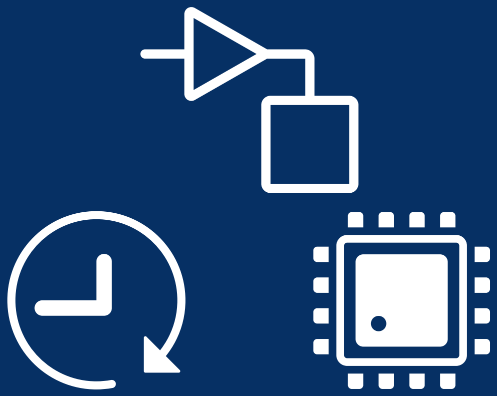
Model-Based Systems Engineering for Machine Control Software

Rareș Curatu

Industry Manager, EMEA

Agenda

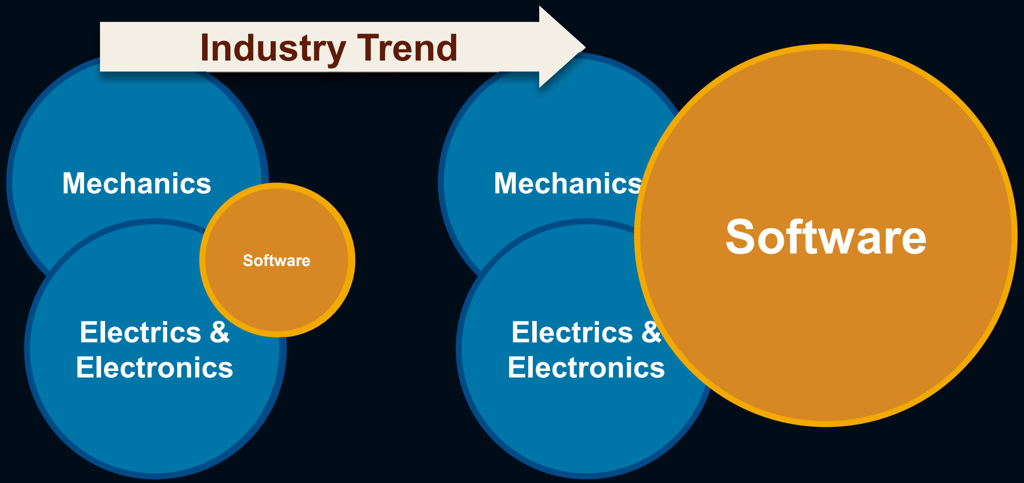


3 Engineering Imperatives and the Role of MBSE

Model-Based Systems Engineering for Machine Control Software

Increasing Software Content

- OEE
- TTM
- Novel Features
- TCO
- Sustainability



Increased Software Content

Increased Complexity & Opportunities

Increased Software Content

Challenges

- Increased Complexity
- Increased Engineering & Development effort

Opportunities

- Design improvement
- New services and products

Software

Software Development with Model-Based Design:

3 Engineering Imperatives

3 Engineering Imperatives

Mastering Mechatronic Complexity and Variant Management

Continuous Simulation & AI Validation

Architectural Flexibility & Algorithm Portability

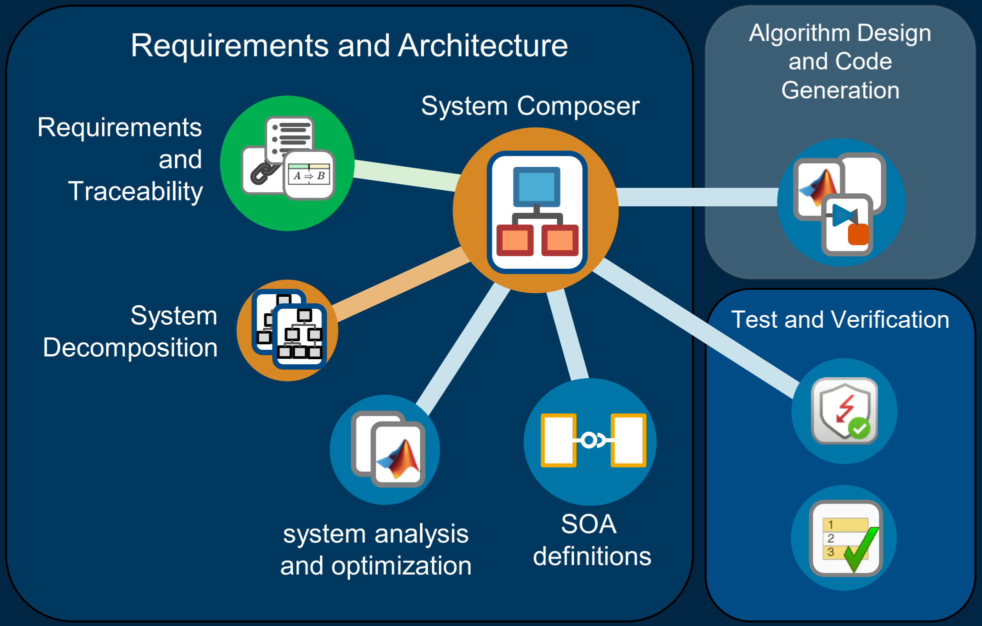
3 Engineering Imperatives

Mastering Mechatronic Complexity and Variant Management

The Bottleneck: "Configure-to-Order" hardware creates a combinatorial explosion of software and system variants.

The Shift: Moving from disconnected documents to systematic, model-based architectures.

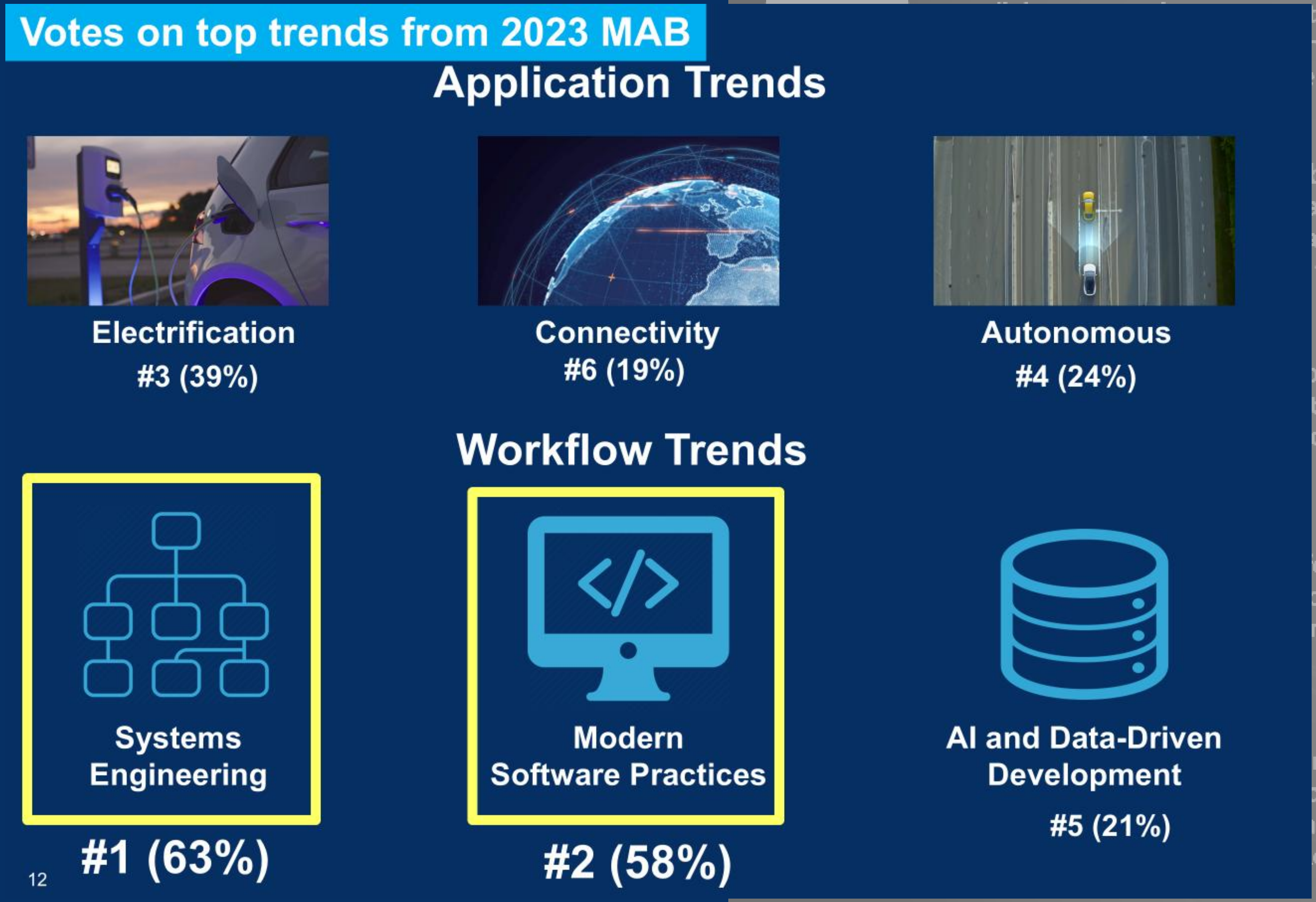
The Value: Manage complex machine platforms from a single source of truth, eliminating copy-paste engineering.



Continuous Simulation & AI Validation

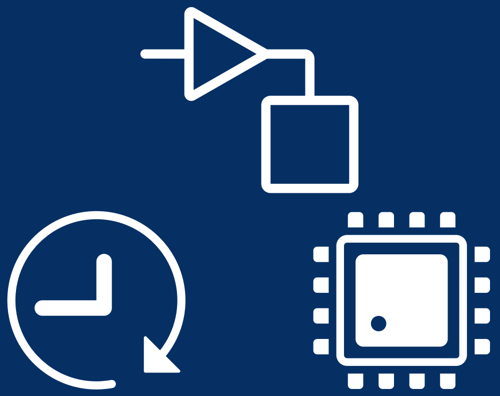
Architectural Flexibility & Algorithm Portability

System Engineering as Top Industry Trend



INCOSE Future of Systems Engineering Vision

Agenda



3 Engineering Imperatives
and the Role of MBSE

Model-Based Systems Engineering for Machine Control Software

Model-Based Systems Engineering for Machine Control Software

What is MBSE, and How Does It Relate to MBD?

