

MATLAB[®]
& SIMULINK[®]

Digital Thread for Aerospace Design & Engineering

Satish Thokala

Aerospace and Defense Industry Manager



Outline

- Digital Threading Through Traditional Engineering
- Continuity Through Digital Engineering
- Examples and Case Studies
- Digital Thread in AI-ML Workflow

MathWorks is the leading provider of technical computing software

- **5 million users**
- Installations at **100,000+ sites** in **185 countries**
- Used for teaching and research by **6500 universities**
- **\$1.25B revenue** in 2021
- **6000 staff** including **2500 engineers**
- **Private, profitable every year** since founding in 1984



MATLAB, the language of engineers and scientists, is a programming environment for algorithm development, data analysis, visualization, and numeric computation.



Simulink is a block diagram environment for simulation and Model-Based Design of multidomain and embedded engineering systems.

Headquarters

Natick, MA USA

North America

United States

Europe

France
Germany
Ireland
Italy
Netherlands
Spain
Sweden
Switzerland
UK

Asia-Pacific

Australia
China
India
Japan
Korea



Customers in many industries innovate with MathWorks software



Aerospace and Defense

Complex multi-domain systems, software-defined and autonomous, model-based and data-driven



Automotive



Communications

Comms infrastructure, plus all types of connected systems across industries



Software and Internet

Big Data, Agile, DevOps, integration with IT systems



Financial Services



Railway Systems

Modernization, often on legacy platforms, becoming data-centric for optimization and maintenance



Energy Production



Electronics

Wide range of compute platforms, many kinds of HW/SW integration



Neuroscience

Collaboration between science, engineering, and informatics



Biological Sciences



Process Industries



Industrial Machinery



Semiconductors



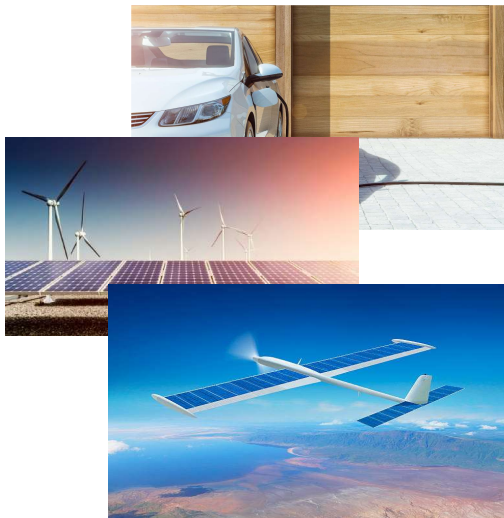
Medical Devices



Biotech and Pharmaceutical

Industries megatrends

Electrification



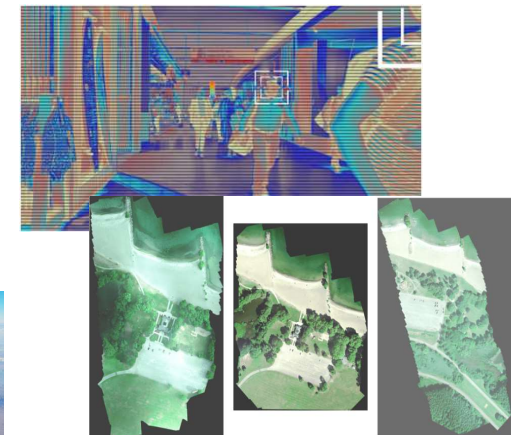
Connectivity



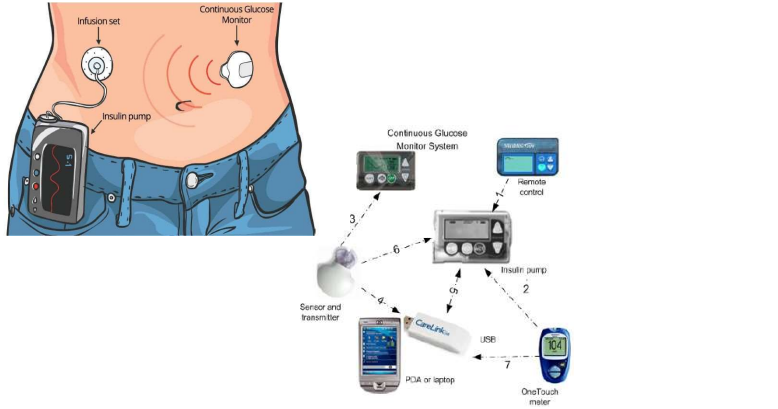
Autonomous



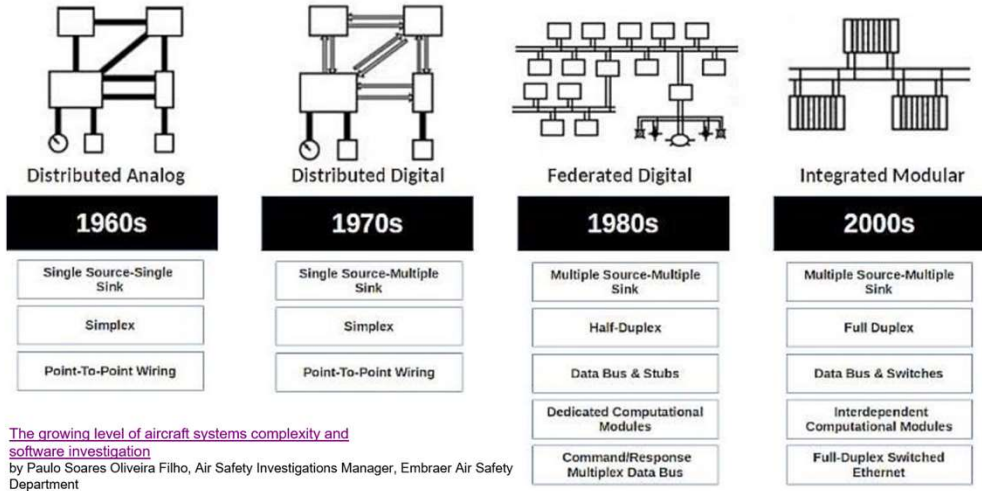
Artificial Intelligence



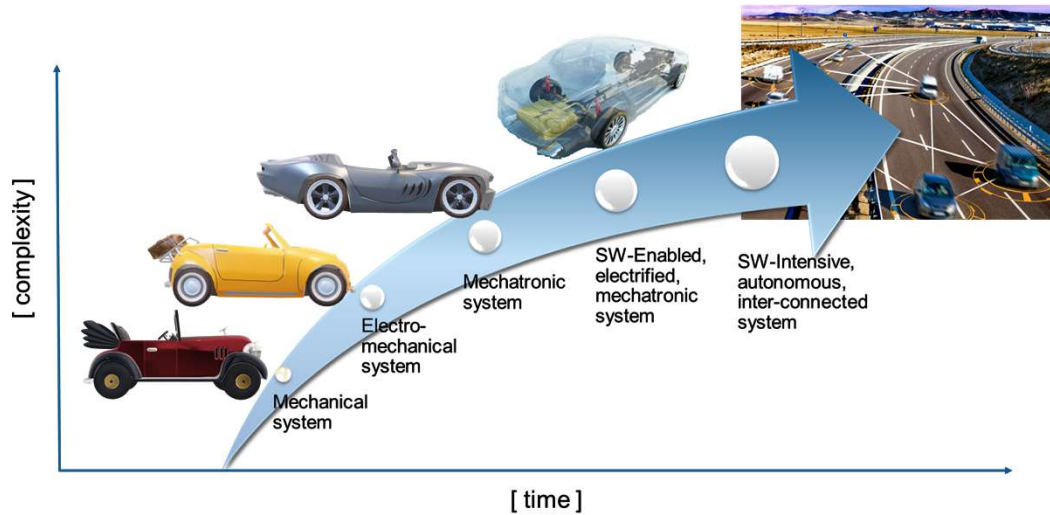
Industries challenges: Increasing systems complexity



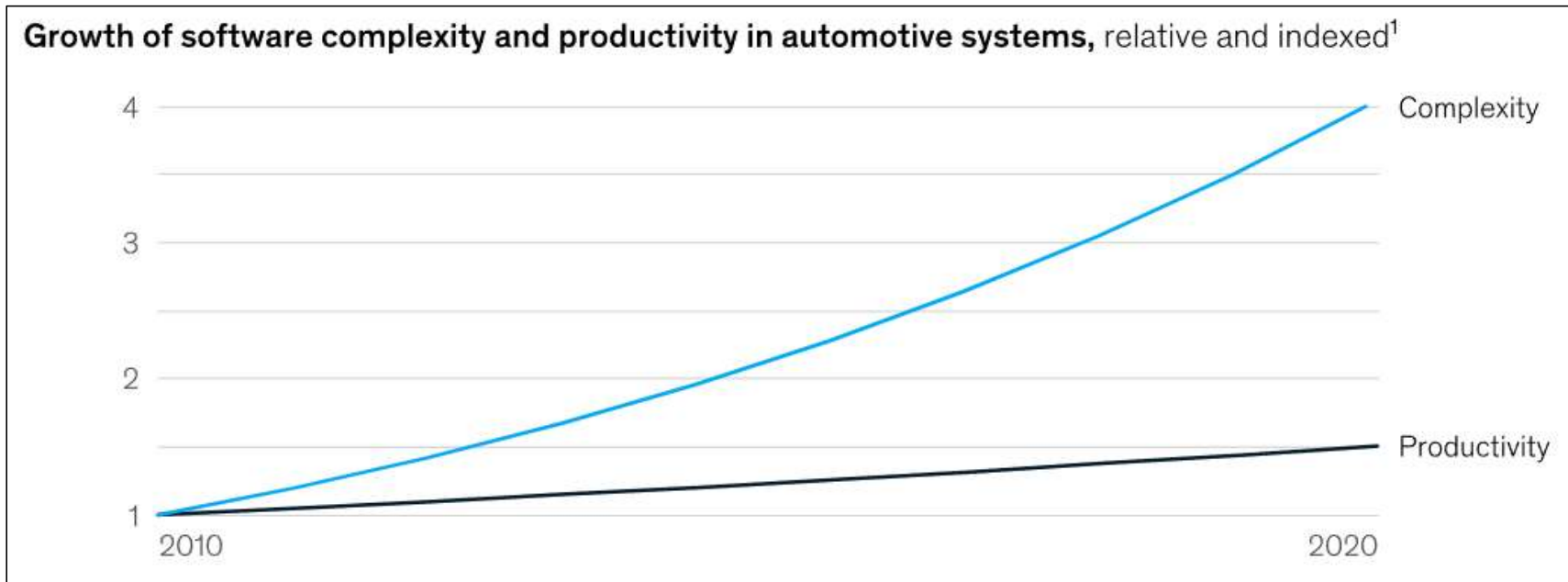
Growing level of integration in avionics architectures



The growing level of aircraft systems complexity and software investigation
 by Paulo Soares Oliveira Filho, Air Safety Investigations Manager, Embraer Air Safety Department



Falling into the complexity trap?

