MATLAB EXPO

인증 표준을 만족하기 위한 요구사항 기반 검증의 자동화 방안

류성연, MathWorks Korea

MathWorks®

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

Table 5 — Notations for software unit design

Notations		ASIL			
		A	B	C	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.

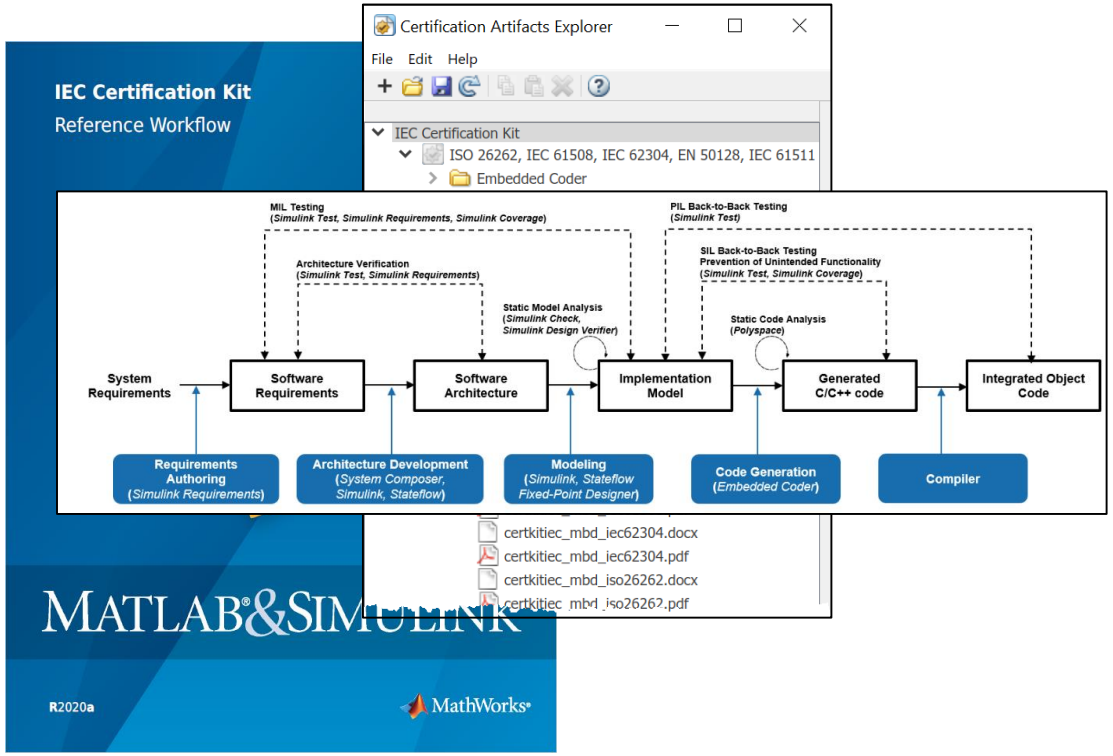
[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

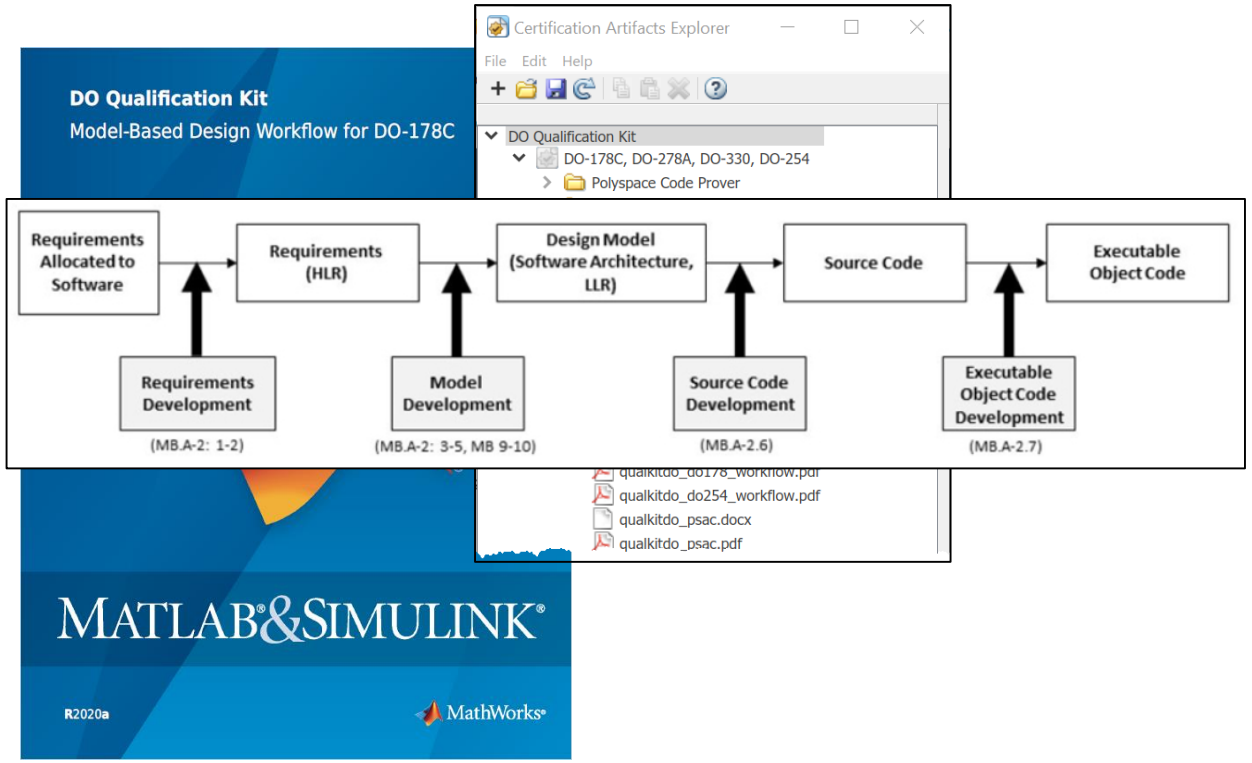
NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

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.

