

# MATLAB EXPO

## 2021

아키텍처와 요구사항에 대한 할당 워크플로우

홍 혁기



## What we've heard from YOU

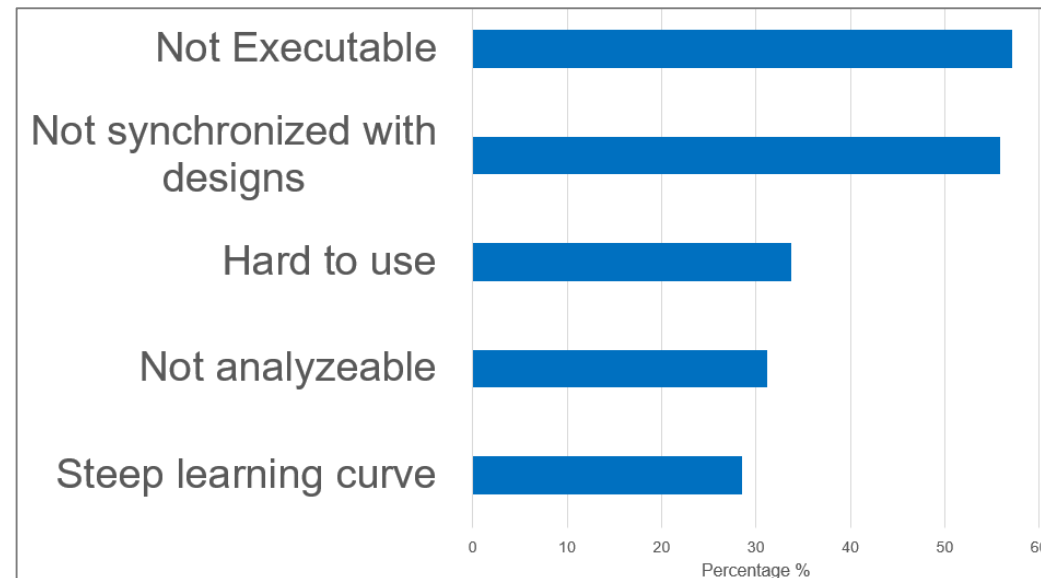
- Model-Based Systems Engineering is a huge improvement over document-based methods

## What we've heard from YOU

- Model-Based Systems Engineering is a huge improvement over document-based methods
- Existing tools are often missing key capabilities

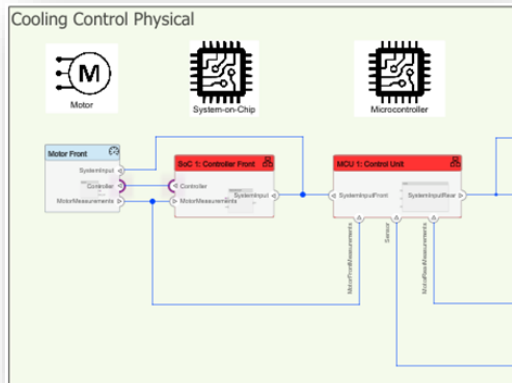
## What we've heard from YOU

- Model-Based Systems Engineering is a huge improvement over document-based methods
- Existing tools are often missing key capabilities



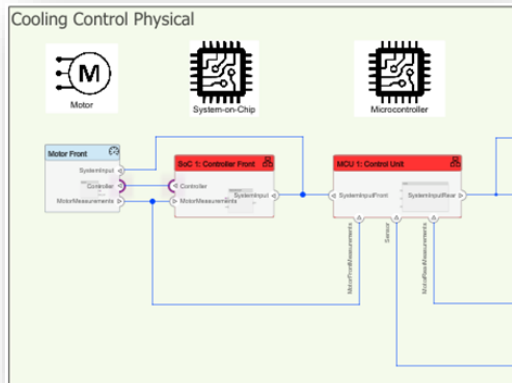
# Why MBSE with MathWorks tools?

## Be Intuitive

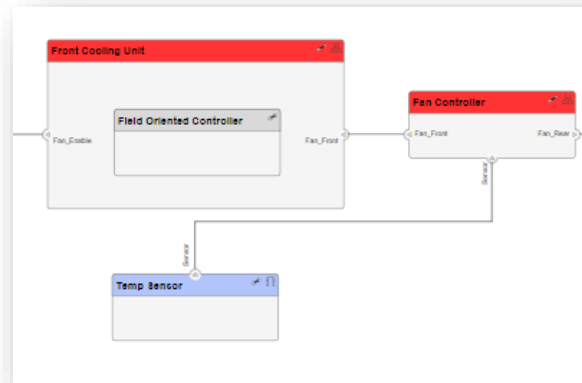


# Why MBSE with MathWorks tools?

## Be Intuitive

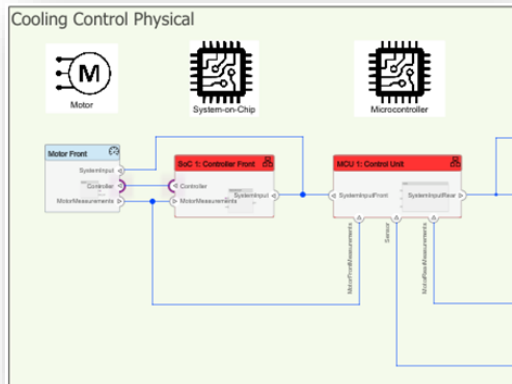


## Tackle Complexity

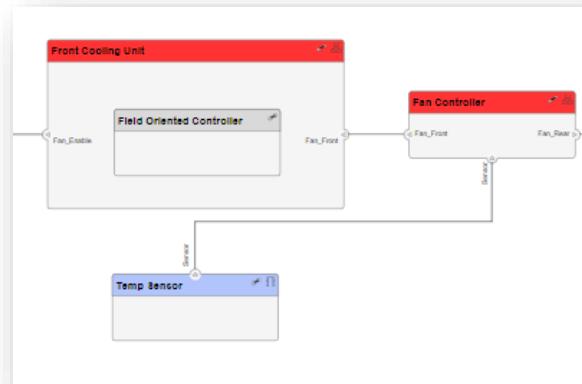


# Why MBSE with MathWorks tools?

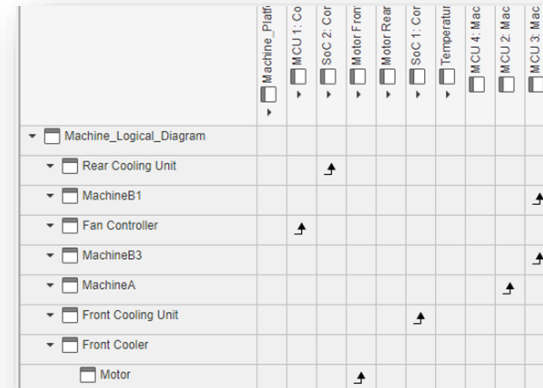
## Be Intuitive



## Tackle Complexity

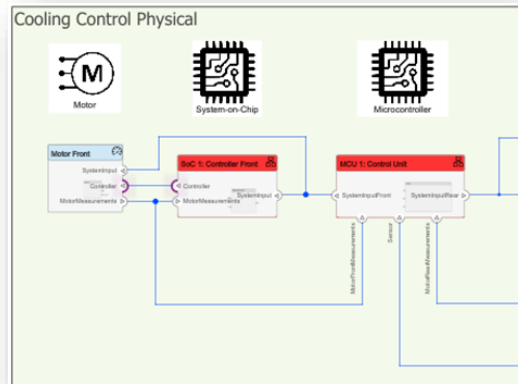


## Facilitate Traceability

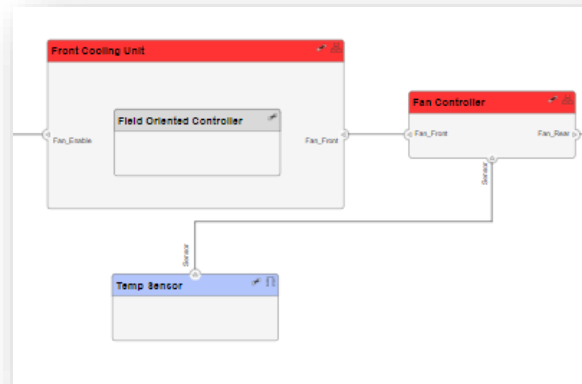


# Why MBSE with MathWorks tools?

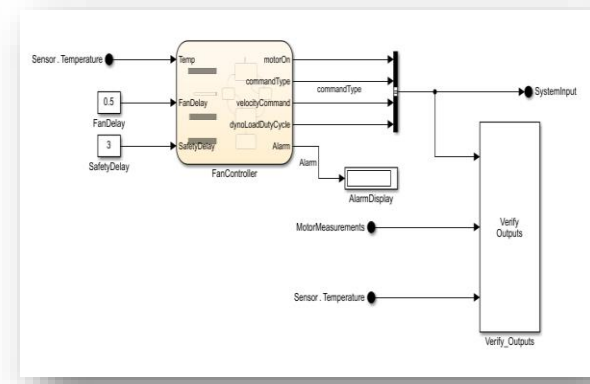
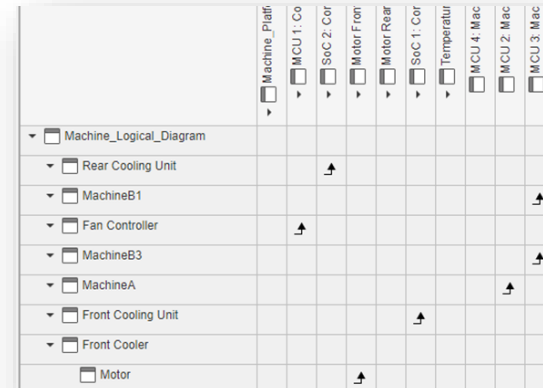
Be Intuitive



Tackle Complexity



Facilitate Traceability Enable Implementation

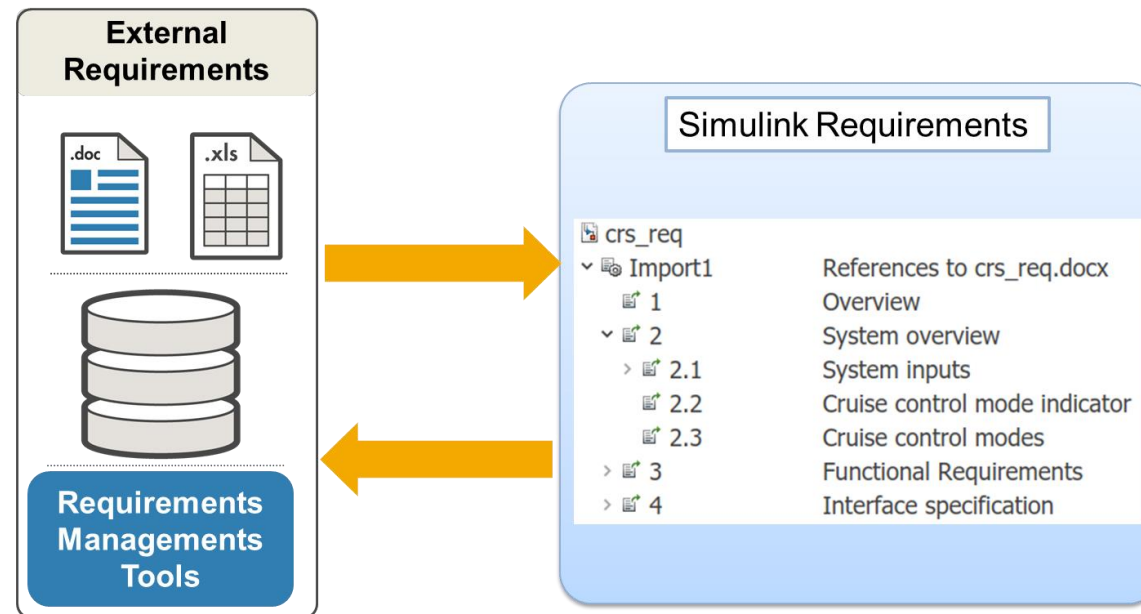


Provide Traceability



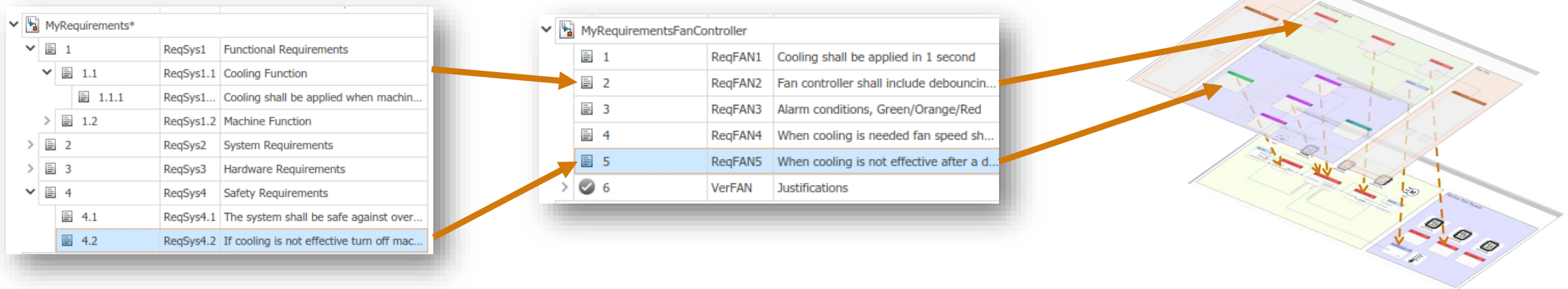
# Key Takeaways

- You can import, write, and store textual requirements right in the same environment as your architecture and design models.



# Key Takeaways

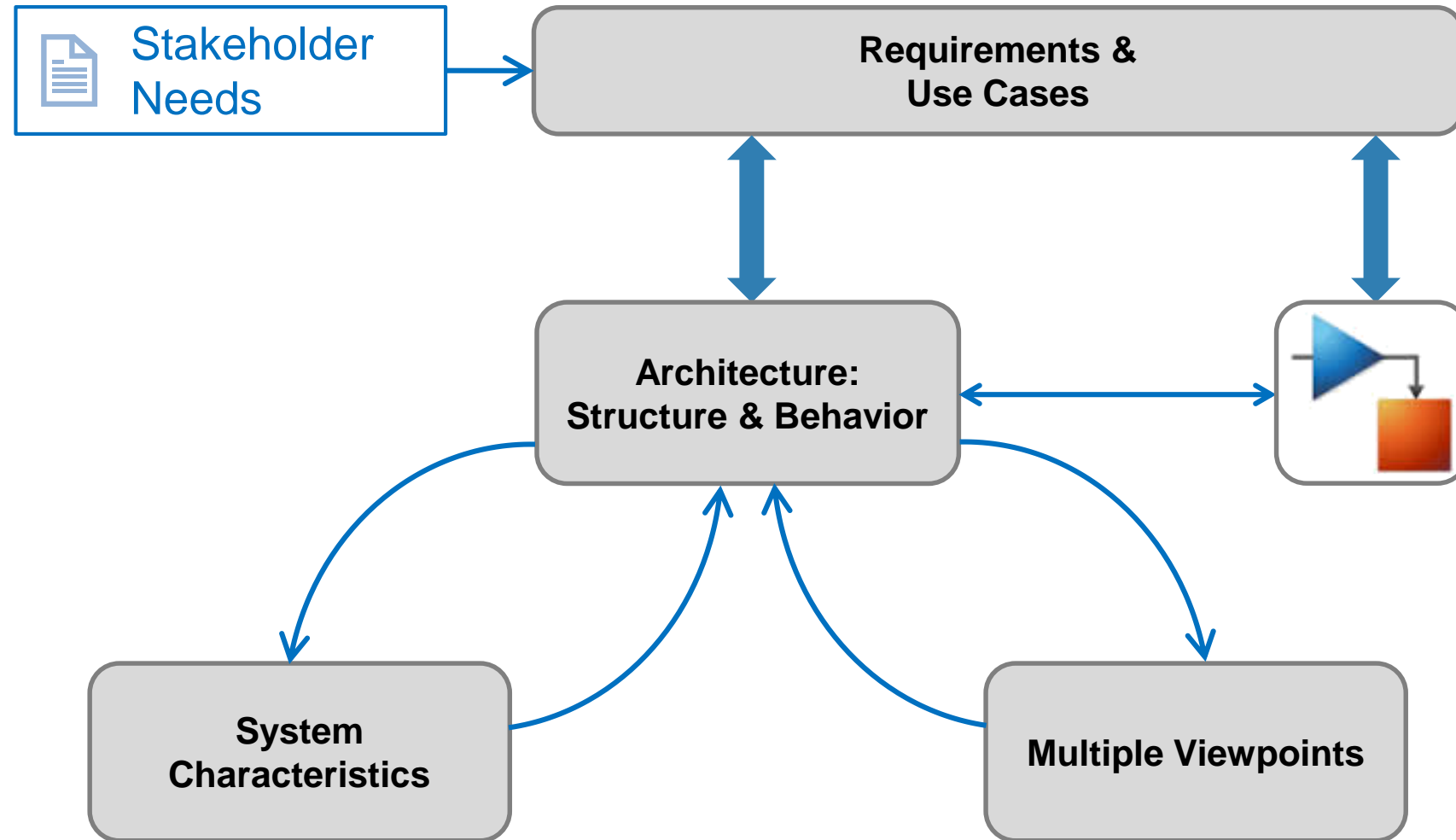
- You can import, write, and store textual requirements right in the same environment as your architecture and design models.
- You can establish relationships among multiple requirements and architecture artifacts to understand the impact of changes.



## Key Takeaways

- You can import, write, and store textual requirements right in the same environment as your architecture and design models.
- You can establish relationships among multiple requirements and architecture artifacts to understand the impact of changes.
- You can visualize those relationships to assess the completeness of your system.