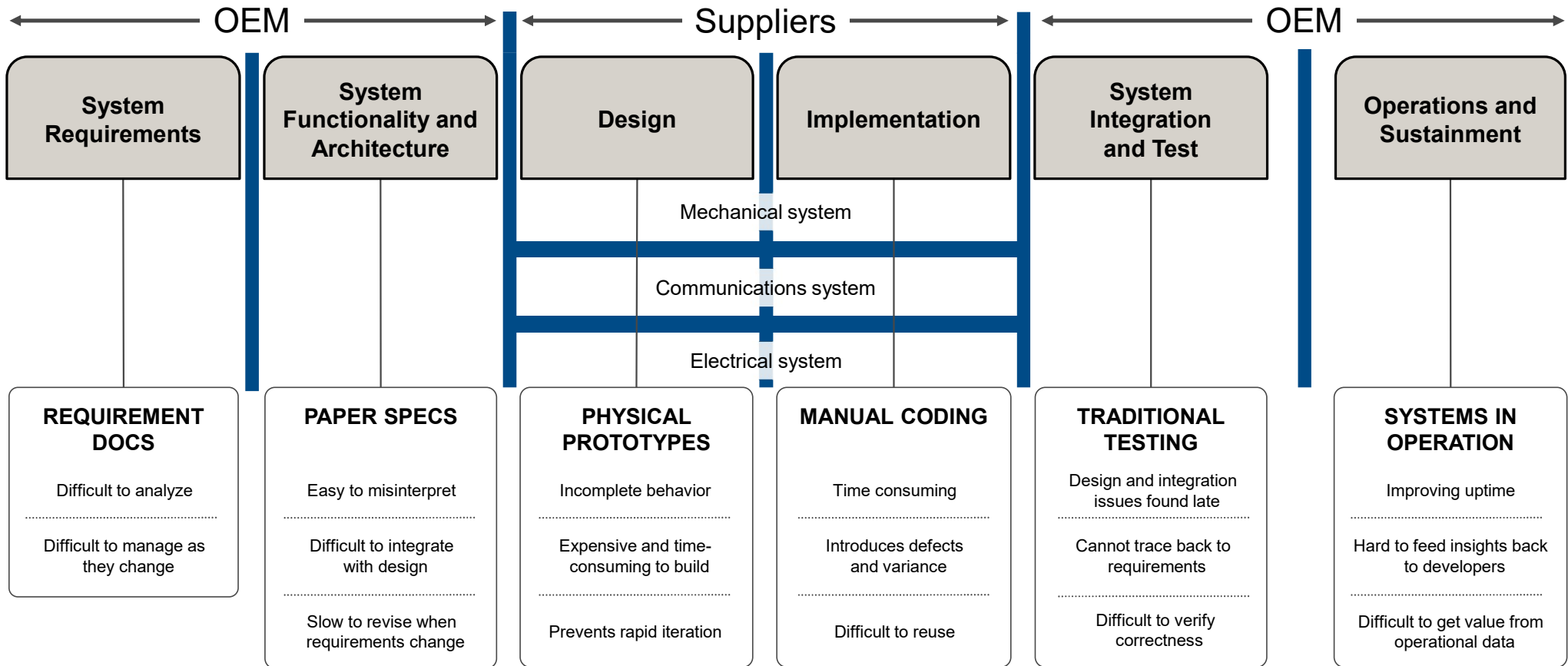


¹"When code is king: Mastering automotive software excellence," February 17, 2021, McKinsey.com.

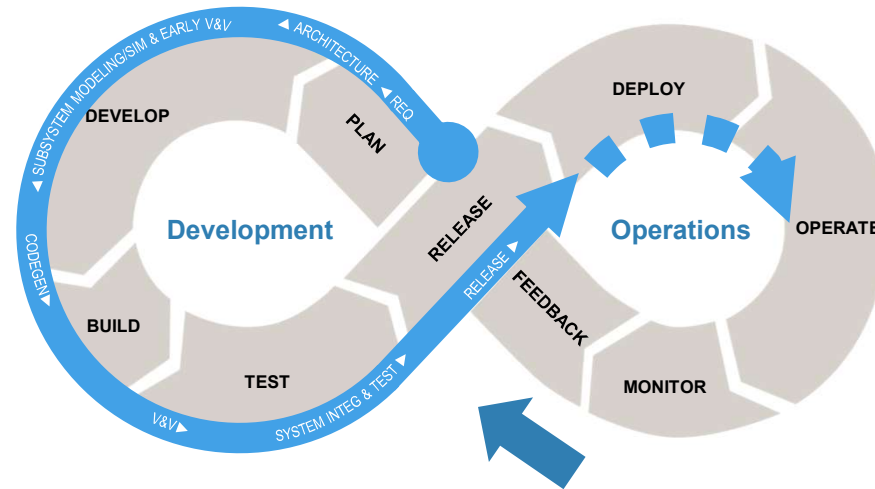
² Thousands of source lines of code.

Source: Paulo Soares Oliveira Filho, "The growing level of aircraft systems complexity and software investigation," International Society of Air Safety Investigators, 2020, isasi.org; McKinsey's SoftCoster embedded software project database

Workflow challenges



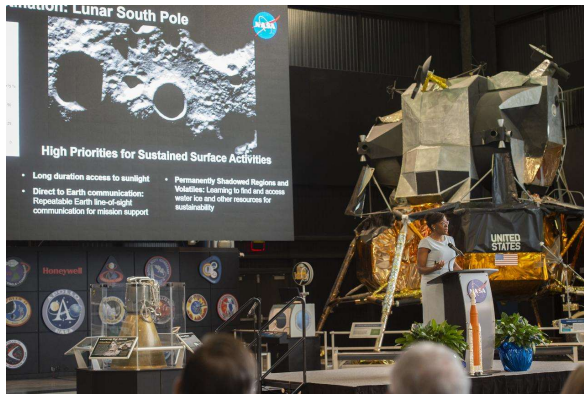
Rising demands to extend DevOps to systems, not only software



Companies are adopting various approaches to deal with these challenges

Challenge	Approach	Benefit
Communication of complex systems and requirements	SysML	Modeling
Change management	Digital Thread	Impact Analysis
Rapidly evolving systems	Agile & DevOps	Rapid delivery of value

NASA and ESA have objectives to advance digital engineering



Revision: Initial Release	Document No HLS-RQMT-001
RELEASE DATE: September 27, 2019	Page: 7 of 315
Title: HLS Requirements Document (SRD)	

2 Documents

For the purpose of this document, the term 'document' can also refer to 'digital artifacts,' 'models,' or 'viewpoints' as needed to convey and exchange configuration managed data or information. An objective of the HLS Program is to advance towards a digital engineering environment and away from the traditional document-based approach for capturing data, reports and baselines.

ESA Agenda 2025:

“ESA will therefore digitalise its full project management, enabling the development of digital twins, both for engineering by using Model Based System Engineering, and for procurement and finance, achieving full digital continuity with industry.”

Digital engineering in practice



Digital engineering in practice

System Requirements

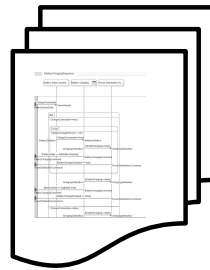
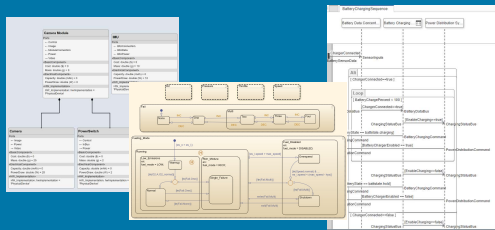
System Functionality and Architecture

Design

Implementation

Test and Verification

SysML

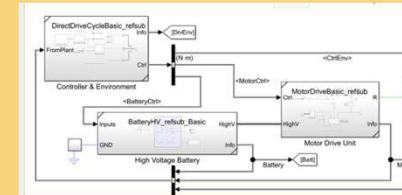


Model-Based Design

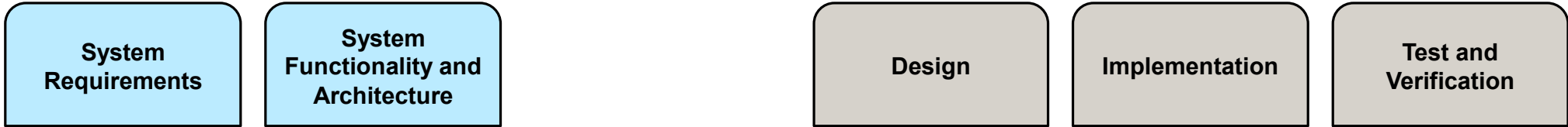
Software

Mechanical

Electronics



Digital engineering in practice



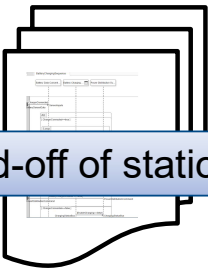
Inconsistent semantics of models breaks the digital thread and loses valuable engineering information

Inability to automate tasks such as requirements verification

Difficult to leverage simulation for systems analysis

SysML

Ad hoc hand-off of static documents



Model-Based Design

Software
Mechanical
Electronics

Difficult to trace authority between systems and design

Companies are adopting various approaches to deal with these challenges

Challenge	Approach	Benefit
Communication of complex systems and requirements	SysML	Modeling
Change management	Digital Thread	Traceability
Rapidly evolving systems	Agile & DevOps	Rapid delivery of value

Companies are adopting various approaches to deal with these challenges

Challenge	Approach	Benefit	Drawback
Communication of complex systems and requirements	SysML	Modeling	Static diagrams must be updated manually.
Change management	Digital Thread	Traceability	No automation of resolving differences.
Rapidly evolving systems	Agile & DevOps	Rapid delivery of value	Requires Human-in-the-loop May not leverage traceability or automation.



