MATLAB EXPO 2021

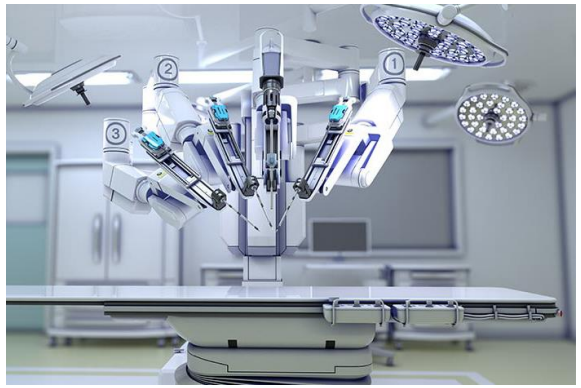
Developing Embedded Software with Model-Based Design to Meet Certification Standards

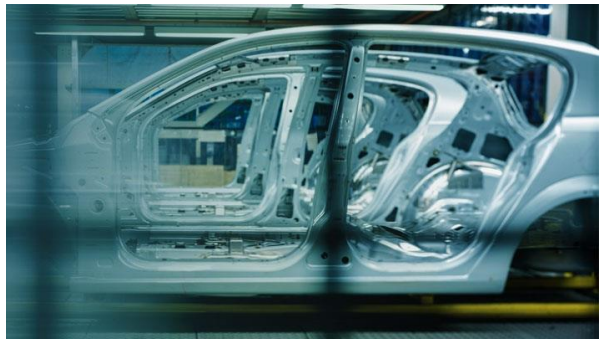
Paul Urban *Jeff Harper*

MathWorks®

Software safety is important across industries

- Software errors are common
  - Automotive: **19%** of all recalls
  - Medical devices: **26%** of all recalls
- Recalls are extremely expensive
  - MCAS failure: **$18B+**
  - Broken hip replacement: **$3B+**
  - Faulty pedals: **$3B+**
  - Airbag issues: **$25B+**



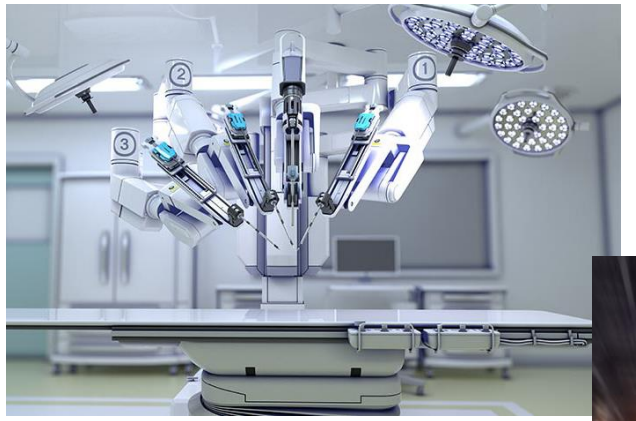


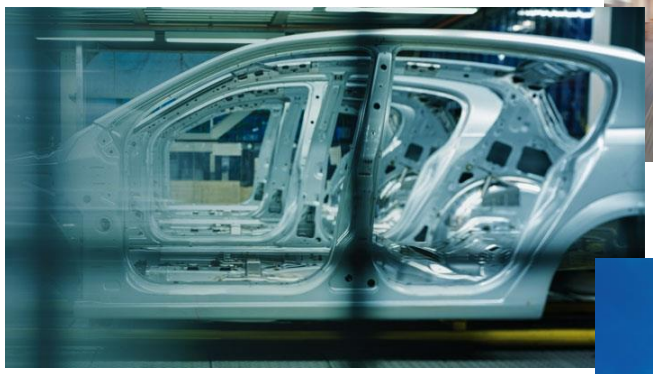
Source: MedTech Dive, Ars Technica, Automotive News, Allianz, Sibros, Medical Design, New York Times, Bloomberg

Why functional safety?

Poll: Safety standards across different industries

- **IEC 61508** – All industries
- **ISO 26262, SOTIF** – Automotive
- **DO 178C** – Avionics
- **IEC 62304** – Medical
- **EN 50128** – Rail
- **ISO 25119** – Agriculture and Forestry
- **IEC 61511** – Process Control
- And many others…..



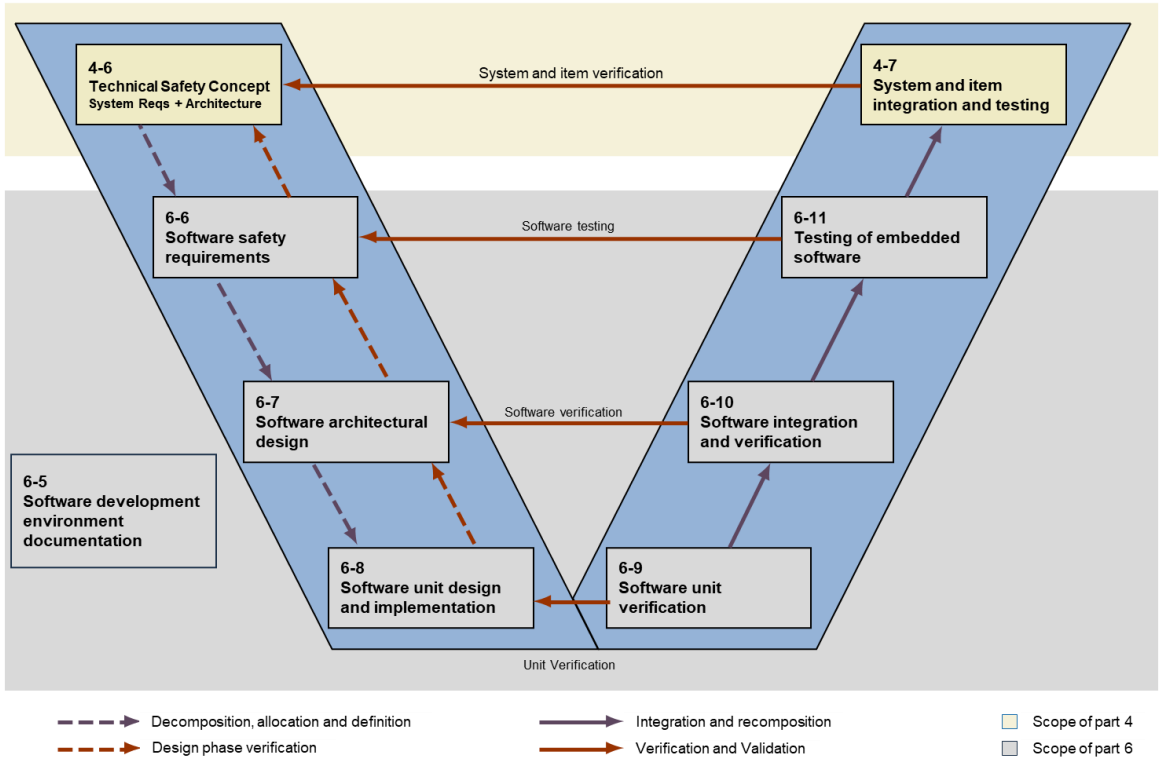


Key Takeaways

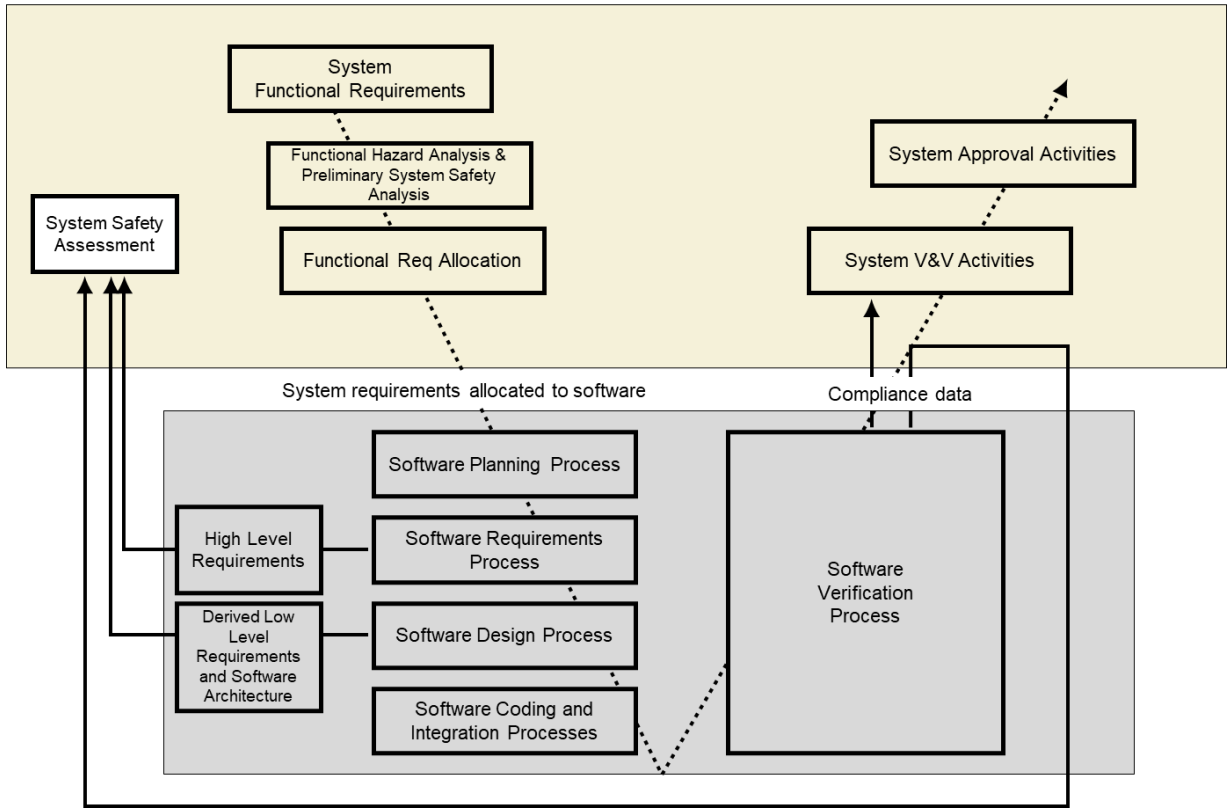
Use Reference Workflow with Model-Based Design to Meet Certification Standards

- Create **traceability** across requirements, architecture, design, test and code
- Detect errors **earlier** by continuous testing
- **Reduce coding errors** with automatic code generation
- **Automate** generation of documents and reports for reviews and audits

