Meet Certification Standards with Automated Requirements Based Testing

Gaurav Ahuja
Application Engineer

Aravind Singh
Senior Application Engineer

Challenge to Deliver Complex Systems and Meet Standards

- Need to meet industry or customer's standards
  - DO-178C (Aero), ISO 26262 (Auto), IEC 62304 (Medical), IEC 61508 (Industrial), MISRA, etc.
- Time and cost for safety critical projects estimated 20-30 times more costly*
- Finding defects late increases cost and time

*Source: [Certification Requirements for Safety-Critical Software]

ISO 26262-6:2018 notes Simulink and Stateflow as Suitable for Software Architecture, Design and as basis for Code Generation

Table 5 — Notations for software unit design

	Notations	ASIL A	ASIL B	ASIL C	ASIL D
1a	Natural language[a]	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations[b]	+	+	++	++
1d	Formal notations	+	+	+	+

[a] Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or provide an explanation and rationale for decisions captured in the notations.

EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

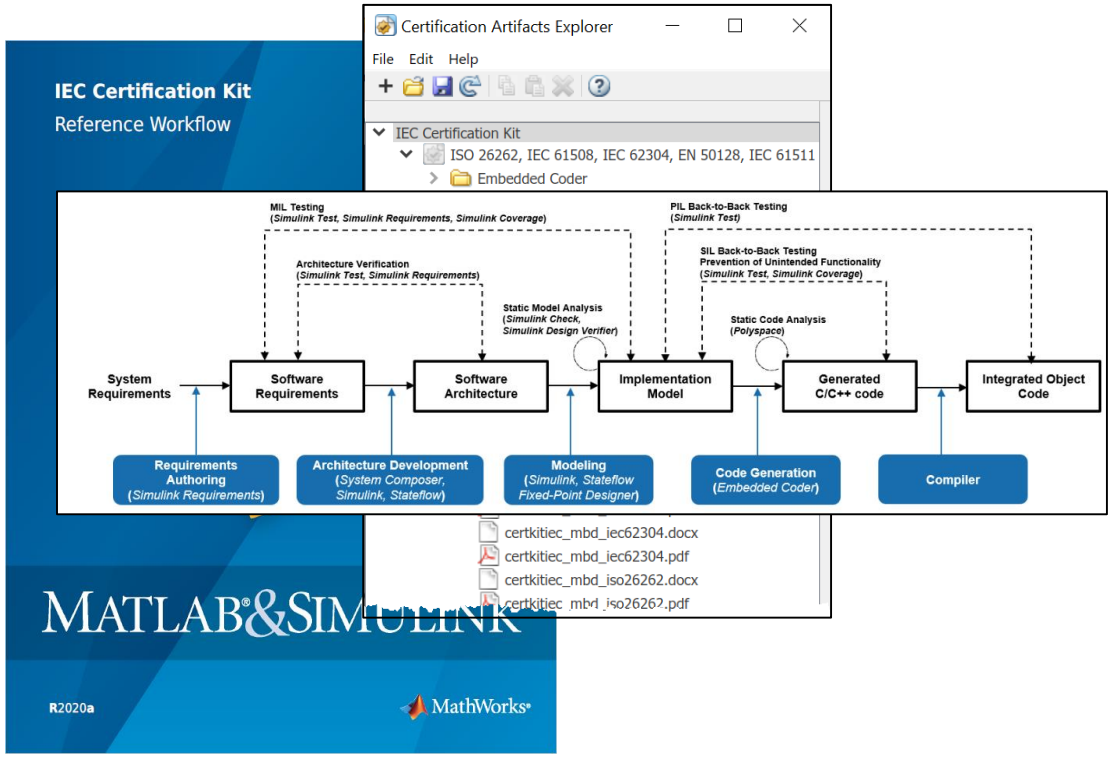
NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

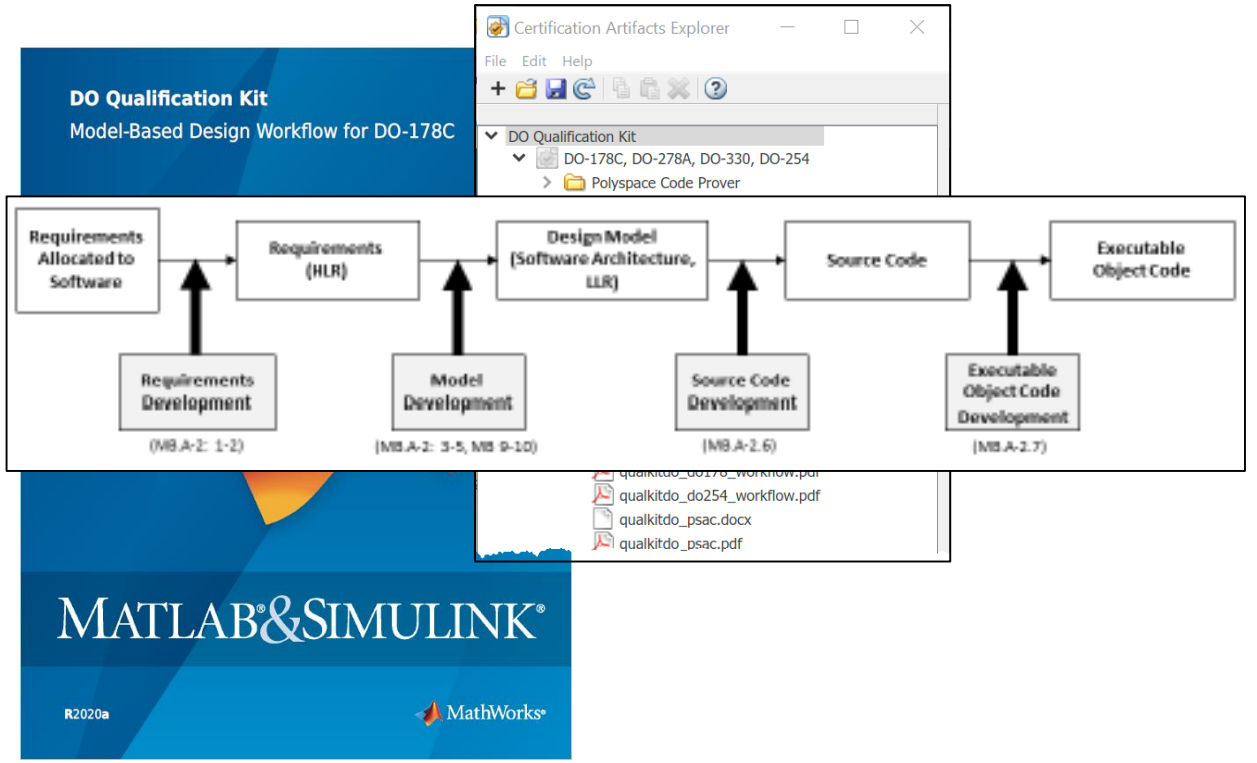
NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design are applied to the model which serves as the basis for the code generation.

Table 2 Software Architecture Design Notations has similar suitability wording for use of Simulink and Stateflow

Qualify tools with IEC Certification Kit and DO Qualification Kit

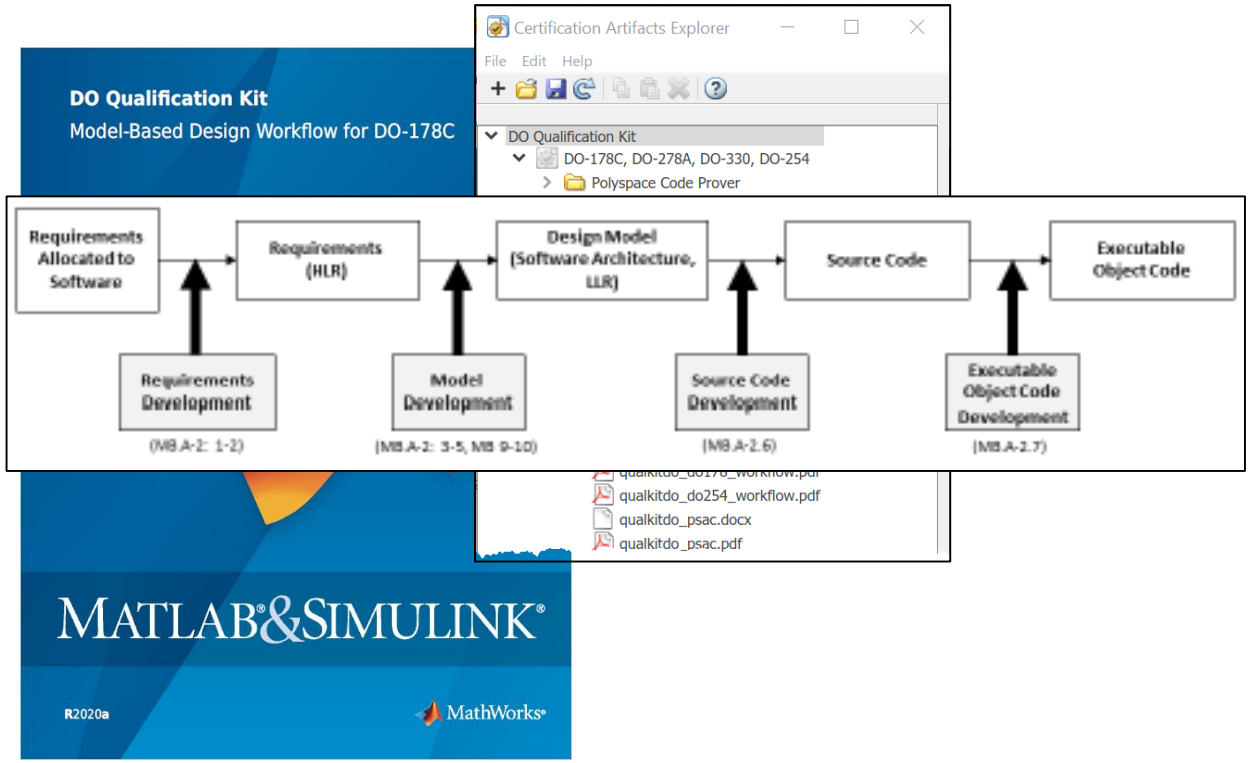
- Qualify code generation and verification products
- Includes documentation, test cases and procedures





Qualify tools with IEC Certification Kit and DO Qualification Kit

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



Qualify tools with IEC Certification Kit and DO Qualification Kit

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

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Desian

Kostal's electronic steering column lock module.

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Primary flight control computers from BAE Systems.

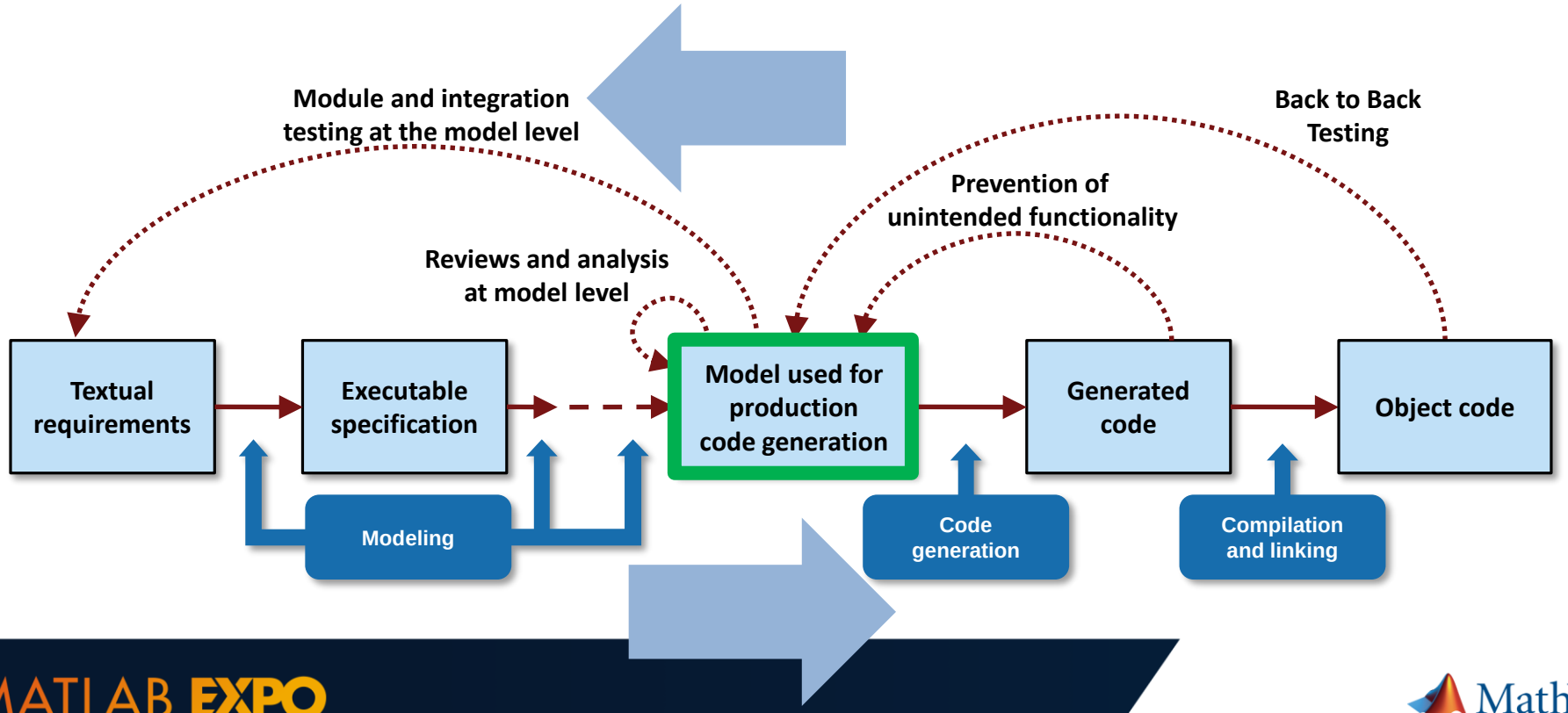
Conform to Certification Standards with Reference Workflow

Model Verification

Discover design errors at design time

Code Verification

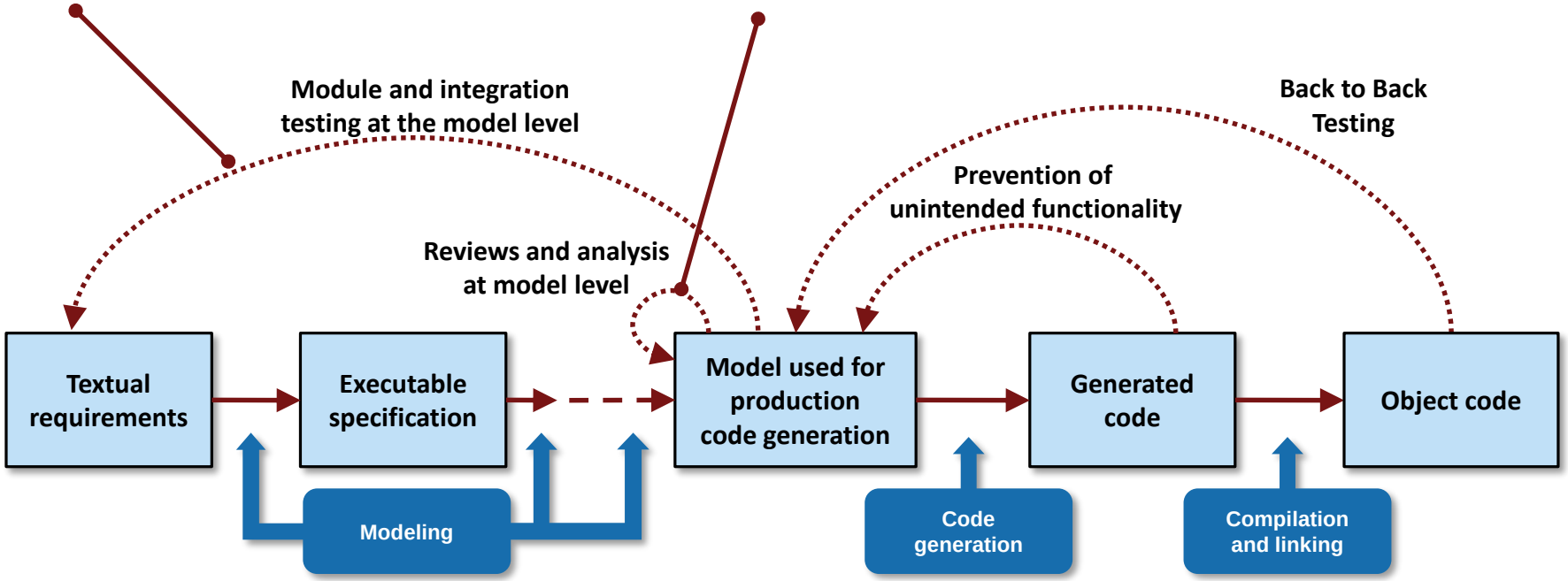
Gain confidence in the generated code



Model Verification: Discover design errors at design time

Model Verification

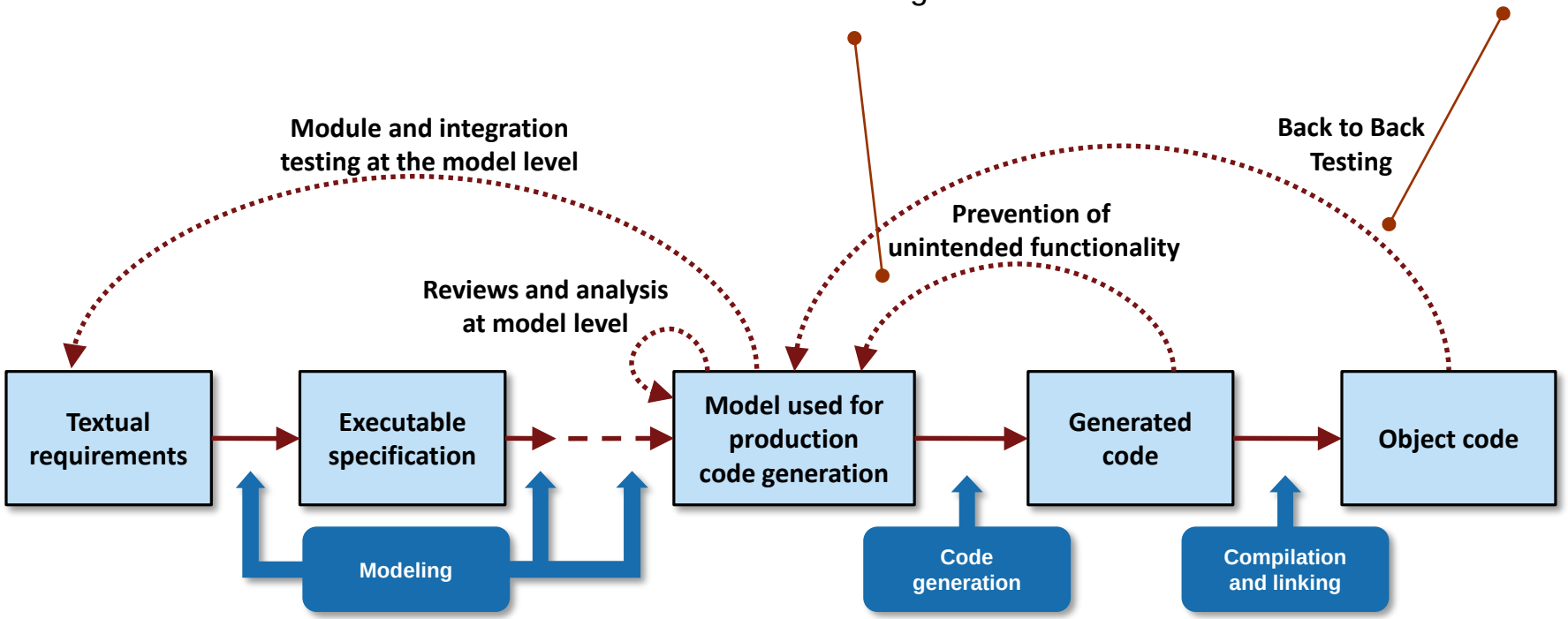
- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance



Code Verification: Gain Confidence in the Generated Code

Code Verification

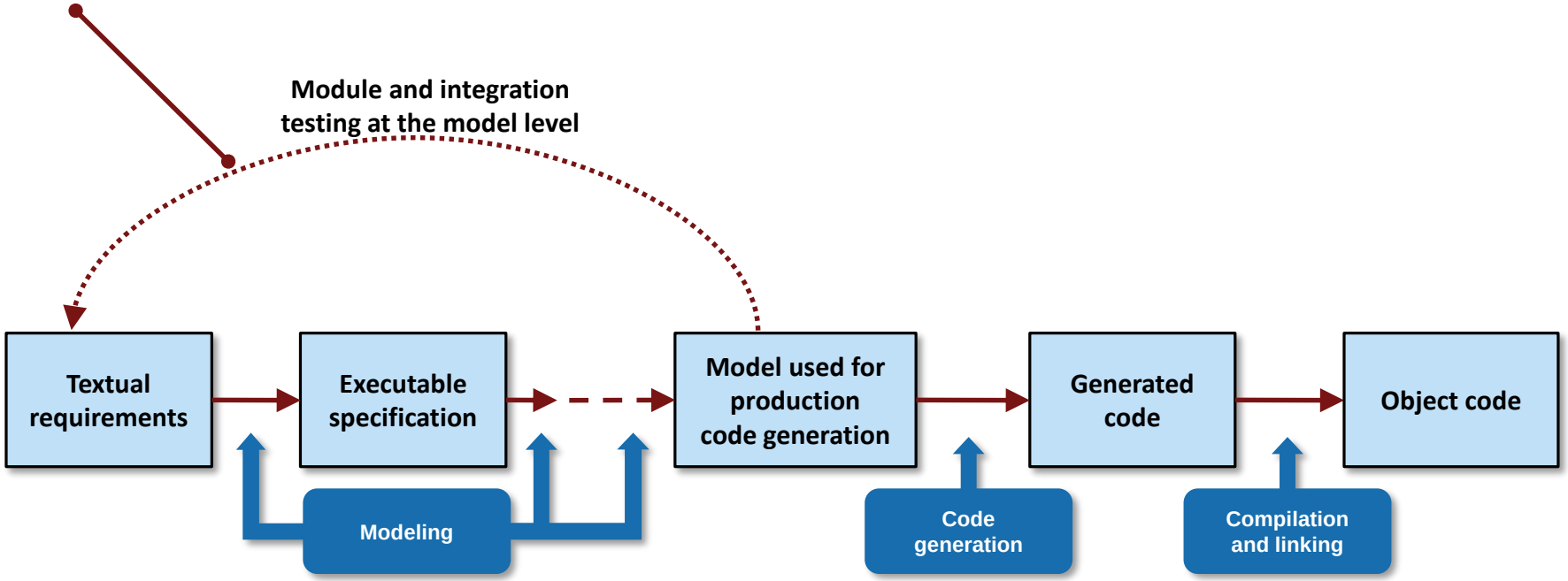
- Trace code to model and requirements
- Measure code coverage
- SIL/PIL equivalence testing
- Generate 100% coverage test vectors



Manage Requirements

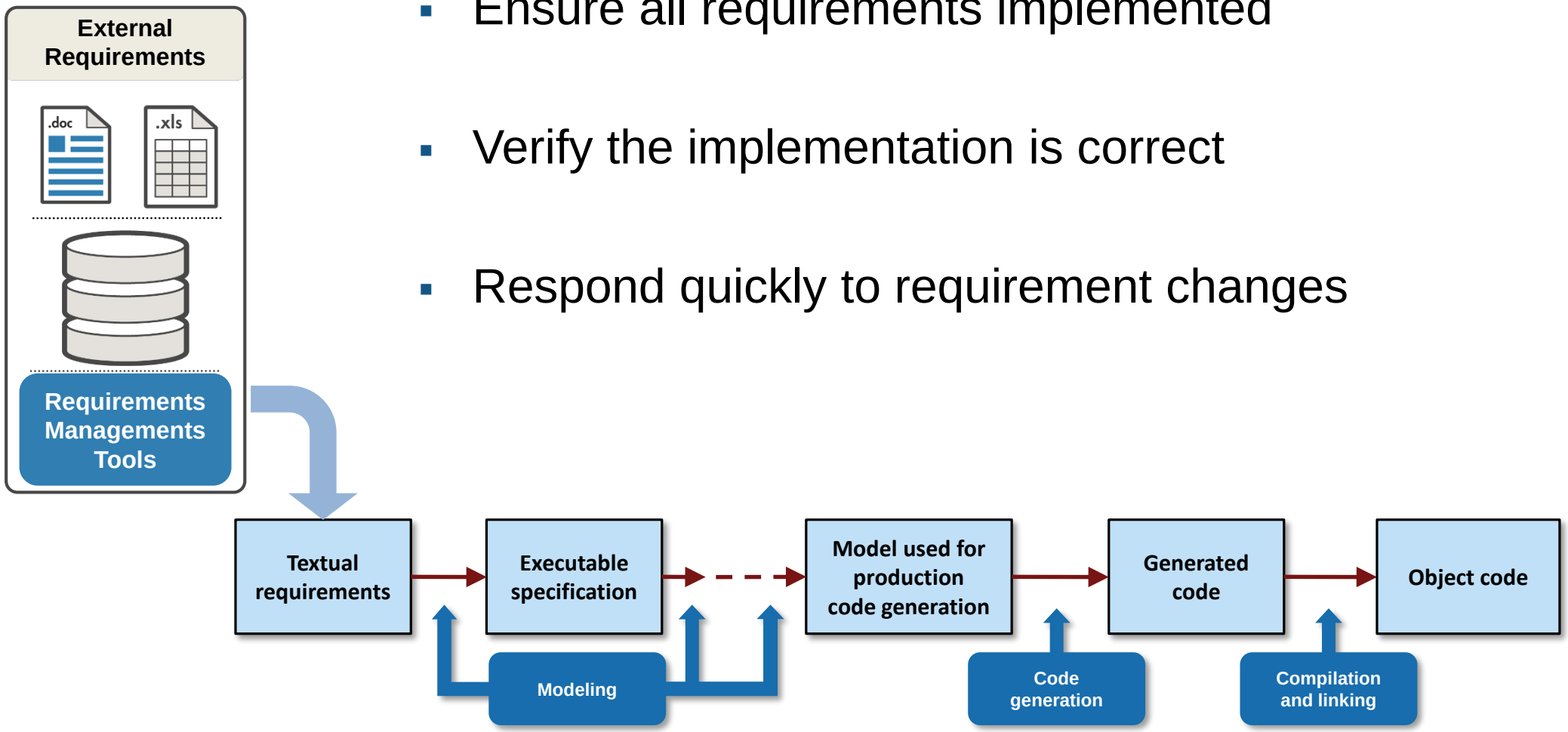
Model Verification

- **Manage requirements**
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

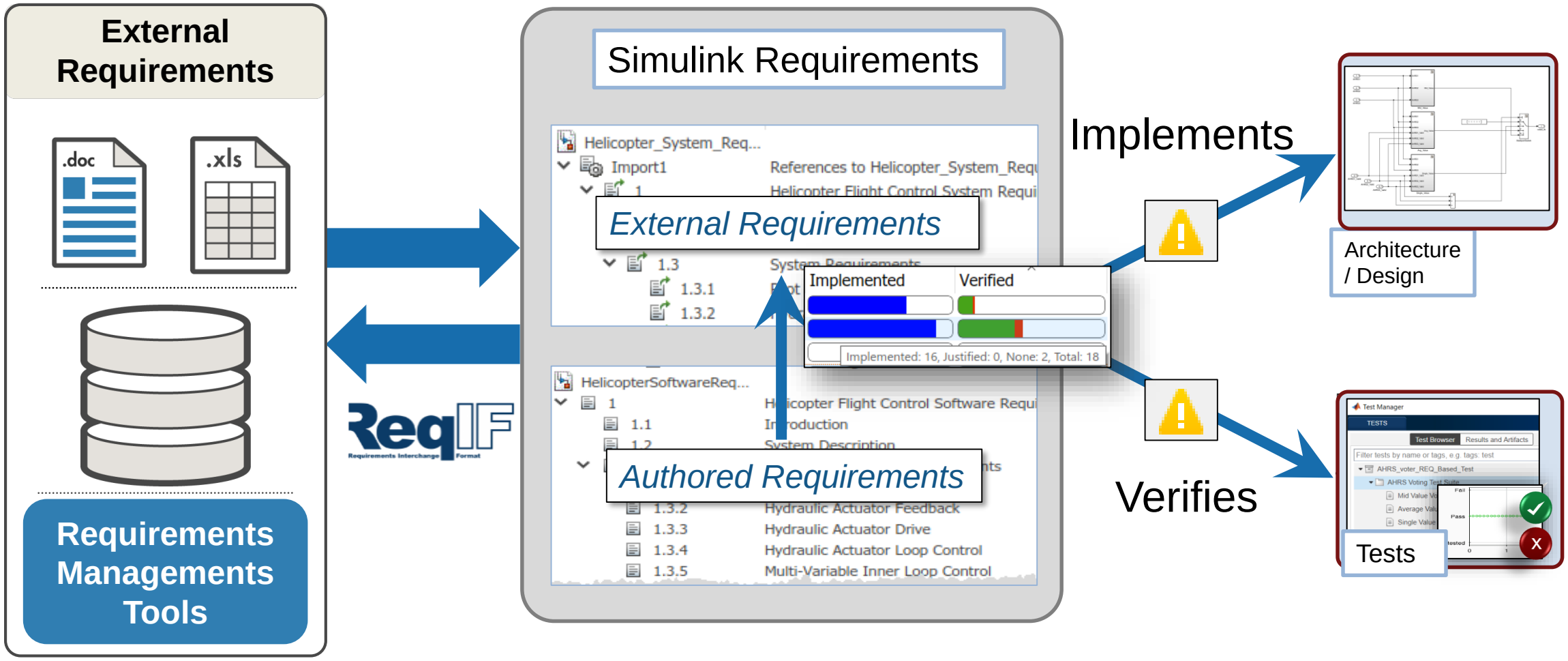


Manage Requirements

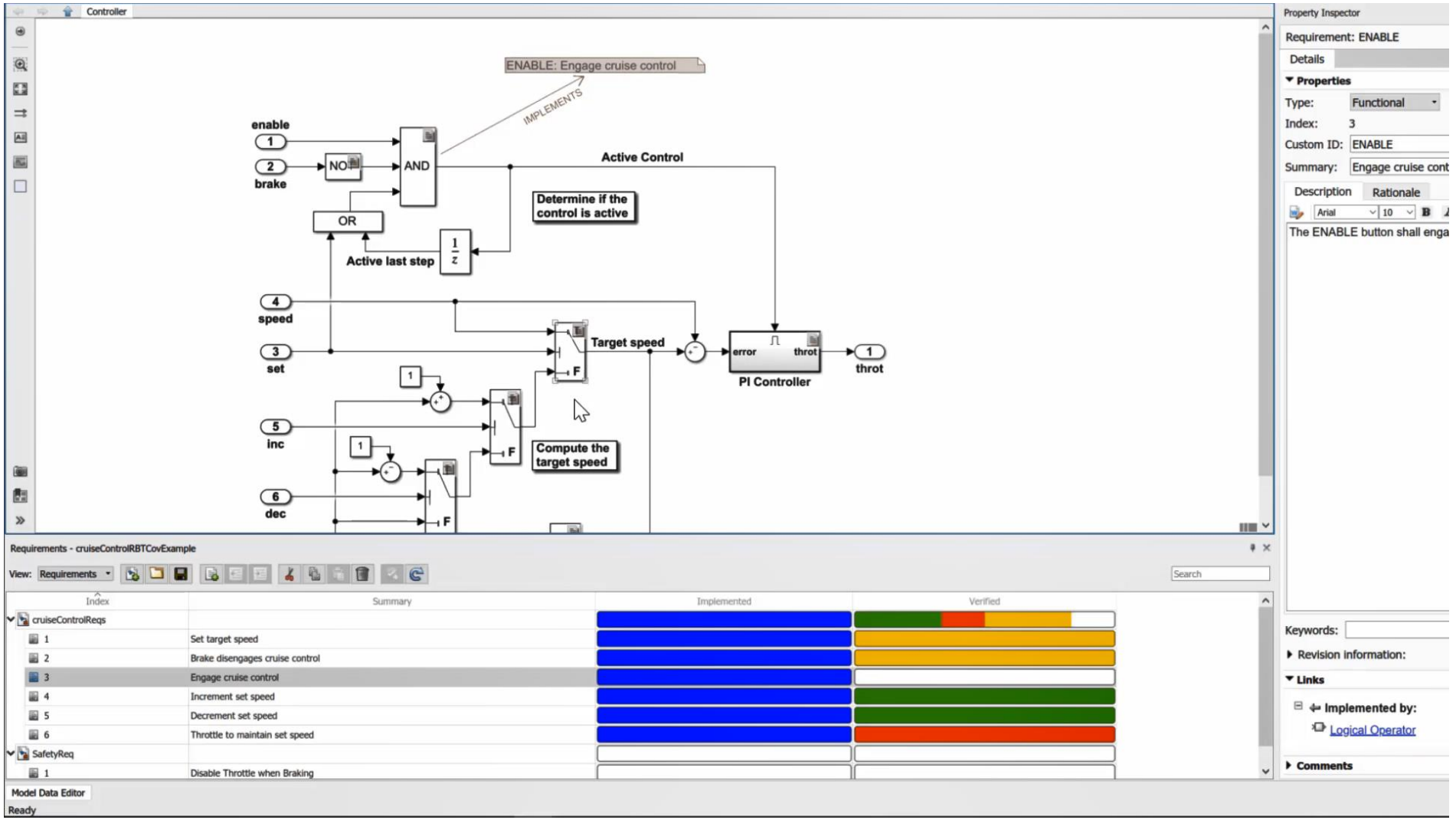
- Ensure all requirements implemented
- Verify the implementation is correct
- Respond quickly to requirement changes



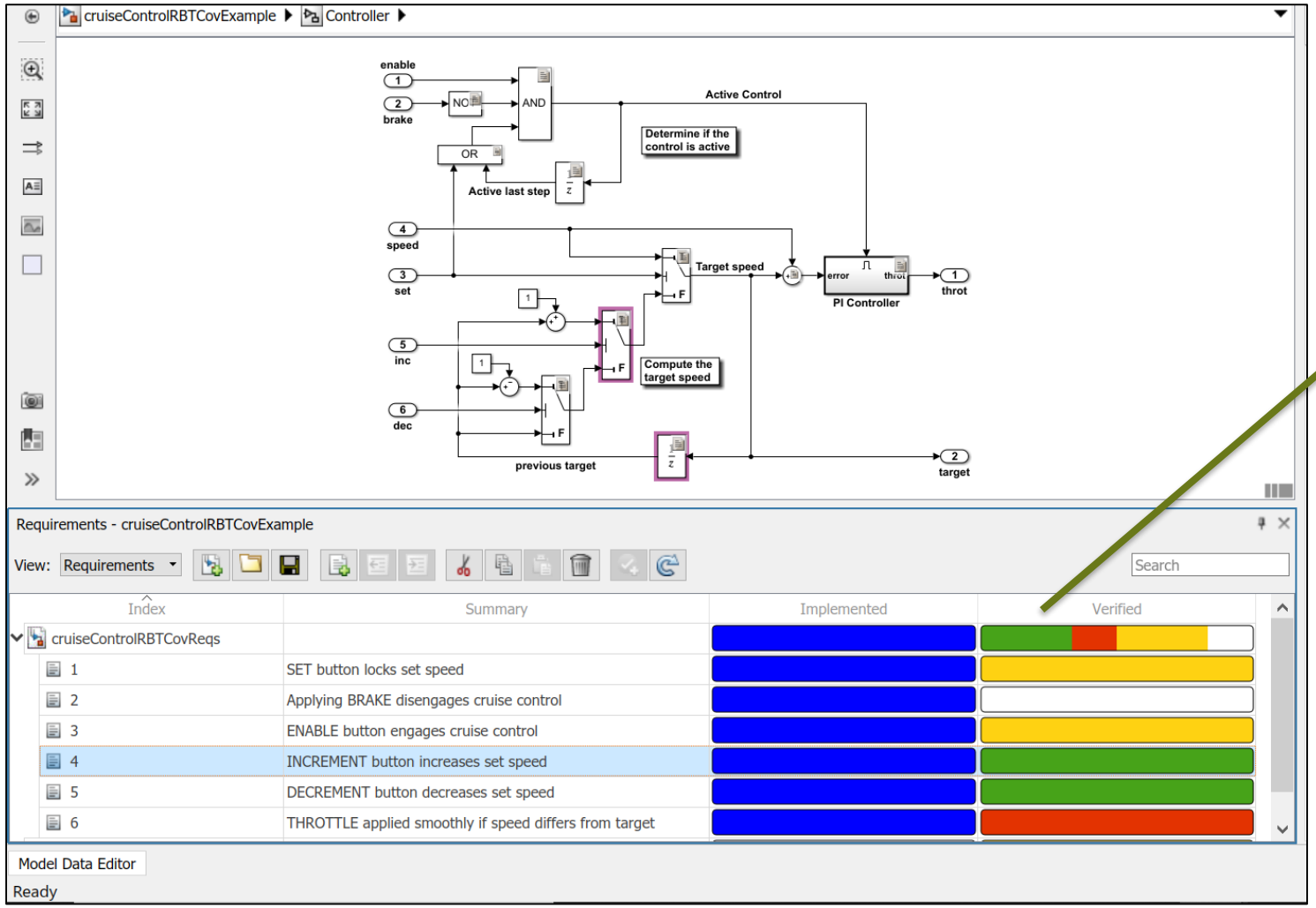
Work with Requirements, Architecture and Design Together