Companies are adopting various approaches to deal with these challenges

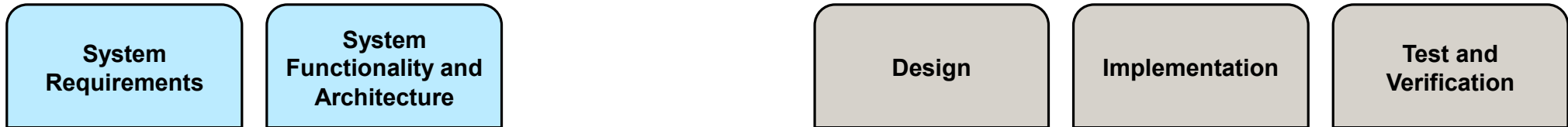
Common Modeling Semantics

Authoritative Source of Truth

Automation

Modeling · Automation · Authoritative Source of Truth

Building a digital engineering ecosystem



Common Modeling Semantics

Authoritative Source of Truth

Automation

Modeling • Automation • Authoritative Source of Truth

Building a digital engineering ecosystem

Common Modeling Semantics

Authoritative Source of Truth

Automation

Digital Engineering Ecosystem

System
Requirements

System
Functionality and
Architecture

Design

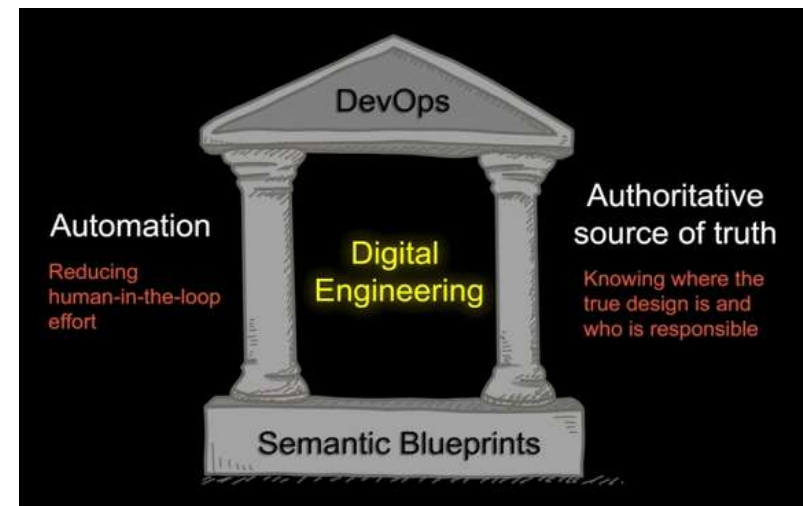
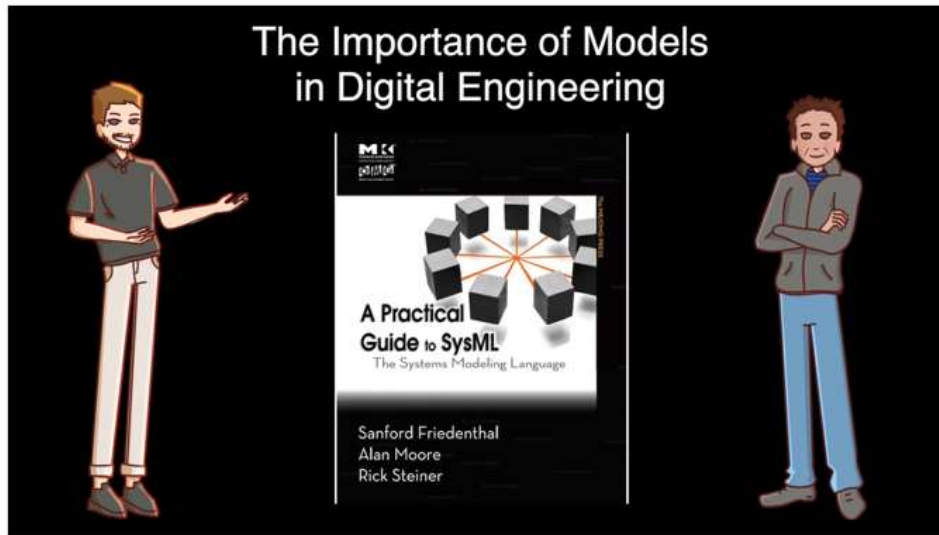
Implementation

Test and
Verification

MATLAB EXPO

Why Models Are Essential to Digital Engineering

Digital engineering is a trending industry buzzword. It's something that organizations strive to embrace and tool vendors claim to implement. But what is the practical reality behind the buzz? What are some of the essential aspects of an engineering ecosystem that actually provide the value promised? In this talk, Brian Douglas of Control Systems Lectures and MATLAB® Tech Talks, and Alan Moore, one of the original authors of SysML and co-author of "A Practical Guide to SysML," discuss exactly these questions and show how models are a central and essential element of digital engineering.



<https://www.mathworks.com/videos/why-models-are-essential-to-digital-engineering-1652969543566.html>

Modeling · Automation · Authoritative Source of Truth

Building a digital engineering ecosystem

Common Modeling Semantics

Authoritative Source of Truth

Automation

Digital Engineering Ecosystem

System
Requirements

System
Functionality and
Architecture

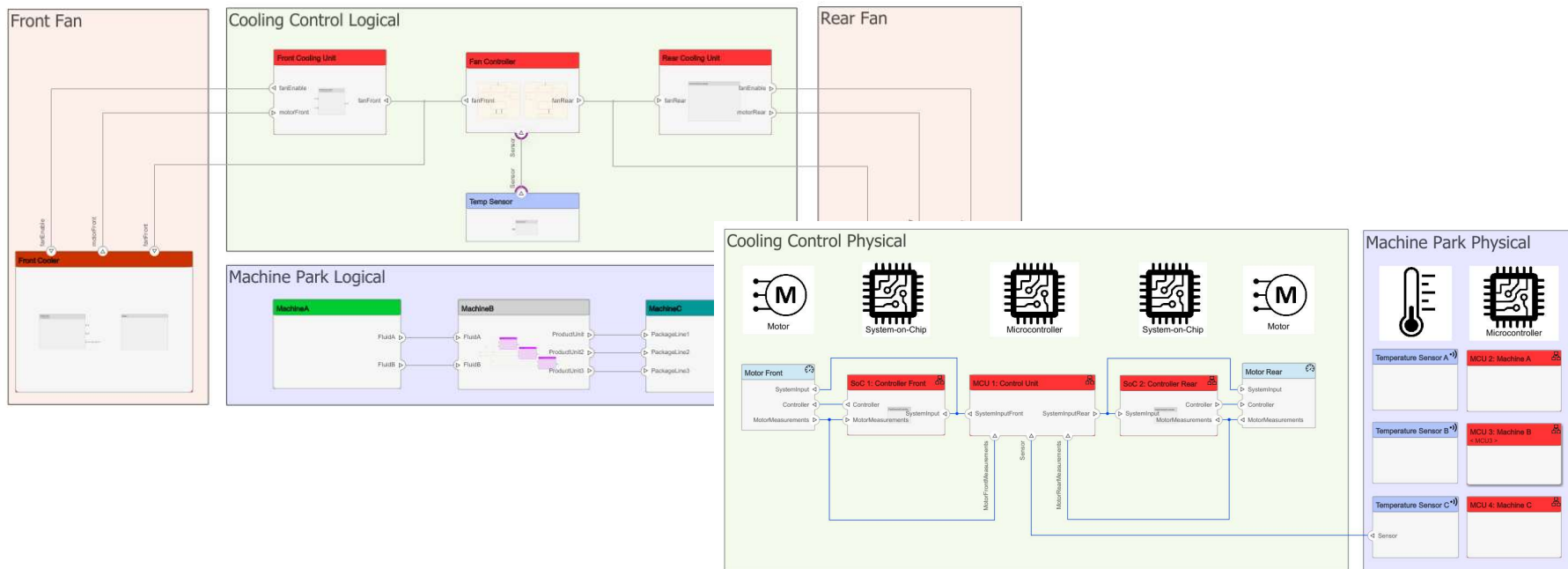
Design

Implementation

Test and
Verification

Modeling

Modeling semantics rich enough for descriptive modeling...



Digital Engineering Ecosystem

System Requirements

System Functionality and Architecture

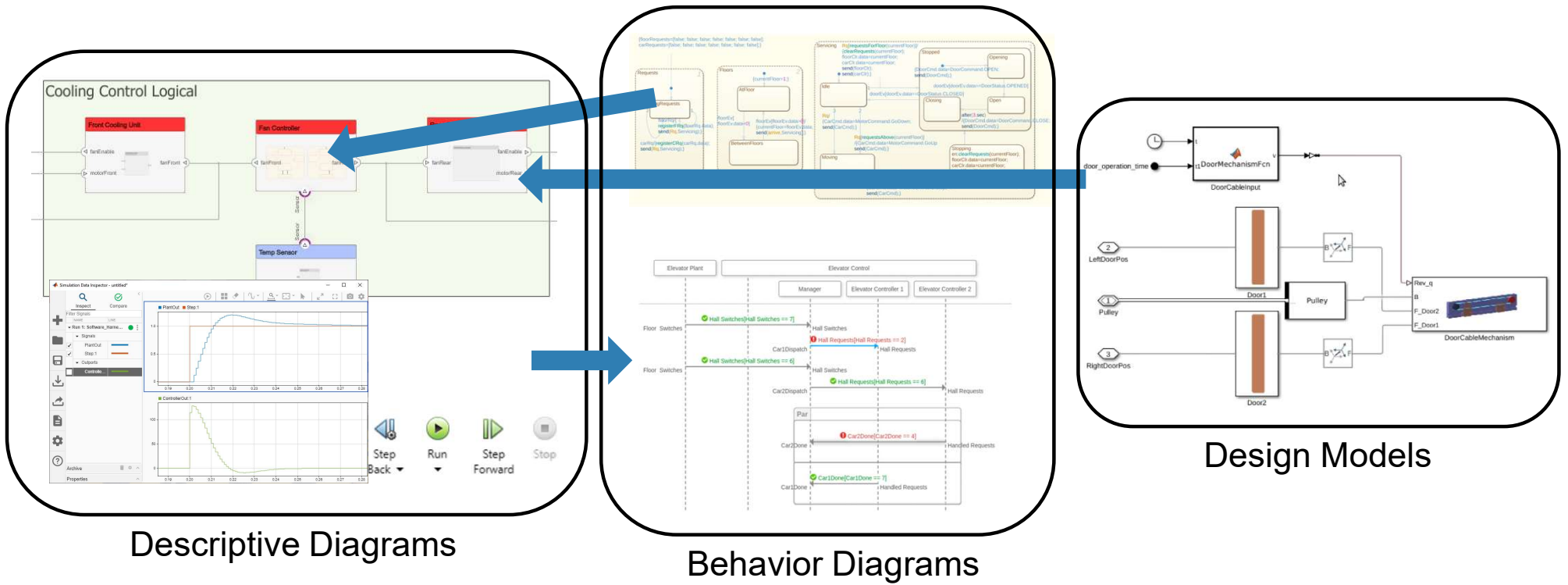
Design

Implementation

Test and Verification

Modeling

...and precise enough for detailed design, simulation and analysis.