Qualify tools with IEC Certification Kit and DO Qualification Kit

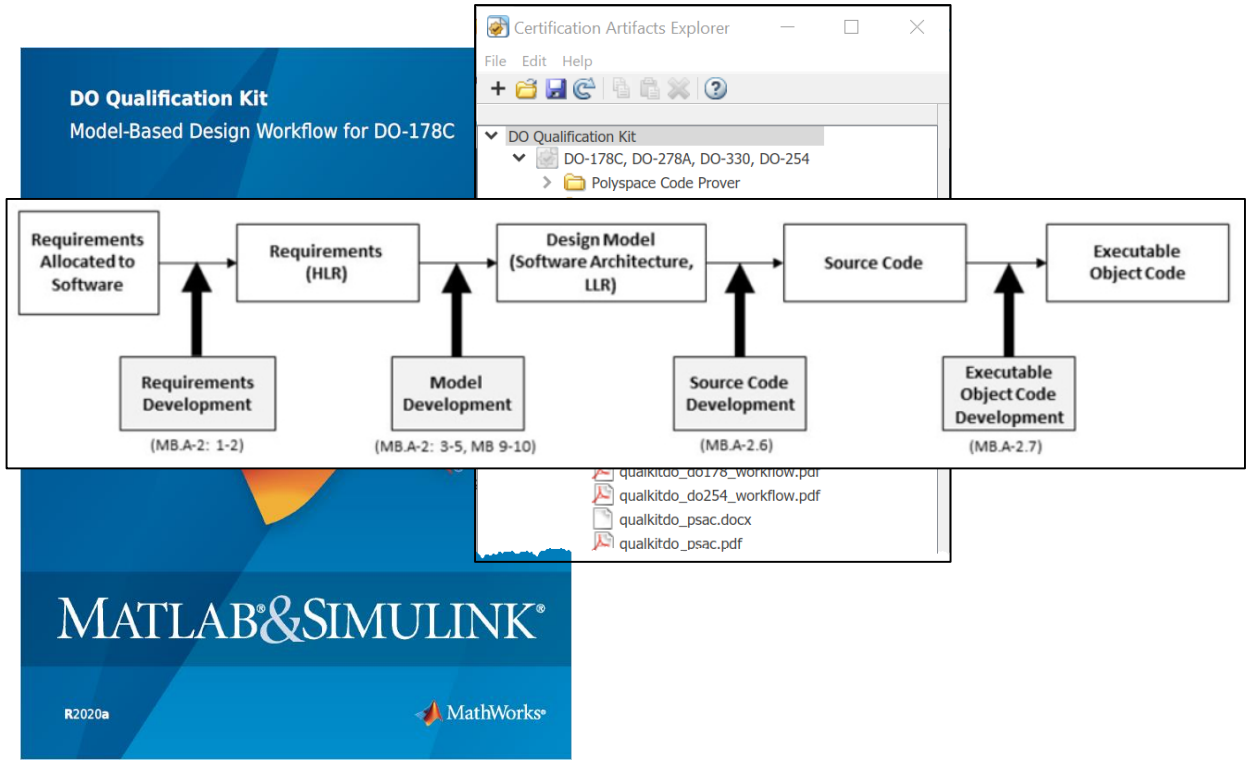
- Qualify code generation and verification products
- Includes documentation, test cases and procedures





Qualify tools with IEC Certification Kit and DO Qualification Kit

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



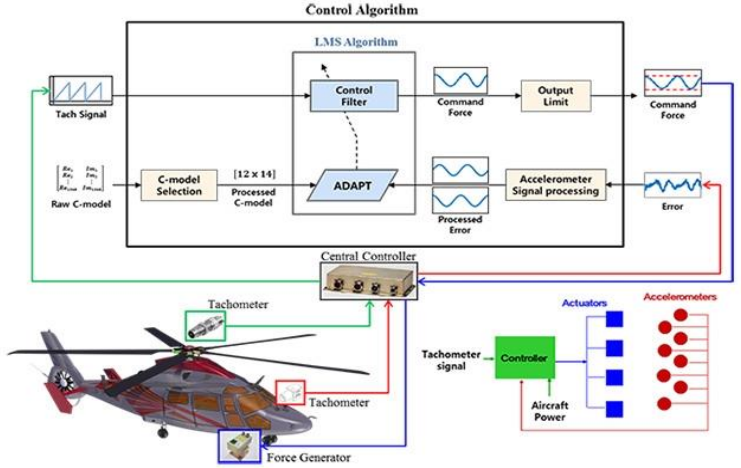
Qualify tools with IEC Certification Kit and DO Qualification Kit

- 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 Design

Kostal's electronic steering column lock module

Korea Aerospace Industries Develops Helicopter Active Vibration Control System Software to DO-178C Standards



The KAI active vibration control system

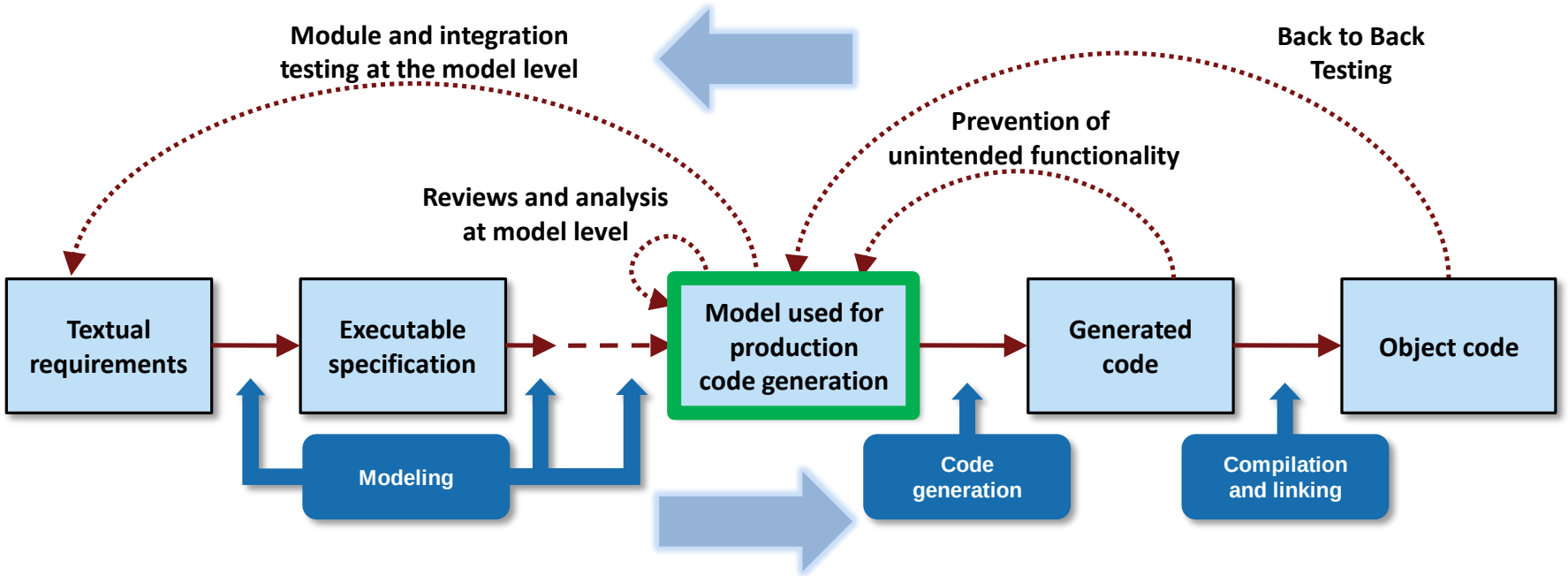
Conform to Certification Standards with Reference Workflow

