satisfies by) traced to (verifies)							
Time	occurred by	occurred by	occurred by	occurred by specified by	occurred by							

Fig. 7. Insert of the LML relationship matrix (LML Steering Committee, 2022).

It can be daunting to begin a modeling effort when there are  $10^{18}$  possible combinations. The relationships within the matrix should be used as a guide to create a conceptual data model (CDM) that is specific to a SoI. A CDM is a mapping of acceptable entities and relationships within the system. It serves as the “blueprint” for the planning and building of the system model. Engineers would not consider building a complex system without a plan, so why should a complex system model be developed without a CDM?

Fig. 8 represents a notional CDM of a SoI. The CDM has four viewpoints: policy and standards viewpoint (shown in yellow); operational viewpoint (shown in orange); systems viewpoint (shown

in blue; and programmatic viewpoint (shown in green). Each viewpoint is composed of entities, examples of entities in {}, and associated relationships, used to represent the virtual representation of the SoI. Each data representing an element with the SoI should be modeled as many times as it is represented in the SoI – only once. This modeling approach allows for concordance to be achieved within the model.

The systems engineering process begins with the definition of need which is usually contained in document that is defined by an Artifact. Artifacts {Policy, Guidance, Governance, Standard} are inputs to the systems engineering process, which are not meant to be changed within the development of the model. While the Artifact is the source of, the data can be parsed into individual Statements {Need, Goal, Objective, Assumption} which provide the actional background needed to develop the SoI.

The Requirements {Originating} are traced from the Statements and serve as the source of the requirements that the SoI must support. Requirements {Originating} are traced to Action {Capability}. A capability is a solution-neutral statement that states the SoI much achieve a desired effect under specified conditions. Action {Capability} is related to Action {Activity} can be further related to Action {Function}.



Fig. 8. Conceptual Data Model Example.

Activities and functions both represent specific actions to achieve the desired objective. The difference is, an activity is associated with the operational steps that are taken, while functions represent what the system is performing. Action {Capability, Activity, Function} generate/receive Input/Output {Data, Information, Energy, Trigger} from an operational perspective.

Inputs/Outputs {Data, Information, Energy, Trigger} are transferred via Connections {Conduit, Interface, Data Link, Pipe}. Connections connect Assets to other Assets. The difference between Inputs/Outputs and Connections are, Inputs/Outputs represents the commodity that is being transferred between Actions, while Connections are the physical interfaces between Assets. Inputs/Outputs and Connections need to be modeled concurrently because the physical interfaces need to be properly sized to support the commodity that is being transferred.

Assets are specified by Characteristics which represent qualitative and quantitative “ilities” (e.g. reliability, availability, composability, useability). When the Characteristics are quantitative, they can be further defined by Measures {Measures of Effectiveness (MOE), Key Performance Parameters (KPP), Measures of Performance (MOP), and Technical Performance Measures (TPM)}. Assets also have a Location {Physical, Virtual, Orbital}.

The Program Management Viewpoint depicts the relationships between Action and Time {Phase, Milestone, Schedule}. Assets specify Risks {Cost, Schedule, Technical} and incurs Cost {Actual, Planned, Total Ownership}. Cost, Risk, and Time are related to Decision {Major Decision, Problem, Resolution, Challenge, Issue}.

To realize a true virtual representation of the SoI, model structure must be implemented into the modeling-process. Given the over-reliance on document-based processes in system engineering, applying model structure to form the virtual representation of the system may seem unnatural. Model structure requires a mindset change to realize the benefits of MBSE. These benefits include gaining insights into SoI connectedness and emergent behavior. Table 6 depicts the assessment and forecast of model structure to the MBSE effectiveness measures.

**Table 6. Model Structures Assessment and Forecast.**

Effectiveness Measure	Model Structure
Expressiveness	Model structures are used to ensure that data can be expressed efficiently by modeling the data, relationships, and attributes only once. This establishes concordance and allows for emergent behavior that identifies SoI characteristics that heretofore have been unrealized.
Precise	Model structures ensure a precise representation of the data by modeling the data, relationships, and data only once. Many current MBSE implementations fail to use model structures fully, thereby causing data to be represented in the model more than once, which could yield in-precise data representations, especially as the data is upgraded during the life of the model.
Presentation/ Communication	Model structures allow for the presentation of a system from a common data set. Many current implementations are still based on the traditional systems engineering product paradigm. Therefore, concordance is not guaranteed across different viewpoints.
Model Construction/ Manageability	Tool dependent. Model structures are implemented differently in various tools.
Interoperable/ Logistical Consistency	Tool dependent. Because model structures are currently tool dependent, exchanging data, and maintaining the data relationships and attributes is problematic.
Usable	Model structures will make the data usable by representing the data only once. However, this requires a mindset change to be effective.
Concordance/ Referential Integrity	Model structures are at the heart of concordance. Without model structures, concordance cannot exist. However, many current implementations focus on collecting and populating data from a product perspective, thereby overlooking concordance.

## CHAPTER 5 MODELING PROCESSES

---

*“The enterprise architecture must serve as the glue and bridge between the vision and its execution.” - Kevin Brett (2022)*

Model-based processes provide the analytical framework to conduct the analysis of the system virtually defined in the model. The model-based processes may be traditional systems engineering processes such as requirements management, risk management, or analytical methods such as discrete event simulation, systems dynamics modeling, and dynamic programming.

In both traditional systems engineering processes, and in MBSE, different analytical approaches are used to address the various challenges throughout the system’s lifecycle. The primary difference with MBSE is, ideally, the system data is collected once, and used to address the many system’s challenges.

There are some common misperceptions about MBSE processes. One misperception is that MBSE is just systems engineering conducted in a model-based tool but yields traditional engineering products. To be effective, the organization must change its processes to get the most benefit from MBSE. Another misperception is that there is a “one size fits all solution” for MBSE processes. Like traditional systems engineering processes, MBSE processes must be tailored for the given system problem. A generic process can serve as a guide to satisfy model-based efforts throughout the systems engineering lifecycle (DoD CIO, 2009).

The MBSE effort starts by determining the intended use of the model, which permits proper model, data, and data structure and definition. “Intended Use” is a description of the problem to be addressed by a model or simulation, and its associated data, including the system or process being represented and the role it plays in the overall program. Intended use is defined by answering the following questions below:

- What problem is being addressed?

- What are the key attributes of the problem (focus, epoch, data and model fidelity, model type(s), and viewpoints to adequately portrait the data and results)? Table 7 defines the key model attributes.

**Table 7.** Key Model Attributes Defined.

Attribute	Definition	Examples
<b>Focus</b>	The system level that is the focus of the model's intended use.	<ul style="list-style-type: none"> <li>• Component</li> <li>• Sub-system</li> <li>• System</li> <li>• System of Systems</li> </ul>
<b>Epoch</b>	The planned time period that the model intends to simulate.	<ul style="list-style-type: none"> <li>• Current</li> <li>• Near-term (1- 5 years)</li> <li>• Long-term (&gt; 5 years)</li> </ul>
<b>Fidelity</b>	<p>The degree to which the model reproduces the state and behavior of the real-world environment and systems.</p> <p>Fidelity is a subjective measure, however, accuracy, precision, repeatability, resolution, scope, and sensitivity.</p>	<ul style="list-style-type: none"> <li>• High fidelity – Measures, standards, and perceptions are very similar to real-world systems and environment.</li> <li>• Medium fidelity – Measures, standards, and perceptions are somewhat similar to real-world systems and environment.</li> <li>• Low fidelity - Measures, standards, and perceptions are marginally similar to real-world systems and environment.</li> </ul>
<b>Model Types</b>	The type and level of model required to achieve the analysis purposes of the intended use.	<ul style="list-style-type: none"> <li>• Static Model – A model that depicts a system, and dependencies, at an instant in time.</li> <li>• Dynamic Model – A model that depicts a system, its dependencies, and behavior as a function of time.</li> </ul>
<b>Viewpoints</b>	Describes data drawn from one or more perspectives and organized in a particular way useful to management decision-making. A viewpoint definition includes the information that should appear in individual views; how to construct and use the views; the modeling techniques for expressing and analyzing the information; and a rationale for these choices.	<ul style="list-style-type: none"> <li>• Capability</li> <li>• Operational</li> <li>• Technical and Standard</li> <li>• Systems</li> <li>• Services</li> <li>• Parametric</li> <li>• Requirements</li> </ul>

The second step is to determine the scope of the systems engineering problem. The following questions should be considered when determining the scope of the MBSE effort:

- What are the functional bounds of the system?

- What are the technological bounds of the system?
- What are the geographical bounds of the system?
- What are the system constraints?

System boundaries are often difficult to determine. Sometimes the systems engineer will expand the boundary of their system to simplify interfaces or inputs. A good rule of thumb to determine the system boundary is, does the system have the requirements and funding for the element in question? If the system's controlling agency does not have jurisdiction of the requirements or funding of the element, it resides outside the boundary of the system.

The third step is to determine the data required to support the MBSE effort. Data takes on several dimensions in this step. What data elements are needed? This includes data from different perspectives (i.e. capability, functional, system, parametric, etc.) to be represented in various viewpoints later in the process.

What level of detail does the data need to support? It is often the practice to over-collect data to the lowest possible level. Data only needs to be collected to the level required to address the systems engineering problem. Collecting data at a lower-level results in a waste of resources and time. Data that is collected to address future detailed concerns will incur a cost to maintain its accuracy.

The fourth step is to collect the data and make relationships between the data. Correlating the data to form relationships allows for concordance that was discussed in the Model Structure section. The data relationships for LML governed by rules defined the LML Relationships Matrix (Fig. 8) and are further defined for each program by the CDM.

The fifth step is to conduct the analysis in support of the MBSE objectives. Traditional MBSE analysis often uses architectural data to conduct capability analysis, gap analysis, interoperability analysis, and architecture closure. To be most effective, MBSE must include analysis types found in operations research (i.e. discrete event simulation, optimization, etc.), and system dynamics modeling, in addition to architectural analysis. Table 8 shows the attributes of modeling analysis for each phase in the systems engineering lifecycle.