# Typical System Engineering Tasks

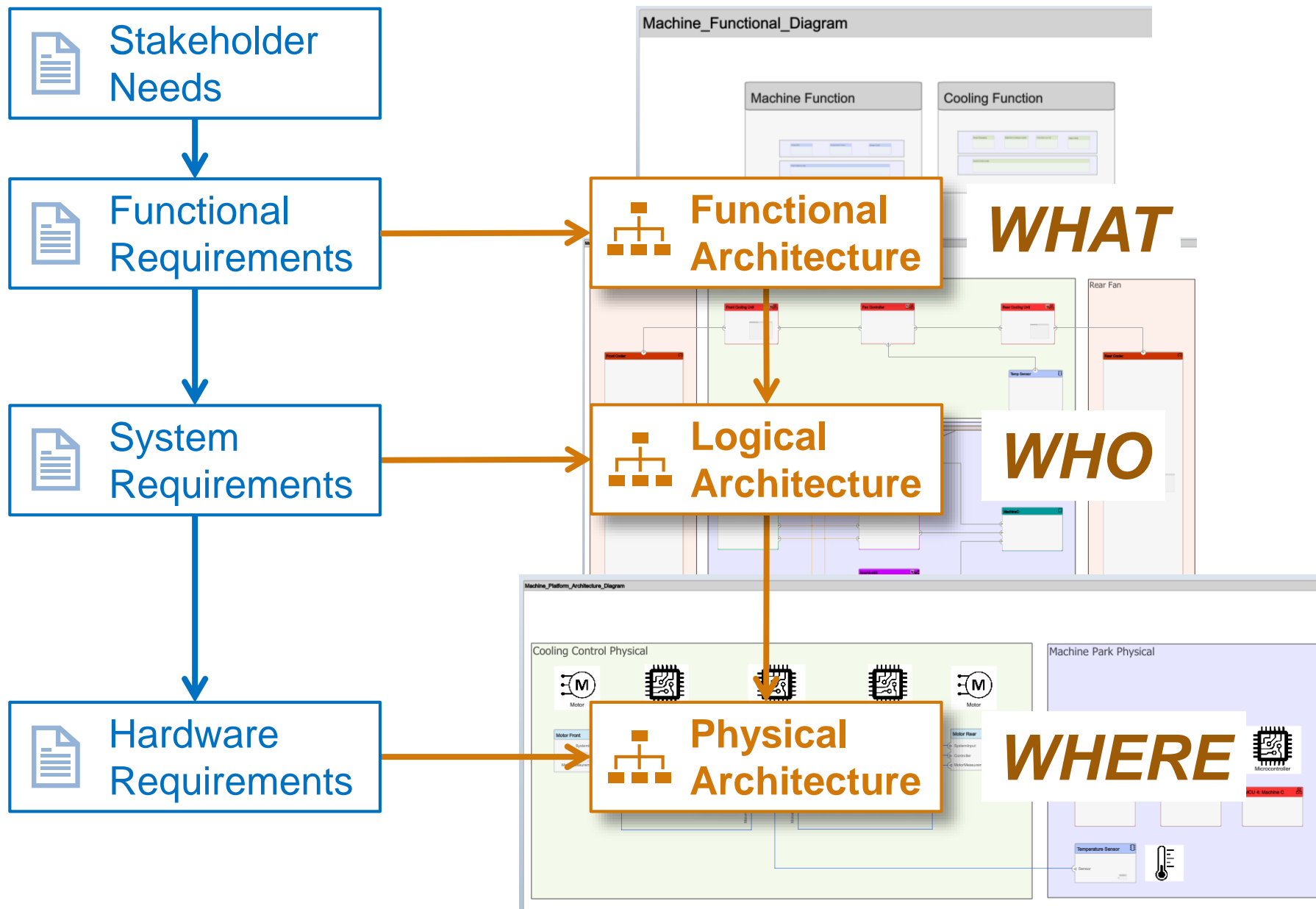




Stakeholder  
Needs

**Requirements &  
Use Cases**

**Architecture:  
Structure & Behavior**



# Traceability between artifacts



# Traceability between artifacts





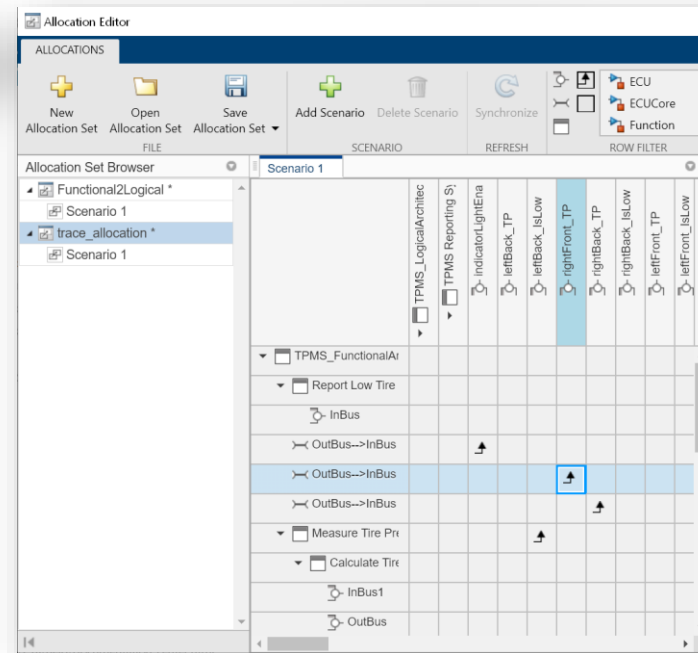
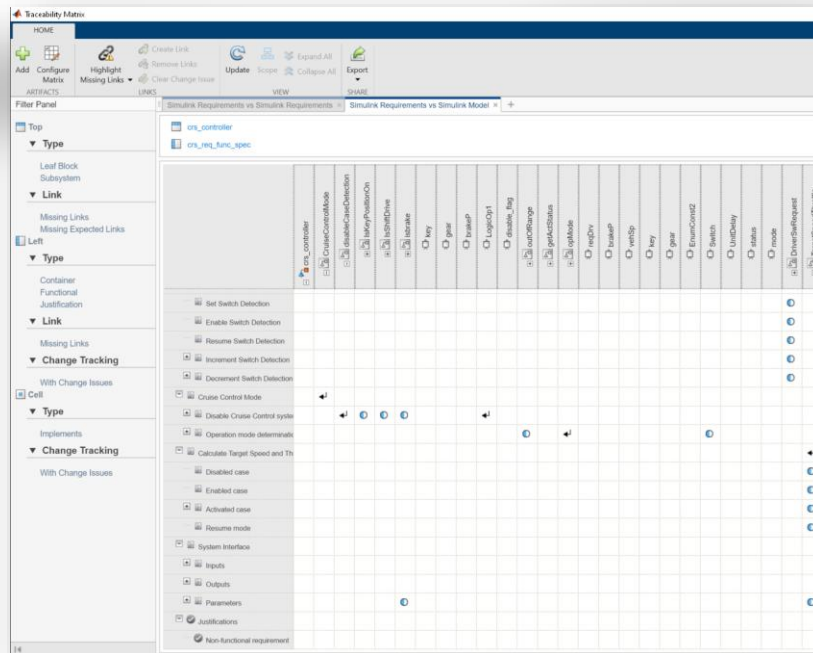
# Traceability between artifacts

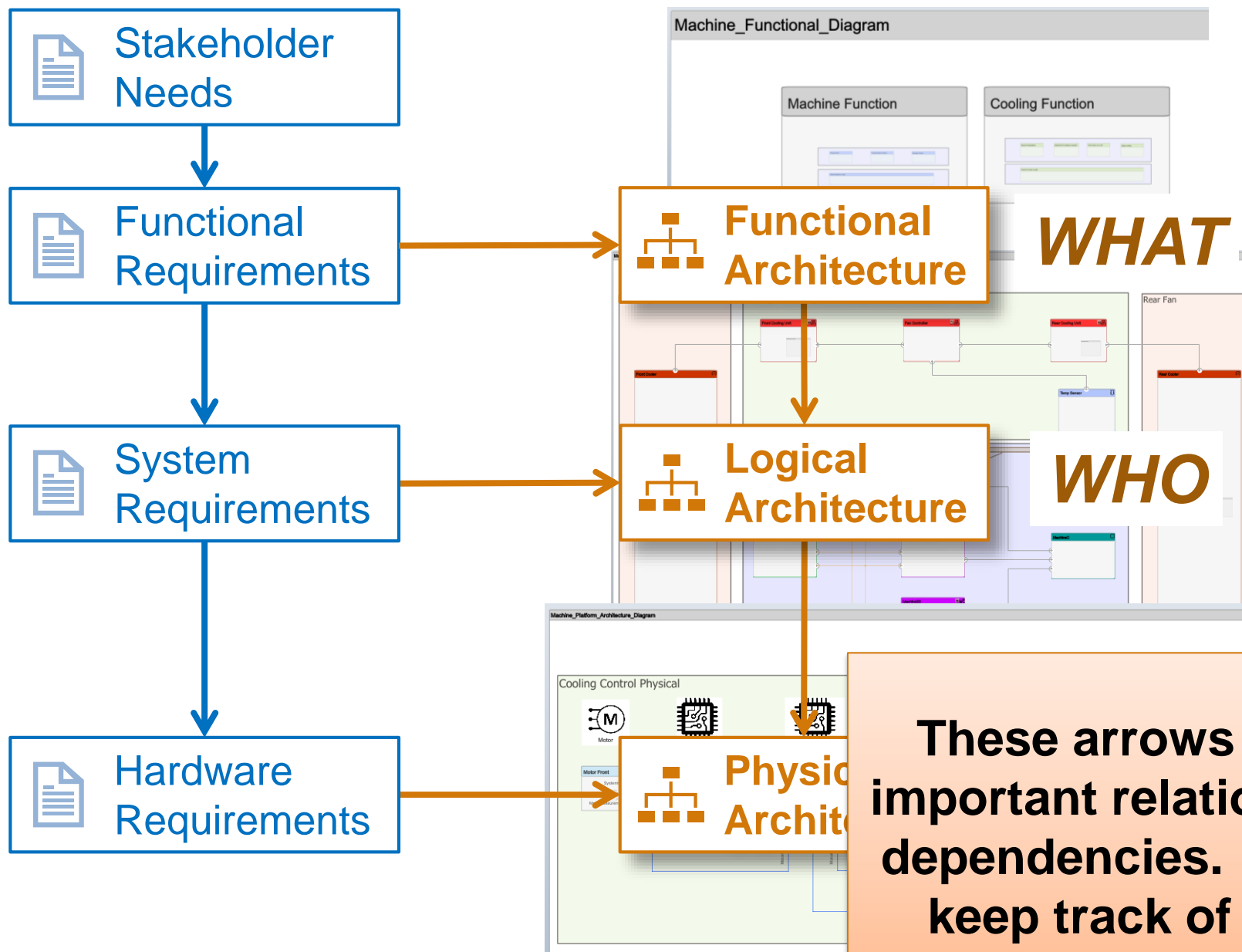


# Traceability between artifacts

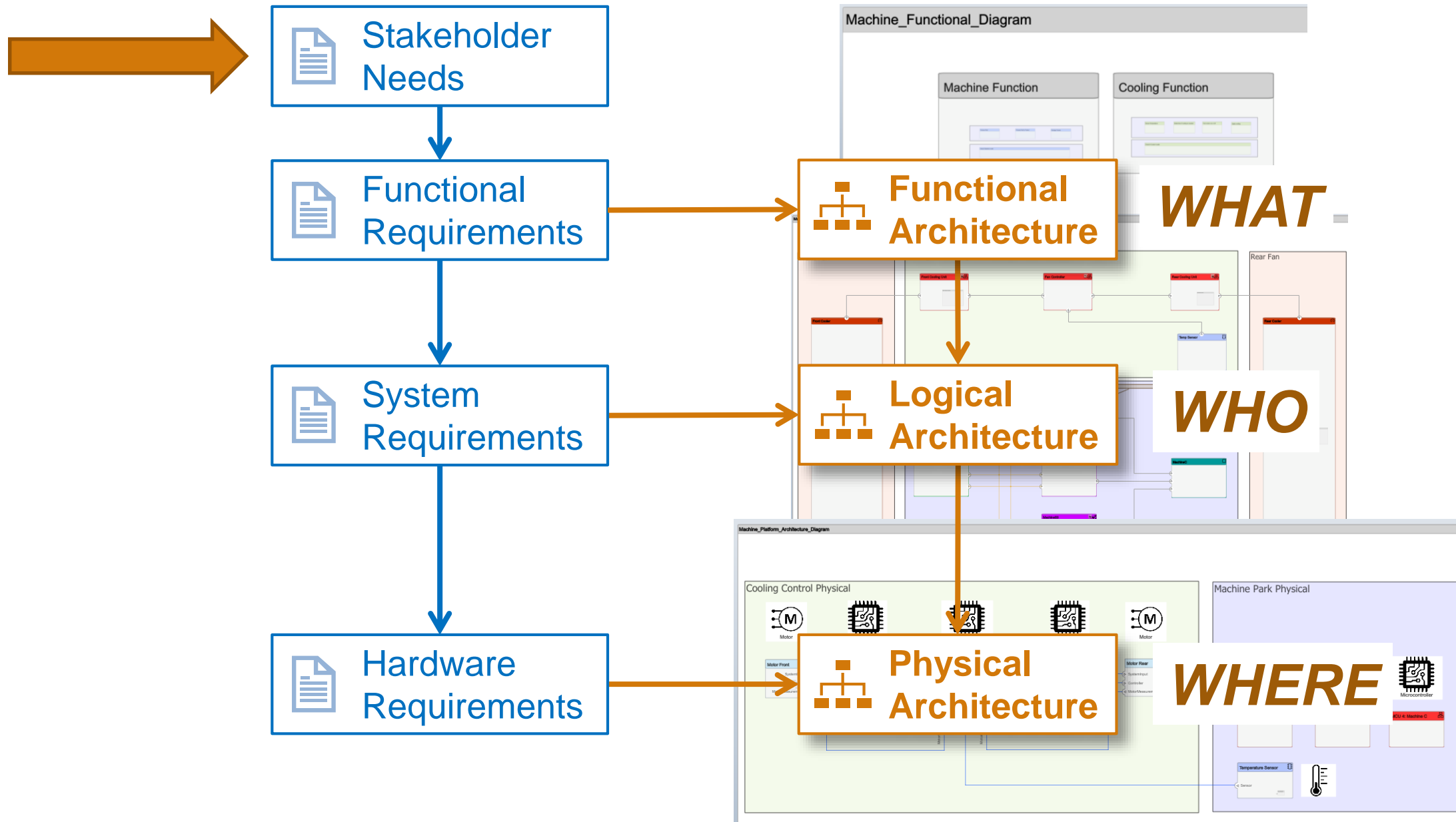


# Traceability between artifacts

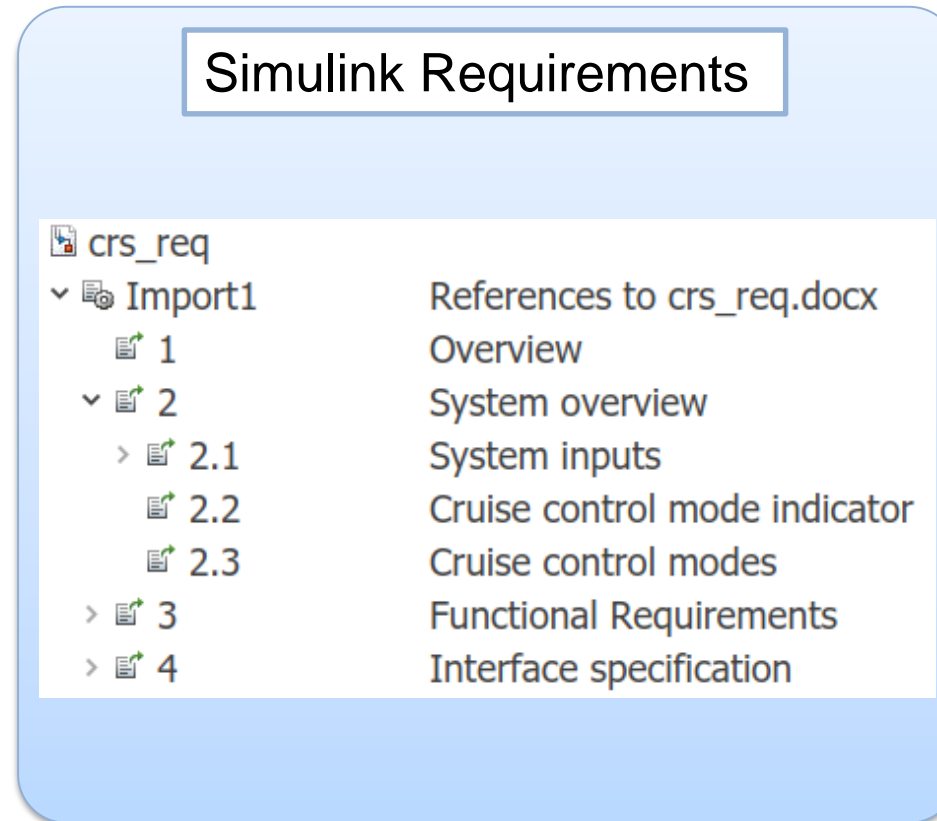
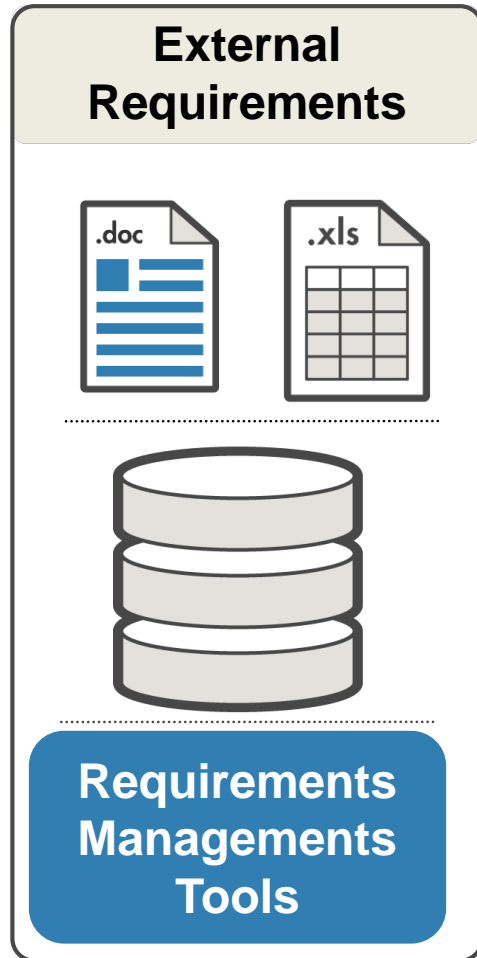




