Model-Based Systems Engineering (MBSE)

Lead Author: *Caitlyn Singam and Jeffrey Carter*

Model-based Systems Engineering [MBSE] is a paradigm that uses formalized representations of systems, known as models, to support and facilitate the performance of Systems Engineering [SE] tasks throughout a system's life cycle. MBSE is frequently contrasted with legacy document-based approaches where systems engineering captures system design information via multiple independent documents in various non-standardized formats. MBSE consolidates of system information in system design models, which provide primary SE artifacts. These system models, which are generally expressed in a standardized modelling language such as Systems Modeling Language [SysML®] express key system information in a concise, consistent, correct, and coherent format. When implemented properly, MBSE models permit the standardized consolidation and integration of system knowledge across engineering disciplines and subsystems and streamline key systems engineering tasks while also minimizing developmental risk.

This article provides an overview of key concepts underlying model-based approaches to systems engineering and highlights the benefits of utilizing MBSE on projects.

Contents

System Models
    Definition of a model

- Model Properties
- Criteria for Effective MBSE Models
- Modeling Languages
- Architecture Frameworks
- Process Frameworks
- Benefits of MBSE
  - Digital Transformation
  - Digital Twins
- MBSE versus DBSE
- References
  - Works Cited
  - Primary References
  - Additional References

System Models

During the systems engineering process, a substantial amount of information is collected, generated, and/or maintained regarding the characteristics of the system(s) of interest, composite elements, and interacting entities/environments. MBSE utilizes models as a means of aggregating and managing these disparate pieces of information about a system in a centralized repository that can serve as a 'single source of truth' and technical baseline regarding a system of interest.

Definition of a model

Models are representations that are used to capture, analyze, and/or communicate information about a system or concept. They can vary in scope, purpose, and type, and can be utilized both individually as stand-alone entities as well as in concert with each other as part of an integrated set (Wymore 1993).

Model Properties

A model can be described and classified with respect to the following properties:

- Scope: the range of relevance of a model. Models can range from capturing the characteristics and interactions of a system's components (broad scope), to only focusing on the form and function of a single

element in isolation (narrow scope).

- Domain: the ‘lens’ through which the model views a system. Models can be holistic in nature or can focus on only highlighting information relevant to certain domains. Domain-specific models generally are used to highlight certain “perspectives” of a system, whether from the lens of a particular application sector (e.g., aerospace, biomedicine), discipline (e.g., electrical, mechanical, thermal), subsystem, or system property (e.g., power, reliability, fault management).
- Formality: the model’s level of adherence to formalized standards for information expression. Models can express information about systems with varying degrees of precision. The most fundamental of models, which simply express a basic representation of a system in an unspecified format, do not convey information with precision and are considered informal. The most formal models comply with well-developed, pre-defined standards (formalisms) for content and organization, which collectively define ‘languages’ that enable consistent and precise interpretations of models.
- Abstraction: the degree to which a model suppresses or excludes out-of-scope, unimportant, or irrelevant details. Abstraction is a necessity with large and complex systems where it is impractical to replicate every aspect of a given system within a reasonable time and resource expenditure margin.
- Physical/conceptual: whether the model is concrete in nature (i.e., a physical model) or fully conceptual (i.e., an abstract model).
- Descriptive/analytical: whether a model details qualitative aspects of a system such as requirements, behaviors, or physical architecture (descriptive model), provides a representation of quantitative aspects of the system such as mass, reliability, power consumption via mathematical relationships (analytical model), or both (hybrid model).
- Fidelity: the degree to which a model comprehensively captures details about a system’s characteristics, ranging from models which only capture general information about a system to those which seek to faithfully capture as much detail about the system as possible.
- Completeness: the extent to which a model captures

all relevant domain- information within its scope and at its intended level of detail.
- Integration: the extent to which a model interacts and interfaces with other relevant models describing the system of interest or other related/interacting entities.
- Quality: the degree to which the model (not the system it represents) meets the needs of the individuals performing systems engineering activities. A high-quality model should be readily usable, have minimal ambiguity, and provide accurate, relevant information needed to support tasks associated with the design, development, operation, and/or maintenance of a system.

Of these properties, formalization and abstraction are generally the most frequently discussed in relation to MBSE (Vogelsang et al. 2017) as they have the greatest impact on whether a model can be effectively used as part of an MBSE workflow.