Descriptive Diagrams

Behavior Diagrams

Design Models

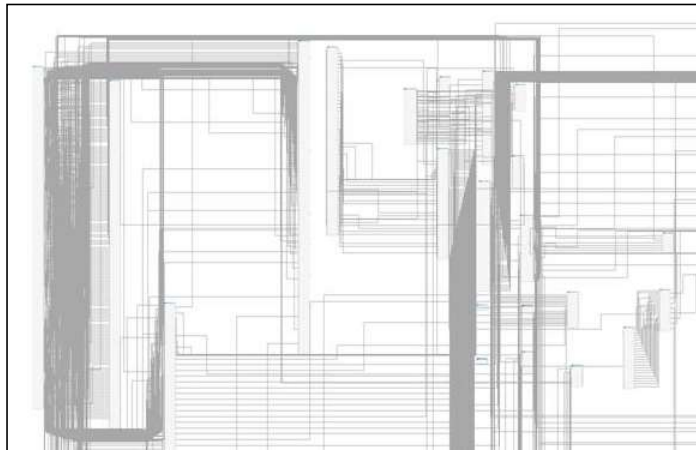
Digital Engineering Ecosystem



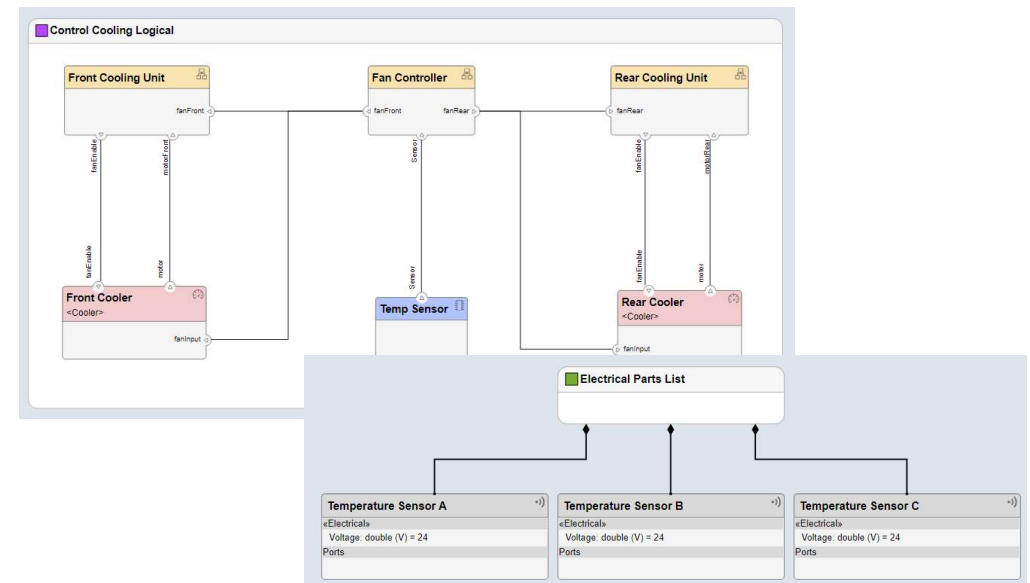
Modeling

Modeling semantics rich enough for descriptive modeling...

Full System Model



Filtered Views



Digital Engineering Ecosystem

System Requirements

System Functionality and Architecture

Design

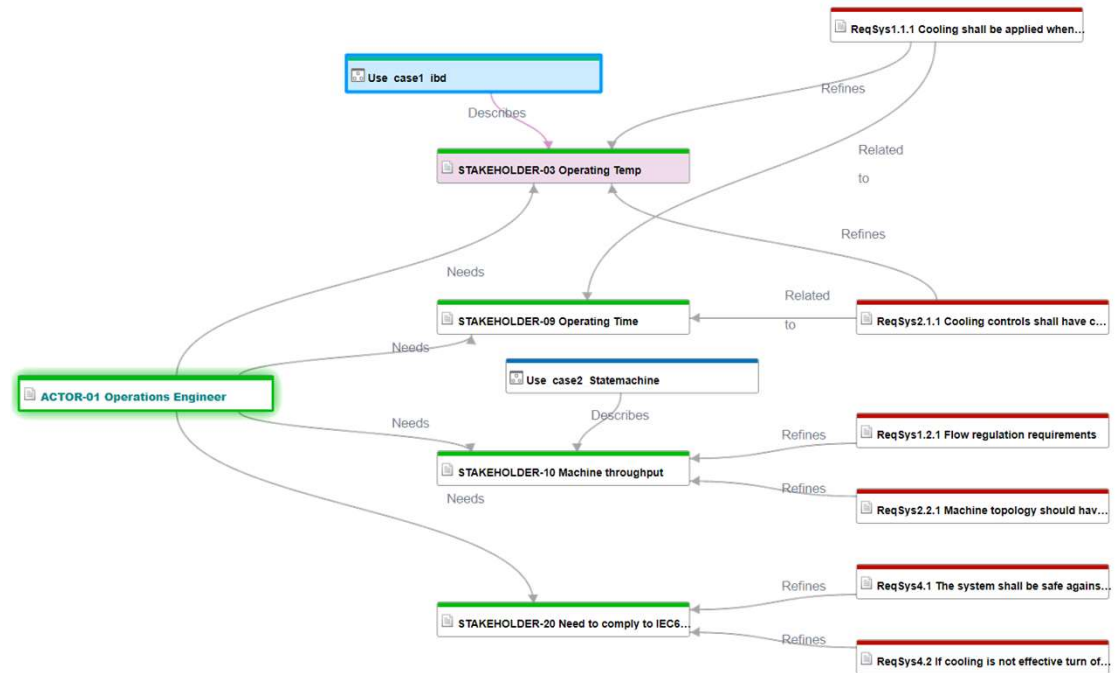
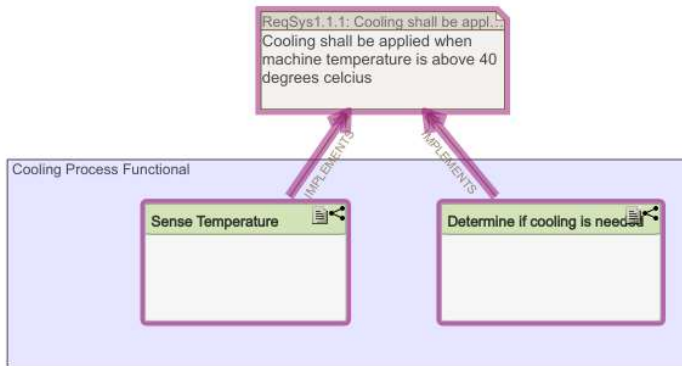
Implementation

Test and Verification

Authoritative Source of Truth

Full traceability of requirements, architectures, and design

The screenshot shows the Requirements Editor interface. On the left, a tree view lists various requirements such as 'Power System', 'Power Source', 'Mass Properties & Geometry', 'Total Volume', 'Total Mass', 'General Control Station', 'Quadrupole', 'Quadrupole Load', 'Quadrupole Inductance', 'Quadrupole Latch', 'Payload', 'Target Characteristics', 'Target Movement', 'Target Environment', 'Target Intensity', 'Target Size', 'Target Color', 'Target Material', 'System Requirements', 'Mission Duration', and 'System Size'. The right pane shows the 'Properties' for a selected requirement, including a description, a diagram of a 'Drone with receiving coil and battery charger' showing components like 'U-controller', 'Clock generation', 'ADC & Envelop Detection', 'MIB', 'Power source', 'Driver for excitation coils', and 'Voltage measurements (U, I, R)', and a table with columns for 'Implemented' and 'Verified'.