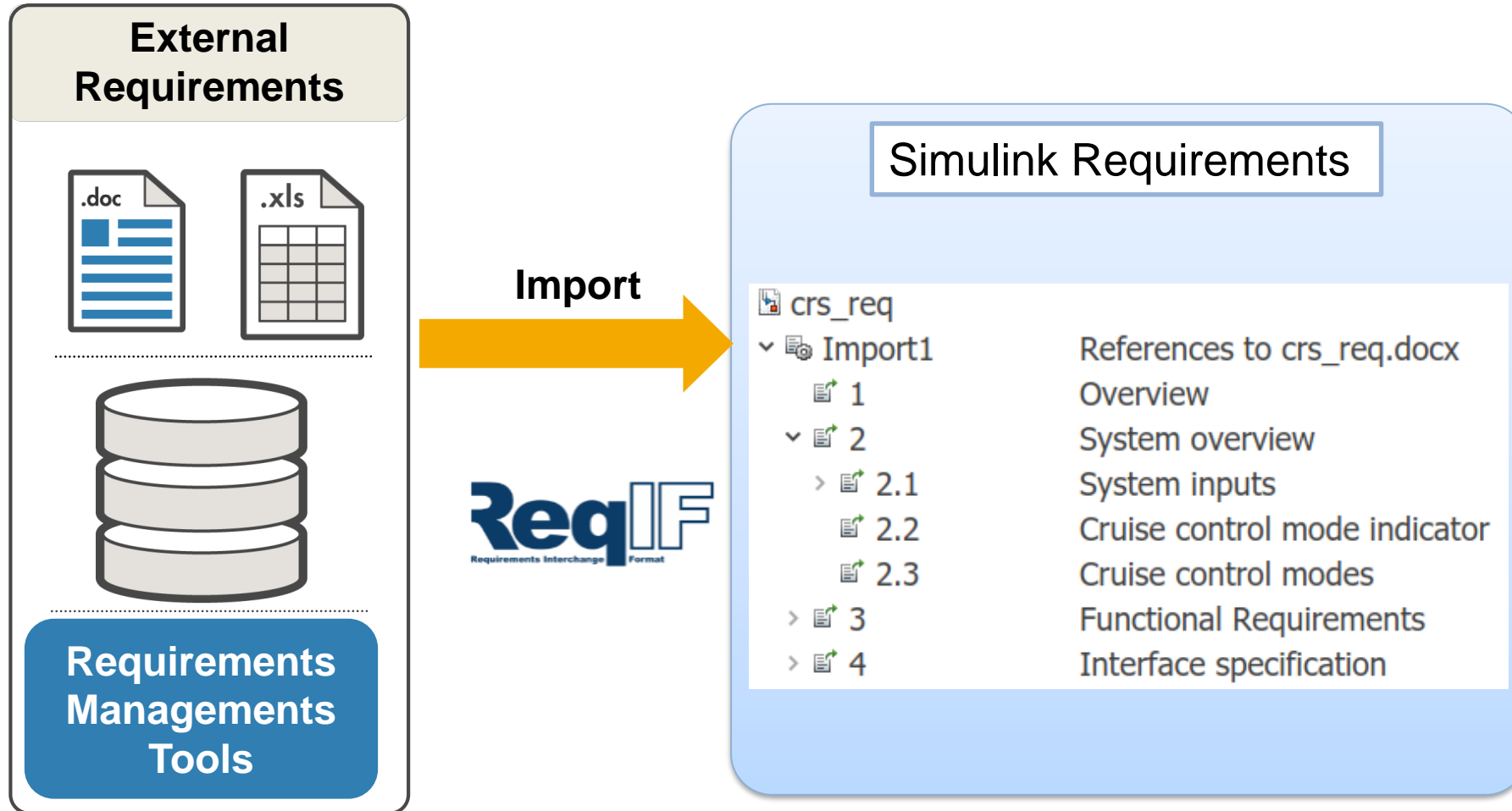
**These arrows represent important relationships and dependencies. How do we keep track of them all?**



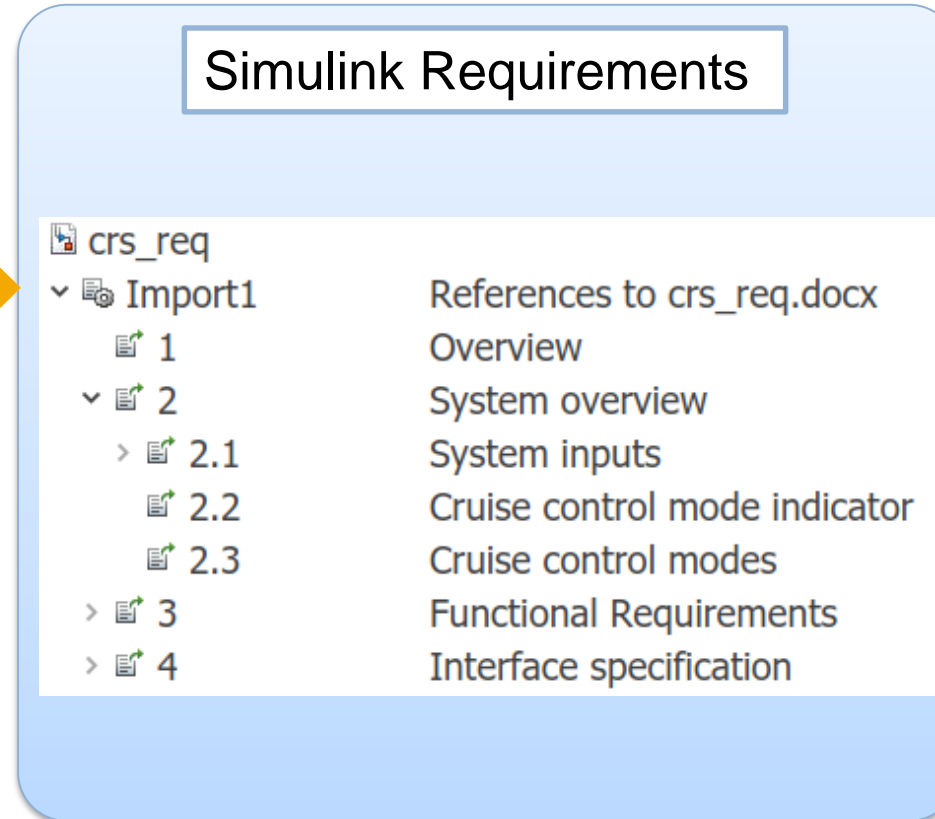
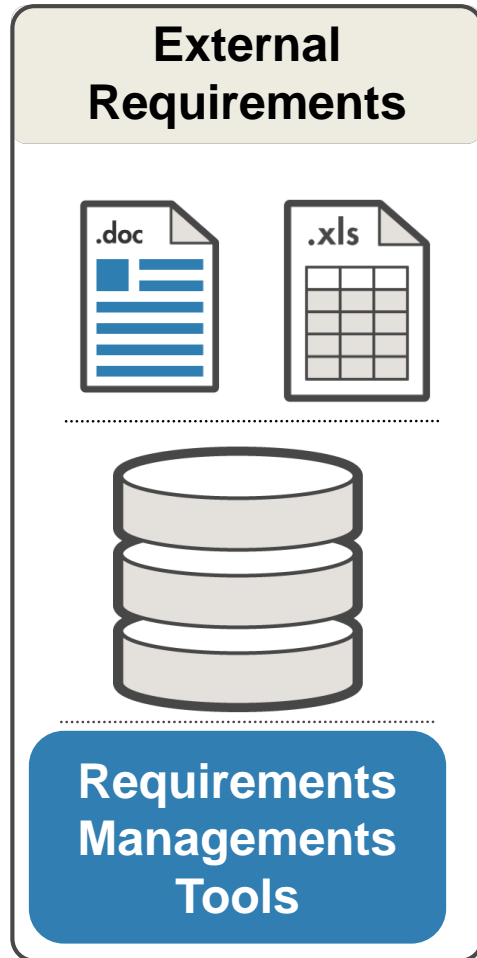
# Exchange Data with Third Party Requirements Tools



# Exchange Data with Third Party Requirements Tools



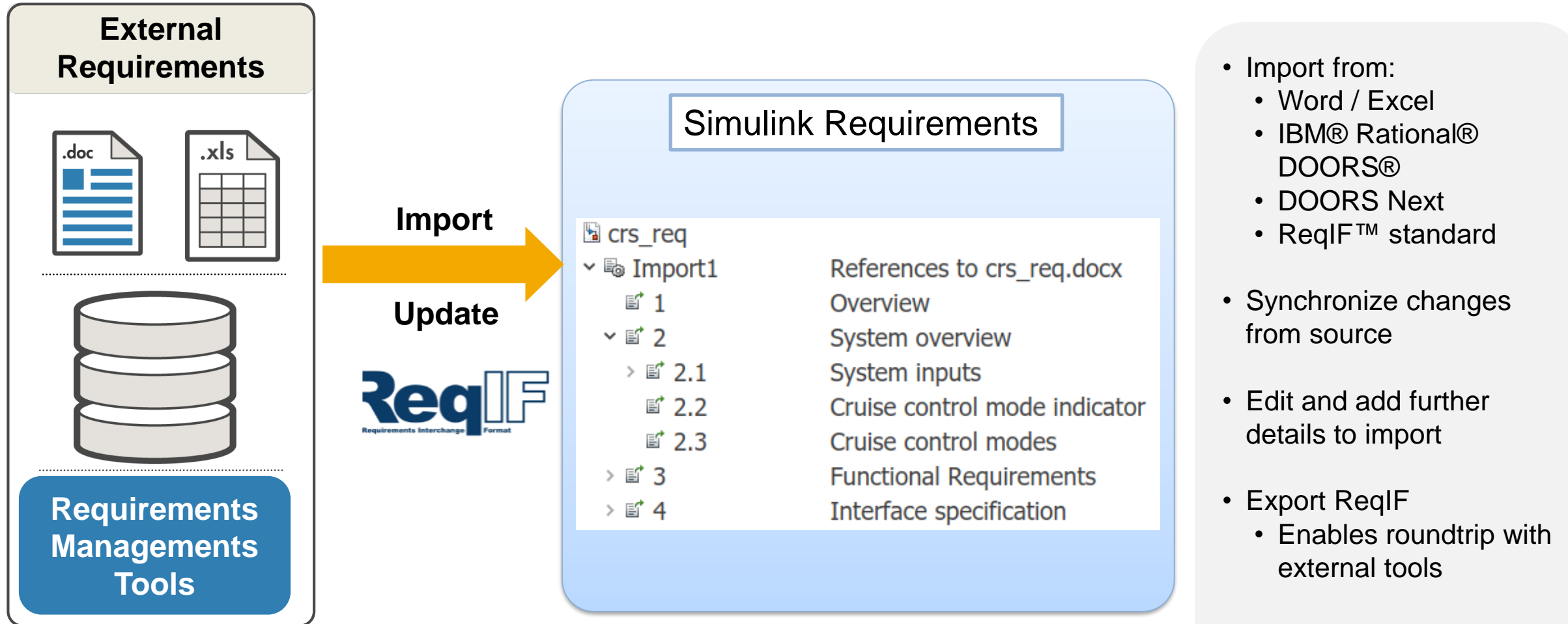
# Exchange Data with Third Party Requirements Tools



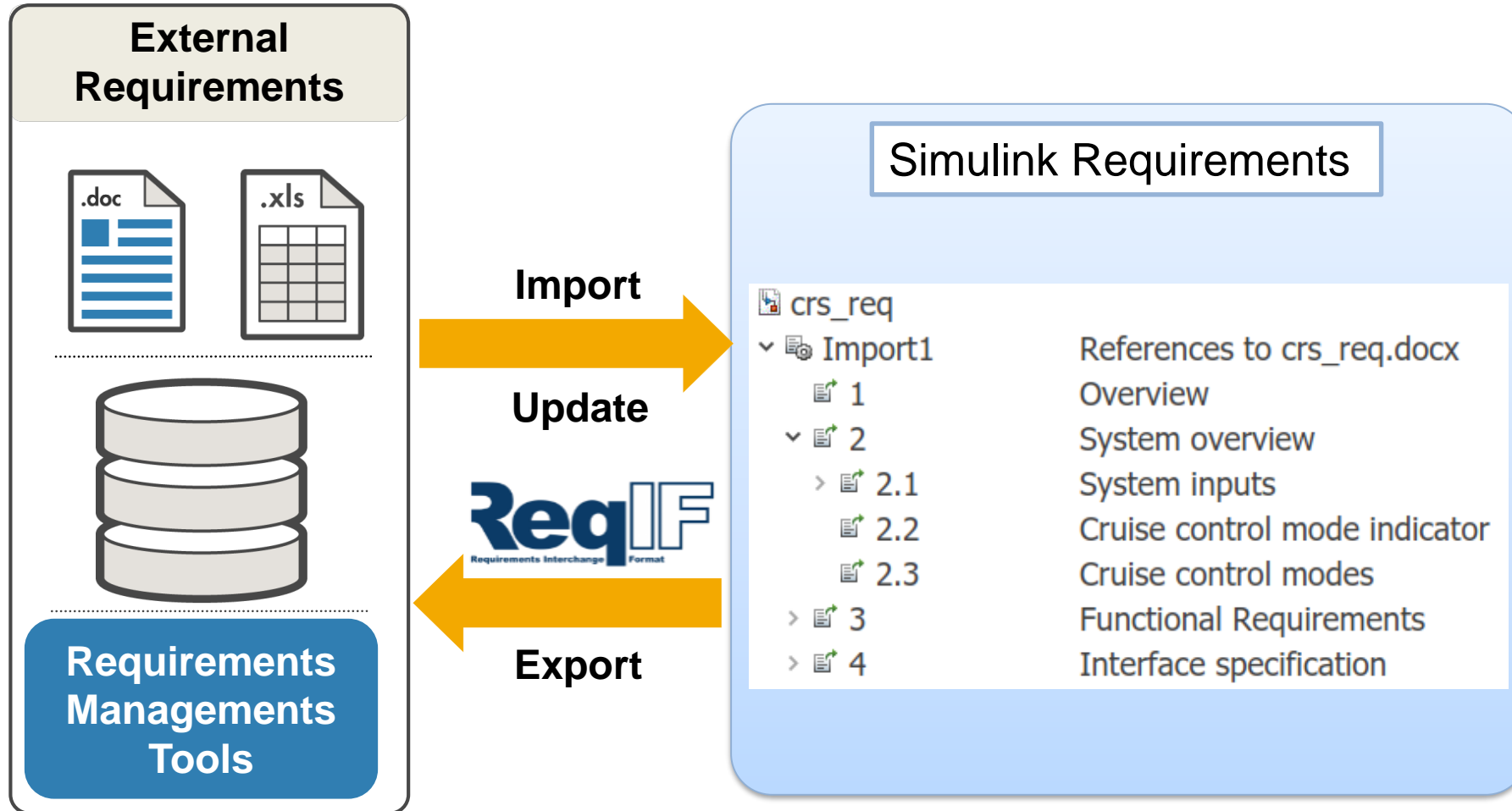
- Import from:
  - Word / Excel
  - IBM® Rational® DOORS®
  - DOORS Next
  - ReqIF™ standard
- Synchronize changes from source
- Edit and add further details to import
- Export ReqIF
  - Enables roundtrip with external tools