Criteria for Effective MBSE Models

While a successful MBSE workflow can involve the use of several different interconnected or standalone models of various scopes and types based on user needs, the main system model in an MBSE projects generally should have the following characteristics:

1. A scope which matches the scope of the project (i.e., it should encompass the entire system of interest);
2. Representative of a holistic perspective from all relevant domains.
3. Strict compliance with a previously established standardized modeling language, whether that be an existing language such as SysML® or a custom formalism.
4. Fully abstracted, to only include relevant information appropriate for the system of interest and its desired use-case(s).
5. Conceptual in nature, to permit the capture of intangible information (e.g., system requirements)
6. Containing a description of the system functional and structural architecture at minimum and supplemented by integrated analytical/quantitative property descriptions as needed.
7. Demonstrating sufficient fidelity to capture relevant

system elements and behavior.
8. Fully complete given its scope.
9. Integrated with any necessary auxiliary models.
10. Sufficiently high-quality as to meet the needs of those designing, developing, or otherwise working on the system.

In terms of content, effective system models are expected to capture key system information regarding requirements, system functionality/behavior, structure/form, properties, and interconnections between system components.

Modeling Languages

Modeling languages are specifications which provide standardized guidelines and structures for expressing system information. These languages, which provide both the structures or 'syntax' in which the information can be expressed, as well as the 'semantics' that govern the way in which the information should be interpreted, can be selected based on user preferences and needs. Different languages utilize different formats to express information (e.g., visual or textual means), as well as different paradigms (e.g., object-oriented, functional, etc.) in order to group information. Visual languages are generally preferred for modeling due to being readily readable, and object-oriented modeling languages are frequently used in systems engineering contexts since they readily lend themselves to systems which can be decomposed, or otherwise thought of, in terms of objects.

SysML®, an extension of Unified Modeling Language [UML] for systems engineering, is a one of the more frequently used modeling languages for MBSE. It is an graphical language that utilizes diagrams and tables in order to express system information, and provides a standard set of nine diagram types which can be used to organize and express system information (Friedenthal, Moore, and Steiner 2014). The collective diagrams (each of which can be considered a model in its own right), when interconnected, provide a means of representing system structure, behavior, and requirements in abstracted form. A number of other options have been proposed as architecture description languages [ADLs] for specifically modeling system architectures. ISO/IEC/IEEE 42010 (Systems and software engineering - Architecture description) specifies minimum requirements for a language to qualify as an ADL (ISO

2011).

MBSE users have the option of using SysML®, a similar graphical-language option like UML, a domain or framework specific language, or potentially developing a custom formalism for their team or organization (Bonnet et al. 2016). It is possible to formalize textual documents to create models, though doing so requires the establishment of a domain dictionary in order to remove the ambiguity inherent in diction choice, as well as the use of rigid grammatical structures which may limit readability.

Regardless of what modeling language is used for an MBSE project, it is important that the language be inherently scalable, standardized, readable, reusable, and abstractable to enable the development of effective MBSE models.

Architecture Frameworks

A second layer of structure that exists overtop a modeling language is an architecture framework. Architecture frameworks are used to organize the information expressed via modeling language. Whereas a modeling language provides the structure needed to express multiple 'views' (diagrams) of system elements and their interactions, architecture frameworks enable the user to group those views based on the elements they represent, and organize them in a way that allows traceability, eases navigation through the model, and aids in the identification of missing information (e.g., an omitted element). Architecture frameworks are a specific type of pattern that frequently get defined and standardized for MBSE models. There are also organization- and domain-specific design patterns that can be employed in MBSE models to meet stakeholder needs in more specific model use-cases.

Architecture frameworks and model design patterns play an important role in enabling the re-use of MBSE models (Wu et al. 2019), as certain architectural design patterns may be frequently used across multiple projects even when the specifications of the individual components differ (e.g. building a house with the same structure but different décor). By organizing a system model in a sufficiently abstracted manner, it may be possible to identify the points of difference between an old project and a new one and make the appropriate changes to element properties in the model without having to redo the entire model development process.