V-Model for Certifiable Product Development

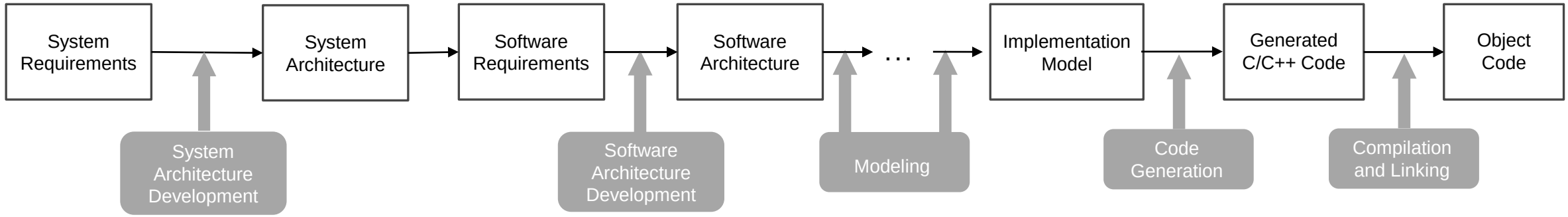


ISO 26262: 2018



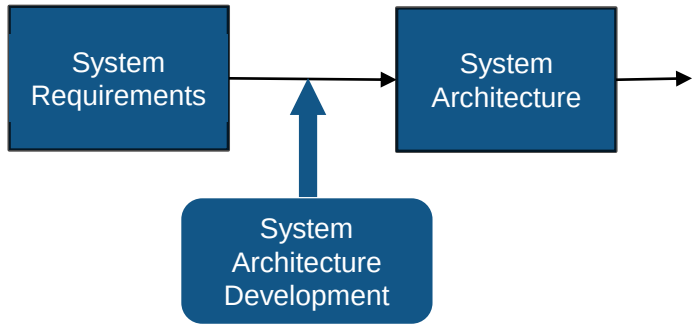
DO 178C, ARP 4754A, ARP 4761

Reference Workflow for Certification



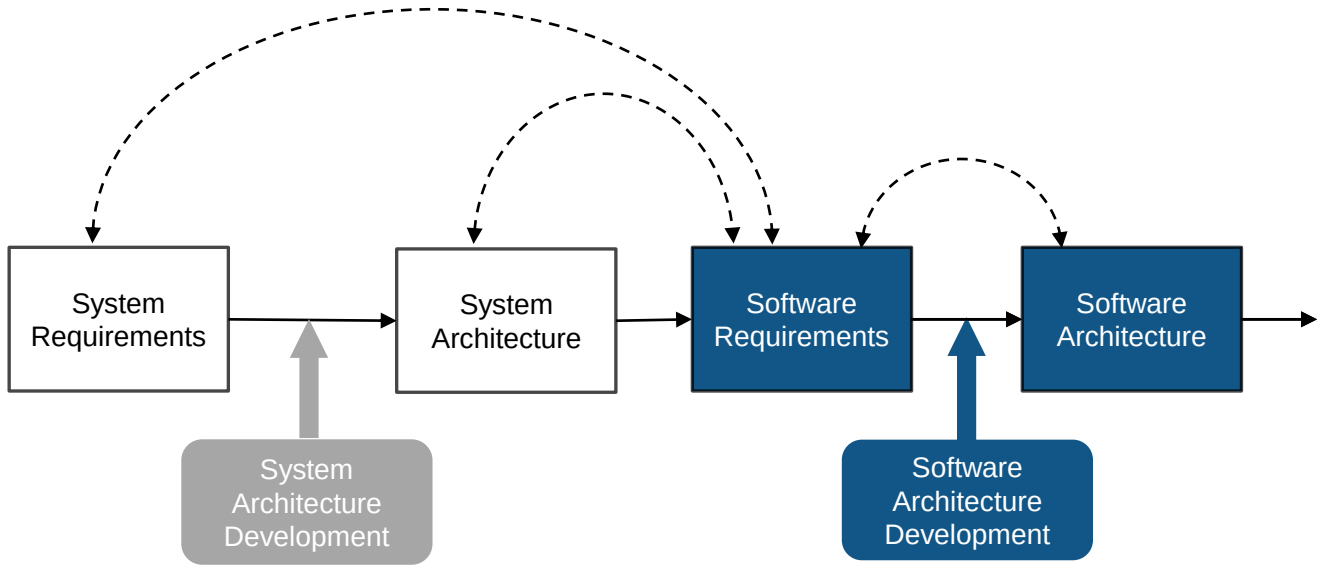
System Level **Software Level**

Reference Workflow for Certification – System Requirements and Architecture



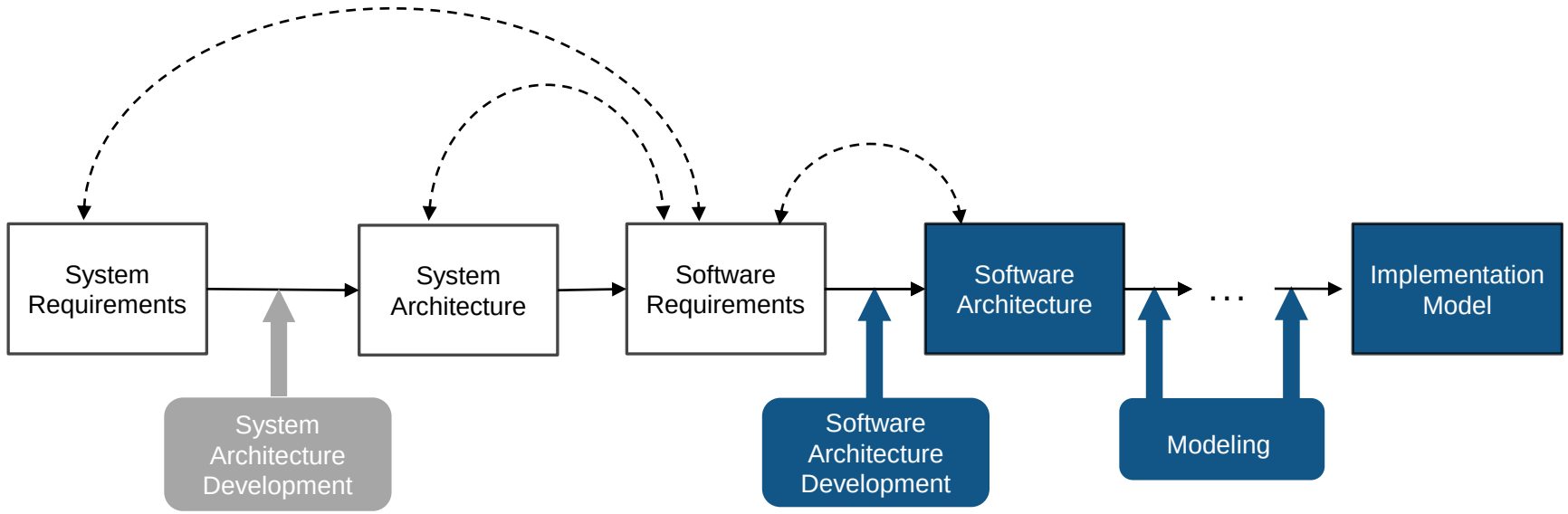
System Level

Reference Workflow for Certification – Software Requirements and Architecture



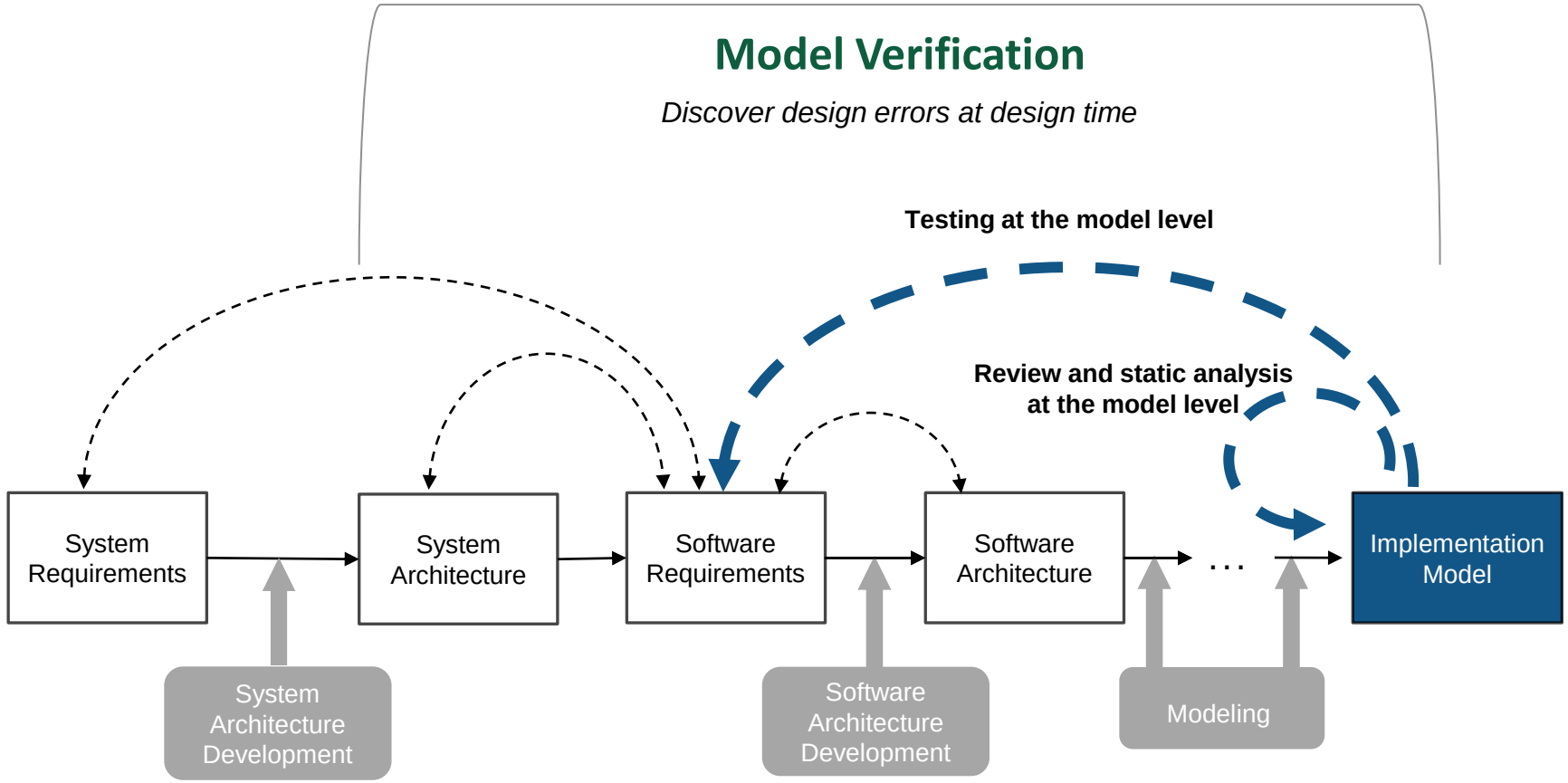
System Level | **Software Level**

Reference Workflow for Certification – Detailed Design

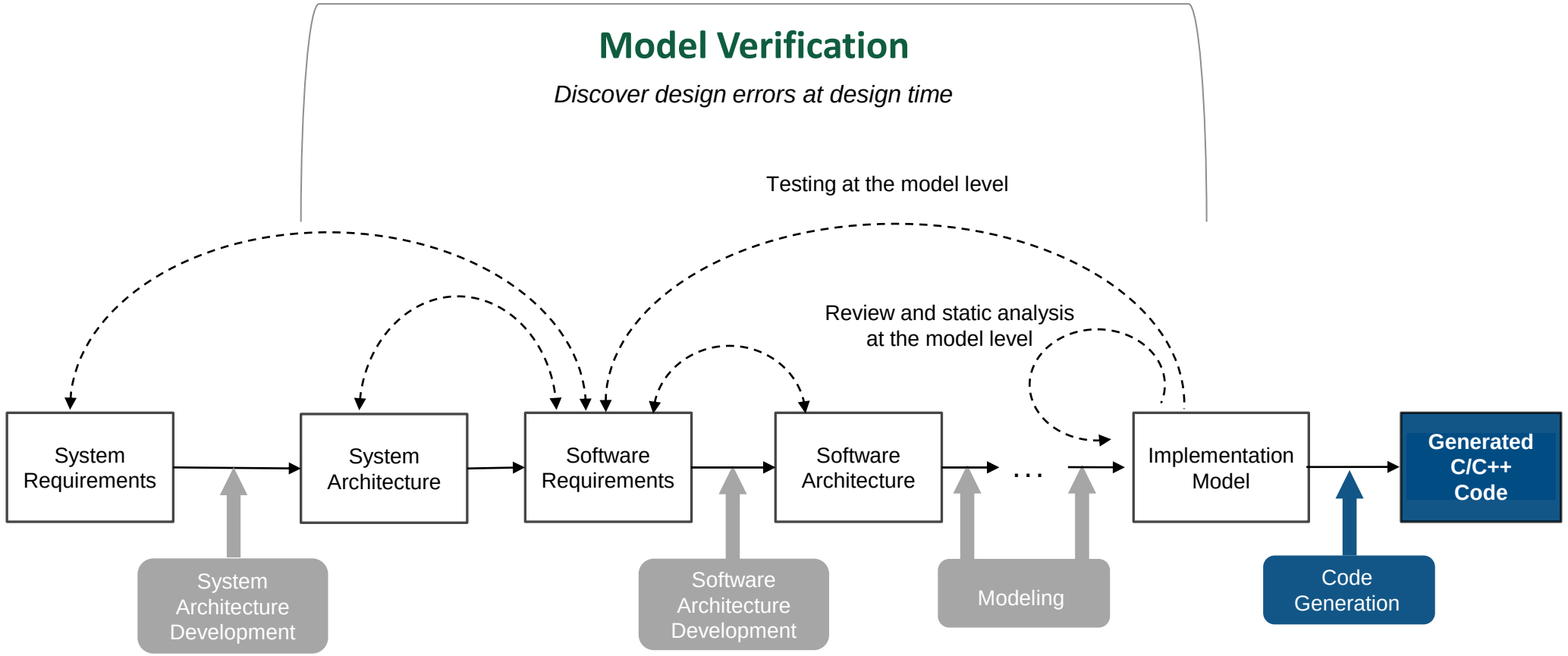


System Level | **Software Level**

Reference Workflow for Certification – Model Verification

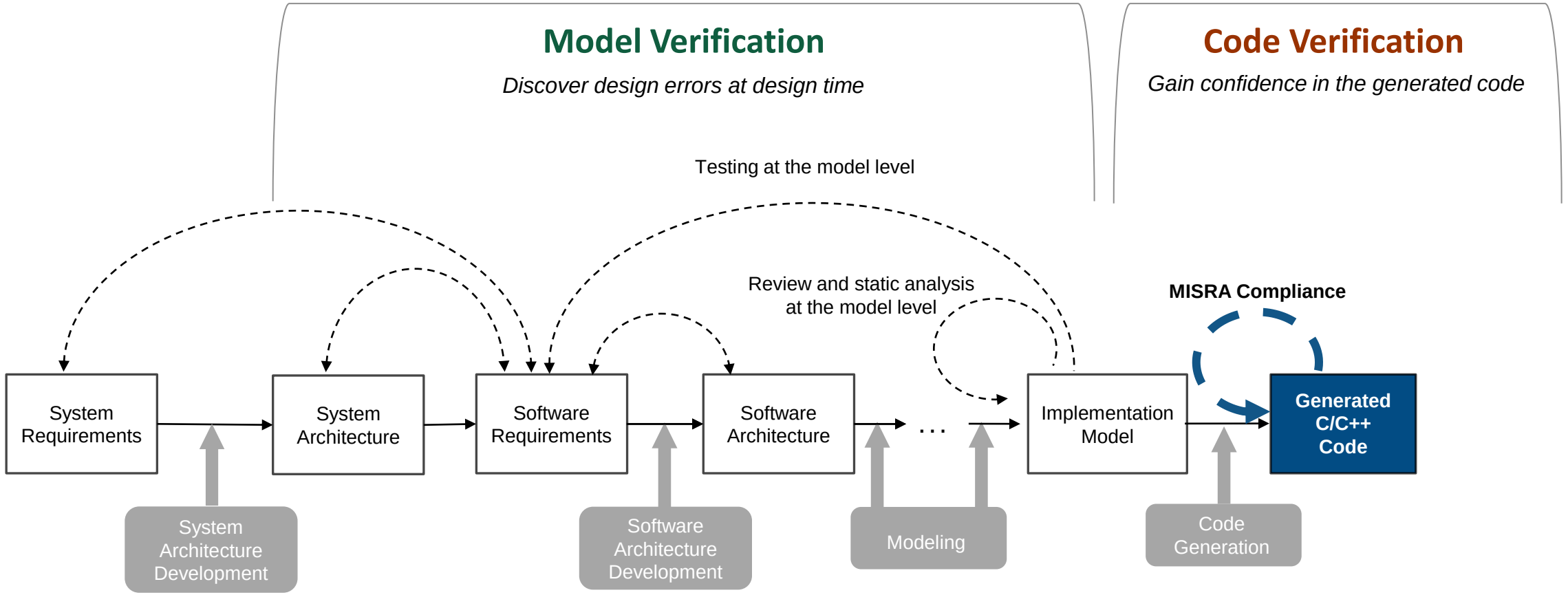


Reference Workflow for Certification – Code Generation

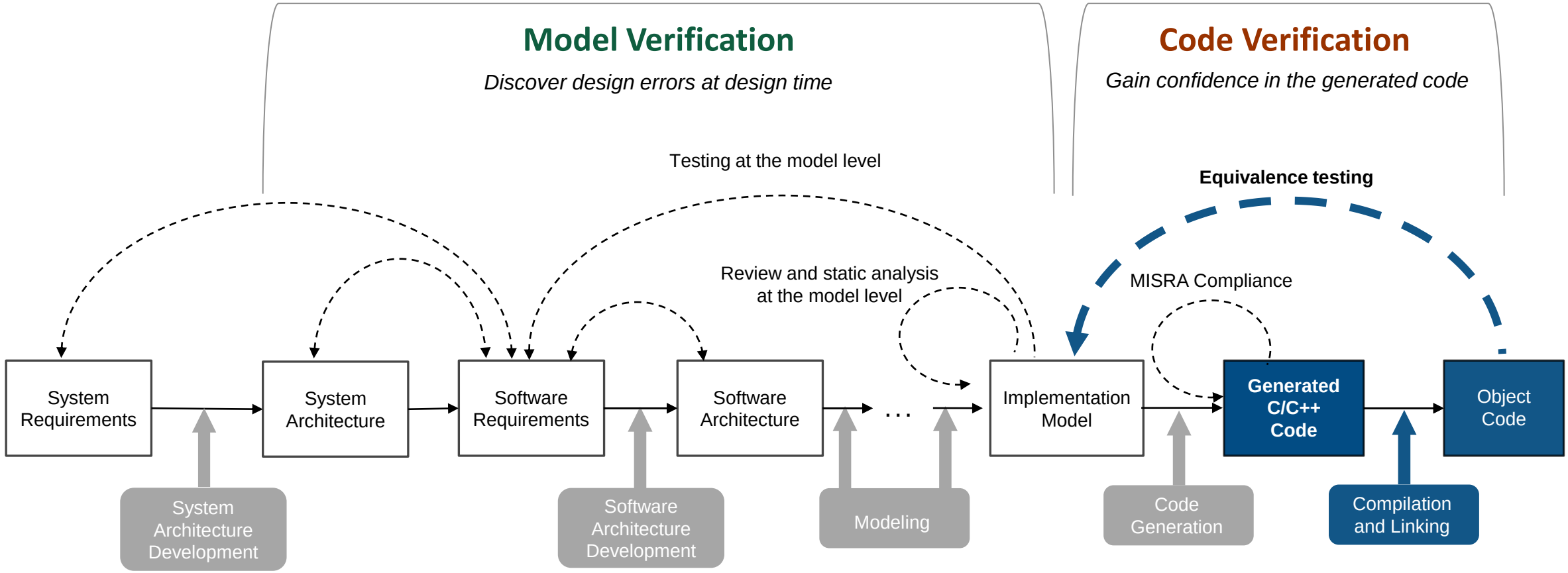


System Level | **Software Level**

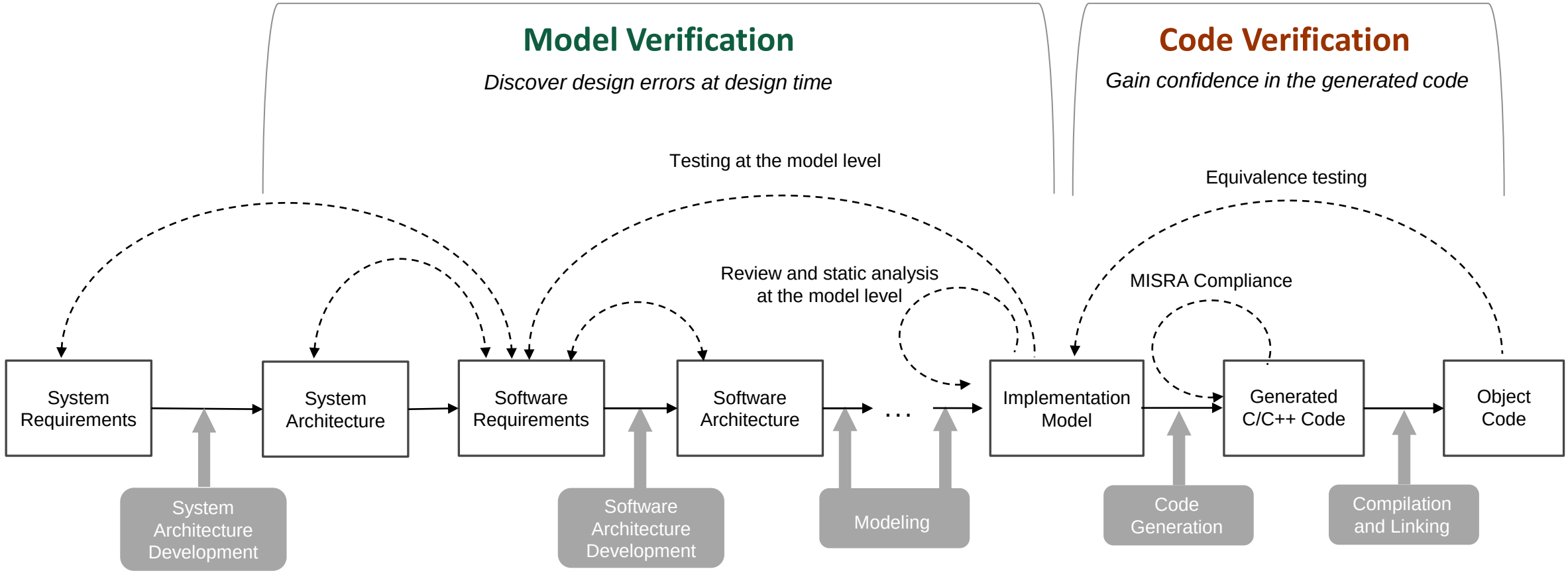
Reference Workflow for Certification – Static Code Verification



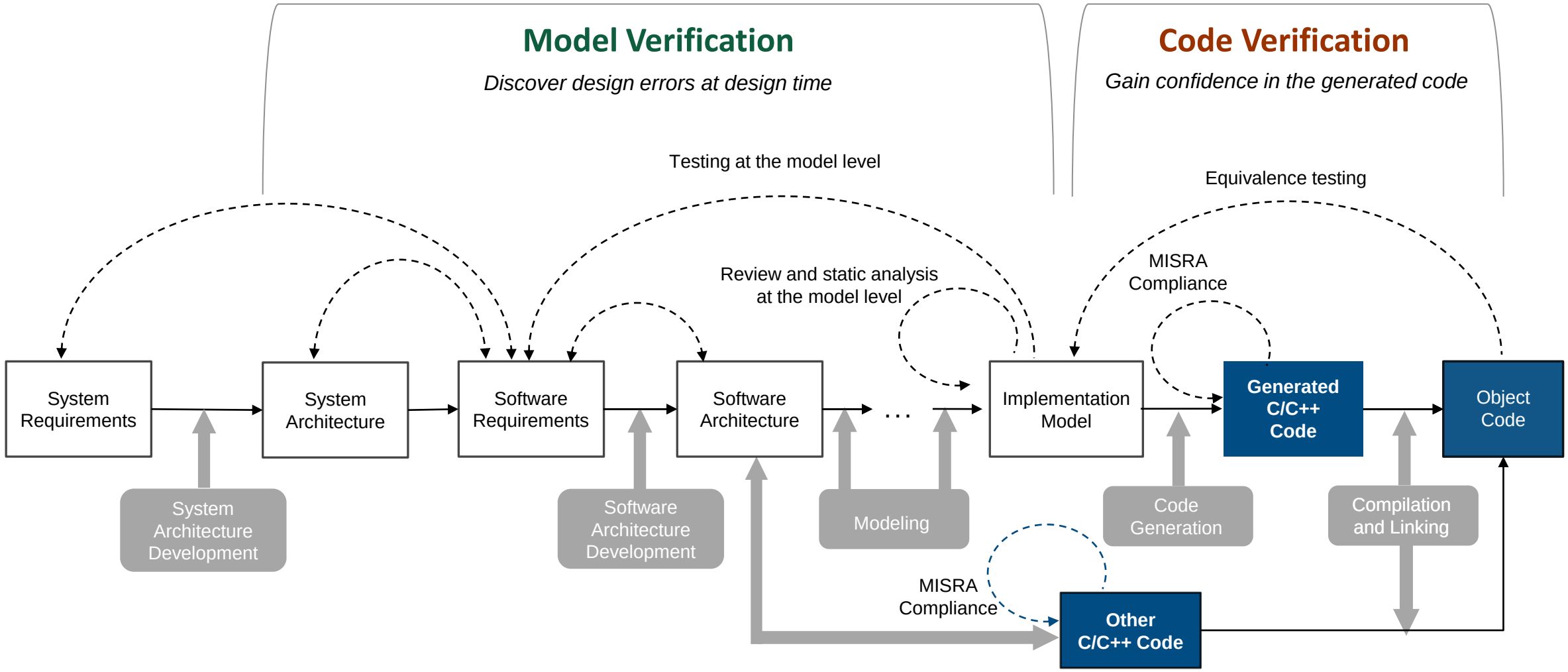
Reference Workflow for Certification – Dynamic Code Verification



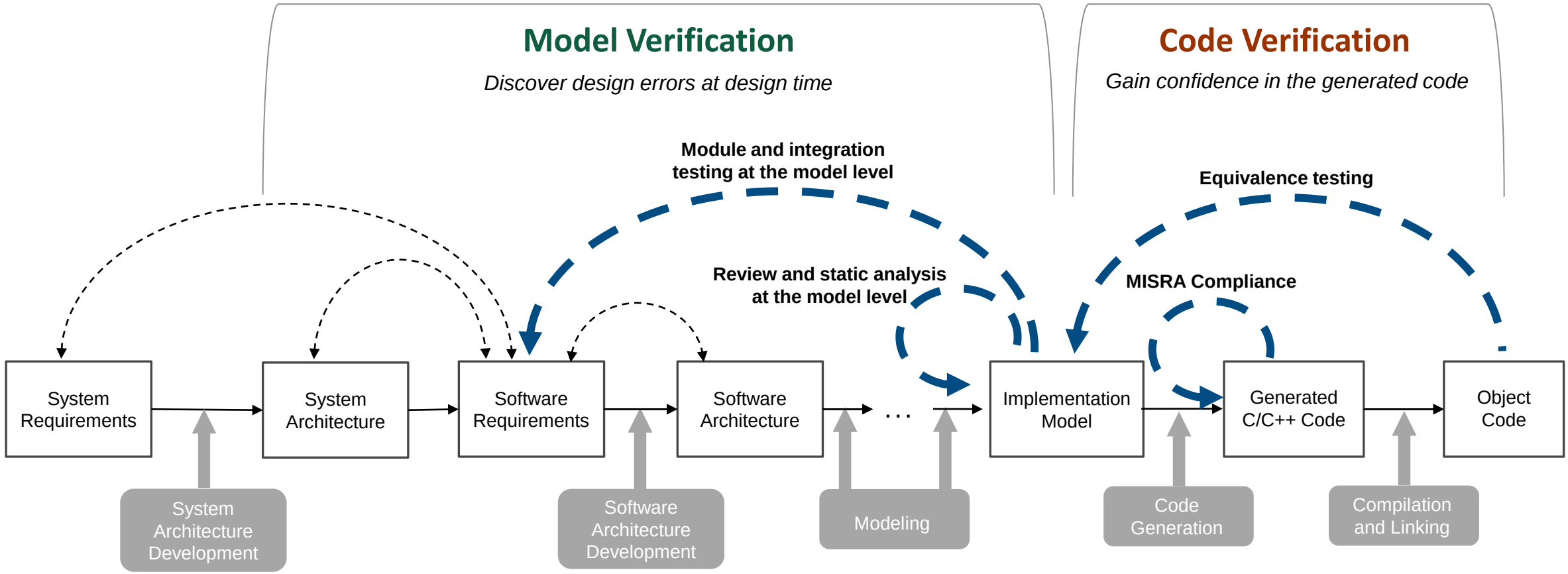
Reference Workflow for Certification – Atomic Components



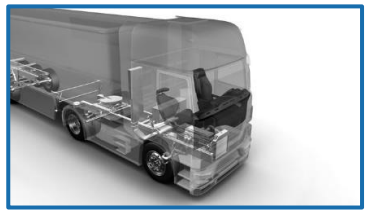
Reference Workflow for Certification – Static Code Verification



Reference Workflow for Certification – Integration and Verification



ROI Results



Continental

Verification time cut by up to 50 percent

Leonardo

Recertification cycle times reduced by more than 90%

Tessella

Models reused on follow-on projects, cutting design effort by up to 80%

Corindus

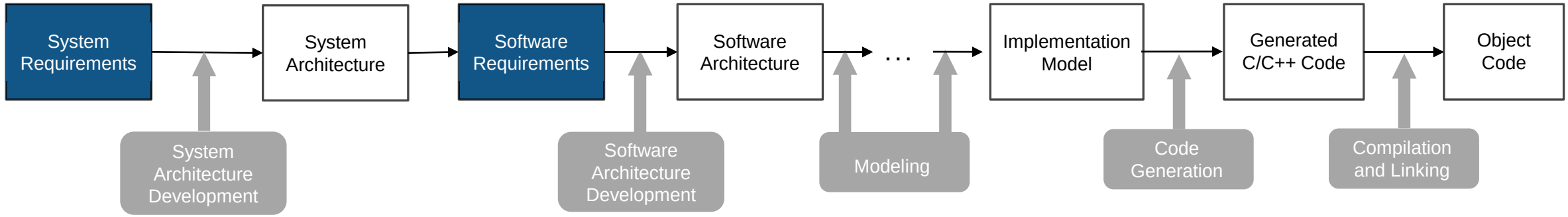
Development time halved and engineering effort reduced by 80%

More User Stories: www.mathworks.com/company/user_stories.html

Poll: What is your biggest challenge in verifying or testing your design? (choose the most important one)

- Tracing requirements to design and test
- Finding requirements errors late in development
- Meeting test coverage goals
- Reusing tests across design phases
- Performing system level validation
- Creating documentation and work products
- Performing manual reviews
- Other (Enter in Chat)

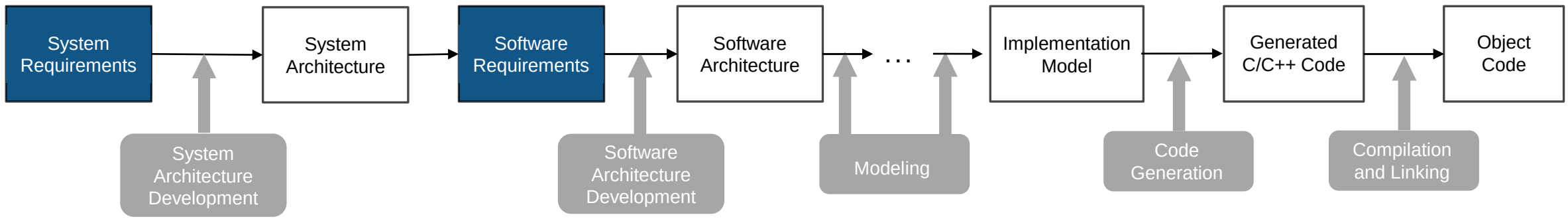
Requirements



System Level | **Software Level**

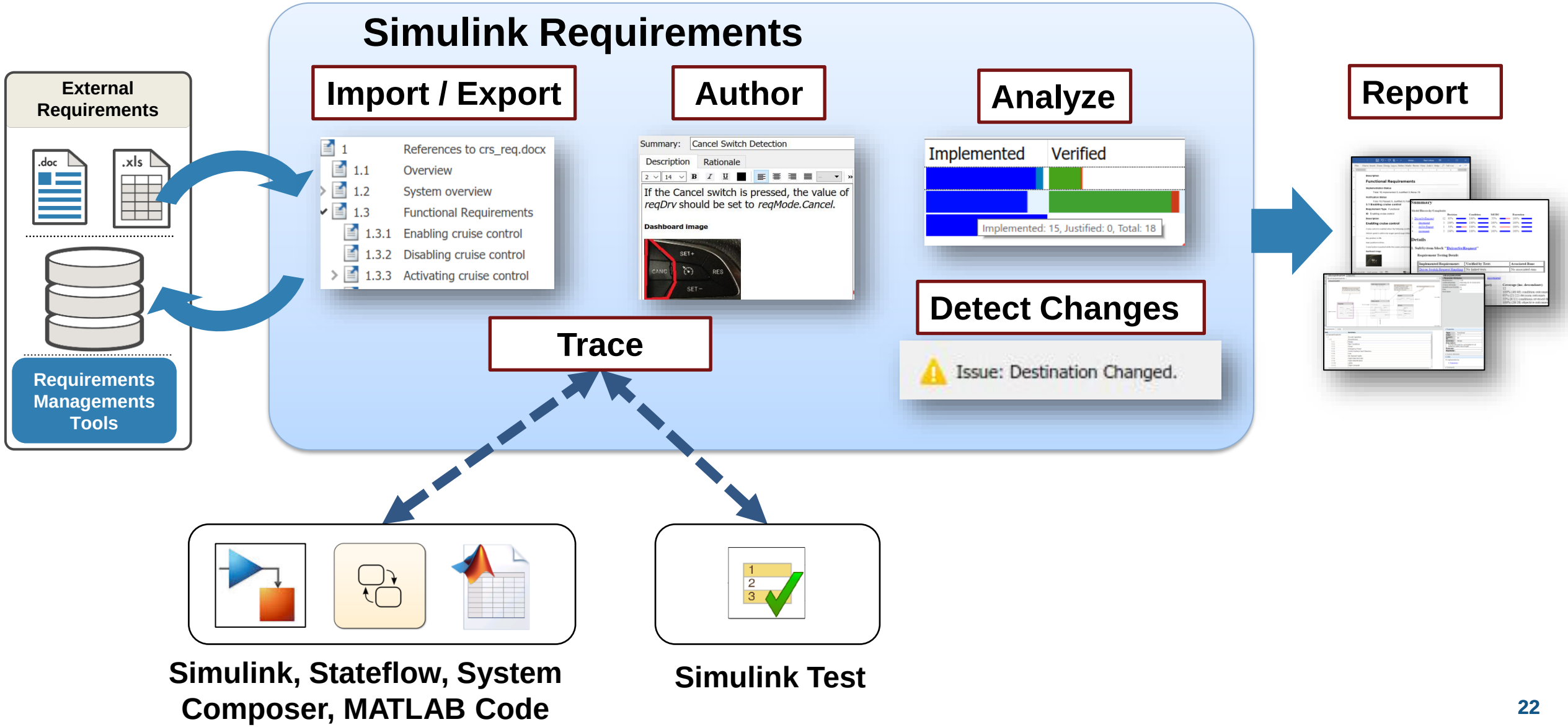
Requirement span multiple levels across workflow

- Are all requirements implemented?
- Is each requirement tested?
- What artifacts are impacted by a change?