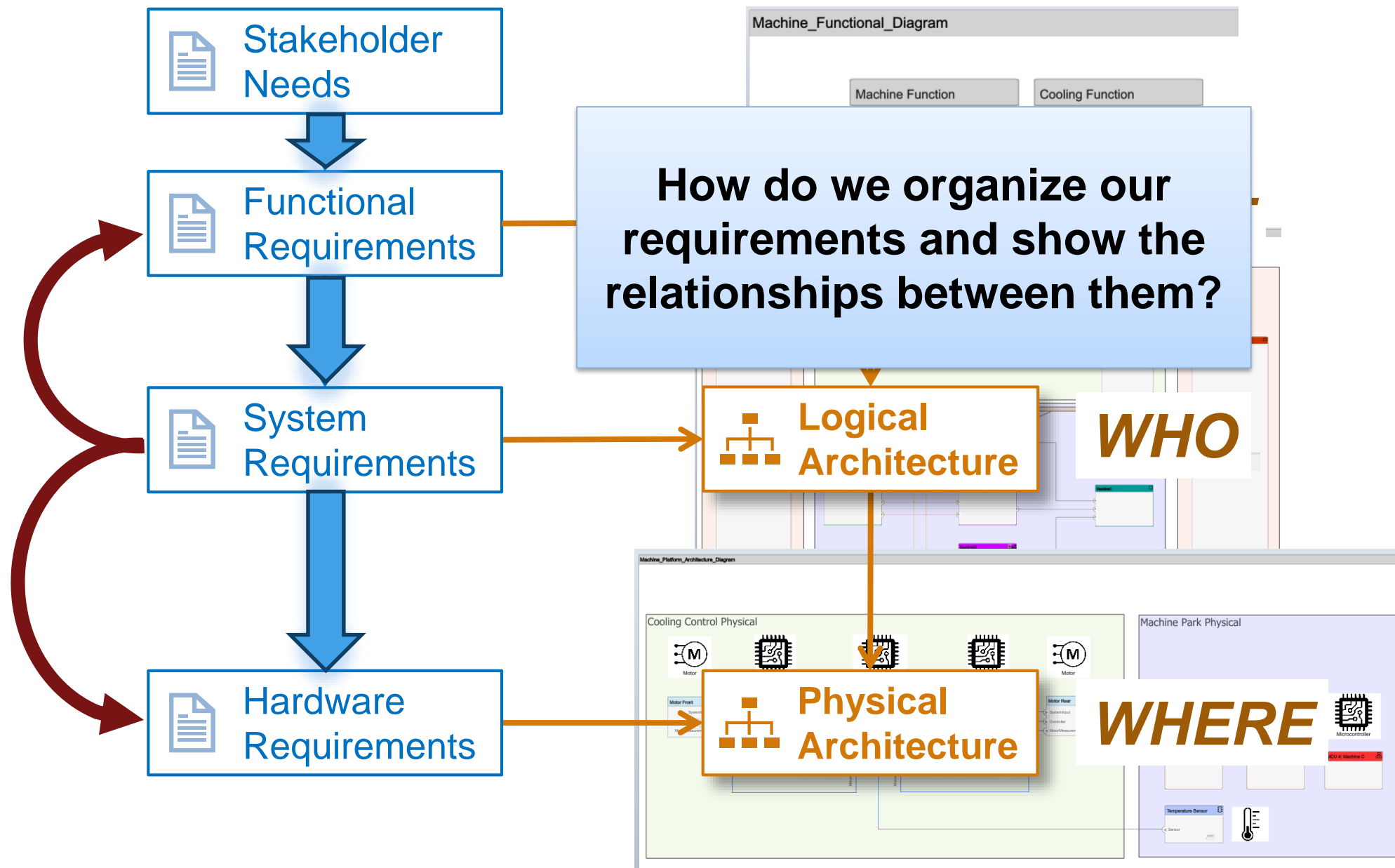
# Exchange Data with Third Party Requirements Tools



# Exchange Data with Third Party Requirements Tools



- Import from:
  - Word / Excel
  - IBM® Rational® DOORS®
  - DOORS Next
  - ReqIF™ standard
- Synchronize changes from source
- Edit and add further details to import
- Export ReqIF
  - Enables roundtrip with external tools



# Traceability between Requirements

The screenshot shows the Requirements Editor interface. The toolbar includes options for New Requirement Set, Open, Save, Import, Close, Add Requirement, Promote Requirement, Demote Requirement, Delete, Add Link, Clear Issue, Show Requirements, Show Links, Search, Traceability Matrix, Export, and Help. The main area is divided into a table of requirements and a details pane.

Index	ID	Summary
MyRequirements		
1	ReqSys1	Functional Requirements
2	ReqSys2	System Requirements
3	ReqSys3	Hardware Requirements
4	ReqSys4	Safety Requirements
MyRequirementsFanController		
MyRequirementsFieldOrientedController		

**Details**

To create a new requirement set to store requirements, click **New Requirement Set** . Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click **Add Requirement** . In the **Properties** pane, enter details for the requirement.

To add a child requirement, right-click a requirement and select **Add Child Requirement**.

To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see [Getting Started](#) in the documentation.

To view the loaded links, click **Show Links** in the toolbar.

Change the source - destination relationship by selecting a link, and choosing a **Type** from the dropdown list in the **Properties** pane.

# Change Management

The screenshot displays the MATLAB Requirements tool interface. The top menu bar includes options like New Requirement Set, Open, Import, Close, Add Requirement, Demote Requirement, Add Link, Clear Issue, Show Requirements, Show Links, Search, Traceability Matrix, Export, and Help. Below the menu is a toolbar with icons for these actions.

The main workspace is divided into two panes. The left pane shows a hierarchical tree of requirements:

Index	ID	Summary
MyRequirements		
1	ReqSys1	Functional Requirements
1.1	ReqSys1.1	Cooling Function
1.1.1	ReqSys1...	Cooling shall be applied when machin...
1.2	ReqSys1.2	Machine Function
2	ReqSys2	System Requirements
3	ReqSys3	Hardware Requirements
4	ReqSys4	Safety Requirements
4.1	ReqSys4.1	The system shall be safe against over...
4.2	ReqSys4.2	If cooling is not effective turn off mac...
MyRequirementsFanController		
1	ReqFAN1	Cooling shall be applied in 1 second
2	ReqFAN2	Fan controller shall include debouncin...
3	ReqFAN3	Alarm conditions, Green/Orange/Red
4	ReqFAN4	When cooling is needed fan speed sh...
5	ReqFAN5	When cooling is not effective after a d...
6	VerFAN	Justifications
MyRequirementsFieldOrientedController		

The right pane shows the details for the selected requirement, ReqFAN5:

**Requirement: ReqFAN5**

**Details**

**Properties**

- Type: Functional
- Index: 5
- Custom ID: ReqFAN5
- Summary: When cooling is not effective after a defined time machines should be turned off

**Description** | Rationale

Time to determine if cooling is effective should be configurable

**Keywords:**

**Revision information:**

**Links**

- Derived from: [ReqFAN1 Cooling shall be applied in 1 second](#)
- Implemented by: [MachineOff](#)
- Related to: [ReqSys4.2 If cooling is not effective turn off machines](#)

**Comments**

# Requirements Traceability Metrics

The screenshot shows the Requirements Editor application window. The interface is divided into several sections:

- Toolbar:** Contains icons for File (New, Open, Save, Import, Close), Requirements (Add, Delete, Promote, Demote), Links (Add, Clear), View (Show Requirements, Show Links), Edit (Search), Analysis (Traceability Matrix), Share (Export), and Documentation (Help).
- Requirements Tree:** A hierarchical tree view showing requirement sets and their children. The tree is expanded to show 'MyRequirements\*' and 'MyRequirementsFanController'.
- Details Pane:** A pane on the right side of the window that displays instructional text for various actions.

Index	ID	Summary
MyRequirements*		
1	ReqSys1	Functional Requirements
1.1	ReqSys1.1	Cooling Function
1.1.1	ReqSys1...	Cooling shall be applied when machin...
1.2	ReqSys1.2	Machine Function
2	ReqSys2	System Requirements
3	ReqSys3	Hardware Requirements
4	ReqSys4	Safety Requirements
4.1	ReqSys4.1	The system shall be safe against over...
4.2	ReqSys4.2	If cooling is not effective turn off mac...
MyRequirementsFanController		
1	ReqFAN1	Cooling shall be applied in 1 second
2	ReqFAN2	Fan controller shall include debouncin...
3	ReqFAN3	Alarm conditions, Green/Orange/Red
4	ReqFAN4	When cooling is needed fan speed sh...
5	ReqFAN5	When cooling is not effective after a d...
6	VerFAN	Justifications
MyRequirementsFieldOrientedController		

**Details**

To create a new requirement set to store requirements, click **New Requirement Set** . Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click **Add Requirement** . In the **Properties** pane, enter details for the requirement.

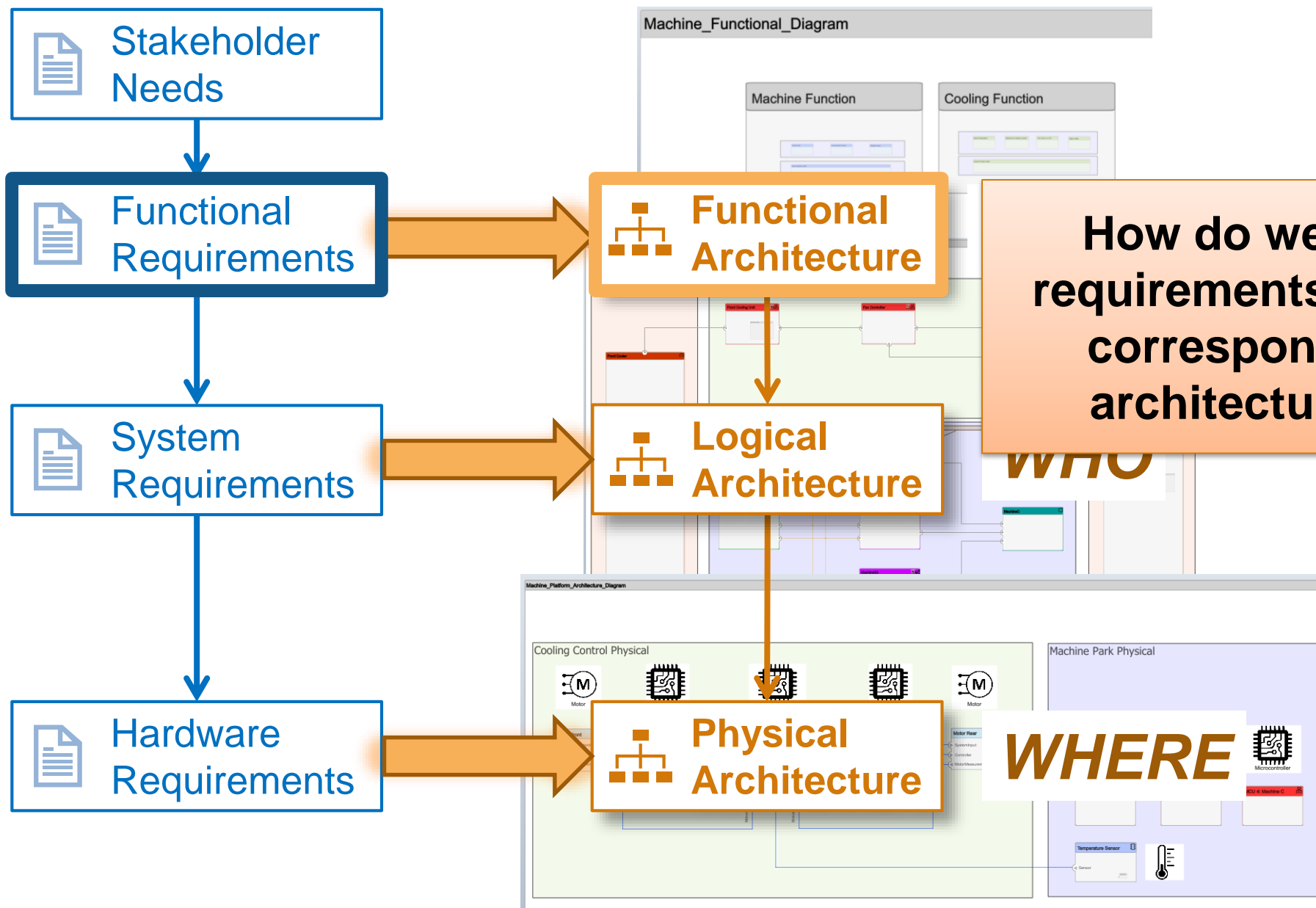
To add a child requirement, right-click a requirement and select **Add Child Requirement**.

To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see [Getting Started](#) in the documentation.

To view the loaded links, click **Show Links** in the toolbar.

Change the source - destination relationship by selecting a link, and choosing a **Type** from the dropdown list in the **Properties** pane.



How do we link requirements to the corresponding architectures?

WHERE

# Traceability between Requirement and Architecture

The screenshot displays the Simulink Machine Functional Diagram (MFD) editor interface. The main workspace shows a diagram titled "Machine\_Functional\_Diagram" containing two functional blocks: "Machine Function" and "Cooling Function". The "Machine Function" block is highlighted with a blue border. The "Cooling Function" block is also visible. The interface includes a top menu bar with tabs for SIMULATION, DEBUG, MODELING, FORMAT, and APPS. The MODELING tab is active, showing various toolbars for design, component, and views. The Property Inspector on the right shows the selected component's properties, including Name (Machine Function) and Stereotype (Add..). The status bar at the bottom indicates "Ready", "228%", and "VariableStepAuto".

Machine\_Functional\_Diagram

Machine\_Functional\_Diagram

Machine Function

Cooling Function

Property Inspector

NAME	VALUE
Architecture	Info
Main	
Name	Machine Function
Stereotype	Add..

Interfaces

Ready

228%

VariableStepAuto



# Hierarchy Diagram

The screenshot displays the Simulink Machine Functional Diagram (MFD) editor. The main workspace shows a hierarchy diagram with two primary components: 'Machine Function' and 'Cooling Function'. The 'Machine Function' component is highlighted with a blue border and contains three sub-components: 'Machine Function', 'Machine Function', and 'Machine Function'. The 'Cooling Function' component contains two sub-components: 'Cooling Function' and 'Cooling Function'. The interface includes a ribbon with tabs for SIMULATION, DEBUG, MODELING, FORMAT, and APPS. The Model Browser on the left shows the current diagram. The Property Inspector on the right shows the selected component's properties, including Name (Machine Function) and Stereotype (Add.).

Machine\_Functional\_Diagram \* - Simulink

SIMULATION DEBUG MODELING FORMAT APPS

Find Compare Environment MANAGE Interface Editor Import base workspace Import MAT-file Import Apply Stereotypes Compose... Reference Compose... Variant Compose... Architecture Views Analysis Model Allocation Editor Update Model Fast Restart Run Stop Stop Time 10.0 Normal SIMULATE

Model Browser Machine\_Functional\_Diagram Machine\_Functional\_Diagram

Machine\_Functional\_Diagram

Machine Function Cooling Function

Property Inspector

Component

Architecture Info

NAME	VALUE
▼ Main	
Name	Machine Function
Stereotype	Add.

Interfaces

Ready 228% VariableStepAuto

# Traceability Metrics

The screenshot displays the Traceability Matrix application window. The interface includes a top toolbar with options like 'Add', 'Configure Matrix', 'Highlight Missing Links', 'Create Link', 'Remove Links', 'Clear Change Issue', 'Update', 'Scope', 'Expand All', 'Collapse All', and 'Export'. A left-hand 'Filter Panel' allows users to filter by 'Type' and 'Link'. The main area shows a table with columns for 'Machine Functional Diagram', 'Machine Logical Diagram', and 'Machine Platform Architecture Diagram'. The table lists various requirements, with 'ReqSys1.1 Cooling Function' highlighted.

	Machine Functional Diagram	Machine Logical Diagram	Machine Platform Architecture Diagram
MyRequirements			
ReqSys1 Functional Requirements			
ReqSys1.1 Cooling Function			
ReqSys1.2 Machine Function			
ReqSys2 System Requirements			
ReqSys3 Hardware Requirements			
ReqSys4 Safety Requirements			

# Traceability Matrix

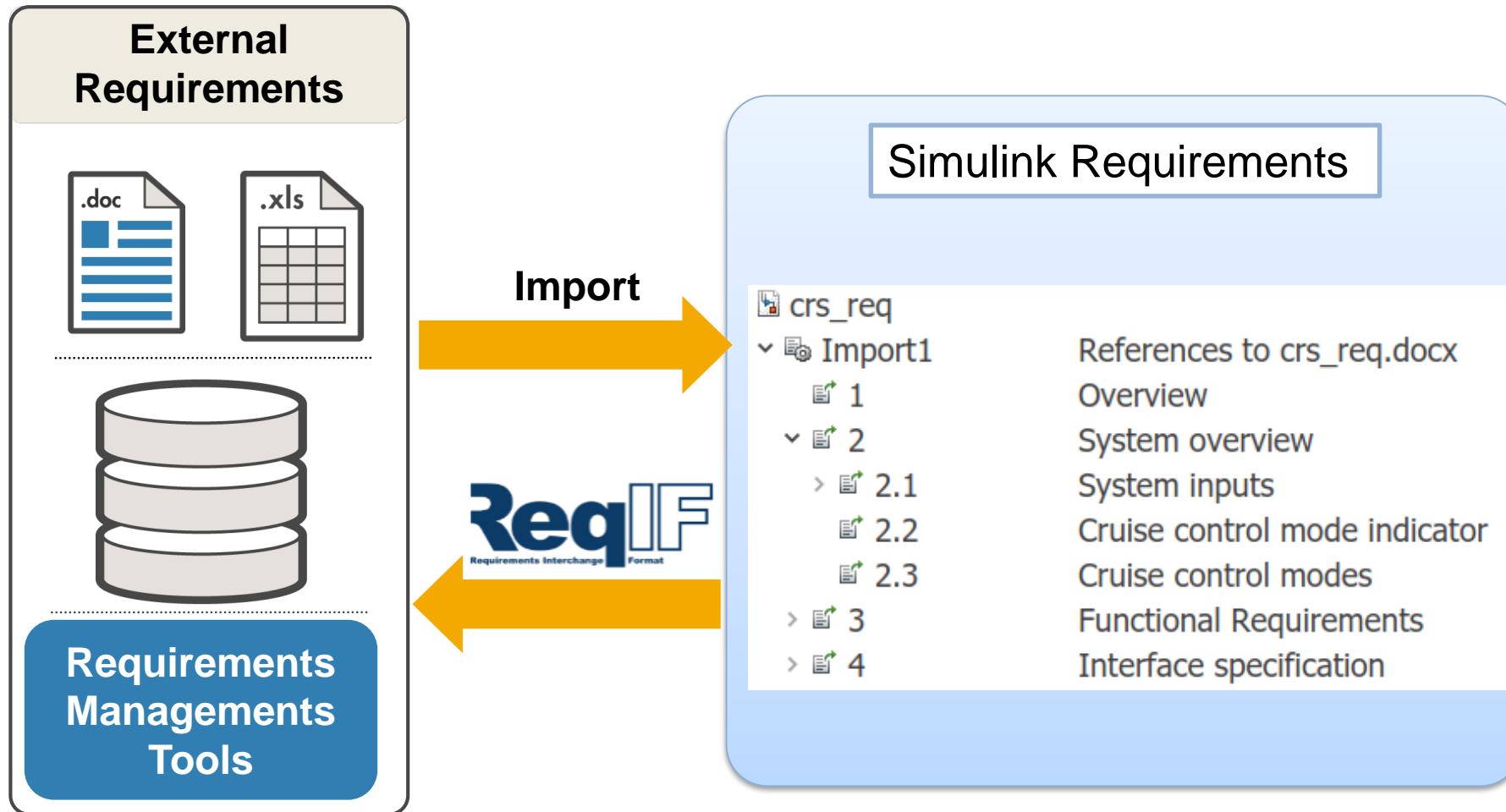
The screenshot shows the Traceability Matrix application window. The title bar reads "Traceability Matrix". The interface includes a ribbon with tabs for "ARTIFACTS", "LINKS", "VIEW", and "SHARE". The "LINKS" tab is active, showing options like "Create Link", "Remove Links", "Update", "Scope", "Expand All", "Collapse All", and "Export".