Process Frameworks

The MBSE model development workflow can be streamlined using pre-defined process frameworks, which provide tailorable guidelines and patterns for integrating MBSE into the generic systems engineering process. While process frameworks are typically defined on an organizational level, they generally all exhibit some form of configuration management process, access guidelines, practices for updating the model, and means of integrating the MBSE model into all or nearly all systems engineering lifecycle activities. The benefits of MBSE usage are limited when the system model falls out of date or otherwise becomes inaccurate, so regular model updates are a minimum requirement for MBSE process frameworks.

For smaller projects, the MBSE process framework may be as simple as utilizing the version control features that come included as part of many collaborative modeling software platforms and integrating model usage and periodic updates as checkpoints in the systems engineering process. More complex projects can formalize MBSE process frameworks in a manner that can be verified against configuration management and systems engineering management plans (Fisher et al. 2014).

Benefits of MBSE

The MBSE workflow and the creation of a centralized system model emphasizes a holistic, standards-based approach to systems engineering (Madni and Sievers 2018). Since the creation of a system model requires reconciliation of information from multiple domains and subsystems, inconsistencies and defects are readily identifiable during the modeling process (Carroll and Malins 2016) and can be addressed or eliminated earlier on in the system lifecycle process than would otherwise be done in a document-based workflow. Similarly, the centralization and standardization of information ensures a reduction in miscommunications and other development risks since all project team members are using the same source of information for reference. Format standardization also makes it easier to search for and extract information, compared to a document-based workflow where information is stored across multiple documents in different formats.

More broadly, MBSE provides a better means of managing complexity than document-based using

formalized structures and abstraction. Cross-referencing within MBSE models makes it possible to begin design verification, requirements validation, and systems assurance earlier on in the system lifecycle, and to continue assessing system design quality throughout a project at minimum cost. Furthermore, models can be reused and adapted for similar systems, which enables accelerated system development with minimal risk.

Digital Transformation

While DBSE has traditionally been the paradigm of preference for artifact generation and for supporting systems engineering efforts in the pre-digital age, digital transformation of the generic systems engineering workflow in recent years has catalyzed the widespread adoption of MBSE and broader model-based [MBx] approaches. Digital environments and software tools have made it easier and faster to generate, maintain, and use system models, especially in a collaborative setting (Ma et al. 2022). If implemented appropriately, digital MBSE models can be used to programmatically identify inconsistencies, enable interactive simulations of system behavior, simultaneously propagate changes across an entire project (rather than updating artifacts one-by-one), automatically generate document-based artifacts, and more. The advent of new software for supporting and automating systems engineering tasks has opened additional avenues for expanding the capabilities of system models, and for increasing the efficiency with which systems engineering tasks can be performed.

Digital Twins

When MBSE models of physical systems are built with sufficient completeness and fidelity, it is possible for them to function as ‘digital twins’ of the systems they represent. Digital twins provide a means of accurately representing a system’s form and function throughout the system’s lifecycle, all within a digital environment. Creating such digital twins provides number of advantages, including allowing individuals to perform testing, analysis, and optimization of systems in a virtual environment at no risk to the actual system of interest and often at a greatly reduced cost/burden (Schluse, Atorf, and Rossmann 2017). Digital twins also make it possible to represent the behavior of systems under conditions which would be impractical or impossible to induce under experimental conditions, thereby making it possible to obtain information not obtainable via study of

the original physical system.

MBSE versus DBSE

Although MBSE and document-based approaches are usually presented as alternatives to each other, it is possible to use MBSE and document-based in conjunction with each other on the same project. In work environments where document-based is the norm, stakeholders may expect or require the submission of textual document artifacts, or there may be issues with a lack of familiarity with any modeling languages (Kim, Wagner, and Jimenez 2019); in such instances, it may be necessary to utilize a hybrid approach where documents are generated from the design model as static representations of the system for project milestones.

References

Works Cited

Bonnet, Stéphane, Jean-Luc Voirin, Daniel Exertier, and Véronique Normand. 2016. “Not (Strictly) Relying on SysML for MBSE: Language, Tooling and Development Perspectives: The Arcadia/Capella Rationale.” In 2016 Annual IEEE Systems Conference (SysCon), 1–6. https://doi.org/10.1109/SYSCON.2016.7490559.

Carroll, Edward Ralph, and Robert Joseph Malins. 2016. “Systematic Literature Review: How Is Model-Based Systems Engineering Justified?.” SAND2016-2607, 1561164. https://doi.org/10.2172/1561164.