**Table 8.** Attributes of Modeling Analysis Throughout the Systems Lifecycle.

Lifecycle Phase	Focus	Epoch	Fidelity	Analysis Type
Conceptual Design	<ul style="list-style-type: none"> <li>• Single System</li> <li>• Multiple Systems</li> <li>• System of Systems</li> </ul>	<ul style="list-style-type: none"> <li>• Near-term (1- 5 years)</li> <li>• Long-term (&gt; 5 years)</li> </ul>	Medium Low	<ul style="list-style-type: none"> <li>• Capability Analysis</li> <li>• Gap Analysis</li> <li>• Mission Analysis</li> </ul>
Preliminary Design	<ul style="list-style-type: none"> <li>• Single System</li> <li>• Multiple Systems</li> <li>• System of Systems</li> </ul>	<ul style="list-style-type: none"> <li>• Near-term (1- 5 years)</li> <li>• Long-term (&gt; 5 years)</li> </ul>	High Medium	<ul style="list-style-type: none"> <li>• Operational Modeling and Architecting</li> <li>• System Modeling and Architecting</li> <li>• Performance Analysis (Predictive)<sup>11</sup></li> <li>• Requirements Analysis<sup>12</sup></li> </ul>
Detailed Design	<ul style="list-style-type: none"> <li>• Component</li> <li>• System</li> <li>• Multiple Systems</li> <li>• System of Systems</li> </ul>	<ul style="list-style-type: none"> <li>• Current</li> <li>• Near-term (1- 5 years)</li> <li>• Long-term (5 years &lt; t &lt; 10 years)</li> </ul>	High Medium	<ul style="list-style-type: none"> <li>• Trade-off Analysis<sup>13</sup></li> <li>• Initial Performance Analysis (Prototypes)</li> <li>• Risk Analysis<sup>14</sup></li> <li>• Project Analysis</li> </ul>
Mission Assurance	<ul style="list-style-type: none"> <li>• Component</li> <li>• System</li> <li>• Multiple Systems</li> <li>• System of Systems</li> </ul>	<ul style="list-style-type: none"> <li>• Current</li> <li>• Near-term (1 year)</li> </ul>	High Medium	<ul style="list-style-type: none"> <li>• Verification</li> <li>• Validation</li> <li>• Accreditation</li> <li>• Certification</li> <li>• Integration<sup>15</sup></li> </ul>
Operations and Support	<ul style="list-style-type: none"> <li>• Single System</li> <li>• Multiple Systems</li> <li>• System of Systems</li> </ul>	<ul style="list-style-type: none"> <li>• Current</li> <li>• Near-term (1 year)</li> </ul>	High Medium	<ul style="list-style-type: none"> <li>• Performance Analysis (Actual)</li> <li>• Risk Analysis</li> </ul>

<sup>11</sup> Performance Analysis use use simulations to analyze system performance, including response times, throughput, and resource utilization, helping to identify bottlenecks and optimize performance.

<sup>12</sup> Requirements Analysis use simulations to validate and refine system requirements by modeling different scenarios and assessing how well the system meets its intended functions.

<sup>13</sup> During the detailed design phase, simulations are used to explore different design alternatives, optimize system performance, and identify potential issues early in the development process.

<sup>14</sup> Risk analysis use static and simulation models help assess risks by modeling potential failure modes and their impacts, enabling engineers to develop mitigation strategies.

<sup>15</sup> Integration and testing use simulation models to plan and evaluate the integration of system components by testing interactions and dependencies in a virtual environment before physical integration.



Due to limitation in modeling-languages and architecture frameworks, MBSE has been relegated to the early phases – conceptual and preliminary system design - of the systems engineering lifecycle. To be truly respected, MBSE must be expanded to include the entire lifecycle to include test and evaluation and operations and sustainment. Live, virtual, and constructive (LVC) offers an untapped potential for expanding MBSE to realize a virtual representation to explore the SoI that have not been previously possible. Table 9 shows the assessment and forecast of model-based processes to the MBSE effectiveness measures.

**Table 9.** Model-Based Processes Assessment and Forecast.

Effectiveness Measure	Model-Based Processes
Expressiveness	MBSE process allows for the expression of multiple concepts to include at a minimum capabilities, functional, system , standards, and programmatic data. Heretofore, the MBSE practices have been relatively silent on mathematically-based processes, except for the most basic problems.
Precise	MBSE process allows for system data to be represented in unambiguously at various levels of fidelity and decomposition, and from various viewpoints.
Presentation/ Communication	The generic MBSE processes allows for the data to be presented in various viewpoints to meet stakeholders needs. However, to be able represent the data accurately and efficiently, the planning, collection, storage, correlation, and the engineering analysis of data is essential.
Model Construction/ Manageability	A well defined CDM will allow for effective model construction, and will yield reusable models, and repeatable analysis results.
Interoperable/Logical Consistency	Not Applicable
Usable	A well-define model-based on a CDM will allow for the effective model-based processes that will yield insights into the SoI that are not available with traditional systems engineering processes.
Concordance/Referential Integrity	Not Applicable

## CHAPTER 6

# PRESENTATION FRAMEWORKS

---

*“The purpose of [model] is to support decision-making, and it must be data driven, not diagram driven.” - Kevin Brett (2022)*

Presentation frameworks<sup>16</sup> provide the logical structure to categorize and organize various system views into thematic viewpoints, that serves as a standard for a given set of stakeholders. The “dirty little secret” of presentation frameworks is that they have no unique views of their own; every view is a standard system engineering diagram that is renamed for the framework.

Presentation frameworks are developed by answering the architecture interrogatives of who, what, when, where, why, and how. All presentation frameworks provide: definitions of standard views within the framework; guidance and rules for organizing viewpoints, structuring data, and systems engineering views; and, references to compare and contrast models when in the same format. The frameworks enable decisions across the enterprise due to the commonality and familiarity with the standard viewpoints and views. Complexity in the model-based ecosystem is significantly reduced by separating and characterizing systems issues into various viewpoints and views.

The Zachman Enterprise Architecture Framework was the first logical structure for classifying and organizing system information to represent all stakeholders. The Zachman Framework is organized by the architecture interrogatives - what (data), how (function), where (network), who (people), when (time), why (motivation)) compared against the stakeholder perspectives (executive perspective(planner), business management perspective (owner), architect perspective (designer), engineer perspective (builder), technician perspective (maintainer), operator perspective (user). At the intersection of each architecture interrogative and stakeholder perspective is the defined view (Fig. 9) (Zachman, 1988; Zachman, n.d.).

---

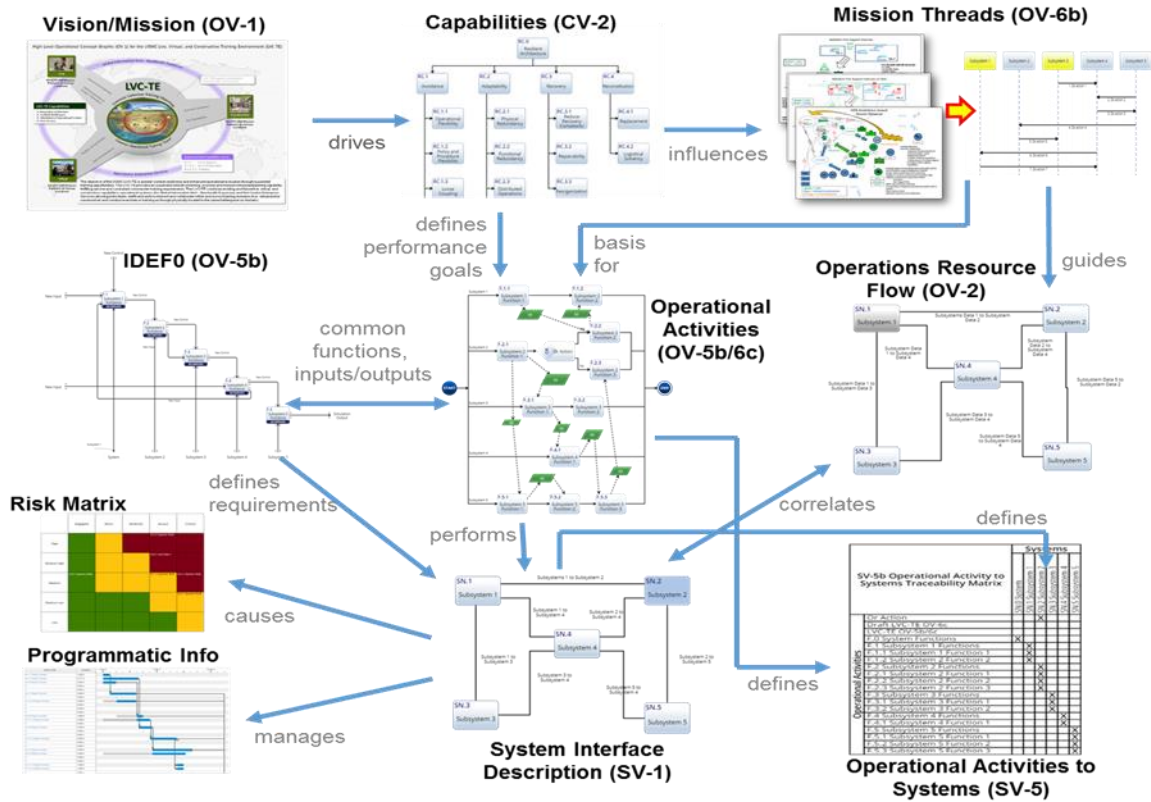
<sup>16</sup> The term “presentation framework” is used in lieu of “architecture framework” because the views defined in architecture frameworks are focused on the early systems engineering activities, while the views in presentation frameworks are extended to include views throughout the system lifecycle. The goal is to perform model-based system engineering and not model-based systems architecting.

