MBSE Modeling Manifesto

VERSION 1.1

DR. BRUCE POWEL DOUGLASS
SENIOR PRINCIPAL AGILE SYSTEMS ENGINEER
MITRE CORPORATION

June 3, 2025

MBSE Modeling Manifesto

We are uncovering better ways of building and using models for the engineering organization by doing it and helping others do it. Through this work, we have come to value:

- **Well-formed models** over diagrams that only resemble the desired structures
- **Verifiable models** over ambiguous or imprecise representations.
- **Incremental development and verification** over big-bang modeling efforts.
- **Simplicity in organization** over unnecessary complexity.
- **Accuracy in representation** over superficial abstraction.
- **Singular representation of elements** over redundant representations of the same element

That is, while there may be value in the items on the right, we value the items on the left more.

Principles Behind the MBSE Modeling Manifesto

1. **Model Well-Formedness**
   Models must adhere to proper SysML standards and comply with the MBSE Modeling Guidelines. Well-formed models are the foundation for clear communication, consistency, and usability across the organization. This includes naming conventions, model organization, and precise simultable model structures.
2. **Model Verifiability**
   Models should be constructed in a way that enables automated and manual verification of their correctness, completeness, and alignment with requirements. A model that cannot be verified is a model that cannot be trusted.
3. **Incremental Model Development and Verification**
   Models should be developed iteratively, with frequent verification and validation steps to ensure quality and alignment with evolving system requirements. Incremental development allows for adaptability and reduces the risk of large-scale rework.

4. **Simplicity of Model Organization**
   Models should be organized in a way that is easy to navigate, understand, and maintain. Avoid unnecessary complexity and focus on clarity and usability for all stakeholders. Group things in common packages where it makes sense.

5. **Model Accuracy**
   Models must accurately reflect the actual systems and software architecture they represent. A model that diverges from reality undermines its purpose and value as a decision-making tool. A model that doesn't represent what the actual system is or how it functions doesn't help.

6. **Collaboration Through Models**
   Models should serve as a shared language for collaboration among engineers, architects, and stakeholders. They should facilitate communication and foster alignment across teams.

7. **Continuous Improvement**
   Modeling practices should evolve based on lessons learned, feedback, and advancements in tools and methodologies. Strive for excellence in modeling by embracing change and innovation.

8. **Focus on Value**
   Models should be created with a clear purpose and deliver tangible value to the organization and/or project. Avoid modeling for the sake of modeling; focus on solving real problems and enabling informed decision-making. Models should start with a clear Purpose, Scope, Required Precision, and Means for Verification and all subsequent work should support this statement.

9. **Traceability and Transparency**
   Models should maintain traceability to requirements, design decisions, and system artifacts. Transparency in modeling ensures accountability and facilitates understanding across the organization.

10. **Respect for Stakeholder Needs**
    Models should be tailored to meet the needs of their intended audience, whether technical or non-technical. Ensure that models provide the right level of detail and abstraction for effective communication.

Application of the Manifesto and Principles

The **MBSE Modeling Manifesto** is not just a set of ideals—it is a practical framework for guiding the creation, development, and maintenance of models within the engineering organization. Here's how these principles can be applied in day-to-day modeling activities:

1. Model Well-Formedness

- Ensure that all models comply with SysML syntax and semantics, as well as the MBSE Modeling Guidelines. Use modeling tool features that enforce proper SysML constructs and provide validation features.
- Regularly review models for adherence to guidelines and standards, leveraging peer reviews and automated checks.
- Train team members on SysML best practices and the MBSE Modeling Guidelines to ensure consistency across projects.
- Be clear that diagrams only visualize aspects of the model for a specific purpose and are not the model per se. There is an important distinction between a model element and the symbol that represents it on a diagram.

2. Model Verifiability

- Build models with verification in mind by using clear relationships, constraints, and traceability links. Ensure that every element in the model can be checked against requirements or design criteria.
- Integrate automated verification tools to validate model correctness, completeness, and compliance with organizational standards.
- Conduct regular verification sessions to identify gaps, inconsistencies, or errors in the model.
- Verification approach should emphasize goal-directed simulation of model elements.
- All behaviors, including interactions, activities and state machines, should simulate properly.

3. Incremental Model Development and Verification

- Break down modeling efforts into manageable increments, focusing on small, achievable goals rather than attempting to model the entire system at once.
- Develop models iteratively, starting with specific modeling objectives and progressively refining them to add detail, accuracy and additional scope.
- Verify each increment as it is developed, ensuring that errors are caught early and do not propagate through the model.

4. Simplicity of Model Organization

- Organize models using a clear and logical structure, such as modular decomposition or hierarchical organization. Use packages, views, and diagrams effectively to maintain clarity.
- Avoid overloading models with unnecessary detail or complexity. Focus on what is essential to meet model objectives and communicate effectively.
- Regularly refactor models to improve their organization and remove redundant or outdated elements.

