33rd Annual INCOSE international symposium

hybrid event

Honolulu, HI, USA
July 15 - 20, 2023

MBSE Model Management Pain Points - Wait, this looks familiar!

Barry Papke (CATIA Magic/Dassault Systemes)
Matthew Hause (System Strategy Inc.)
David Hetherington (System Strategy Inc.)
Sean McGervey (System Strategy Inc.)
Sami Rodriguez (Strategic Technology, Inc.)

MBSE Adoption – Where are we today?

- It has been almost 20 years since INCOSE and the OMG kicked off an effort to create a standardized model-based systems engineering modeling language. That effort culminated in 2006 with the creation of the Systems Modeling language (SysML).
- Over the past several years, numerous industries have increased their adoption of SysML and MBSE as a core practice within their engineering lifecycles. However, they have not achieved many of the originally envisioned benefits.
- On closer inspection, many of the challenges and barriers with which MBSE practitioners are currently struggling are remarkably similar to the problems seen in large DoD software programs from the 1970s and 80s and were resolved decades ago by the software development community regarding scale and complexity.

MBSE Adoption – Where are we today?

- The similarity between MBSE's model management pain points and those experienced years ago by the software industry is not unexpected.
  - Adoption of MBSE requires the replacement of the largely manual, document-based engineering processes with a complex engineering information processing system.
  - Unlike documents, the models produced by that system are living artifacts that require management over the project lifecycle and which have all the characteristics and complexity of software.
- This session presents a framework for MBSE planning and model lifecycle management based on the key practices from Systems Engineering and Software Engineering to provide an actionable set of best practices that can be applied today to address current MBSE lifecycle management issues.

The Blind Men and the Elephant

- The first blind man reached out and touched the side of the huge animal. "An elephant is smooth and solid like a wall!" he declared. "It must be very powerful."
- The second blind man put his hand on the elephant's limber trunk. "An elephant is like a giant snake," he announced.
- The third blind man felt the elephant's pointed tusk. "I was right," he decided. "This creature is as sharp and deadly as a spear."
- The fourth blind man touched one of the elephant's four legs. "What we have here," he said, "is an extremely large cow."
- The fifth blind man felt the elephant's giant ear. "I believe an elephant is like a huge fan or maybe a magic carpet that can fly over mountains and treetops," he said.
- The sixth blind man gave a tug on the elephant's coarse tail. "Why, this is nothing more than a piece of old rope. Dangerous, indeed," he scoffed.

It's a fan!
It's a wall!
It's a rope!
It's a spear!
It's a tree!
It's a snake!

The Near-Sighted Systems Engineers and the Model

- The model needs to provide the following data to other engineering disciplines. (Requirements)
- The model generate outputs from queries, validation rules and simulations. (Functions)
- The model needs to load, commit changes and support a specified number of concurrent users. (Performance)
- The model needs to exchange data with other models and tools. (Interfaces)
- The model must comply with modeling standards. (Quality Assurance)
- Changes to the model must be managed. (Configuration Management)
- The model needs to be useful over the full lifecycle of the system. (Lifecycle Management)

Knowing how it could change the lives of canines everywhere, the dog scientists struggled diligently to understand the Doorknob Principle.

What do we call something that has allocated functional and performance requirements, performs specific functions, interfaces with other systems, executes on processing hardware, must meet quality standards, must be under configuration management and is a living artifact throughout the system lifecycle?