Model Verification

Discover design errors at design time

Code Verification

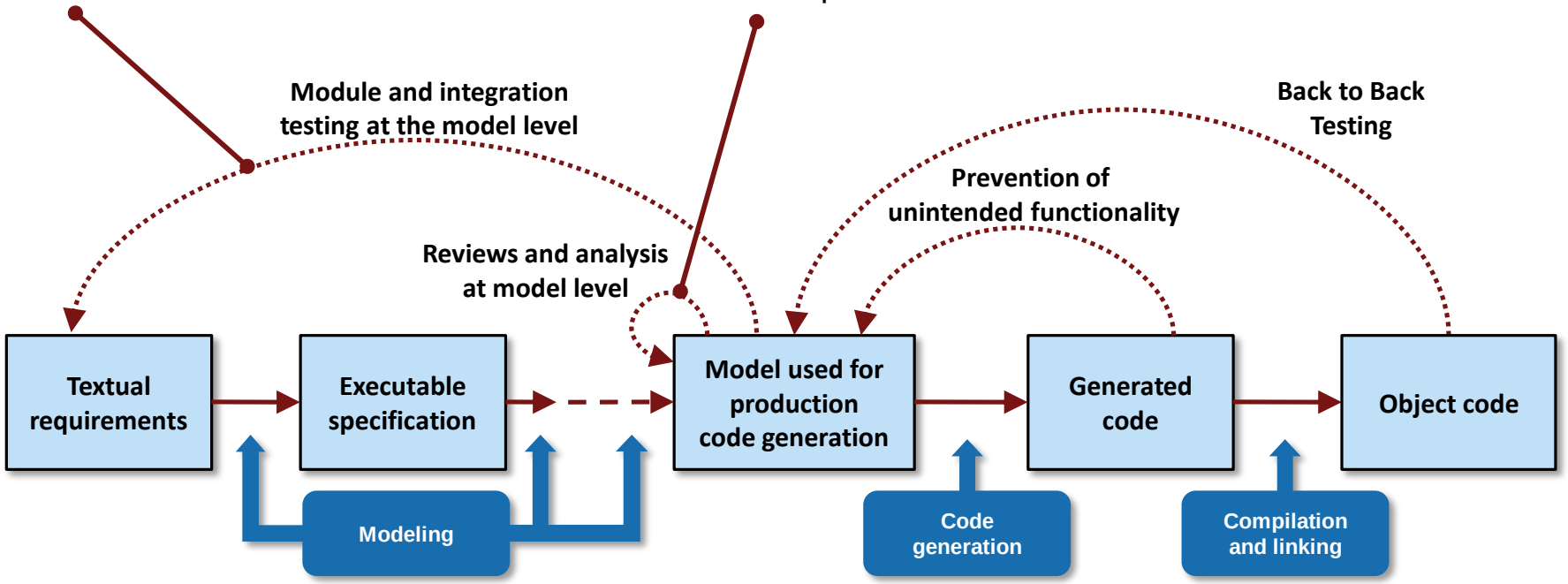
Gain confidence in the generated code



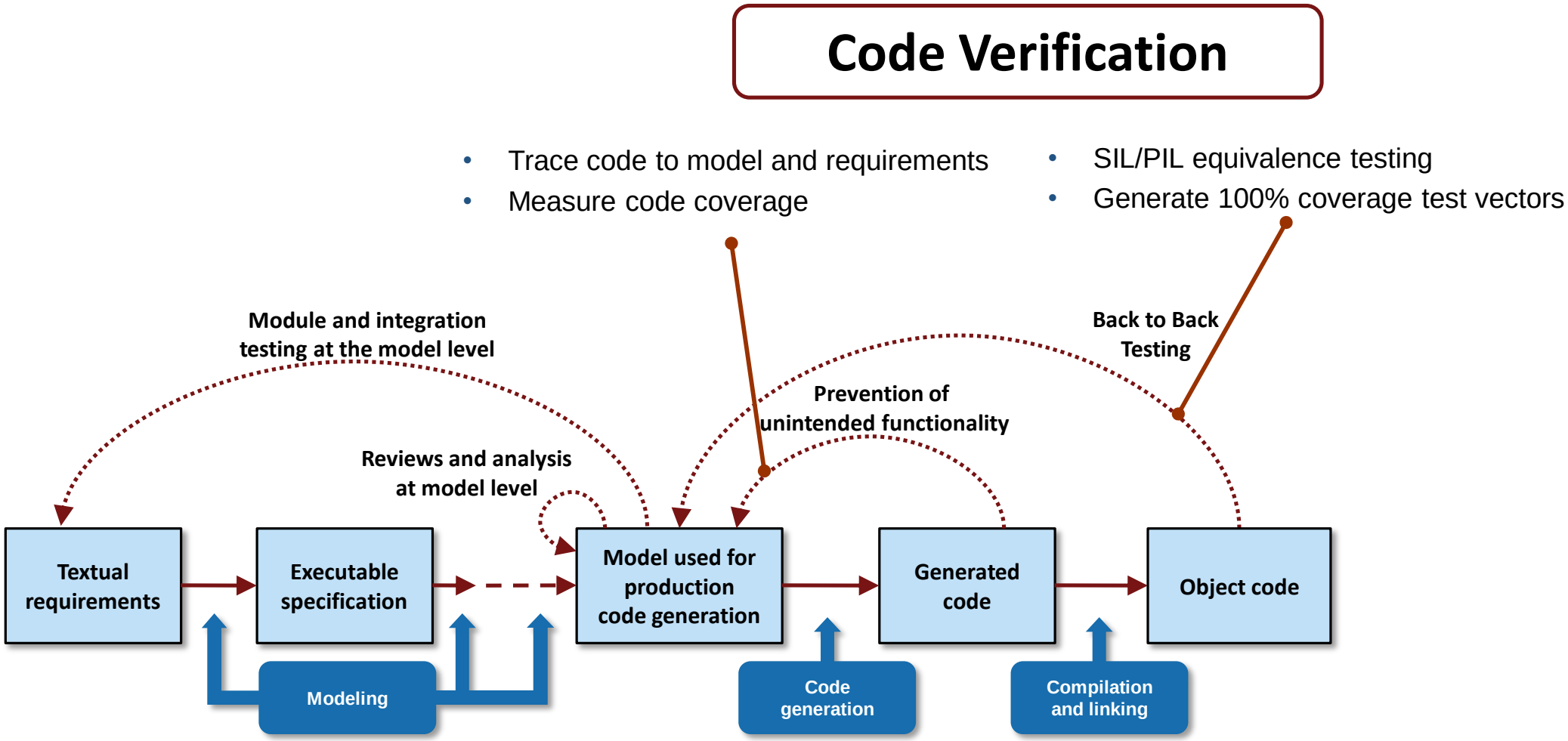
Model Verification: Discover Design Errors at Design Time