5. Model Accuracy

- Collaborate closely with system architects, engineers, and domain experts to ensure that models accurately reflect the actual systems and/or software architecture.
- Use real-world data, measurements, and validated assumptions to inform modeling decisions.
- Continuously update models to reflect changes in the system or architecture, ensuring they remain relevant and accurate.
- Interfaces should reflect the actual interfaces, although they might be modeled at a different level of abstraction. That means, especially for software models, interfaces should emphasize services that may carry or return data and not data flows.

6. Collaboration Through Models

- Use models as a shared communication tool across teams and stakeholders. Encourage collaborative modeling sessions to align on requirements, designs, and decisions.
- Ensure that models are accessible to all relevant stakeholders, providing appropriate levels of abstraction for different audiences.
- Foster a culture of openness and feedback, where stakeholders can contribute to and refine models collaboratively.
- Design models should drive implementation in a clear and obvious way. There should be traceability to requirements, implementation, and test structures.

7. Continuous Improvement

- Regularly review modeling practices and tools to identify areas for improvement. Incorporate lessons learned from past projects into future modeling efforts. Use the review approach from the MBSE Modeling Guidelines.

- Stay informed about advancements in modeling methodologies, tools, and standards, and adopt them where appropriate.
- Encourage team members to share best practices and innovations in modeling.

8. Focus on Value

- Clearly define the purpose and objectives of each model before beginning development. Ensure that models are solving real problems and providing actionable insights.
- Avoid modeling unnecessary aspects of the system. Focus on areas that deliver the most value to stakeholders and decision-makers.
- Periodically assess the value of existing model elements and diagrams and remove those that are no longer useful.

9. Traceability and Transparency

- Maintain traceability between model elements and system requirements, design decisions, and implementation artifacts. Use tools that support traceability features.
- Ensure that models are transparent and understandable, providing clear documentation and annotations where needed.
- Use traceability to support impact analysis, helping stakeholders understand the consequences of changes to requirements or designs.
- Always naming model elements with meaningful names from the domain of discourse
- Add in overview diagrams with links to relevant views, packages, or model elements to aid in model navigation.

10. Respect for Stakeholder Needs

- Always follow the model directives of stated Purpose, Scope, Precision, and Means for Verification to direct model development.
- Tailor models to the needs of their intended audience. For technical stakeholders, provide detailed and precise representations; for non-technical stakeholders, use simplified and high-level views.
- Use visualization techniques, such as diagrams, tables, and matrices, to make models more accessible and understandable.

- Continuously engage stakeholders to ensure that models are meeting their needs and expectations.

Modeling To-Dos and To-Don'ts

To ensure adherence to the **MBSE Modeling Manifesto** and its principles, it is important to recognize common pitfalls and best practices. Below are actionable "to-dos" and "to-don'ts" for each principle, with examples to guide modelers.

---

1. Model Well-Formedness

To-Do:

- Use SysML-compliant constructs such as blocks, activities, and state machines appropriately.
- Models should, in general, always support simulation.
- Follow the MBSE Modeling Guidelines for naming conventions, diagram organization, and element relationships.
- Regularly validate models using automated tools to check for syntax errors and inconsistencies.

To-Don't:

- Don’t create diagrams with disconnected elements or ambiguous relationships.
  Example: Avoid floating blocks in a Block Definition Diagram (BDD) that are not connected to other elements.
- Don’t use informal or inconsistent naming conventions.
  Example: Avoid naming blocks "Thing1" or "Miscellaneous" instead of descriptive names like "PowerSubsystem" or "CommunicationModule."

---

2. Model Verifiability

To-Do:

- Include constraints, parametric relationships, and traceability links to ensure the model can be verified against requirements.
- Use verification tools to check model completeness and correctness.

To-Don't:

- Don’t leave model elements unconnected to requirements or other design artifacts.
  Example: Avoid creating a Use Case Diagram without linking use cases to relevant requirements.
- Don’t use vague or non-verifiable constraints.
  Example: Avoid constraints like "System must be fast" without specifying measurable criteria such as "System response time < 2 seconds." This is especially important with specifying requirements as well as constraints.

---

3. Incremental Model Development and Verification

To-Do:

- Start with high-level abstractions and refine the model incrementally, verifying each step.
- Prioritize modeling the most critical system components first.

To-Don't:

- Don’t attempt to model the entire system in one pass.
  Example: Avoid trying to build a complete Internal Block Diagram (IBD) for all subsystems before verifying individual blocks.
- Don’t skip verification steps during incremental development.
  Example: Avoid moving to detailed design without validating high-level system architecture.

---

4. Simplicity of Model Organization

To-Do:

- Use packages to organize model elements logically (e.g., grouping by subsystem or functionality).
- Create diagrams that are easy to read and focus on essential information. Give each diagram a mission statement and only expose model elements and features that support that mission.
- Create separate diagrams to expose separate concepts or concerns. If you have 5 questions to answer, that means you will generally have 5 different diagrams.
- Always have a singular definition as the authoritative source of truth for any given model element.