A SOFTWARE INTENSIVE SYSTEM- The Elephant in the Room

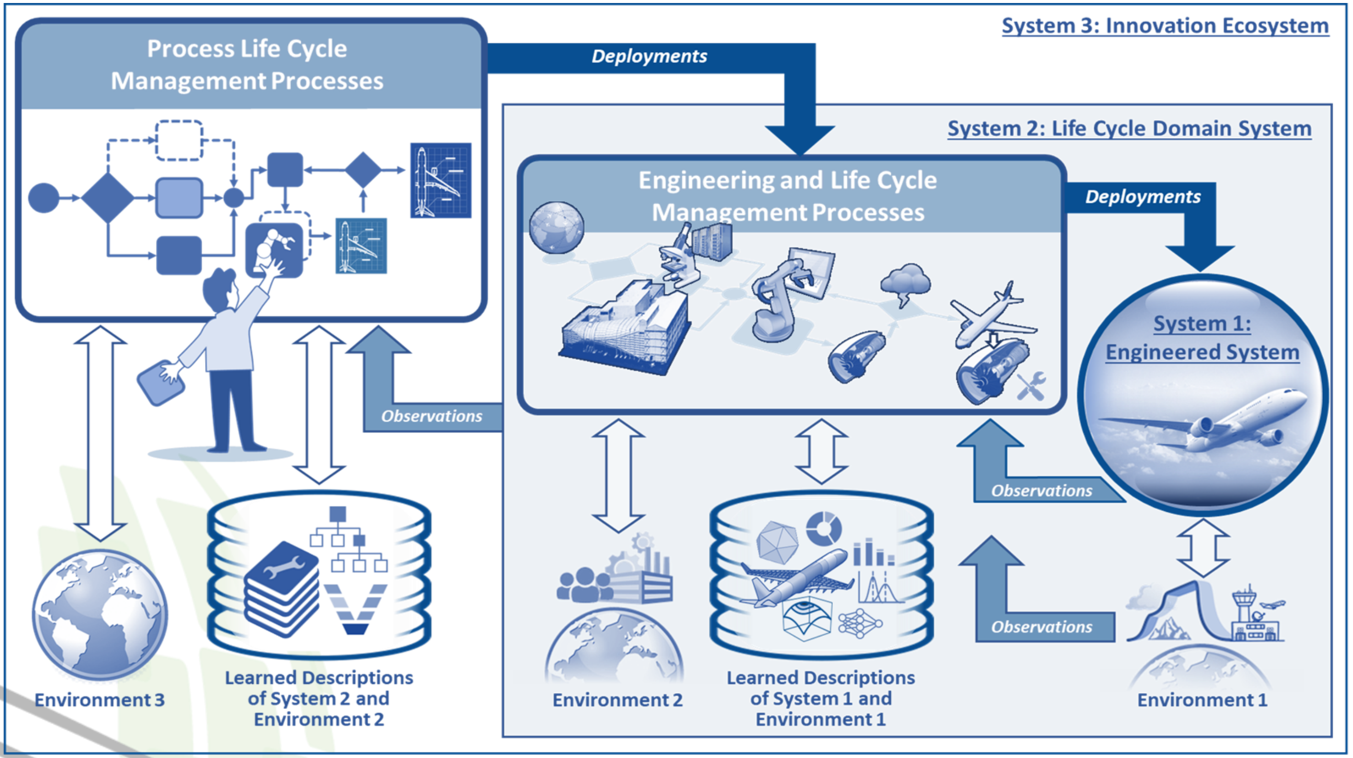
Taking a Systems Engineering Perspective

INCOSE Agile Systems Engineering Life Cycle Management (ASELCM) Reference Model

- **System 1**– The Engineered System of Interest: at all times in its life cycle.

- **System 2** – The Life Cycle Management Domain: The environment with which the Engineered System interacts, across its life cycle.

- **System 3** – The Overall System of Innovation: The system responsible to plan, deploy, and evolve System 2.


System 3: Innovation Ecosystem
Process Life Cycle Management Processes
Deployments
System 2: Life Cycle Domain System
Engineering and Life Cycle Management Processes
Deployments
System 1: Engineered System
Observations
Observations
Observations
Environment 3
Learned Descriptions of System 2 and Environment 2
Environment 2
Learned Descriptions of System 1 and Environment 1
Environment 1

INCOSE ASELCM Level 0 Reference Model—Systems 1, 2, and 3

System 3: Process definition, advancement | System 2: Engineering, production, support, science | System 1: Products

Copyright 2022 William D. Schindel, Realizing the Value Promise of Digital Engineering: Planning, Implementing, and Evolving the Ecosystem, Published in INCOSE Insight