Fisher, Amit, Mike Nolan, Sanford Friedenthal, Michael Loeffler, Mark Sampson, Manas Bajaj, Lonnie VanZandt, Krista Hovey, John Palmer, and Laura Hart. 2014. “3.1.1 Model Lifecycle Management for MBSE.” INCOSE International Symposium 24 (1): 207–29. https://doi.org/10.1002/j.2334-5837.2014.tb03145.x.

Friedenthal, Sanford, Alan Moore, and Rick Steiner. 2014. A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann. ISO. 2011. “ISO/IEC/IEEE 42010.” Geneva, Switzerland: International Organization for Standardization (ISO). https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/05/50508.html.

Kim, So Young, David Wagner, and Alejandro Jimenez. 2019. “Challenges in Applying Model-Based Systems

Engineering: Human-Centered Design Perspective," September. https://trs.jpl.nasa.gov/handle/2014/51368.

Ma, Junda, Guoxin Wang, Jinzhi Lu, Hans Vangheluwe, Dimitris Kiritsis, and Yan Yan. 2022. "Systematic Literature Review of MBSE Tool-Chains." Applied Sciences 12 (7): 3431. https://doi.org/10.3390/app12073431.

Madni, Azad M., and Michael Sievers. 2018. "Model-Based Systems Engineering: Motivation, Current Status, and Research Opportunities." Systems Engineering 21 (3): 172–90. https://doi.org/10.1002/sys.21438.

Schluse, Michael, Linus Atorf, and Juergen Rossmann. 2017. "Experimentable Digital Twins for Model-Based Systems Engineering and Simulation-Based Development." In 2017 Annual IEEE International Systems Conference (SysCon), 1–8. https://doi.org/10.1109/SYSCON.2017.7934796.

Vogelsang, Andreas, Tiago Amorim, Florian Pudlitz, Peter Gersing, and Jan Philipps. 2017. "Should I Stay or Should I Go? On Forces That Drive and Prevent MBSE Adoption in the Embedded Systems Industry." In Product-Focused Software Process Improvement, edited by Michael Felderer, Daniel Méndez Fernández, Burak Turhan, Marcos Kalinowski, Federica Sarro, and Dietmar Winkler, 182–98. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-69926-4_14.

Wu, Quentin, David Gouyon, Sophie Boudau, and Éric Levrat. 2019. "Capitalization and Reuse with Patterns in a Model-Based Systems Engineering (MBSE) Framework." In 2019 International Symposium on Systems Engineering (ISSE), 1–8. https://doi.org/10.1109/ISSE46696.2019.8984571.

Wymore, A. Wayne. 1993. Model-Based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design. Boca Raton: CRC Press.

Primary References

Estefan, J. 2008. A Survey of Model-Based Systems Engineering (MBSE) Methodologies, Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Available at: http://www.incose.org/ProductsPubs/pdf/techdata/MTTC/MBSE_Methodology_Survey_2008-0610_RevB-JAE2.pdf.

INCOSE. 2021. Systems Engineering Vision 2035. Torrance, CA, USA: International Council on Systems Engineering.

OMG. "MBSE Wiki." Object Management Group (OMG). Available at: http://www.omgwiki.org/MBSE/doku.php. Accessed 05 April 2022.

Additional References

Downs, E., P. Clare, and I. Coe. 1992. Structured Systems Analysis and Design Method: Application and Context. Hertfordshire, UK: Prentice-Hall International.

INCOSE. 2007. Systems Engineering Vision 2020. Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

Kossiakoff, A. and W. Sweet. 2003. "Chapter 14," in Systems Engineering Principles and Practice. New York, NY, USA: Wiley and Sons.

NDIA. 2011. Final Report of the Model Based Engineering (MBE) Subcommittee. Arlington, VA, USA: National Defense Industrial Association. Available at: http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/M_S%20Committee/Reports/MBE_Final_Report_Document_(2011-04-22)_Marked_Final_Draft.pdf

Oliver, D., T. Kelliber, and J. Keegan. 1997. Engineering Complex Systems with Models and Objects. New York, NY, USA: McGraw-Hill.

< Previous Article | Parent Article | Next Article >
SEBoK v. 2.9, released 20 November 2023