To-Don't:

- Don’t overload diagrams with excessive detail or unrelated elements.
  Example: Avoid creating a BDD with hundreds of blocks in a single diagram—break it into smaller, focused diagrams adding diagram hyperlinks when useful.
- Don’t have multiple definitions or blocks representing the same model element spread throughout the model.
- Don’t use overly complex naming or nesting structures.
  Example: Avoid naming packages "Subsystems_Version1_Iteration2_FinalDraft" instead of "Subsystems."

---

5. Model Accuracy

To-Do:

- Collaborate with domain experts to ensure the model reflects the actual system capabilities, structure and behavior.
- Continuously update the model to reflect changes in the system design, simulating and verify the model as you do so.
- Use appropriate model relations, such as specialization, composition, and association correctly when defining the ways in which model elements relate

To-Don't:

- Don’t use placeholder elements or assumptions without validating them.
  Example: Avoid creating a block called "UnknownComponent" without specifying its role or function.
- Don’t let the model diverge from the real system architecture.
  Example: Avoid modeling a system with a centralized architecture when the actual system is distributed.

---

6. Collaboration Through Models

To-Do:

- Share models with stakeholders regularly and solicit feedback.
- Use views and diagrams tailored to different audiences (e.g., technical vs. non-technical stakeholders).

- Use good configuration management disciple to ensure model integrity over time.
- Have a singular “model owner” responsible for the integrity of the model. They can reject ill-formed or inappropriate model updates.

To-Don't:

- Don’t keep models siloed within a single team or individual.
  Example: Avoid creating models that only one person understands or has access to.
- Don’t use overly technical diagrams for non-technical stakeholders.
  Example: Avoid showing a complex Parametric Diagram to a business manager—use a simplified Use Case Diagram instead.
- Don’t pollute the model with “sand box” packages. If you need to do an experiment, create a separate model that you own to do so, importing the actual model to access model structures if necessary.

---

7. Continuous Improvement

To-Do:

- Regularly review and refine models based on lessons learned and stakeholder feedback.
- Stay updated on new project needs.
- Incorporate approved model changes coming from model reviews, inspections, and simulations.
- Refactor the model when necessary to maintain model simplicity and clarity.
- Emphasize “correct even if incomplete” over “complete but incorrect”

To-Don't:

- Don’t retain identified model semantic or syntactic errors.
  Example: When a problem is discovered in a model inspection and a change is agreed upon, make and verify the changes to the model then close the loop to the requestor of the change.
- Don’t ignore feedback from stakeholders or lessons learned.
  Example: Avoid repeating the same mistakes in model organization or structure across projects.

---

8. Focus on Value

To-Do:

- Clearly define the purpose of each model and ensure it addresses stakeholder needs.
- Focus on modeling areas that provide actionable insights or solve real problems.

To-Don't:

- Don’t model for the sake of modeling without a clear purpose.
  Example: Avoid creating a State Machine Diagram for a subsystem that doesn’t have state-dependent behavior.
- Don’t spend excessive time modeling trivial aspects of the system.
  Example: Avoid modeling every single wire in a system when a high-level connection diagram suffice for the appropriate level of abstraction.

---

9. Traceability and Transparency

To-Do:

- Maintain traceability links between requirements, design elements, test structures, and implementation artifacts.
- Provide clear documentation and annotations for model elements. *Every important model element should have a clear and meaningful description*.
- Use documentation/annotation template forms from the MBSE Modeling Guidelines, where provided.

To-Don't:

- Don’t leave traceability gaps between requirements and design.
  Example: Avoid creating a model where system requirements are not linked to corresponding blocks or activities.
- Don’t create opaque models without explanations.
  Example: Avoid using cryptic element names or relationships without providing context in documentation.

---

10. Respect for Stakeholder Needs

To-Do:

- Tailor model views (diagrams, tables and matrices) to the needs of their audience, providing appropriate levels of detail and abstraction.
- Use diagrams and visualizations that are fit for purpose, for example, easy to understand for non-technical stakeholders when they are they primary target audience for a diagram.

To-Don't:

- Don’t overwhelm stakeholders with unnecessary technical details.
  Example: Avoid showing a Parametric Diagram with complex equations to a stakeholder interested in high-level system functionality.
- Don’t ignore stakeholder feedback or fail to address their concerns.
  Example: Avoid creating models that focus solely on technical aspects while neglecting business or operational needs.

References

[1] **Agile Manifesto** https://agilemanifesto.org/
[2] **MBSE Modeling Guidelines V2** By Dr. Bruce Powel Douglass
[3] **Agile Systems Engineering** by Dr. Bruce Powel Douglass, Morgan Kaufmann Publishers, 2015
[4] **Agile Model-Based Systems Engineering Cookbook, 2nd Edition** by Dr. Bruce Powel Douglass, Packt Publishing, 2022