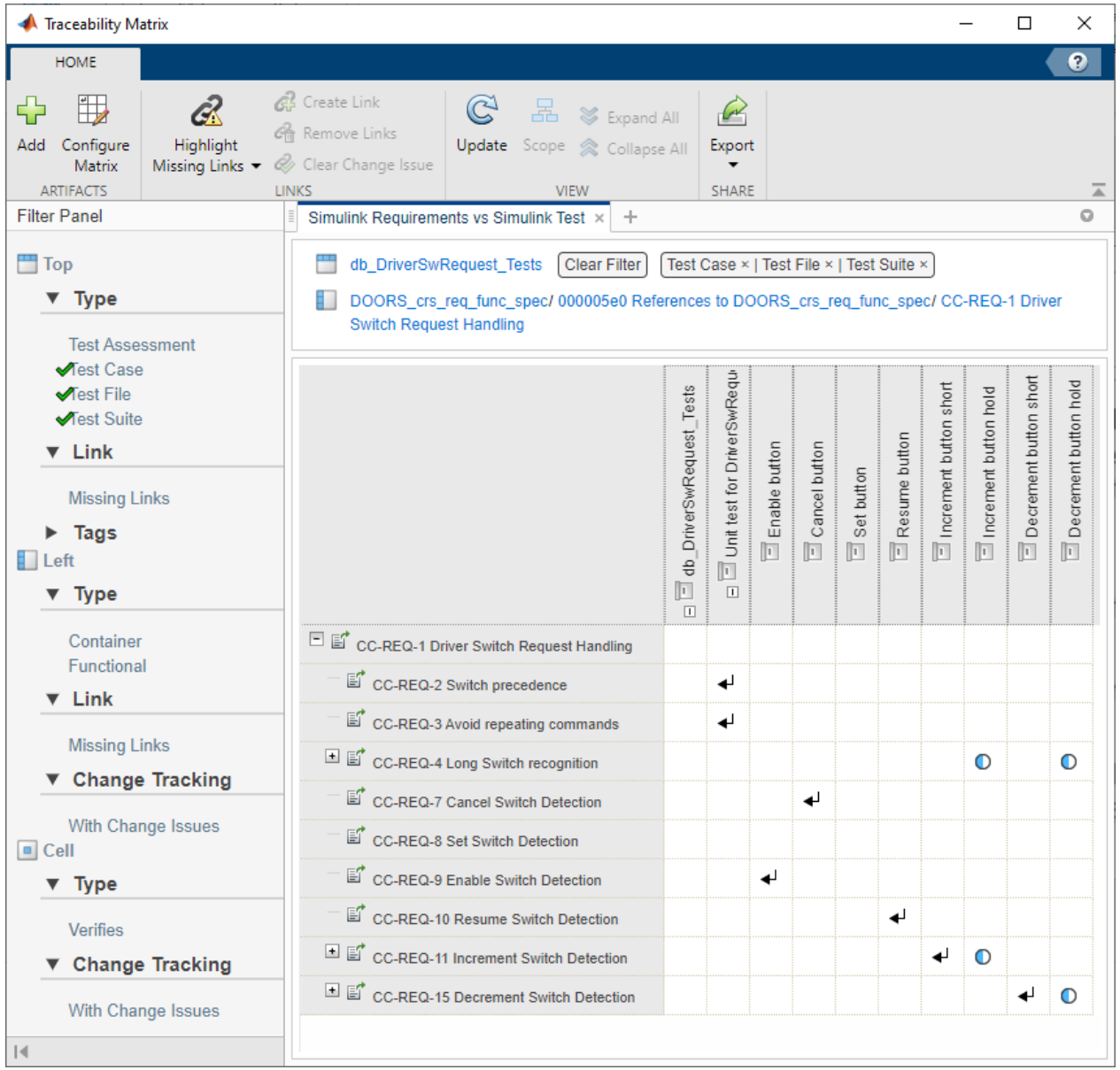


System Level | **Software Level**

Work with Requirements, Architecture and Design Together

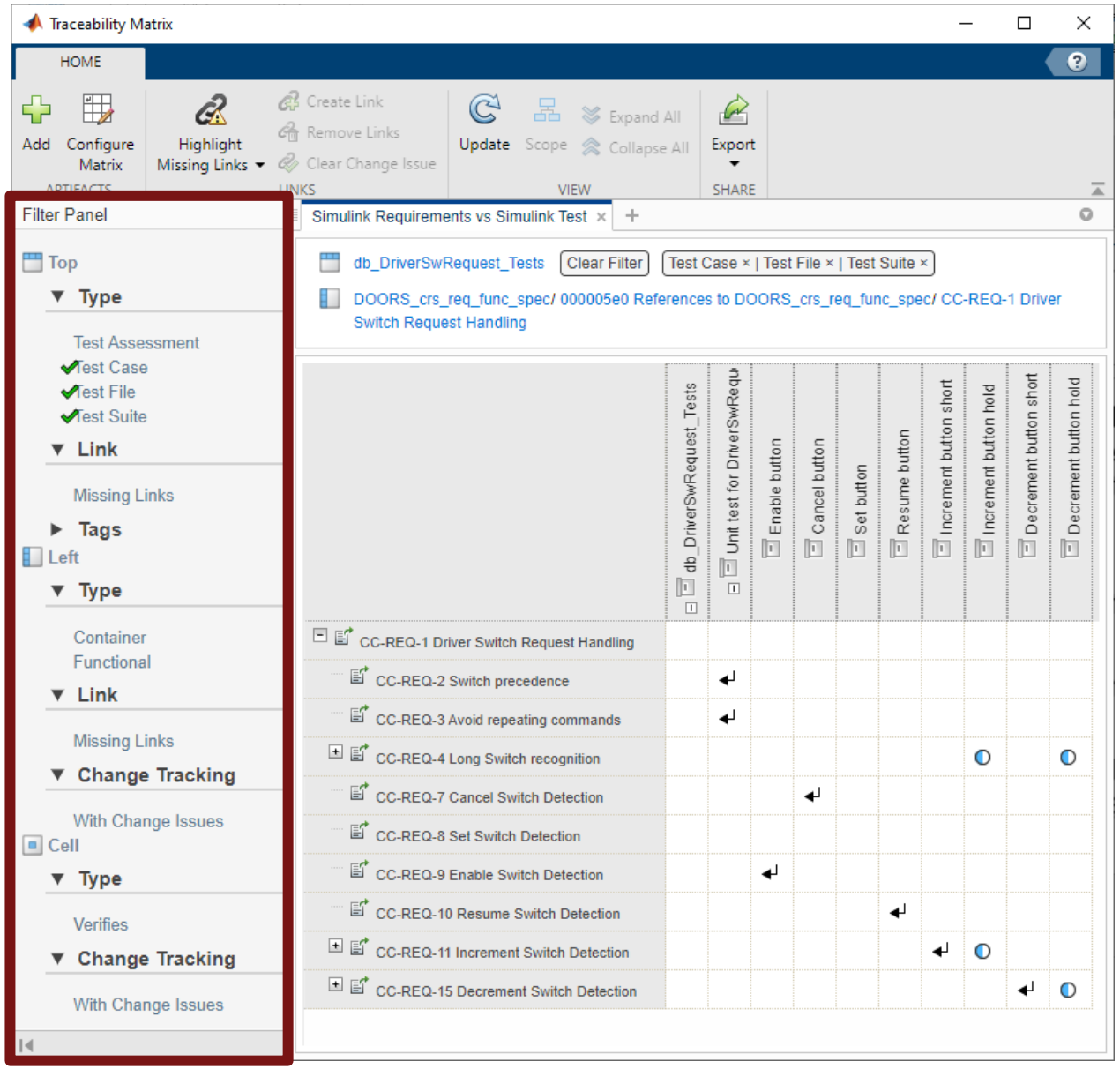


Review and Analyze Traceability with Traceability Matrix

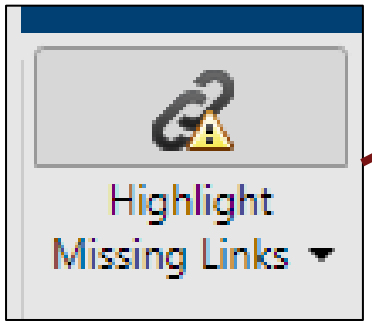


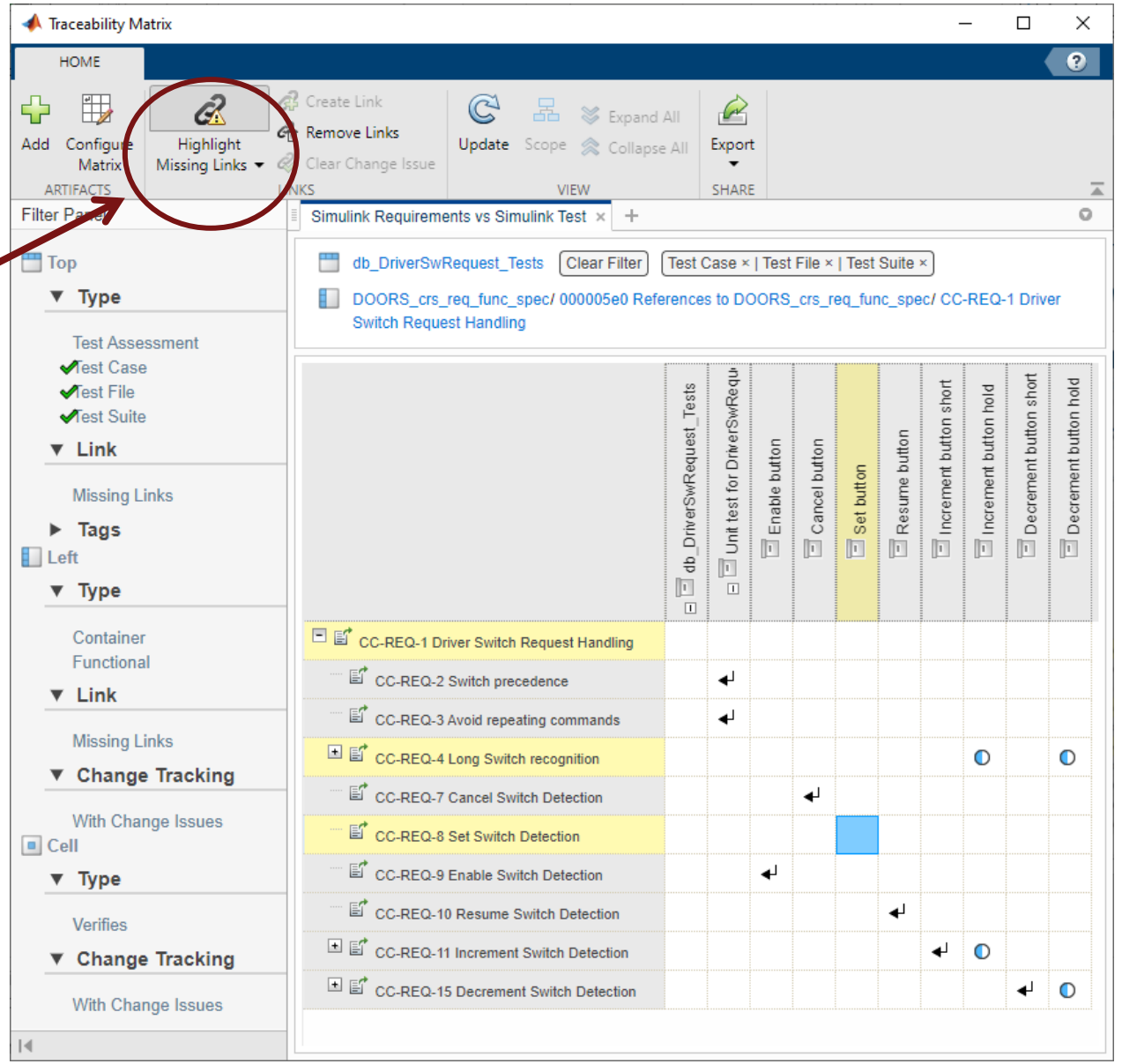
Review and Analyze Traceability with Traceability Matrix

- **Filters to focus view**

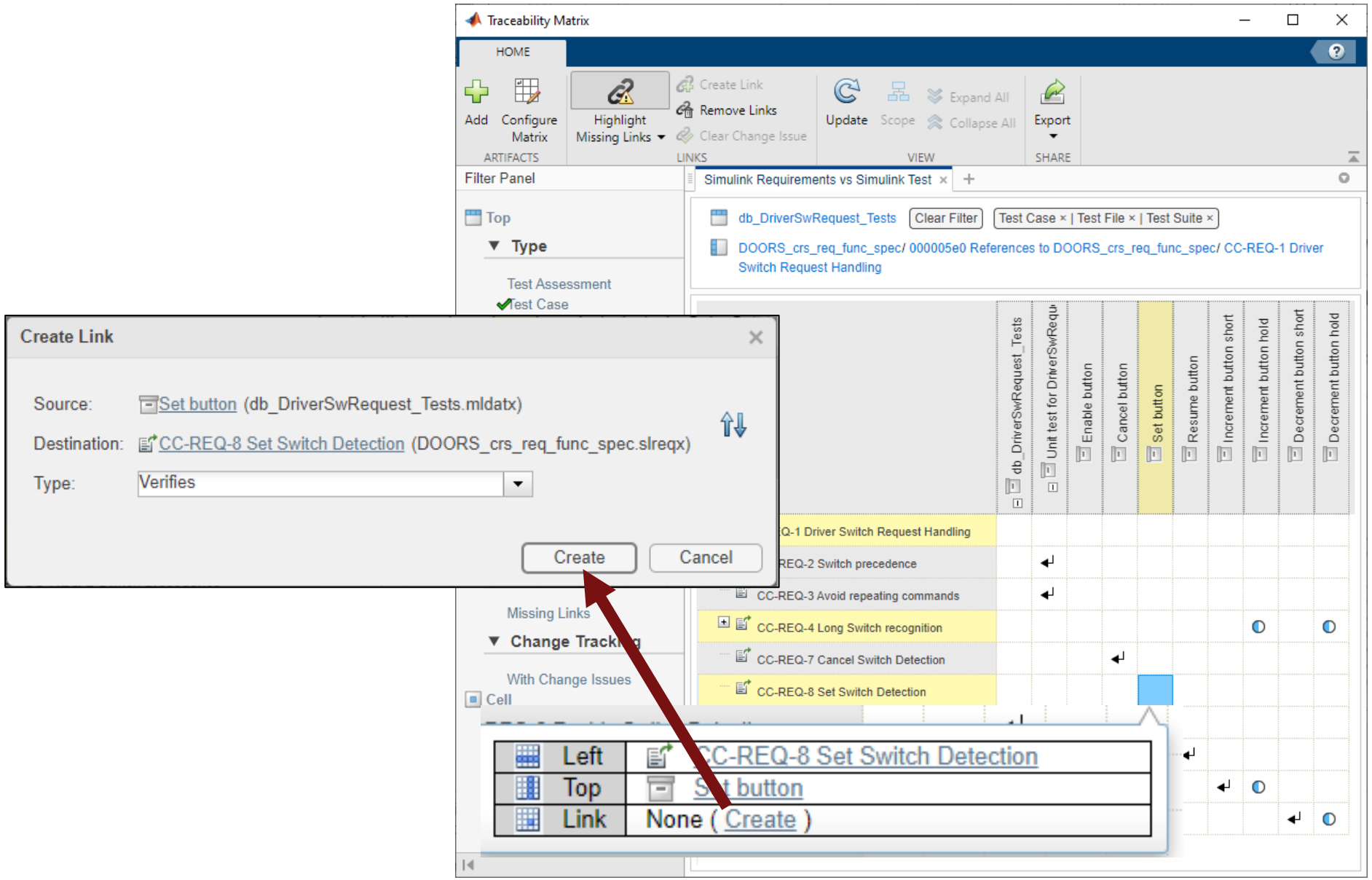


Review and Analyze Traceability with Traceability Matrix



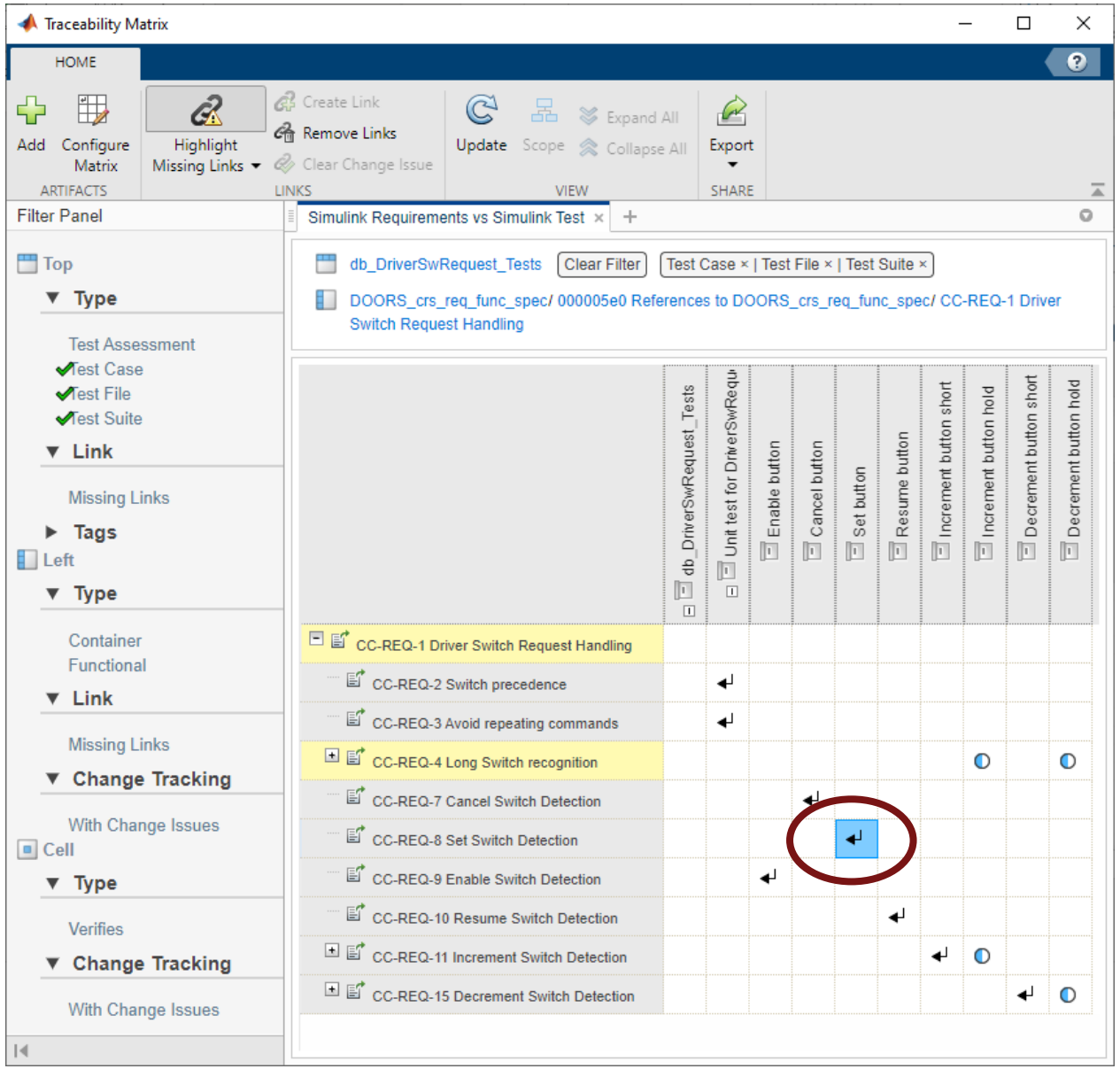


Review and Analyze Traceability with Traceability Matrix



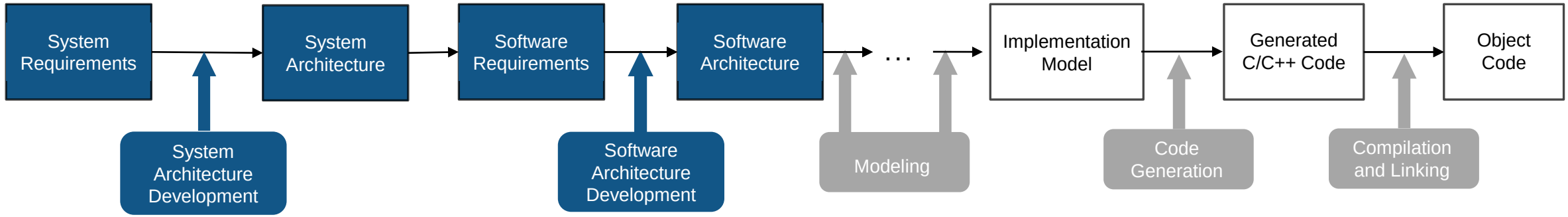
- **Create links to address gaps**

Review and Analyze Traceability with Traceability Matrix



- **Create links to address gaps**

System and Software Architecture



System Level | **Software Level**

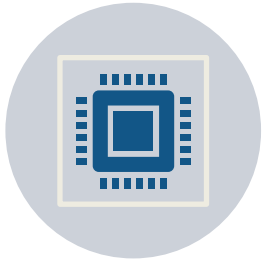
Why is architecture important?