Model Verification

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



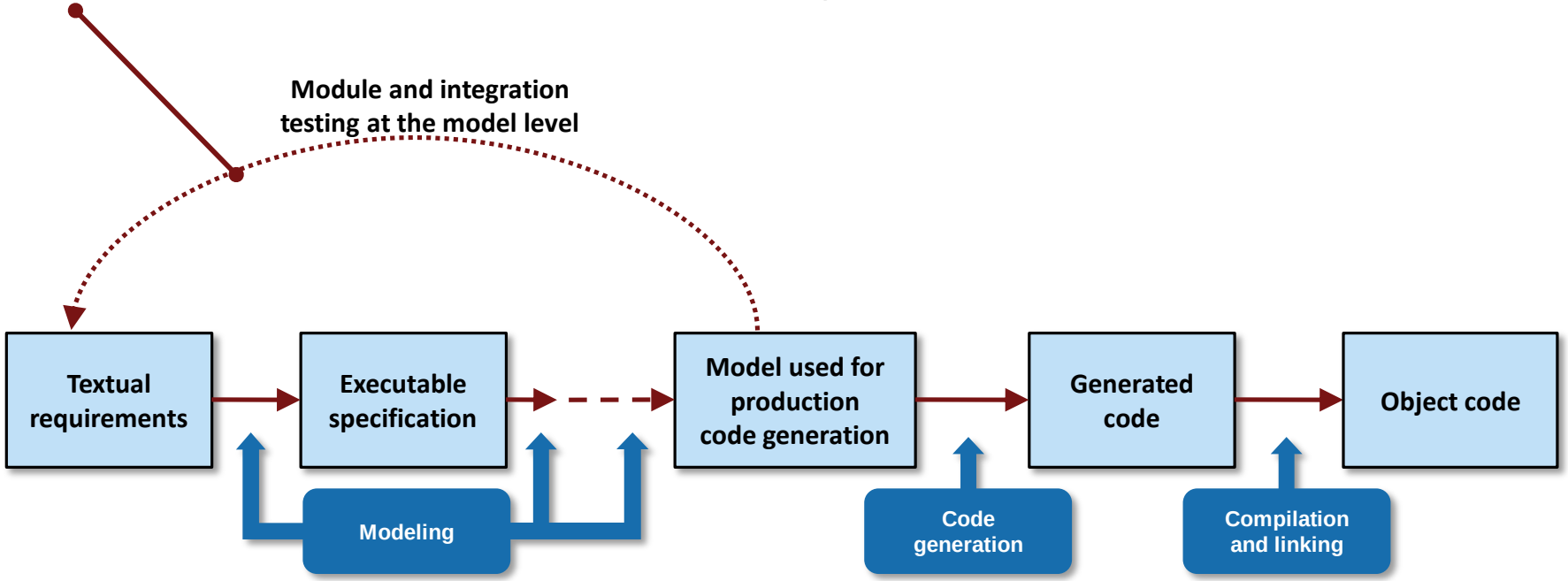
Code Verification: Gain Confidence in the Generated Code