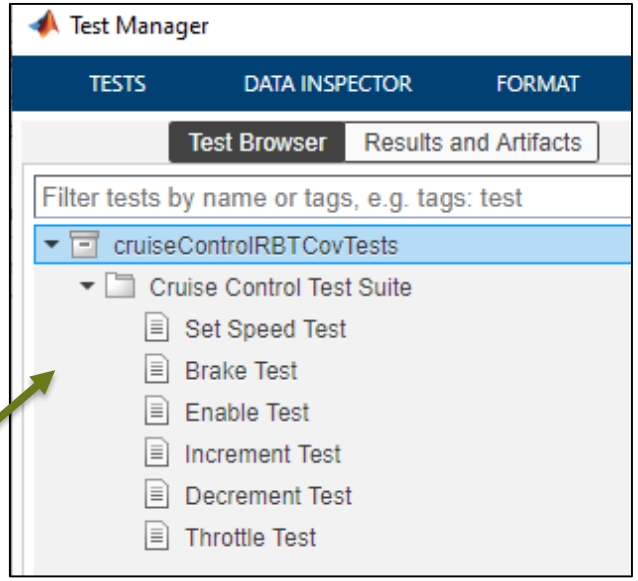


Demo: Requirements Perspective



Test and Requirements Traceability

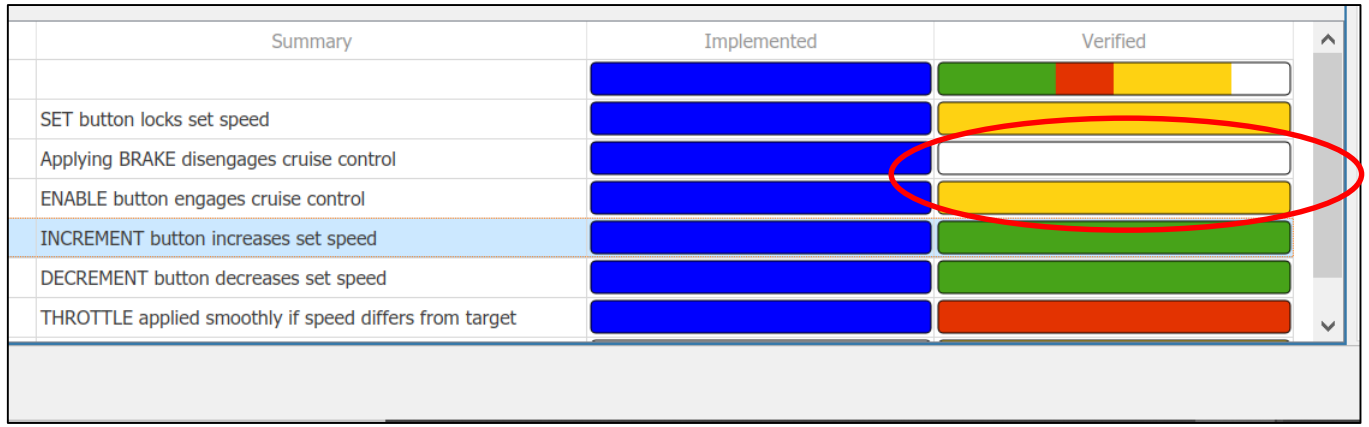




Verification Status

- Passed
- Failed
- Unexecuted
- Missing

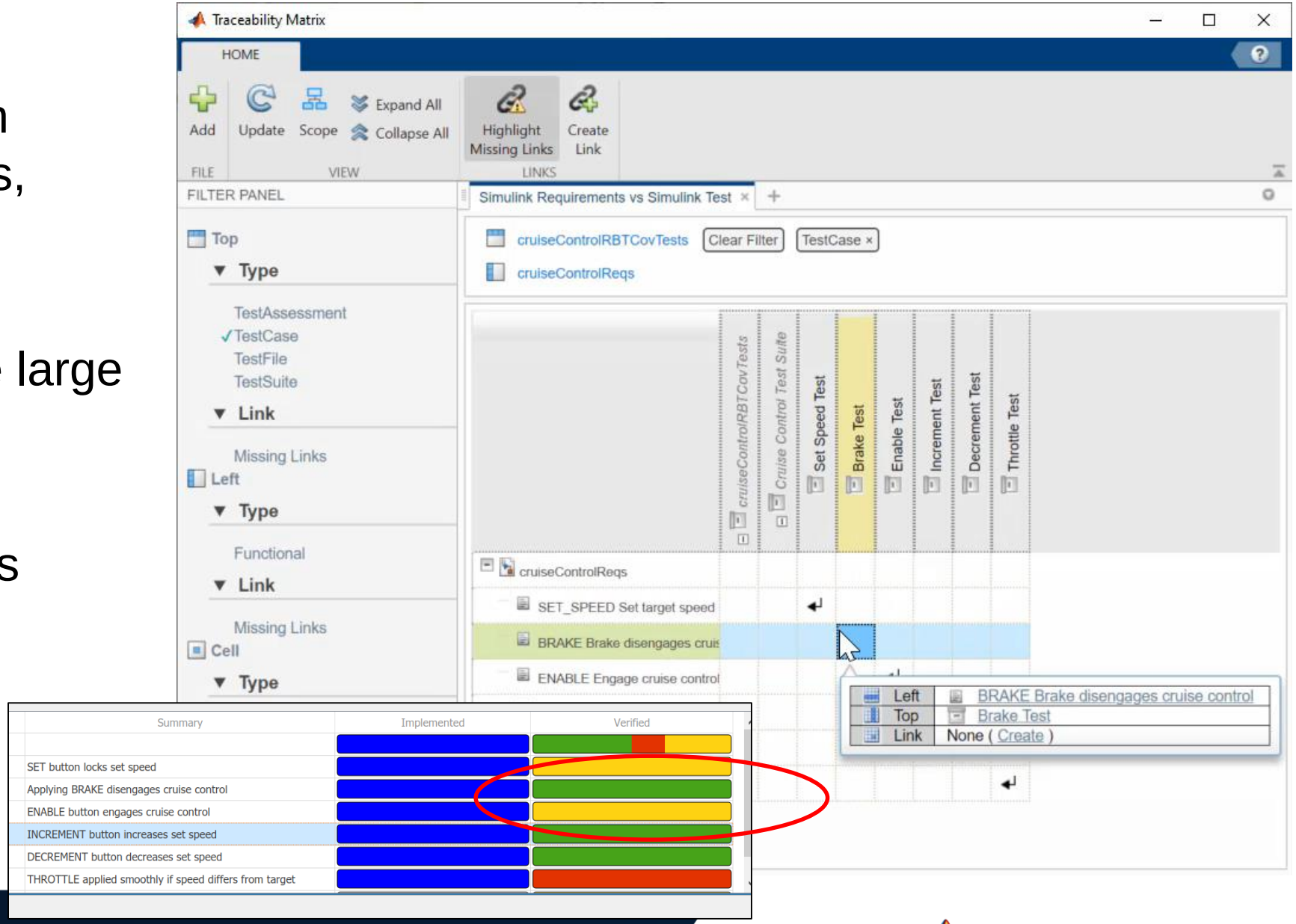
Review and Analyze Traceability with Traceability Matrix



Requirement is missing
link to Test Case

Review and Analyze Traceability with Traceability Matrix

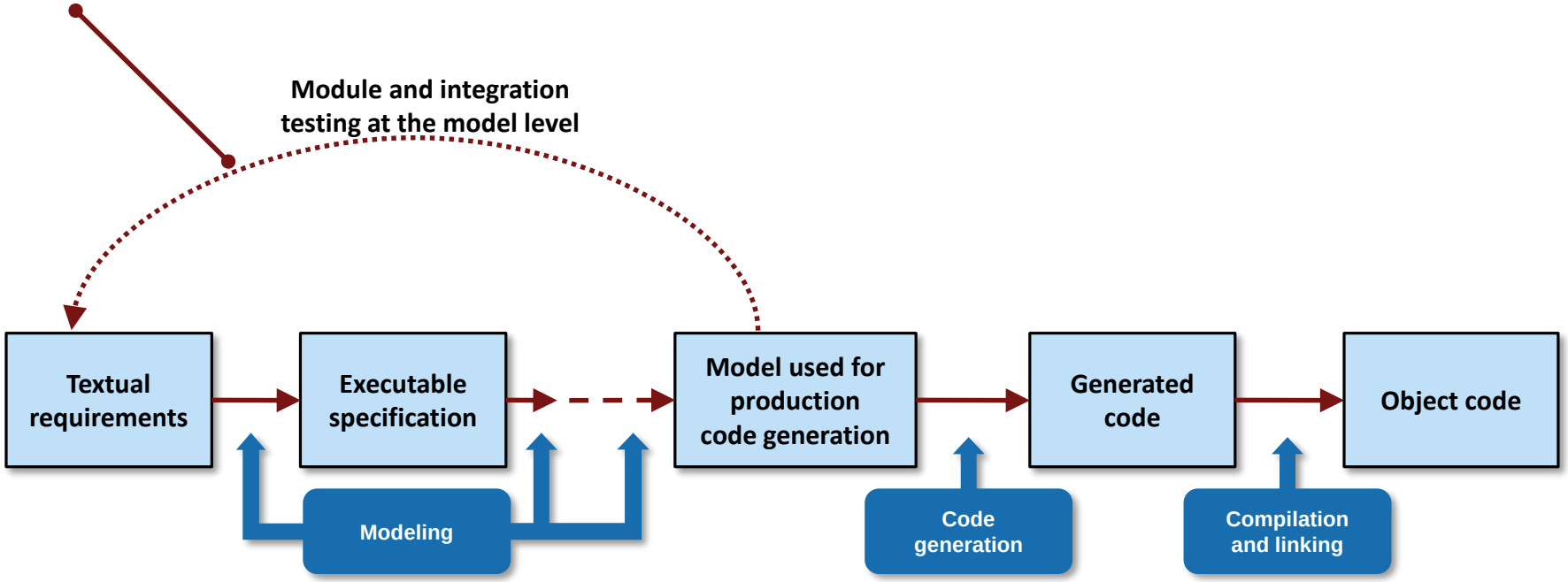
- Review links between different requirements, model, test
- Filter view to manage large sets of artifacts
- Highlight missing links
- Directly add links to address gaps



Systematic Functional Testing of Model

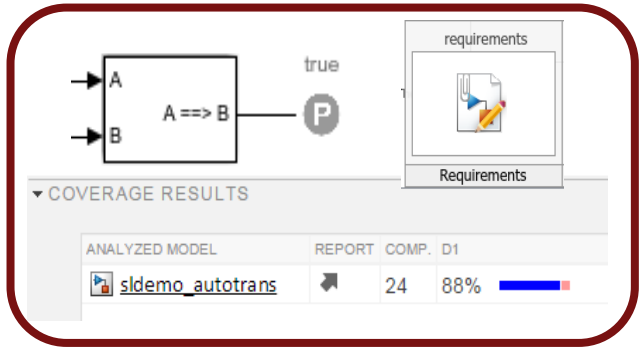
Model Verification

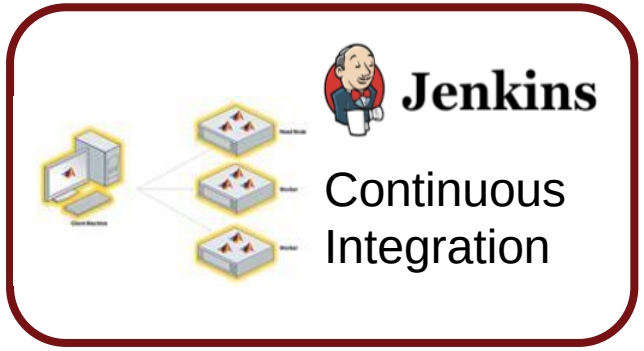
- Manage requirements
- **Systematically test**
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

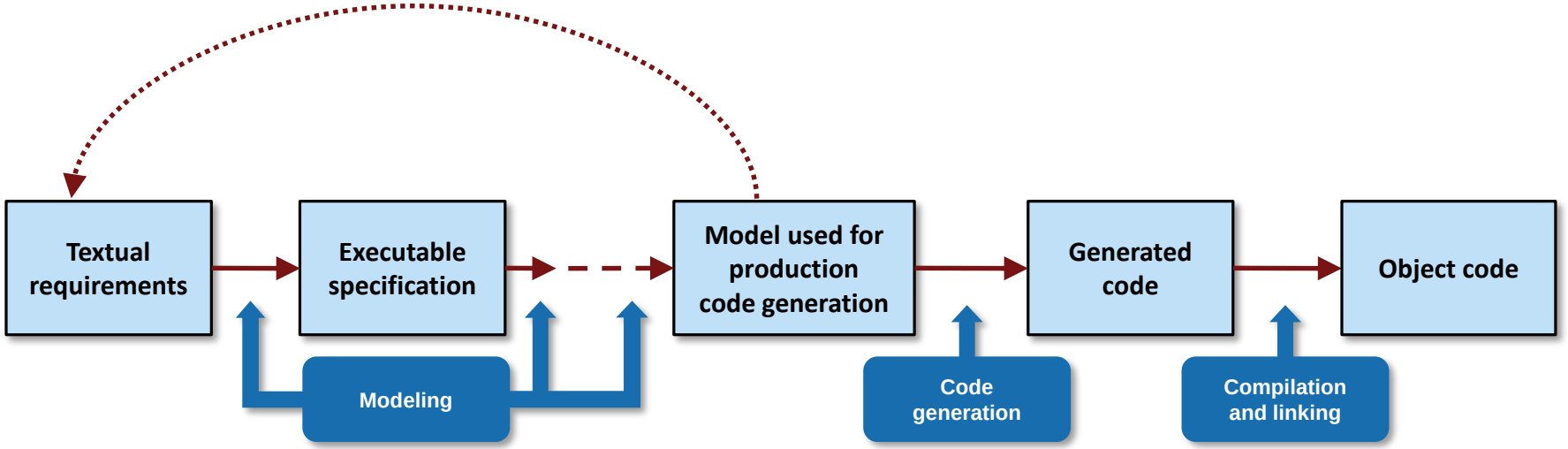


Automate functional testing with Simulink Test

Systematic, **integrated**, and **automated** functional testing solution





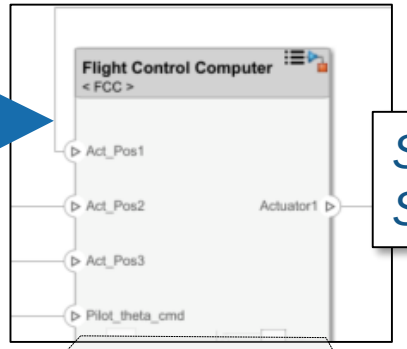


Requirements Based Verification with Simulink Test

FUNCTIONAL REQUIREMENTS
The flight control system shall ...

Implemented By

Verified By



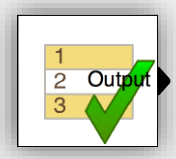
System Composer / Simulink / Stateflow

Test Case

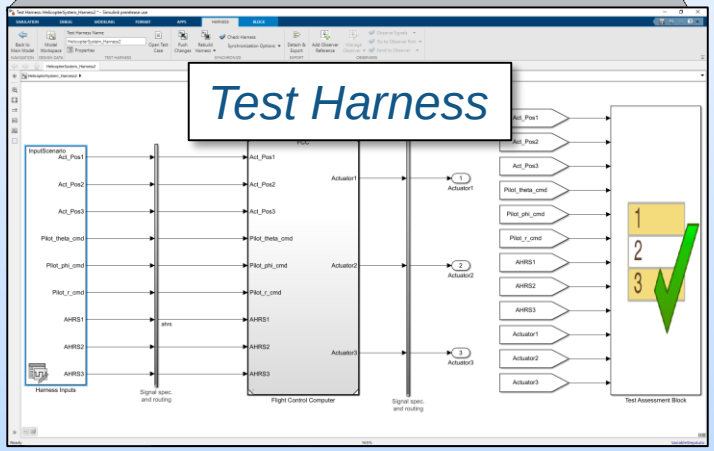
Inputs

MAT / Excel file (input)

Signal Editor



Test Sequence

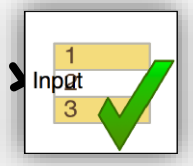


Test Harness

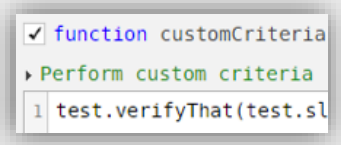
Simulink Test

Assessments

MAT / Excel File (baseline)



Test Assessments

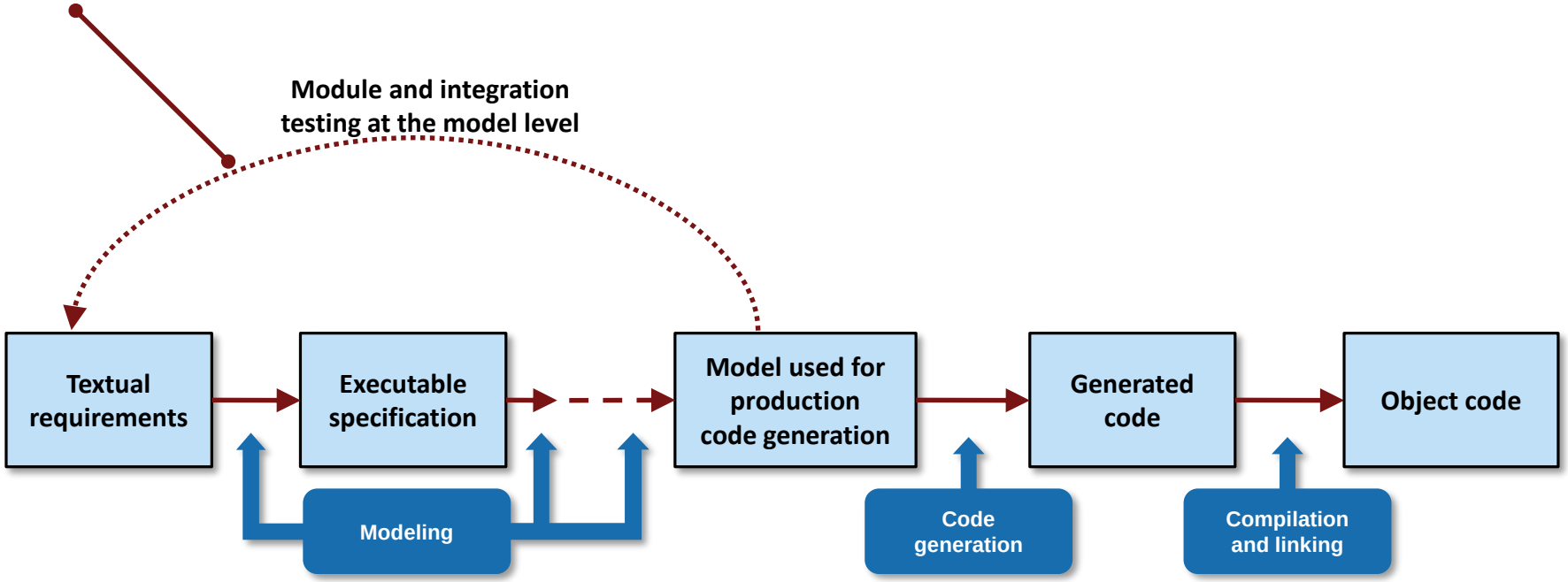


MATLAB Unit Test

Measure completeness of testing

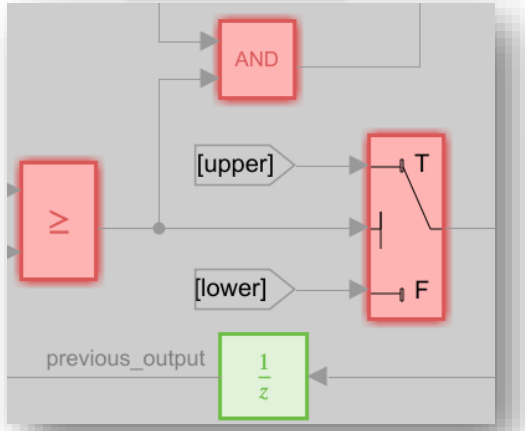
Model Verification

- Manage requirements
- Systematically test
- **Measure model coverage**
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