AUTHOR AND VALIDATE REQUIREMENTS EARLIER

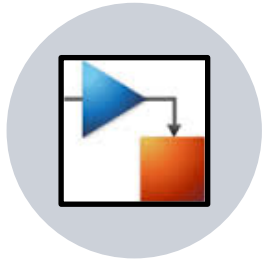
APPORTION THE SAFETY OBJECTIVES

COORDINATE SOFTWARE AND HARDWARE INTERFACES

SHARE REQUIREMENTS WITHOUT AMBIGUITY

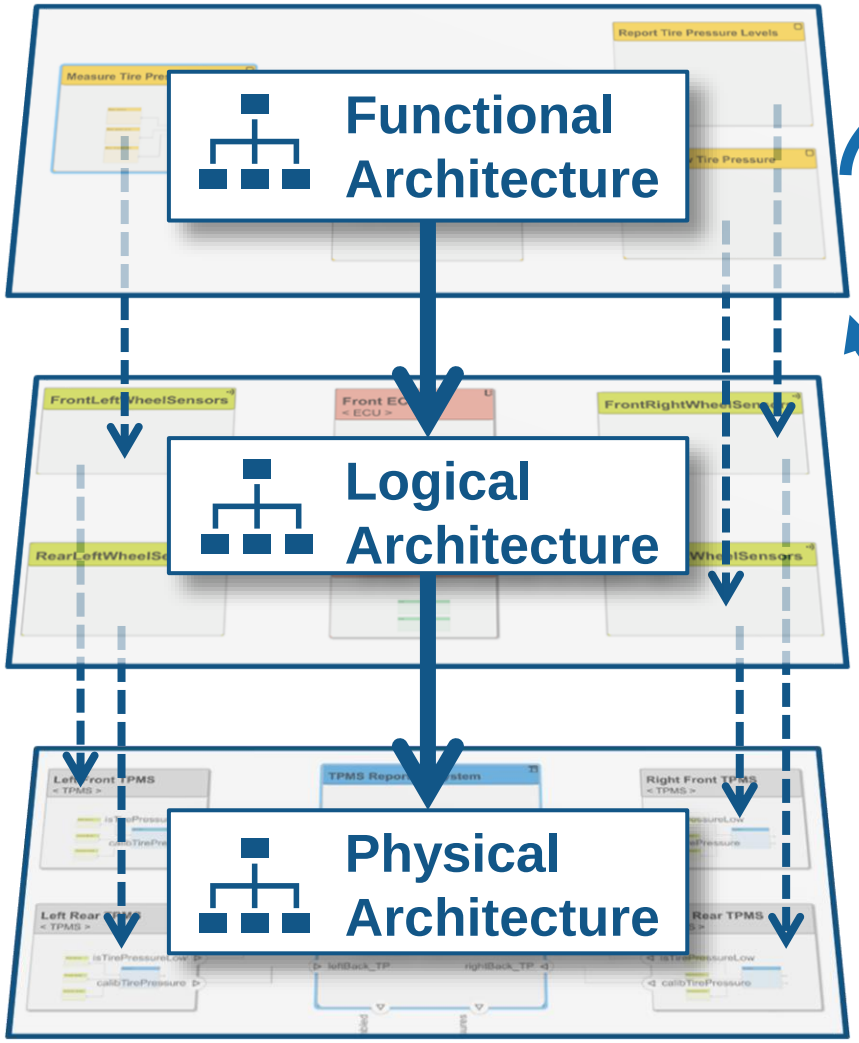


DETERMINE ALLOCATION TO SOFTWARE OR HARDWARE



LINK TO BEHAVIORAL MODELS

Describe systems using multiple architecture models to organize and check the completeness of the requirements

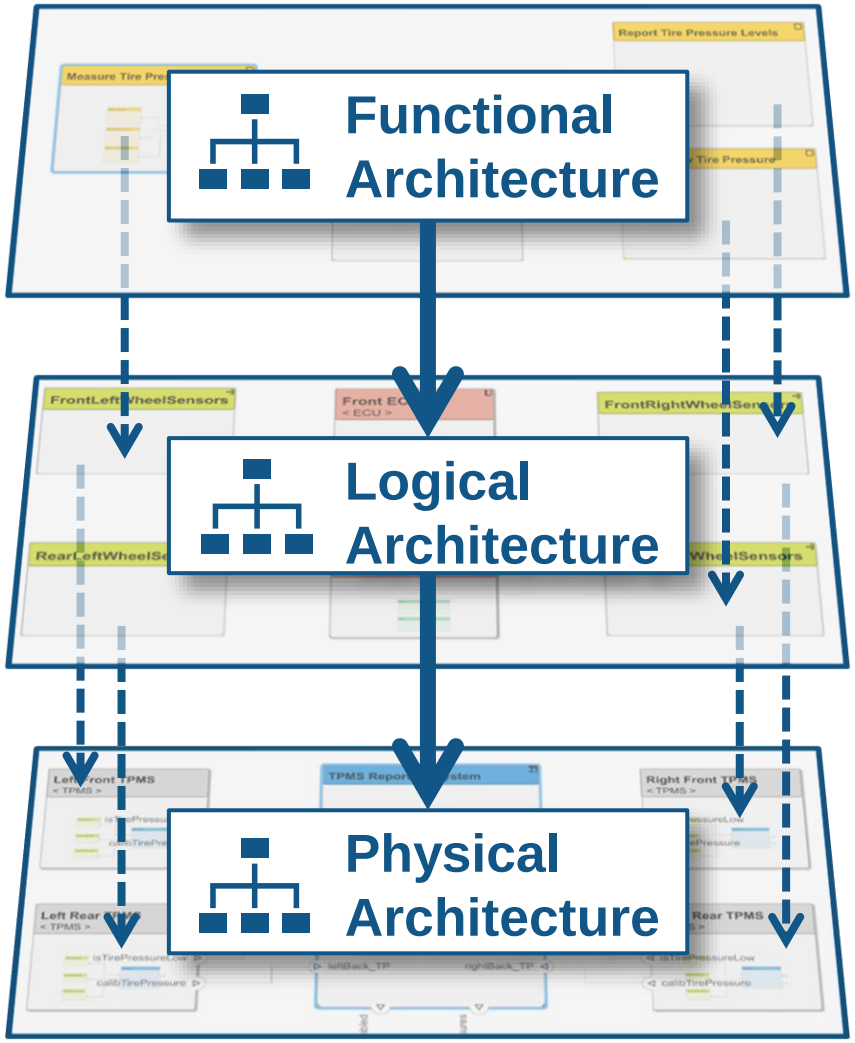


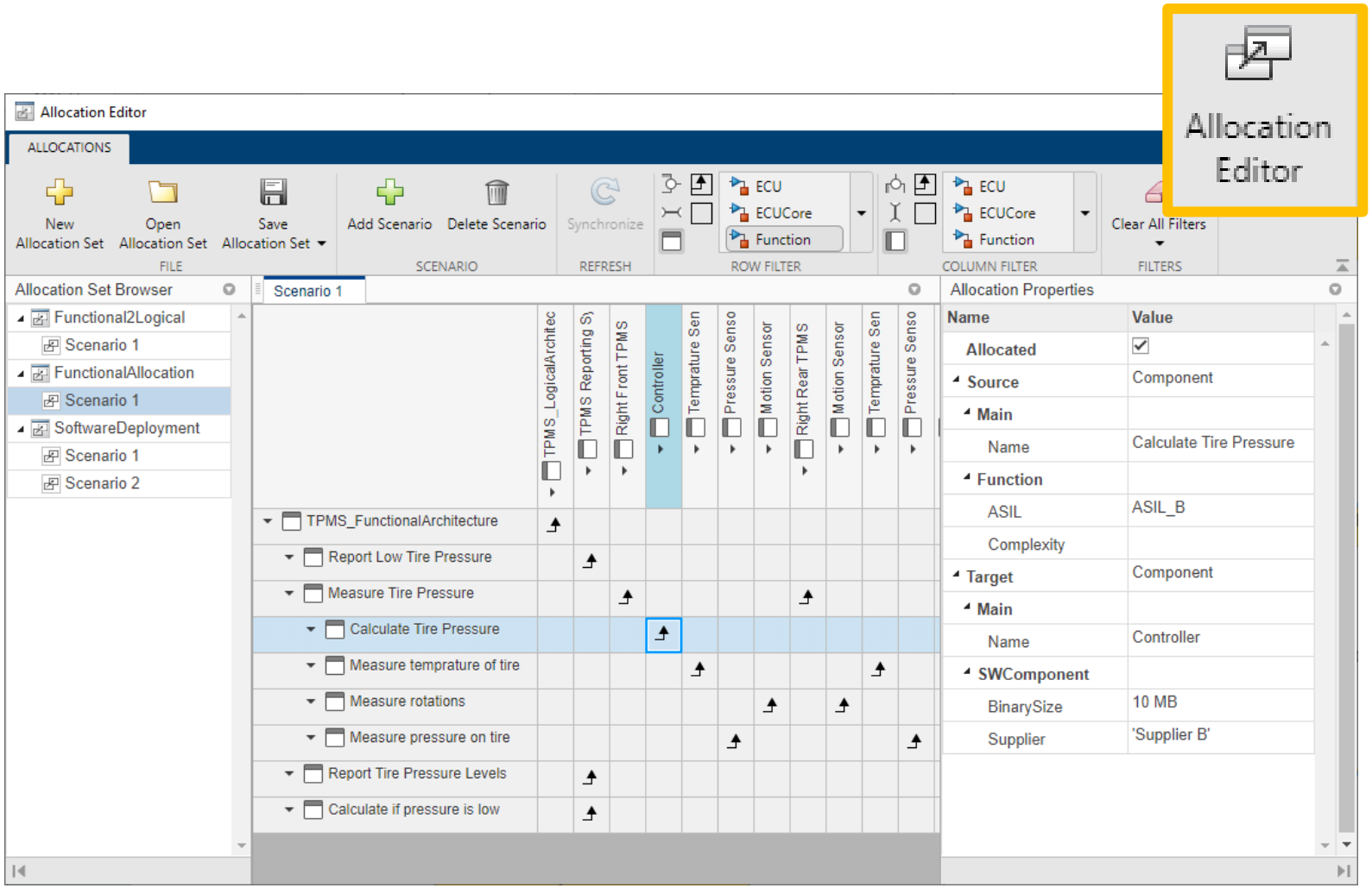
Analyze & Simulate

Update & Refine

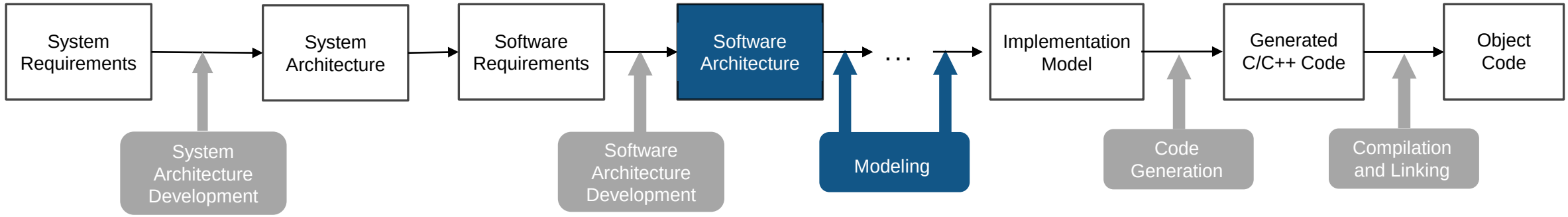
- Stakeholder Needs
- Functional Requirements
- System Requirements
- Hardware Requirements

Allocate elements from one model to another to create a traceable & analyzable link



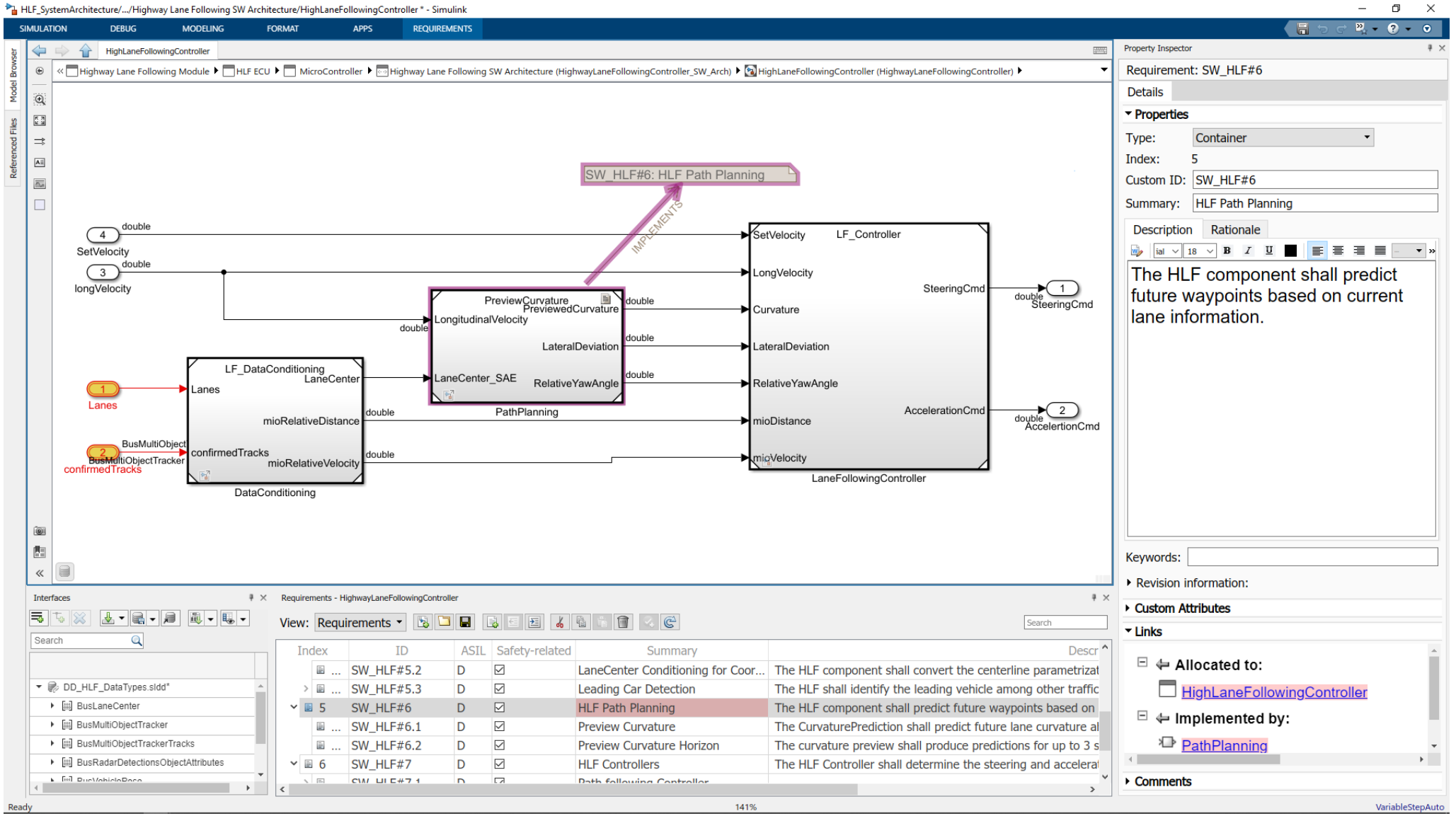


Reference Workflow for Certification – Detailed Design

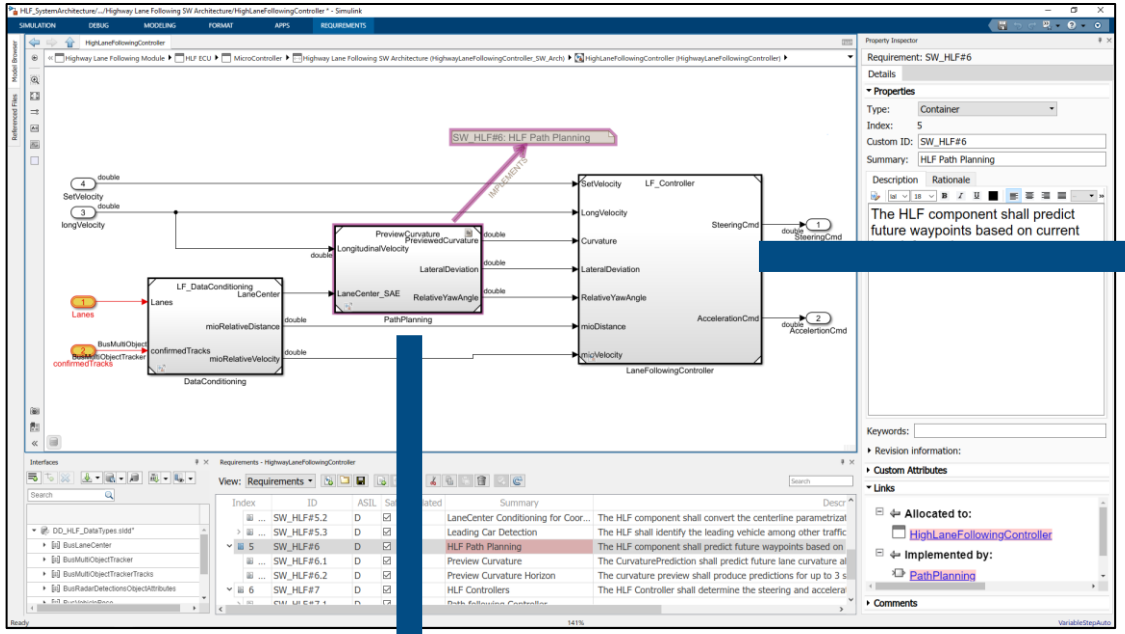


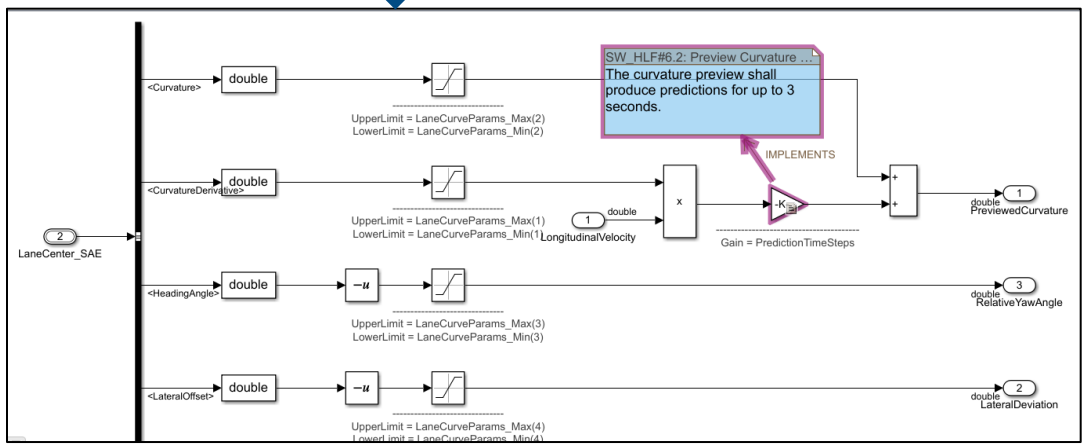
System Level | **Software Level**

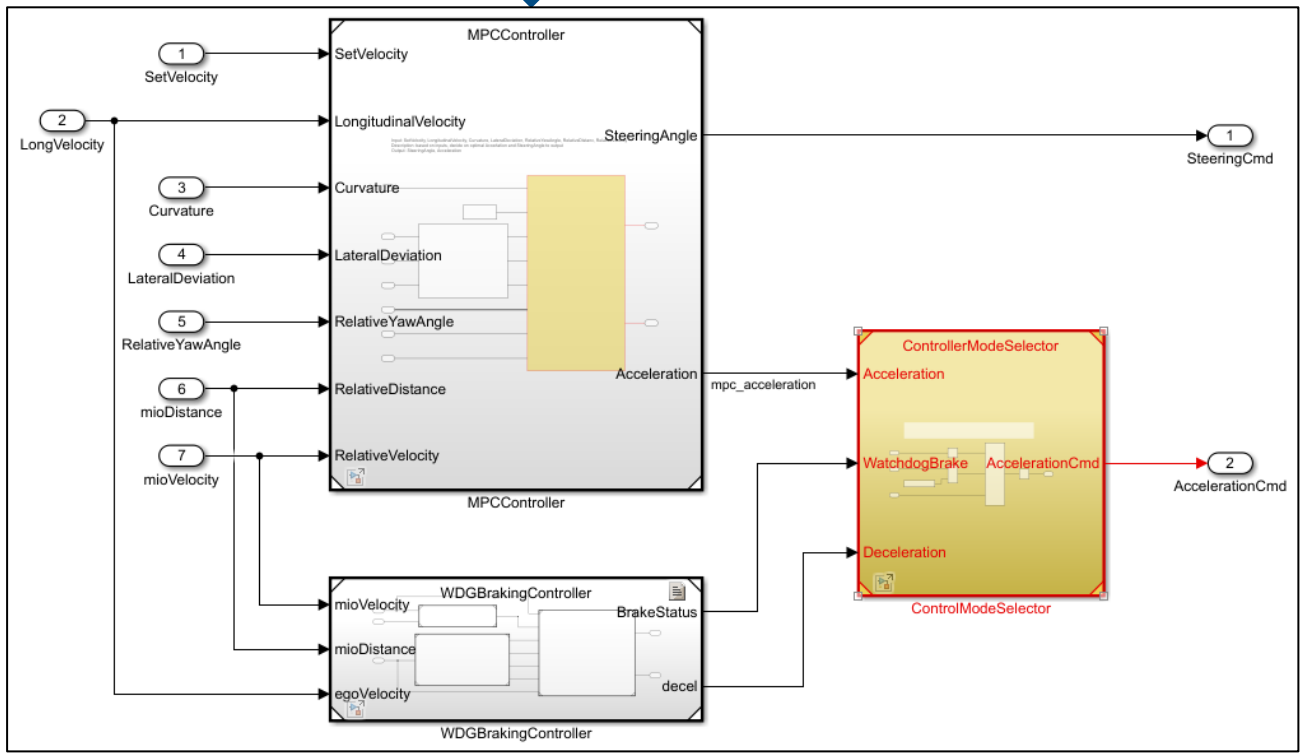
Reference Workflow for Certification – Detailed Design