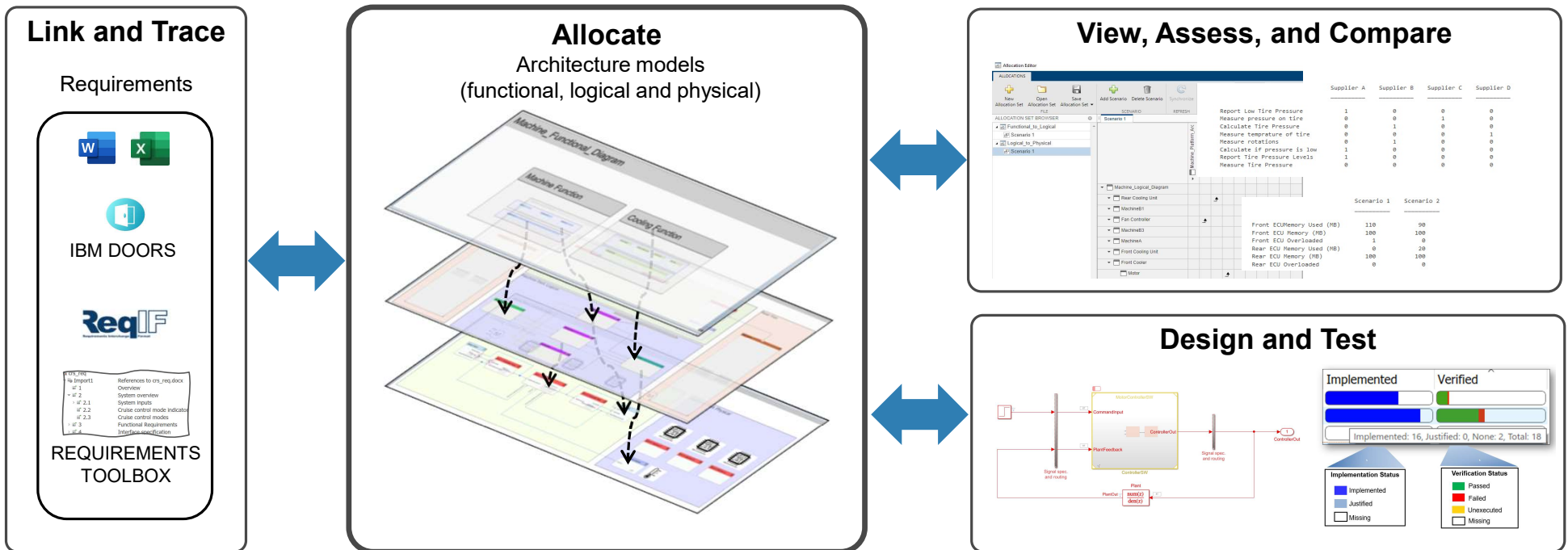


Digital Engineering Ecosystem



Authoritative Source of Truth

Automate analysis and assessments of linked artifacts

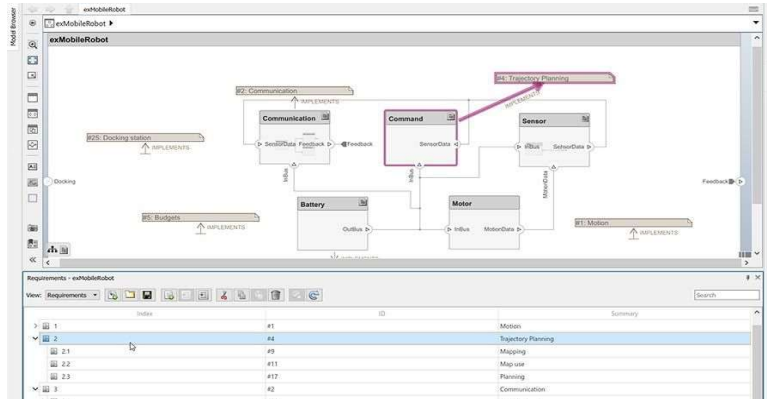


Digital Engineering Ecosystem



UAV design using digital engineering

Digital Thread



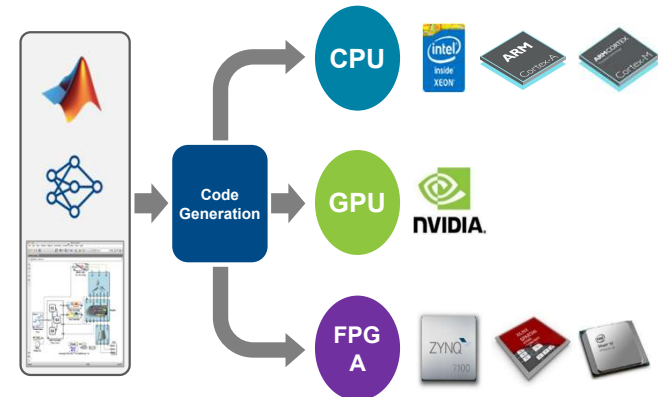
Real-time testing on hardware



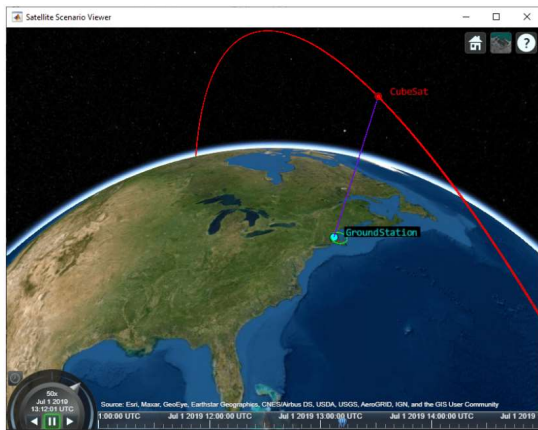
3D visualization



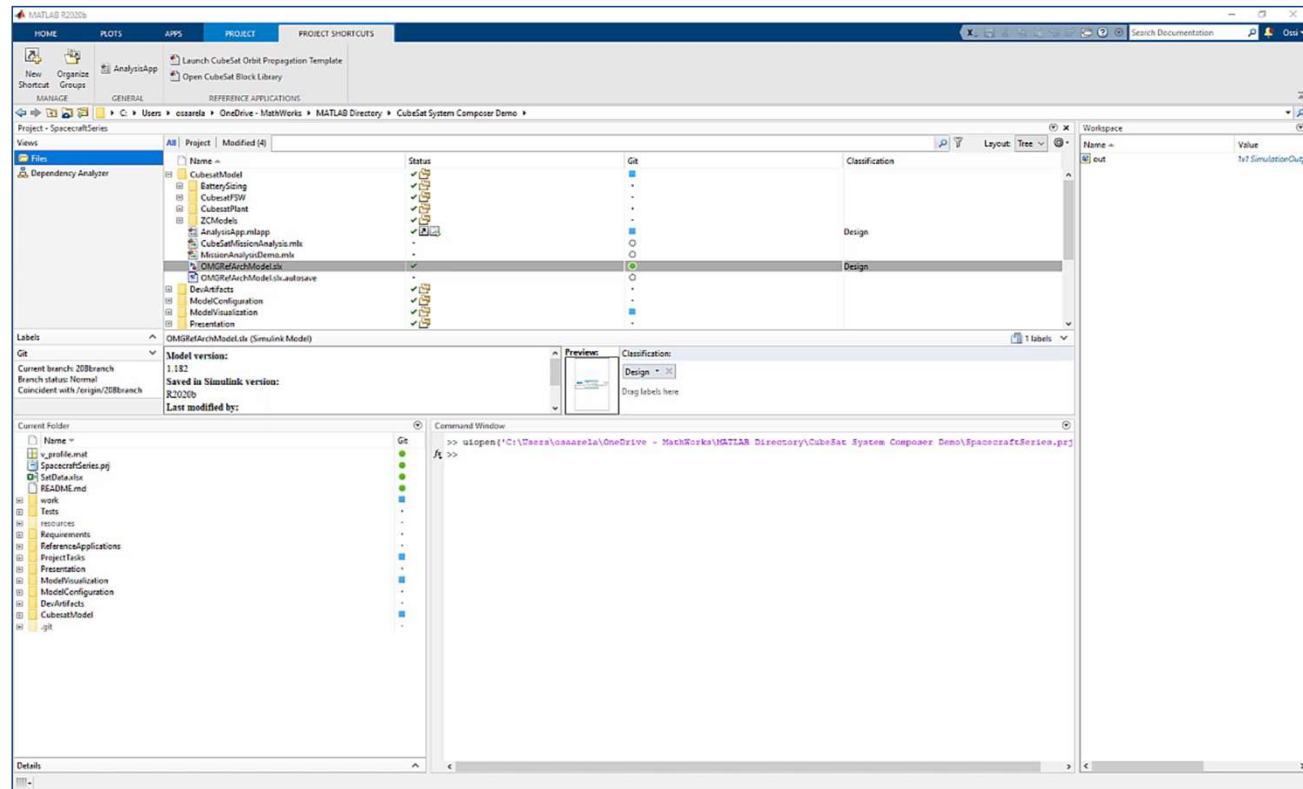
Deployment



Digital engineering for space systems

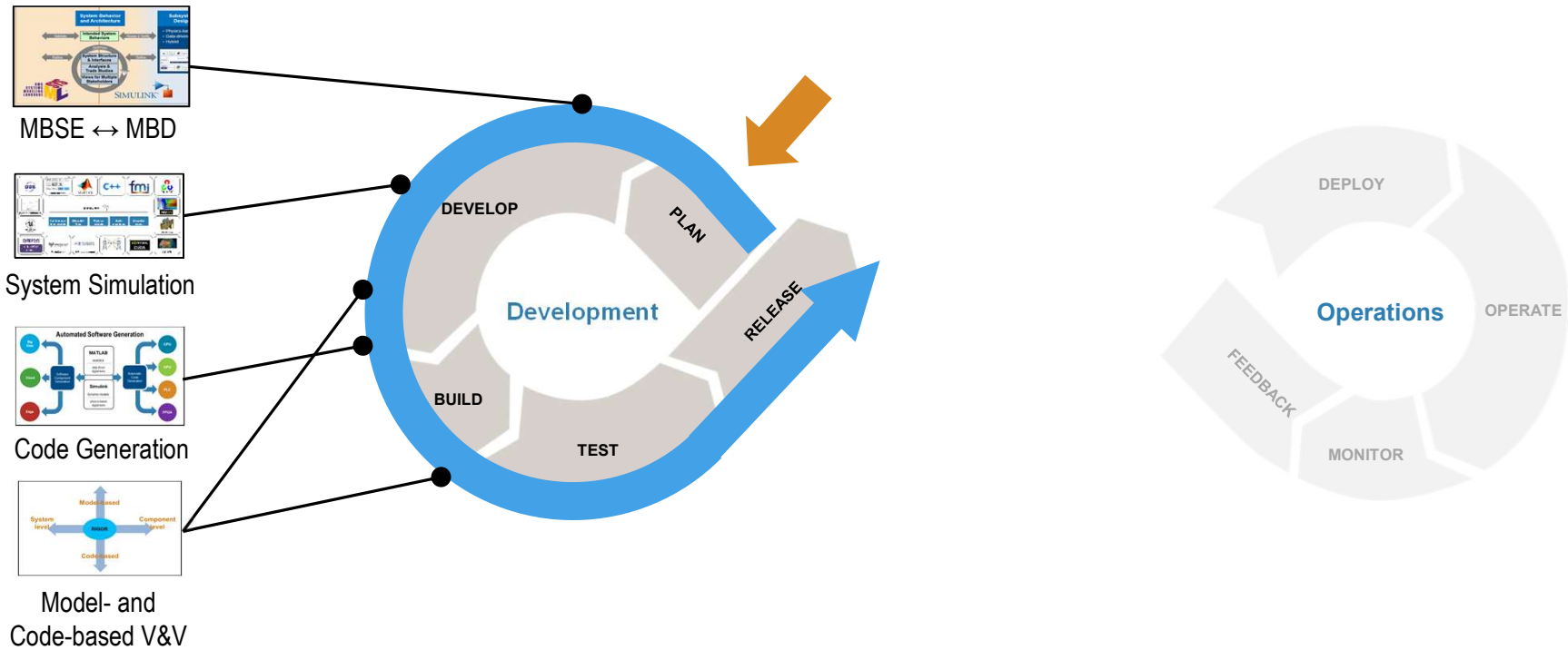


The system shall provide and store visual imagery of MathWorks headquarters [42.2775 N, 71.2468 W] 1 time daily at 10 meters resolution.



Automation

Integrate models at every level into CI pipeline



Digital Engineering Ecosystem

System Requirements

System Functionality and Architecture

Design

Implementation

Test and Verification

Gulfstream: Electronic System Architecture Modeling using Digital Engineering

"System Composer adds additional capabilities for modeling integration between systems, ...capturing important system and component properties, ...directly connecting system architecture models to software functional models, and flowing data down into specialized design tools."

System Architecture Modeling for Electronic Systems Using MathWorks System Composer and Simulink

Christopher B. Watkins
Gulfstream Aerospace Corporation
Savannah, GA, U.S.
chris.watkins@gulfstream.com

Jerry Varghese
Gulfstream Aerospace Corporation
Savannah, GA, U.S.
jerry.varghese@gulfstream.com

Michael Knight
Gulfstream Aerospace Corporation
Savannah, GA, U.S.
michael.knight@gulfstream.com

Becky Petreys
The MathWorks, Inc.
Natick, Massachusetts, U.S.
bpetreys@mathworks.com

Jordan Ross
The MathWorks, Inc.
Natick, Massachusetts, U.S.
jordanr@mathworks.com

Abstract—Electronic system architectures have traditionally been documented as static block diagrams in tools such as Microsoft[®] Visio[®] or through a richer modeling approach such as System Modeling Language (SysML). These approaches did not fully meet the modeling needs for the Gulfstream authors, which led to an alternative approach.

This paper introduces the Electronic System Architecture Modeling (eSAM) method, which leverages a new system architecture modeling tool called System Composer[™]. eSAM was created by the authors to define a standard method for applying the generic System Composer modeling constructs to build functional, physical, and logical architecture models of electronic systems. The eSAM methods are applied to an example avionics architecture to demonstrate capabilities needed for system modeling, collaborative OEM-supplier workflows, data management and ICD generation, systems integration activities, generation of system architecture deliverables for the avionics certification standards governed by SAE ARP4754A, and a Model-Based Design approach that connects a software function to its system-level ICD.

System Composer is built on MATLAB[®] and Simulink[®] and leverages the modeling, analysis, and simulation capabilities of these well-established tools. System Composer adds additional capabilities for modeling integration between systems, filtering large models into manageable views, capturing important system and component properties, allocating between different descriptive architecture models, directly connecting system architecture models to software functional models, and flowing data down into specialized design tools.

This paper summarizes desirable features in system architecture modeling tool, introduces the features and concepts of System Composer and describes application of the eSAM method.