	<b>Data (What)</b>	<b>Function (How)</b>	<b>Network (Where)</b>	<b>People (Who)</b>	<b>Time (When)</b>	<b>Motivation (Why)</b>
<b>Executive Perspective (Planner)</b>	List of Systems	List of Process	List of Locations	List of Organizations	List of Events	List of System Goals
<b>Business Management Perspective (Owner)</b>	Semantic View	Business Process View	Logistic Network View	Workflow View	Master Schedule	System Engineering Plan
<b>Architect Perspective (Designer)</b>	Logical Data View	Application View	Distributed Systems View	Human Interface View	Processing Structure View	System Rules View
<b>Engineer Perspective (Builder)</b>	Physical Data View	System Design View	Technical Network View	System Presentation View	Control Structure View	Sub-system Rules Views
<b>Operator Perspective (User)</b>	Data Definition View	Program View	Network View	Security View	Timing Distribution View	Rules Specification

**Fig. 9.** Intersection of Architecture Interrogatives and Stakeholder Perspectives.

Architectural frameworks are usually established at an enterprise level, and not with individual programs. The U.S. government uses several architectural frameworks, with one of the most common being the Department of Defense Architecture Framework (DoDAF), provides guidance for describing architectures for both DoD operations and business processes. The framework provides the guidance, rules, and product data descriptions for developing and presenting architecture descriptions that ensure a common denominator for understanding, comparing, and integrating families of systems, systems of systems, and interoperating and interacting architectures (DoD CIO, 2009).

DoDAF categorizes the SoI into eight viewpoints: capabilities; operations; systems; services; standards; project; data and information; and all viewpoints. These eight viewpoints are further defined by 52 views (formerly called diagrams) to represent the SoI, thus allowing the system to be explored holistically, or from a single perspective. The framework also has the flexibility to include other models, or “fit for purpose views,” that may be needed to address perspectives that are not included in the framework (DoD CIO, 2009). The DoDAF viewpoints and views complement each other from a compressive virtual representation of the SoI. Fig. 10 shows an example of interactions within a systems architecture (Vaneman and Carlson, 2018).



**Fig. 10.** Example of Interactions within DoDAF (Vaneman and Carlson, 2018).

While DoDAF is good for the defense related systems, it is not representative of other enterprises with the U.S. Government. A popular non-defense architecture is the Federal Enterprise Architecture Framework (FEAF), whose purpose is to standardizes and streamline architecture development processes, reduce duplication, and increase the agility, flexibility, and effectiveness of primarily information technology systems. NASA has the Space Mission Architectural Framework (SMAF) that represents the information required for the unique unmanned space missions. A brief description of frameworks commonly used by the U.S. Government are shown in Table 10.

**Table 10.** Summary of Frameworks used by the U.S. Government.

Architectural Framework	Brief Description
<b>Department of Defense Architecture Framework (DoDAF)</b>	DoDAF addresses the conceptual model enabling the development of architectures and provides guidance on the development of architectures supporting the adoption and execution of systems and services. DoDAF consists of eight viewpoints and 52 views.(DoD CIO, 2009).
<b>Federal Enterprise Architecture Framework (FEAF)</b>	FEAF was established to facilitate shared development of common processes and information among Federal Agencies and other government agencies. FEAF is partitioned into: <ul style="list-style-type: none"> <li>• Business Architecture – Address the architecture interrogatives of “what,” “when,” “how,” “who,” and “why.” It does not address “where” as the Zachman Framework does.</li> <li>• Data Architecture: Information used by the organization to conduct business and make administrative decisions.</li> <li>• Application Architecture - System and service applications that process the data according to defined business rules.</li> <li>• Technology Architecture - Systems that supports the business, data, and application architectures. (OMB, accessed April 5, 2024)</li> </ul>
<b>NASA Space Mission Architecture Framework (SMAF)</b>	SMAF is a framework that represents uncrewed space missions. Viewpoints include: <ul style="list-style-type: none"> <li>• Enterprise/Mission Concept Viewpoint – Models the top-level organizational perspective.</li> <li>• Mission Operation Viewpoint - Models launch, ground, and flight operations.</li> <li>• Engineering Viewpoint – Models the technical activities and resulting products involved in formulation and implementation of the mission. This includes the technical solution view that model the conceptual and functional perspectives, and the product realization viewpoint that models the implementation of the functional architecture into a physical system.</li> <li>• Project Implementation Viewpoint – Addresses the architecture from the perspective of overall project planning, management, and control.</li> <li>• Science Viewpoint – Models the science mission being conducted (NASA, 2021).</li> </ul>
<b>Unified Architecture Framework (UAF)</b>	UAF defines ways of representing an enterprise architecture that enables stakeholders to Maintain a holistic perspective while focusing on detailed specific areas of interest. UAF contains 75 views, and bridges the existing DoDAF, U.K.’s Ministry of Defence’s Architecture (MoDAF) and the NATO Architecture Framework (NAF) (OMG, 2020).

The Unified Architecture Framework (UAF) has been developed as the next generation architecture framework for defense enterprises. The UAF is not developed as a new architectural framework *per se*, but a way of bridging the differences between DoDAF, United Kingdom’s Ministry of Defense Architecture Framework (MODAF), and the NATO Architecture Framework

(NAF) so that these defense related frameworks are more compatible. Given the cooperative system development, UAF is the first modeling standard to enjoy adoption from around the world (Hause, 2012).

UAF defines ways of representing an enterprise architecture that enables stakeholders to focus on specific areas of interest in the enterprise while retaining sight of the big picture. The scope of UAF 10 viewpoints (metadata, strategic, operational, services, personnel, resources, security, projects, standards, and actual resources), and extends DoDAF from 56 to 75 views. Table 11a-h shows a mapping between DoDAF, UAF, SysML, and LML views<sup>17</sup> (DoD CIO, 2009; OMG, 2019; OMG, 2020; LML, 2022).

**Table 11a.** Mapping Capability Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
CV-1: Vision	<ul style="list-style-type: none"> <li>St-Sr: Strategic Structure</li> </ul>	UML Use Case Diagram	Asset Diagram
CV-2: Capability Taxonomy	<ul style="list-style-type: none"> <li>St-Tx: Strategic Taxonomy</li> </ul>	Activity Diagram	Hierarchy Diagram
CV-3: Capability Phasing	<ul style="list-style-type: none"> <li>St-Rm: Strategic Roadmap: Phasing</li> </ul>	N/A	Timeline Chart
CV-4: Capability Dependencies	<ul style="list-style-type: none"> <li>St-Cn: Strategic Connectivity</li> </ul>	N/A	Matrix based on <i>Actions</i> and <i>Assets</i>
CV-5: Capability to Organizational Development Mapping	<ul style="list-style-type: none"> <li>St-Rm: Strategic Roadmap</li> </ul>	N/A	Matrix based on <i>Actions</i> and <i>Assets</i>
CV-6: Capability to Operational Activities Mapping	<ul style="list-style-type: none"> <li>Op-Tr: Operational Traceability</li> </ul>	N/A	Matrix based on <i>Actions</i>
CV-7: Capability to Services Mapping	<ul style="list-style-type: none"> <li>Sv-Tr: Services Traceability</li> <li>St-Tr: Strategic Traceability</li> </ul>	N/A	Matrix based on <i>Actions</i> and <i>Assets</i>

<sup>17</sup> Table 10a-h do not address the UAF views for meta-data and security since a DoDAF correlation does not exist.

**Table 11b.** Mapping Operational Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
OV-1: High Level Operational Concept Graphic	<ul style="list-style-type: none"> <li>Op-Tx:Operational Taxonomy</li> <li>Sm-Ov: Summary and Overview</li> </ul>	UML Use Case Diagram	Asset Diagram
OV-2: Operational Resource Flow Description	<ul style="list-style-type: none"> <li>Op-Sr:Operational Structure</li> <li>Op-Tx:Operational Taxonomy</li> <li>Op-Cn:Operational Connectivity</li> </ul>	UML Class Diagram	Asset Diagram
OV-3: Operational Resource Flow Matrix	<ul style="list-style-type: none"> <li>Op-Cn: Operational Conectivity</li> </ul>	N/A	Matrix of <i>Input/Outputs</i>
OV-4: Organizational Relationships Chart	<ul style="list-style-type: none"> <li>Ar-Cn Actual Resources Connectivity</li> <li>Rs-Sr: Actual Resources Structure</li> <li>Pr-Cn:Personnel Connectivity</li> <li>Pr-Ct:Personnel Constratints: Competence</li> </ul>	UML Class Diagram	Asset Diagram
OV-5a: Operational Activity Decomposition Tree	<ul style="list-style-type: none"> <li>Op-Pr:Operational Processes</li> <li>Pr-Sr:Pesonnel Structure</li> <li>Pr-Tx:Personnel Taxonoomy</li> </ul>	UML Class Diagram	Hierarchy Chart
OV-5b: Operational Activity Model	<ul style="list-style-type: none"> <li>Op-Pr:Operational Processes</li> </ul>	Activity Diagram	IDEFD0 Diagram
OV-5b/6c – Action Diagram (Fit-for-purpose)	<ul style="list-style-type: none"> <li>Op-Pr:Operational Processes</li> <li>Op-Is:Operational Interacton Scenarios</li> <li>Pj-Pr: Projects Processes</li> </ul>	N/A	Action Diagram
OV-6a: Operational Rules Model	<ul style="list-style-type: none"> <li>Op-Ct:Operational Constraints</li> </ul>	N/A	N/A
OV-6b: State Transition Description	<ul style="list-style-type: none"> <li>Op-St:Operational-States</li> </ul>	State-machine Diagram	State-Machine Diagram
OV-6c: Event-Trace Description	<ul style="list-style-type: none"> <li>Op-Is:Operational Interaction Scenarios</li> </ul>	Sequence Diagram	Sequence Diagram