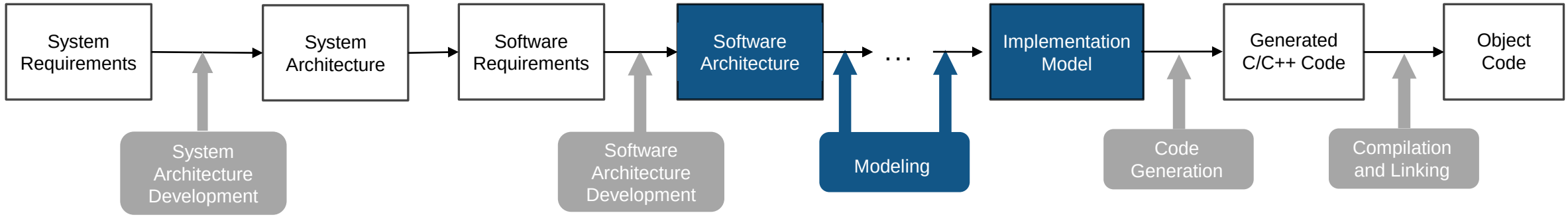
Reference Workflow for Certification – Detailed Design





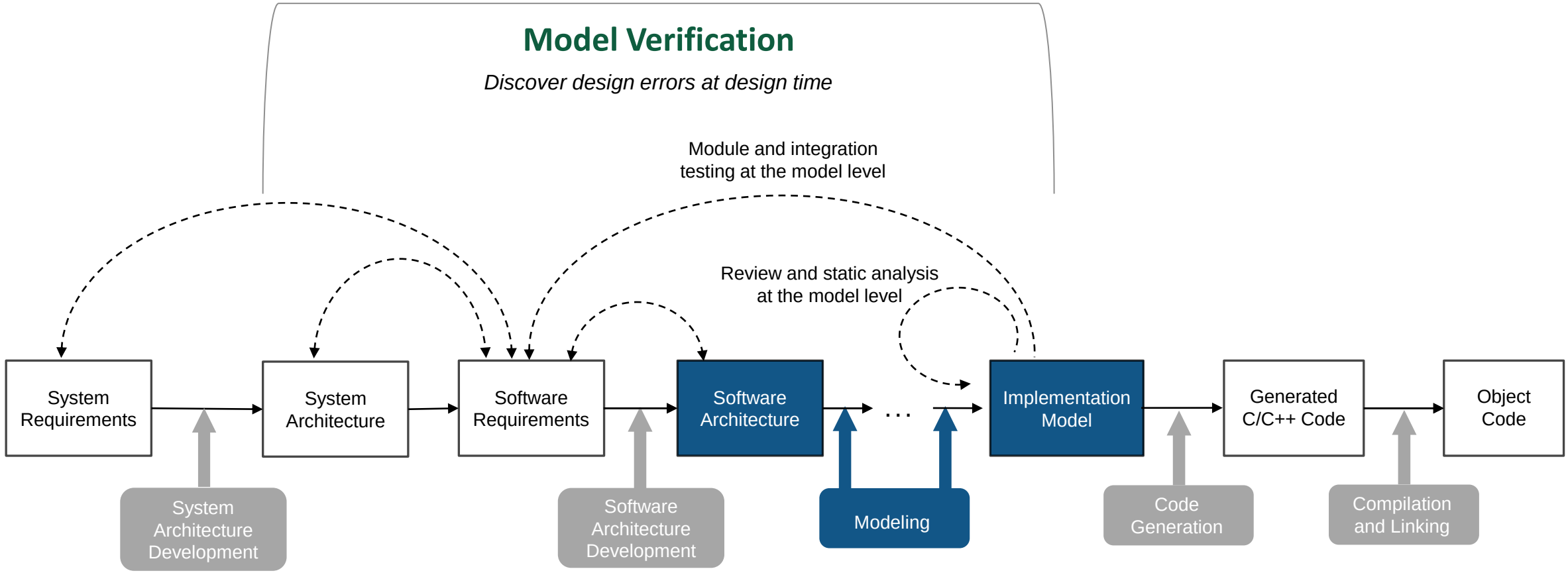


Reference Workflow for Certification – Detailed Design



System Level | **Software Level**

Reference Workflow for Certification – Model Verification

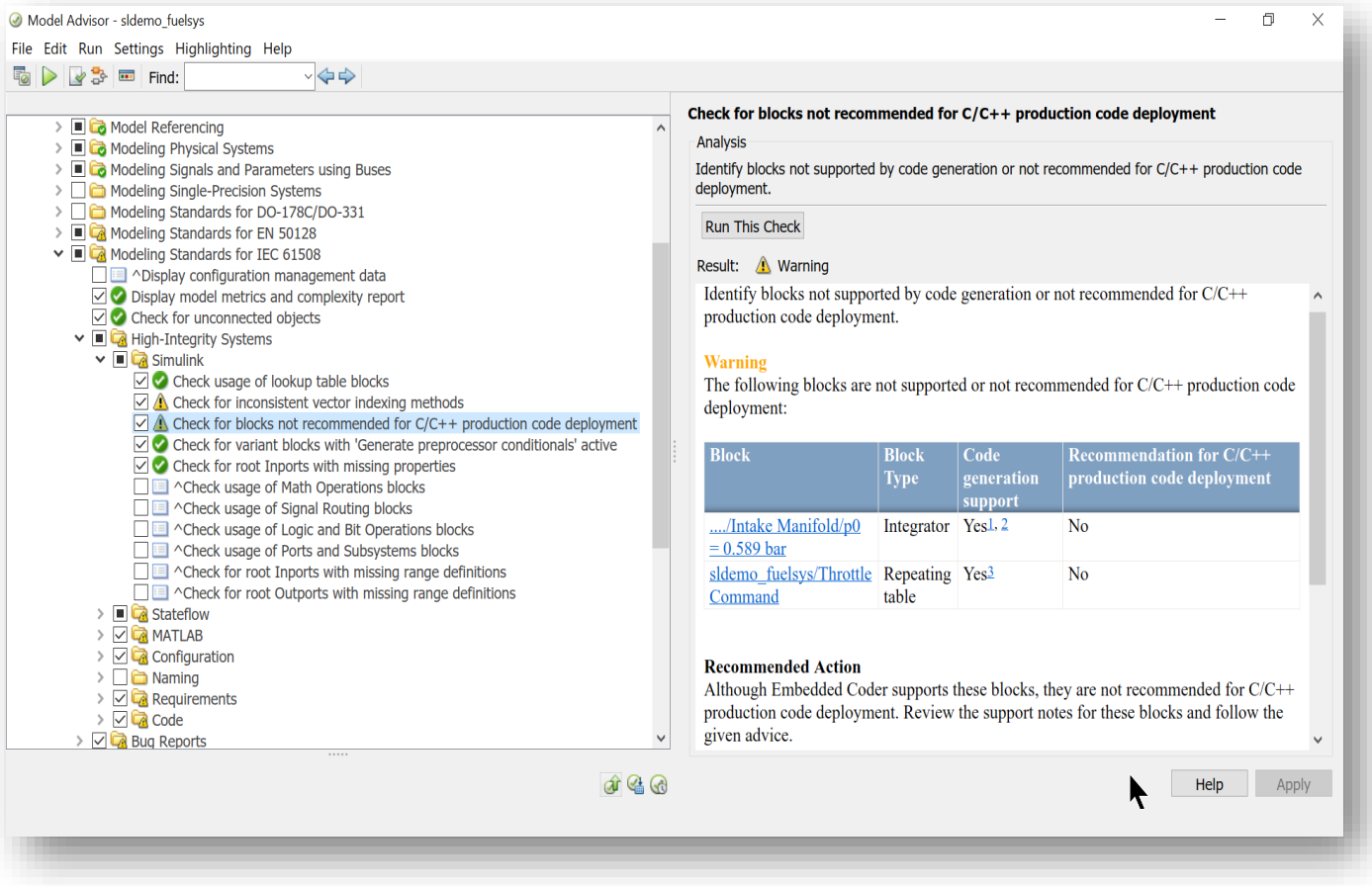


Verify Model Complies with Guidelines and Standards

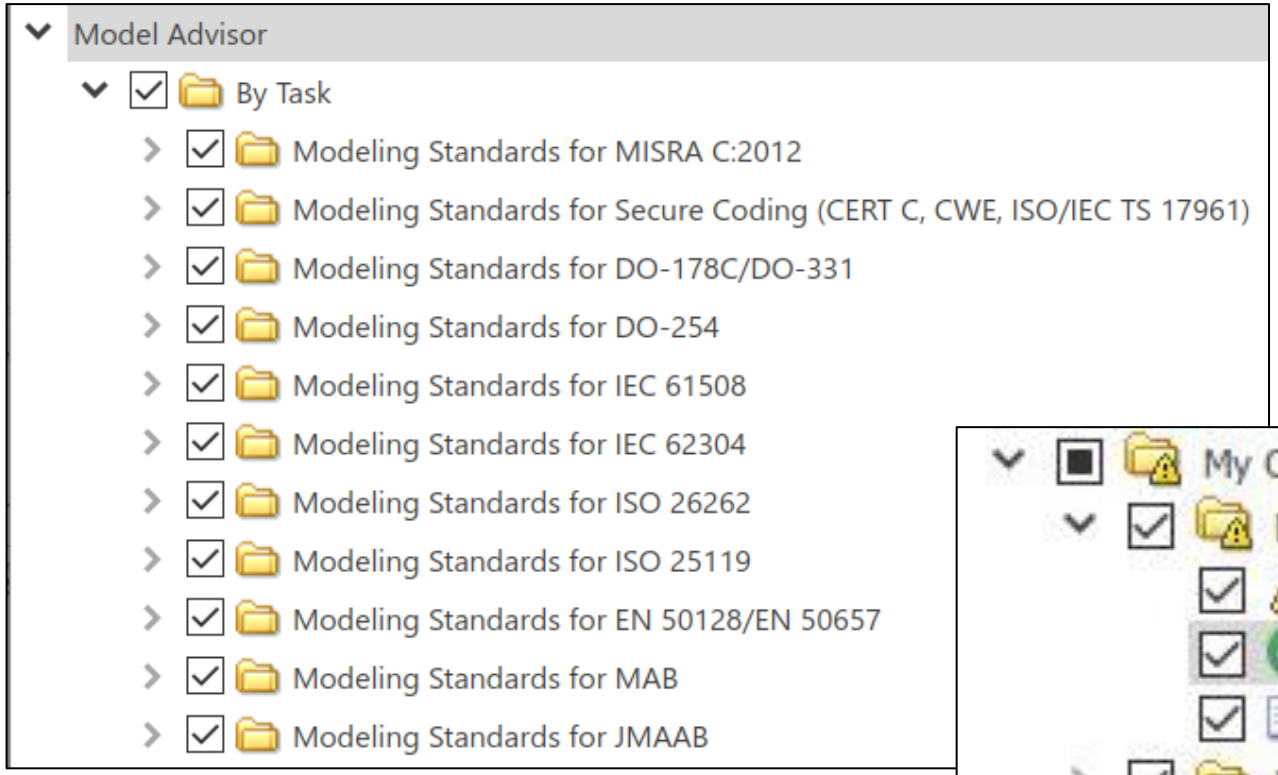
Static analysis checks for:

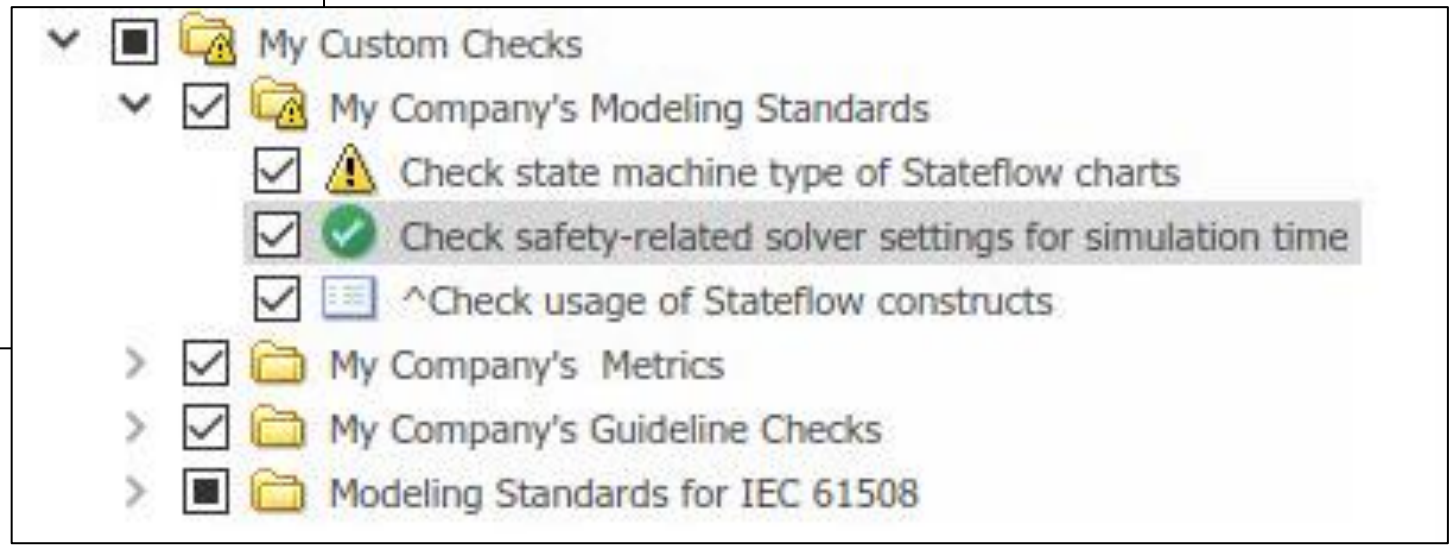
- Readability and Semantics
- Performance and Efficiency
- Reusability
- And more……

Model Advisor Analysis

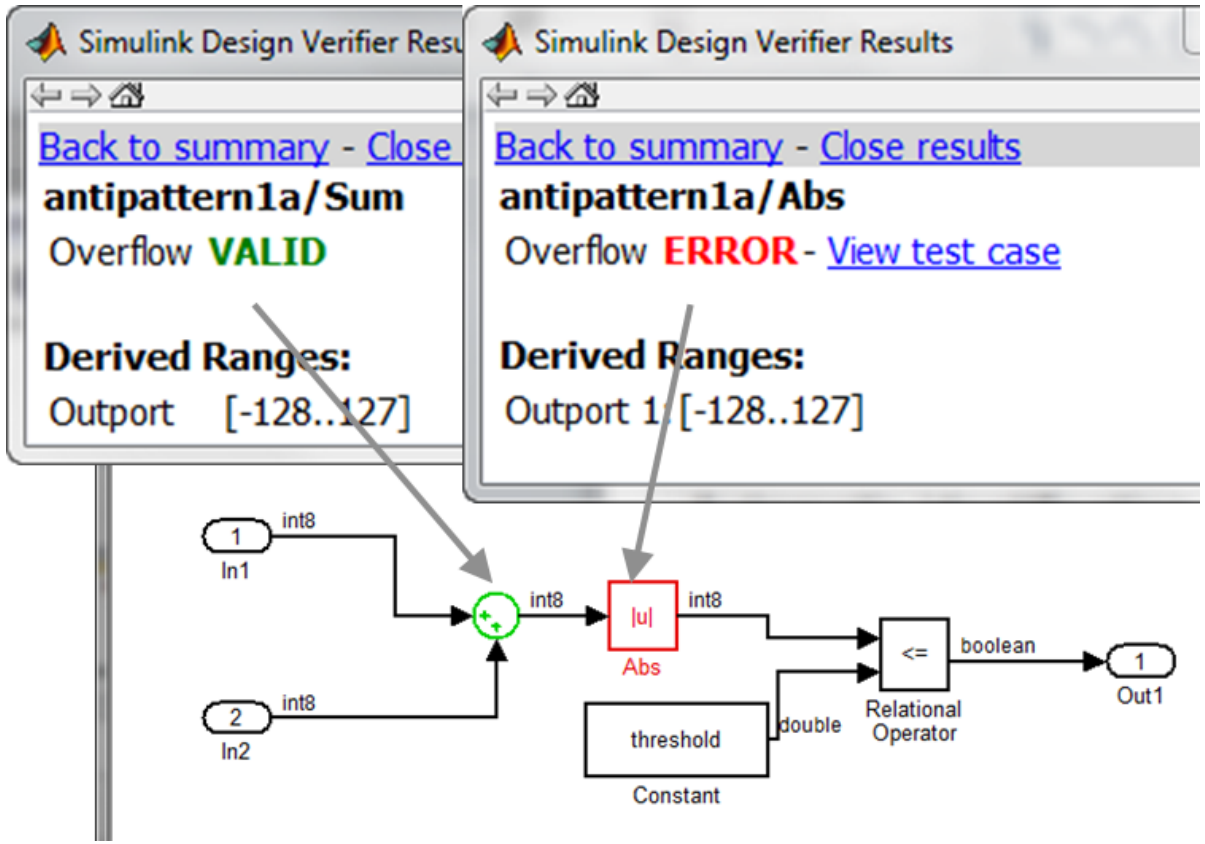


Built in checks for industry standards and guidelines



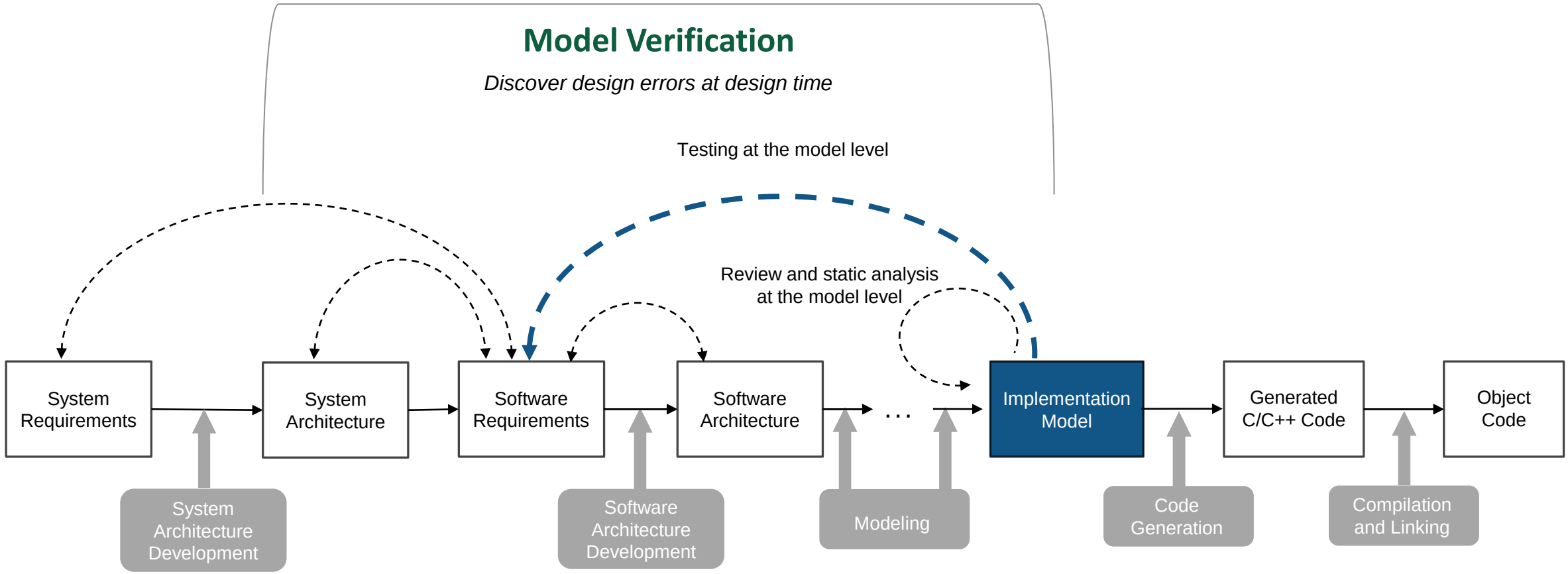


Detect Design Errors Using Formal Methods



- Find design errors
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations
- Generate counter example to reproduce error