MBSE Workflow and Tools

MBSE and Variant Management at Siemens Energy

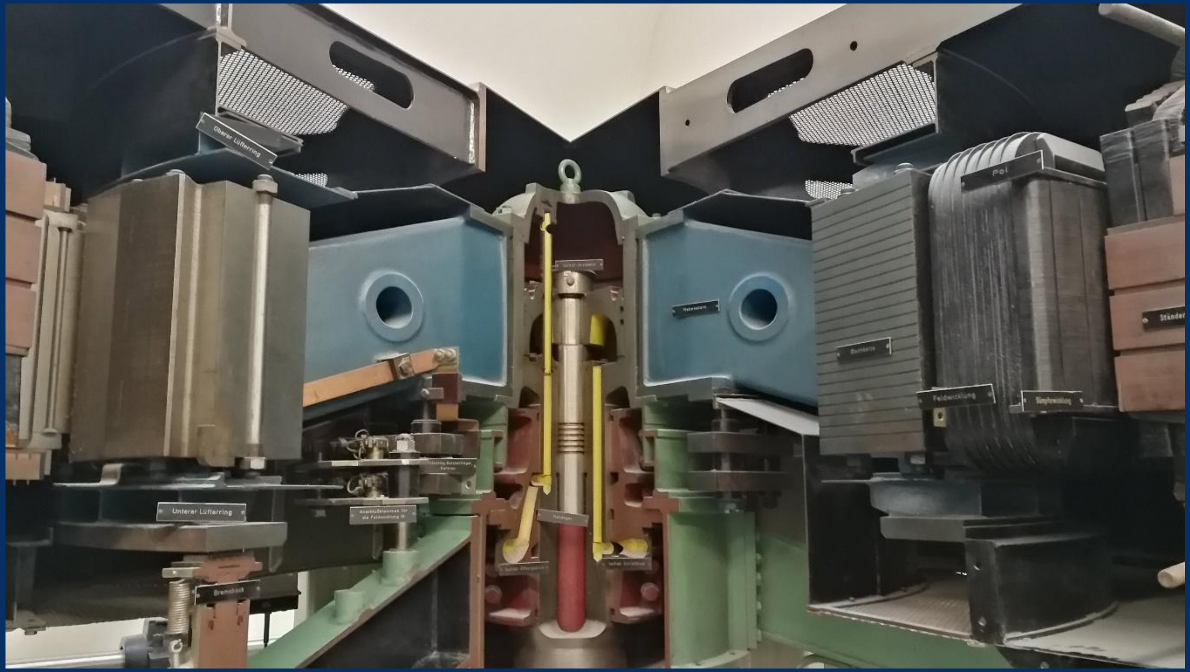
The future of MBSE for industrial automation and machinery, resources, and conclusions

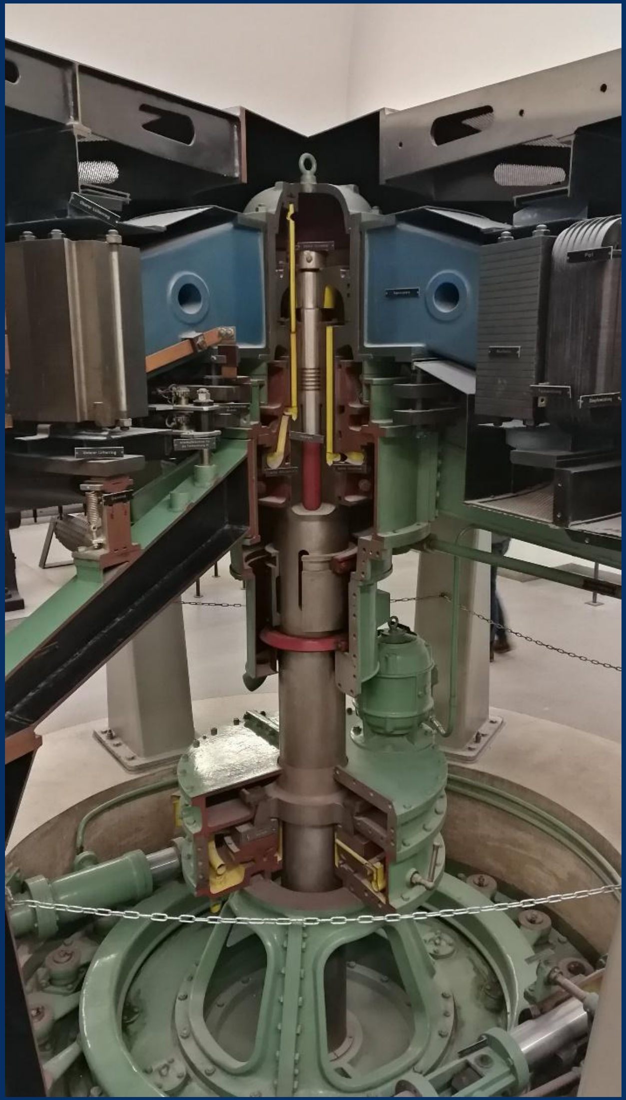
What is MBSE, and How Does It Relate to MBD?

Control a hydro turbine

Numerous components and control loops

- Guide vane, runner blade pitch, bypass valve, rotor breaks
- Main governor sump, positive displacement pumps, accumulators, control valve manifold





Images: photographs of Deutsches Museum Munich exhibits, by Rareș Curatu

Control a hydro turbine
Numerous components and control loops

- Design the control logic based on requirements
- Document-based
- Manual PAC programming

Image by Rareș Curatu

Requirements (documents) → Control logic (documents) → Review and iteration → Implementation on PLC (manual coding) → Test in the office (“manual DI/O emulation”) → Test on site

Challenges?

Yes

Challenges?

- No testing capabilities during design:
  - Individual control loop
  - Dependencies between different control loops and subsystems
- No link between:
  - Requirements
  - Control design
  - Control implementation
  - PLC/PAC testing
- Low reusability of code
- No system model

What if we had multiple projects?

- Engineer- or Configure-to-Order
  - No two systems are identical
  - Re-use of similar components
  - Re-use of similar control logic
  - But different details!
- Challenges:
  - How do we avoid having *n* control design copies and variants for *n* projects?
  - How do we maintain and improve the reference design?
  - How do we prove the coverage of requirements?
  - How do we prove compliance?

How would a Model-Based approach look?

Model-Based Design

Systematic use of **models** throughout the development process

Modeling

Simulation **Analysis**

+

Automation

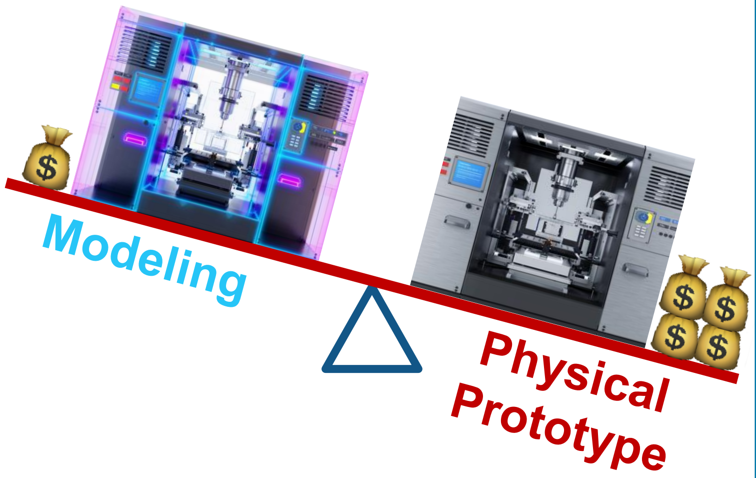
Coding **Verification**

Try out new ideas
Fast repeatable tests

Eliminate manual steps
and **human error**

Challenges of "traditional" development workflows

Physical prototypes are **costly** and iterate **slowly**



Design-space **exploration**

Continuous design **improvement**

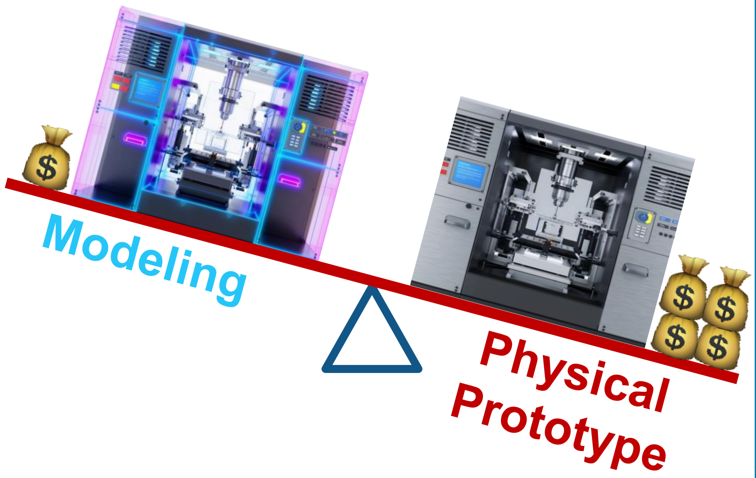
Virtual Commissioning

Costly and **time-consuming** to build

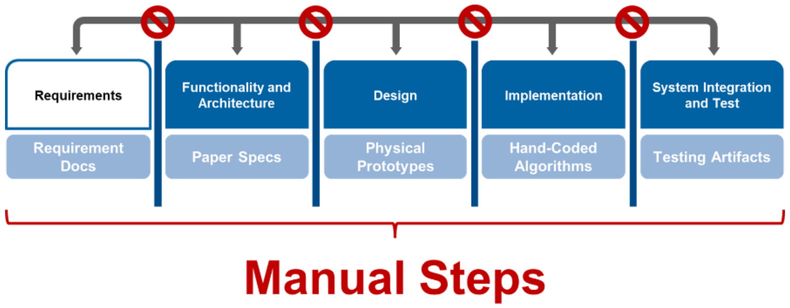
Hinders rapid iterations

Challenges of “traditional” machinery development workflows

Physical prototypes are **costly** and iterate **slowly**



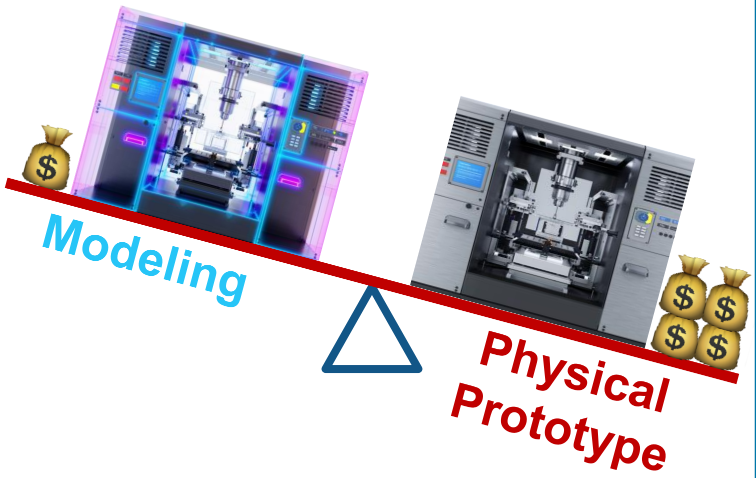
Requirements and artifacts are **hard** to **manage**, **change**, and **trace**