On the left is a "Filter Panel" with sections for "Top", "Link", "Type", and "Left". The "Type" section is expanded, showing various artifact types like "Adapter", "Architecture View", "Component", "Leaf Block", "System Architecture", and "Port".

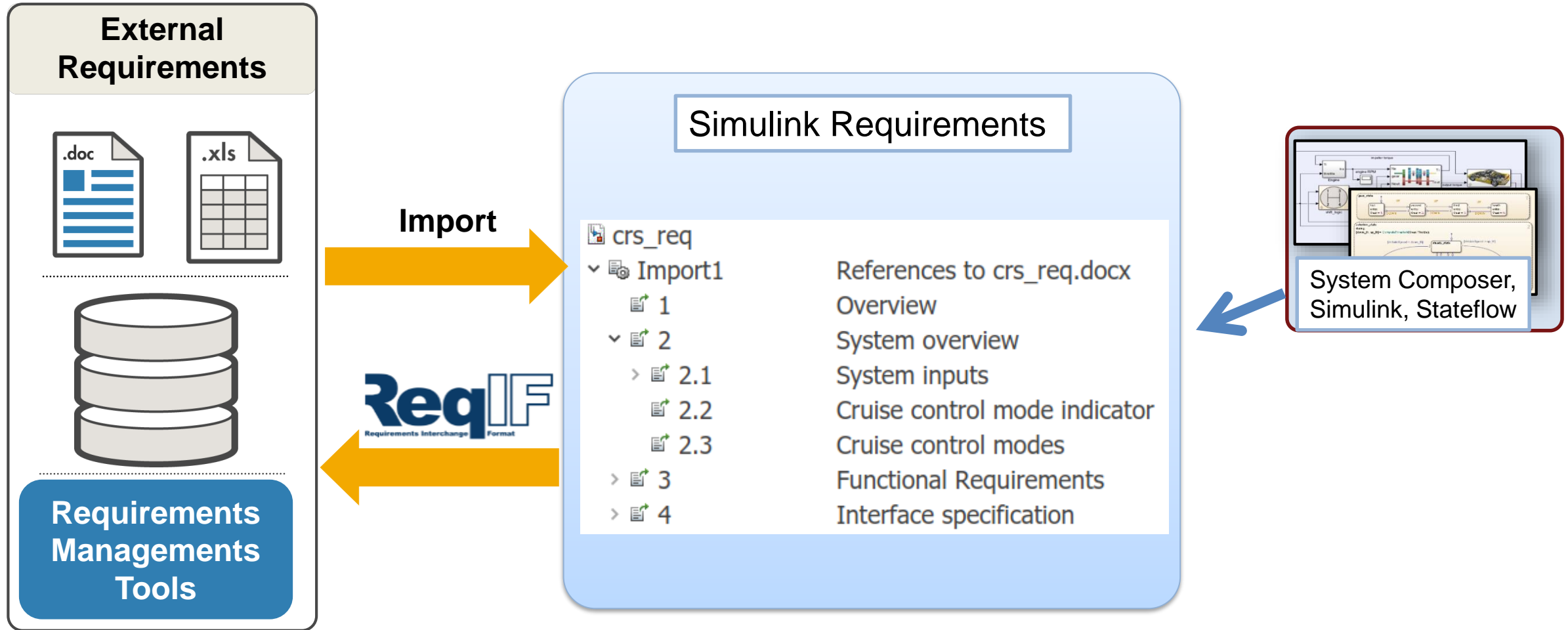
The main area displays a traceability matrix for "Simulink Requirements vs Simulink Requirements". At the top, it lists artifacts: "Machine Functional Diagram, Machine Logical Diagram, Machine Platform Architecture Diagram, MyRequirements".

	Machine_Functional_Diagram	Machine_Logical_Diagram	Machine_Platform_Architectui
MyRequirements			
ReqSys1 Functional Requirem			
ReqSys1.1 Cooling Function		○	
ReqSys1.2 Machine Functio		○	
ReqSys2 System Requirement		○	
ReqSys3 Hardware Requireme			○
ReqSys4 Safety Requirements	○	○	

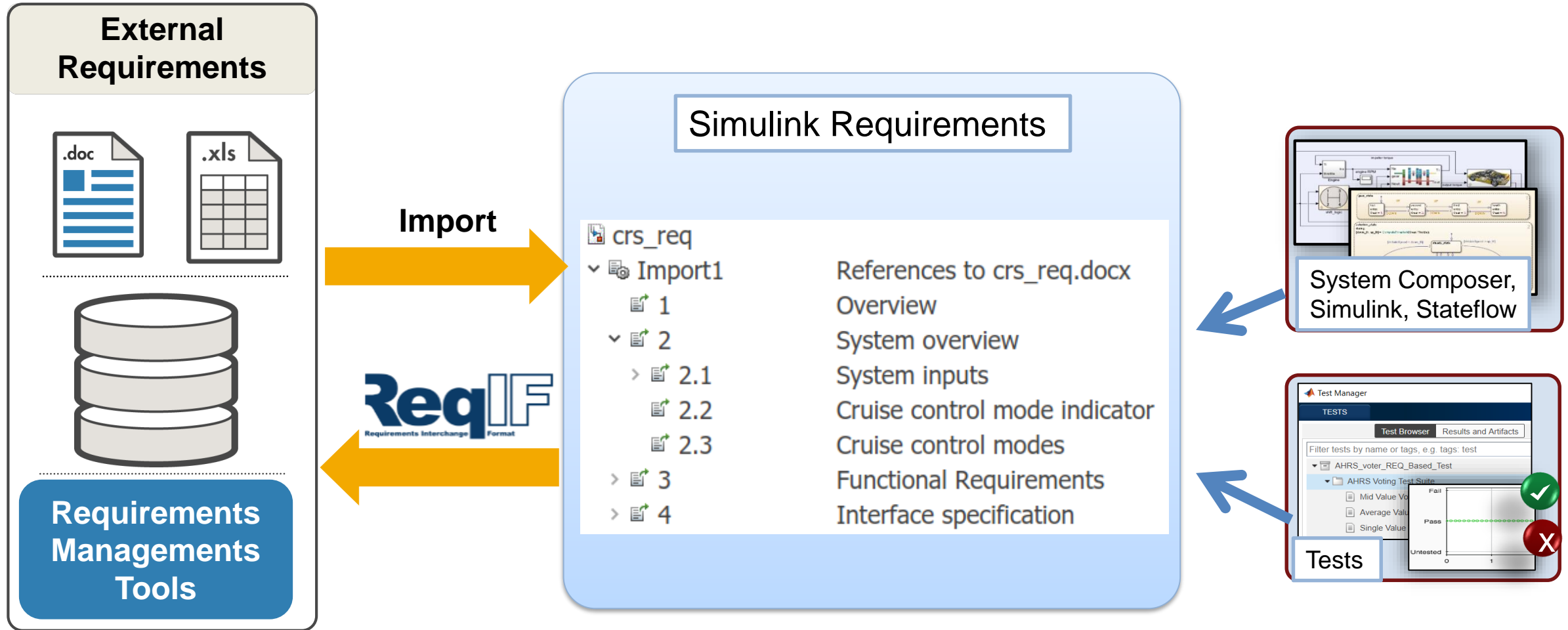
# Export Links for Traceability to Model and Test



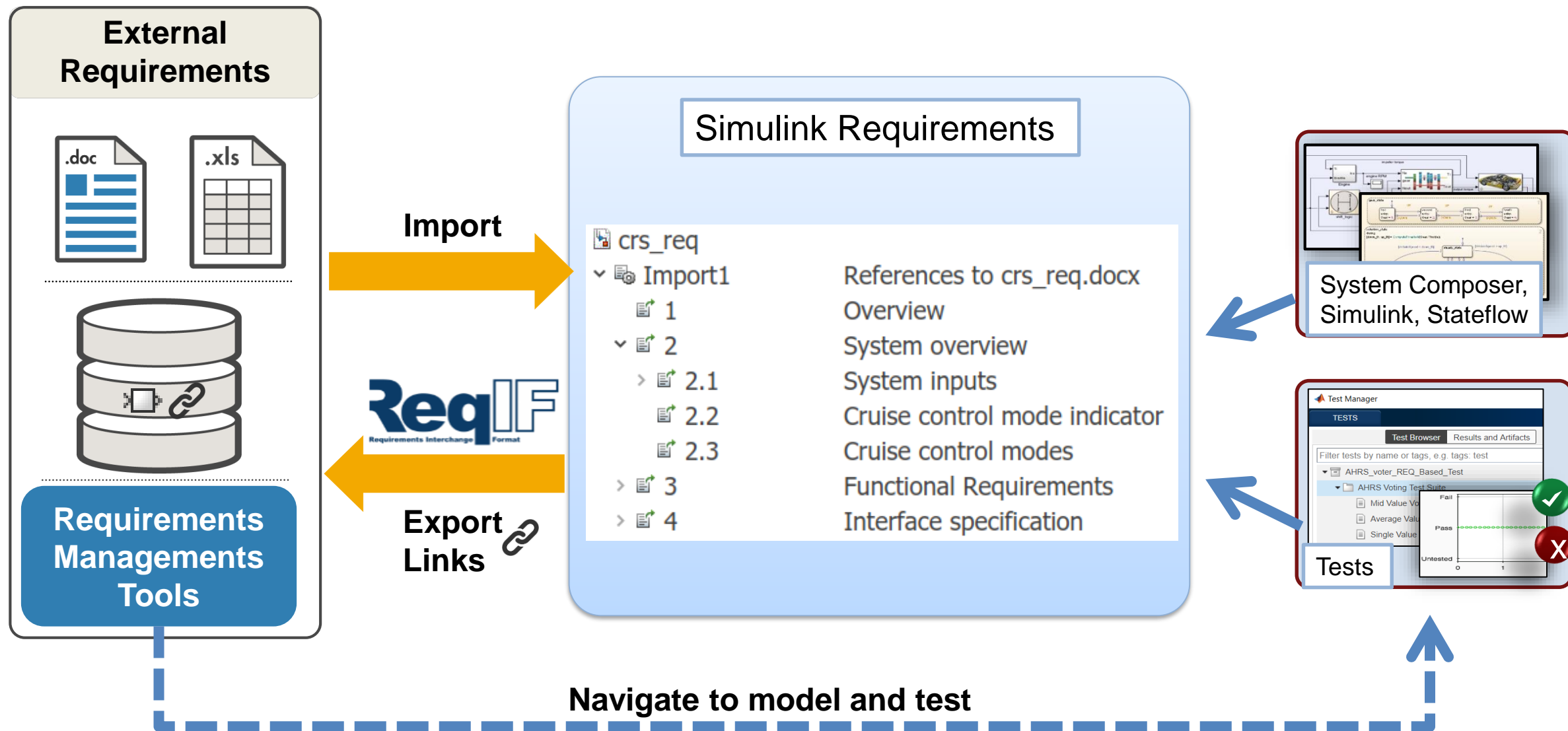
# Export Links for Traceability to Model and Test



# Export Links for Traceability to Model and Test



# Export Links for Traceability to Model and Test



Stakeholder Needs

Functional Requirements

How do we show the relationships between architectures?

Hardware Requirements

Functional Architecture

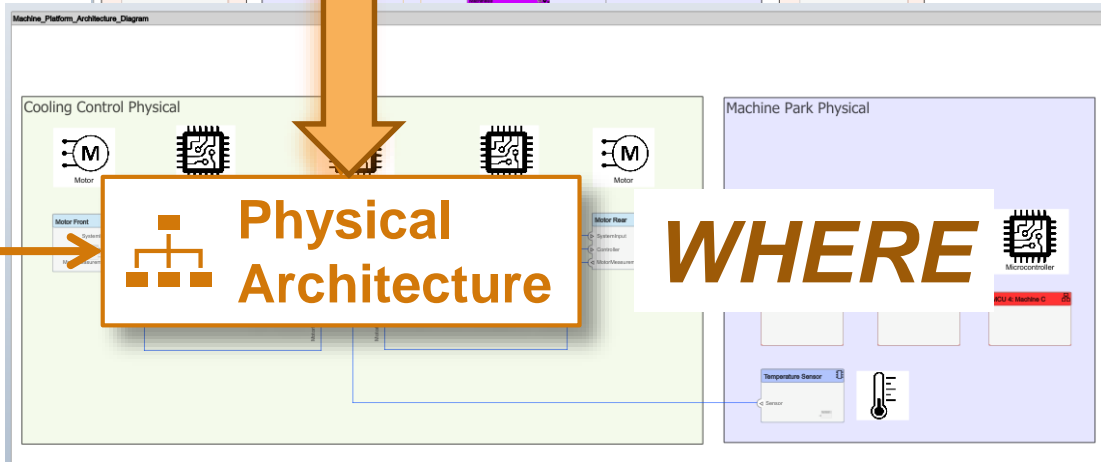
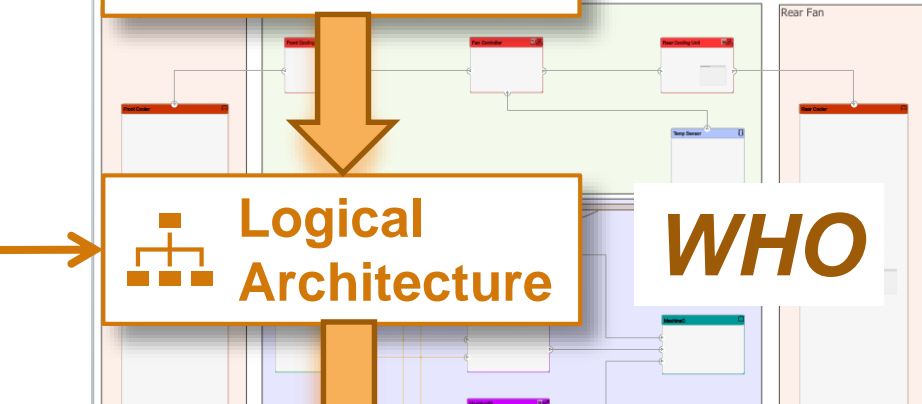
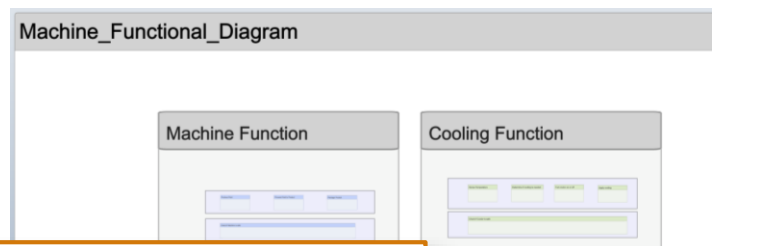
Logical Architecture