Model-Based Design Reference Workflow



Systematic Functional Testing with Simulink Test

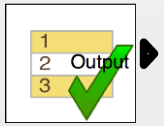
Test Case

Model Sim through SIL, PIL and HIL
Scale with Parallel Computing Toolbox and Continuous Integration

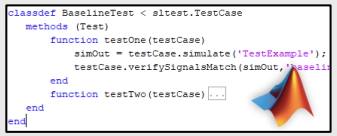
Inputs

Data file (input)

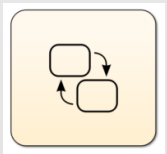
Signal Editor



Test Sequence

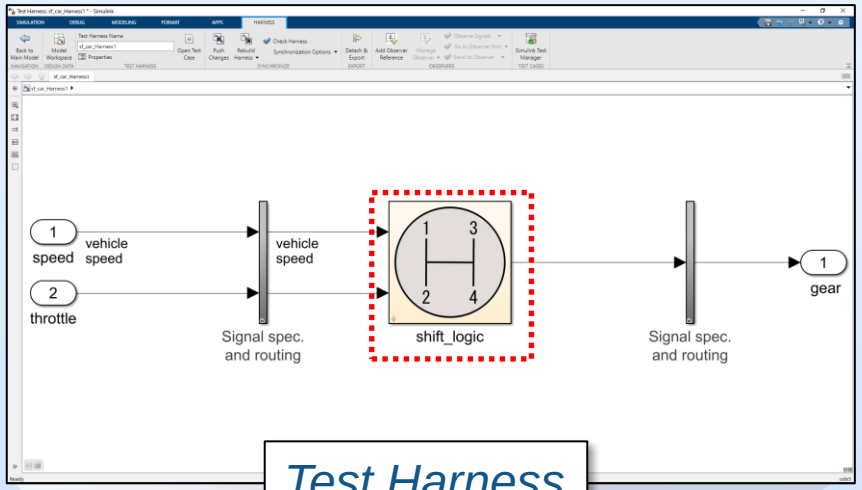


MATLAB Code

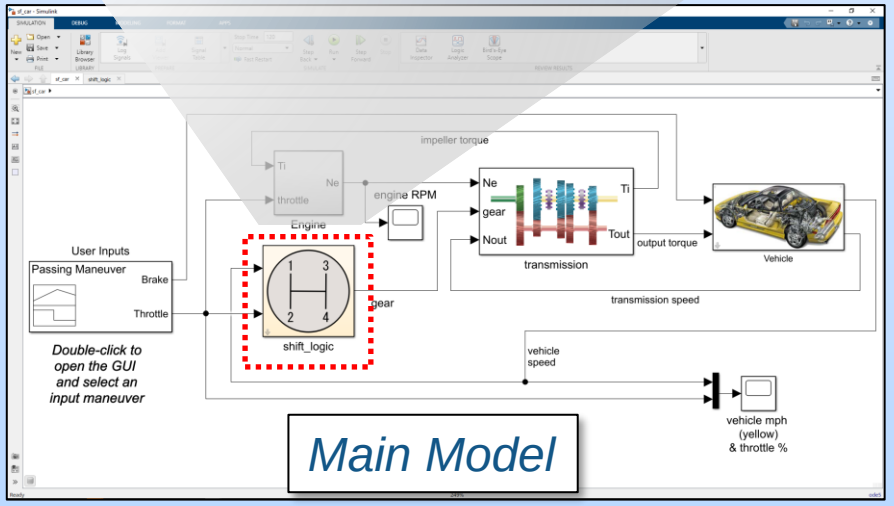


Stateflow

and more!

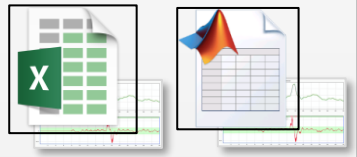


Test Harness

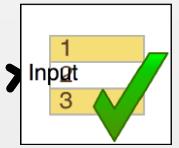


Main Model

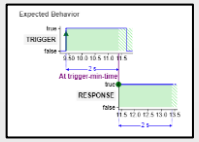
Assessments



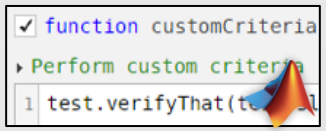
Data file baseline)



Test Assessment



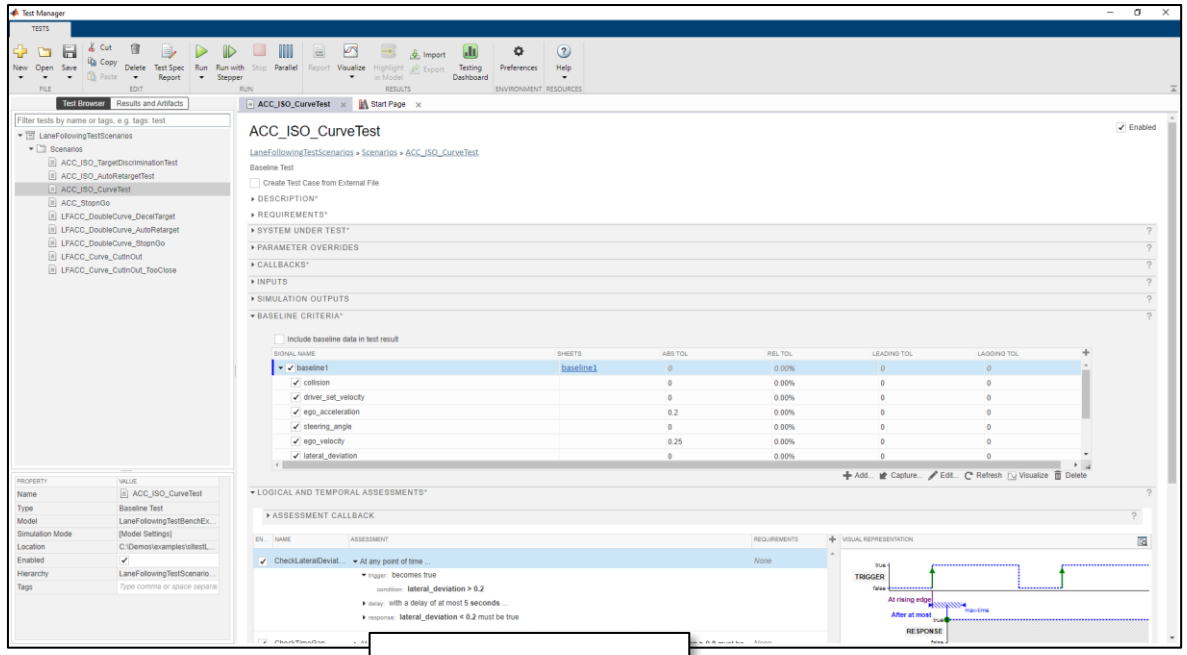
Temporal Assessment

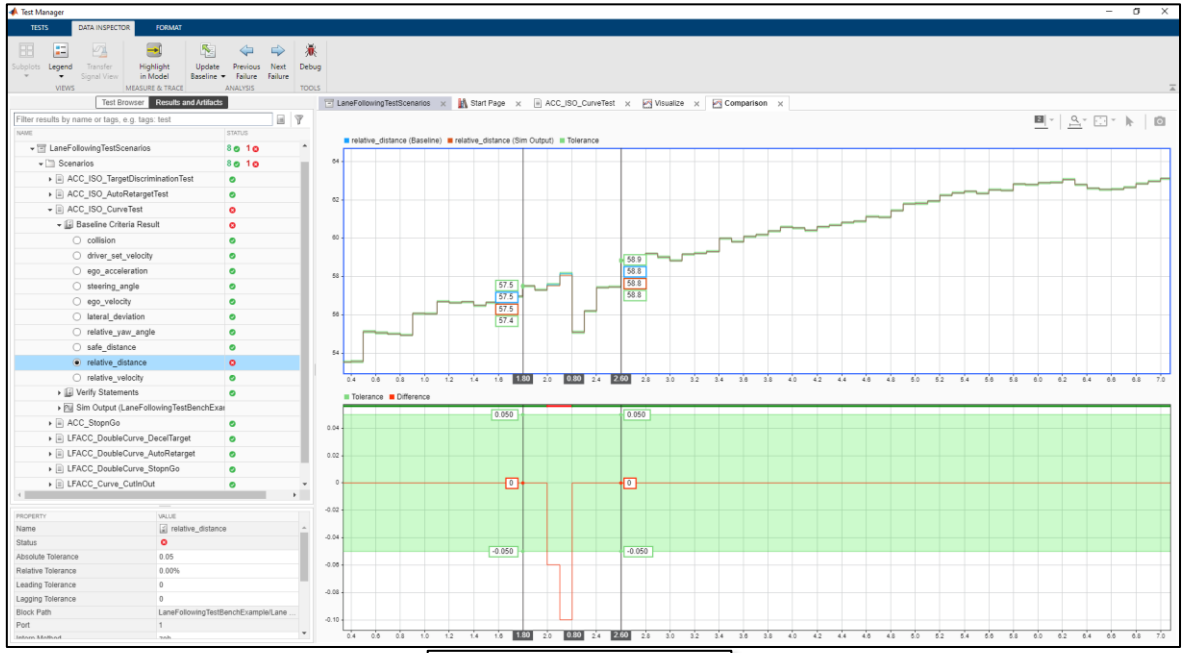


MATLAB Code

and more!

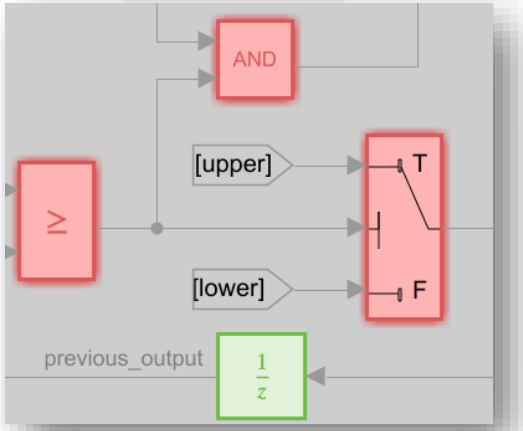
Manage Testing and Test Results



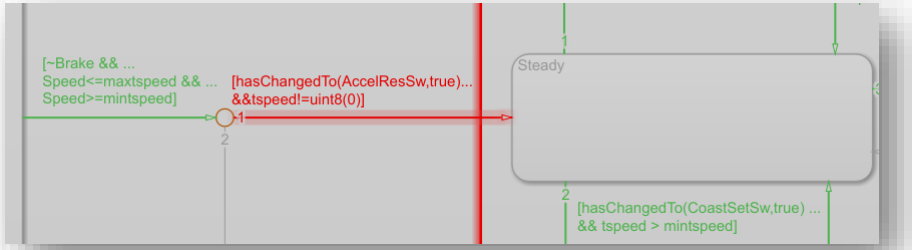


Coverage Analysis to Measure Testing

Simulink



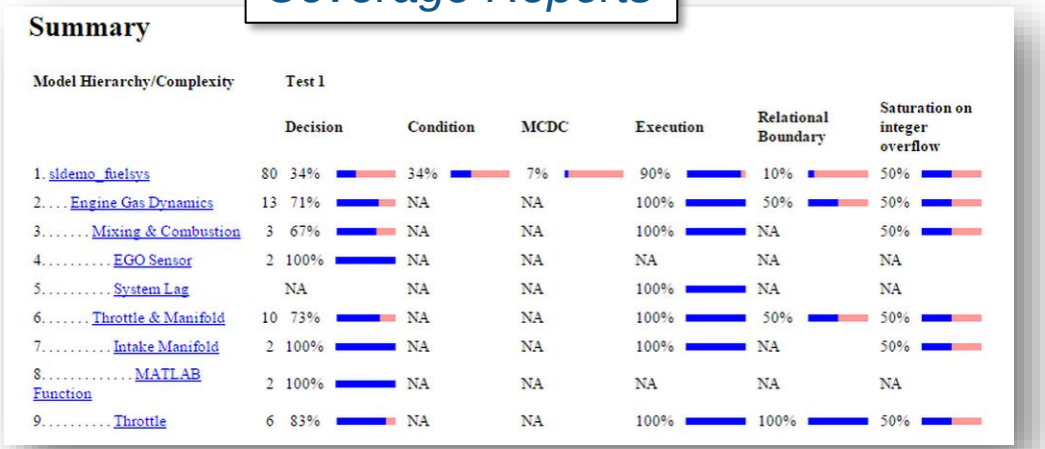
Stateflow



Code

- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors

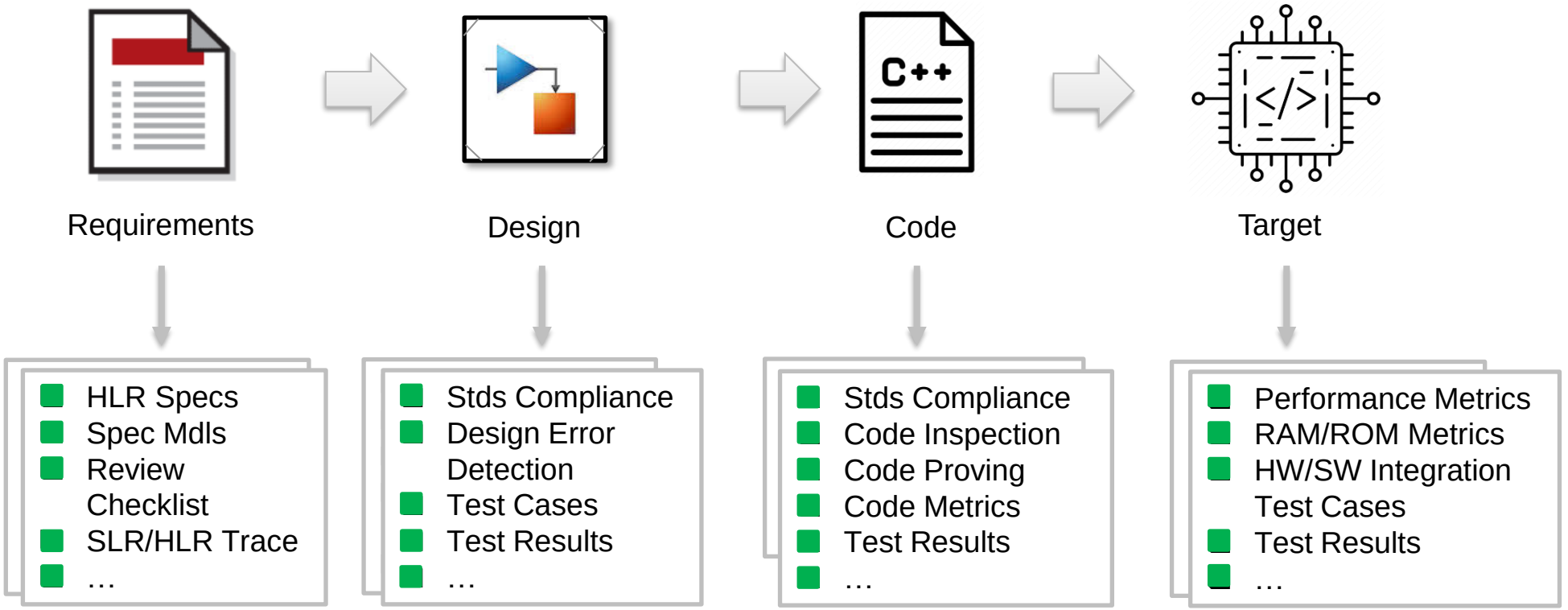
Coverage Reports



Summary

Model Hierarchy/Complexity	Test 1									
	Decision		Condition		MCDC		Execution	Relational Boundary		Saturation on integer overflow
1. sldemo_fuelsys	80	34%		34%		7%	90%		10%	50%
2. Engine Gas Dynamics	13	71%		NA		NA	100%		50%	50%
3. Mixing & Combustion	3	67%		NA		NA	100%		NA	50%
4. EGO Sensor	2	100%		NA		NA	NA		NA	NA
5. System Lag		NA		NA		NA	100%		NA	NA
6. Throttle & Manifold	10	73%		NA		NA	100%		50%	50%
7. Intake Manifold	2	100%		NA		NA	100%		NA	50%
8. MATLAB Function	2	100%		NA		NA	NA		NA	NA
9. Throttle	6	83%		NA		NA	100%		100%	50%

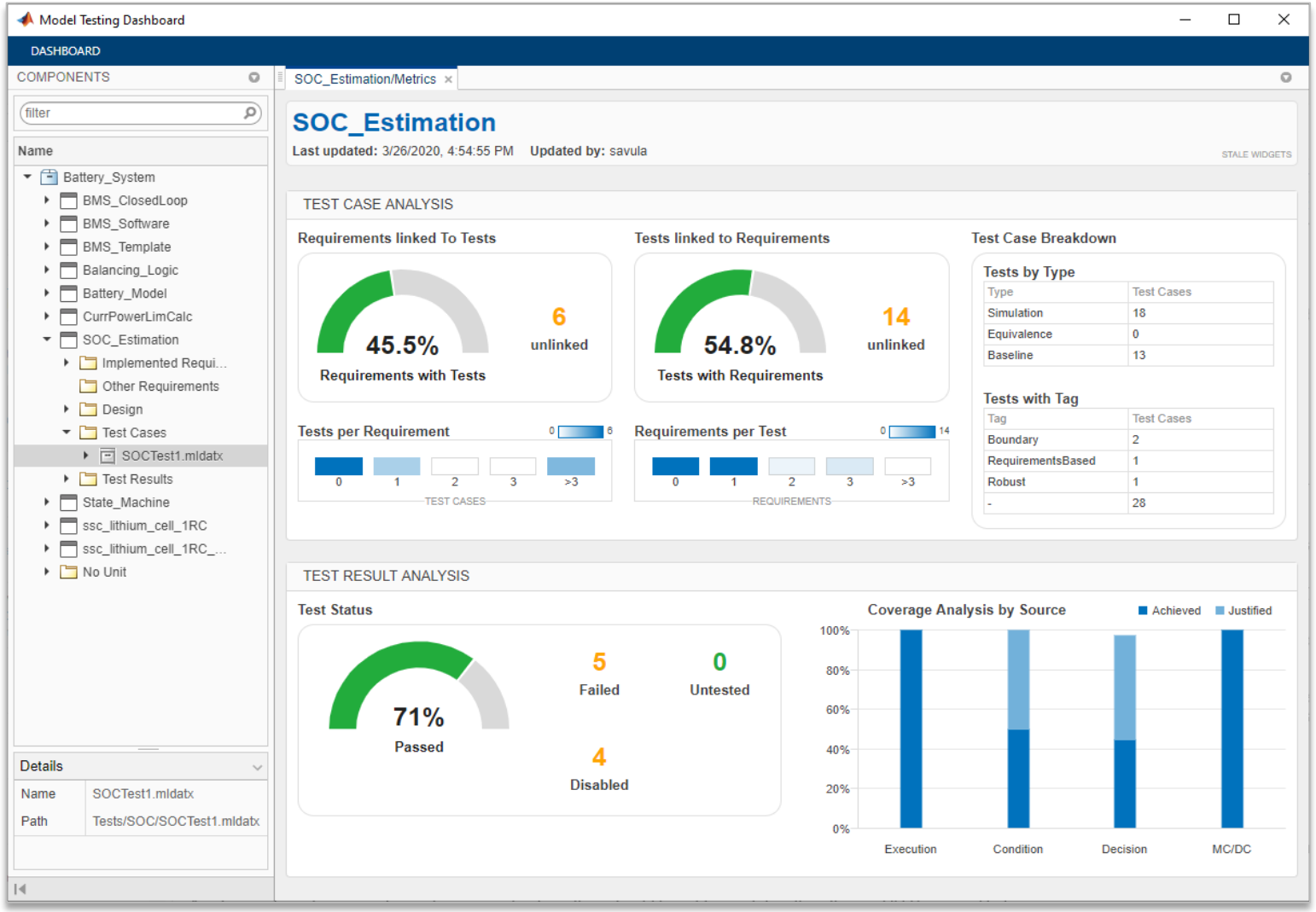
Challenge: Managing the many activities and artifacts needed to meet certification standards