In the context of the MBSE Digital Environment

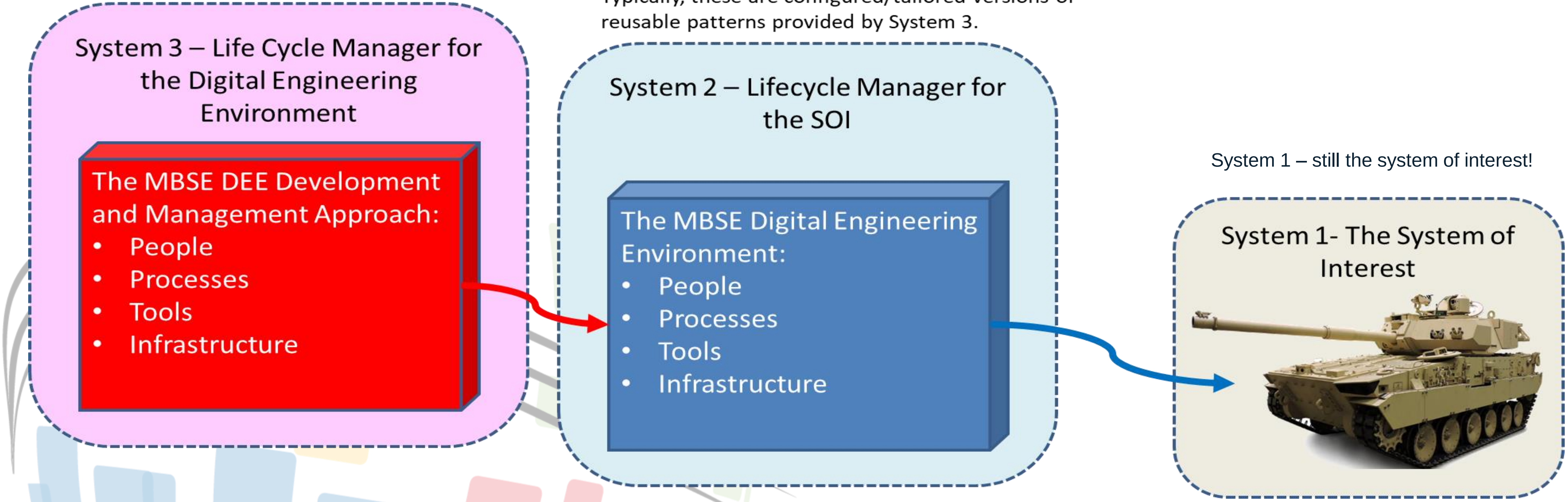
System 3 produces the configurable patterns that are deployed each time a new SOI project is initiated. Typically these patterns include templates for:

- Requirements/Architecture
- Implementation
- Project Plan

System 2 develops and manages the Digital Artifacts for the SOI:

- Requirements/Architecture
- Implementation
- Project Plan

Typically, these are configured/tailored versions of reusable patterns provided by System 3.



System 3 – Life Cycle Manager for the Digital Engineering Environment

The MBSE DEE Development and Management Approach:

- People
- Processes
- Tools
- Infrastructure

System 2 – Lifecycle Manager for the SOI

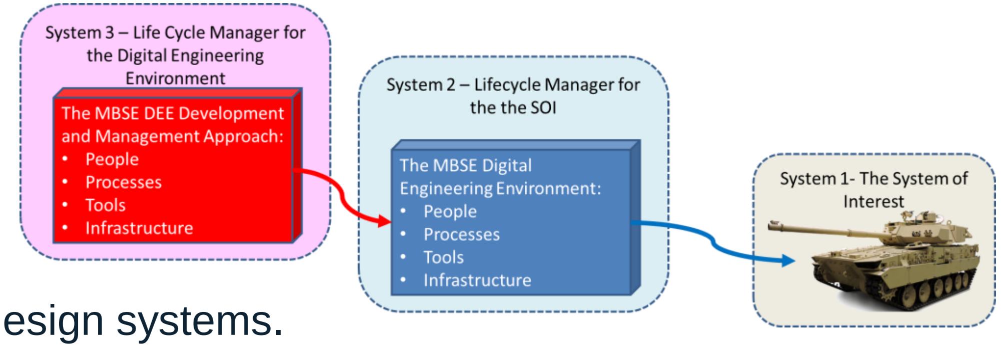
The MBSE Digital Engineering Environment:

- People
- Processes
- Tools
- Infrastructure

System 1 – still the system of interest!

System 1- The System of Interest

What are we doing wrong?