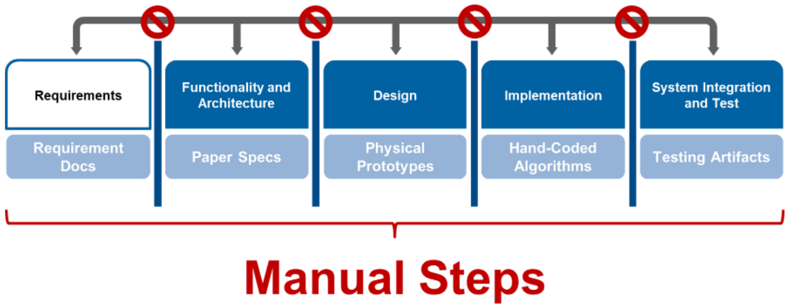
Manual steps introduce **errors** and **slow down** the development process

Challenges of “traditional” machinery development workflows

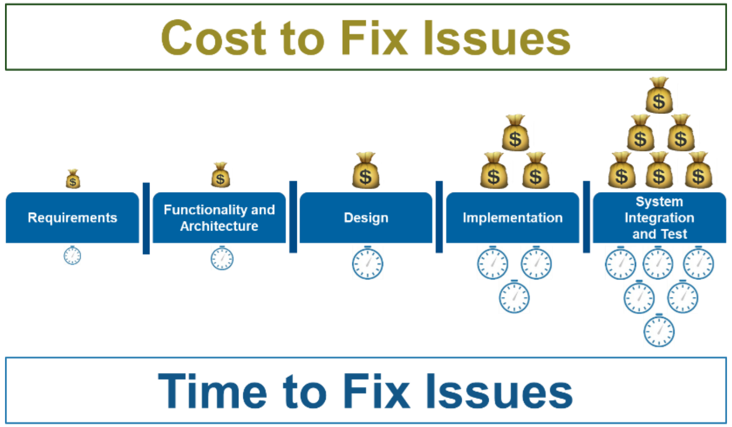
Physical prototypes are **costly** and iterate **slowly**



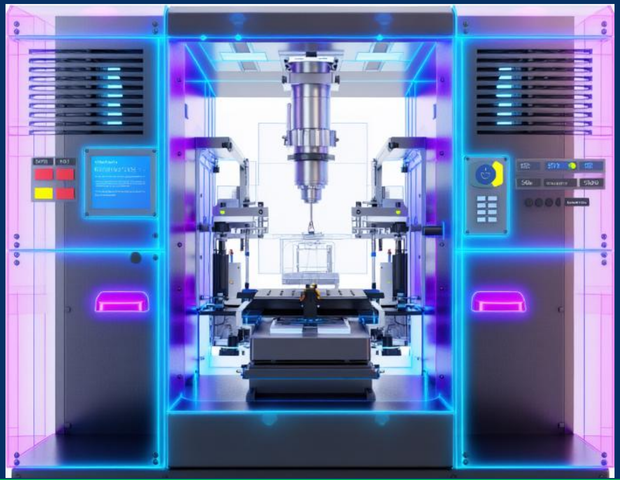
Requirements and artifacts are **hard** to **manage**, **change**, and **trace**

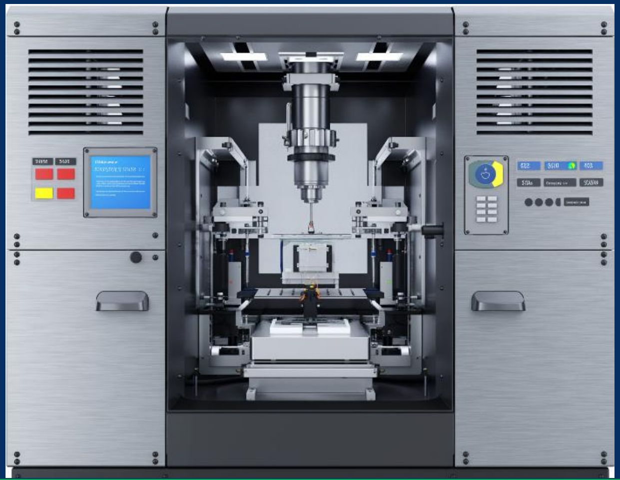


Issues found late in the process are more **costly** and **time-consuming** to fix



Modeling and simulation help cut costs and speed up design iterations





Modeling

Simulation | Analysis

Try out new ideas

Fast repeatable tests

Automation

Coding | Verification

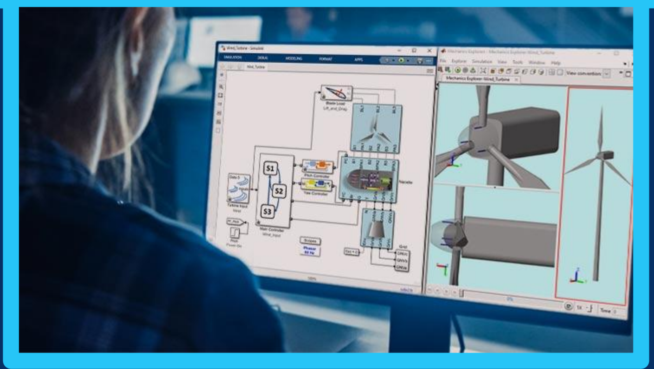
Eliminate manual steps

and human error

Requirements **capture** and artifact **traceability** throughout the process

Model-Based Design

Requirements | Functionality and Architecture | Design | Implementation | System Integration & Test



Models are at the **center** of your development process

Create a **digital thread**

Systematic Use of Models in product Engineering and Development

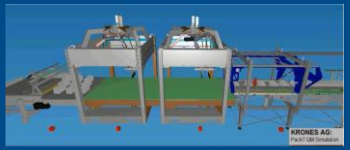
Authoritative Source of Truth ▪ Reuse/Refine ▪ Value at Multiple Stages

Requirements

- Use Cases
- Requirement docs & models

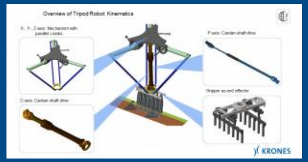
Functionality and Architecture

- Behavior models
- Architecture/ICD
- Multi-domain



Design

- Physics-based
- Data-driven
- AI
- Hybrid



Implementation

Code generation:

- C/C++
- Structured text
- GPU code

Docker containers

System Integration & Test

- Model-based V&V
- Code-based V&V
- Virtual Commissioning

◄ shift-left

Digital Thread

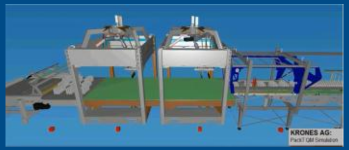
Systematic Use of Models in the Lifecycle

Authoritative Source of Truth ▪ Reuse/Refine ▪ Value at Multiple Stages

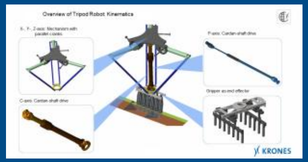
Requirements
- Use Cases
- Requirement docs & models

ReqIF

Functionality and Architecture
- Behavior models
- Architecture/ICD
- Multi-domain



Design
- Physics-based
- Data-driven
- AI
- Hybrid



Implementation
Code generation:
- C/C++
- Structured text
- GPU code

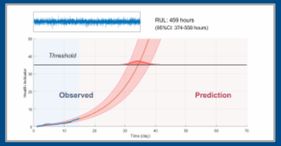
Docker containers

</>
docker

System Integration & Test
- Model-based V&V
- Code-based V&V
- Virtual Commissioning

◀ shift-left

Operation and Sustainment
- Predictive maintenance
- Digital Twins



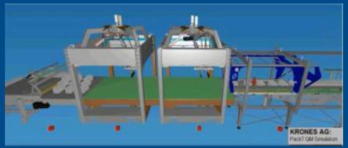
Digital Thread

Systematic Use of Models in the Lifecycle

Continuity between MBSE and MBD Workflows

Requirements
- Use Cases
- Requirement docs & models

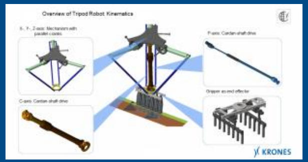
Functionality and Architecture
- Behavior models
- Architecture/ICD
- Multi-domain



Descriptive Models

Model-Based Design

Design
- Physics-based
- Data-driven
- AI
- Hybrid



Implementation
Code generation:
- C/C++
- Structured text
- GPU code

Docker containers

</> docker

System Integration & Test
- Model-based V&V
- Code-based V&V
- Virtual Commissioning