Keywords—system architecture, modeling, Model-Based Design, MBD, MATLAB, Simulink, System Composer, eSAM

I. INTRODUCTION

A traditional systems development workflow starts with early concepts and requirements and flows down to the implementation in hardware and software. A common view of this process is the V-diagram where the design and validation

Data-Message Modeling for Multi-Lane Architectures on an IMA Platform Using the eSAM Method

Michael Knight
Gulfstream Aerospace Corporation
Savannah, GA, U.S.
michael.knight@gulfstream.com

Becky Petreys
The MathWorks, Inc.
Natick, Massachusetts, U.S.
bpetreys@mathworks.com

When applying design tools to support this process are two gaps of interest. Gap #1 exists between specification and the implementation of the system; engineers must sufficiently describe the behavior and of the system such that engineers can accurately implement system. Graphical modeling tools, such as Simulink, which allows them to validate that the behavior of the system. For software systems they can automatically generate code for the algorithm that is used in production software. Using this Model-Based Design approach, the design engineers are able to work at a level of abstraction in the graphical environment implementation engineers are able to elaborate models when working the implementation details. These tools help to bridge this gap between engineering disciplines by eliminating unnecessary rework and encouraging effective communication in a common tool environment.

The other significant gap, Gap #2, in the development workflow occurs earlier in the process, when moving from early concepts to design. The tool requirements for each stage

Figure 1: Simplified development process

are performed along the left-hand side and test and validation are performed along the right-hand side. Figure 1 is a simplified view of the V-diagram and summarizes the design activities.

When applying design tools to support this process are two gaps of interest. Gap #1 exists between specification and the implementation of the system; engineers must sufficiently describe the behavior and of the system such that engineers can accurately implement system. Graphical modeling tools, such as Simulink, which allows them to validate that the behavior of the system. For software systems they can automatically generate code for the algorithm that is used in production software. Using this Model-Based Design approach, the design engineers are able to work at a level of abstraction in the graphical environment implementation engineers are able to elaborate models when working the implementation details. These tools help to bridge this gap between engineering disciplines by eliminating unnecessary rework and encouraging effective communication in a common tool environment.

The other significant gap, Gap #2, in the development workflow occurs earlier in the process, when moving from early concepts to design. The tool requirements for each stage

processor (HAP) modules host software applications from multiple suppliers. The transition from federated to IMA architectures is discussed in prior work [1]. IMA architectures provide significant integration flexibility, allowing systems with different bus protocols to communicate, while providing data integration services to overcome discrepancies between data sets (e.g., conversions from Celsius to Fahrenheit). IMA systems any connected system to subscribe to any data on the network. Therefore, IMA system integration is achieved via data integration instead of physical wiring.

IMA brings the benefit of integration flexibility but also reduces integration complexity. Modeling methods have not been effective in representing the integration of systems on an IMA platform in an easily understandable format [2]. To address this need, the authors have developed a novel modeling method known as the Electronic System Architecture Modeling (eSAM) method. Gulfstream teamed with MathWorks to implement this eSAM method in the System Composer tool (R2022a) [3]. The eSAM method is described in previous work [4]. This paper provides a detailed description of how data exchanges are modeled, which is fundamental to this novel method in IMA systems allocated to an IMA platform.

MODELING CHALLENGES FOR IMA ARCHITECTURES

Federated vs. IMA Architectures

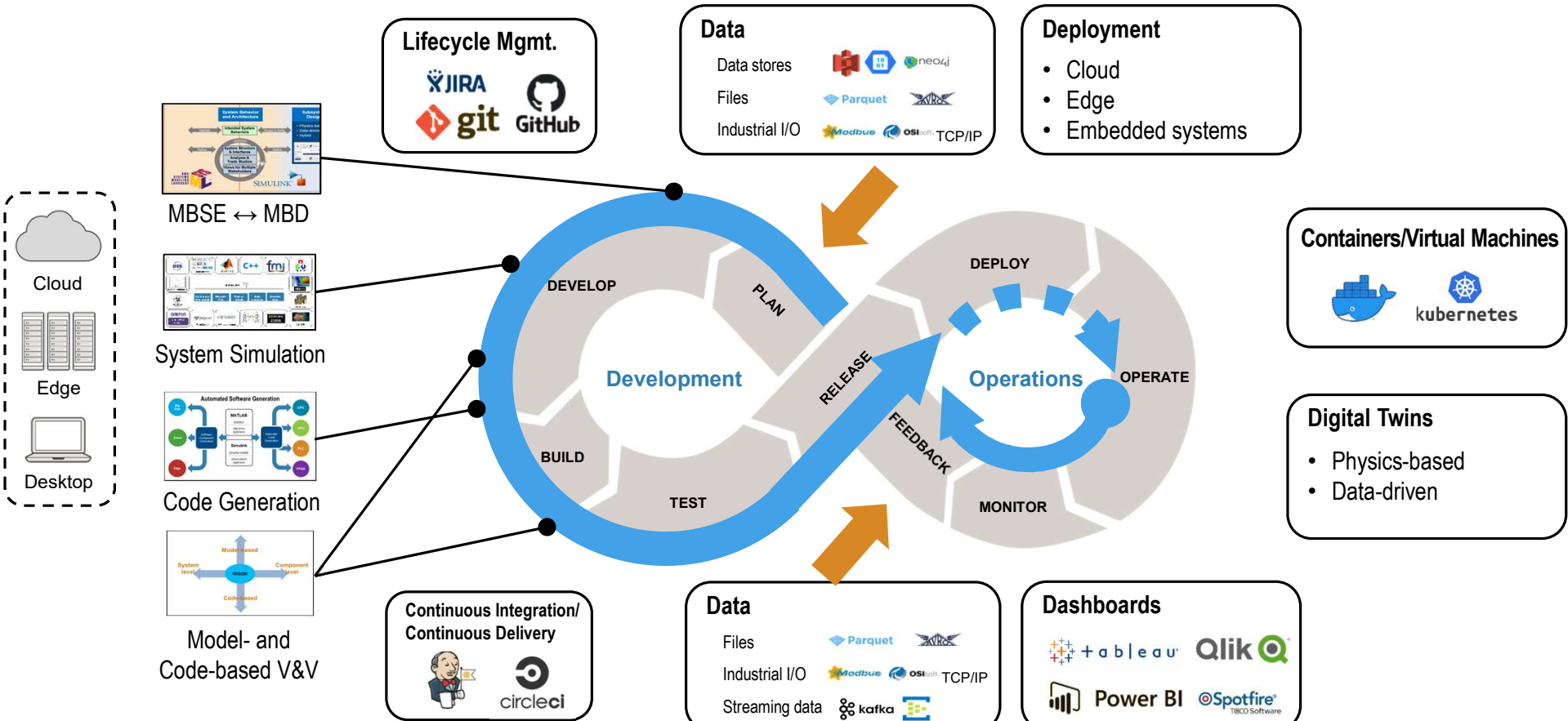
A federated system consists of multiple components that are integrated to perform a set of system functions. In a simplified example of a federated architecture, the Radio Altimeter provides an ARINC 429 (A429) digital bus input to the AutoPilot which uses an analog voltage to command the Throttle position. The system architect can perform this integration using point-to-point wiring as shown in Fig. 1.

Fig. 1. Traditional Point-to-Point System Architecture

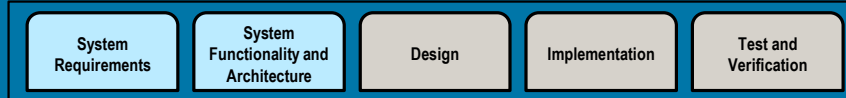
<https://ieeexplore.ieee.org/document/9256753>