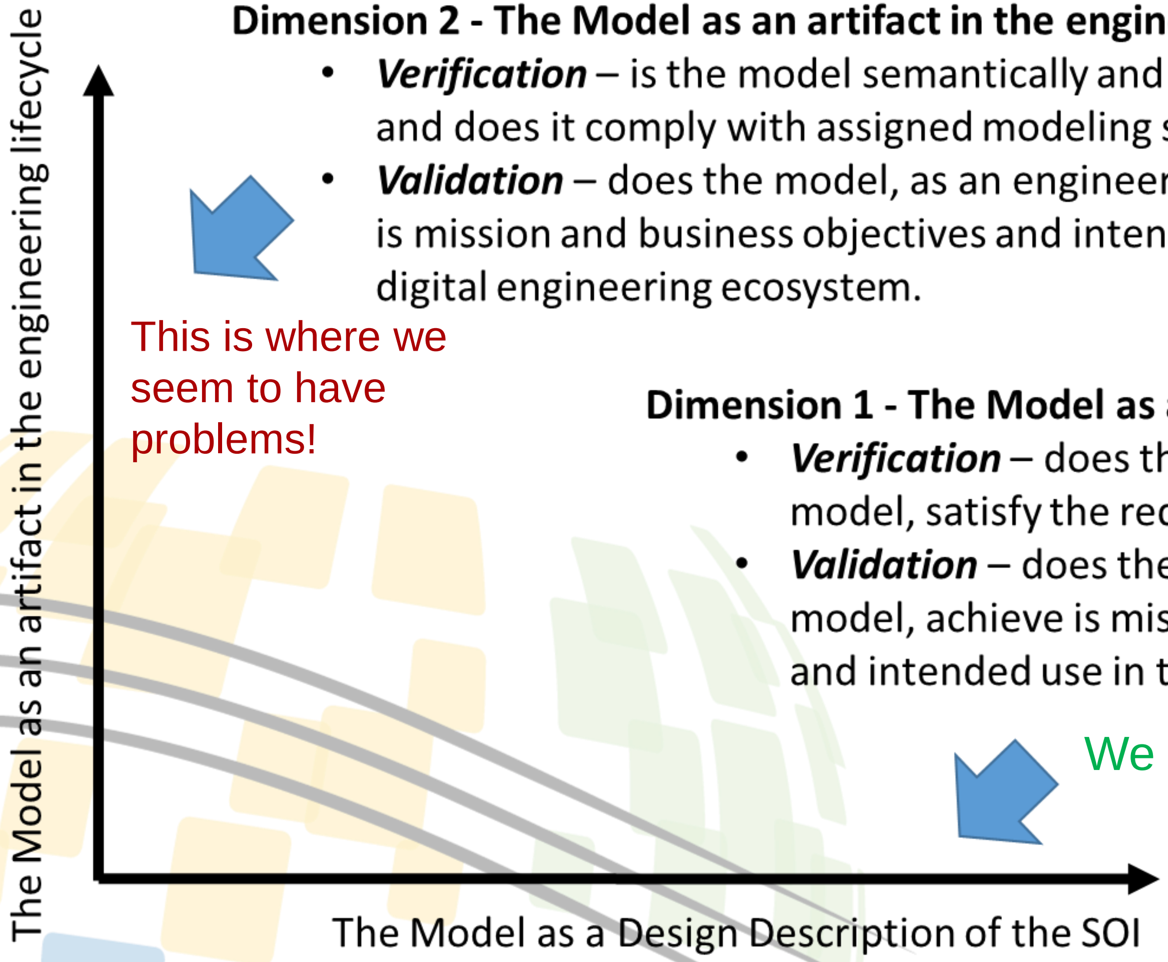

System 3 – Life Cycle Manager for the Digital Engineering Environment
The MBSE DEE Development and Management Approach:
• People
• Processes
• Tools
• Infrastructure
System 2 – Lifecycle Manager for the the SOI
The MBSE Digital Engineering Environment:
• People
• Processes
• Tools
• Infrastructure
System 1- The System of Interest

- **System 1 – The System of Interest:**
  - The SE process is sound and proven. Projects know how to design systems.

- **System 2 - The MBSE Digital Engineering Ecosystem:**
  - System-2 consists of the people, processes, tools and infrastructure responsible for the planning, deployment and evolution of System-1.
  - **<u>System 2 has not fully evolved from legacy document-based practices.</u>**

- **System 3 – The MBSE Digital Engineering Ecosystem Lifecycle Planners and Managers:**
  - System-3 consists of the people, processes, tools and infrastructure responsible for the planning, deployment and evolution of System-1.
  - System 3 has been dormant since CAD became a commodity!
  - **<u>System-3 has not been successful in the planning, deployment and evolution of System-2.</u>**

Two Dimensions of Model V&V


The Model as an artifact in the engineering lifecycle
The Model as a Design Description of the SOI

Dimension 2 - The Model as an artifact in the engineering lifecycle

- ***Verification*** – is the model semantically and syntactically correct and does it comply with assigned modeling standards.
- ***Validation*** – does the model, as an engineering artifact, achieve is mission and business objectives and intended use in the digital engineering ecosystem.