Manage Requirements

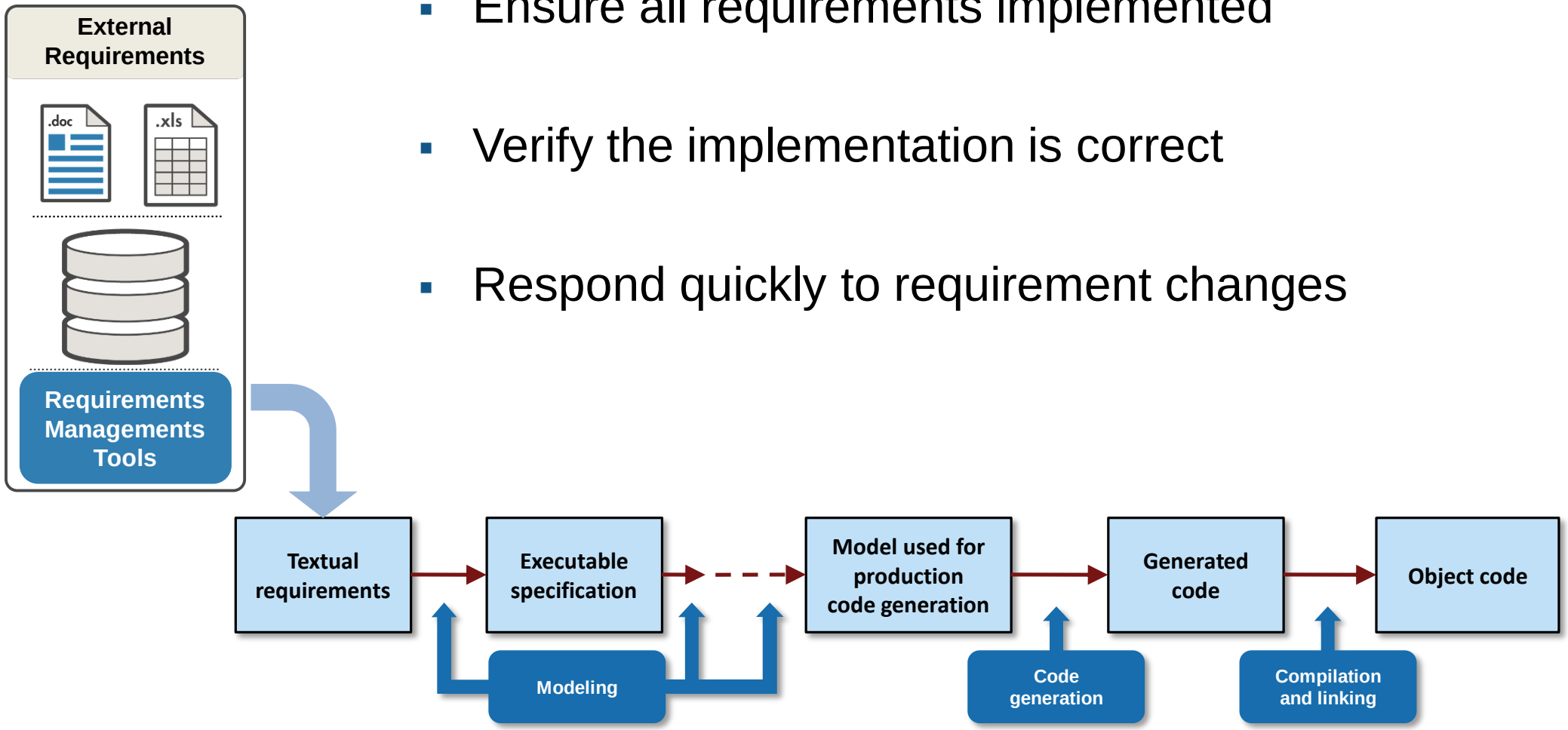
Model Verification

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

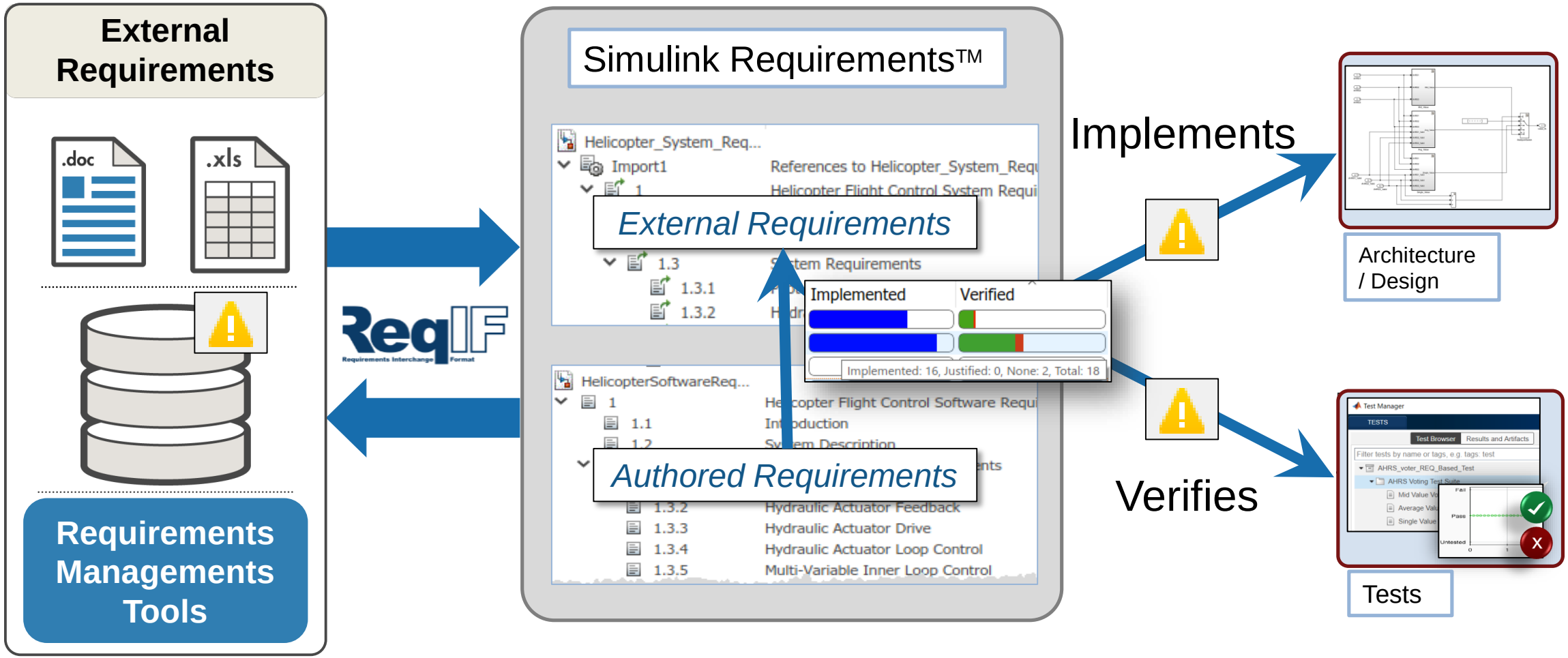


Manage Requirements

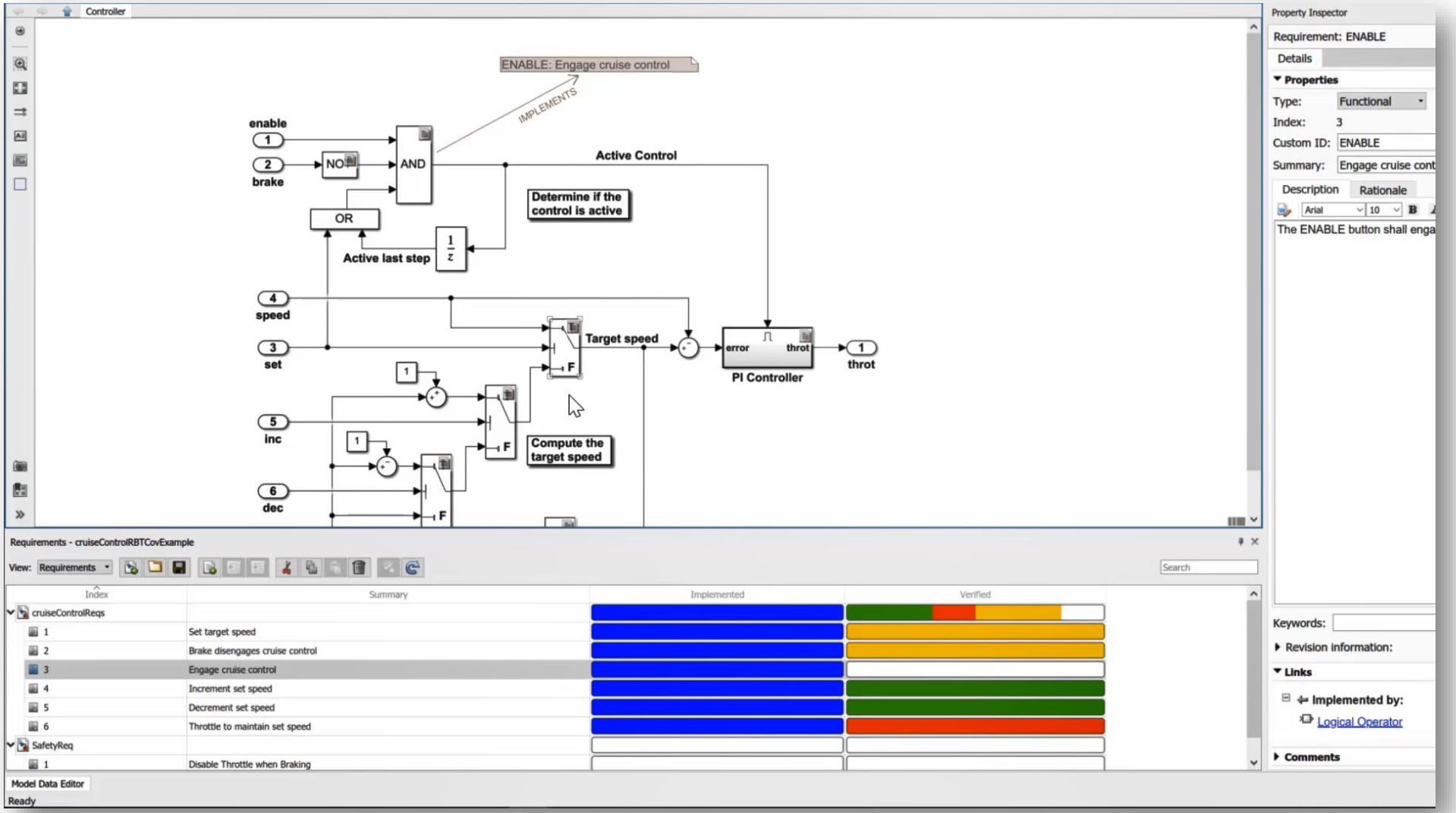
- Ensure all requirements implemented
- Verify the implementation is correct
- Respond quickly to requirement changes