**Table 11c.** Mapping Systems Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
SV-1 Systems Interface Description	<ul style="list-style-type: none"> <li>Rs-Tx:Resources Taxonomy</li> <li>Rs-Sr:Resources Structure</li> <li>Ar-Cn: Actual Resource Connectivity</li> </ul>	Block Definition Diagram	Asset Diagram
SV-2 Systems Resource Flow Description	<ul style="list-style-type: none"> <li>Rs-Sr:Resources Structure</li> <li>Rs-Tx:Resource Taxonomy</li> <li>Ar-Cn:Actual Resource Connectivity</li> </ul>	Internal Block Definition Diagram	Asset Diagram
SV-3 Systems-Systems Matrix	<ul style="list-style-type: none"> <li>Rs-Cn:Resources Connectivity</li> </ul>	N/A	Matrix based on <i>Assets</i>
SV-4 Systems Functionality Description	<ul style="list-style-type: none"> <li>Rs-Pr: Resource Processes</li> <li>Pr-Pr: Personnel Processes</li> </ul>	Activity Diagram	IDEF0 Diagram
SV-5a Operational Activity to Systems Function Traceability Matrix	<ul style="list-style-type: none"> <li>Pr-Tr:Personnel Tracability</li> <li>Rs-Tr:Resources Traceability</li> </ul>	N/A	Matrix based on <i>Actions</i>
SV-5b Operational Activity to Systems Traceability Matrix	<ul style="list-style-type: none"> <li>Pr-Tr:Personnel Traceability</li> <li>Rs-Tr:Resources Traceability</li> </ul>	N/A	Matrix based on <i>Actions and Assets</i>
SV-6 Systems Resource Flow Matrix	<ul style="list-style-type: none"> <li>Pr-Cn:Personnel Connectivity</li> <li>Rs-Cn:Resources Connectivity</li> </ul>	N/A	Matrix of <i>Input/Outputs</i>
SV-7 Systems Measures Matrix	<ul style="list-style-type: none"> <li>Pm-Me:Parameters Measurements</li> <li>Pr-Ct:Personnel Constraints: Performance</li> </ul>	Parametrics Diagrams	Matrix based on <i>Measures and Assets</i>
SV-8 Systems Evolution Description	<ul style="list-style-type: none"> <li>Rs-Rm:Resources Roadmap: Evolution</li> </ul>	State-machine Diagram	Timeline Diagram
SV-9 Systems Technology & Skills Forecast	<ul style="list-style-type: none"> <li>Rs-Rm:Resources Roadmap: Forecast</li> </ul>	N/A	Timeline Diagram
SV-10a Systems Rules Model	<ul style="list-style-type: none"> <li>Pr-Ct:Personnel Contrataints: Drivers</li> <li>Rs-Ct:Resources Constraints</li> </ul>	N/A	N/A
SV-10b Systems State Transition Description	<ul style="list-style-type: none"> <li>Rs-St:Resources States</li> </ul>	State-Machine Diagram	State-Machine Diagram
SV-10c Systems Event-Trace Description	<ul style="list-style-type: none"> <li>Rs-Is:Resources Interaction Scenarios</li> </ul>	Sequence Diagram	Sequence Diagram



**Table 11d.** Mapping Services Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
SvcV-1 Services Context Description	<ul style="list-style-type: none"> <li>Rs-Tx:Resource Taxonomy</li> <li>Rs-Sr: Resource Structure</li> <li>Ar-Cn: Actual Resource Connectivity</li> <li>Sv-Sr:Services Structure</li> <li>Services Taxonomy</li> </ul>	Block Definition Diagram	Asset Diagram
SvcV-2 Services Resource Flow Description	<ul style="list-style-type: none"> <li>Rs-Sr: Resource Structure</li> <li>Rs-Tx:Resource Taxonomy</li> <li>Ar-Cn:Actual Resource Connectivity</li> <li>Sv-Sr:Services Structure</li> </ul>	Internal Block Definition Diagram	Asset Diagram
SvcV-3a Systems-Services Matrix	<ul style="list-style-type: none"> <li>Sv-Cn:Services Connectivity</li> </ul>	N/A	Matrix based on <i>Assets</i>
SvcV-3b Services-Services Matrix	<ul style="list-style-type: none"> <li>Sv-Cn:Services Connectivity</li> </ul>	Activity Diagram	IDEF0 Diagram
SvcV-4 Services Functionality Description	<ul style="list-style-type: none"> <li>Rs-Pr: Resource Processes</li> <li>Pr-Pr: Personnel Processes</li> <li>Sv-:Services Processes</li> </ul>	N/A	Matrix based on <i>Actions</i>
SvcV-5 Operational Activity to Services Traceability Matrix	<ul style="list-style-type: none"> <li>Sv-Tr:Services Traceability</li> </ul>	N/A	Matrix based on <i>Action and Asset</i>
SvcV-6 Services Resource Flow Matrix	<ul style="list-style-type: none"> <li>Sv-Cn:Services Connectivity</li> </ul>	N/A	Matrix based on <i>Input/Outputs</i>
SvcV-7 Services Measures Matrix	<ul style="list-style-type: none"> <li>Pm-Me:Parameters Measurements</li> </ul>	Parametrics Diagram	Matrix based on <i>Measures and Assets</i>
SvcV-8 Services Evolution Description	<ul style="list-style-type: none"> <li>Pr-Rm:Personnel Roadmap: Evolution</li> <li>Sv-Rm:Services Roadmap: Evolution</li> </ul>	State-machine Diagram	Timeline Diagram
SvcV-9 Services Technology & Skills Forecast	<ul style="list-style-type: none"> <li>Pr-Rm:Personnel Roadmap: Forecast</li> <li>Sv-Rm:Services Roadmap: Forecast</li> </ul>	N/A	Timeline Diagram
SvcV-10a Services Rules Model	<ul style="list-style-type: none"> <li>Sv-Ct:Services Constraints</li> </ul>	N/A	N/A
SvcV-10b Services State Transition Description	<ul style="list-style-type: none"> <li>Sv-St:Services States</li> </ul>	State-Machine Diagram	State-Machine Diagram
SvcV-10c Services Event-Trace Description	<ul style="list-style-type: none"> <li>Pr-Is:Personnel Interaction Scenarios</li> <li>Sv-Is:Services Sequences</li> </ul>	Sequence Diagram	Sequence Diagram

**Table 11e.** Mapping All Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
<b>All Viewpoint</b>			
AV-1: Overview and Summary Information	<ul style="list-style-type: none"> <li>Sm-Ov: Summary and Overview</li> </ul>	N/A	<i>Statements</i>
AV-2: Integrated Dictionary	<ul style="list-style-type: none"> <li>Dc: Dictionary</li> </ul>	N/A	<i>Statements</i>

**Table 11f.** Mapping Project Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
PV-1: Project Portfolio Relationships	<ul style="list-style-type: none"> <li>PI-St: Project Structure</li> </ul>	N/A	Matrix based on <i>Time</i> and <i>Assets</i>
PV-2: Project Timelines	<ul style="list-style-type: none"> <li>Pr-Rm: Personnel Roadmap: Availabiliy</li> <li>Pj-Cn: Projects Connectivity</li> <li>Pj-Rm: Projects Roadmap</li> </ul>	N/A	Timeline Chart
PV-3: Project to Capability Mapping	<ul style="list-style-type: none"> <li>Pj-Tr: Projects Tracability</li> </ul>	N/A	Matrix based on <i>Time</i> and <i>Actions</i>

**Table 11g.** Mapping Data and Information Views between DoDAF, UAF, SysML, and LML.

DoDAF View	UAF View	UML/SysML Diagram	LML Model
DIV-1: Conceptual Data Model	<ul style="list-style-type: none"> <li>If: Information</li> </ul>	UML Class Diagram, Block Definition Diagram	<i>Actions, Assets, and Inputs/Outputs</i>
DIV-2: Logical Data Model	<ul style="list-style-type: none"> <li>If: Information</li> </ul>	UML Class Diagram, Block Definition Diagram	<i>Actions, Assets, and Inputs/Outputs</i>
DIV-3: Physical Data Model	<ul style="list-style-type: none"> <li>If: Information</li> </ul>	UML Class Diagram, Block Definition Diagram	<i>Actions, Assets, and Inputs/Outputs</i>

**Table 11h.** Mapping Standards Views between DoDAF, UAF, SysML, and LML.

<b>DoDAF View</b>	<b>UAF View</b>	<b>UML/SysML Diagram</b>	<b>LML Model</b>
StdV-1 Standards Profile	Sd-Tr: Standards Traceability	N/A	<i>Statements</i>
StdV-2 Standards Forecast	Sd-Rm: Standards Roadmap	N/A	Timeline Chart

Architecture frameworks should be extended into presentation frameworks to include data that is relevant across the system lifecycle. The goal is to implement MBSE, which is greater than “model-based system architecting,” therefore, new viewpoints and views need to be defined to represent the full systems engineering lifecycle. Table 12 suggests new viewpoints and views to be included in a presentation framework. Table 13 shows the assessment and forecast of presentation frameworks to the MBSE effectiveness measures.