This is where we seem to have problems!

Dimension 1 - The Model as a Design Description of the SOI

- ***Verification*** – does the design, as represented by the model, satisfy the requirements specified for the SOI
- ***Validation*** – does the system, as represented by the model, achieve is mission and business objectives and intended use in the operational environment.

We seem to get this right most of the time!

Model Management Imperatives

These model management imperatives acknowledge the complexity of the MBSE digital environment, and the increased rigor required in its planning, development and lifecycle management:

Imperative #1 – Ensure the models are a trusted and useful engineering artifact that performs its function within the engineering lifecycle.

Imperative #2 – Ensure the IT infrastructure has the performance and capacity to meet the modeling environments network, processing performance and storage needs.

Imperative #3 – Apply appropriate model lifecycle management practices for a software intensive system-of-systems.

Imperative #1 – Ensure the model is a trusted and useful engineering artifact.

- The system of models for a large MBSE project requires the same systems engineering approach as any other complex system:
  - **Understand Model Stakeholders Needs**
    - Its more than the modelers!
  - **Define MBSE Model Requirements**
    - Model scope and content, load and commit time, etc.
    - Model purpose – descriptive vs analytical, etc.
  - **Define the MBSE Model Architecture**
    - Architecture of Models as well as internal model package structure
    - Model usages, libraries, contractor boundaries, etc.
    - Apply proper software architecture heuristics (loose coupling, proper cohesion, interface complexity, modularity, maintainability.)
    - Plan and manage model-to-model interfaces
  - **Plan for Model Integration**
    - Perform incremental deliveries that exercise not just the model but the integrated digital engineering environment, both horizontally and vertically

Imperative #1 – Ensure the model is a trusted and useful engineering artifact (continued).

- **Perform Verification Testing**
  - Perform testing to ensure that all data exchange interfaces perform as specified, and that all outputs and reports can be produced.
  - Verify the results of calculations, simulations and queries to ensure they produce the correct results. (more on this in Imperative #3)
- **Perform Quality Assurance**
  - Develop modeling standards for package content, required model element relationships within and between abstraction layers.
  - Conduct peer review and quality assurance reviews. (more on this in Imperative #3)
- **Perform Configuration Management**
  - Perform the fundamental processes for Software Configuration Management (Configuration Identification, Configuration Control, Configuration Status Accounting, Configuration Audits, Interface Control and Subcontractor Control.)

Imperative #2 – Ensure the IT infrastructure has the performance and capacity to meet the modeling environment's needs.

- As with any software intensive system, the processing hardware and the deployment of modeling tools to hardware has a direct impact on total system performance.
- While this seems to be an obvious activity, many projects fail this practice due to organizational and stakeholder misalignment.
- Often, the IT infrastructure is defined and deployed based on pre-planned IT budgets or in many cases, as an afterthought.

Long ago, software engineering learned that providing developers with the highest performance development hardware provided the largest increase in productivity at the lowest cost!

Imperative #2 – Ensure the IT infrastructure has the performance and capacity to meet the modeling environments needs.

- The IT infrastructure needs be planned in the same way as the processing recourses, data storage, network/interface performance and throughput for a software intensive system:
  - **Understand Stakeholders and Stakeholder Needs**
    - Identify all stakeholders including both model content creators and those responsible for “backend” model management such as tool installation, network security, model database management, managing backups and logs.
  - **Define IT Infrastructure Requirements**
    - Define the processing and network performance and storage capacity needs based on projected model size, number of users and the infrastructure requirements defined by the modeling tool documentation.
    - Identify security and network protection requirements.
    - Plan for growth in IT resource demand.
  - **Define the IT Infrastructure Architecture**
    - Define the architecture for the entire digital engineering ecosystem including customer, prime contractor and subcontractor environments and their access levels.
    - Identify all processing enclaves and their required levels of security and protection.
    - Evaluate tool deployment and IT technologies (such as virtual machines, server clustering, containerization, etc.) to optimize overall MBSE ecosystem performance.