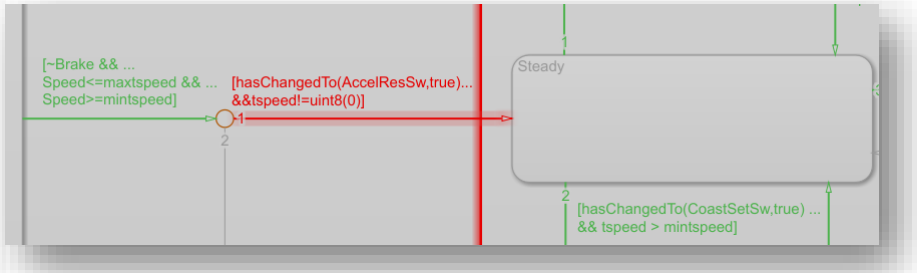


Coverage Analysis to Measure Testing

Simulink

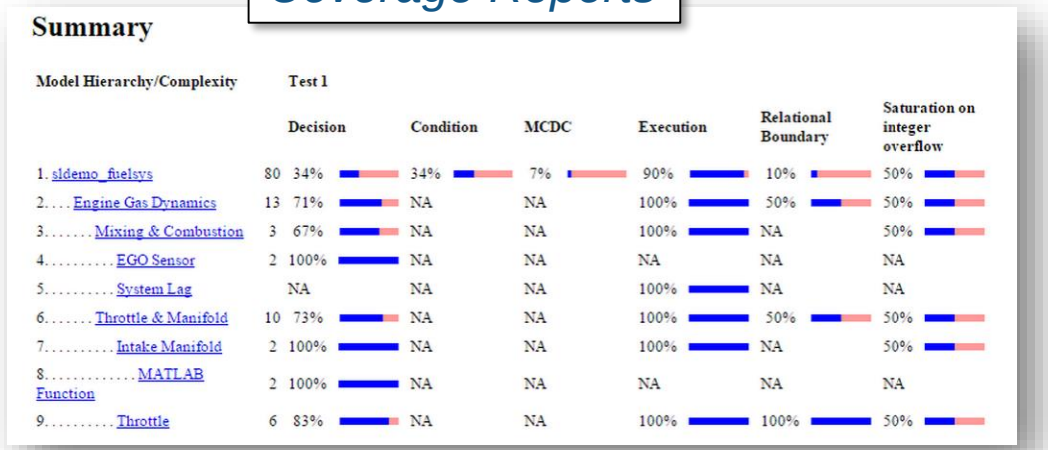


Stateflow



Code

Coverage Reports

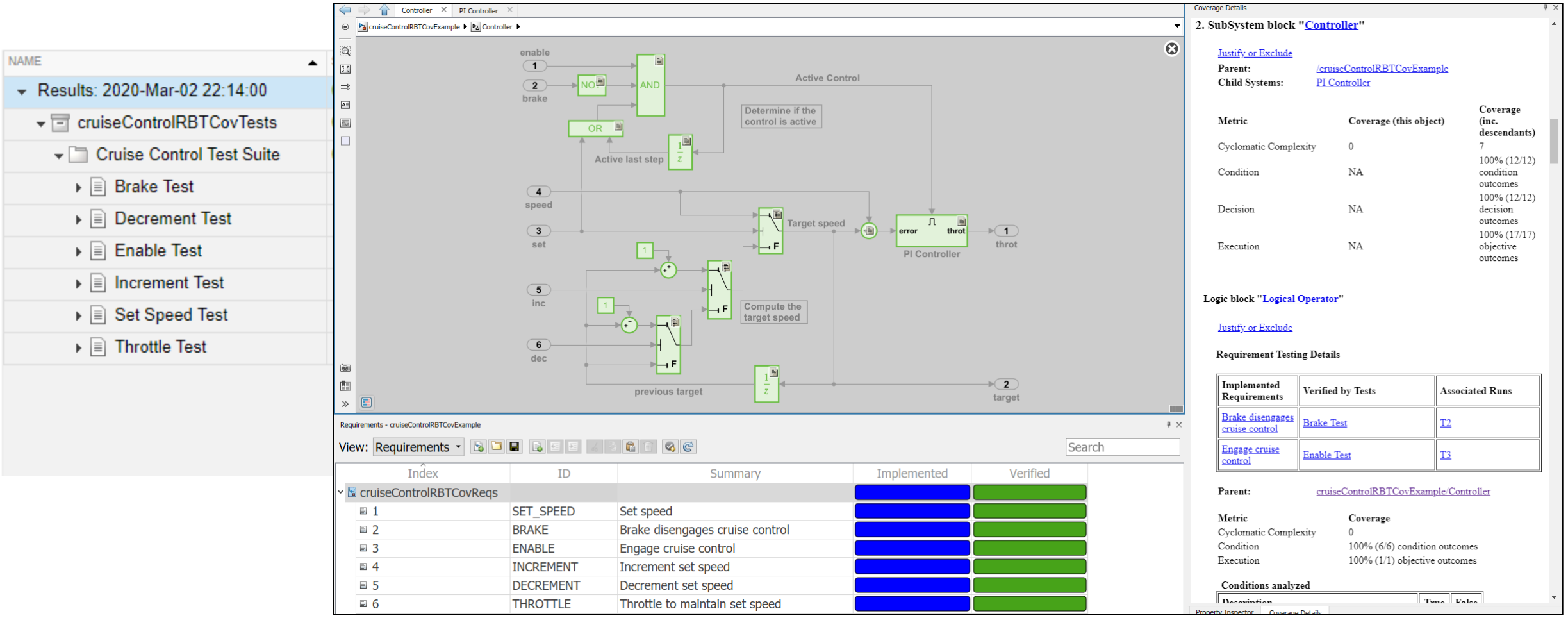


Summary

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%

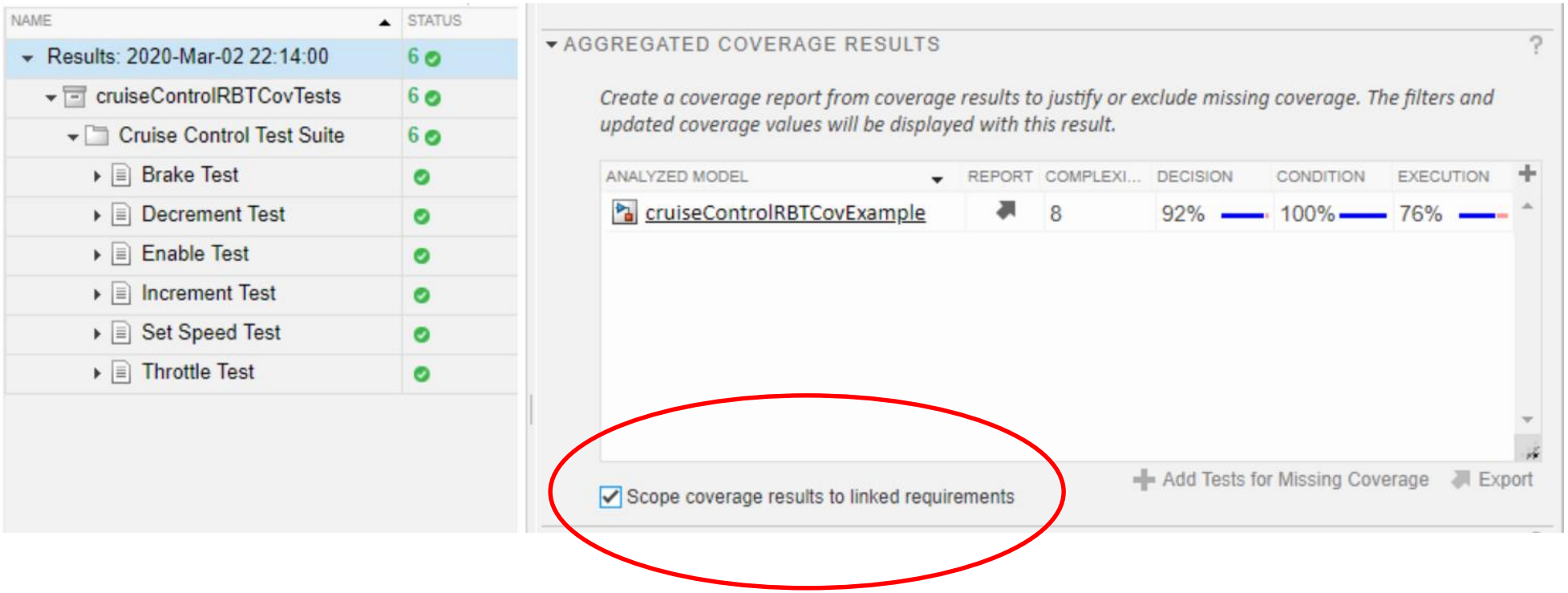
- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors

Test and Requirements Traceability in Coverage Results



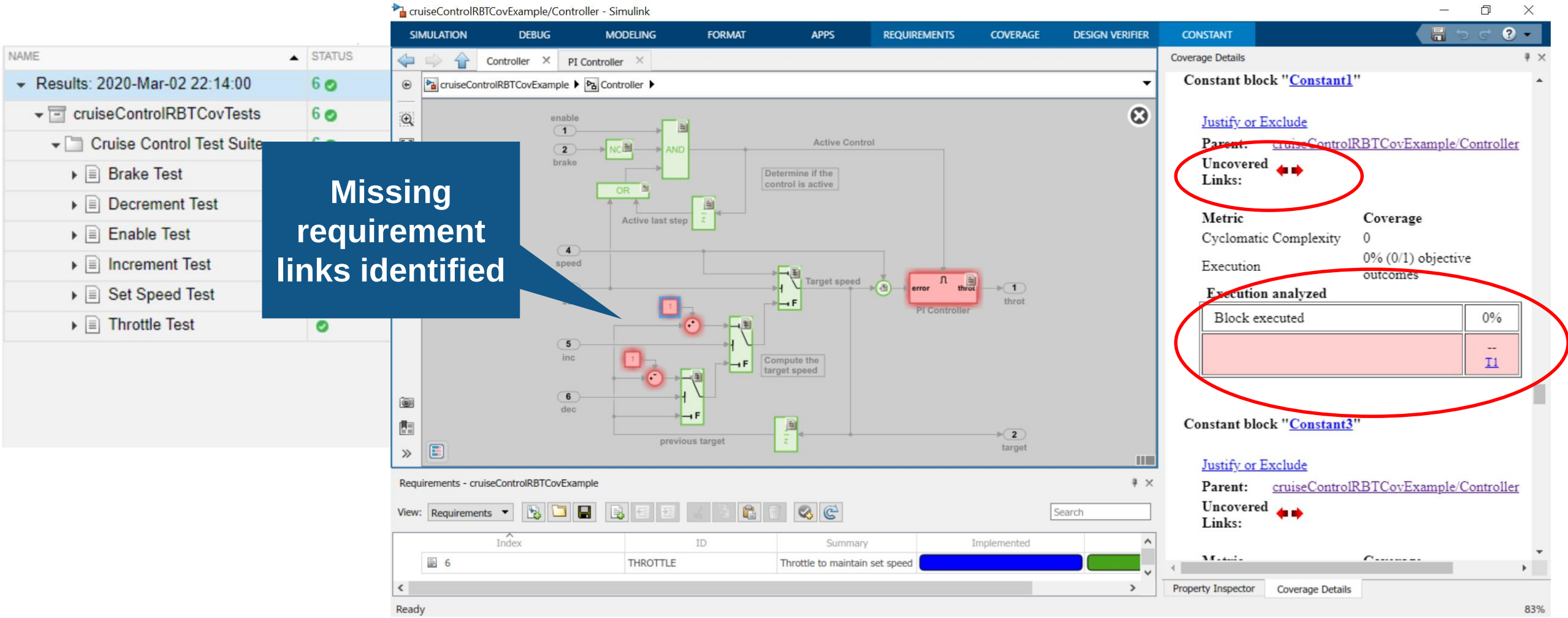
Scoping Model Coverage to Requirements-Based Tests

R2020a

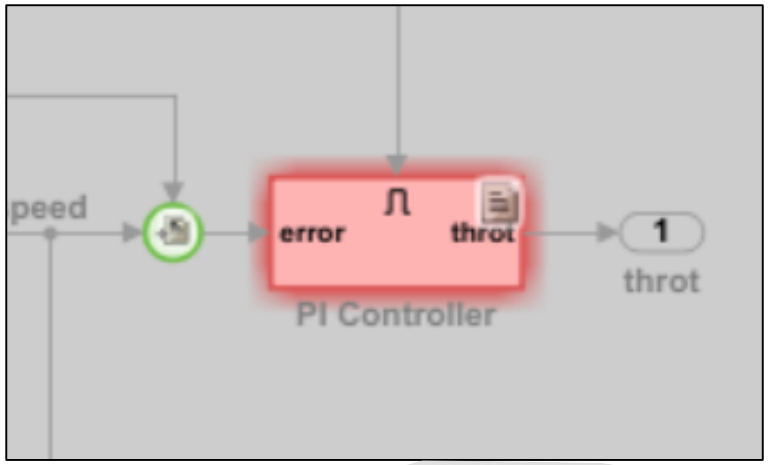


Scoping Model Coverage to Requirements-Based Tests

R2020a



Test and Requirements Traceability in Coverage Results

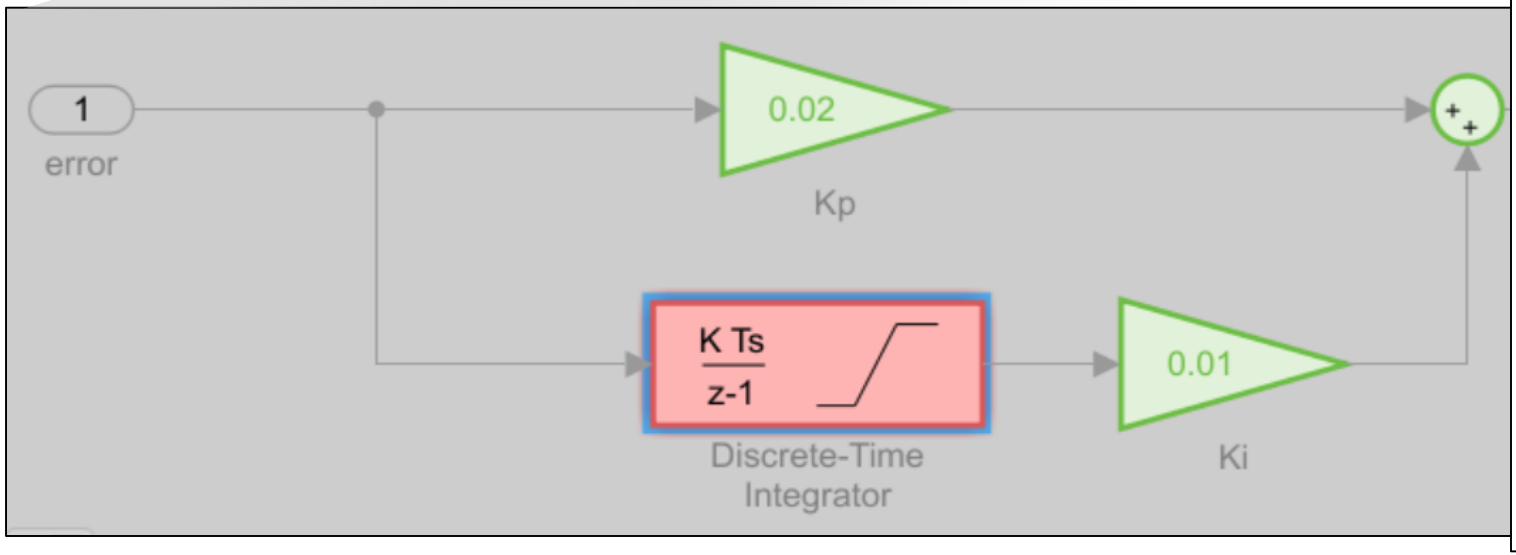


3. SubSystem block "PI Controller"

Justify or Exclude

Requirement Testing Details

Implemented Requirements	Verified by Tests	Associated Runs
Throttle to maintain set speed	Throttle Test	T6



DiscreteIntegrator block "Discrete-Time Integrator"

Justify or Exclude

Parent: cruiseControlRBTCovExample/Controller/PI Controller

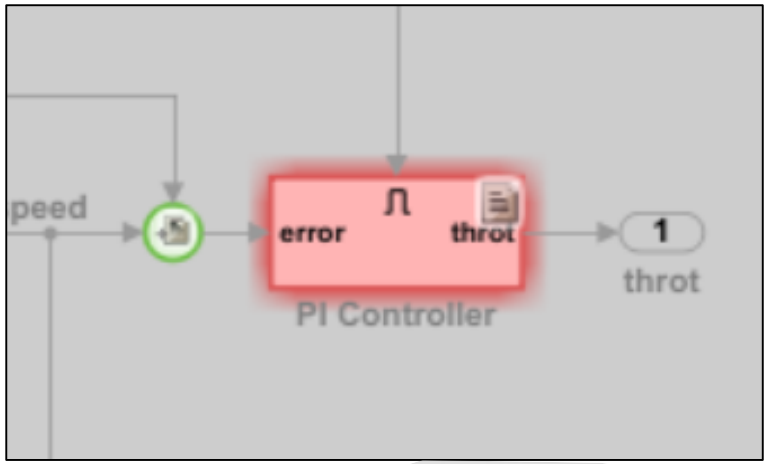
Uncovered Links:

Metric	Coverage
Cyclomatic Complexity	2
Decision	75% (3/4) decision outcomes
Execution	100% (1/1) objective outcomes

Decisions analyzed

integration result <= lower limit	100%
false	391/801 T6
true	410/801 T6
integration result >= upper limit	50%
false	801/801 T6
true	0/801 T4

Test and Requirements Traceability in Coverage Results



3. SubSystem block "PI Controller"

Justify or Exclude

Requirement Testing Details

Implemented Requirements	Verified by Tests	Associated Runs
Throttle to maintain set speed	Throttle Test	T6

DiscreteIntegrator block "Discrete-Time Integrator"

Justify or Exclude

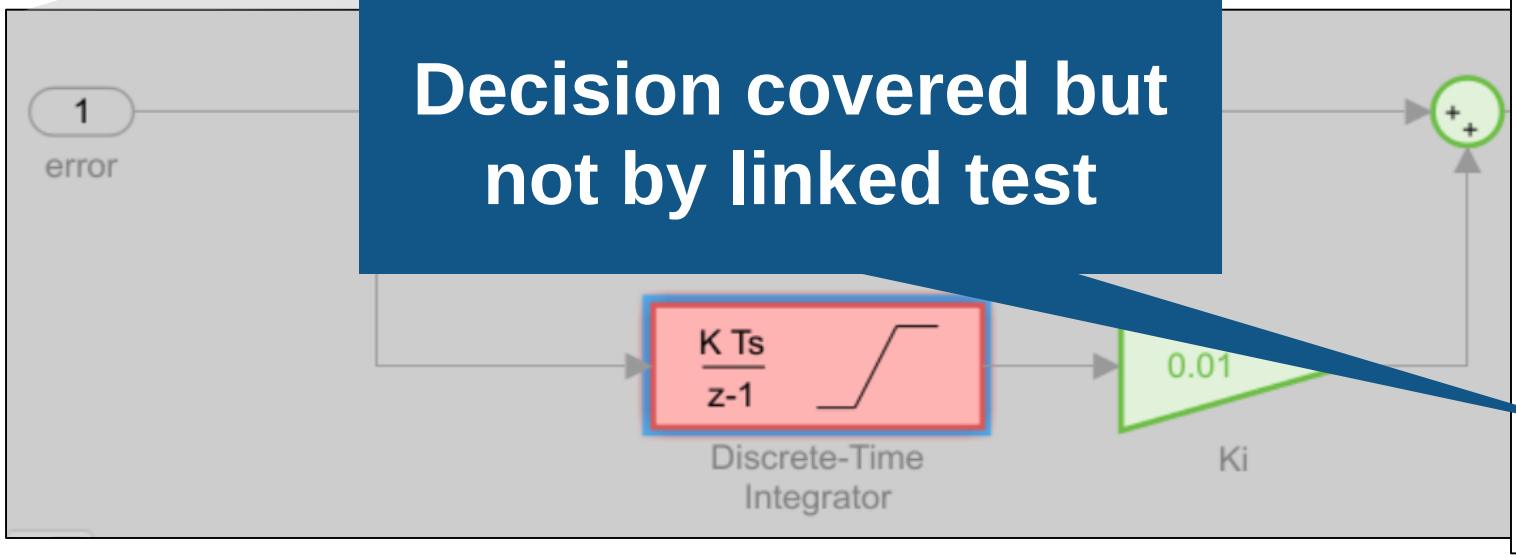
Parent: cruiseControlRBTCovExample/Controller/PI Controller

Uncovered Links:

Metric	Coverage
Cyclomatic Complexity	2
Decision	75% (3/4) decision outcomes
Execution	100% (1/1) objective outcomes

Decisions analyzed

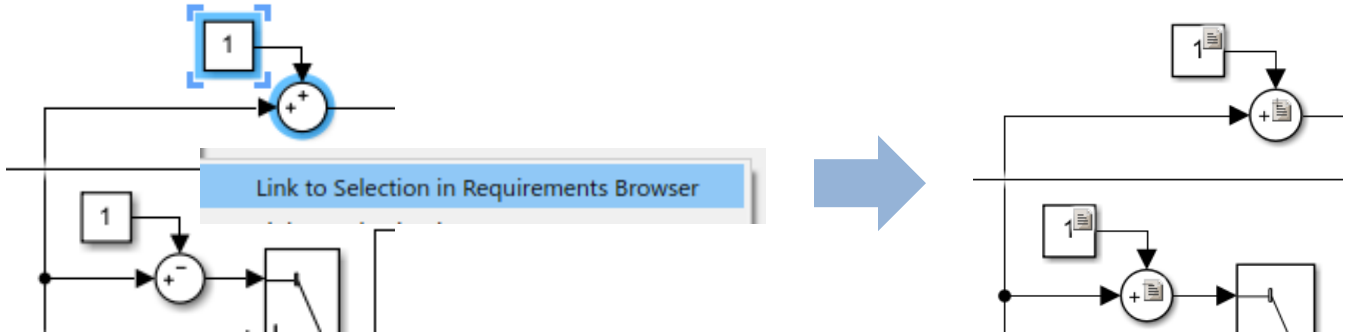
integration result <= lower limit	100%
false	391/801 T6
true	410/801 T6
integration result >= upper limit	50%
false	801/801 T6
true	0/801 T4