◀ shift-left

Design Models

Operation and Sustainment
- Predictive maintenance
- Digital Twins

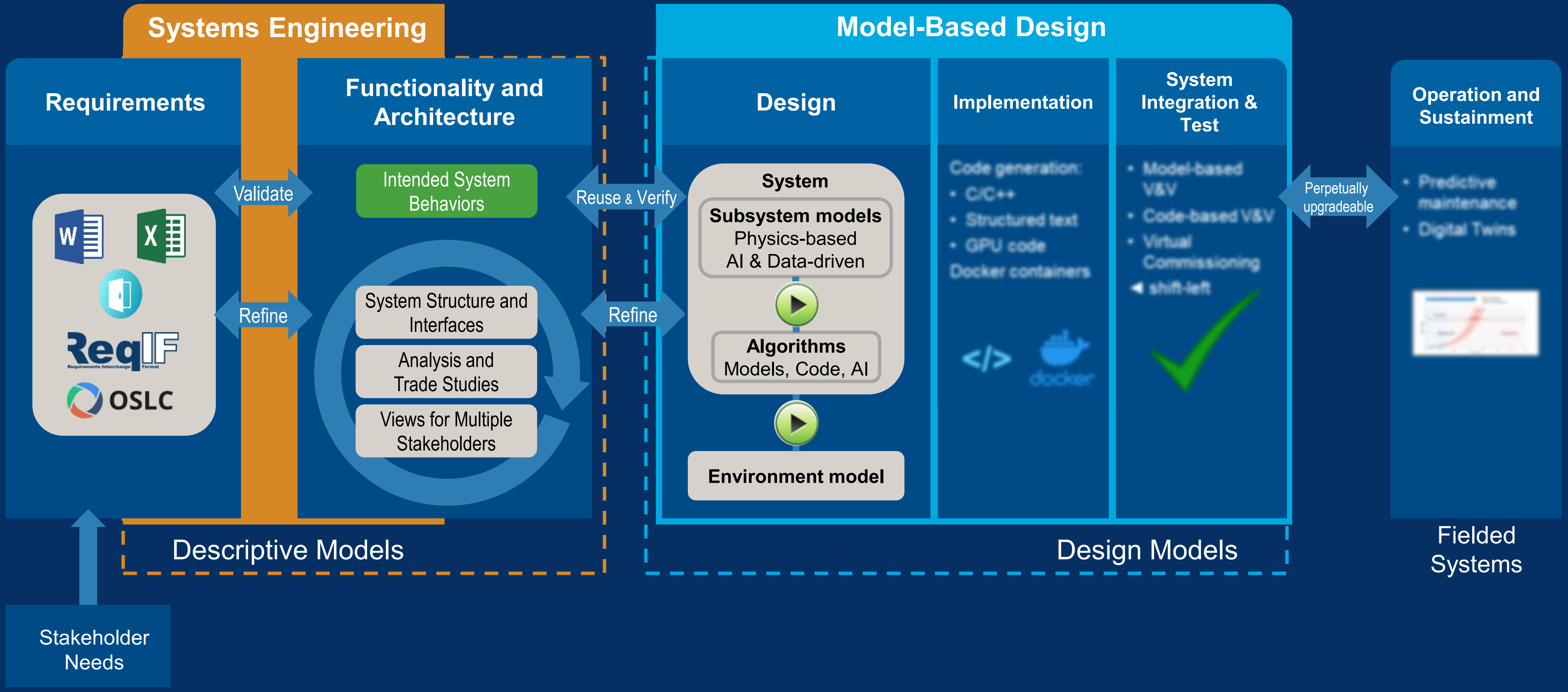
Fielded Systems

Digital Thread

MathWorks

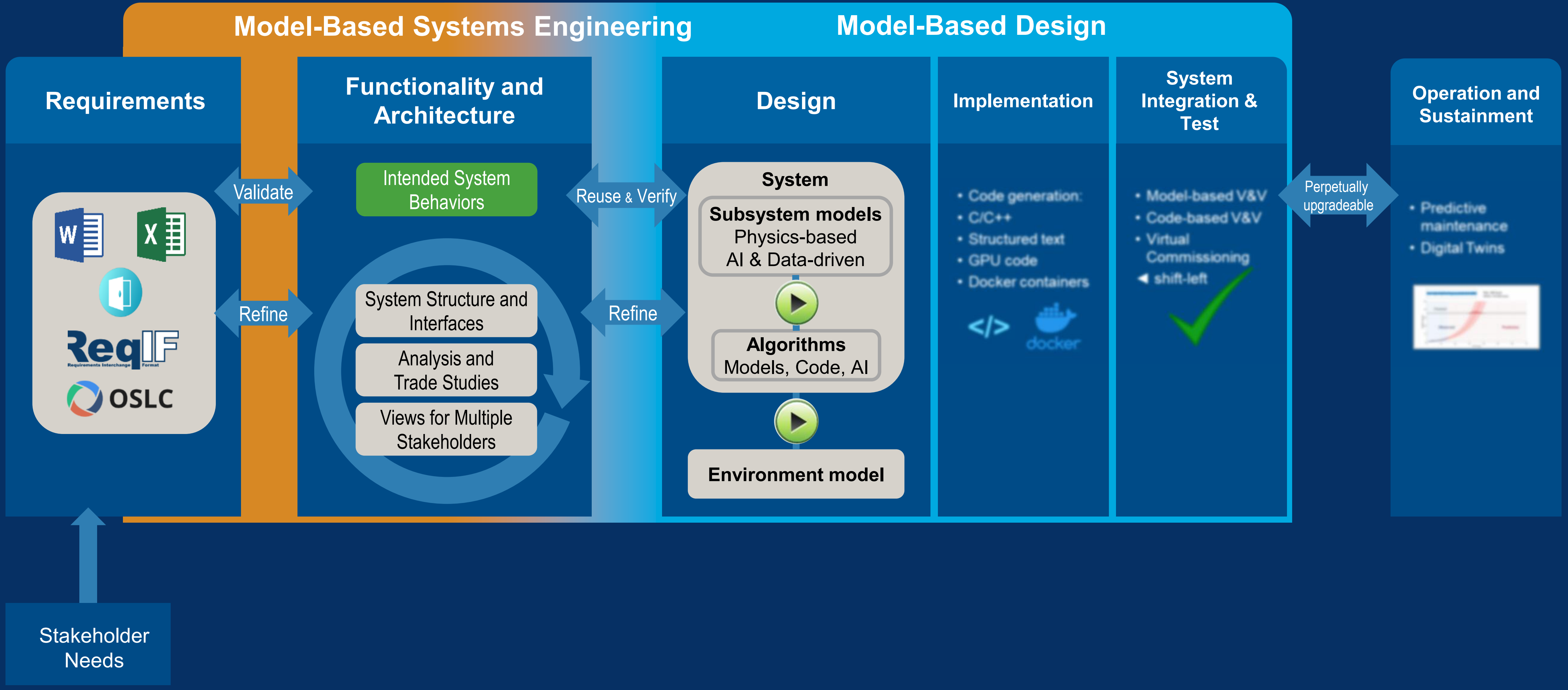
Systematic Use of Models in the Lifecycle

Continuity between MBSE and MBD Workflows



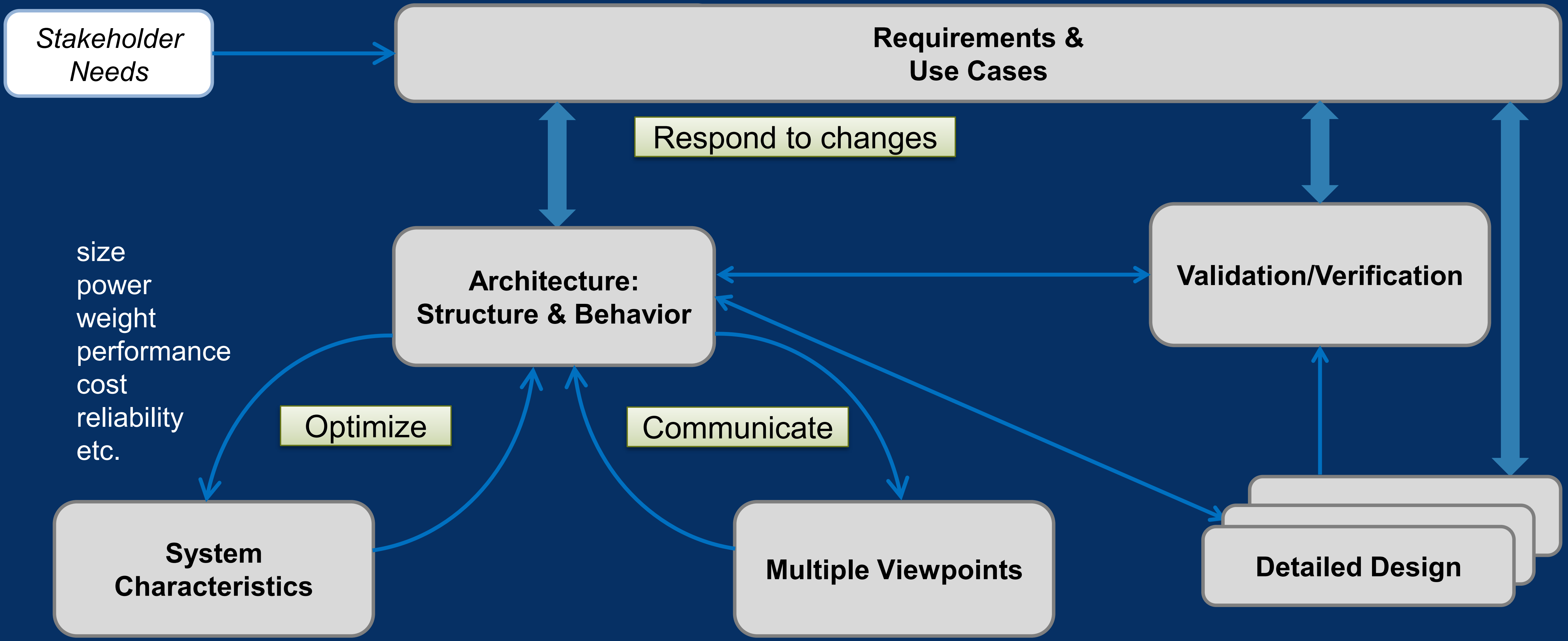
Systematic Use of Models in the Lifecycle