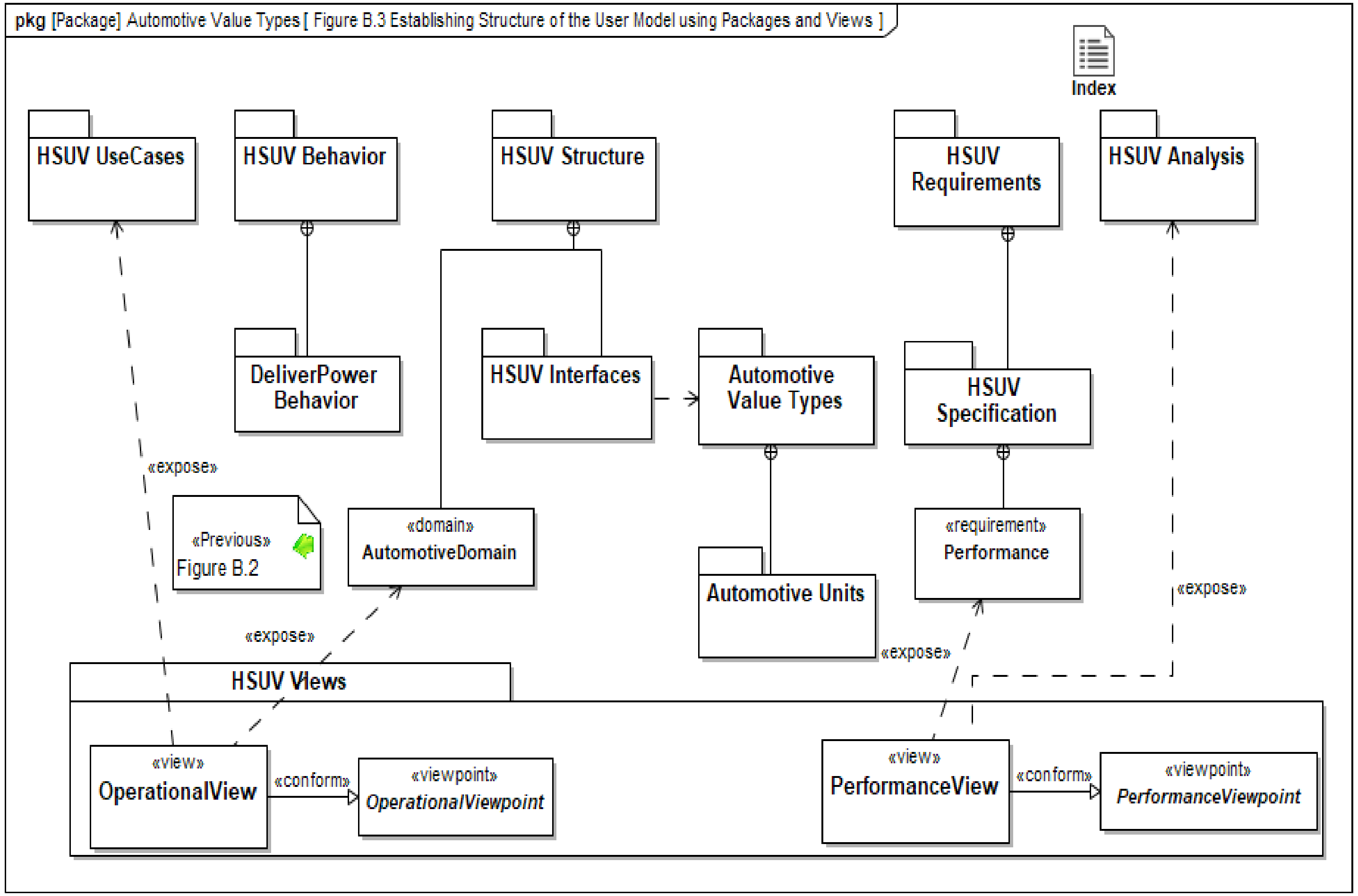
Imperative #2 – Ensure the IT infrastructure has the performance and capacity to meet the modeling environments needs (continued).

- **Plan for IT Network Integration**
  - Develop a network integration strategy and plan based on network security and performance requirements and other constraints.
- **Perform Periodic Verification and Performance Testing**
  - Perform testing to ensure that all IT network interfaces perform as specified, and that key tool functions such as downloads, change commits, simulations and queries execute with acceptable performance.
  - Monitor key performance parameters (such as open and commit times) as the number of models, the number of users and the size of models increase.