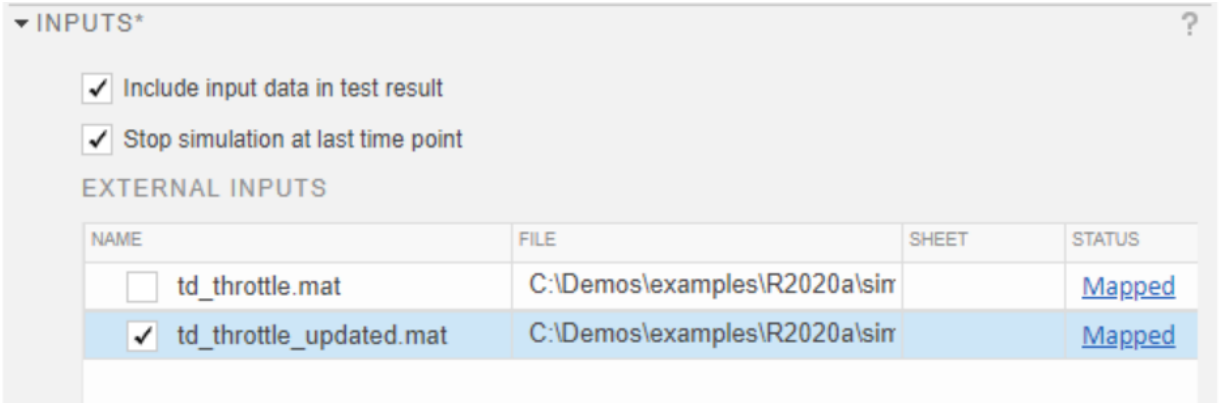
Decision covered but not by linked test

Address missing Requirements Based Test Coverage

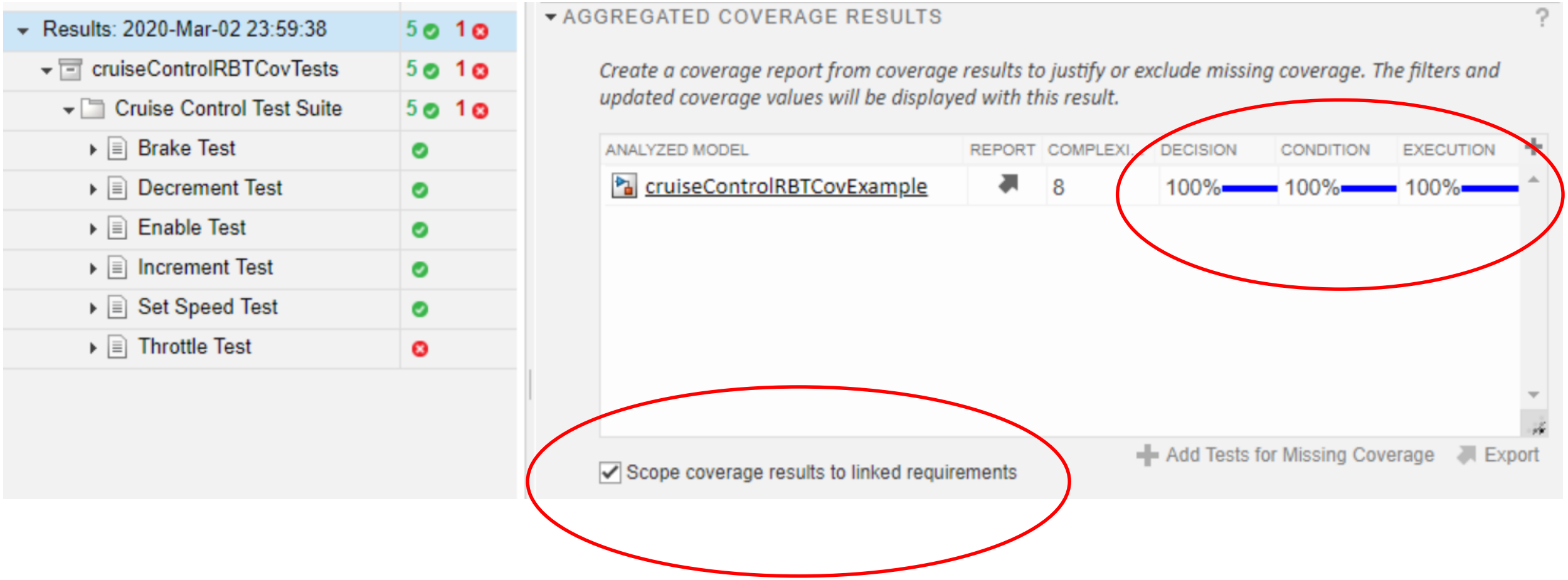
- Add missing implementation links to requirements



- Update test to increase target speed



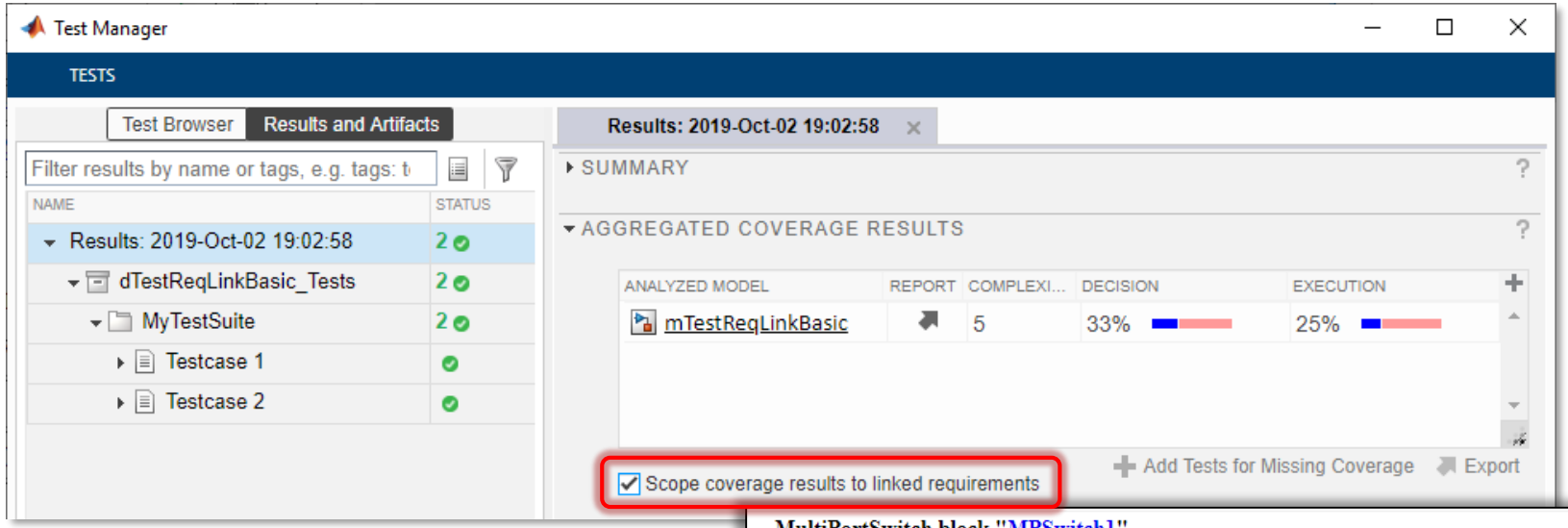
100% Coverage but Testing Identified Error in Implementation



Additional Testing Identified Error in Implementation

Scoped Model Coverage to Requirements-Based Tests

R2020a



DO-178C 6.4.4.2

… coverage information collected during requirements-based testing to confirm that …

MultiPortSwitch block "MPSwitch1"

Requirement Testing Details

Implemented Requirements	Verified by Tests	Associated Runs
Requirement 1	Testcase 1	T1

Metric	Coverage
Cyclomatic Complexity	2
Decision	33% (1/3) decision outcomes
Execution	100% (1/1) objective outcomes

Decisions analyzed

truncated input value	33%
= 1 (output is from input port 1)	51/51 T1
= 2 (output is from input port 2)	0/51 T2
= *,3 (output is from input port 3)	0/51

Hit by linked RBT -- **Satisfied**

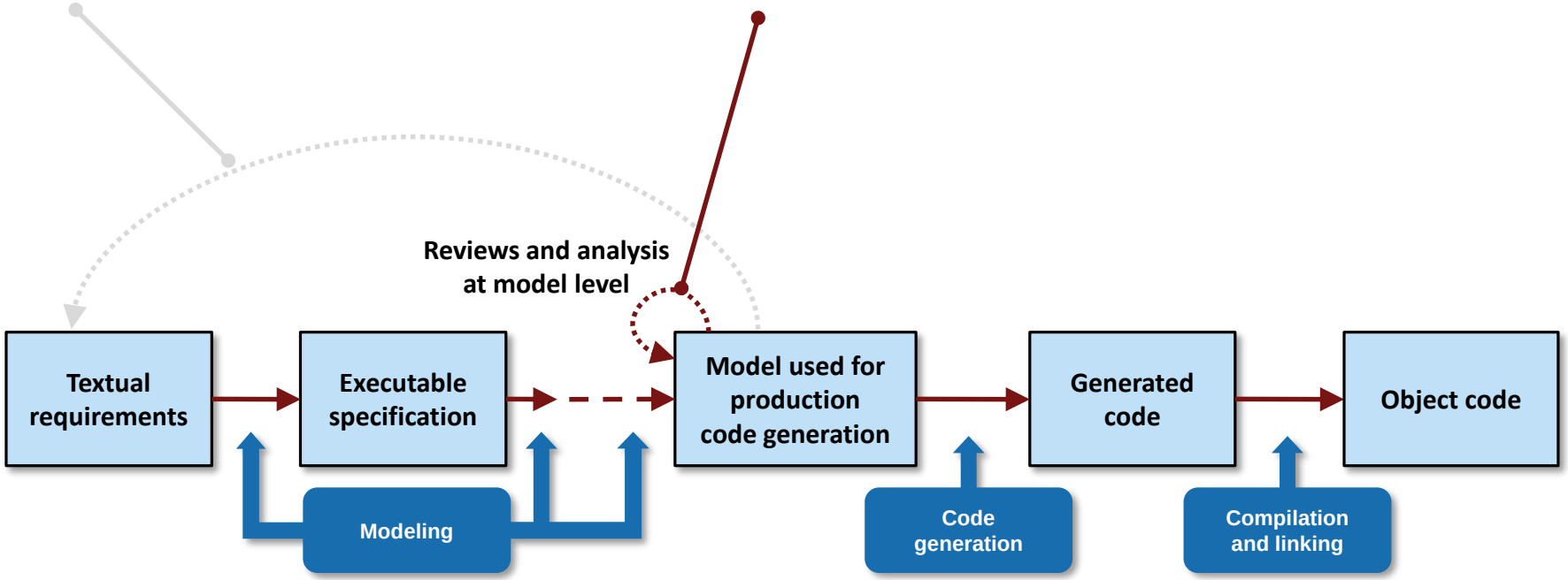
Hit, but not by linked RBT -- **Unsatisfied**

Check standard compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage

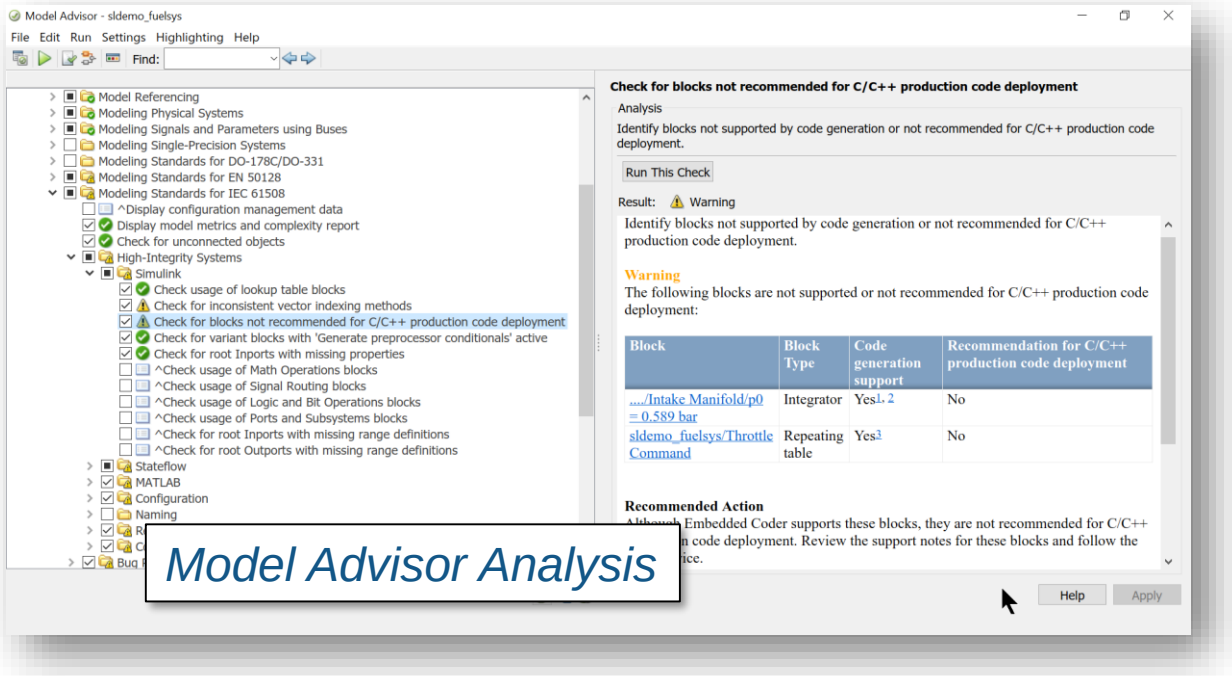
- **Check standard compliance**
- Detect design errors
- Prove model behavior compliance

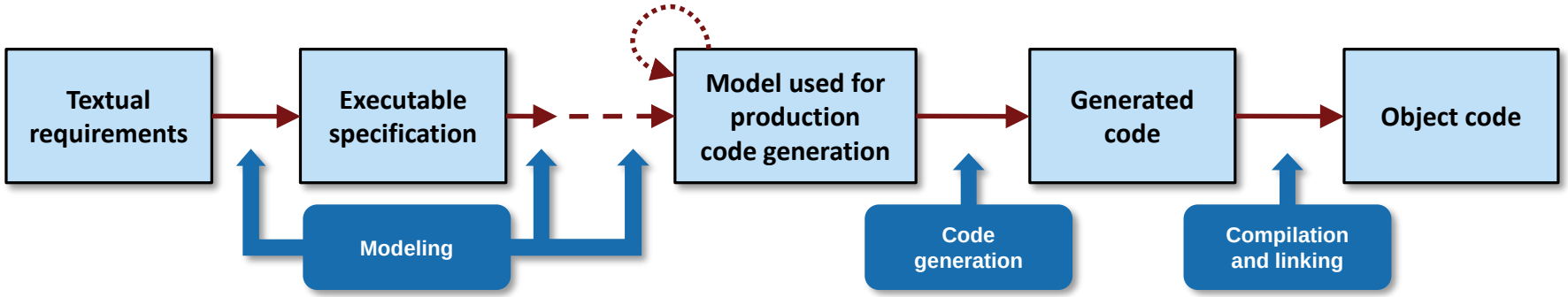


Verify Design to Guidelines and Standards

Check for:

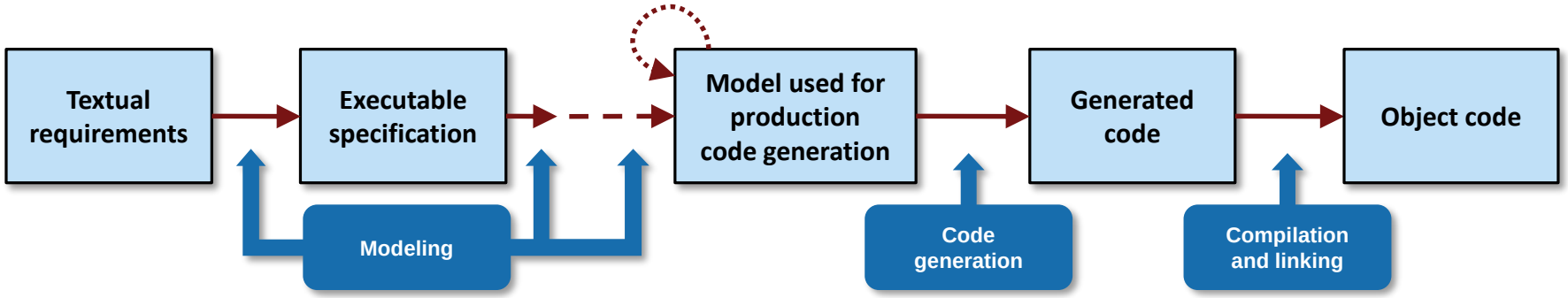
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more……





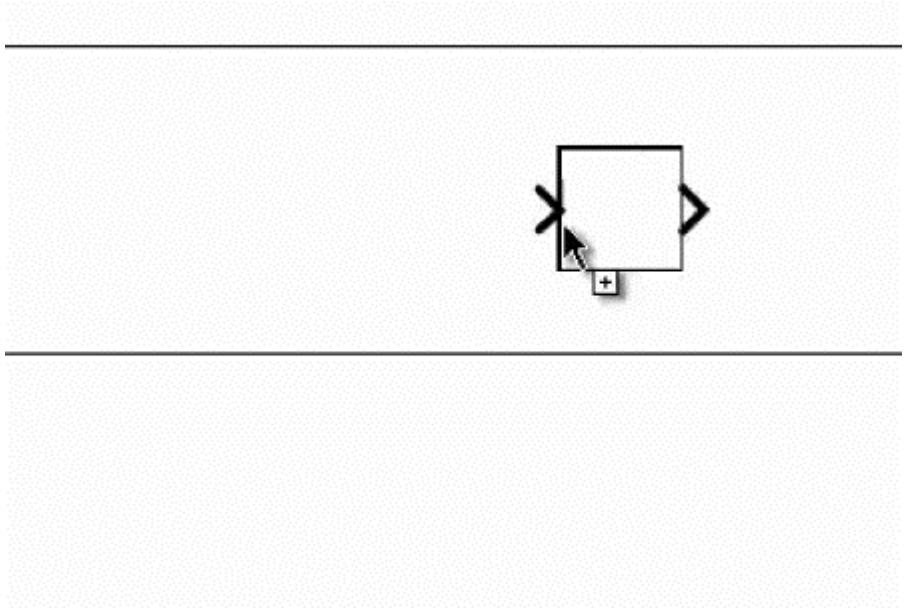
Built in checks for industry standards and guidelines

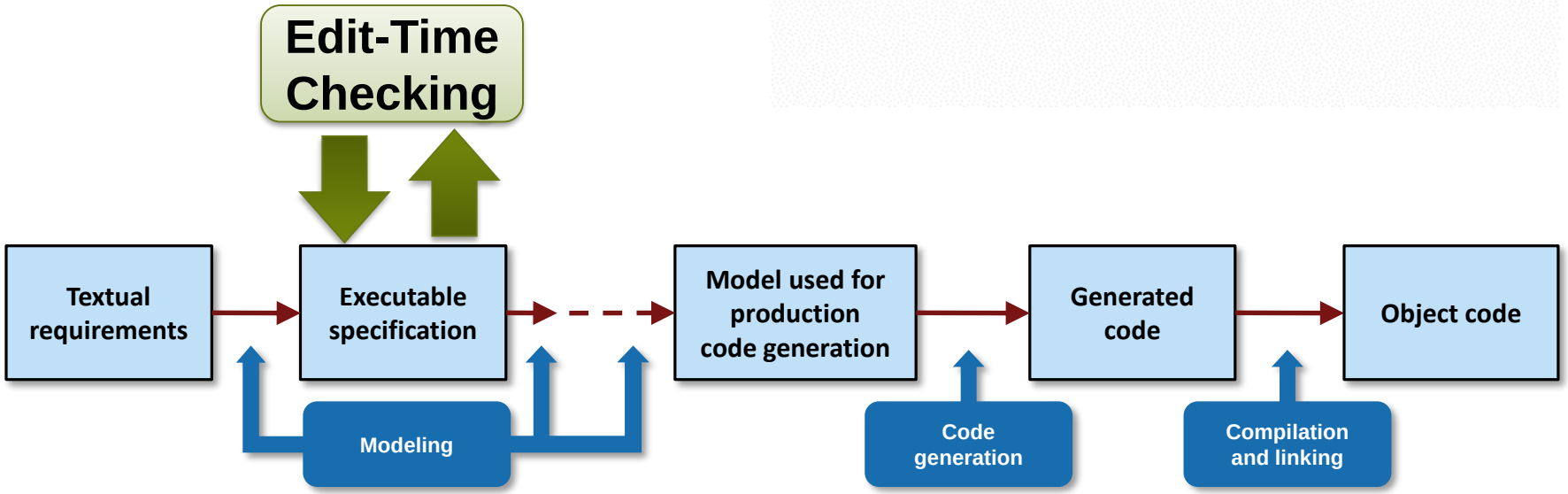
- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128
- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAB (MathWorks Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)