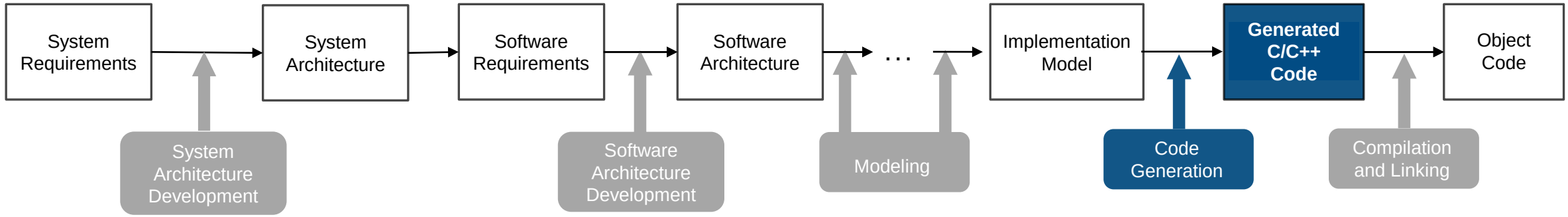
Model Testing Dashboard: Manage progress, completeness and quality of requirements-based testing

R2020b

- Central view that summarizes testing data and status
- Identify gaps and respond faster to requirements changes
- View results from a continuous integration system

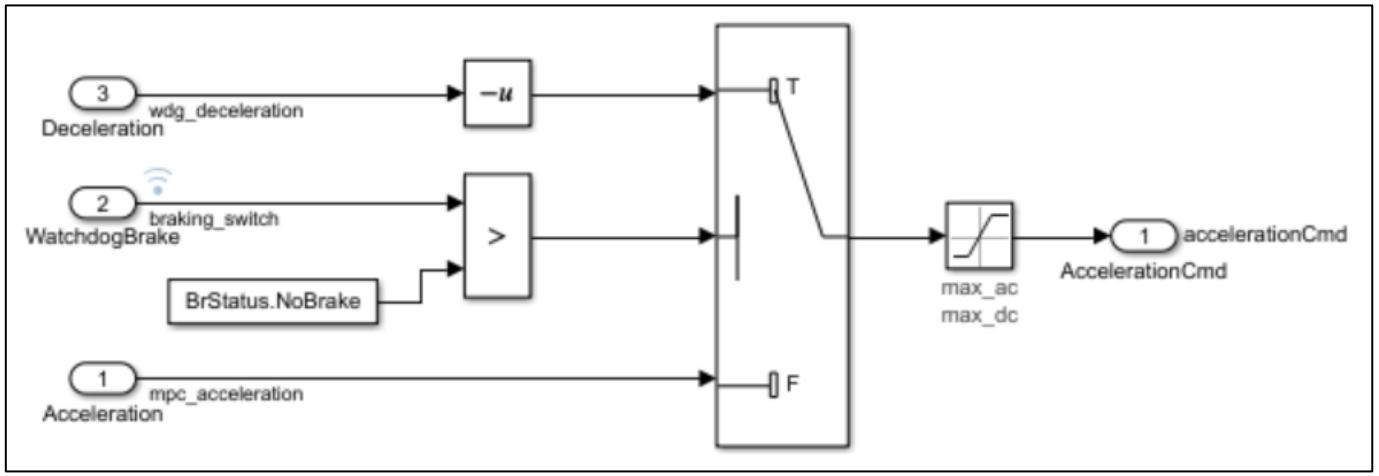


Code Generation



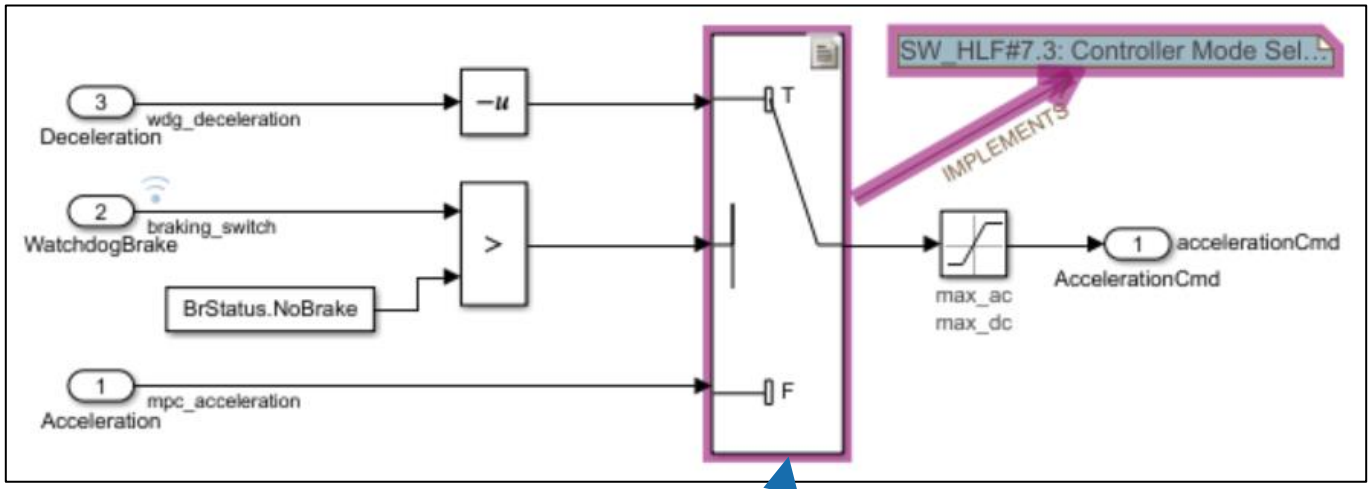
System Level | **Software Level**

Automatically generate production-quality code that behaves the same way as the model you created in Simulink

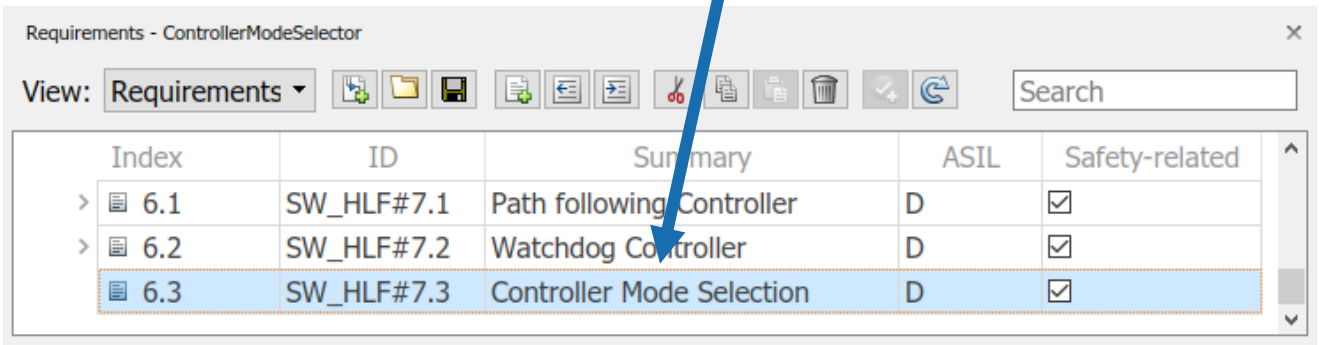


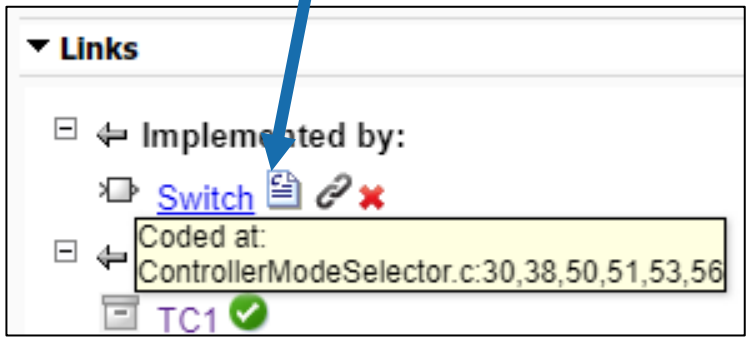

```
27   /* Switch: '<Root>/Switch' incorporates:
28    *  Constant: '<Root>/Constant'
29    *  RelationalOperator: '<Root>/GreaterThan'
30    *
31    * Block requirements for '<Root>/Switch':
32    *  1. SW_HLF#7.3 Controller Mode Selection
33    */
34   if ((*rtu_WatchdogBrake) > NoBrake) {
35     /* UnaryMinus: '<Root>/UnaryMinus' */
36     rtb_UnaryMinus = -(*rtu_Deceleration);
37   } else {
38     rtb_UnaryMinus = *rtu_Acceleration;
39   }
40
41   /* End of Switch: '<Root>/Switch' */
```

Requirement Traceability included in Generated Code

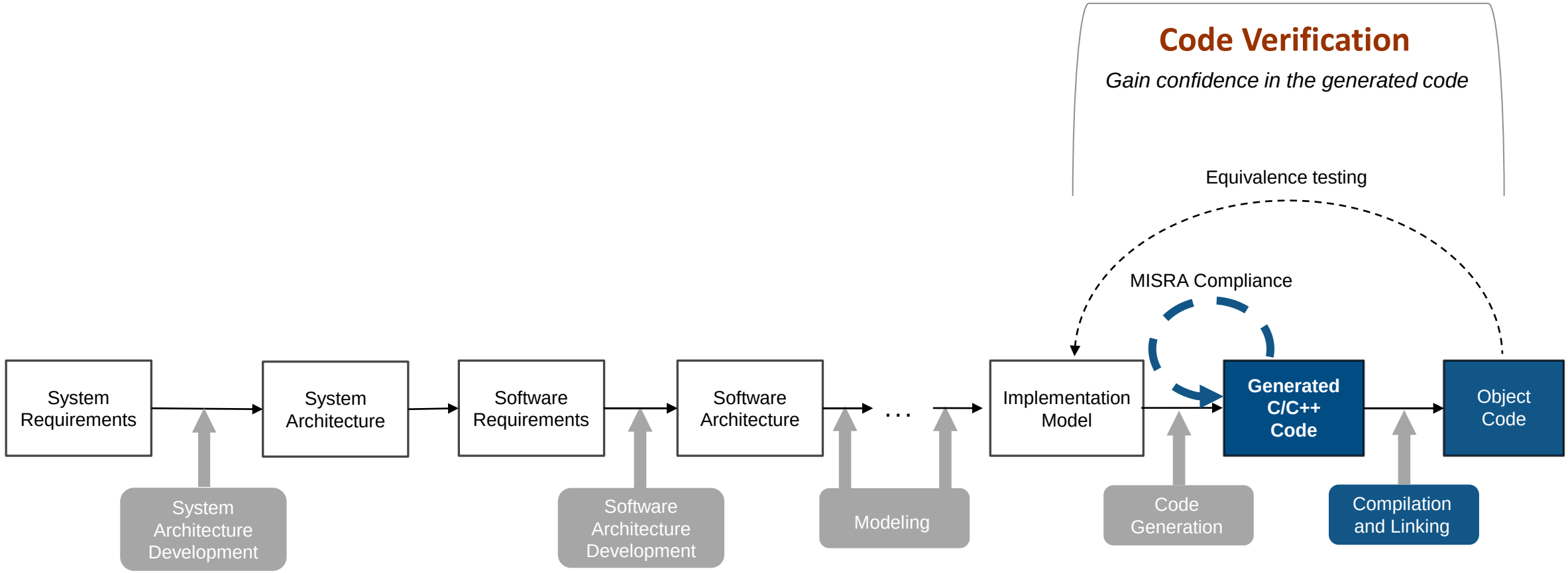



```
27  /* Switch: '<Root>/Switch' incorporates:
28   *  Constant: '<Root>/Constant'
29   *  RelationalOperator: '<Root>/GreaterThan'
30   *
31   * Block requirements for '<Root>/Switch':
32   *  1. SW_HLF#7.3 Controller Mode Selection
33   */
34  if ((*rtu_WatchdogBrake) > NoBrake) {
35    /* UnaryMinus: '<Root>/UnaryMinus' */
36    rtb_UnaryMinus = -(*rtu_Deceleration);
37  } else {
38    rtb_UnaryMinus = *rtu_Acceleration;
39  }
40
41  /* End of Switch: '<Root>/Switch' */
```



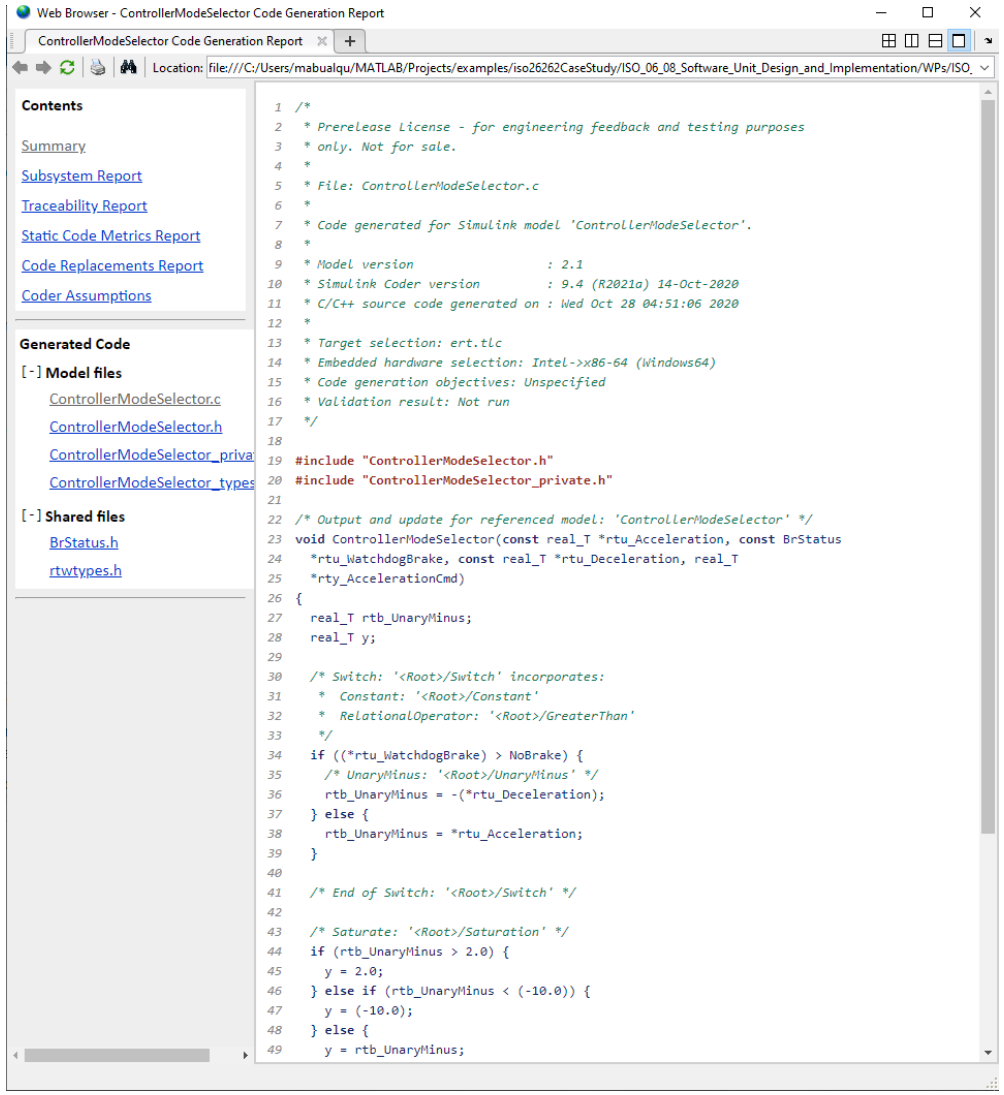


Reference Workflow for Certification – Code Verification



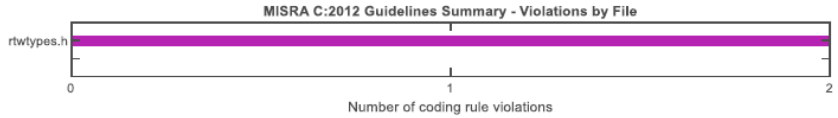
Reference Workflow for Certification – Code Verification

Static Code Verification for MISRA Compliance

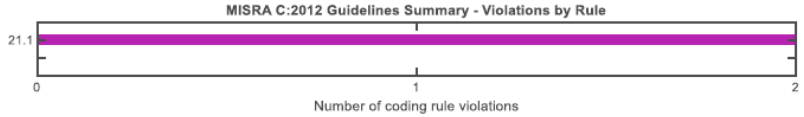


Chapter 1. MISRA C:2012 Guidelines

MISRA C:2012 Guidelines Summary - Violations by File



MISRA C:2012 Guidelines Summary - Violations by Rule



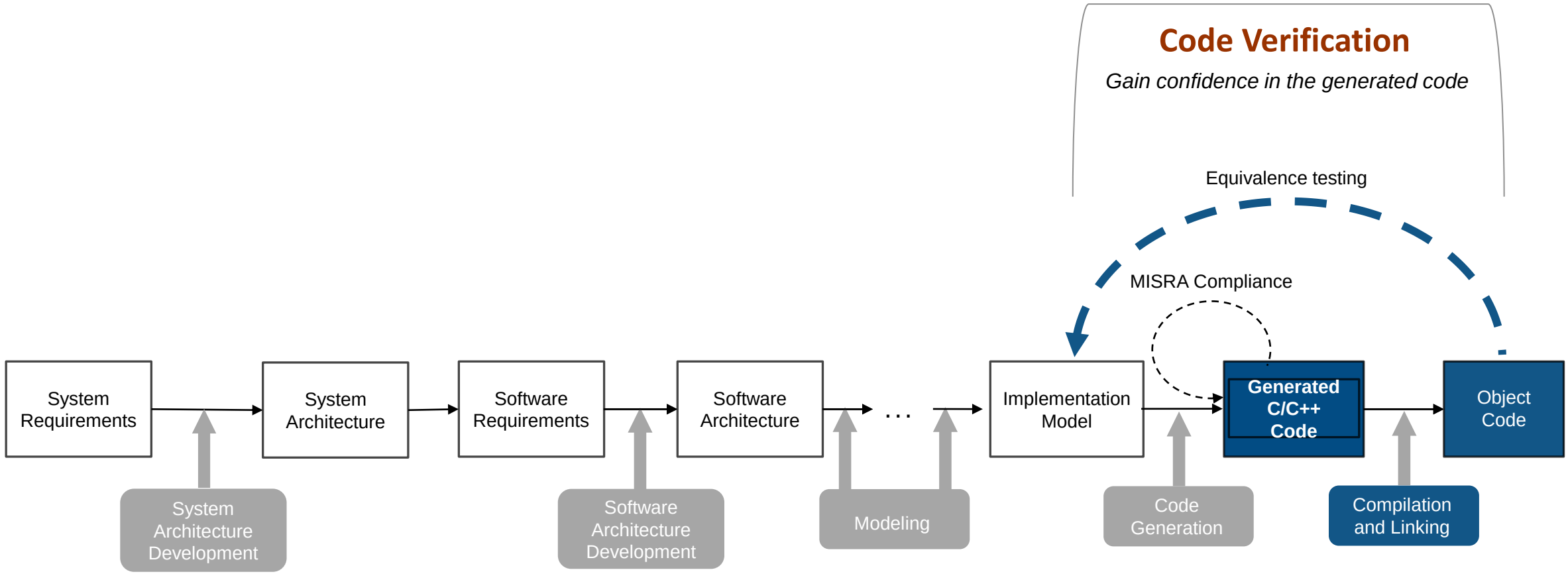
MISRA C:2012 Guidelines Summary for all Files

File	Total
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector.c	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector_private.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector_types.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert_sharedutils\BrStatus.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_6_8_5_2_Software_Unit_Implementation\slprj\ert_sharedutils\rtwtypes.h	2
Total	**2**

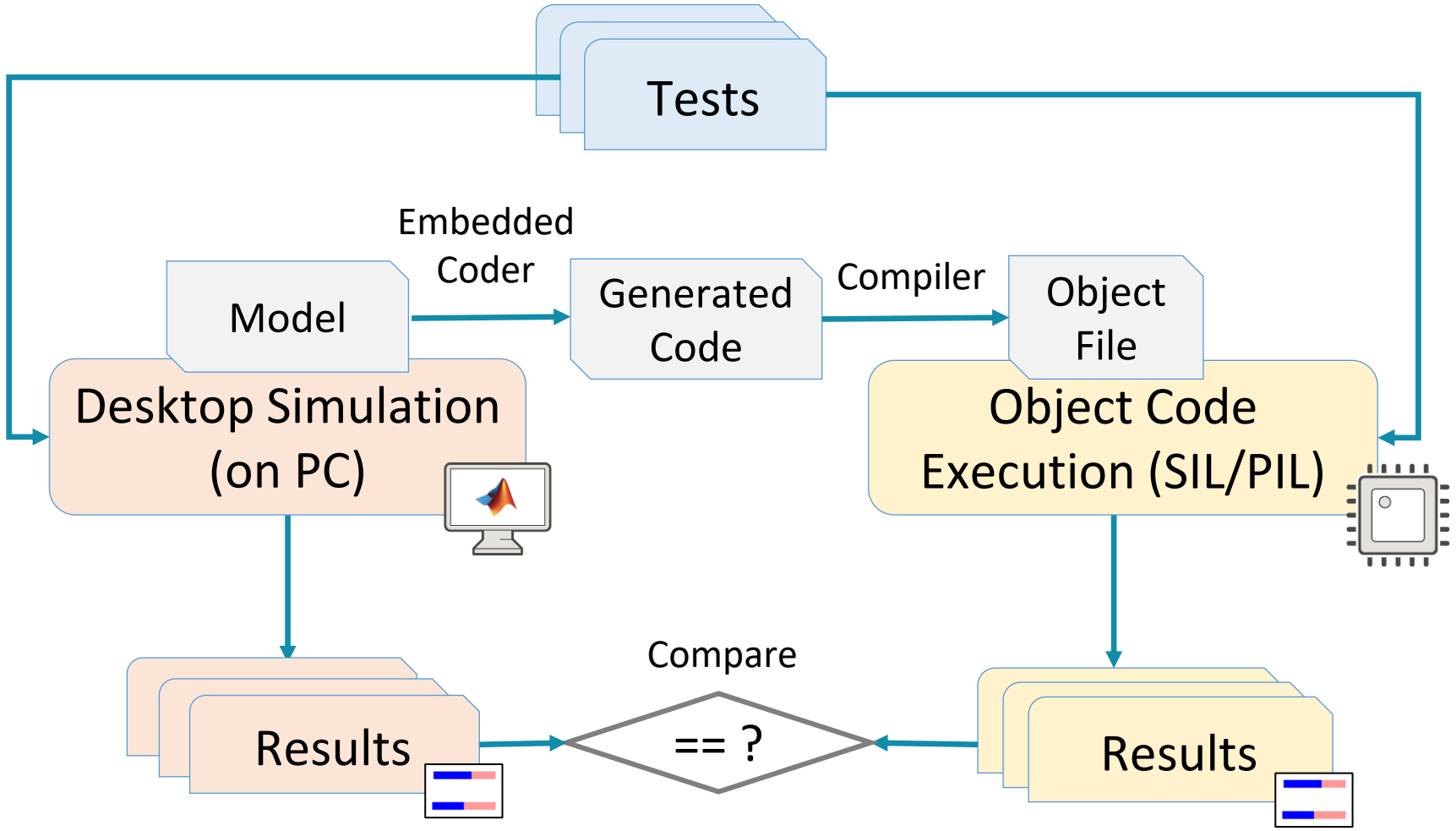
MISRA C:2012 Guidelines Summary for Enabled Guidelines