The IT infrastructure is not a separate system from the rest of the modeling ecosystem.

Imperative #3 - Apply appropriate model lifecycle management practices for a software intensive system-of-systems.

- Monolithic software is difficult to update, reuse and maintain
  - Monolithic models as well
- Model partitioning via packages enables modularity
- Model libraries ensure interface and type consistency and enable reuse.
- Enables Modular Open System Approach (MOSA)



What About Model Reuse?

- Hause, Thom, (2004) “A proper approach towards model partitioning can help in the establishment of a component-based development approach, leading to reuse of requirements, design, test, and implementation model artifacts.”
- Donna Rhodes (2019) “Although reuse of models can have benefits, the reality is that legacy models are not widely used beyond their original purpose.”
- Wu et al (2021) describes a maturity assessment of Systems Engineering reusable assets to facilitate MBSE adoption, basically a Capability Maturity Model (CMM) for model and asset reuse.
- So, although we understand the problem, what we lack are the standards and tools to promote and support reuse.
- SysML v2 may help via packaging mechanisms and its standard API.
- An OMG standard and third-party tools may also provide a solution.

Building the Right Models and Building the Models Right: Standards and Style Guides - Models

- SysML books and standards do not stress style
- Modeling Style Guide
  - Scott W. Ambler published The Elements of UML 2.0 Style in 2005
  - https://bilder.buecher.de/zusatz/15/15097/15097493_vorw_1.pdf
  - Useful for SysML modelers
- SAIC Digital Engineering Validation Tool
  - 226 Validation Rules (both language and style) for MagicDraw, Rhapsody
  - Customizations (including methods to connect deeply-nested ports, manage classification and data rights, and conduct failure analysis)
  - Model-Based Style Guide
  - Free to download
  - https://www.saic.com/digital-engineering-validation-tool
- Executable and testable models
  - With 3rd party tools and standalone

Building the Right Models and Building the Models Right: What Does “Right” Mean?

- DoDAF 2.0 emphasized data driven models and diagrams
  - – Diagrams are generated by the model
  - – Modifying the diagram modifies the model and vice versa
- Fit for Purpose Views
  - – Implies that we know the purpose of the view!
  - – What questions is the model/diagram meant to answer?
  - – Who is the intended audience?
  - – What is their level of MBSE sophistication?
  - – When to stop modeling?
  - – Etc.

Agility in Model Development and Management

- Today, many projects are in some phase of their first, large MBSE undertaking.
  - Many things are being done for the first time.
  - Very few projects or engineering teams have completed a full delivery lifecycle using MBSE.
- The principles of agile software development also apply to model development and lifecycle management.
  - Many initial assumptions may be wrong.
  - MBSE adoption and deployment cannot be implemented overnight.

Agility in the System of Innovation – This is an organization and culture issue!

- The System of Innovation (System 3) is supposed to provide agility through three basic principles:
  - Sensing
    - MBSE projects must monitor and measure key indicators of both project progress (a measure of System 2 performance) as well as data from System 1.
    - This requires more than assigning budget, scope, and schedule to the MBSE deployment activity.
    - It means recognizing that many assumptions made about the MBSE workflows, MBSE model architecture and network architecture may be incorrect and are yet to reveal themselves.
  - Respond
    - MBSE projects must make decisions about what they see and be prepared to react to address the actual performance of their MBSE deployment.
    - Detailed planning and compliance are not sufficient. Initial plans will need to evolve as new information comes to light.
  - Evolve
    - MBSE projects must embrace the fact that their process will evolve, and this evolution must be supported with a culture of experimentation, re-evaluation and new institutional memories.
- Effective deployment and lifecycle management processes will not develop overnight.
- It requires an agile approach!

Modeling Worst Practices

- Over constrained models with excessive user-defined model elements (stereotypes).
- Poor model architecture in terms of large monolithic models
- Poor “used project” structure with too many usage levels
- Circular references between used projects.
- Poor package structure and failure to comply with package content rules resulting in duplicate elements and broken traceability.
- Poor management or enforcement of reuse libraries.
- Inclusion of “sandbox” packages, diagrams and model elements (used by modelers to facility their modeling activity) in the final delivery.

How to Kill a Modeling Tool

- Overuse of processor intensive tool features:
  - Large tables that “query the world”
  - Derived properties in tables
  - Derived properties with circular references
  - Dynamic legends
  - Opaque behaviors
  - Smart packages
  - Queries and structured expressions
- All these are useful, but must be used sparingly.
- Modeling validation suites can uncover these.