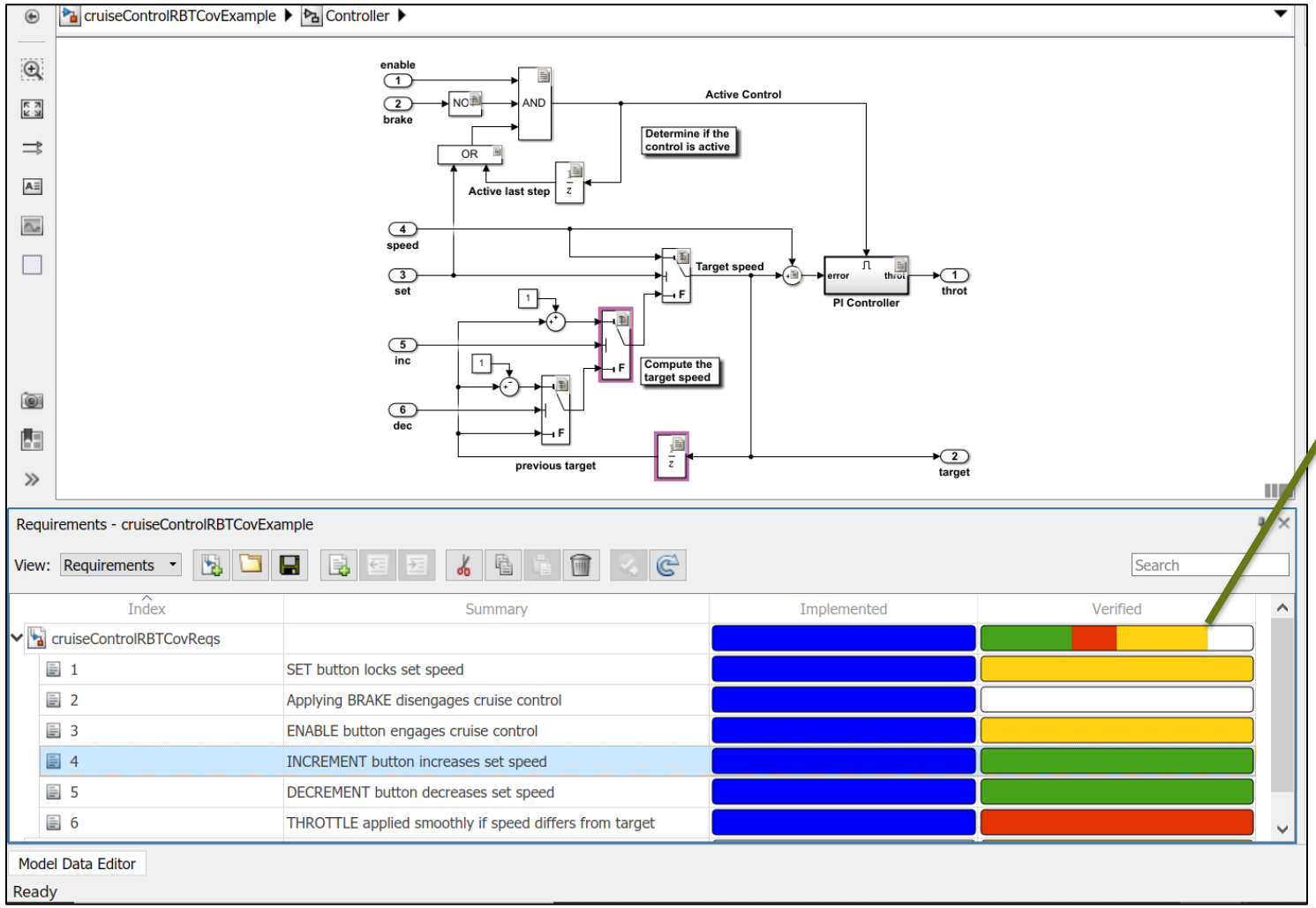
Work with Requirements, Architecture and Design Together