Physical Architecture

WHAT

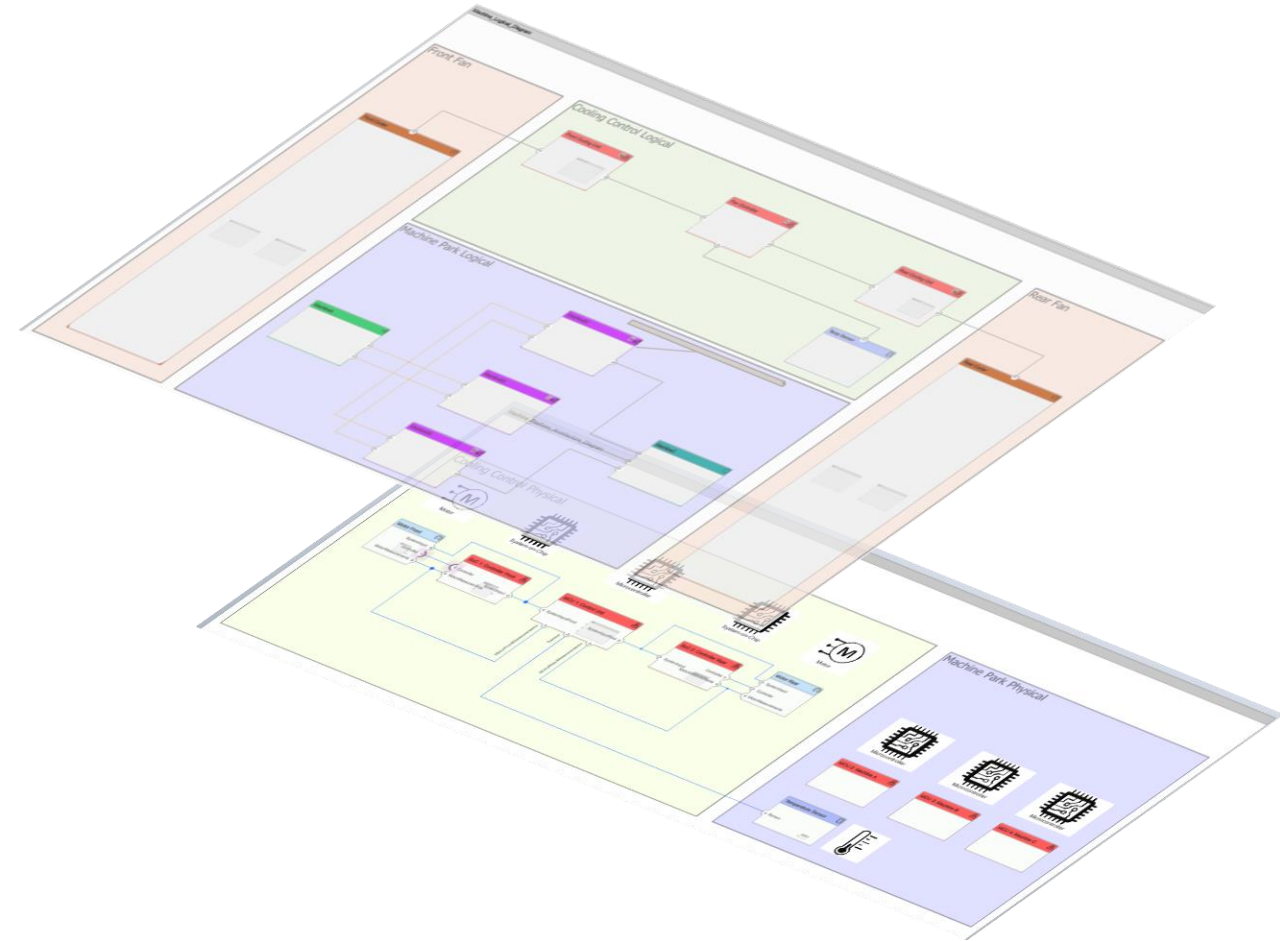
WHO

WHERE

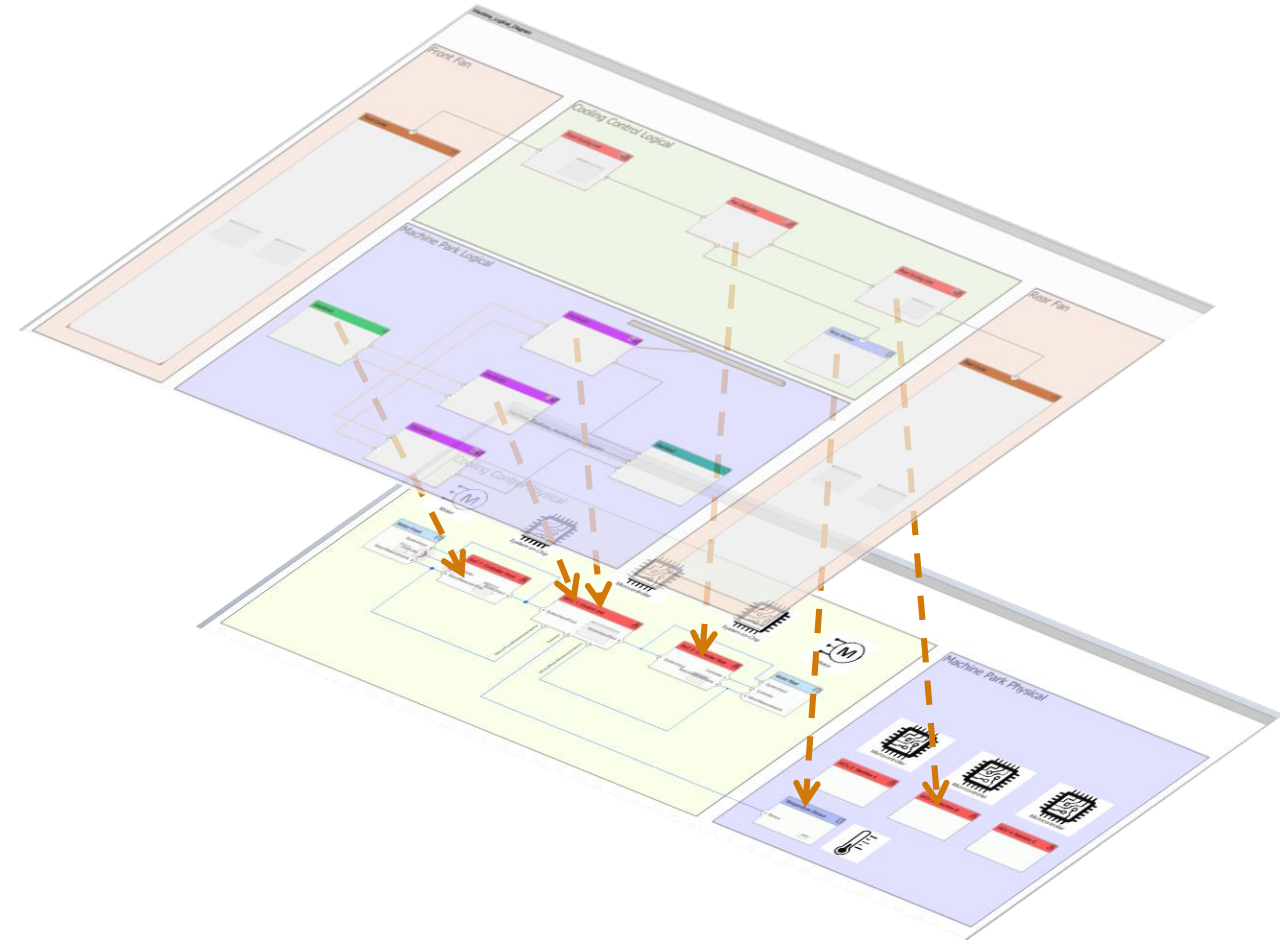




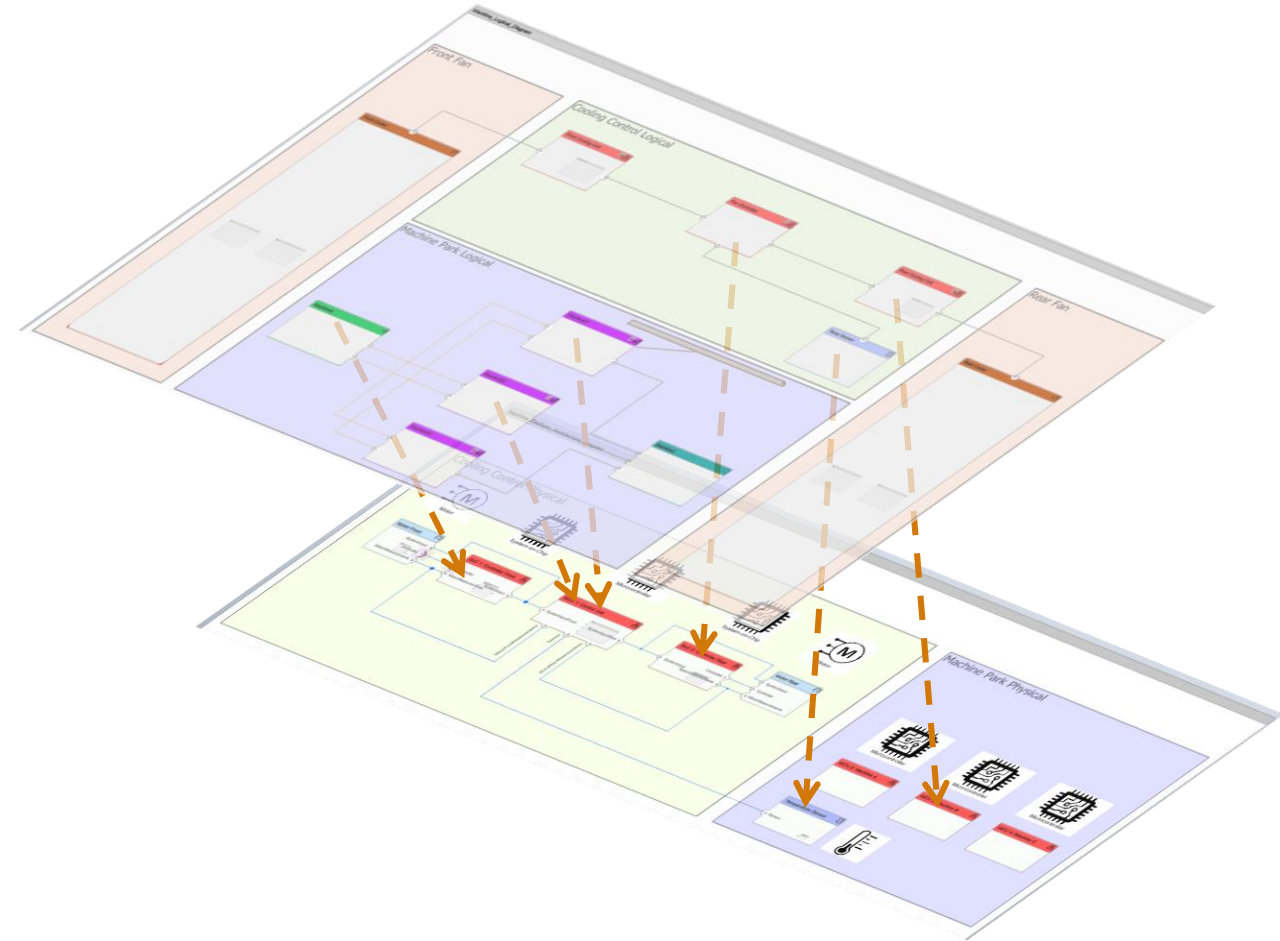
# Allocating between Architectures



# Allocating between Architectures



# Allocating between Architectures



**Allocation Editor**

ALLOCATIONS

Allocated
  Un-Allocated

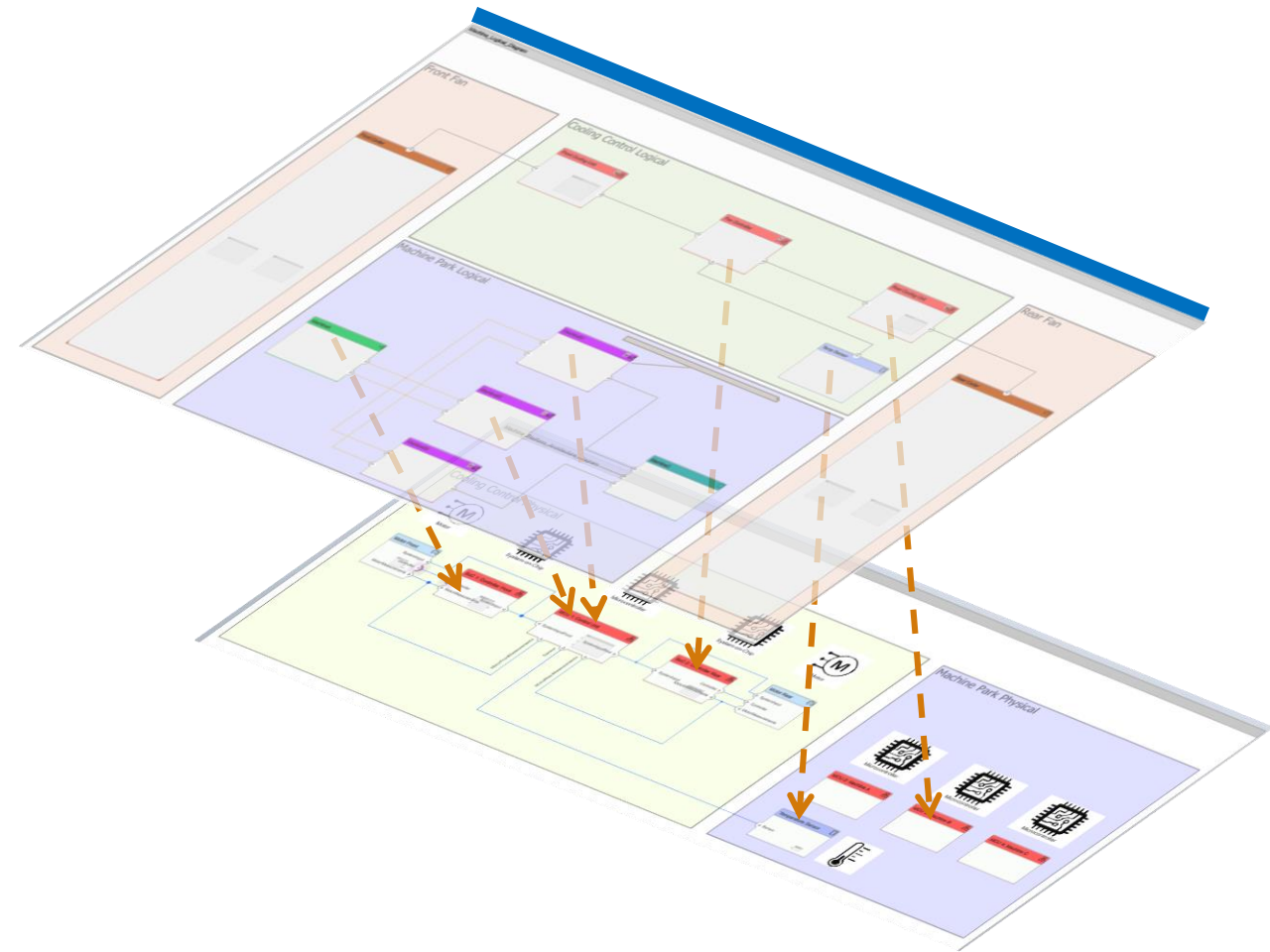
FILE SCENARIO REFRESH ROW FILTER

ALLOCATION SET BROWSER

- Scenario 1
- Functional\_to\_Logical
  - Scenario 1
- Logical\_to\_Physical
  - Scenario 1

	Machine_Platform_Arc	MCU 1: Control Un	SoC 2: Controller F	Motor Front	Motor Rear	SoC 1: Controller F	Temperature Sensc	MCU 4: Machine C	MCU 2: Machine A	MCU 3: Machine B
Machine_Logical_Diagram										
Rear Cooling Unit			↑							
MachineB1										↑
Fan Controller			↑							
MachineB3										↑
MachineA									↑	
Front Cooling Unit						↑				
Front Cooler										
Motor				↑						
MachineB2										↑
MachineC									↑	
Rear Cooler										↑
Motor				↑						
Temp Sensor							↑			

# Allocating between Architectures



**Allocation Editor**

ALLOCATIONS

Allocated
  Un-Allocated

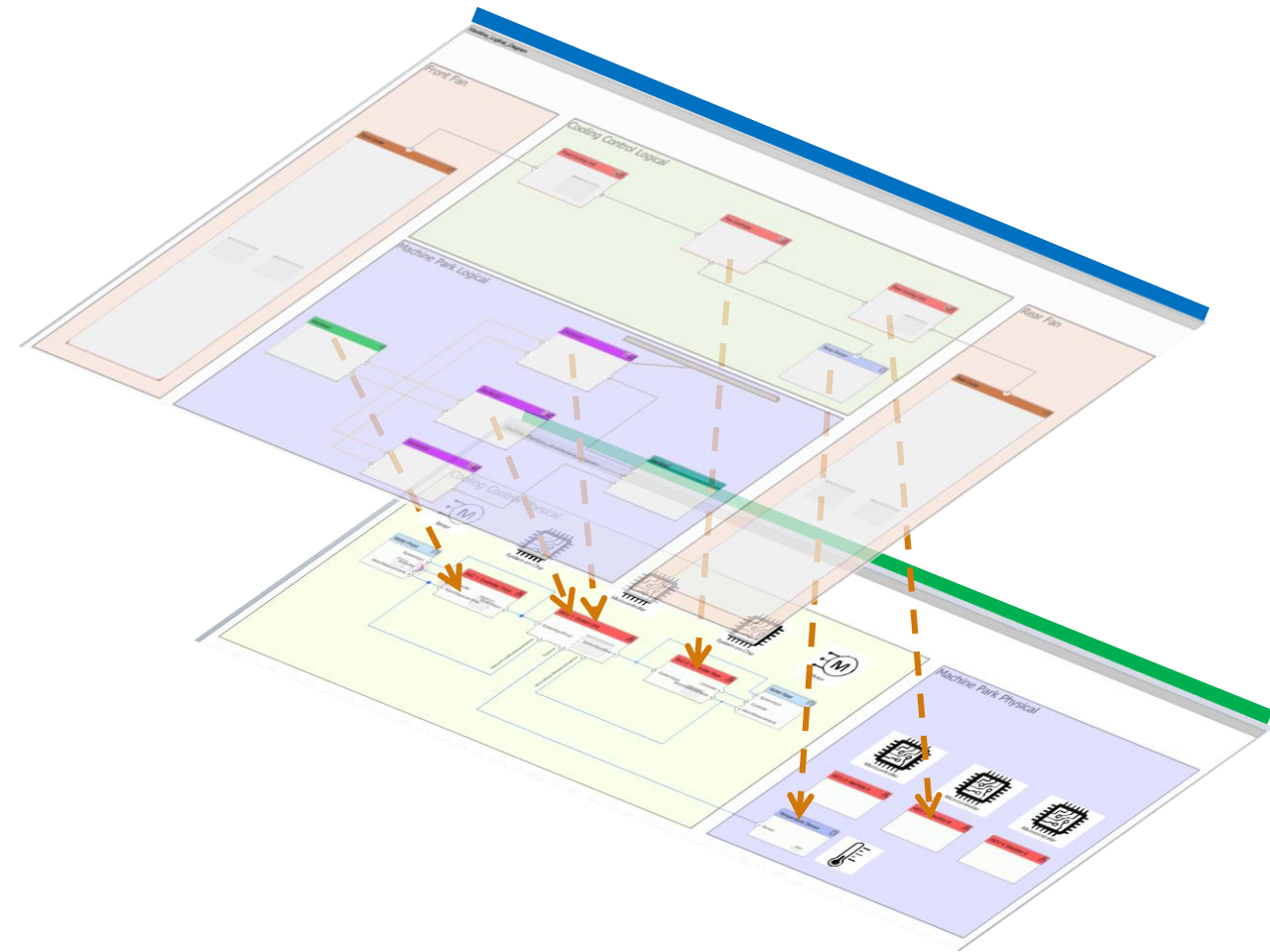
FILE SCENARIO REFRESH ROW FILTER

ALLOCATION SET BROWSER

- Scenario 1
- Functional\_to\_Logical
  - Scenario 1
- Logical\_to\_Physical
  - Scenario 1