Guideline	Description	Mode	Total

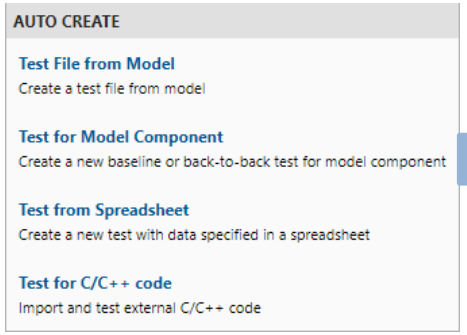
Model-Based Design Reference Workflow

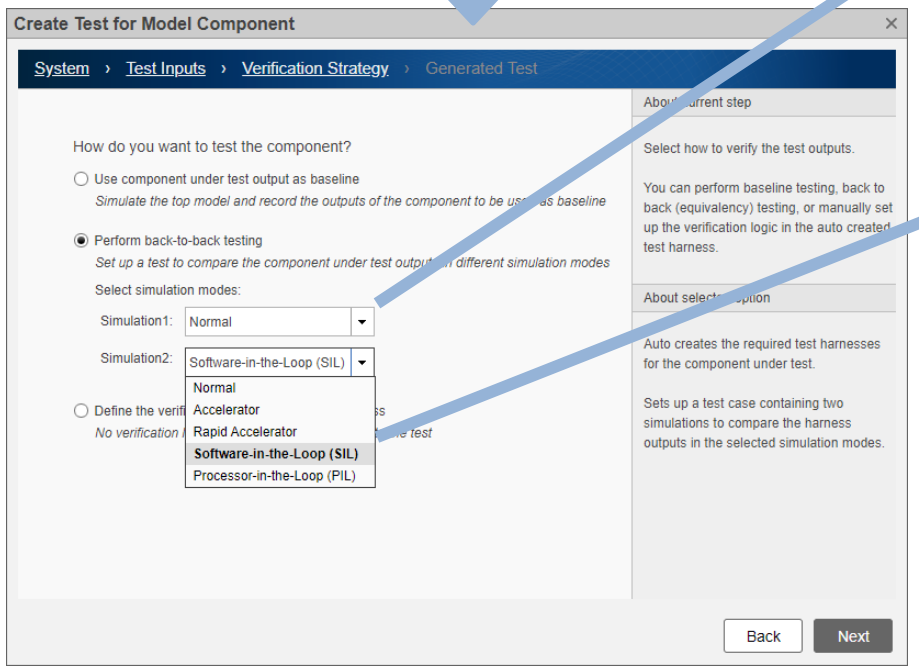


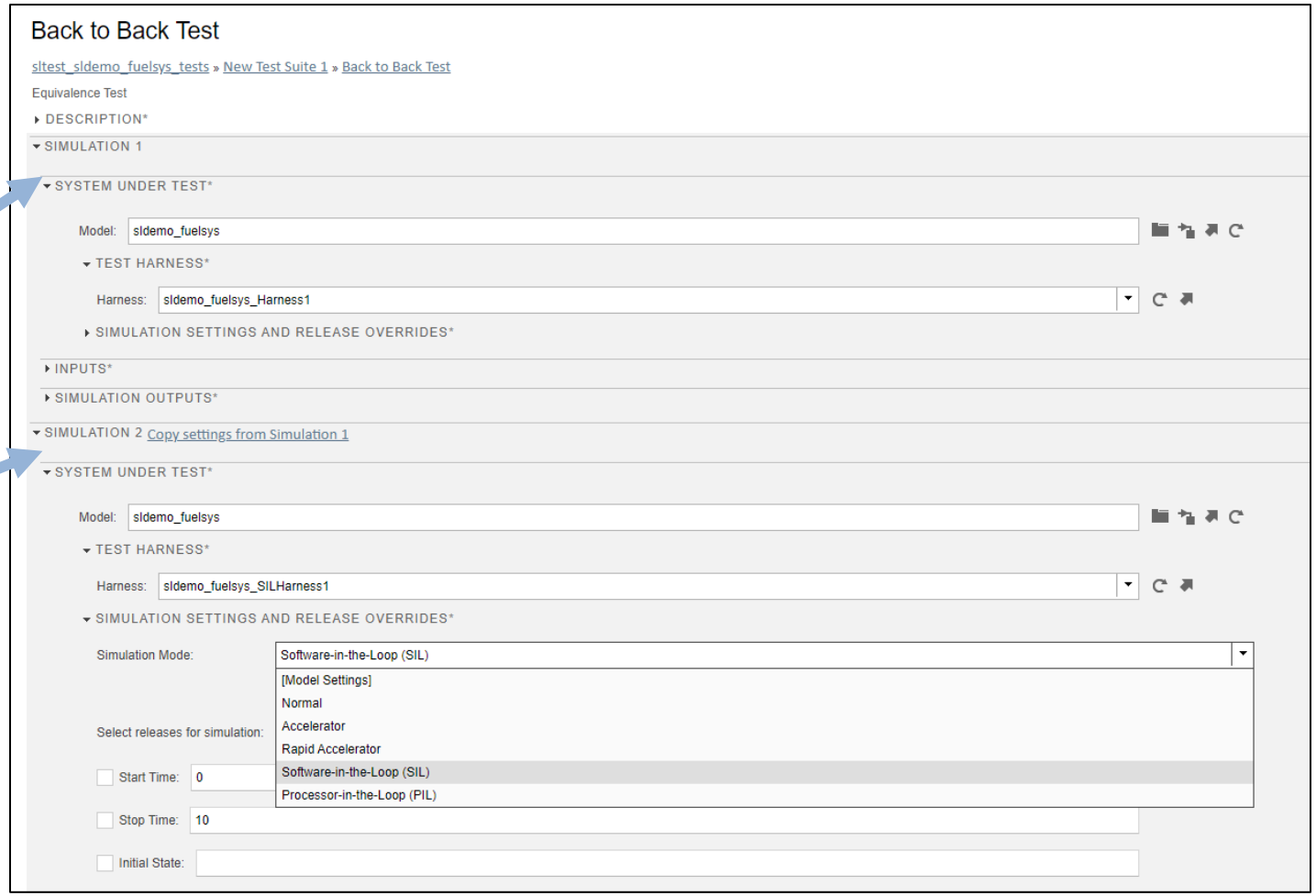
Back-to-Back Testing



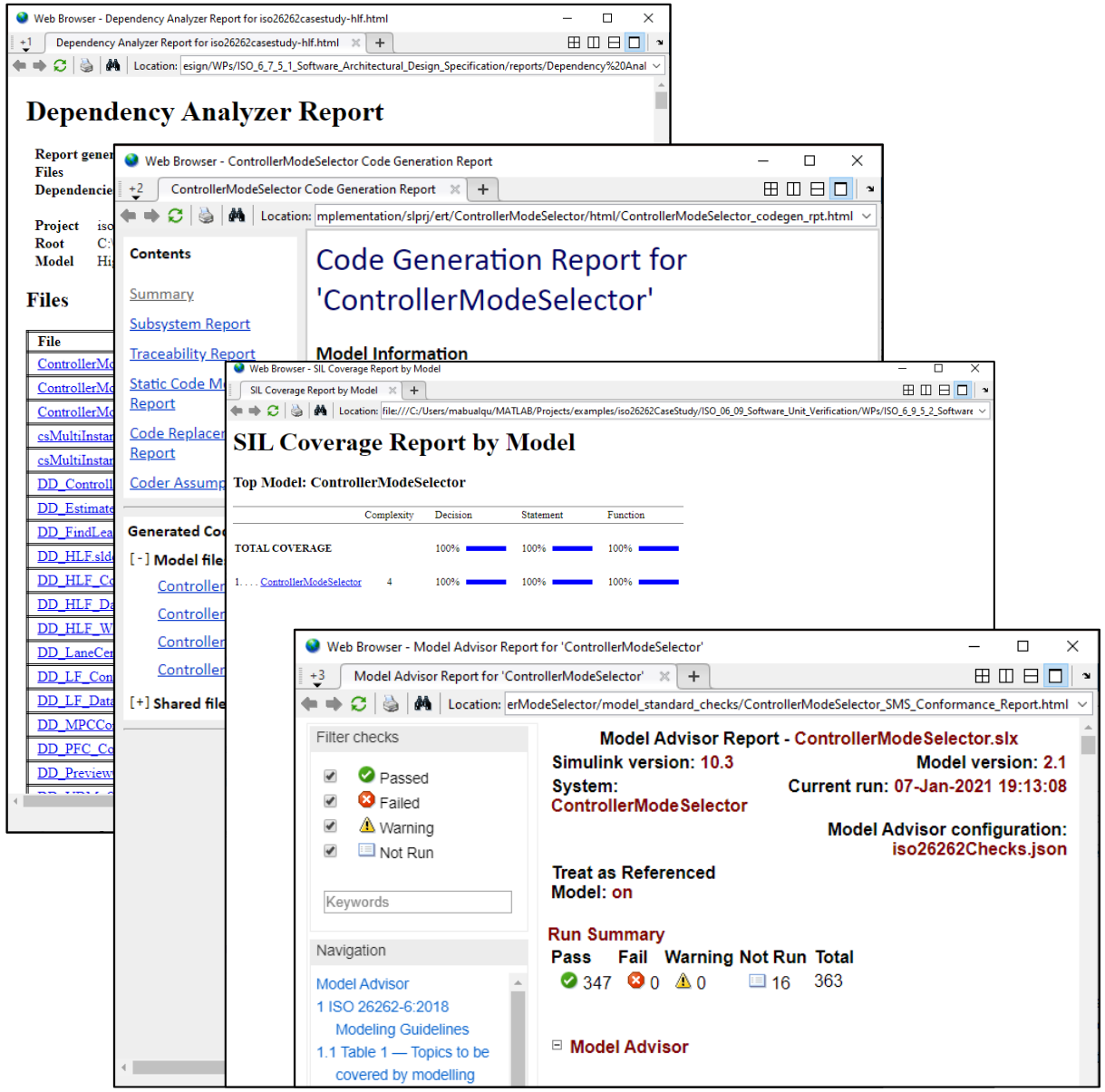
Automate Test Creation using Test Manager Wizard

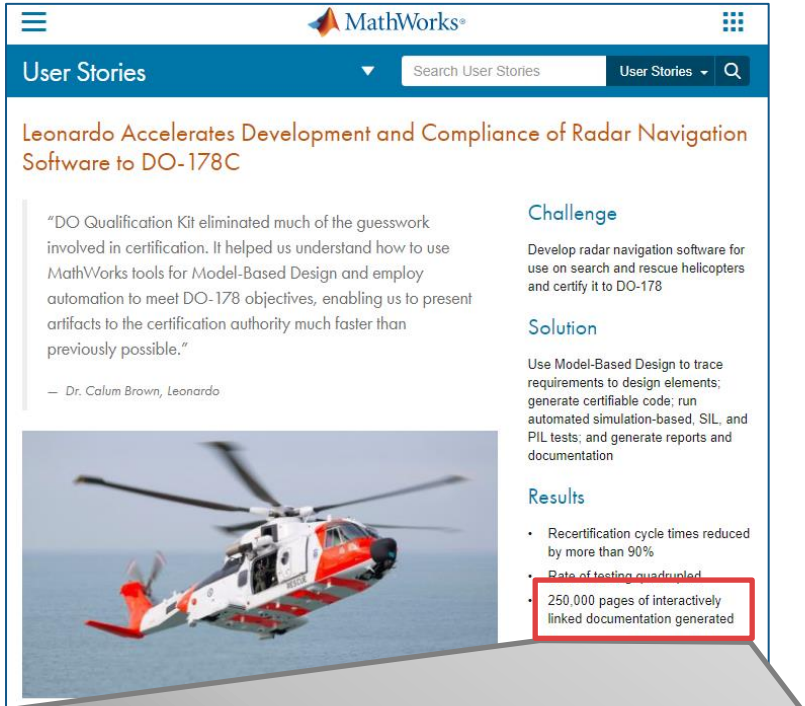






Generate Reports for Reviews and Audits





250,000 pages of interactively linked documentation generated. "Altogether, we generated the equivalent of around 250,000 pages of interactively linked test reports and other documentation using Simulink Test and Simulink Report Generator," notes Brown. "As the certification authority has *carte blanche* to review whatever they want, we felt it was easier to be able to provide evidence for everything we had done. If the DER wanted to see any particular result, it was available for them and fully linked to our model, which really built their trust."

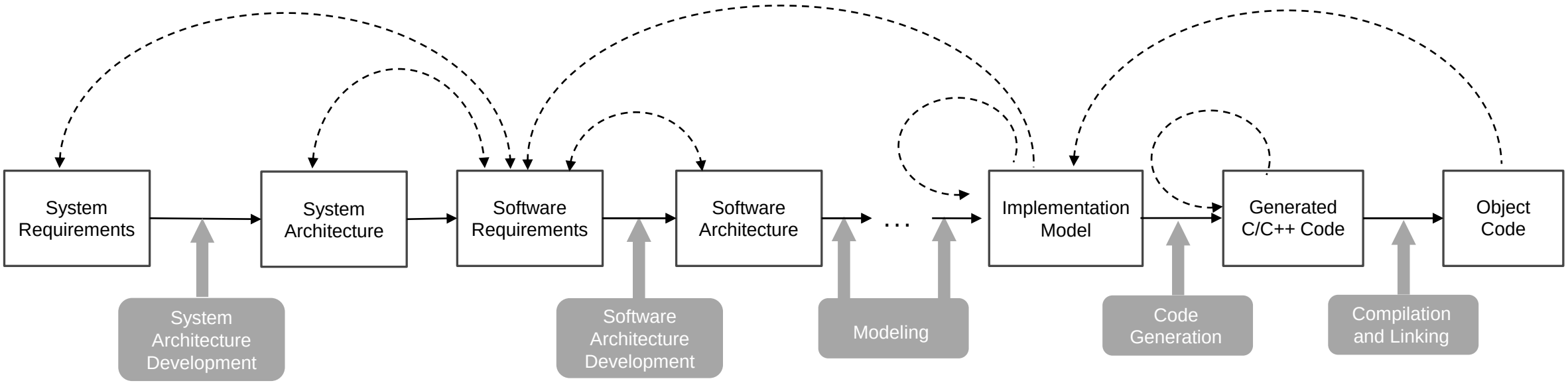
[Leonardo User Story]

Interactivity Poll: What capabilities presented today could help you improve your development? (choose all that apply)

- Requirements Traceability
- Architecture Analysis for safety
- Standards and Guideline Checking
- Functional Testing at Model level
- Measuring test coverage
- Dashboards and process management
- Automatic code generation
- Back-to-Back or Equivalence Testing
- Report generation
- Other (Enter in Chat)

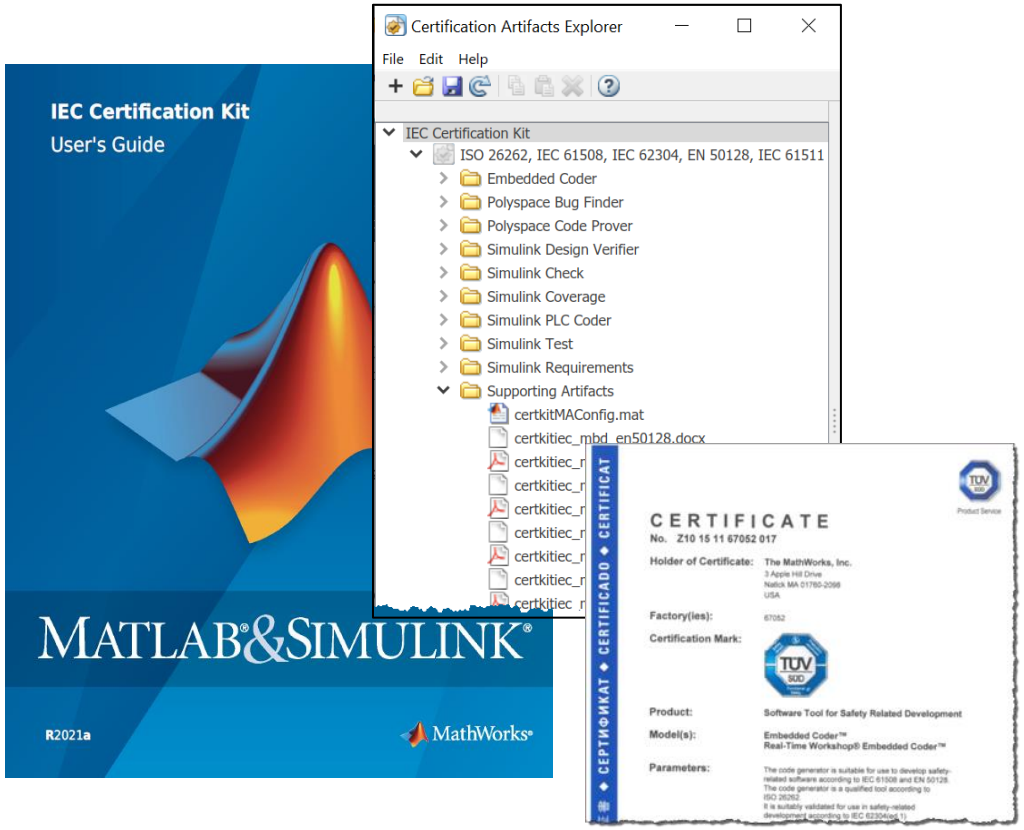
Summary: Reference Workflow with Model-Based Design

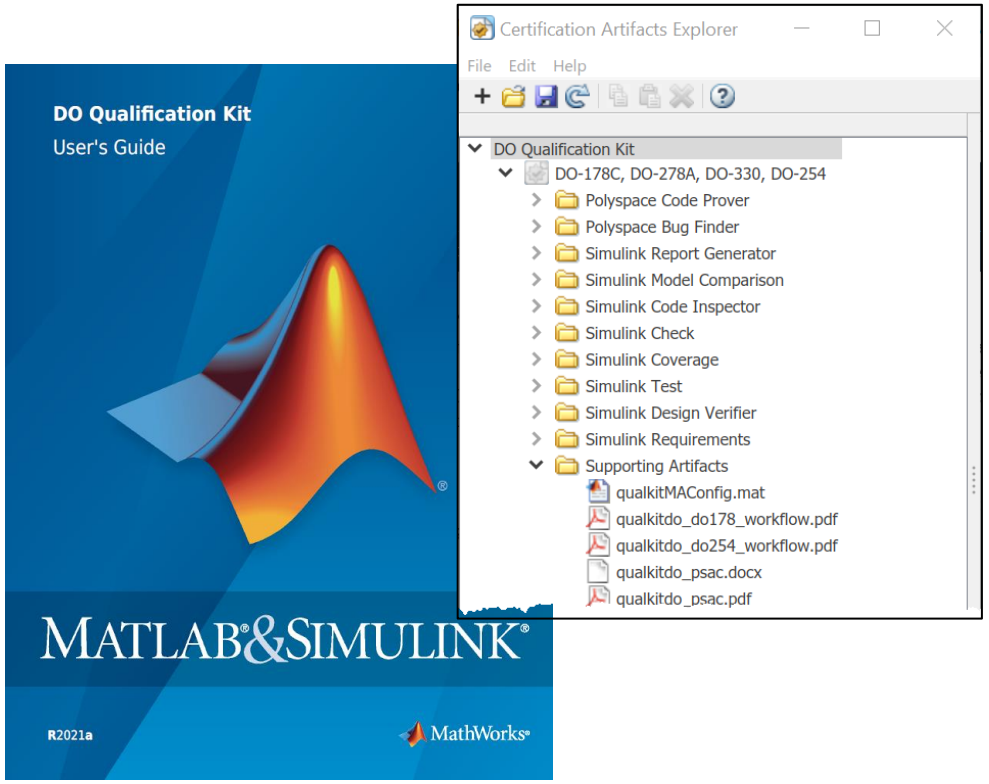
- Create traceability across requirements, architecture, design, test and code
- Detect design errors early by continuous testing
- Automatic code generation
- Generate documents and reports for reviews and audits



IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases, and procedures to help certify





Learn More

- Embedded Code Generation
- Verification, Validation, and Test Solution Page
- Verifying Models and Code for High-Integrity Systems
- Using Model Based Design in the ISO 26262:2018 Case Study
- Helicopter Flight Control: A Model-Based Design Example for DO-178C and DO-331

MATLAB EXPO 2021

Thank you