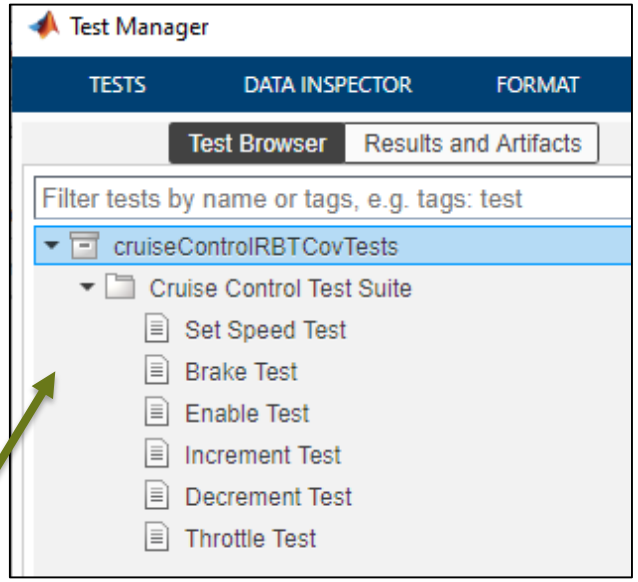


Demo: Requirements Perspective



Test and Requirements Traceability

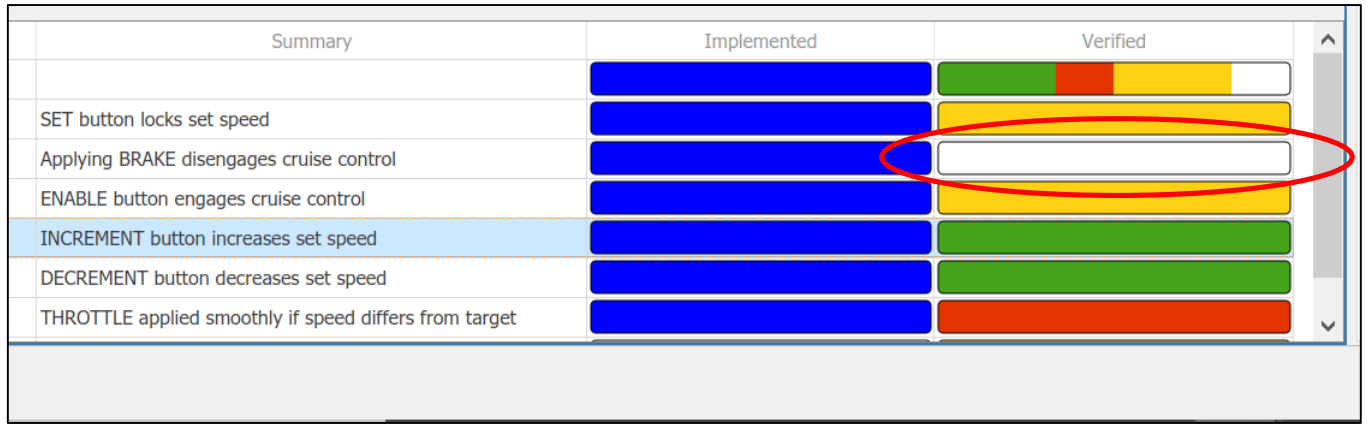




Verification Status

- Passed
- Failed
- Unexecuted
- Missing

Review and Analyze Traceability with Traceability Matrix

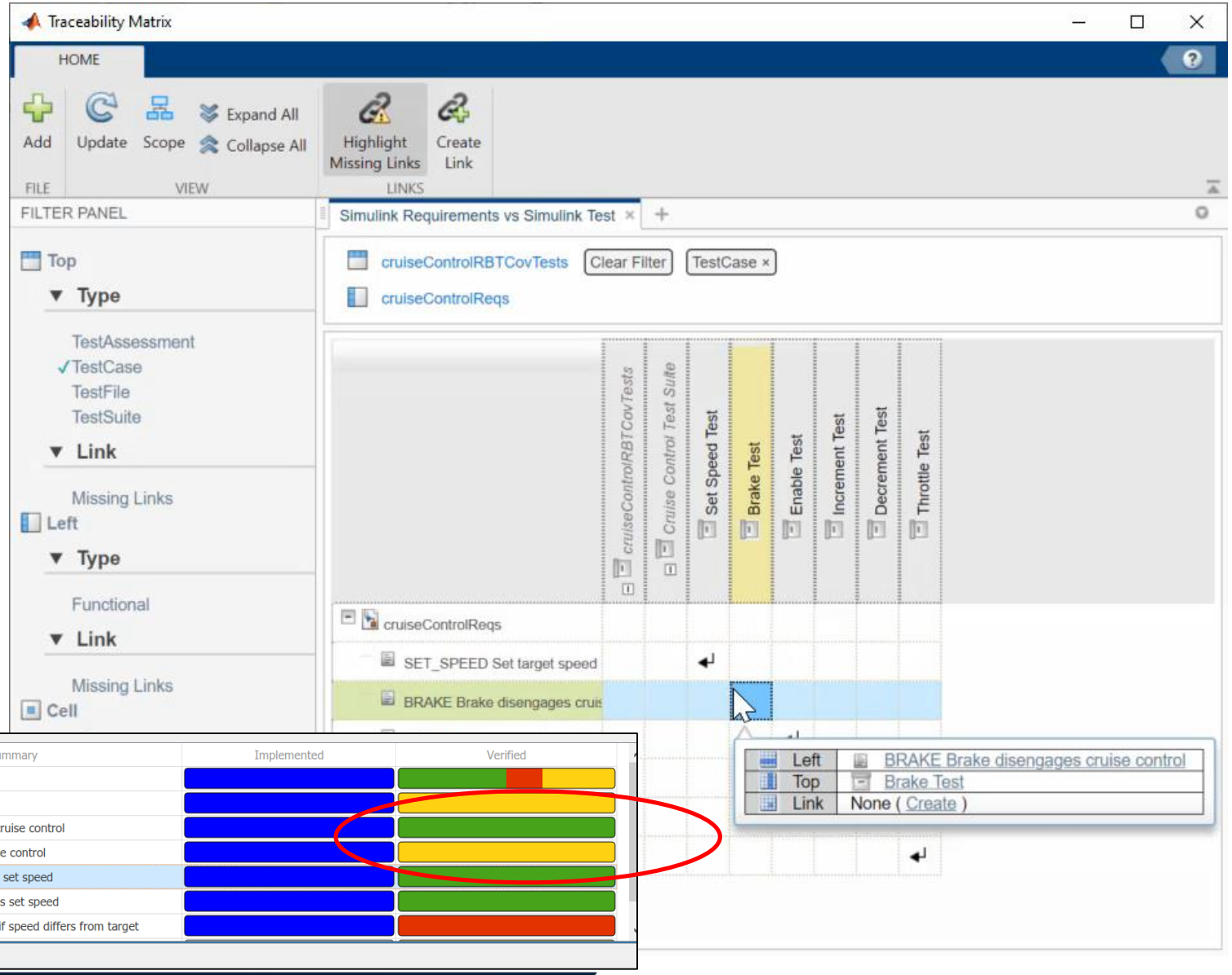


Requirement is missing
link to Test Case

Review and Analyze Traceability with Traceability Matrix

R2020a

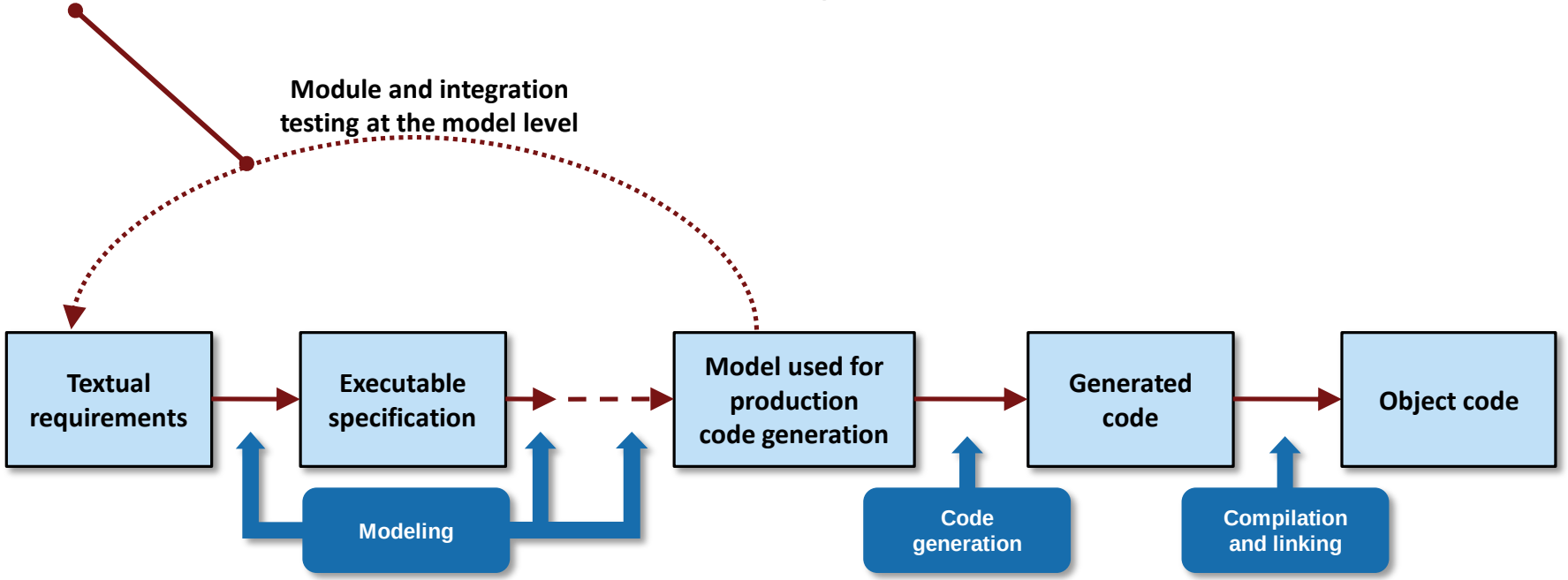
- Review links between different requirements, model, test
- Filter view to manage large sets of artifacts
- Highlight missing links
- Directly add links to address gaps



Systematic Functional Testing of Model

Model Verification

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



Overview to Requirements-Based Test

Goal: Assess extent to which requirements-based tests exercise corresponding designs

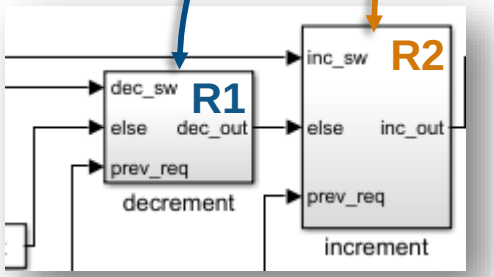
Requirements

Implemented by links

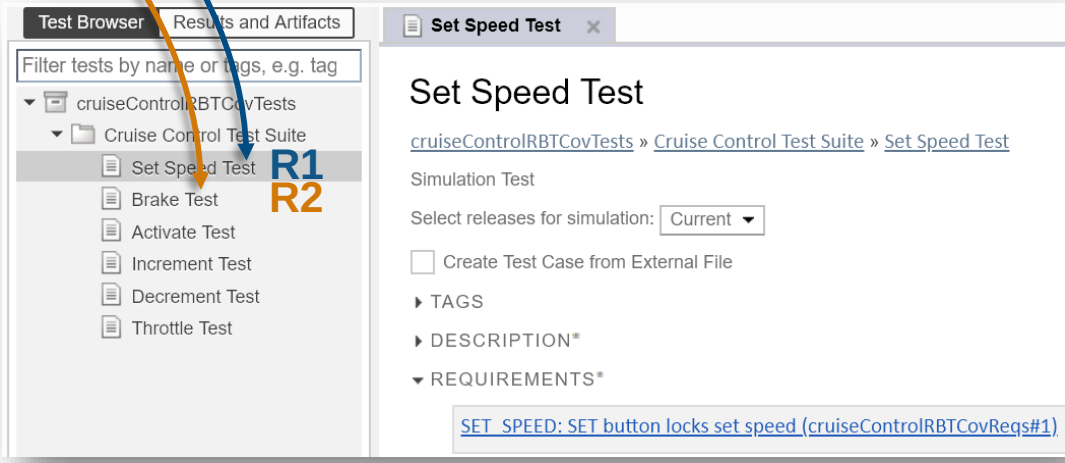
Verified by links

Index	ID	Summary
	cruiseControlRBTCovReqs	
1	SET_SPEED	SET button locks set speed
2	BRAKE	BRAKE disengages cruise control
3	ACTIVATE	ACTIVATE cruise control when enabled, speed set, and no brake
4	INCREMENT	INCREMENT button increases set speed
5	DECREMENT	DECREMENT button decreases set speed
6	THROTTLE	THROTTLE applied smoothly if speed differs from target