**Table 12.** Suggested Viewpoints and Views for the Presentation Framework.

Viewpoint	Views
Detailed Design	<ul style="list-style-type: none"> <li>• Computer Aided Design (CAD)</li> <li>• Computer Aided-Engineering (CAE)</li> </ul>
Maintainability Analysis	<ul style="list-style-type: none"> <li>• Modularity Diagrams</li> <li>• Dependency Graphs</li> <li>• Action/Activity Diagram</li> <li>• Matrix</li> </ul>
Performance Analysis	<ul style="list-style-type: none"> <li>• Scenario Analysis</li> <li>• Simulations</li> <li>• Flow Chart</li> <li>• Bar Charts</li> <li>• Pie Charts</li> <li>• Dashboards</li> <li>• Matrix</li> </ul>
Prototypes	<ul style="list-style-type: none"> <li>• Mock-ups</li> <li>• Interactive Prototypes</li> <li>• “Non-qualified” components</li> </ul>
Reliability Analysis	<ul style="list-style-type: none"> <li>• Fault Tree Analysis (FTA) Diagram</li> <li>• Reliability Block Diagrams</li> <li>• Event Tree Analysis (ETA)</li> <li>• Pareto Charts</li> <li>• Matrix</li> </ul>
Requirements Analysis	<ul style="list-style-type: none"> <li>• SysML Requirements Diagram</li> <li>• Hierarchy Chart</li> <li>• Sequence Diagrams</li> <li>• System Context Diagrams</li> <li>• Process Maps and Flowcharts</li> <li>• Functional Decomposition Diagrams</li> </ul>
Risk Analysis	<ul style="list-style-type: none"> <li>• Risk Matrix</li> <li>• Fault Tree Analysis (FTA) Diagram</li> <li>• Event Tree Analysis (ETA)</li> <li>• Scenario Analysis</li> </ul>
Training	<ul style="list-style-type: none"> <li>• Interactive Simulations</li> <li>• Storyboards</li> <li>• Mind Maps</li> </ul>
Verification and Validation (Testing)	<ul style="list-style-type: none"> <li>• Test Execution Dashboards</li> <li>• Error Tracking Charts</li> <li>• Dependency Graphs</li> <li>• Performance Monitoring Charts</li> </ul>

**Table 13.** Presentation Frameworks Assessment and Forecast.

Effectiveness Measure	Presentation Framework
Expressiveness	Presentation frameworks provide the definitions, references, guidance and rules for structuring, classifying, and organizing architectures. Current frameworks focus on the early systems engineering phases. Presentation frameworks should be extended to include data that is relevant across the system lifecycle (e.g. requirements, risk, test and evaluations, programmatic data).
Precise	Complexity in a model-based environment is significantly reduced by separating and characterizing systems issues into various data-driven viewpoints and views.
Presentation/ Communication	Presentation frameworks promote familiarity and allow it to be easier to compare and contrast models when in the same format. However, after three decades of formal frameworks, wide-spread decision-making based on the models has not been realized. The improvement of the presentation and communication of models based on the frameworks is not a technological issue, but a human issue where decision-makers must understand how the models can be used to make decisions.
Model Construction/ Manageability	Presentation frameworks establish standard views, descriptions, data structures, and generic approach to developing the models. Since presentation frameworks are implemented in MBSE tools, the ease of model construction and manageability is tool dependent.
Interoperable/ Logistical Consistency	Interoperability and logical consistency of presentation frameworks is based on modeling languages and model structures; therefore, the presentation frameworks have the same characteristics.
Usable	The presentation frameworks establish standard views, descriptions, data structures, and a generic approach to developing the models. This commonality fosters familiarity among all stakeholders.
Concordance/ Referential Integrity	When visualization models are derived from the data, concordance will likely result. While visualization models within a viewpoint are related, visualization models from across the viewpoints is also defined. These relationships exist because relationships exists within the data, thus ensuring concordance.

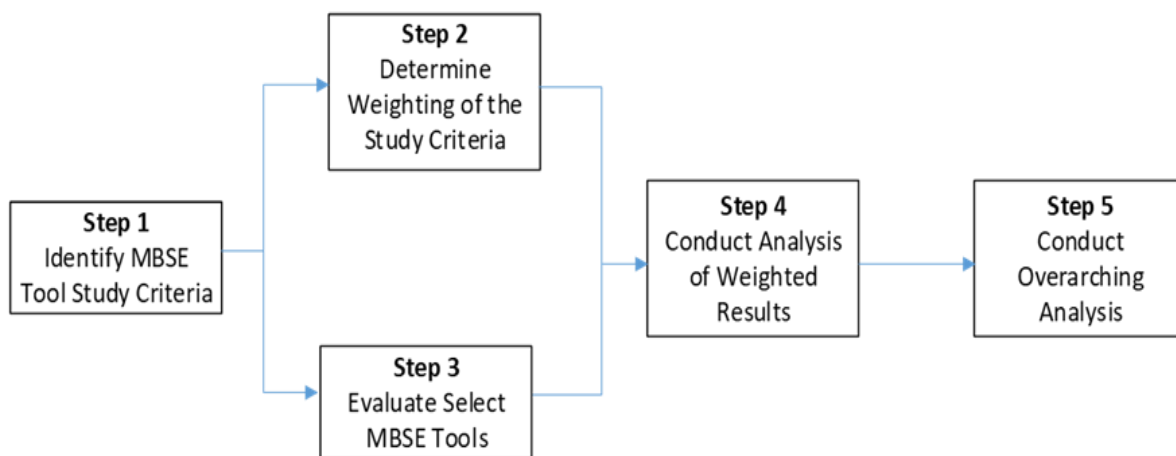
## CHAPTER 7 MODELING TOOLS SELECTION

---

*“All models are wrong; some models are useful.”* – W. Edwards Deming

Model-Based Systems Engineering tools are general purpose software applications that use modeling languages, and support the specifications, to generate, modify, and manage the system model and complex system representations therein. The tools are represented in the center of the Venn diagram (Fig. 2) because each tool has different strengths and weaknesses with respect to the four MBSE components. Each organization has a different program focus, the engineering staff has different modeling skills, and managers want to see the system data portrayed in a way that maximizes the decision-making process. The “bottom-line” of MBSE tools is, “one size does not fit all.”

The goal of the tool selection process is to explore the trade space (maximum effectiveness area in Fig. 2) that exists between the four MBSE components, using a multi-criteria approach. By evaluating these tools within this trade-space, organizations can understand which MBSE tool best meets their systems engineering needs. Fig. 11 shows the five-step process used in the tool selection process.



**Fig. 11.** Five Step Process used for MBSE Tool Study (Vaneman, *et al.*, 2021).

The first step uses the four components of MBSE are further defined by deriving additional criteria that is specific to the organization. This also defines a rubric for evaluation of the criteria. The rubric served as a guide for the tool evaluation team while evaluating each tool and will be further discussed in Step 3. The general scale for the rubric is:

- 1 = Tool does not meet the criteria
- 3 = Tool partially meets the criteria
- 5 = Tool fully meets the criteria

While each MBSE component will be further defined, and an associated rubric created for illustrative purposes on the analysis of the modeling process component will be shown. Modeling processes provide the analytical framework to conduct the analysis of the system virtually defined in the model, throughout all phases of the systems engineering lifecycle. Tables 14a-b show the criteria and rubric defined for modeling processes focus on the types of models developed during the systems engineering and architecture process, and include the ability to model:

- Operational architectures;
- Physical architectures;
- Architecture planning;
- Program management;
- System requirements;
- Discrete event simulations;
- Stochastic simulations;
- System dynamics;
- Verification & validation;
- Entire system lifecycle to include all lifecycle disciplines.

**Table 14a.** Model-Based Processes Criteria and Rubric.

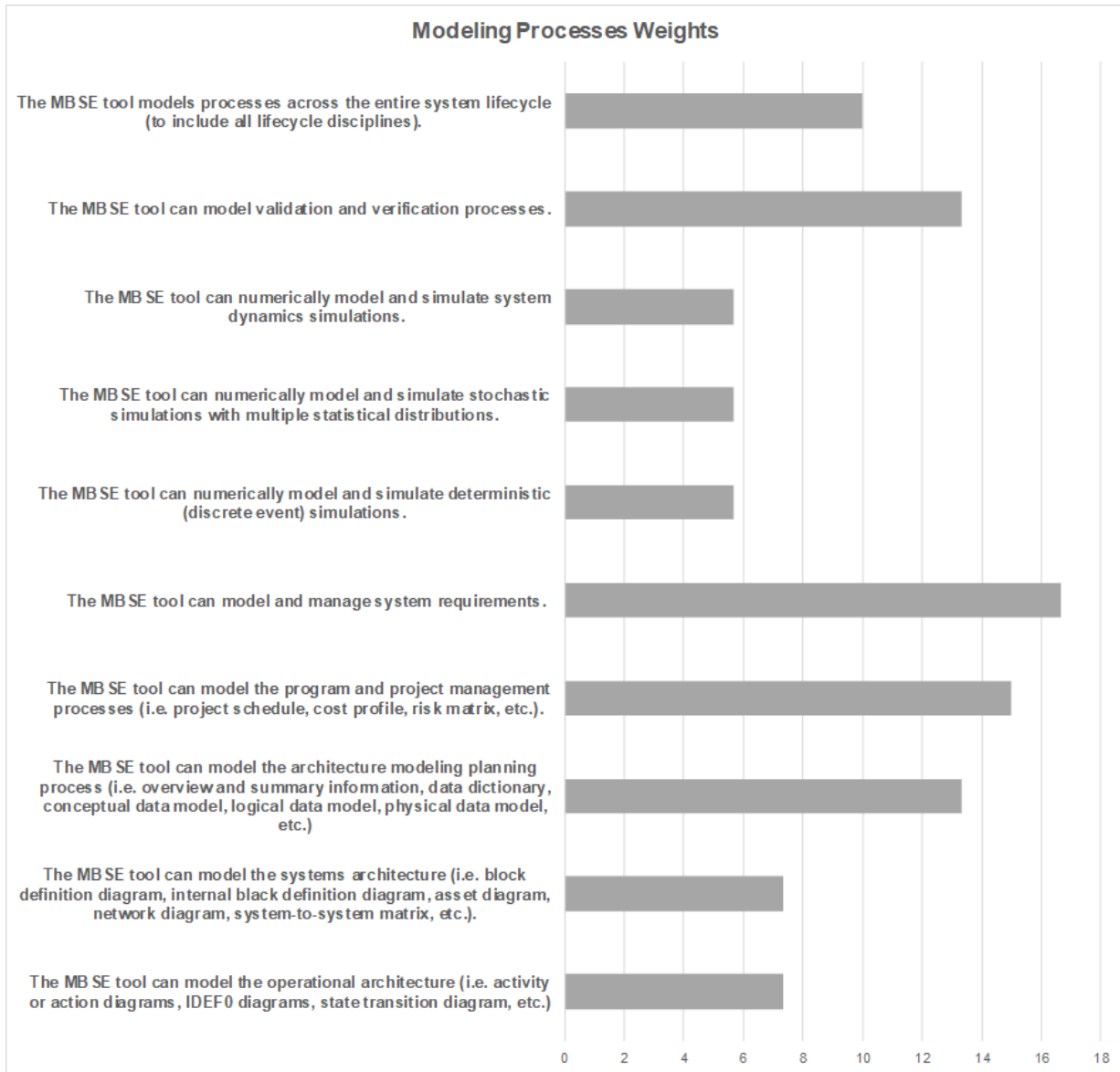
Criteria	Rubric
The MBSE tool can model the operational architecture (i.e. activity or action diagrams, IDEF0 diagrams, state transition diagram, etc.)	1 = The MBSE tool cannot model operational architectures; 5 = The MBSE tool can model operational architectures.
The MBSE tool can model the systems architecture (i.e. block definition diagram, internal block definition diagram, asset diagram, network diagram, system-to-system matrix, etc.).	1 = The MBSE tool cannot model operational architectures; 5 = The MBSE tool can model operational architectures.
The MBSE tool can model the architecture modeling planning process (i.e. overview and summary information, data dictionary, conceptual data model, logical data model, physical data model, etc.)	1 = The MBSE tool does not model any aspect of the architecture modeling planning process; 3 = The MBSE tool partially models the architecture modeling planning process, but does not include all aspects (i.e. overview and summary information, data dictionary, conceptual data model, logical data model, physical data model, etc.); 5 = The MBSE tool fully models the architecture modeling planning process.
The MBSE tool can model the program and project management processes (i.e. project schedule, cost profile, risk matrix, etc.).	1 = The MBSE tool cannot model program and project management processes; 5 = The MBSE tool can model program and project management processes.
The MBSE tool can model and manage system requirements.	1 = The MBSE tool does not model or manage system requirements; 3 = The MBSE tool does model system requirements, but does not allow for those requirements to be managed within the tool (i.e. the system requirements can be captured and portrayed within the MBSE tool, but cannot manage requirements); 5 = The MBSE tool models and manages requirements (i.e. managing requirements means that requirements can be managed within the MBSE tool, or has a direct, real-time, two-way interface with another tool that will manage requirements (e.g. DOORS)).
The MBSE tool can numerically model and simulate deterministic (discrete event) simulations.	1 = The MBSE tool cannot numerically model and simulate discrete event simulations; 5 = The MBSE tool can numerically model and simulate discrete event simulations.