Continuity between MBSE and MBD Workflows



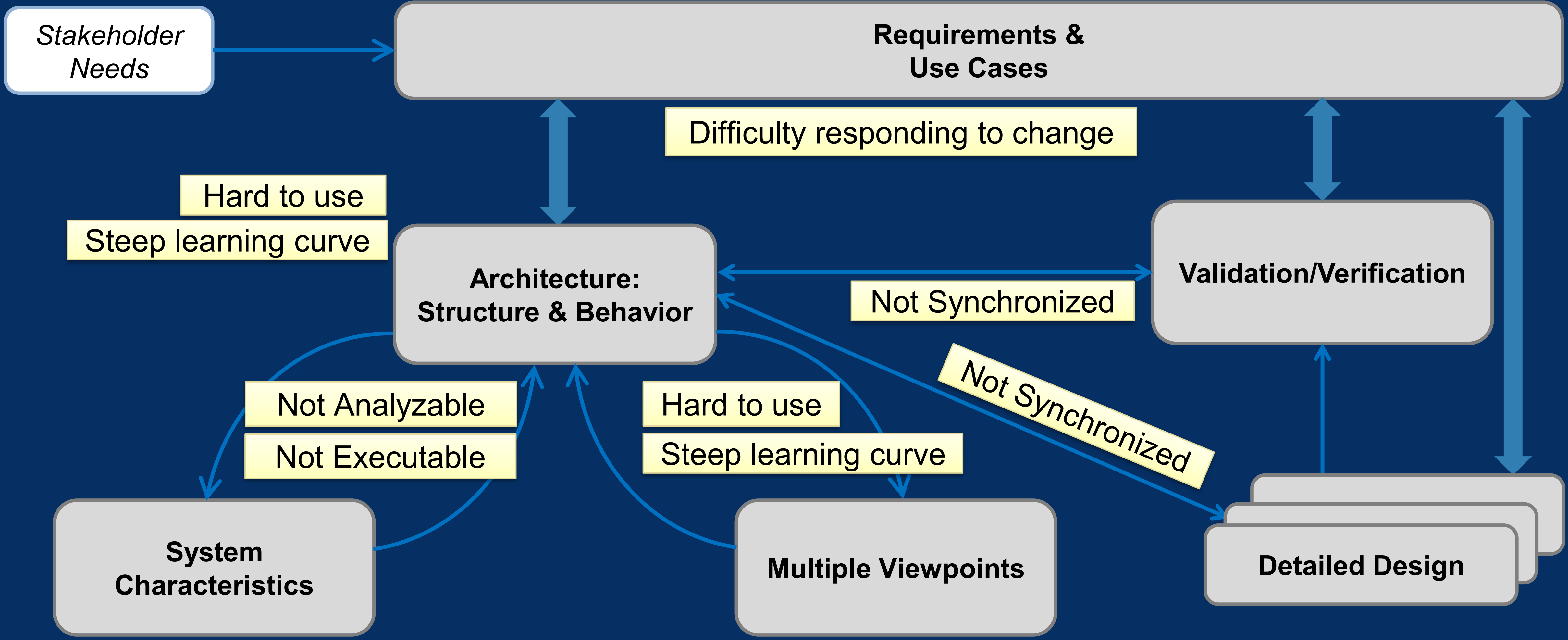
MBSE Workflow and Tools

System Engineering/Development Workflow

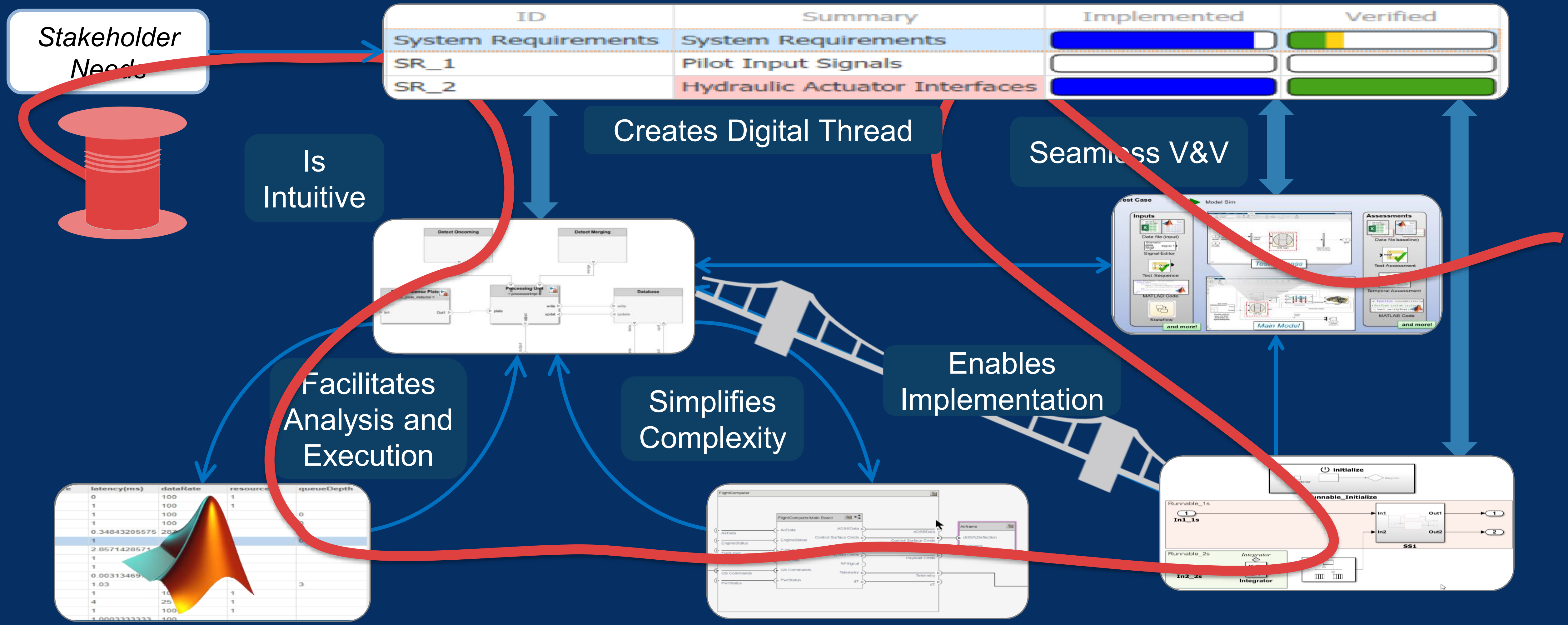


size
power
weight
performance
cost
reliability
etc.

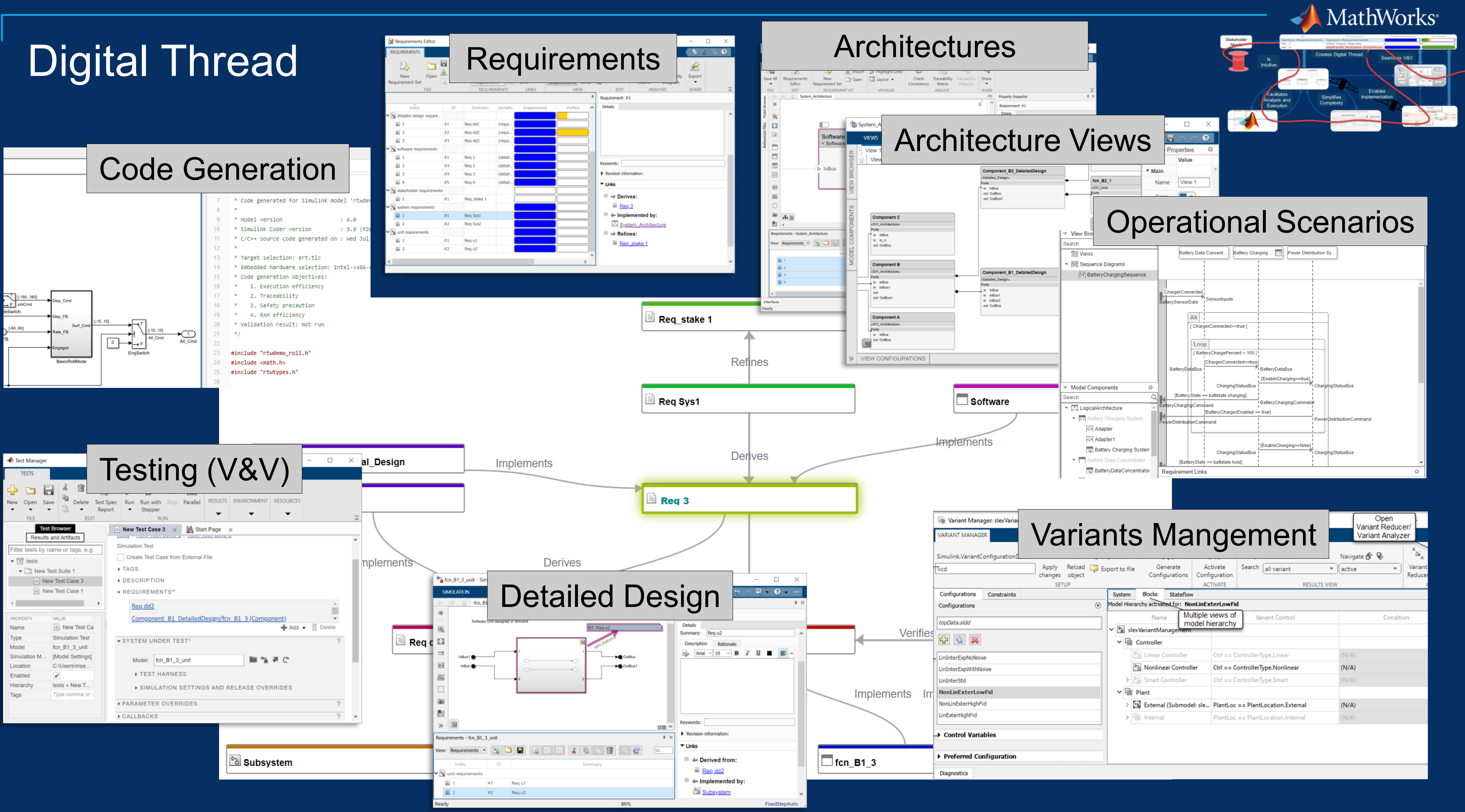
System Engineering/Development Common Users Challenges



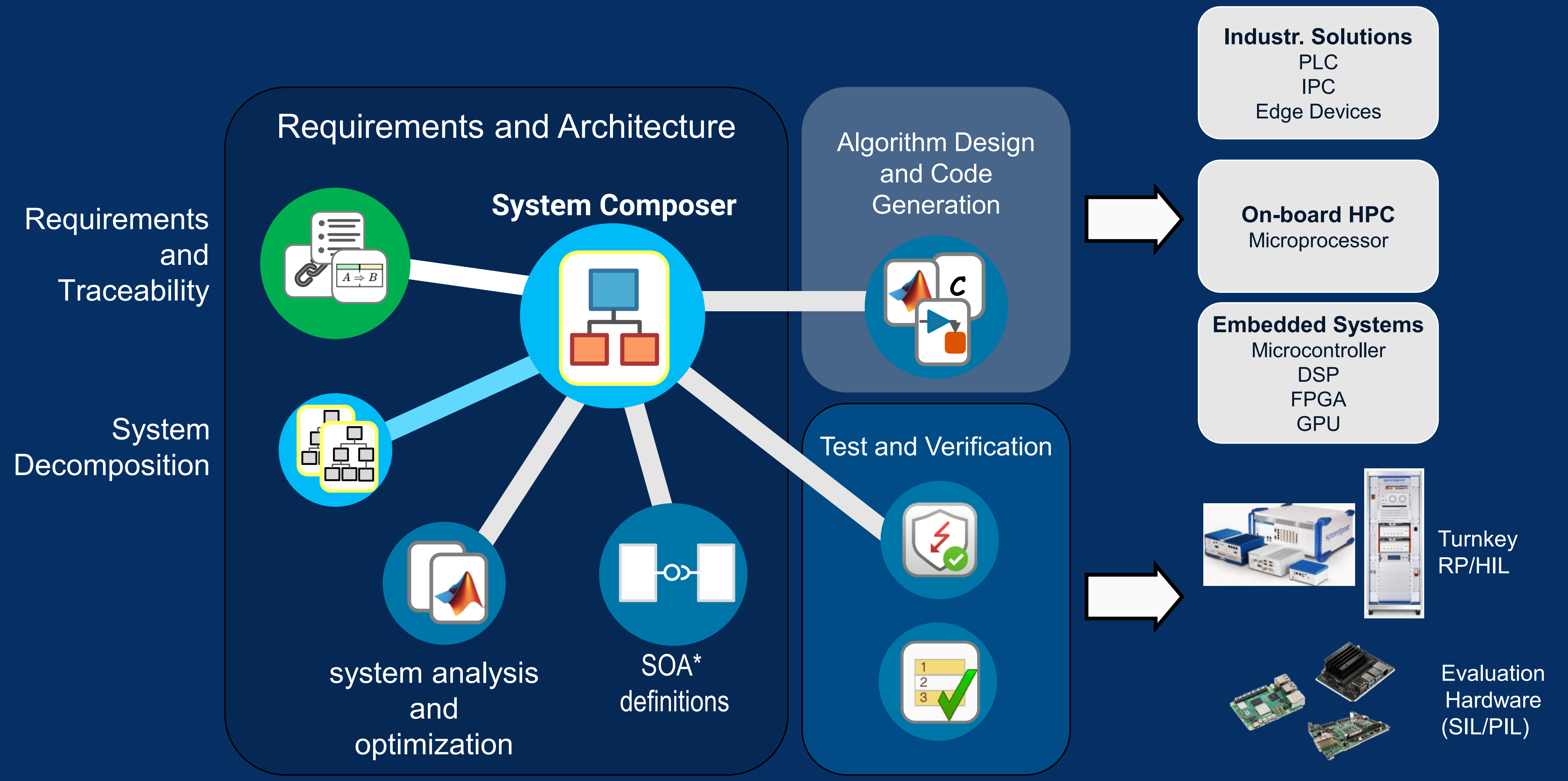
System Engineering/Development Unified Environment



Digital Thread



Build models that span the entire development cycle.