<https://ieeexplore.ieee.org/document/9925816>

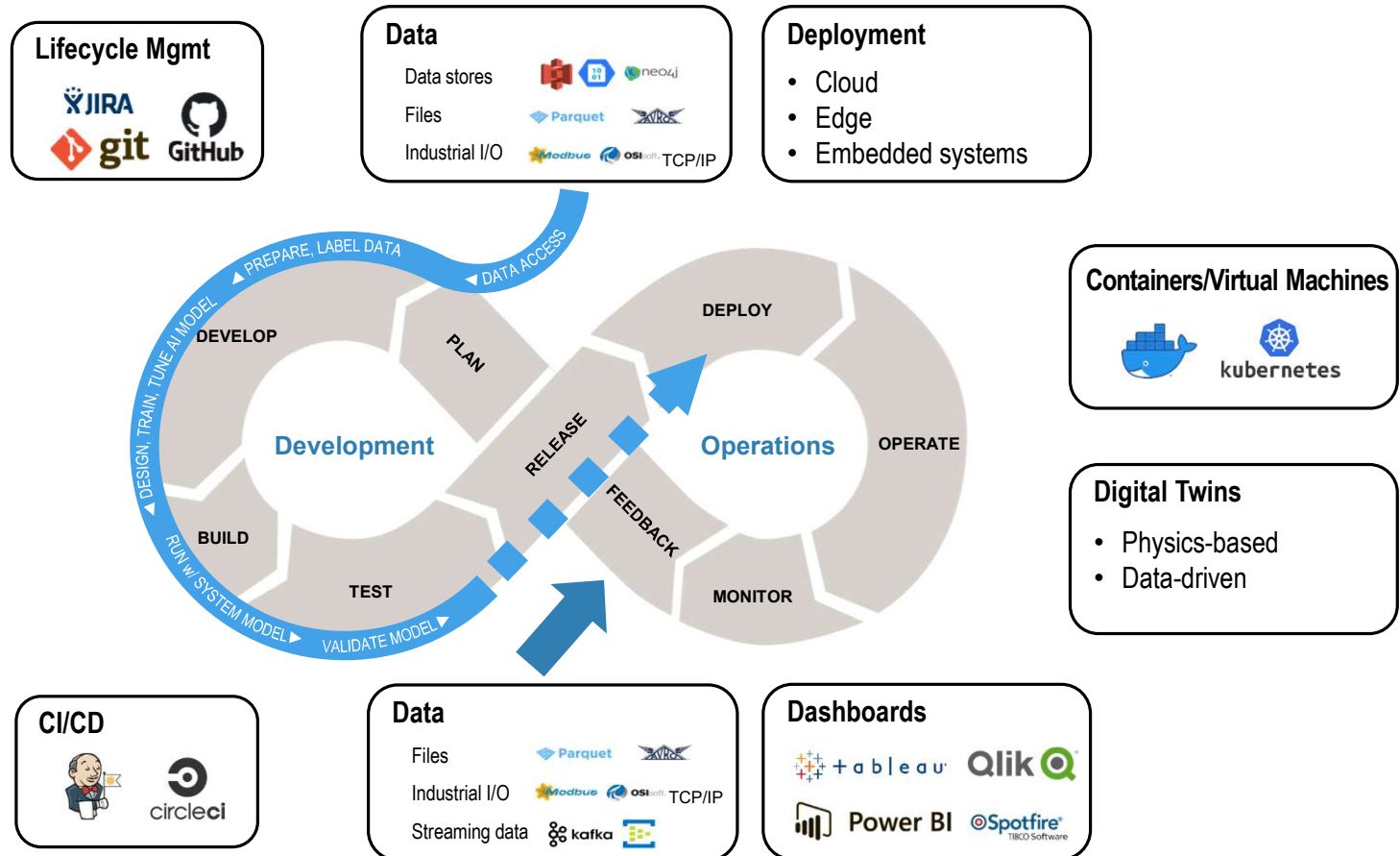
Software and systems – Agile to DevOps



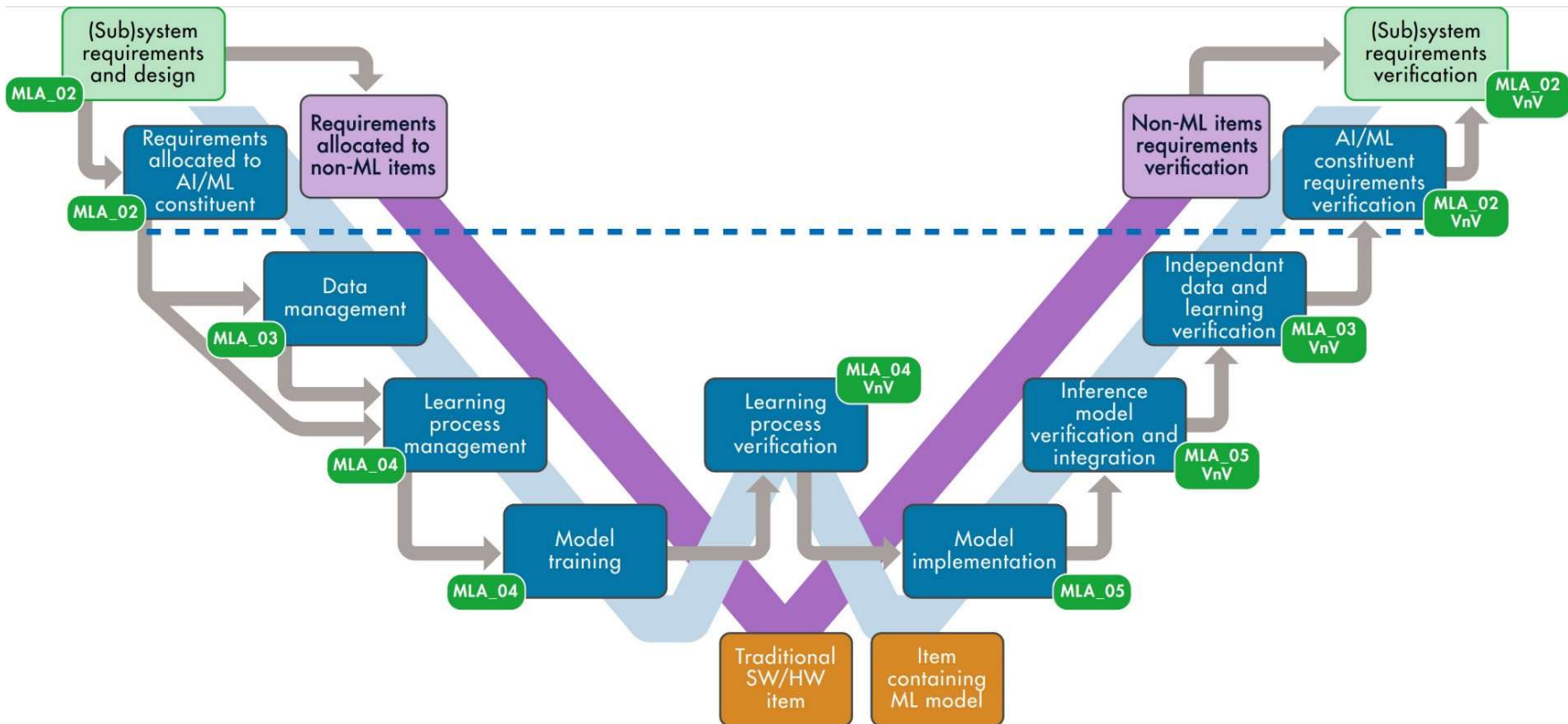
Digital Engineering Ecosystem



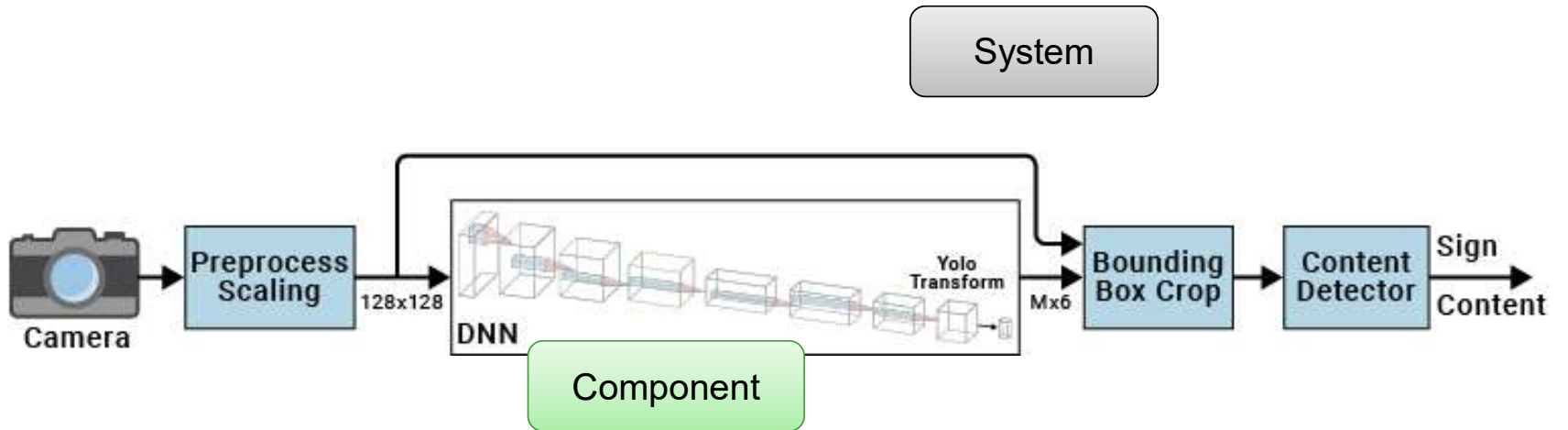
Machine Learning workflow follows the same cycle



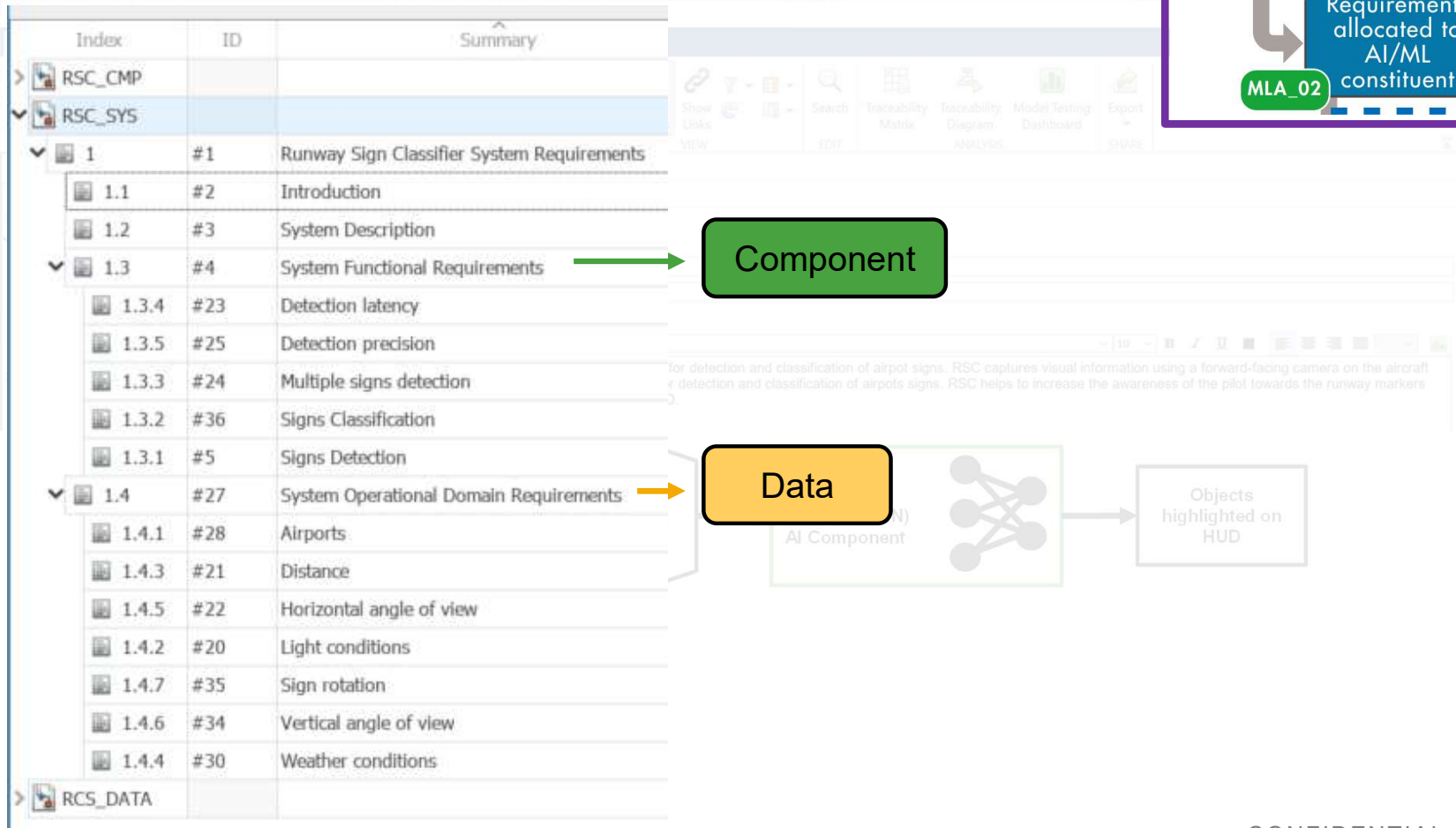
Digital Thread in an Airborne Deep Learning System



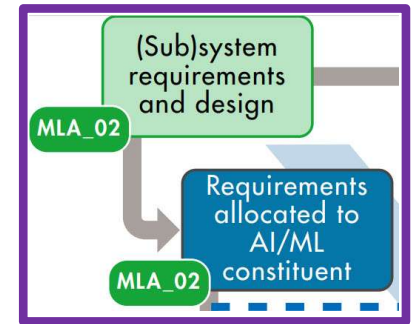
Case Study: A Visual Sign Recognition System



Defining system requirements and allocate to the AI constituent



Link system requirements to data requirements



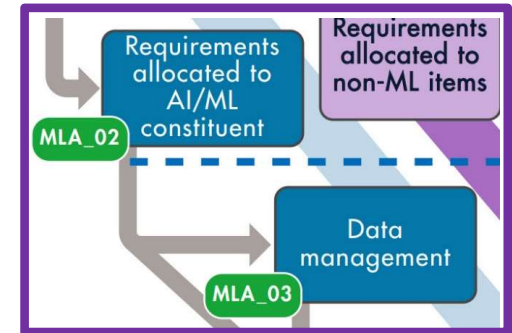
The screenshot displays a requirements management tool interface. On the left, a tree view shows a hierarchy of requirements under "RSC_SYS" and "RCS_DATA". The "Weather conditions" requirement (Index 1.4.4, ID #30) is selected. The main pane shows its properties: Type: Functional, Index: 1.4.4, Custom ID: #30, Summary: Weather conditions. The description reads: "The RSC shall operate in all expected weather conditions when it is possible to see and identify signs within the operational distance range." The "Links" section shows "Implemented by:" with links to "FAIR weather condition", "RAIN weather condition", "SNOW weather condition", and "FOG weather condition", and "Related to:" with a link to "Airports".