**Table 14b.** Model-Based Processes Criteria and Rubric.

Criteria	Rubric
The MBSE tool can numerically model and simulate stochastic simulations with multiple statistical distributions.	1 = The MBSE tool cannot numerically model and simulate stochastic simulations; 5 = The MBSE tool can numerically model and simulate stochastic event simulations.
The MBSE tool can numerically model and simulate system dynamics simulations.	1 = The MBSE tool cannot numerically model and simulate system dynamics simulations; 5 = The MBSE tool can numerically model and simulate system dynamics simulations.
The MBSE tool can model validation and verification processes.	1 = The MBSE tool cannot model validation and verification processes; 5 = The MBSE tool can model validation and verification processes.
The MBSE tool models processes across the entire system lifecycle (to include all lifecycle disciplines).	1 = The MBSE tool models processes for only a certain portion of the system lifecycle (e.g., architecture development phase); 3 = The MBSE tool models the entire system lifecycle, but only from a systems engineering perspective (but not other lifecycle disciplines such as program and project management); 5 = The MBSE tool models the entire system lifecycle (to include all lifecycle disciplines).

Each criteria defined in Step 1 may have a different importance to different organizations. In the second step, the organization determines the weighted priority for criteria. One method is to apply a “hundred-coin exercise” to determine the individual criteria within the components. In a “hundred-coin exercise,” the organization is asked to evaluate each criteria as if they had a “hundred coins,” placing the corresponding “number of coins” against the importance of each criteria. If there are multiple respondents, the results are averaged, with the result representing the importance of each criteria to the respondents collectively. The modeling process criteria weights are shown in Fig. 12.



**Fig.12.** Modeling Process Weights (Vaneman, *et al.*, 2021).

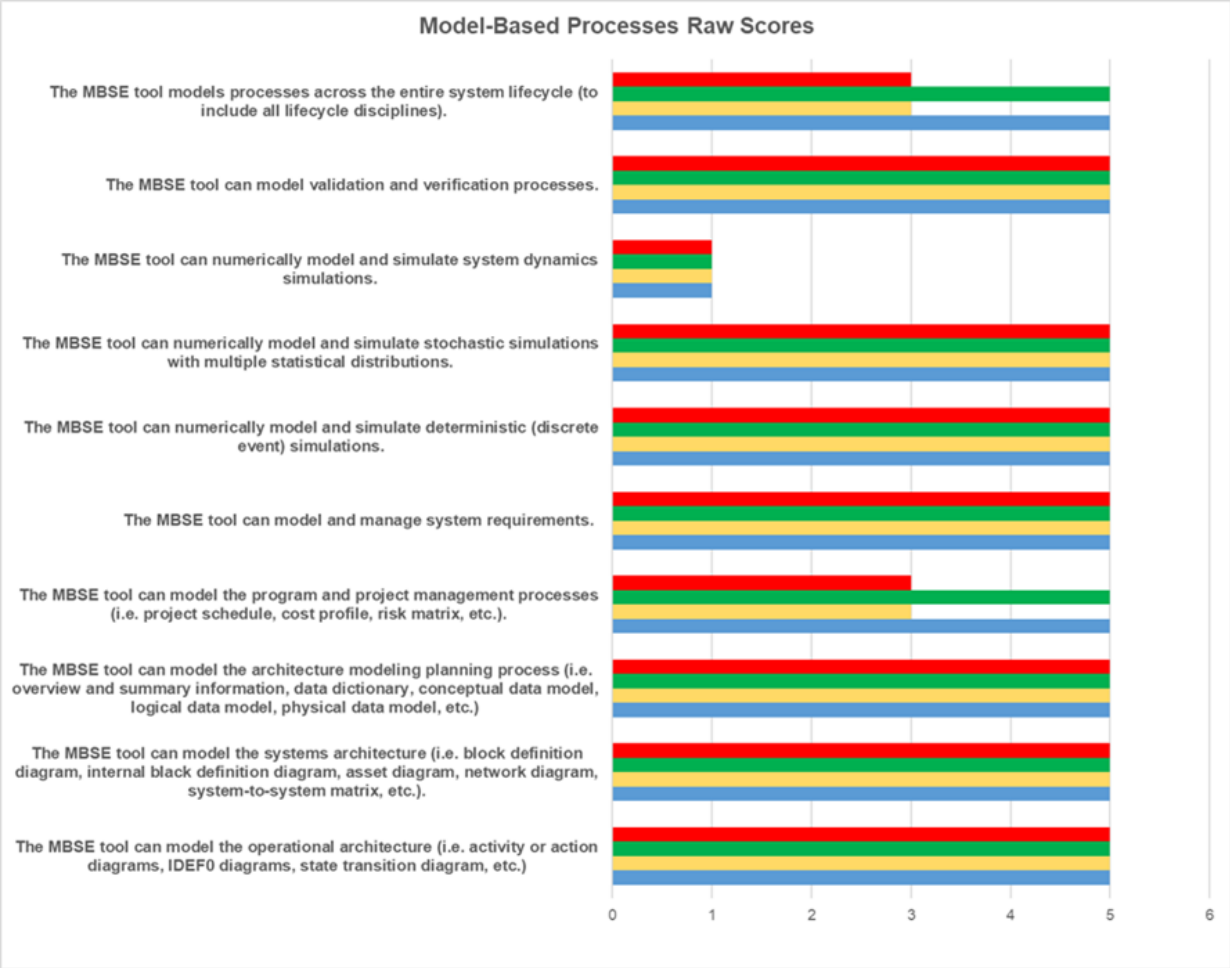
Four modeling process criteria emerged as the most important to the organization. The most important criteria is that the MBSE tool can model and manage system requirements. The modeling processes places a high emphasis on the ability of the MBSE tool to model program and project management processes. Two additional processes that were significantly weighted and are also fundamental to systems engineering, are:

- The tool can model validation and verification processes.
- The tool model the architecture modeling planning process (i.e. overview and summary information, data dictionary, conceptual data model, logical data model, physical data model, etc.).

In the third step, the tool reviewers evaluate each candidate tool against the defined criteria using the rubric. The candidate tools are assumed to have been qualified as being approved by the organization, with this evaluation selecting the best tool, from the approved tools, for the required modeling task.

To maintain an unbiased tool evaluation, it is important that the tool reviewers don't know the weights determined by the organization in Step 2, and that those determining the weights don't know how each tool is being evaluated. Fig. 13 shows the raw scores for the four candidate tools (each bar color represents a separate tool).

All tools were evaluated satisfactorily against most of the criteria. One notable exception is all the tools scored poorly against the criteria, "The MBSE tool can numerically model and simulate system dynamics simulations." This is not a major concern since systems dynamics modeling is not widely used in most program offices. While each tool developer likes to boast about their tool's capabilities, no single tool does everything, and certainly not everything well.