*SOA =Service-Oriented Architecture

MBSE and Variant Management at Siemens Energy

What is MBSE, and How Does It Relate to MBD?

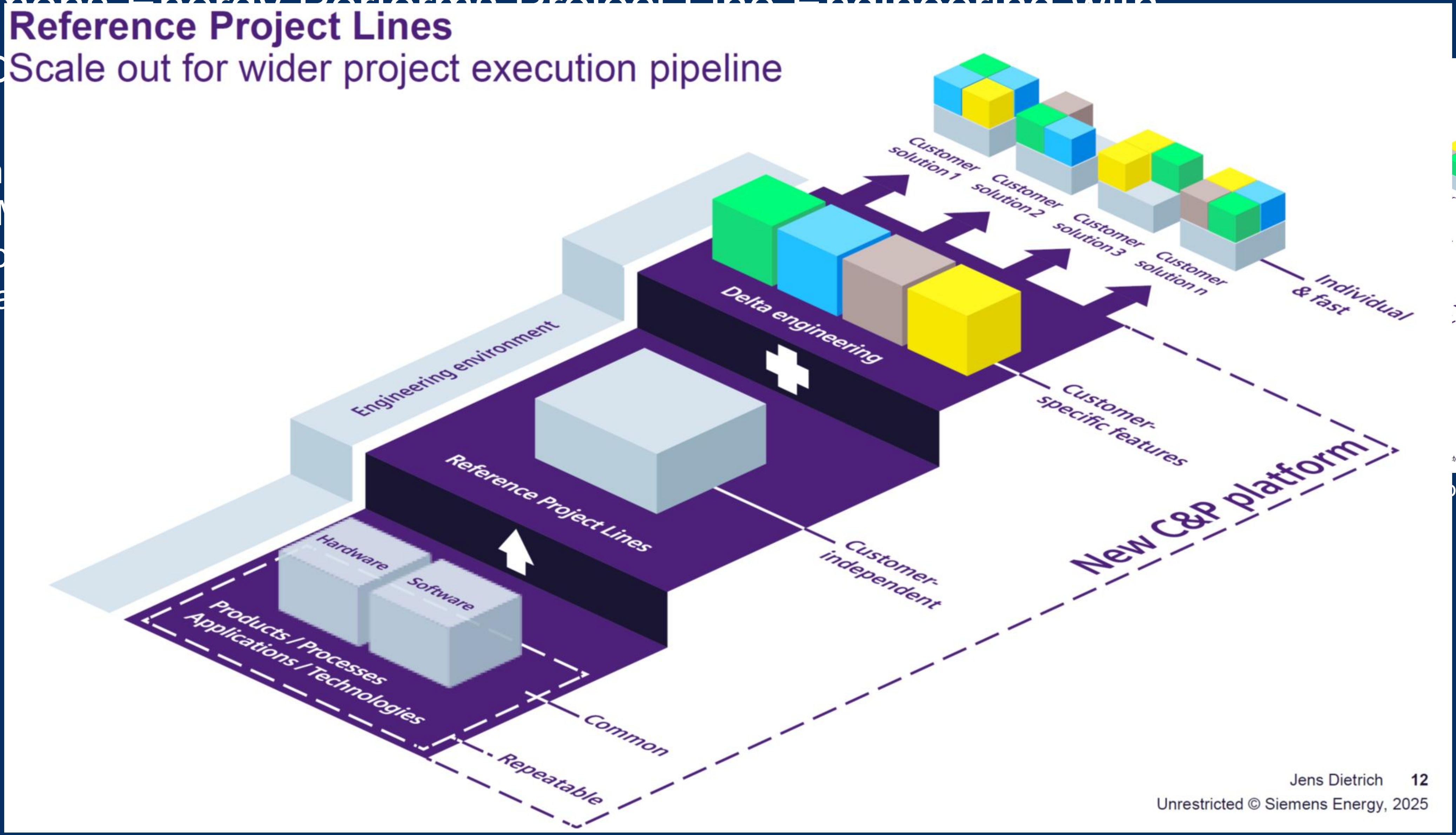
MBSE Workflow and Tools

>> MBSE and Variant Management at Siemens Energy

The future of MBSE for industrial automation and machinery, resources, and conclusions

Siemens Energy Performs Project Line Engineering with Mod

Siem
with M
comp
repea



n pipeline

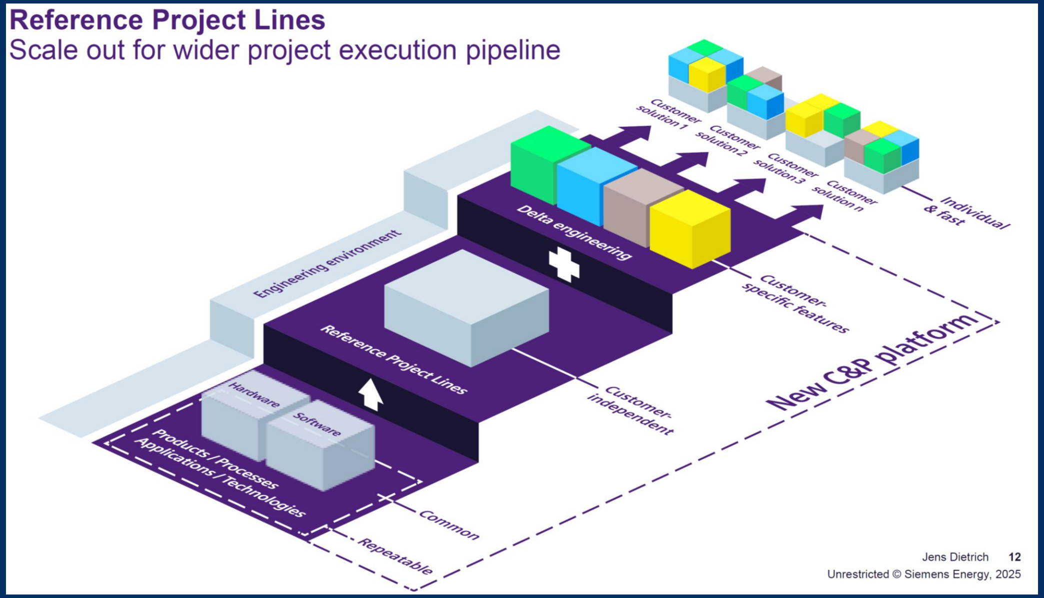
[Link to presentation video]

Siemens Energy Performs Project Line Engineering with Model-Based Systems Engineering

Siemens Energy uses Model-Based Systems Engineering with MATLAB, Simulink and System Composer to make complex Configure-to-Order projects more easily repeatable.

Key Take-Aways:

- **Validate requirements early:** A system-level simulation at project start confirms behavior with the customer before any hardware is committed.
- **Variant management:** One reusable baseline carries every option and is trimmed per project, serving many configurations without re-engineering from scratch.
- **More projects in parallel:** This reuse let them scale from a handful of expected projects to around twenty running at once.

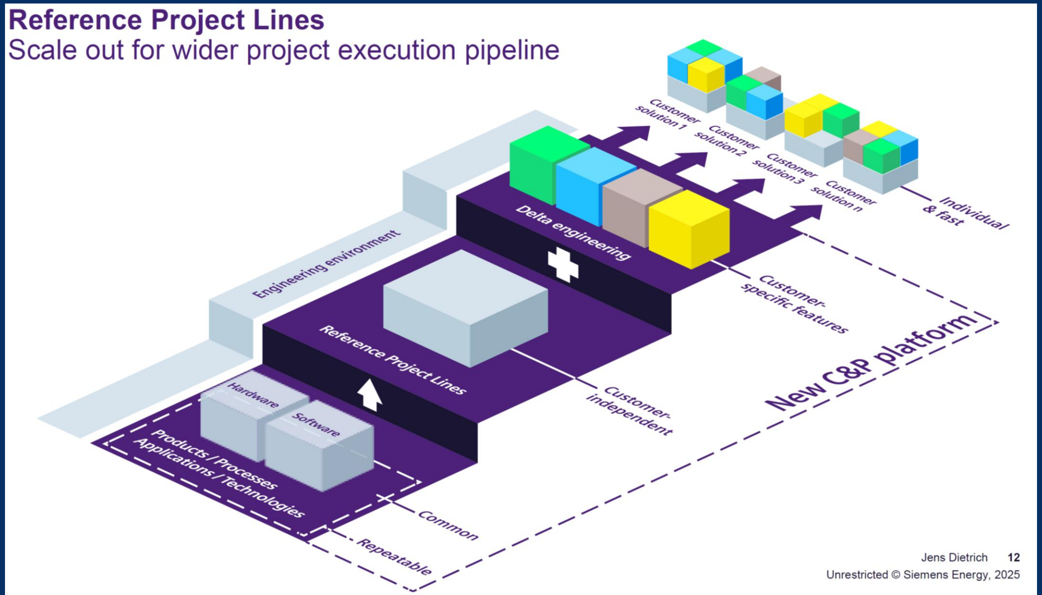


Reference Project Lines: scale out for wider project execution pipeline

Link to presentation video

Siemens Energy Performs Project Line Engineering with Model-Based Systems Engineering

Siemens Energy uses Model-Based Systems Engineering with MATLAB, Simulink and System Composer to make complex Configure-to-Order projects more easily repeatable.