Shift Verification Earlier With Edit-Time Checking

- Highlight violations as you edit
- Fix issues earlier
- Avoid rework

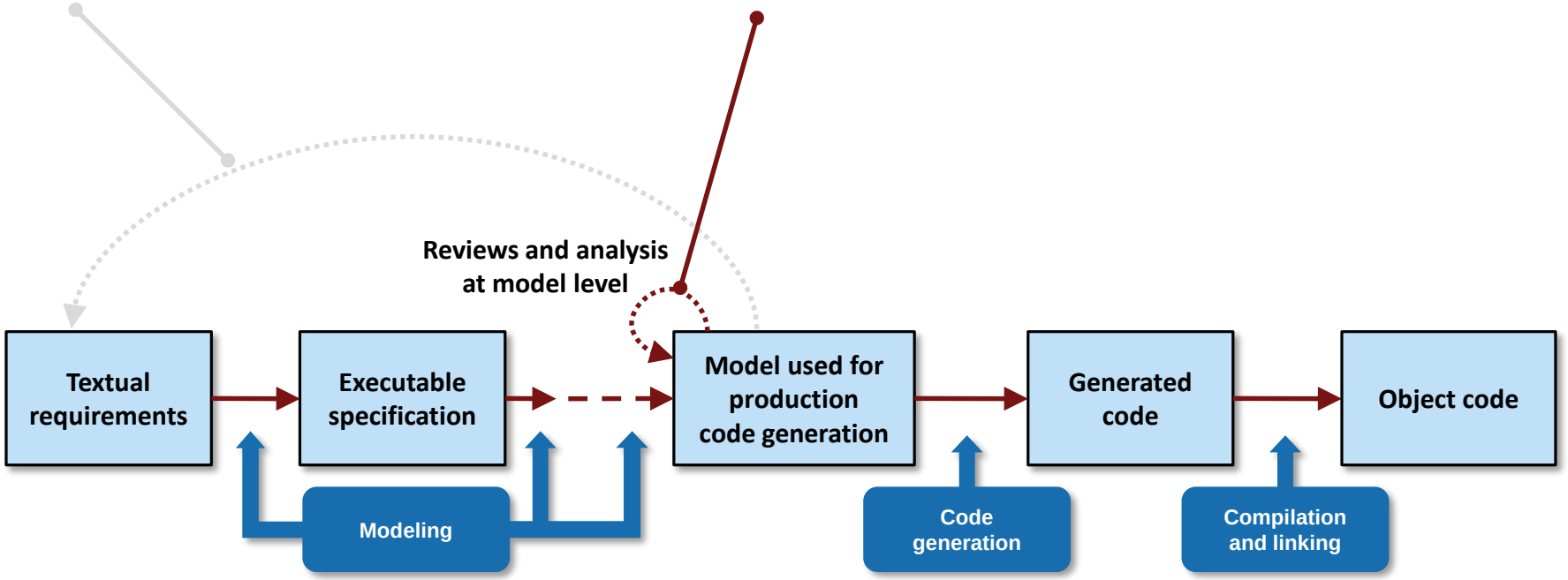




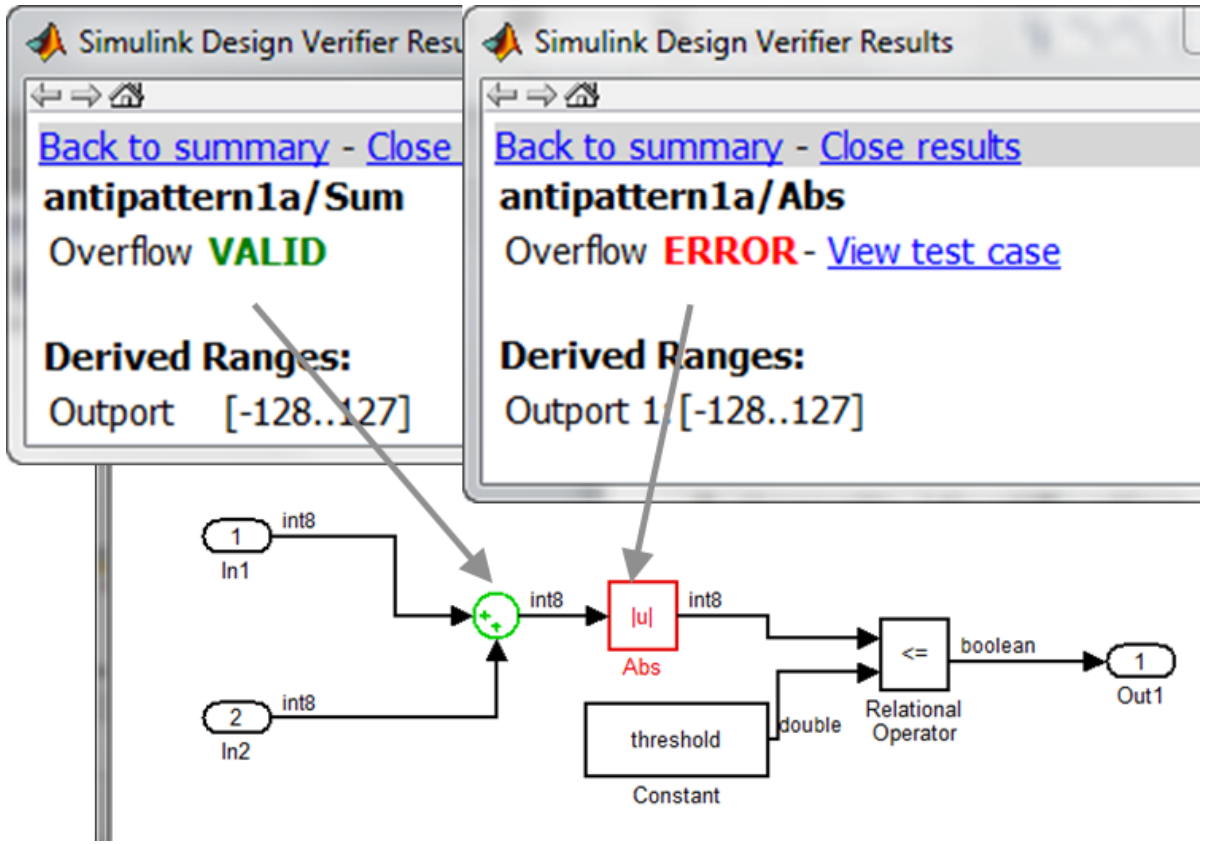
Detect Design Errors with Formal Methods

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- **Detect design errors**
- Prove model behavior compliance



Detect Design Errors Using Formal Methods

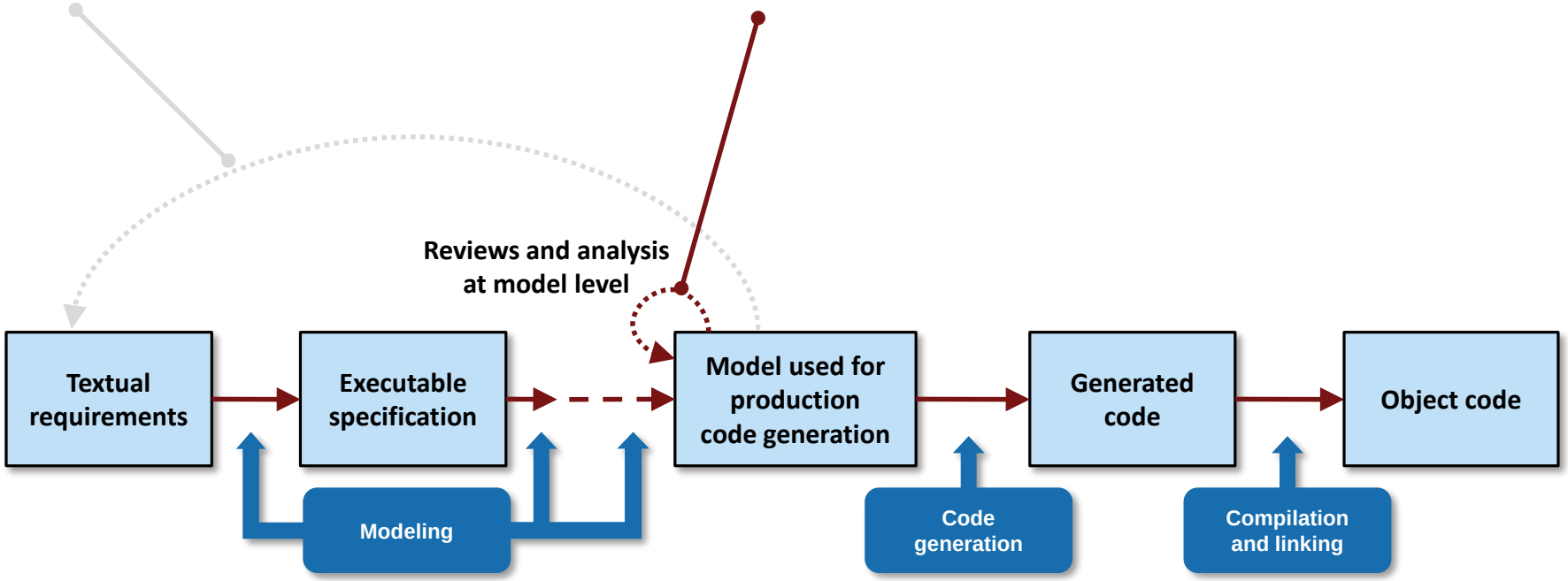


- Find design errors
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations
- Generate counter example to reproduce error

Prove Model Behavior Compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- **Prove model behavior compliance**

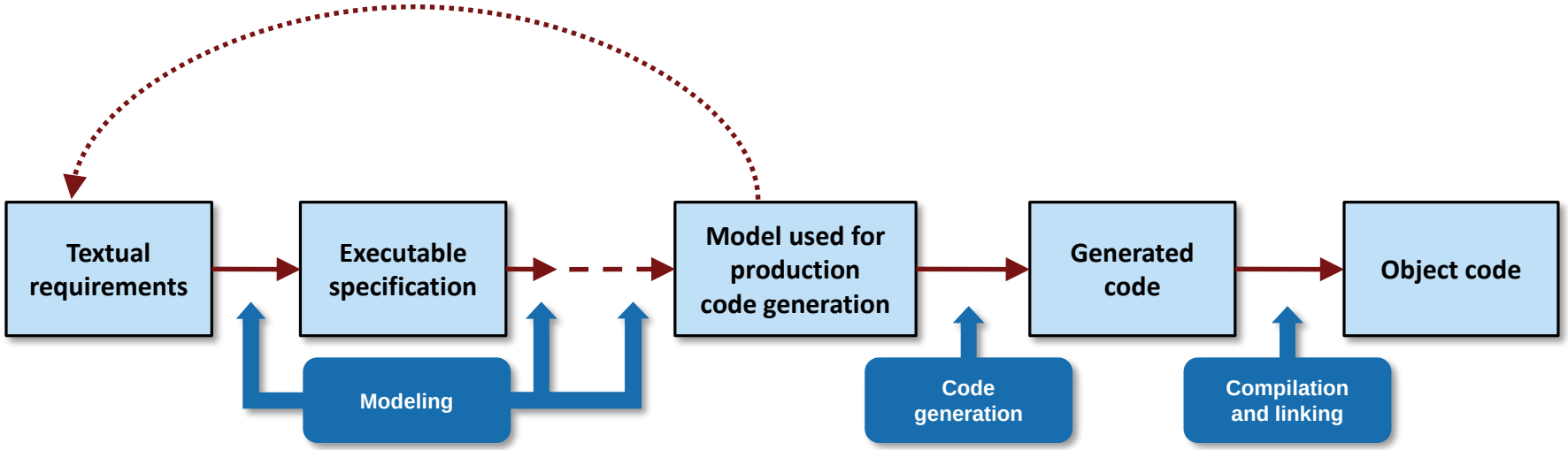


Proving Model Meets Requirements

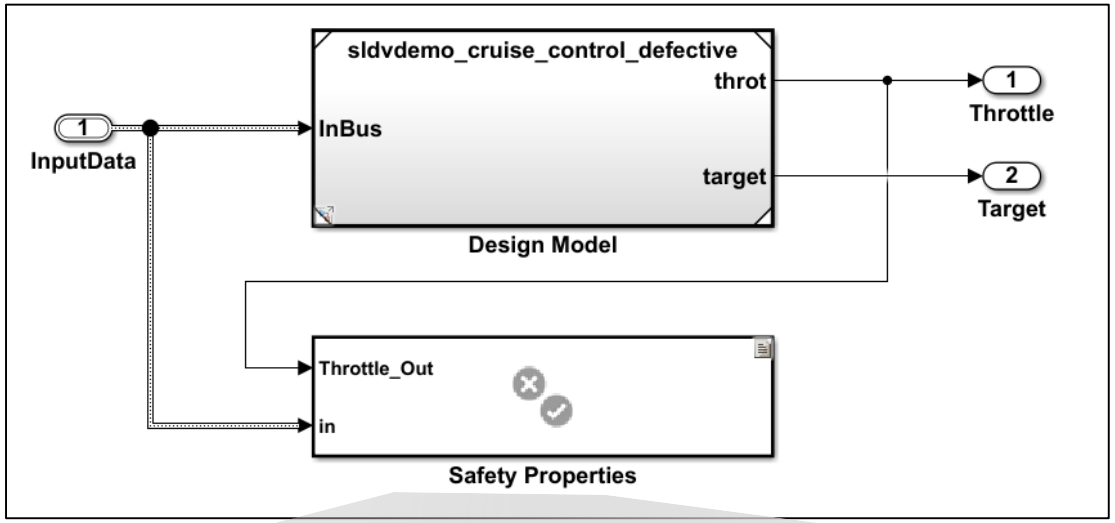
Safety Requirement:

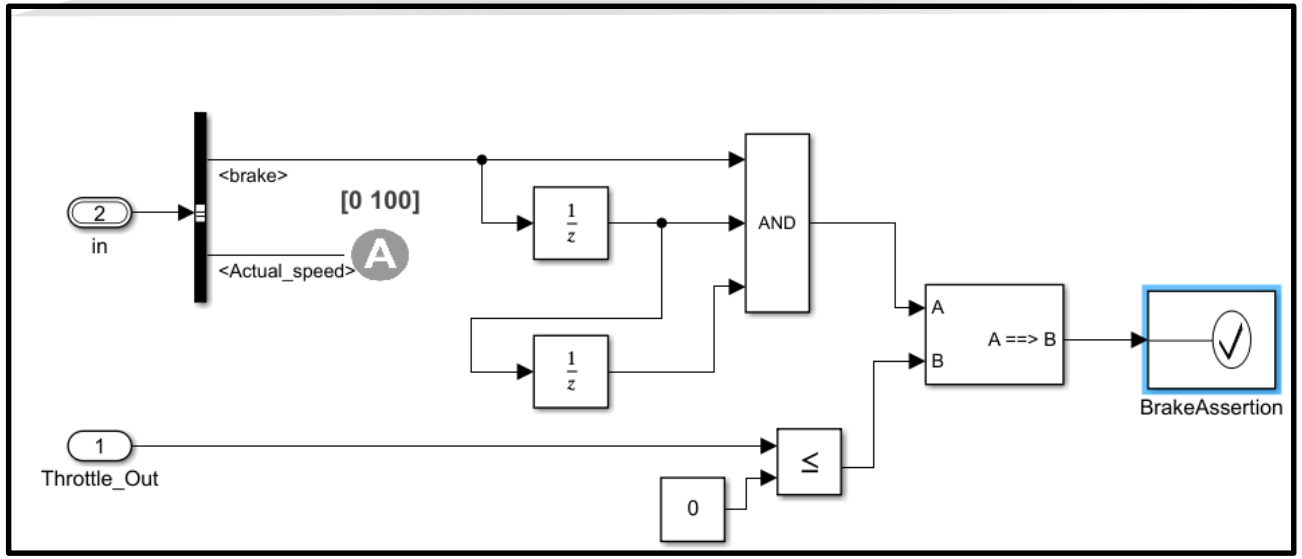
When the brake is applied for three consecutive steps, the throttle shall go to zero.

- Need to ensure the design performs correctly

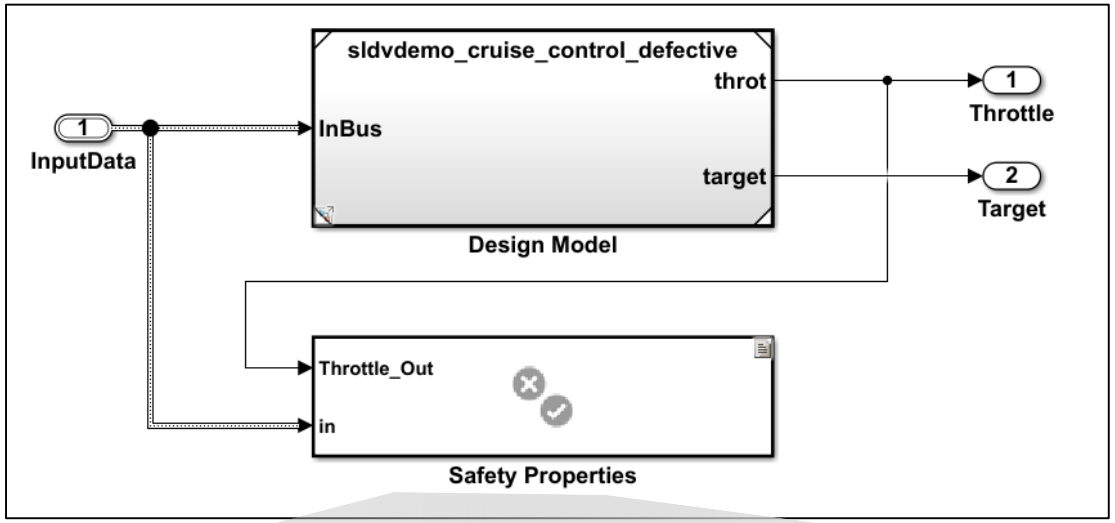


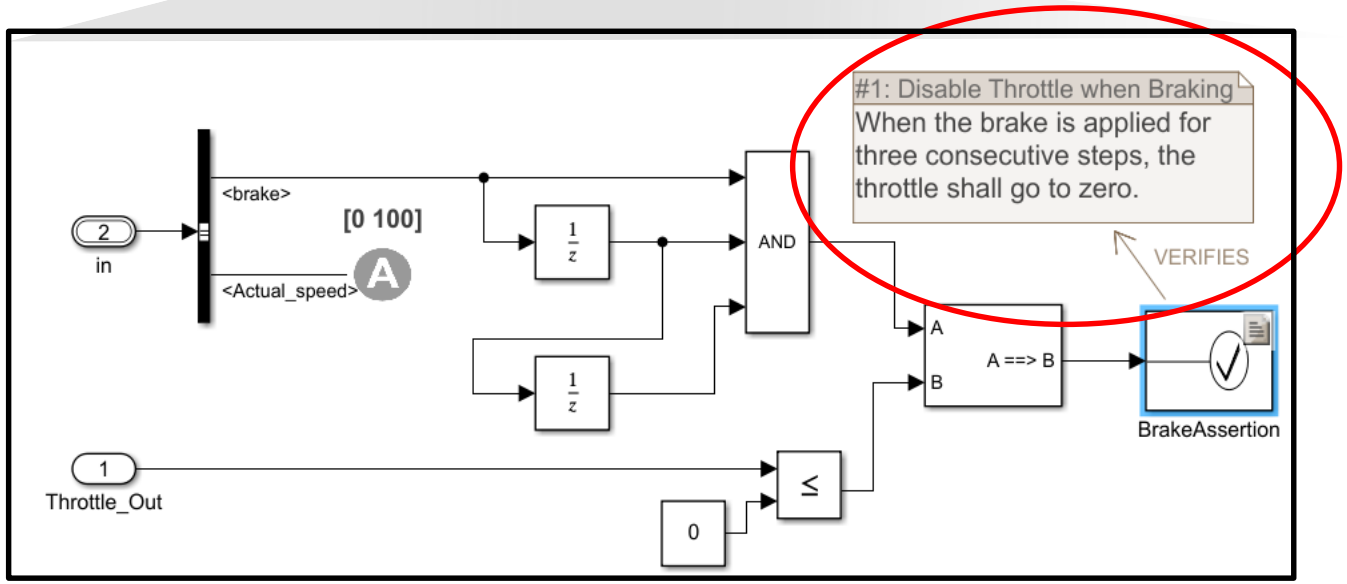
Model functional and safety requirements