	Machine_Platform_Arc	MCU 1: Control Un	SoC 2: Controller F	Motor Front	Motor Rear	SoC 1: Controller F	Temperature Sensc	MCU 4: Machine C	MCU 2: Machine A	MCU 3: Machine B
Machine_Logical_Diagram										
Rear Cooling Unit			↑							
MachineB1										↑
Fan Controller			↑							
MachineB3										↑
MachineA									↑	
Front Cooling Unit						↑				
Front Cooler										
Motor				↑						
MachineB2										↑
MachineC									↑	
Rear Cooler										
Motor					↑					
Temp Sensor							↑			

# Allocating between Architectures



Allocation Editor

ALLOCATIONS

Allocated
  Un-Allocated

FILE SCENARIO REFRESH ROW FILTER

ALLOCATION SET BROWSER

- Functional\_to\_Logical
  - Scenario 1
- Logical\_to\_Physical
  - Scenario 1

Component	Machine_Platform_Arc	MCU 1: Control Un	SoC 2: Controller F	Motor Front	Motor Rear	SoC 1: Controller F	Temperature Sensc	MCU 4: Machine C	MCU 2: Machine A	MCU 3: Machine B
Machine_Logical_Diagram										
Rear Cooling Unit					↑					
MachineB1										↑
Fan Controller				↑						
MachineB3										↑
MachineA									↑	
Front Cooling Unit							↑			
Front Cooler										
Motor				↑						
MachineB2										↑
MachineC									↑	
Rear Cooler										↑
Motor					↑					
Temp Sensor							↑			

Machine\_Functional\_Diagram \* - Simulink

SIMULATION    DEBUG    MODELING    FORMAT    APPS

Find    Compare    Environment    Interface Editor    Import base workspace    Import MAT-file    Import    Apply Stereotypes    Compose...    Reference Compose...    Variant Compose...    Architecture Views    Analysis Model    Allocation Editor    Update Model    Stop Time 10.0    Normal    Run    Stop    Fast Restart

MANAGE    DESIGN    PROFILES    COMPONENT    VIEWS    SIMULATE

Model Browser

Machine\_Functional\_Diagram

# Machine\_Functional\_Diagram

## Machine Function

Process Flow

Process Flow to Product

Package Product

Check if Machine is safe

## Cooling Function

Sense Temperature

Determine if cooling is needed

Turn machine on or off

Apply cooling

Check if Cooling is safe

Property Inspector

Architecture

Architecture    Info

NAME	VALUE
▼ Main	
Name	Machine_Functional_Diagram
Stereotype	Add. ▼

Interfaces

Ready    287%    VariableStepAuto

# Assess Different Allocation Scenarios Quantitatively

## Allocate Architectures in a Tire Pressure Monitoring System

R2021a

This example shows how to use allocations to analyze a tire pressure monitoring system.

[View MATLAB Command](#)

### Overview

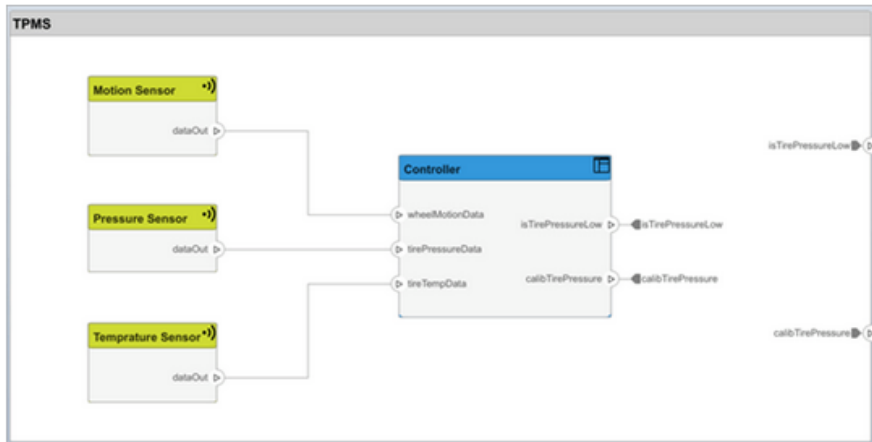
In systems engineering, it is common to describe a system at different levels of abstraction. For example, you can describe a system not have any behavior associated with them but most likely trace back to some operating requirements the system must fulfill. We refer to this as *architecture*. In this example, an automobile tire pressure monitoring system is described in three different architectures:

1. Functional Architecture — Describes the system in terms of its high-level functions. The connections show dependencies between functions.
2. Logical Architecture — Describes the system in terms of its logical components and how data is exchanged between them. Add simulation.
3. Platform Architecture — Describes the physical hardware needed for the system at a high level.

The allocation process is defined as linking these three architectures that fully describe the system. The linking captures the information accessible to the others.

Use this command to open the project.

scExampleTirePressureMonitorSystem

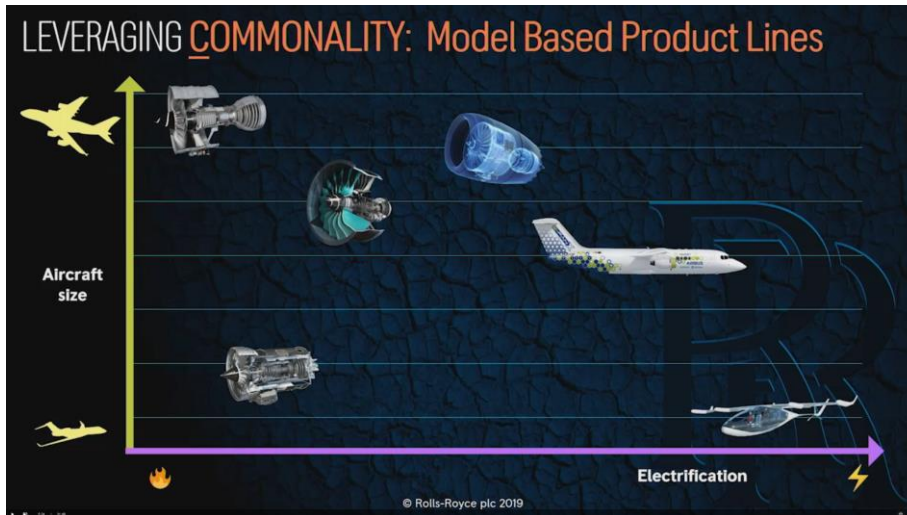


	Supplier A	Supplier B	Supplier C	Supplier D
Report Low Tire Pressure	1	0	0	0
Measure pressure on tire	0	0	1	0
Calculate Tire Pressure	0	1	0	0
Measure temprature of tire	0	0	0	1
Measure rotations	0	1	0	0
Calculate if pressure is low	1	0	0	0
Report Tire Pressure Levels	1	0	0	0
Measure Tire Pressure	0	0	0	0

	Scenario 1	Scenario 2
Front ECUMemory Used (MB)	110	90
Front ECU Memory (MB)	100	100
Front ECU Overloaded	1	0
Rear ECU Memory Used (MB)	0	20
Rear ECU Memory (MB)	100	100
Rear ECU Overloaded	0	0

# Who is doing Model Based Systems Engineering with MathWorks tools?

Rolls Royce, UK Expo, Oct 2019



<https://www.mathworks.com/videos/our-journey-towards-model-based-product-lines-1573233985120.html>

## 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](mailto:chris.watkins@gulfstream.com)

Jerry Varghese  
Gulfstream Aerospace Corporation  
Savannah, GA, U.S.  
[jerry.varghese@gulfstream.com](mailto:jerry.varghese@gulfstream.com)

Michael Knight  
Gulfstream Aerospace Corporation  
Savannah, GA, U.S.  
[michael.knight@gulfstream.com](mailto:michael.knight@gulfstream.com)

Becky Petteys  
The MathWorks, Inc.  
Natick, Massachusetts, U.S.  
[bpetteys@mathworks.com](mailto:bpetteys@mathworks.com)

Jordan Ross  
The MathWorks, Inc.  
Natick, Massachusetts, U.S.  
[jordanr@mathworks.com](mailto: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 Systems 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

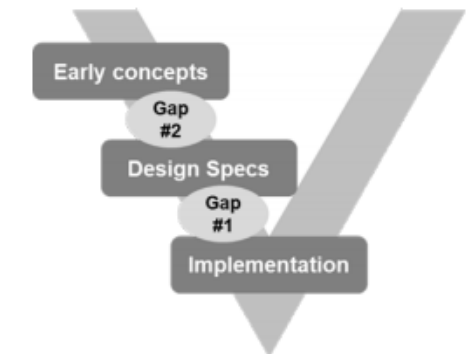


Figure 1: Simplified development process

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



# Who is doing Model Based Systems Engineering with MathWorks tools?

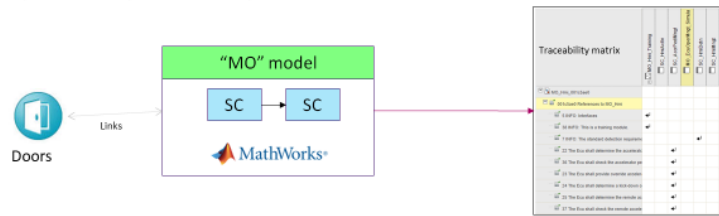


## MathWorks Automotive Conference 2021

Felix Raab, Bosch

### Automotive SPICE With Model Based Systems Engineering SYS.3 BP2 allocation, BP6 traceability and BP7 consistency of system requirements

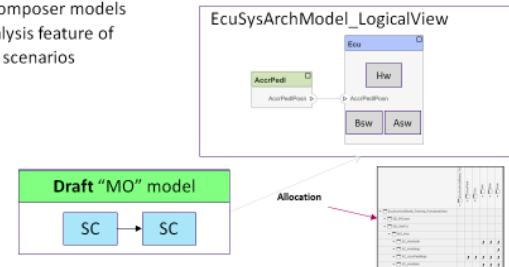
- ▶ Bidirectional links to DOORS with help of the Simulink requirement toolbox.
- ▶ Easy take over of requirements allocations between projects, as requirements are included in the architecture import/export.
- ▶ Consistency is checked by the traceability matrix.



7 Internal | Powertrain Solutions | FS-60102-01 | 2021-03-13  
© 2021 MathWorks. All rights reserved. See [https://www.mathworks.com/help/req/ug/requirements-management.html](#) for more information.

### Automotive SPICE With Model Based Systems Engineering SYS.3 BP5 Evaluate alternative system architectures

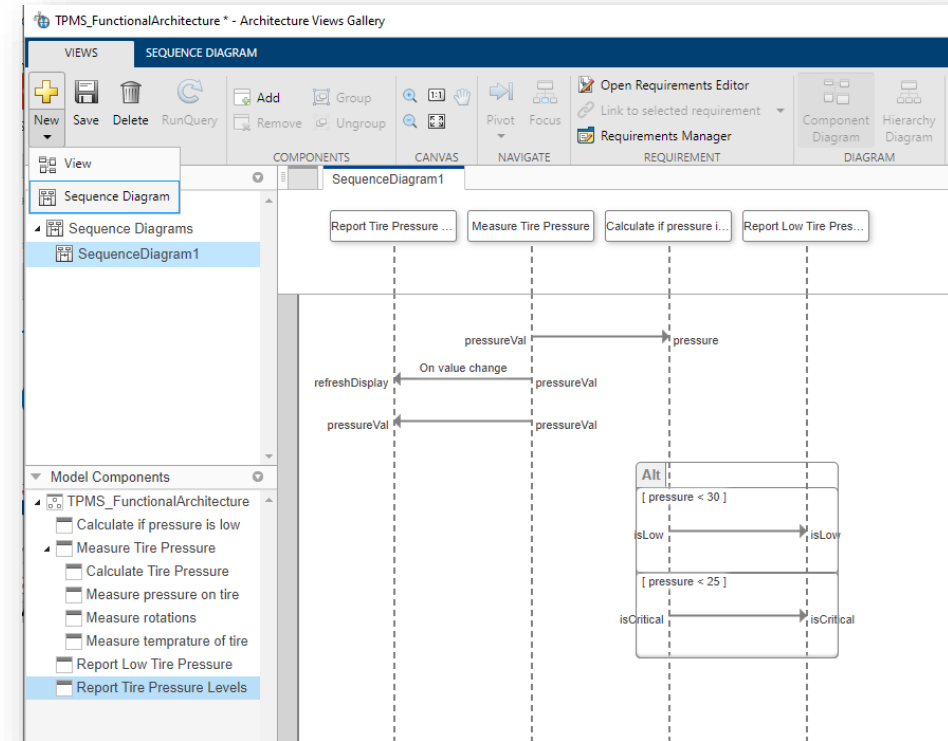
- ▶ For the evaluation of alternative system architectures a draft model or analysis is created based on the respective use case. In addition to the traditional tools like power point and Excel, System Composer models gain user acceptance. The model analysis feature of System Composer and the allocation scenarios strengthen this approach.



10 Internal | Powertrain Solutions | FS-60102-01 | 2021-03-13  
© 2021 MathWorks. All rights reserved. See [https://www.mathworks.com/help/req/ug/requirements-management.html](#) for more information.