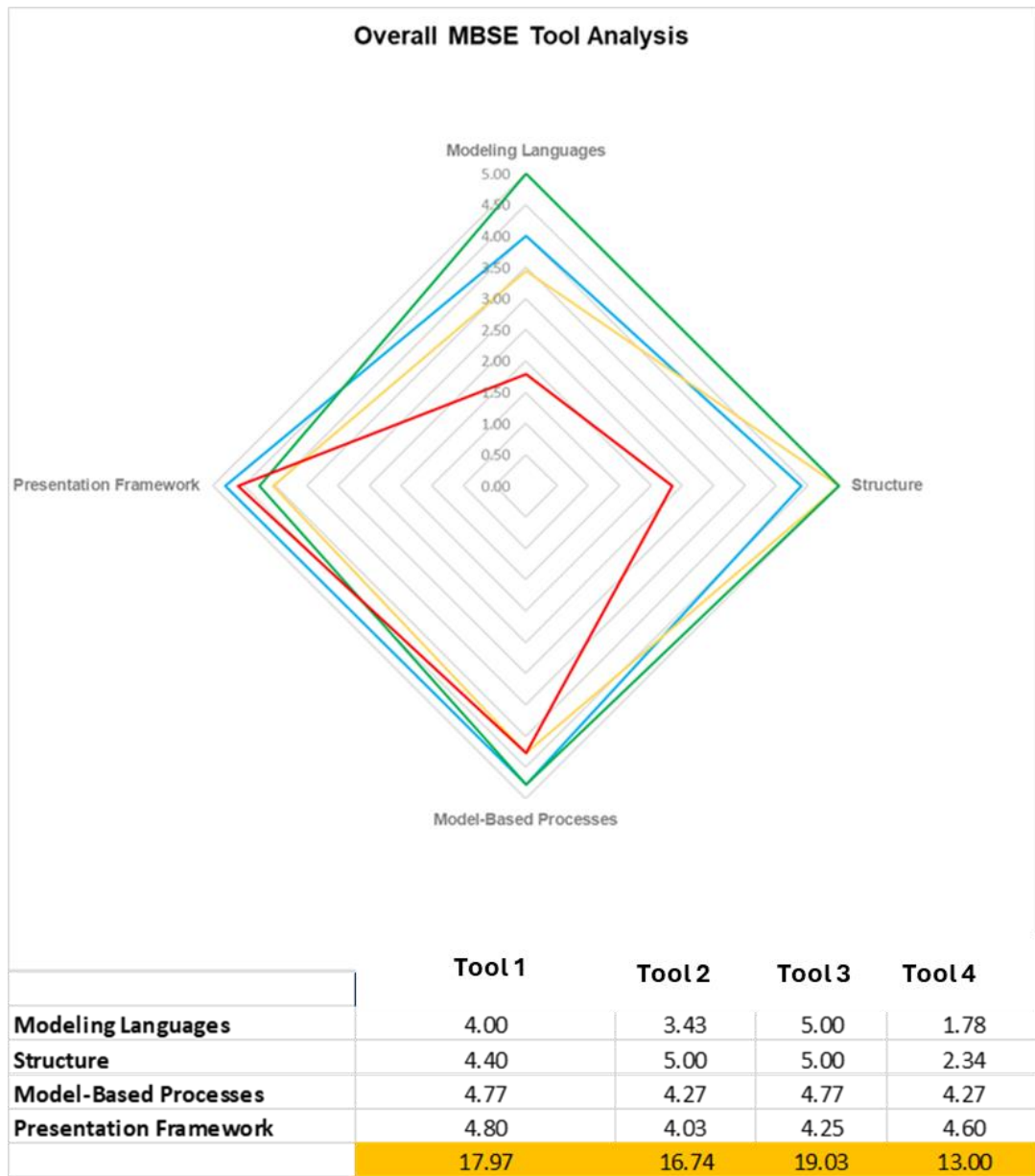
**Fig. 13.** Model-Based Processes Raw Scores (Vaneman *et al.*, 2022).

The fourth step brings together the weighted criteria as defined by the organization and the tool evaluation. The weighted criteria defined in Step 2 multiplied by the tool evaluation scores in Step 3. The purpose of this step is to identify which tool is best for a given criteria as defined by the organization. Since each organization is likely to have different weights to match their organizational needs, the weighted scores will better differentiate which tool best meets their needs. Table 15 shows the modeling process weighted scores (Vaneman, *et al.*, 2021).

**Table 15.** Modeling Processes Weighted Scores (Vaneman *et al.*, 2022).

Criteria	Weight	#1 Score	#1 Weighted Score	#2 Score	#2 Weighted Score	#3 Score	#3 Weighted Score	#4 Score	#4 Weighted Score
The MBSE tool can model the operational architecture (i.e. activity or action diagrams, IDEF0 diagrams, state transition diagram, etc.)	0.07	5	0.37	5	0.37	5	0.37	5	0.37
The MBSE tool can model the systems architecture (i.e. block definition diagram, internal block definition diagram, asset diagram, network diagram, system-to-system matrix, etc.).	0.07	5	0.37	5	0.37	5	0.37	5	0.37
The MBSE tool can model the architecture modeling planning process (i.e. overview and summary information, data dictionary, conceptual data model, logical data model, physical data model, etc.)	0.13	5	0.67	5	0.67	5	0.67	5	0.67
The MBSE tool can model the program and project management processes (i.e. project schedule, cost profile, risk matrix, etc.).	0.15	5	0.75	3	0.45	5	0.75	3	0.45
The MBSE tool can model and manage system requirements.	0.17	5	0.83	5	0.83	5	0.83	5	0.83
The MBSE tool can numerically model and simulate deterministic (discrete event) simulations.	0.06	5	0.28	5	0.28	5	0.28	5	0.28
The MBSE tool can numerically model and simulate stochastic simulations with multiple statistical distributions.	0.06	5	0.28	5	0.28	5	0.28	5	0.28
The MBSE tool can numerically model and simulate system dynamics simulations.	0.06	1	0.06	1	0.06	1	0.06	1	0.06
The MBSE tool can model validation and verification processes.	0.13	5	0.67	5	0.67	5	0.67	5	0.67
The MBSE tool models processes across the entire system lifecycle (to include all lifecycle disciplines).	0.10	5	0.5	3	0.30	5	0.50	3	0.30
Sum	1		4.77		4.27		4.77		4.27

The fifth step recognizes that each component may not be of equal importance to the users. As such, the “hundred-coin exercise” is repeated for the four MBSE components. In this example, each of the four components are equally weighted. The overarching results are then compared using a radar chart (Fig. 14) as a visual representation to show the relative strengths of each tool with respect to the four MBSE components (Vaneman, *et al.*, 2021).



**Fig. 14.** Overall MBSE Tool Analysis Results (equally weighted) (Vaneman *et al.*, 2021).

The criteria weightings were unique to the organization. However, the criteria can be weighted by other organizations to derive what is important to them. One thing that is certain is different organizations will have different MBSE needs. If the criteria are accepted by another organization, the tool evaluation step is not necessary to redo if the same tools are being considered. The tool evaluation can be applied to the new organizations weights and re-evaluated in Steps 4 and 5.

Likewise, if the organization wants to evaluate other tools, the weighting will not have to be re-done because they already know what is important to them. The new tool(s) can be independently evaluated, and then can be evaluated against Steps 4 and 5.

## CHAPTER 8 EPILOGUE

---

*“Often the largest single cause of an enterprise failing to adopt or shift to a new paradigm is the inability or refusal to see beyond current patterns of thought, behavior, structures, rules and assumptions.” – Kevin Brett (2022)*

Despite the almost two-decade emphasis placed on MBSE, the discipline has failed to keep pace with the ever-increasing complexities of systems. Over-reliance on an extremely limited number of modeling languages and tools, the failure to address all lifecycle disciplines, and the focus on modeling mechanics instead of modeling the system, has resulted in many model-based activities being relegated to the beginning of the system’s lifecycle, thereby neglecting the holistic approach that is the cornerstone of the systems engineering discipline.

The principles of MBSE are the foundation of model-based approaches SoI. To further the modeling approaches, the concept of digital engineering is being pursued. Digital engineering is an integrated digital approach that uses authoritative sources of system’s data to create an interconnected, federation of models that integrates engineering models, from various engineering domains, as a holistic representation of all relevant system perspectives, to design, develop, analyze, test, optimize, and manage complex systems throughout the system’s lifecycle.

Three goals require the successful understanding and implementation of MBSE for digital engineering to be effective. First, effective digital engineering requires formalized model development to establish the virtual representation of the SoI which was conceptually presented in Fig.1 (DASD, 2018). This approach is significantly different than document-based engineering where each entity can be represented multiple times, there is no concordance between entities, and emergent behavior cannot be achieved. Digital engineering is a paradigm shift and requires a mindset change in engineering processes and expectations of the artifacts required during the system’s lifecycle. The failure to recognize this paradigm is one of the key inhibitors to effective digital engineering and MBSE implementation.



Second, the establishment of an authoritative source of truth (ASoT), which thoroughly qualifies the best models to represent a given aspect of the SoI (DASD, 2018). If an ASOT is going to be effective each entity has a singular representation. Multiple representations will further propagate the problems encountered in document-based engineering. To achieve an ASoT, there must be an understanding of how the data in the models are vetted from an authoritative source and are properly verified, validated, and accredited (VV&A)<sup>18</sup> to represent the SoI<sup>19</sup>. The CDM is a good tool to understand how the models, representing different components, processes, or disciplines interact to form a cohesive, and correlated, data structure that depicts how data is shared within the ASoT.

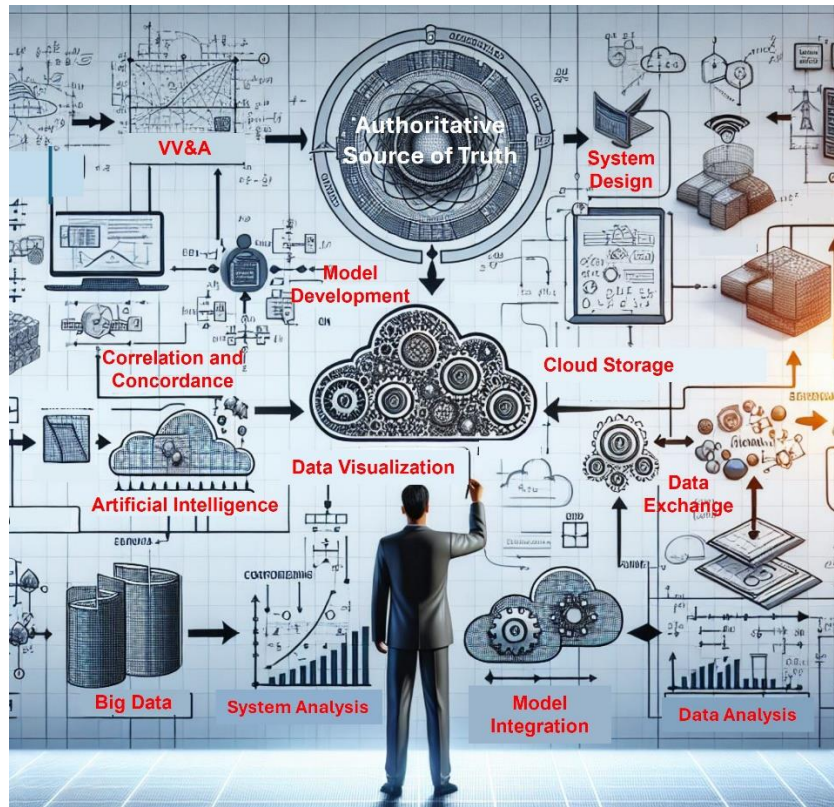
As digital engineering, and the ASoT evolves, technical approaches such as artificial intelligence, big data, and cloud and edge computing will have to be adopted and embraced to address scalability, data and model interoperability, and data management. Fig. 15 is a conceptual diagram showing the various aspects of a future data-driven engineering ecosystem. Notice that the human is central to this concept, representing the human-technology fusion that is essential to fully realizing the digital engineering paradigm shift.