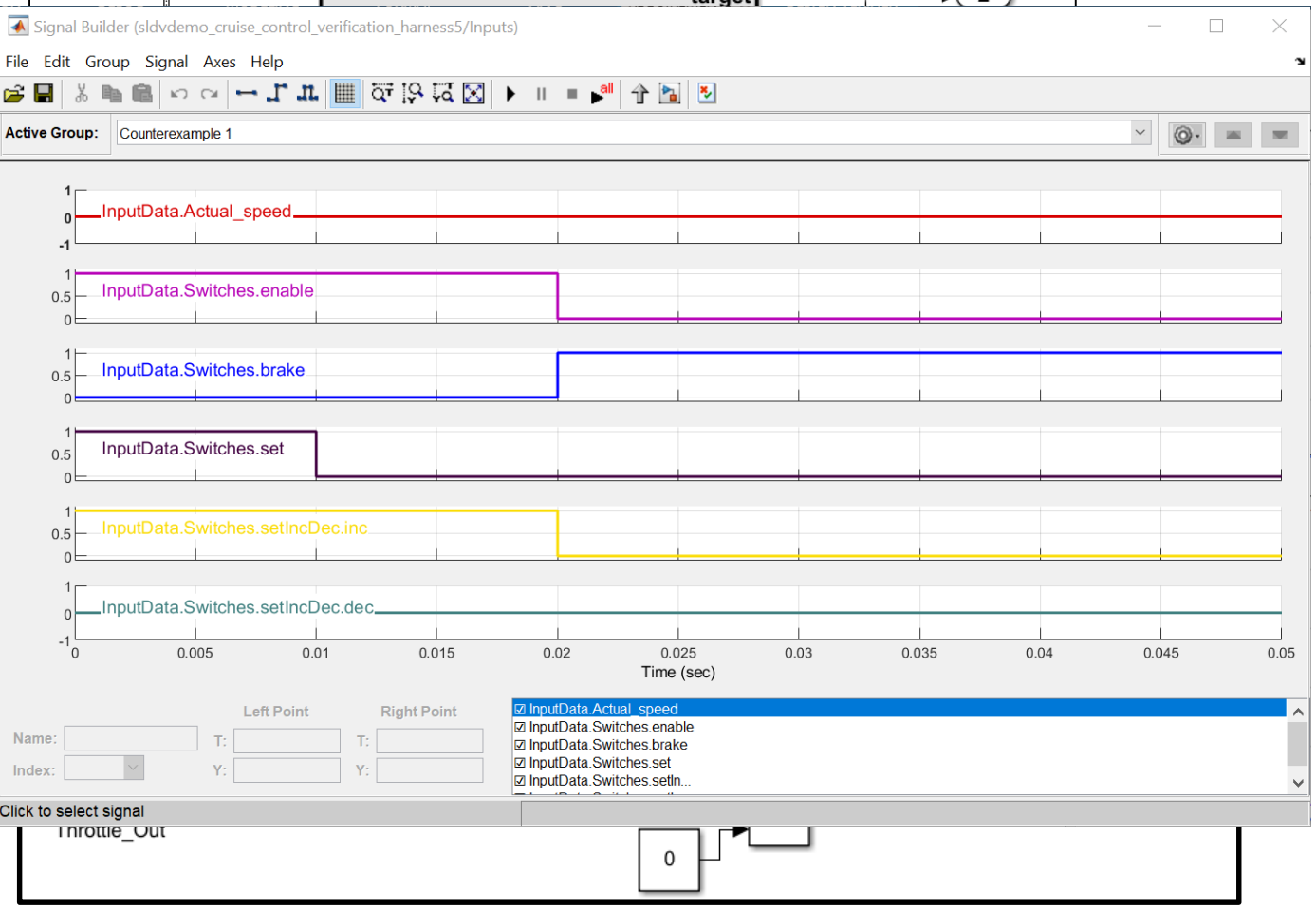


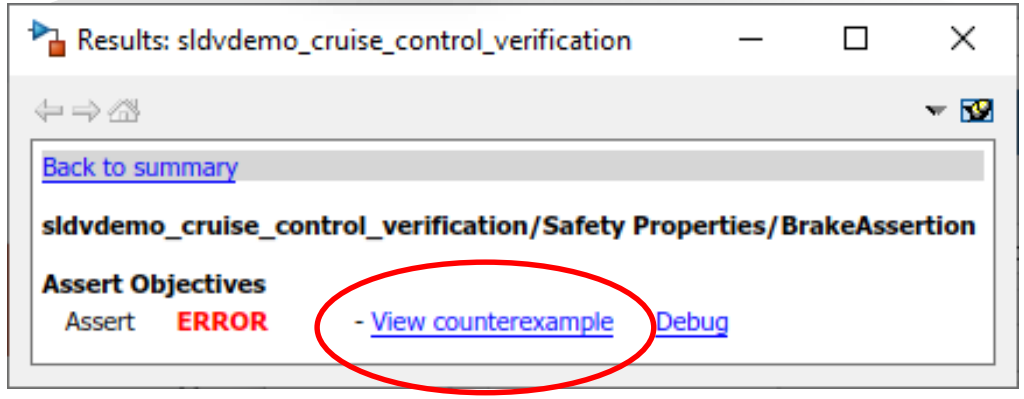
Link requirements to properties



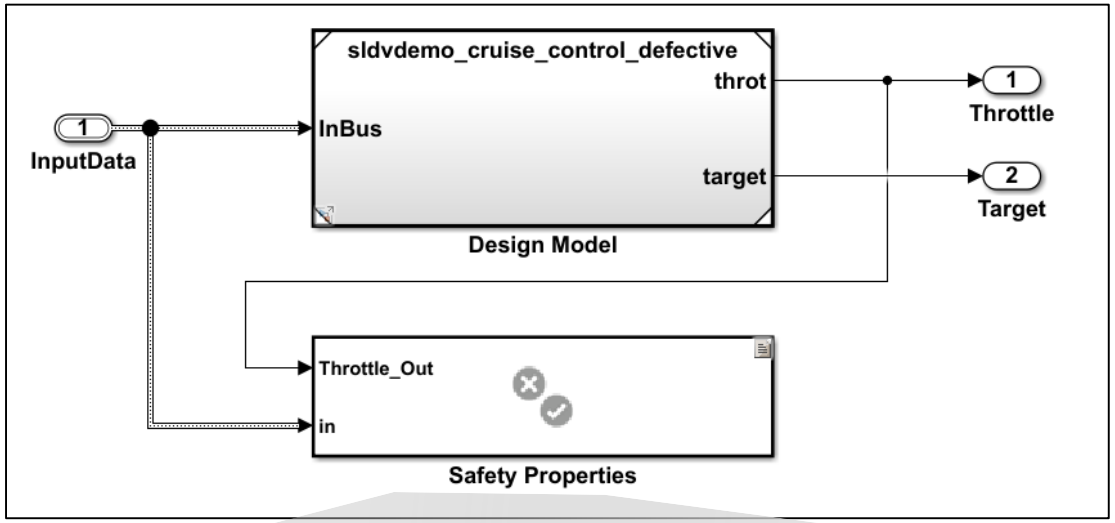


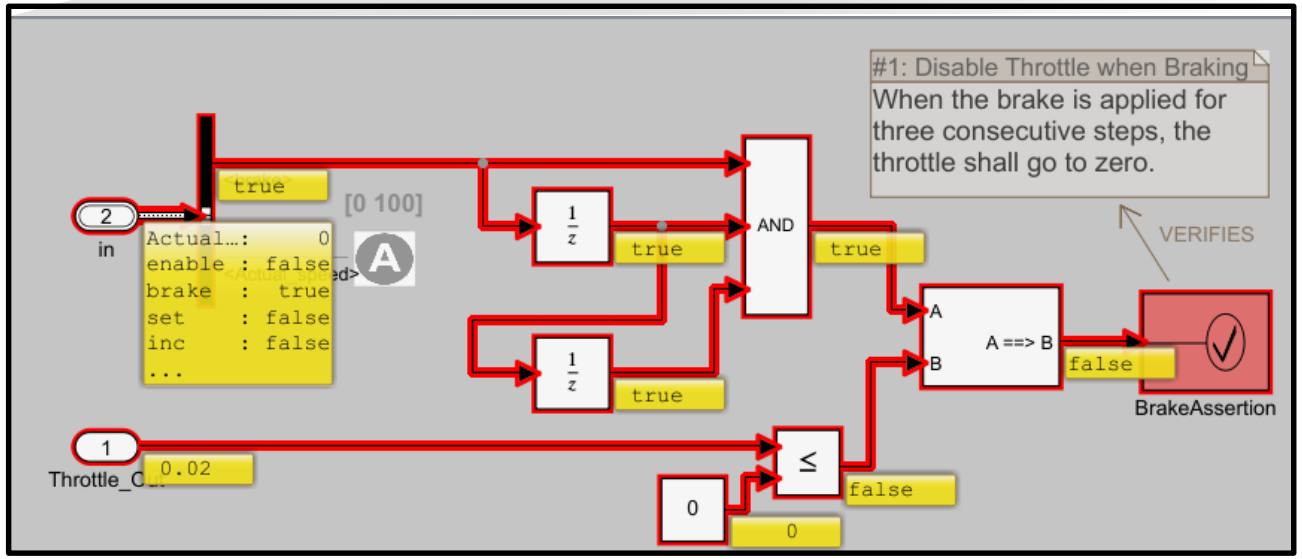
Prove That Design Meets Requirements

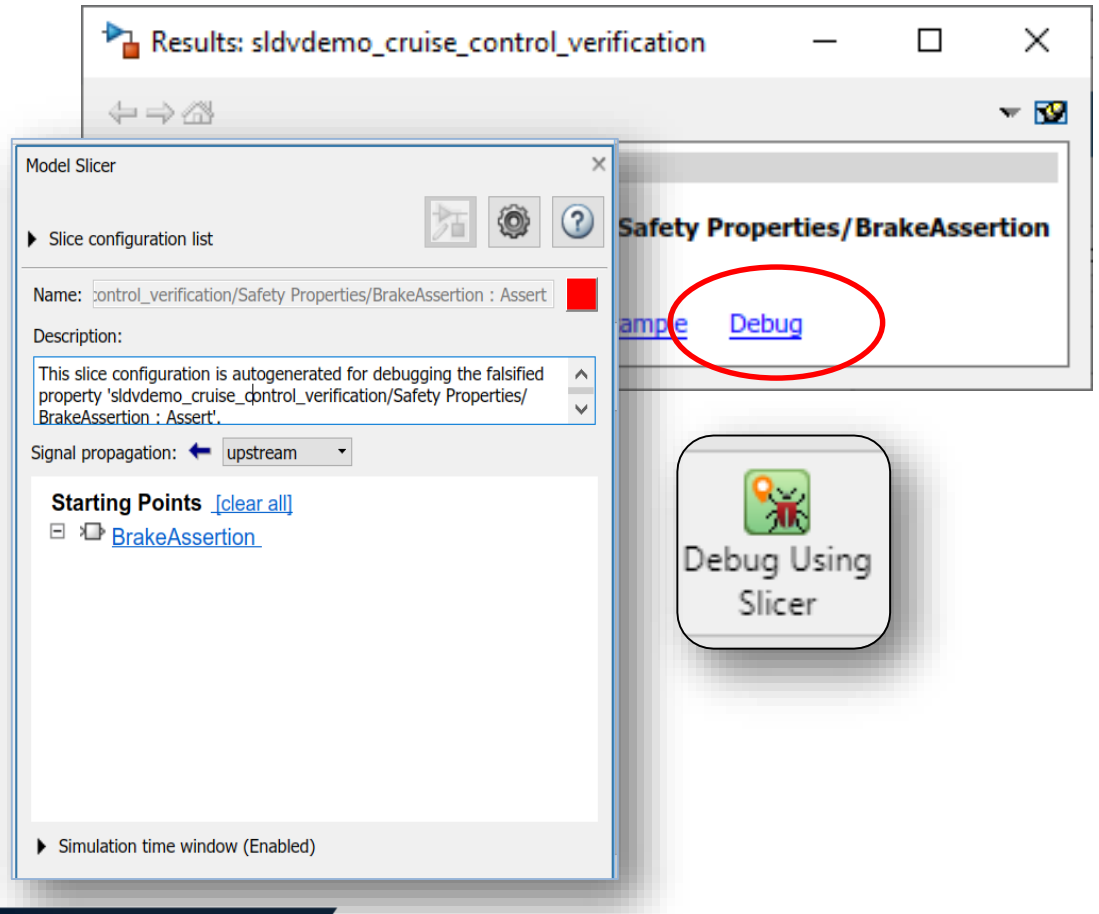




Debugging Property Proving Violations







Resolve unexpected behavior in a model with Model Slicer

Isolate
Find the area of the model responsible for unexpected behavior

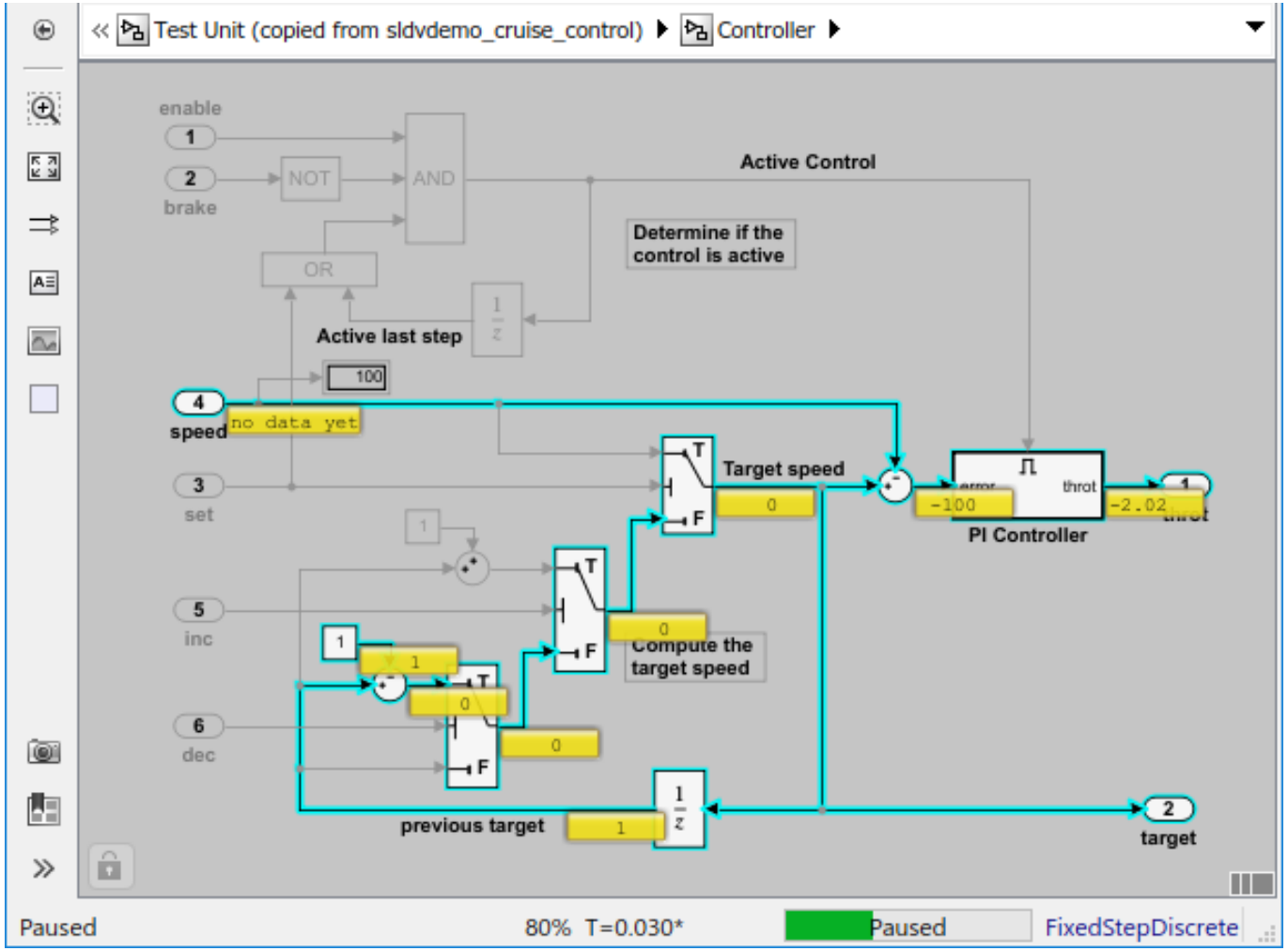
Analyze dependencies
Understand data & control dependencies in large or complex models

Inspect slice regions
Highlight model slices for time windows or failure states & transitions for state flow.