Reference Project Lines: scale out for wider project execution pipeline

Key Take-Aways:

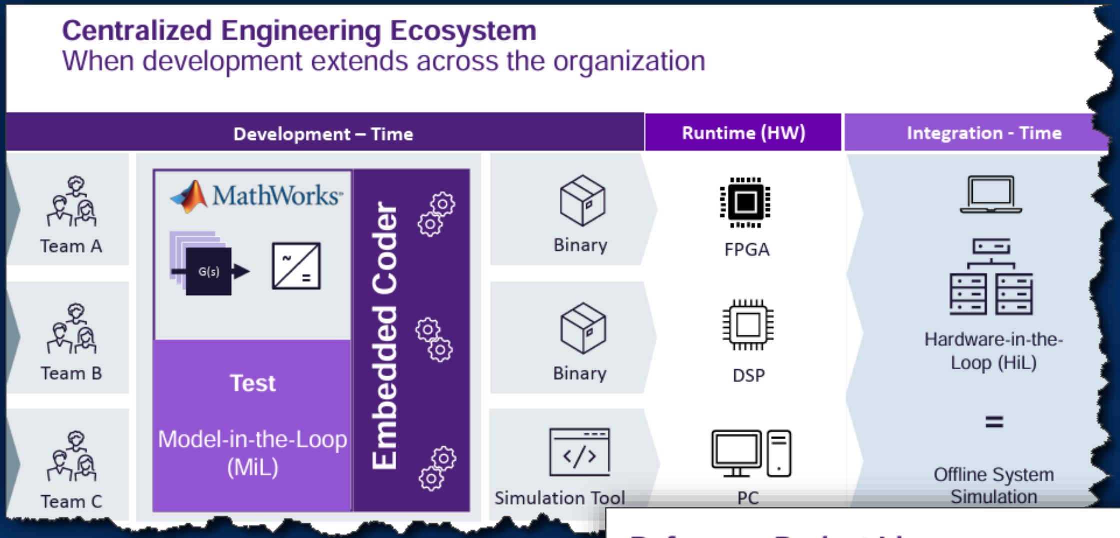
- **Validate requirements early:** A system-level simulation at project start confirms behavior with the customer before any hardware is committed.
- **Variant management:** One reusable baseline carries every option and is trimmed per project, serving many configurations without re-engineering from scratch.
- **More projects in parallel:** This reuse let them scale from a handful of expected projects to around twenty running at once.

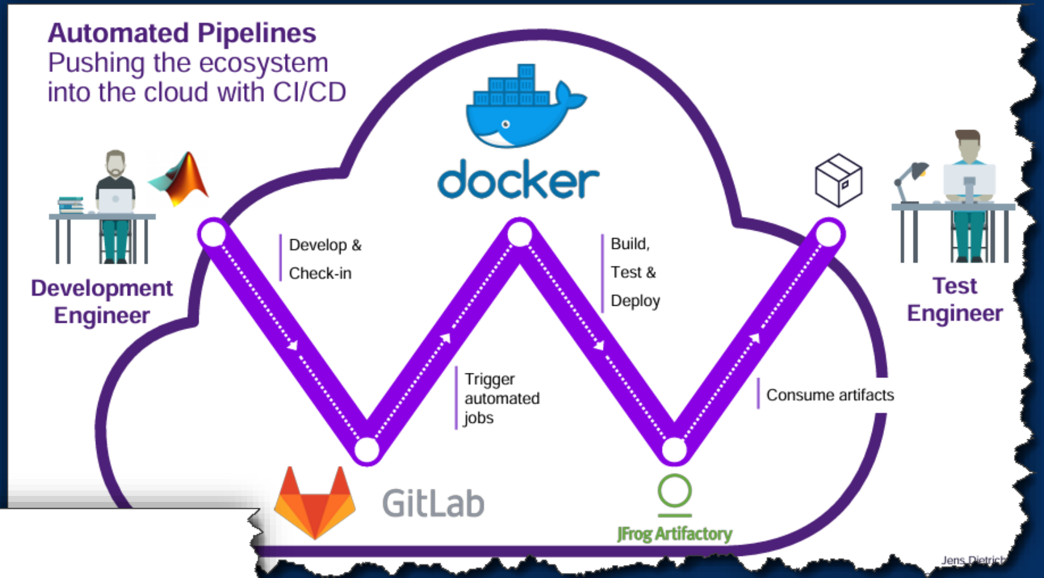
"The goal is to have a running model at the beginning of the project, so we can focus on what really matters — talking to the customer and understanding the requirements and engineering a system that he/she really needs."

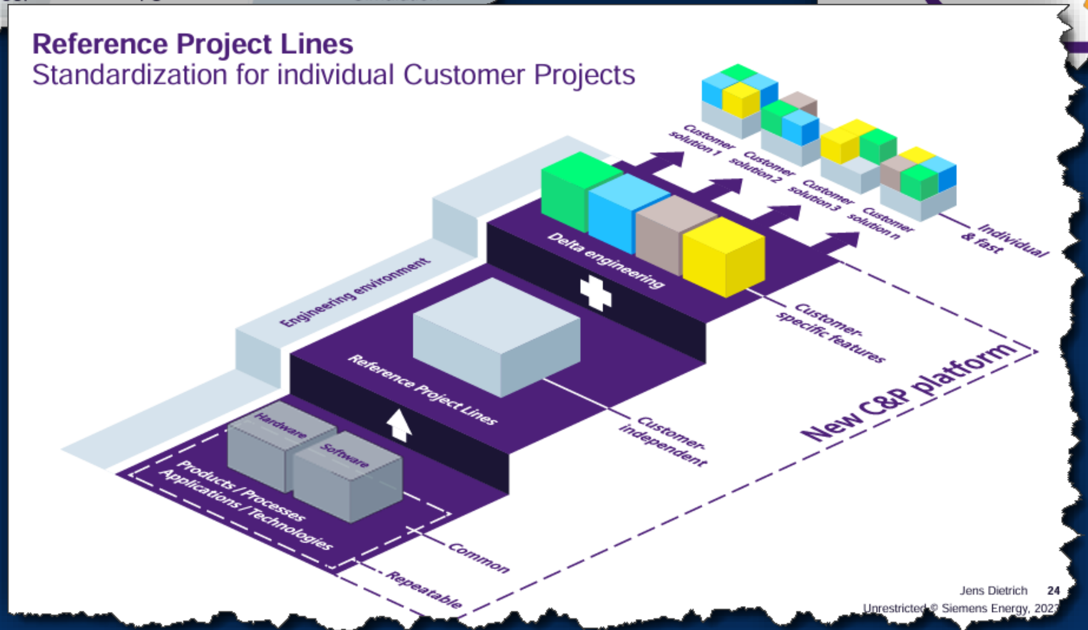
- Jens Dietrich, head of transmission software development, Siemens Energy

Link to presentation video

Siemens Energy using Model-Based Design and Model-Based Systems Engineering across Development Organization







Full user story & Short video summary

MATLAB EXPO

The future of MBSE for industrial automation and machinery, resources, and conclusions

Over the past decade, system engineering has evolved significantly **from document-based to model-based**