---

<sup>18</sup> Verification, validation, and accreditation (VV&A) are three interrelated, but distinct processes used to ensure the credibility and reliability of models and simulations. Here's a brief overview of each ((AcqNotes, n.d.):

- Verification: This process determines whether a model's implementation and its associated data accurately represent the developer's conceptual description and specifications. Essentially, it's about checking if the model was built correctly.
- Validation: This process assesses whether a model and its associated data provide an accurate representation of the real world from the perspective of the intended uses of the model. It's about ensuring the model behaves as expected in real-world scenarios.
- Accreditation: This is the official certification that a model, simulation, or a federation of models, along with its associated data, is acceptable for use for a specific purpose. It confirms that the model meets the necessary standards and requirements.

<sup>19</sup> Verification and validation are well-known processes in modeling, simulation, and analysis, but are mostly absent in systems engineering. If the data in the ASoT is going to be "authoritative," the V&V processes must be established for systems engineering models individually and for federated models collectively.



**Fig. 15.** Data-Driven Engineering Concept.

The third goal is to exchange and integration of models and data sources between disparate system lifecycle disciplines, modeling languages, tools, and presentation frameworks, while maintaining and storing its native format (DASD, 2018). The focus on modeling artifacts, vice data, makes the exchange and integration of data impossible. An often-ignored common denominator between the model-based engineering approaches is that data is the foundation of all models. The translation and mapping of data represented by various modeling-languages and presentation frameworks is the foundation of sharing data. Often, modeling languages and presentation frameworks are thought to be mutually exclusive with organizations failing to realize that each describes the SoI slightly differently with different syntax and semantics, yet they are describing the same thing.

MBSE De-Mystified is more than an attempt to make sense of a topic that is more complex than recognized, but it is a call to actions that requires fundamental to realize the benefits of this exciting model-based renaissance.

## REFERENCES

---

- AcqNotes n.d., The Defense Acquisition Encyclopedia.  
<https://acqnotes.com/acqnote/tasks/verification-validation-and-accreditation> (Accessed December 25, 2024)
- Bajaj, Manasm Sanford Friedenthal and Ed Seidewitz. 2022. Systems Modeling Language (SysML v2) Support for Digital Engineering. INSIGHT March 2022 Volume 25 Issue 1.
- Booch, Grady. James Rumbaugh,, and Ivar Jacobson 2005. The Unified Modeling Language User Guide (2 ed.). Boston, MA: Addison-Wesley.
- Brett, Kevin. 2022. The Bionic Enterprise: Architecting the Intelligent Society of the Future. Stafford, VA: Blue Eagle Books.
- Dam, Steven H. 2015. DoD Architecture Framework 2.02: A Guide to Applying Systems Engineering to Develop Integrated, Executable Architectures. Manassas, VA: SPEC Innovations
- Dam, Steven H. 2019. Real MBSE. Manassas, VA: SPEC Innovations
- Department of Defense Chief information Officer. 2009. DoD Architecture Framework Version 2.0. Washington, D.C. Department of Defense Chief Information Officer.
- Friedentahl, Sanford and Roger Burkhart. 2015. Evolving SysML and the System Modeling Environment to Support MBSE. INSIGHT, v18, n2, pp. 39-41.
- Friedentahl, Sanford. Alan Moore, and Rick Steiner 2015. A Practical Guide to SysML. Boston: MA: Morgan Kaufmann.
- Hause, Matthew. 2012. UAF – Unified Profile for DoDAF/MODAF. Unpublished presentation.
- International Council on Systems Engineering (INCOSE). 2007. Systems Engineering Vision 2020.
- Lifecycle Modeling Language Steering Committee. 2022. Lifecycle Modeling Language (LML) Specification. Document URL: <http://www.lifecyclemodeling.org/spec/1.4>.
- Lifecycle Modeling Language Steering Committee. 2025 (pending). Lifecycle Modeling Language (LML) 2.0.
- Long, David and Zane Scott. 2011. A Primer for Model-based Systems Engineering (2nd Edition). Blacksburg, VA: Vitech Corporation.

- National Aeronautical and Space Administration, 2021. NASA Space Mission Architecture Framework (SMAF) Handbook for Uncrewed Space Missions. NASA-HDBK-1005.
- Office of Management and Budget n.d., Federal Enterprise Architecture (FEA).  
<https://obamawhitehouse.archives.gov/omb/e-gov/fea> (Accessed April 5, 2024)
- Object Management Group (OMG). 2012. OMG Systems Modeling Language (OMG SysML).  
[www.omg.org/spec/SysML/20120401/SysML.xml](http://www.omg.org/spec/SysML/20120401/SysML.xml).
- Object Management Group (OMG), 2019. Unified Architecture Framework Traceability Version 1.1.
- Object Management Group (OMG), 2020. Unified Architecture Framework Version 1.1.
- Object Management Group (OMG). 2014. OMG Unified Modeling Language (OMG UML).  
<http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF>.
- Object Management Group (OMG). 2023.. Kernel Modeling Language.  
<https://www.omg.org/spec/KerML/1.0/Beta1/About-KerML>
- Office of the Deputy Assistant Secretary of Defense for Systems Engineering (DASD SE). 2018, June. “Department of Defense Digital Engineering Strategy.” Department of Defense.
- SEBoK. n.d. Complexity. <https://sebokwiki.org/wiki/Complexity> (accessed December 4, 2024).
- Vaneman, Warren K. 2016. Enhancing Model-Based Systems Engineering with the Lifecycle Modeling Language. Orlando, FL: Proceedings of the 10th Annual IEEE Systems Conference.
- Vaneman, Warren K. 2018. Evolving Model-Based Systems Engineering Ontologies and Structures. Washington, D.C.: Proceedings of the 29th Annual INCOSE International Symposium.
- Vaneman, Warren K. and Ronald Carlson. 2018. Managing Complex Systems Engineering and Acquisition Through Lead Systems Integration. Monterey, CA: Naval Postgraduate School. NPS-AM-19-008.
- Vaneman, Warren K., Ronald R. Carlson, and Gary W. Parker. 2021. Model Based Systems Engineering Tool Study Report. Monterey, CA: Naval Postgraduate School. Unpublished technical report.
- Vaneman, Warren K., Steven Dam, and Jerry Sellers. 2019. Essential LML: a Thinking Tool for Capturing, Connecting and Communicating Complex Systems.
- Vaneman, Warren K., Corina White, Ronald r. Carlson, Christopher Wolfgeher, Christopher Ritter, Geraldo Rodriguez Melo, Henry Sulca. 2023. Implementing A Model-Based

Systems Engineering Land Construct for the Marine Corps. Monterey, CA: Naval Postgraduate School. NPS-SE-23-004.

Wikipedia n.d . Booch Method. [https://en.wikipedia.org/wiki/Booch\\_method](https://en.wikipedia.org/wiki/Booch_method) Accessed March 28, 2024)

Zachman, John A. 2008. The Concise definition of the Zachman Framework. Zachman International, Inc. <https://www.zachman.com/about-the-zachman-framework>. (Accessed Feb 28, 2019)

Zachman, John A. n.d. About the Zachman Framework. Zachman International, Inc. <https://www.zachman.com/about-the-zachman-framework>. (Accessed Aug 14, 2023)

## MEET THE AUTHOR

---

**Dr. Warren Vaneman** is an Expert Systems Engineering Professional, certified by the International Council on Systems Engineering (INCOSE), who serves as Professor of Practice in the Systems Engineering Department at the Naval Postgraduate School, Monterey, CA, since 2012, and as an Adjunct Professor at Virginia Tech, Blacksburg, VA, since 2007. He has four decades of leadership and technical positions within the U.S. Navy and the Intelligence Community. Dr. Vaneman conducts research and teaches courses in systems engineering, Model-Based Systems Engineering (MBSE), Systems Architecting, and Systems of Systems Engineering and Integration. Prior to joining NPS, Dr. Vaneman held various systems engineering positions within the Intelligence Community, including Chief, Architecture Analysis Division, and Chief Architect of the Enterprise Ground Architecture at the National Reconnaissance Office (NRO), and Lead Systems Engineer for Modeling and Simulation at the National-Geospatial Intelligence Agency (NGA).

Dr. Vaneman is also a Retired Navy Reserve Captain, where he qualified as a Surface Warfare Officer, Information Dominance Warfare Officer, and Space Cadre Expert. He served as the navigator of a guided missile cruiser during Desert Storm, NRO Liaison to U.S. Pacific Command, Flag Advisor for Systems and Space Engineering for the Space and Naval Warfare Systems Command Chief Engineer, and had the honor of six command tours, including a command tour in Afghanistan during Operation Enduring Freedom, and as the Commanding Officer, and Deputy Director for Intelligence at U.S. Fleet Forces Command.

Dr. Vaneman has a B.S. in Meteorology and Oceanography from the State University of New York Maritime College, an M.S. in Systems Engineering and a Ph.D. in Industrial and Systems Engineering from Virginia Tech, and a Joint Professional Military Education Phase 1 Certificate from the Naval War College. He is the recipient of several engineering and academic awards and honors including the Virginia Tech Outstanding Dissertation Award (2003) and induction into the Virginia Tech Industrial and Systems Engineering Department's Academy of Distinguished Alumni (2019). Dr. Vaneman is a proudly decorated Naval Officer with 15 personal awards including two Legions of Merit.