Debug simulation behavior
Step through precompiled slices to understand signal and port value propagation

Correct Model

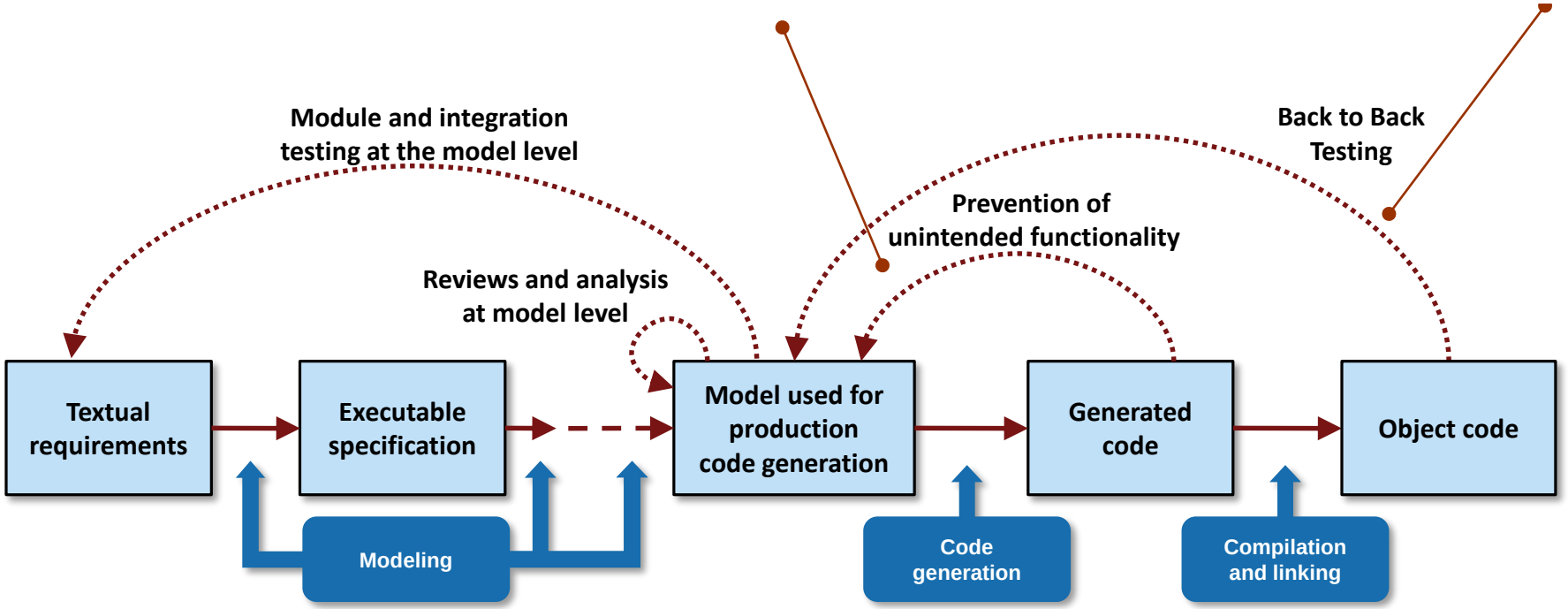
Iterate



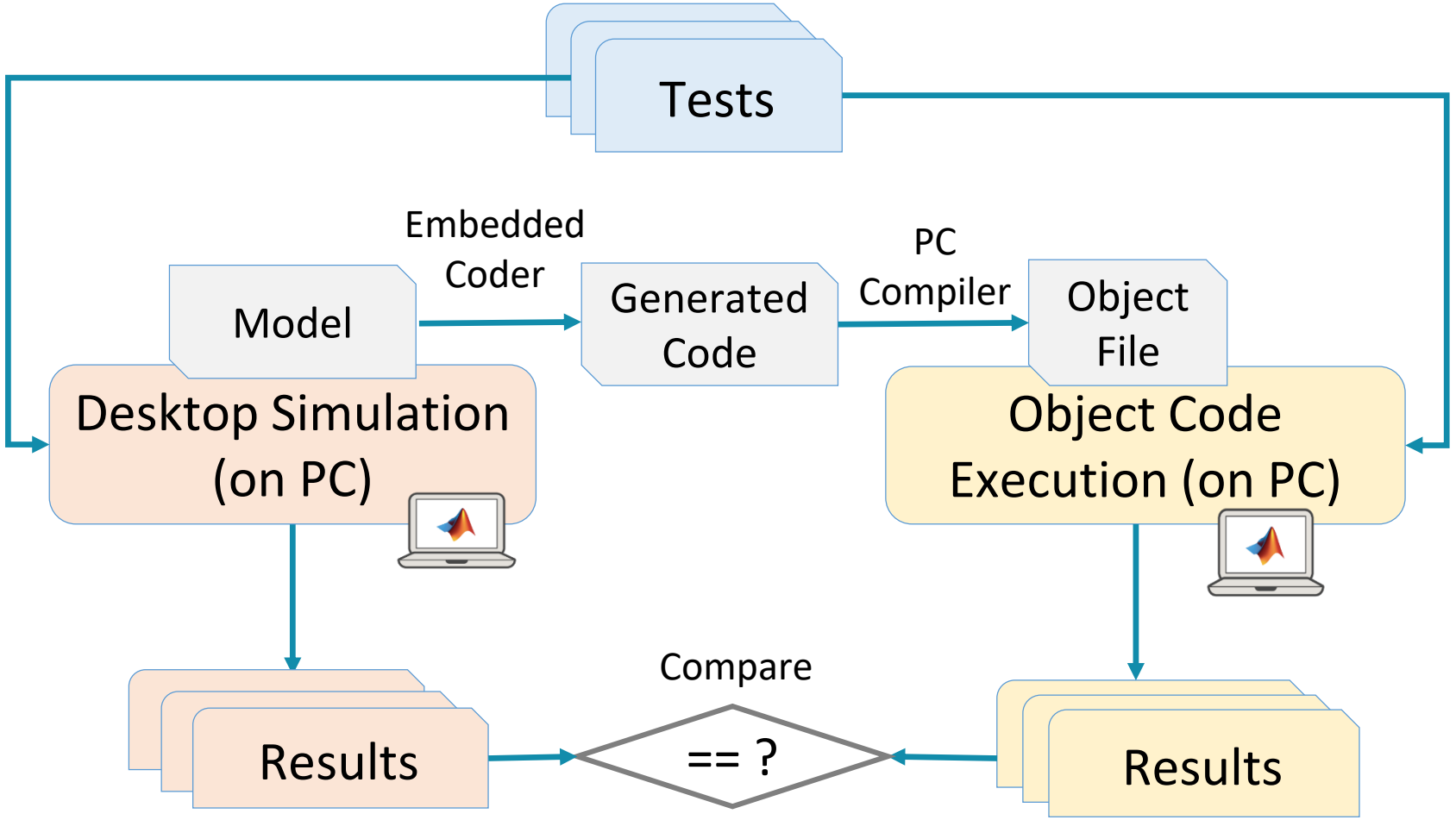
Code Verification: Gain Confidence in the Generated Code

Code Verification

- Trace code to model and requirements
- Measure code coverage
- SIL/PIL equivalence testing
- Generate 100% coverage test vectors

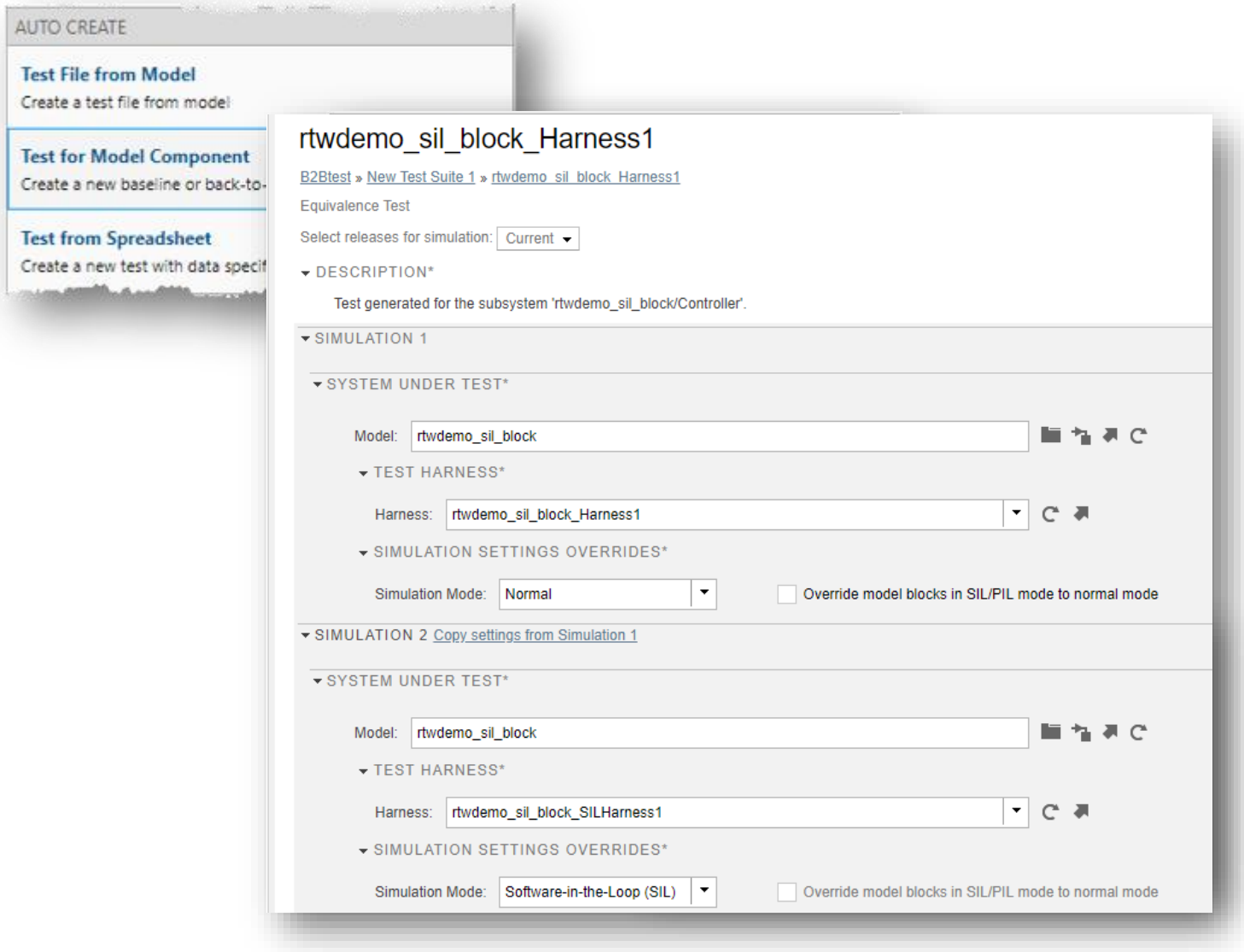


Back-to-Back Testing



- Automate SIL testing using Simulink Test.
- Testing across releases.

Automate Test Creation using Test Manager Wizard

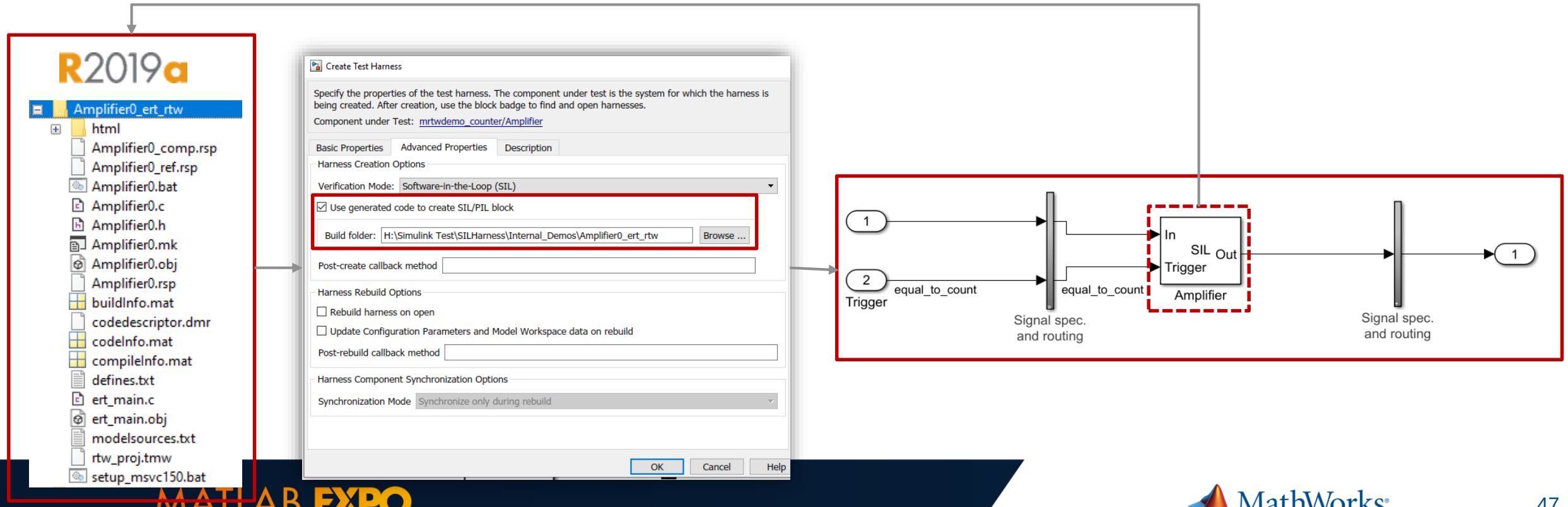


- Guided steps to define component to test, inputs, type of test and format for output
- Auto generate tests using Simulink Design Verifier

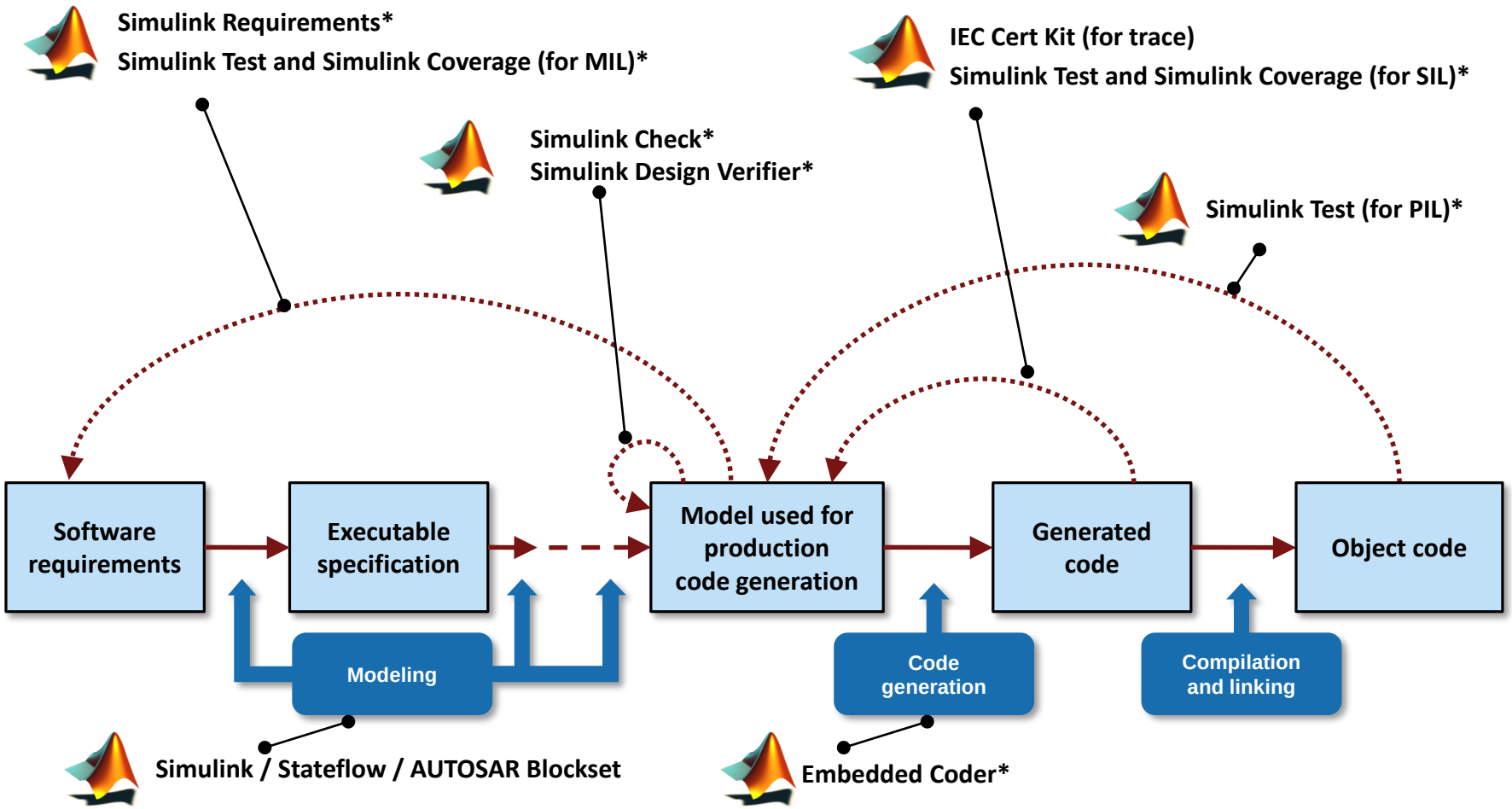
Cross Release SIL/PIL Test Harness Generation

R2020a

- Create a SIL/PIL test harness using code that was generated in a previous release
- Modify existing SIL/PIL test harnesses to store the build folder path information which can be used for rebuild



Reference Workflow for Generated Code



*Qualifiable

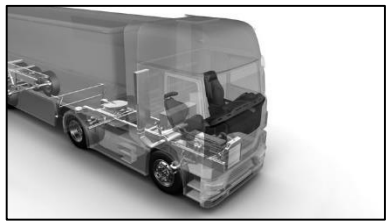
Customer References and Applications

Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design

Software testing time cut by two-thirds

LS Automotive Reduces Development Time for Automotive Component Software with Model-Based Design

Specification errors detected early



Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks

Verification time cut by up to 50 percent

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

Services

MathWorks Training Service

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses

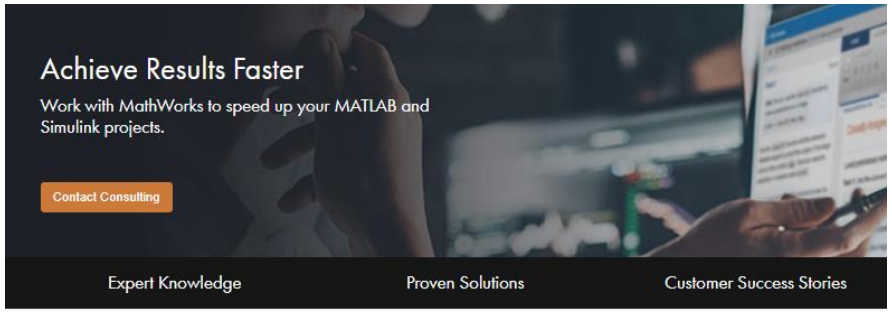
More than 48 course offerings:

- Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
- Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis, and other areas

https://www.mathworks.com/services/training.html

MathWorks Consulting Service

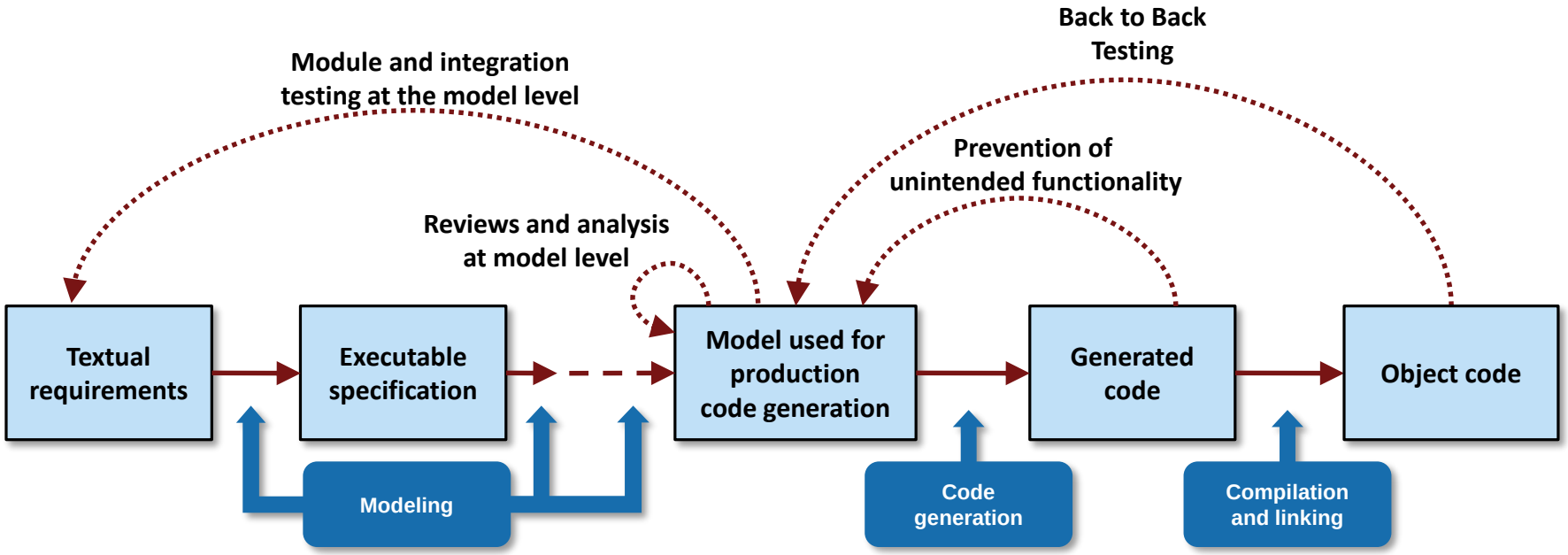
- Early Verification and Validation with Model-Based Design
- DO 178 Certification Advisory Service
- ISO 26262 Process Deployment Advisory Service
- Model-Based Design Process Establishment
- Model-Based Design Process Assessment and Maturity Framework
- Tools Integration
- Developing Embedded Software
- Ask about on how we can quickly get you started on System Composer for defining system architecture (contact consulting)



https://www.mathworks.com/services/consulting.html

Use reference workflow to conform to standards

- Shift verification earlier
- Automate manual verification tasks (coding, compiling, back-to-back)
- Measure completeness of Requirements Based Testing



Learn More

- Verification, Validation, and Test Solution Page
- Requirements-Based Testing Workflow Example
- Verifying Models and Code for High-Integrity Systems
- Getting Started with Model Verification and Validation

Thank You!