Below this, another requirement is shown: "FAIR weather condition" (Index 1.2.4, ID #30). Its properties include: Type: Functional, Index: 1.2.4, Custom ID: FAIR, Summary: FAIR weather condition. The description states: "The dataset shall include the images captured in FAIR weather condition". The "Links" section shows "Implements:" with a link to "Weather conditions".

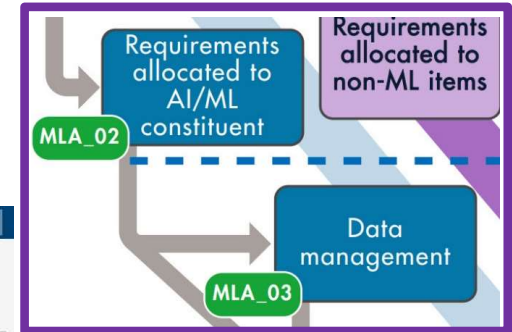
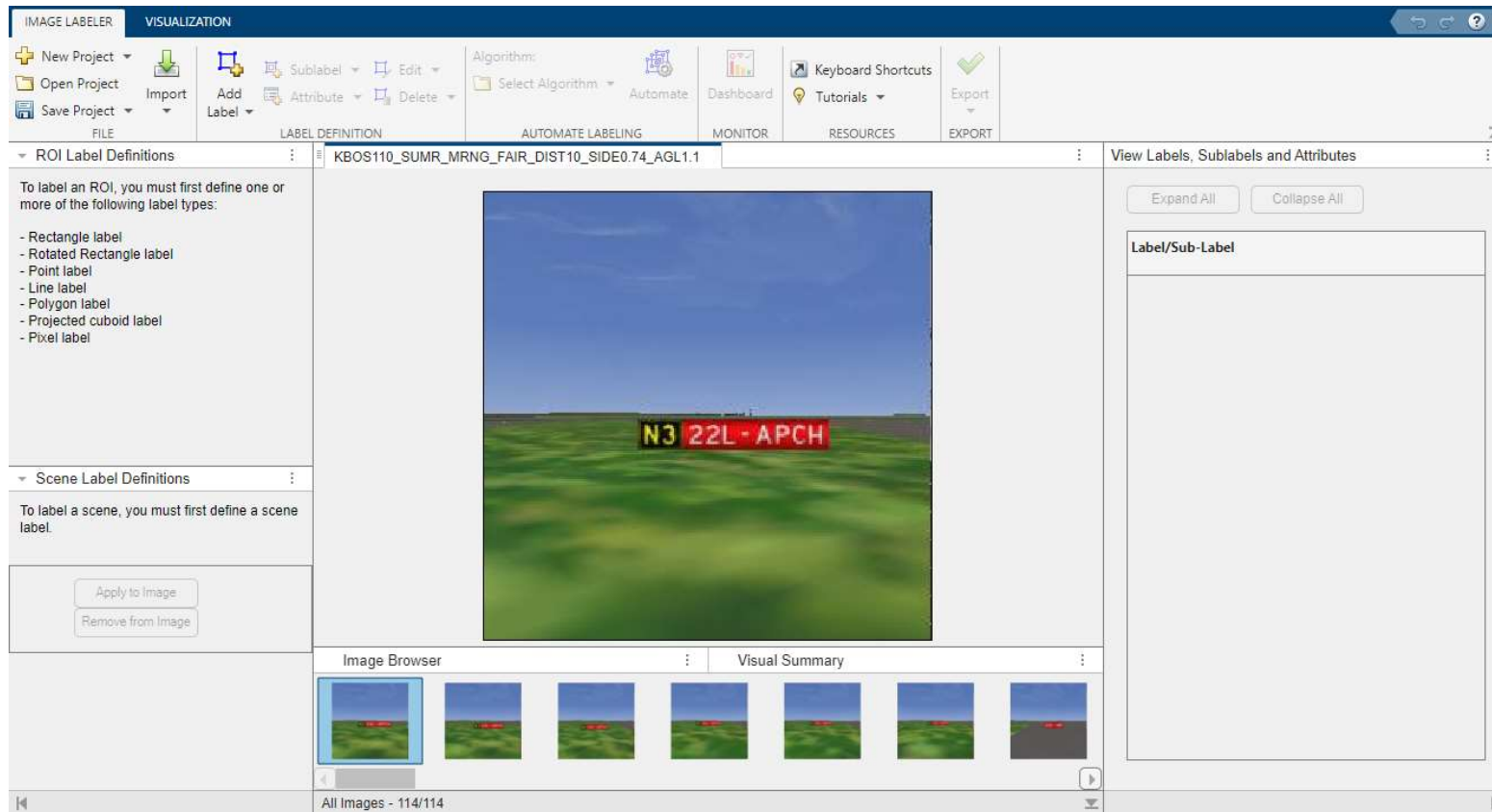
An orange arrow points from the "Implemented by:" link in the first requirement's properties to the "FAIR weather condition" requirement in the tree view.

Map data requirements to the data

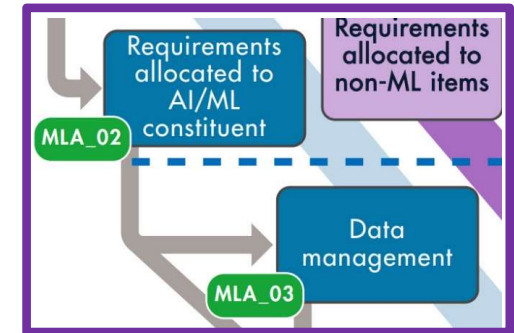
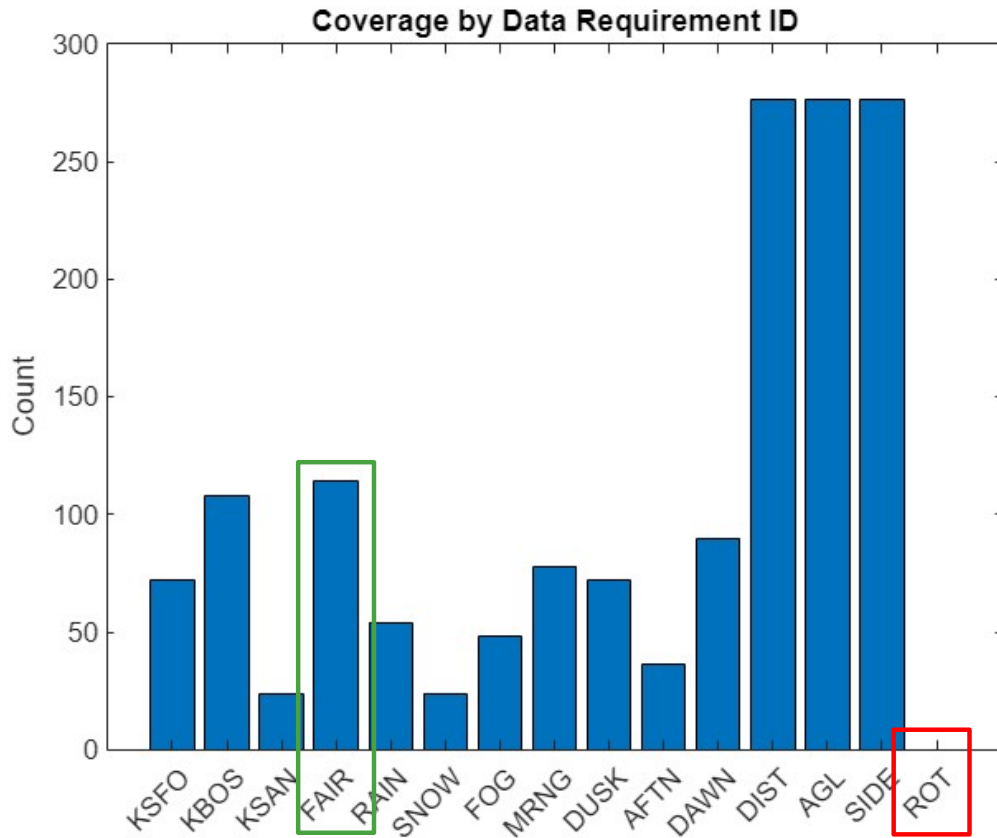
Data Requirement ID	Datastore	Datastore Size	Datastore Link
{'KSFO'}	{1x1 matlab.io.datastore.ImageDatastore}	72	{["Open Datastore"]}
{'KBOS'}	{1x1 matlab.io.datastore.ImageDatastore}	108	{["Open Datastore"]}
{'KSAN'}	{1x1 matlab.io.datastore.ImageDatastore}	24	{["Open Datastore"]}
{'FAIR'}	{1x1 matlab.io.datastore.ImageDatastore}	114	{["Open Datastore"]}
{'RAIN'}	{1x1 matlab.io.datastore.ImageDatastore}	54	{["Open Datastore"]}
{'SNOW'}	{1x1 matlab.io.datastore.ImageDatastore}	24	{["Open Datastore"]}
{'FOG' }	{1x1 matlab.io.datastore.ImageDatastore}	48	{["Open Datastore"]}
{'MRNG'}	{1x1 matlab.io.datastore.ImageDatastore}	78	{["Open Datastore"]}
{'DUSK'}	{1x1 matlab.io.datastore.ImageDatastore}	72	{["Open Datastore"]}
{'AFTN'}	{1x1 matlab.io.datastore.ImageDatastore}	36	{["Open Datastore"]}
{'DAWN'}	{1x1 matlab.io.datastore.ImageDatastore}	90	{["Open Datastore"]}
{'DIST'}	{1x1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'AGL' }	{1x1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'SIDE'}	{1x1 matlab.io.datastore.ImageDatastore}	276	{["Open Datastore"]}
{'ROT' }	{0x0 double }	0	{0x0 double }



Review data Manually



Compute data coverage per data requirement



Examine data coverage in each of the operational conditions

Missing requirement on sign rotation

Thank you