Conclusion

- The advancement and maturity of the MBSE community is accelerating.
- Projects are performing engineering with and demanding more of MBSE models than ever before.
- Tool technology is advancing, but not at the pace needed by the MBSE community.
- While some of the issues will require advancement of tool features and other technologies, the majority can be traced back to project culture and the difficulties associated with culture change.
- Successful execution of large scale MBSE projects requires planning and rigor beyond that developed for document-based systems engineering.

Summary

- The model-based digital ecosystem is itself, a complex software intensive system that requires the same systems engineering technical process required for development of the SOI itself.
- The problems being experienced by large scale MBSE projects are not unique and certainly not new.
- The MBSE modeling imperatives in this paper are intended to help address the problems currently experienced by large MBSE projects and to provide guidance to future projects.

About the Authors

Barry Papke is the Manager for the Cyber System Industry Process Success Team for CATIA/No Magic. He has thirty-five years of systems engineering and operations analysis experience in the aerospace and defense industry across the entire systems engineering lifecycle from concept development through integration, test and post-delivery support. Throughout his career, he has been actively involved in application of model-based methods including requirements management, enterprise architecture, cost estimation, system design, and operations analysis. Barry has a Bachelor of Science in Mechanical Engineering from Texas A&M University and a Master of Science in Systems Engineering from Steven's Institute of Technology. He is a member of the INCOSE Agile and Security Working Groups and the MBSE Initiative.

David Hetherington is a Principal Systems Engineer with System Strategy, Inc. He serves multiple defense and commercial industry sectors. He has extensive personal experience in designing and leading design teams for both software and hardware covering an unusually broad range of system types. These complex systems have varied from real-time control, to software internationalization, to offshore oil drill ships, to enterprise software applications, to automotive radar chipsets,
to electronic publishing, and more. In addition to MBSE, he has a strong concentration of domain knowledge in safety, reliability, maintainability, and diagnostics.

Matthew Hause is an SSI Principal and MBSE Technical Specialist, a former PTC Fellow, a co-chair of the UAF group and a member of the OMG SysML specification team. He has been developing multi-national complex systems for over 45 years as a systems and software engineer. He started out working in the power systems industry and has been involved in military command and control systems, process control, manufacturing, factory automation, communications, SCADA, distributed control, office automation and many other areas of technical and real-time systems. His roles have varied from project manager to developer. His role at SSI includes mentoring, sales presentations, standards development, presentations at conferences, specification of the UAF profile and developing and presenting training courses. He has written over 100 technical papers on architectural modeling, project management, systems engineering, model-based engineering, and many other subjects.

About the Authors

Sean McGervey is the Director of MBSE Services at Dassault Systèmes. Previously, Sean was a Systems Engineer and MBSE Instructor at the Johns Hopkins University Applied Physics Laboratory, where he was the Architecture Lead on an ACAT-1 Program and supported the Digital Engineering Transformation of APL's Sponsors. Sean founded the APL MBSE Community of Practice and taught three internal courses in MBSE. Prior to APL, Sean worked for 15 years in the Systems Engineering Department at Northrop Grumman in Baltimore, Maryland. While there, Sean founded the Northrop Grumman Corporate Model-Based Engineering (MBE) Community of Practice. Sean is President-Elect of INCOSE's Chesapeake Chapter and Chairperson-Emeritus of the INCOSE Digital Engineering Information Exchange Working Group (DEIXWG). Sean is also an Adjunct Professor of Systems Engineering at Johns Hopkins University where he teaches a graduate course titled "Applied Analytics for MBSE".

Sami is an Aerospace Engineer turned MBSE Consultant at STC with experience working in Systems Engineering, Modeling, and Enterprise Architecture since 2018. Before joining STC, Sami was a Senior Consultant at Deloitte Consulting and a System Architect for the NASA Advanced Air Mobility (AAM) Mission Integration Office (AMIO) providing Model Based System Engineering (MBSE) expertise (SysML and UAF) for various AAM Projects, US Space Force, and NAVAIR. He holds the OCSMP Model Builder Fundamental certification.

Questions?

33rd Annual INCOSE international symposium

hybrid event

Honolulu, HI, USA
July 15 - 20, 2023

www.incose.org/symp2023
#INCOSEIS