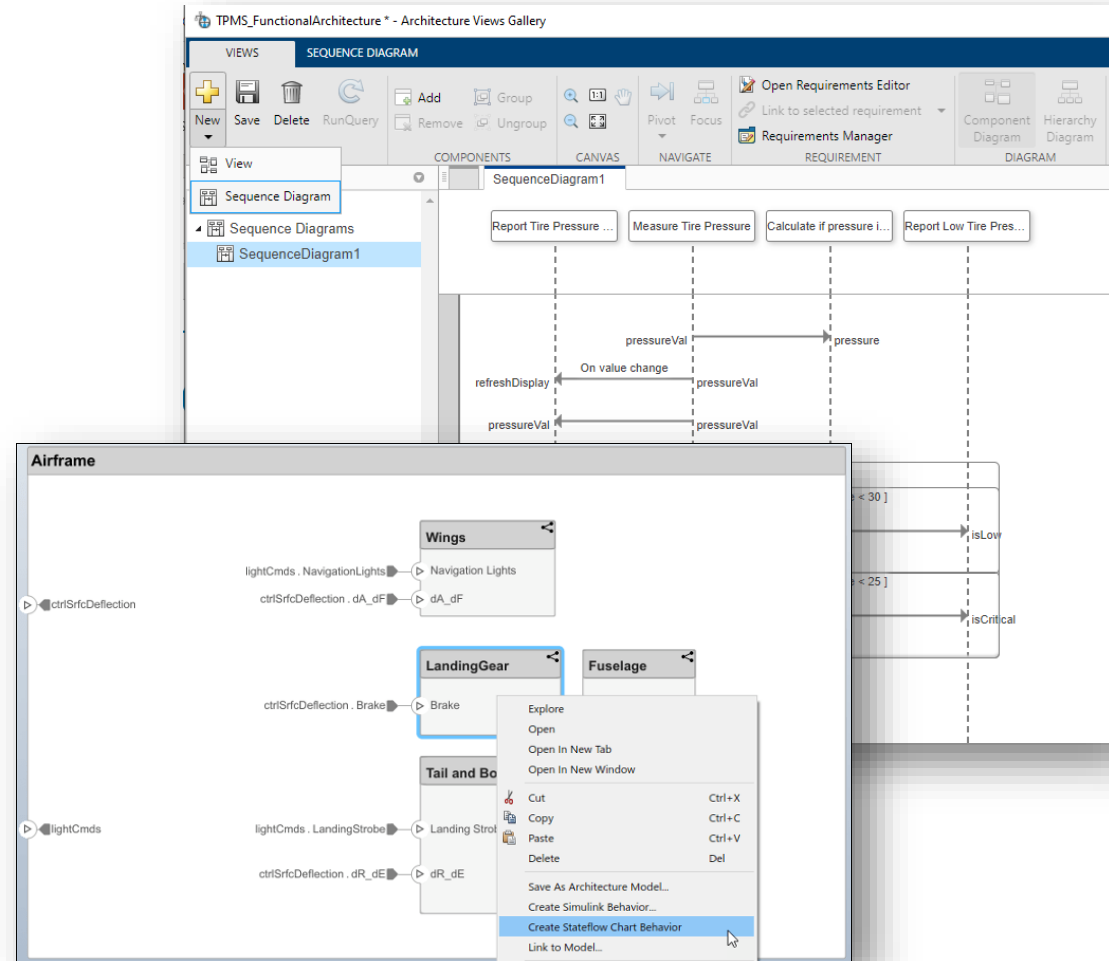
# New Features in R2021a

- System Composer
  - Sequence diagrams



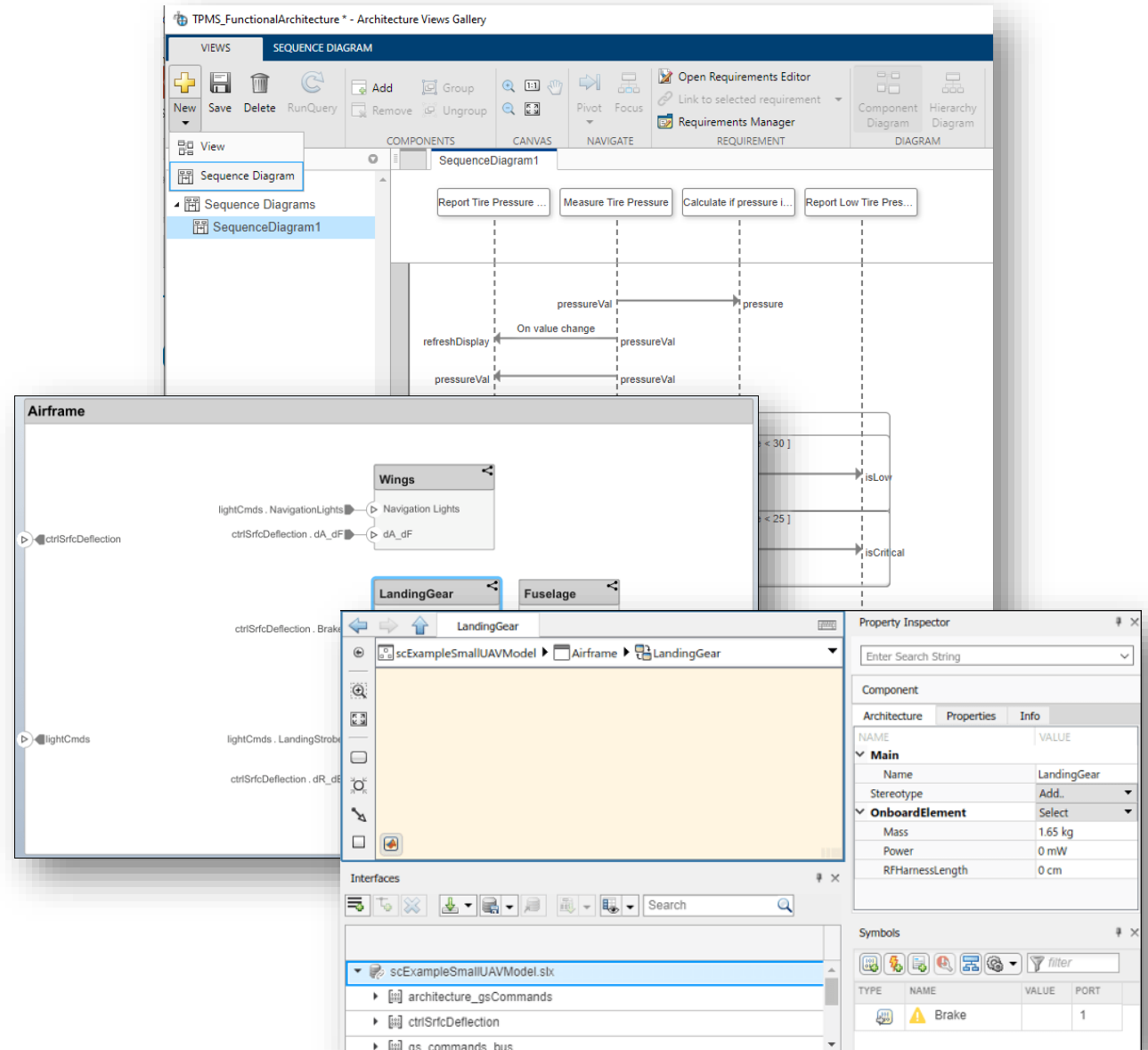
# New Features in R2021a

- System Composer
  - Sequence diagrams
  - Stateflow charts in components



# New Features in R2021a

- System Composer
  - Sequence diagrams
  - Stateflow charts in components



# New Features in R2021a

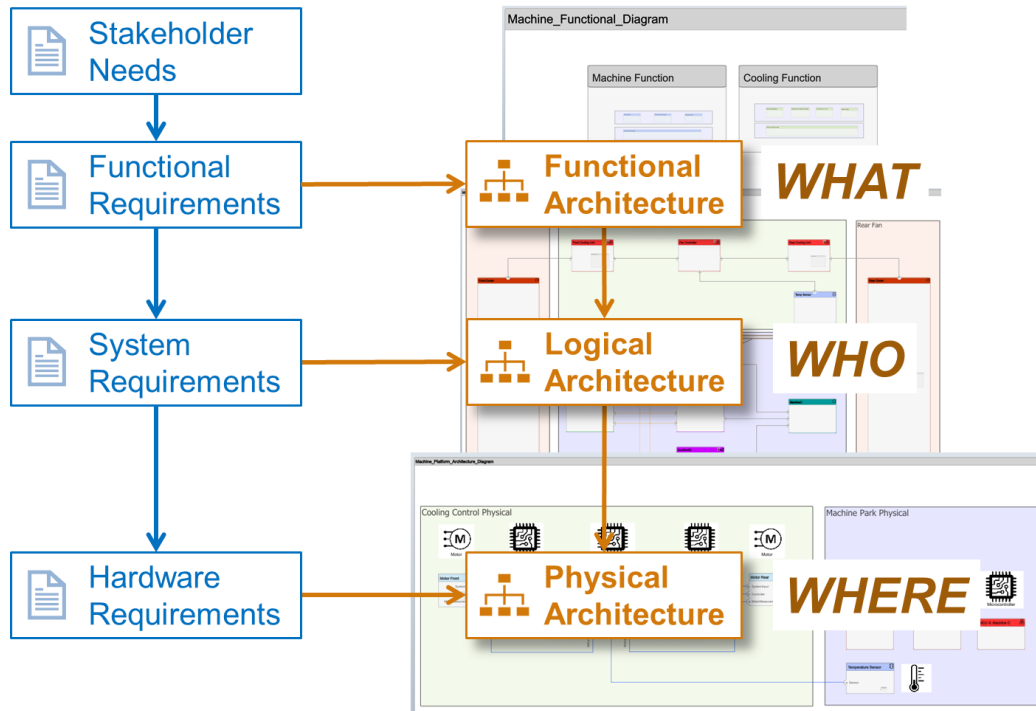
- System Composer
  - Sequence diagrams
  - Stateflow charts in components
  - Software architectures
  
- Simulink Requirements
  - Editor improvements
  - Multi-artifact traceability matrix

The image displays several key features of MATLAB R2021a:

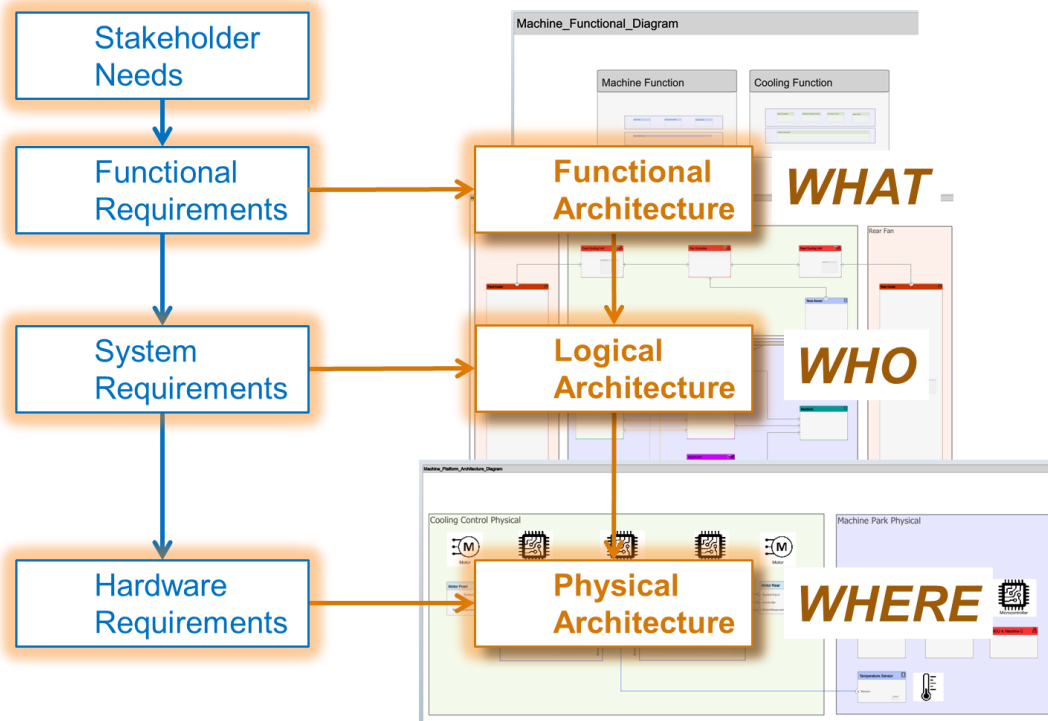
- Sequence Diagram Editor:** A screenshot of the 'SEQUENCE DIAGRAM' editor showing a toolbar with options like 'Add', 'Group', 'Remove', and 'Ungroup'. Below the toolbar is a sequence diagram with lifelines and messages.
- Software Architecture Model:** A screenshot of the 'Airframe' model showing a hierarchical structure of components: 'Wings' (containing 'Navigation Lights'), 'LandingGear', and 'Fuselage'. Connections are shown between 'ctrlSrfcDeflection' and 'dA\_dF'.
- Requirements Traceability Matrix:** A screenshot of the 'MyRequirements' matrix showing a table of requirements and their status.
 

Requirement	Machine_Functional_Diagram	Machine_Logical_Diagram	Machine_Platform_Architecture
ReqSys1 Functional Requirement	○	○	○
ReqSys2 System Requirement	○	○	○
ReqSys3 Hardware Requirement	○	○	○
ReqSys4 Safety Requirement	○	○	○
- System Composer:** A screenshot showing the 'System Composer' interface with 'Architecture Model' and 'Software Architecture Model' tabs. The 'Software Architecture Model' tab is active, showing a diagram of a variant component with multiple input and output buses.
- Component Properties:** A screenshot of the 'Properties' window for a 'LandingGear' component, showing details like 'Name', 'Stereotype', 'Mass', 'Power', and 'RFHarnessLength'.

# Key Takeaways

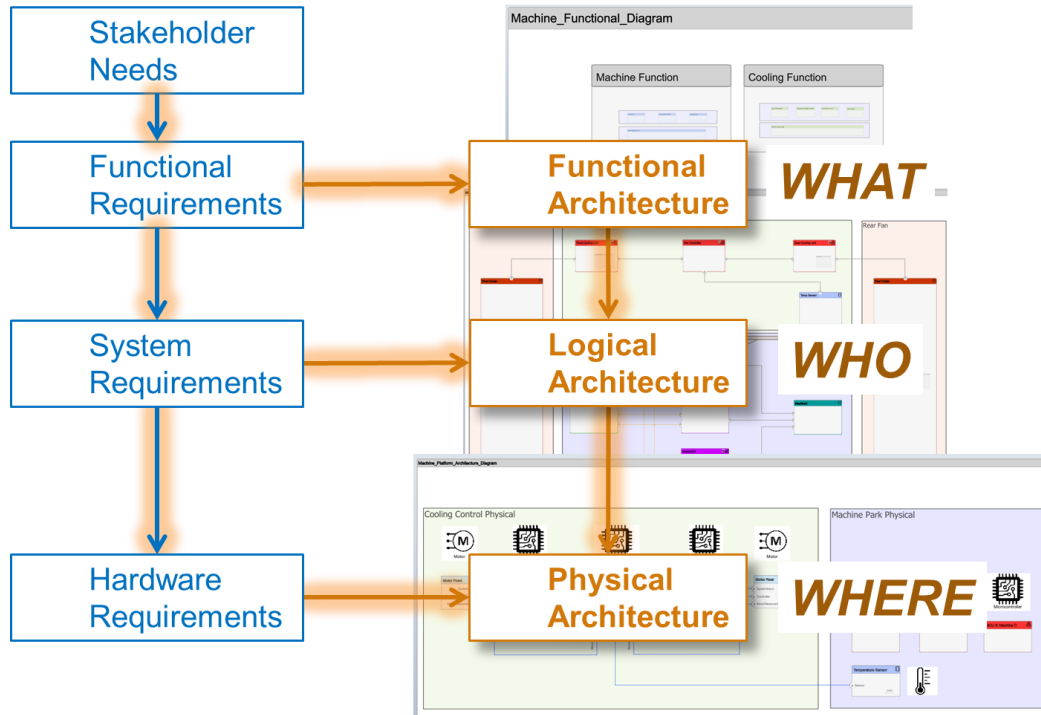


# Key Takeaways



- You can import, write, and store textual requirements right in the same environment as your architecture and design models.

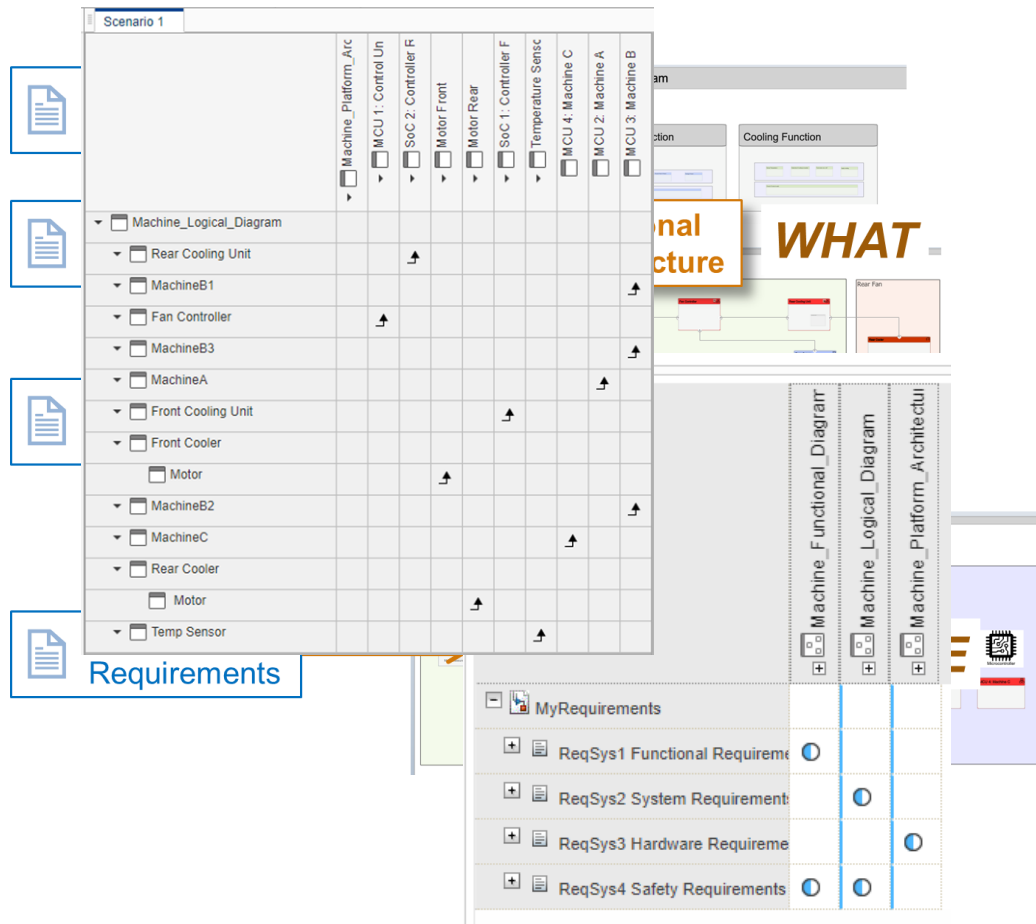
# Key Takeaways



- You can import, write, and store textual requirements right in the same environment as your architecture and design models.
- You can establish relationships among multiple requirements and architecture artifacts to understand the impact of changes.

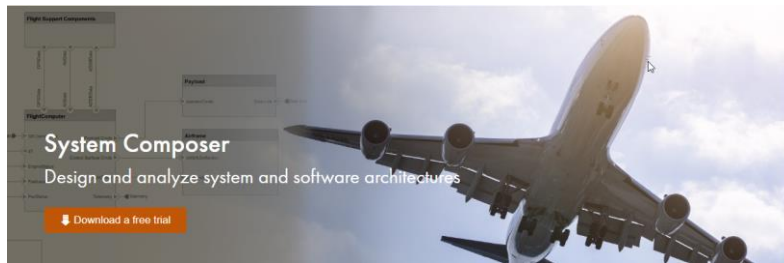


# Key Takeaways

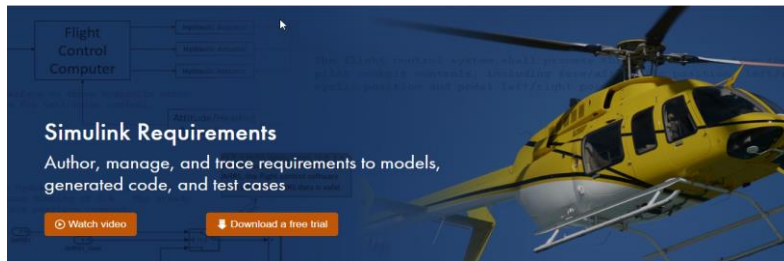


- You can import, write, and store textual requirements right in the same environment as your architecture and design models.
- You can establish relationships among multiple requirements and architecture artifacts to understand the impact of changes.
- You can visualize those relationships to assess the completeness of your system.

# Learn More



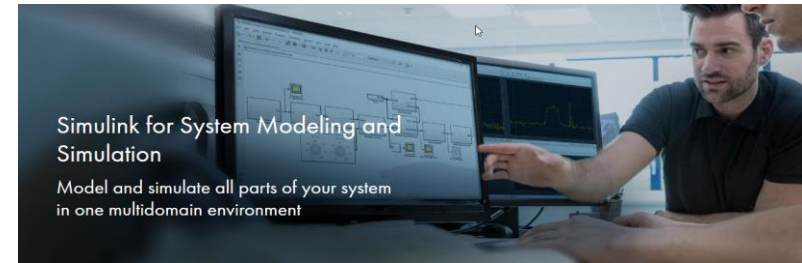
[System Composer](#)



[Simulink Requirements](#)



[Simulink Test](#)



[Model-Based Systems Engineering](#)



[System Modeling and Simulation](#)



[AUTOSAR](#)

# MATLAB EXPO 2021

Thank you