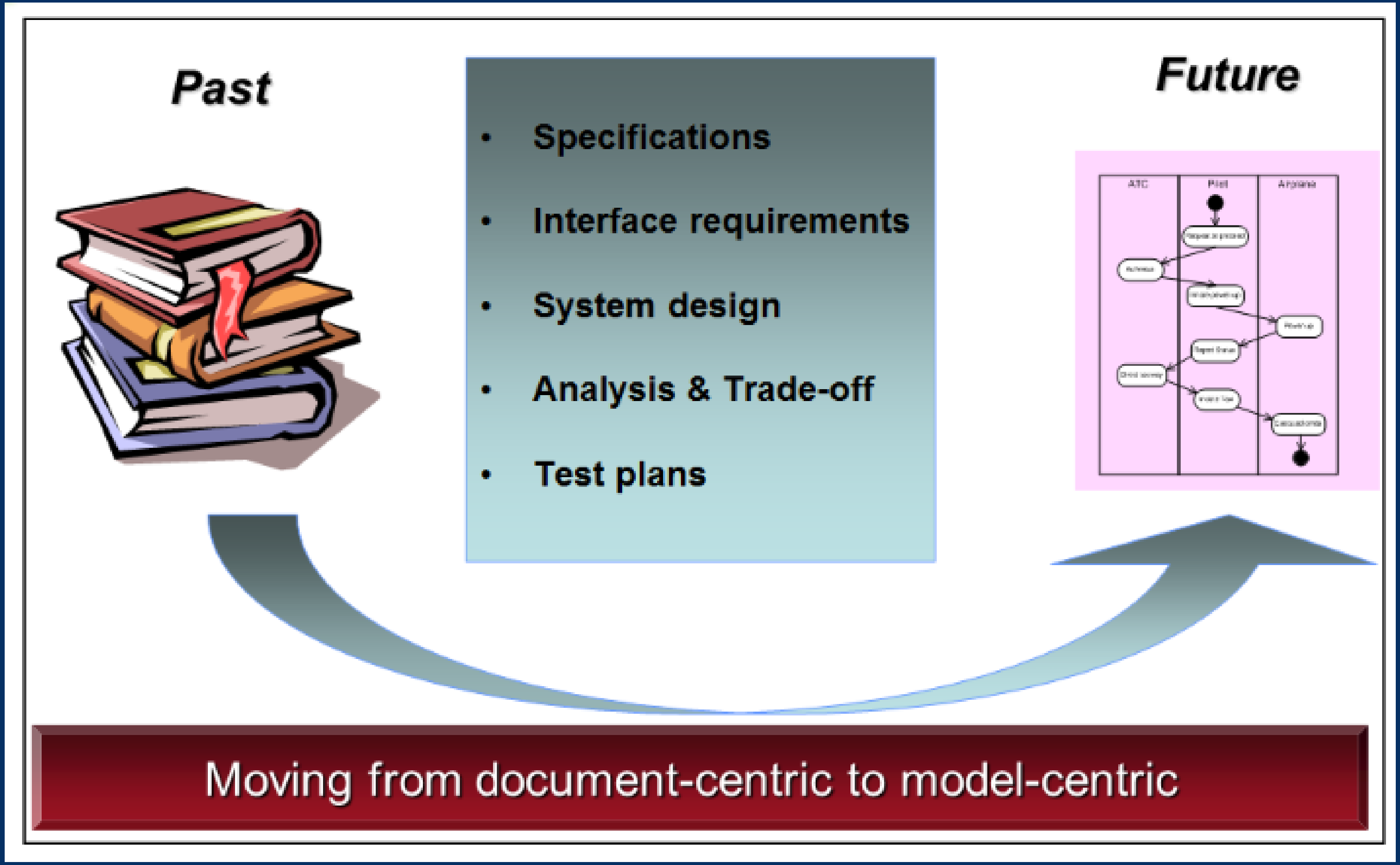
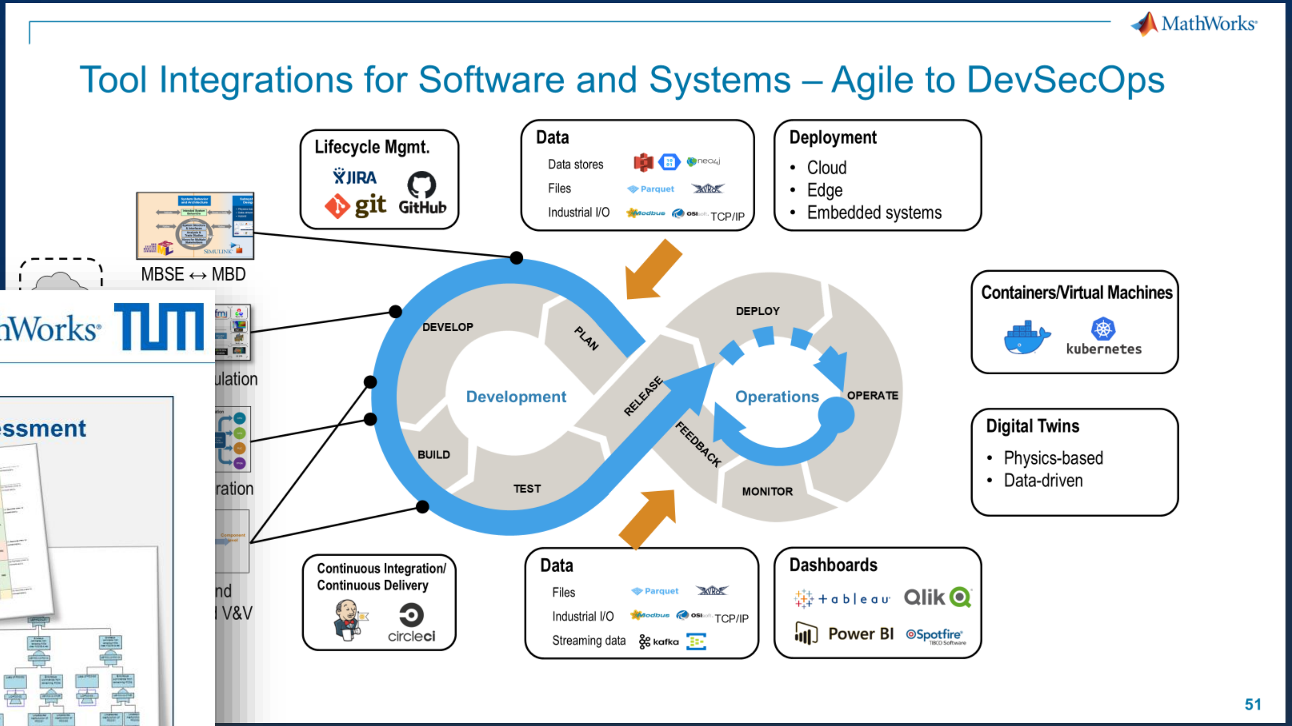
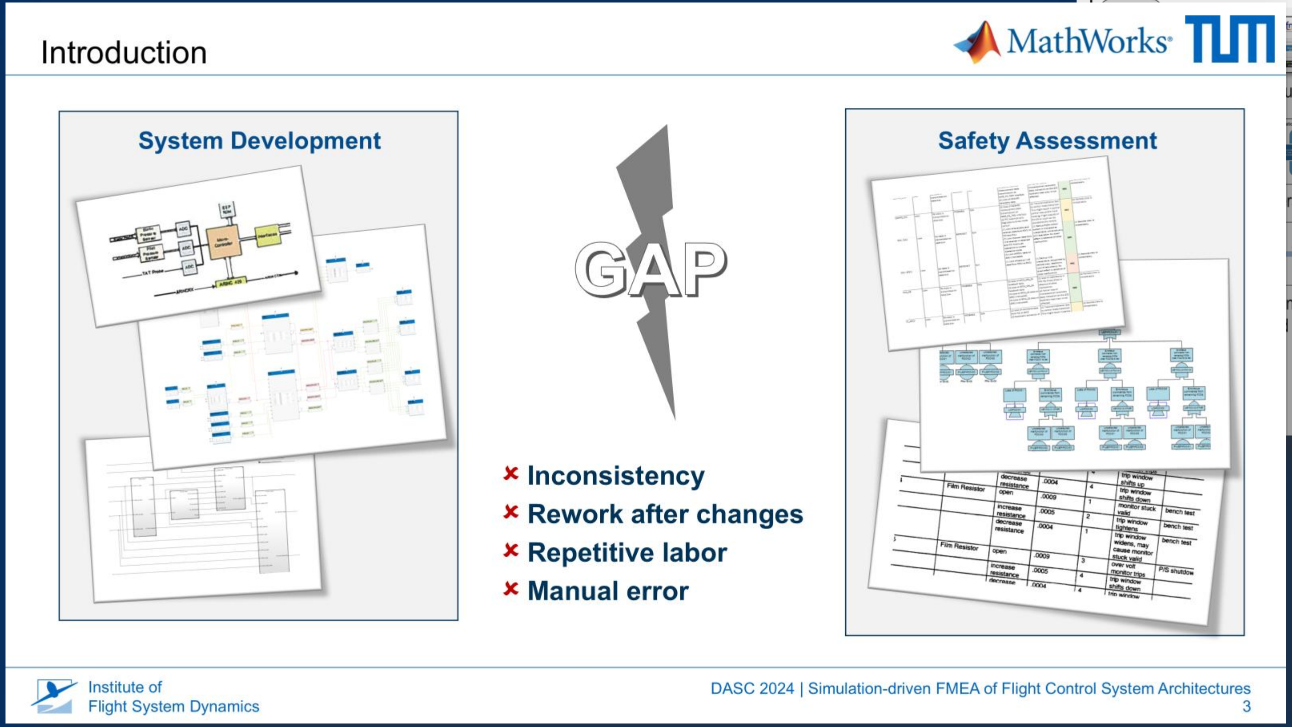


Image credit: INCOSE MBSE Workshop, Feb 2010

What’s the future of systems engineering?

It’s not just about systems engineering. It’s about the entire development cycle.





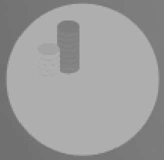
What’s the future of systems engineering?

▶ The future of systems engineering is model-based, leveraging next generation modeling, simulation, and visualization environments powered by the global digital transformation, to specify, analyze, design, and verify systems. High fidelity models, advanced visualization, and highly integrated, multi-disciplinary simulations will allow systems engineers to evaluate and assess an order of magnitude more alternative designs more quickly and thoroughly than can be done on a single design today.

▶ Artificial Intelligence, powered by large data sets and expert domain knowledge will drive major changes in systems engineering methods and tools, and within

▶ The future of systems engineering is model-based, leveraging next generation modeling, simulation, and visualization environments powered by the global digital transformation, to specify, analyze, design, and verify systems. High fidelity models, advanced visualization, and highly integrated, multi-disciplinary simulations will allow systems engineers to evaluate and assess an order of magnitude more alternative designs more quickly and thoroughly than can be done on a single design today.

els



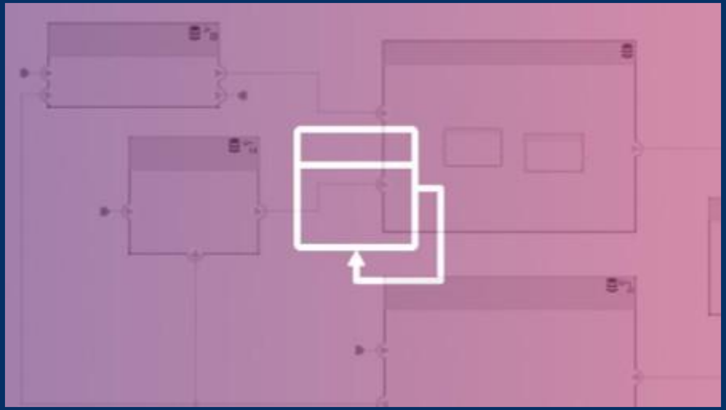
▶ The theoretical foundations for systems engineering will be based on established science and mathematics that characterize systems phenomena and stakeholder value, and provide the basis for systems education and evolving methods and tools.

▶ Ongoing education and training of systems engineers and the infusion of systems thinking across a broad range of the engineering and management workforce will meet the demands for a growing number of systems engineers with the necessary technical and leadership competencies.

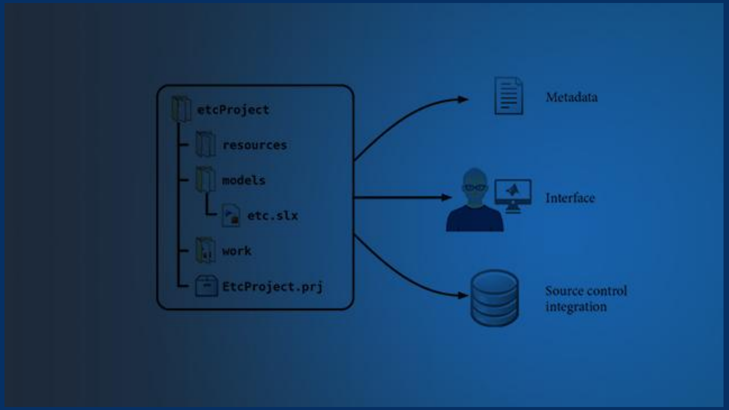
▶ Systems engineering will be embraced by a greater number and broader range of small and medium enterprises and will be continually adapted to manage systems complexity while also driving incremental market value.

[INCOSE Future of Systems Engineering Vision]

Systems Engineering Getting Started
Onramps & Trainings



System Composer Onramp



Simulink Model Management and Architecture

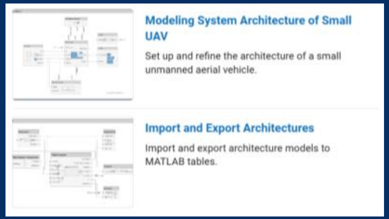
System Composer for Architecture Modeling

matlabacademy.mathworks.com

Resources

[Project Line Engineering: Bridging Between Product and Solution Business]
Presented by Jens Dietrich (Siemens Energy) at MATLAB EXPO Deutschland 2025

[System Composer]
Product overview page



[Get Started with System Composer]

[Model-Based Systems Engineering Enablement: Design, Analyze, and Test System and Software Architectures]

Model-Based Systems Engineering
Conclusions

- Bridge the Gap between system engineering and detailed design, test & implementation
- Reusable baseline for configure-to-order machine variants
- Early behavior validation before build, FAT, or commissioning
- Traceability from customer specs and regulatory requirements to controls implementation
- Smooth handoff from system design to control code generation and testing