Model



Tests



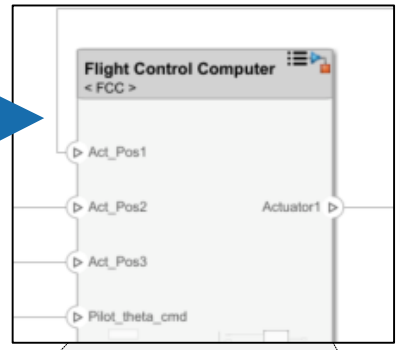
DO-178C 6.4.4.2

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

Requirements Based Verification with Simulink Test

FUNCTIONAL REQUIREMENTS
The flight control system shall ...

Implemented by



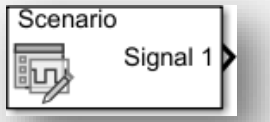
System Composer / Simulink / Stateflow

Verified by

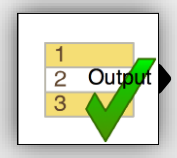
Test Case

Inputs

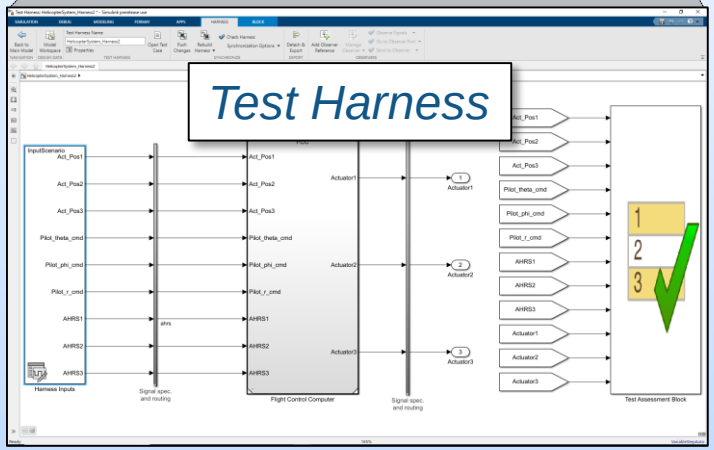
MAT / Excel file (input)



Signal Editor



Test Sequence

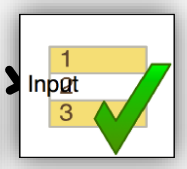


Test Harness

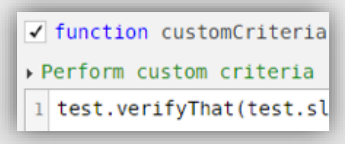
Simulink Test™

Assessments

MAT / Excel File (baseline)



Test Assessments

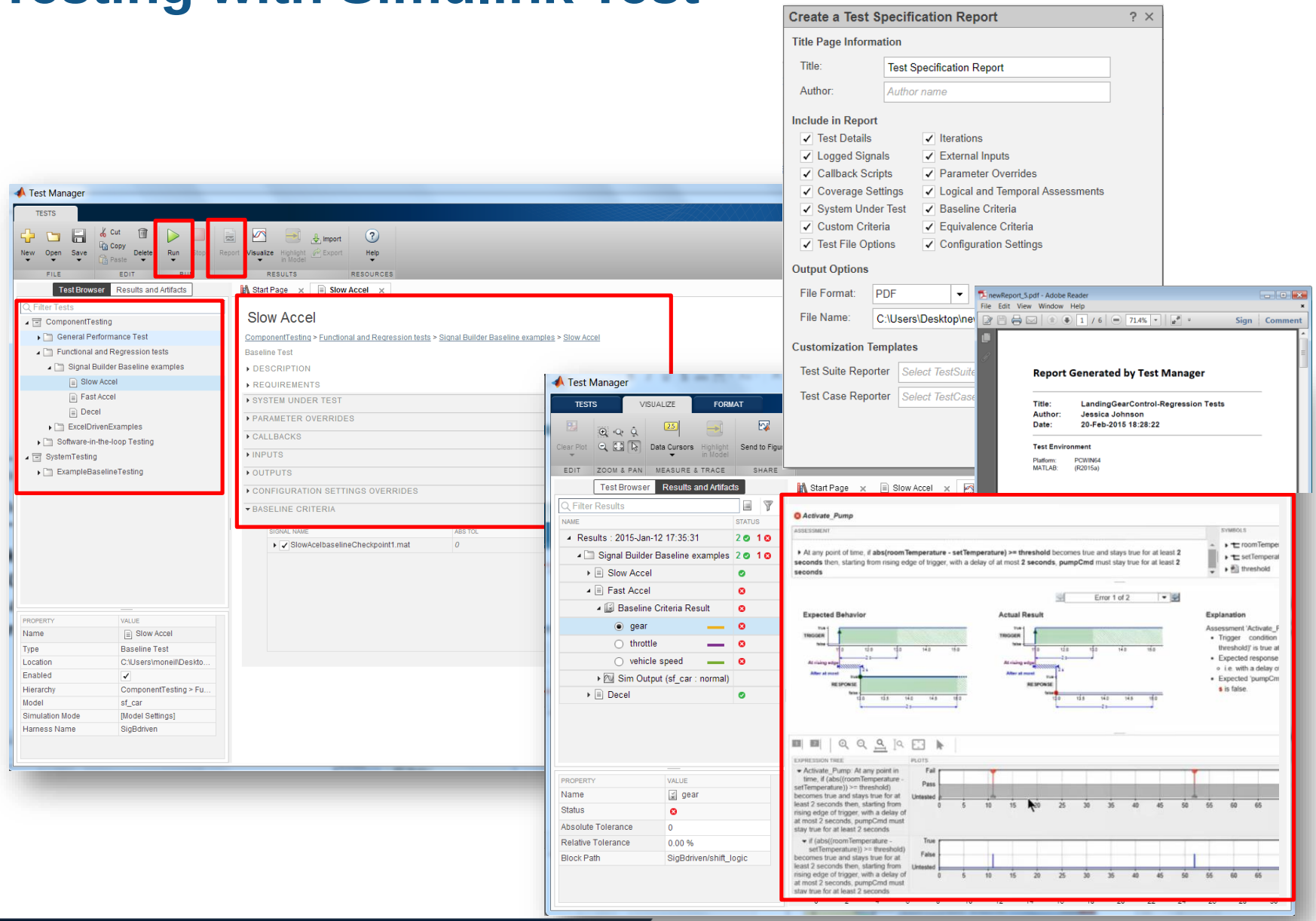


MATLAB Unit Test

Automate Functional Testing with Simulink Test

Test Manager

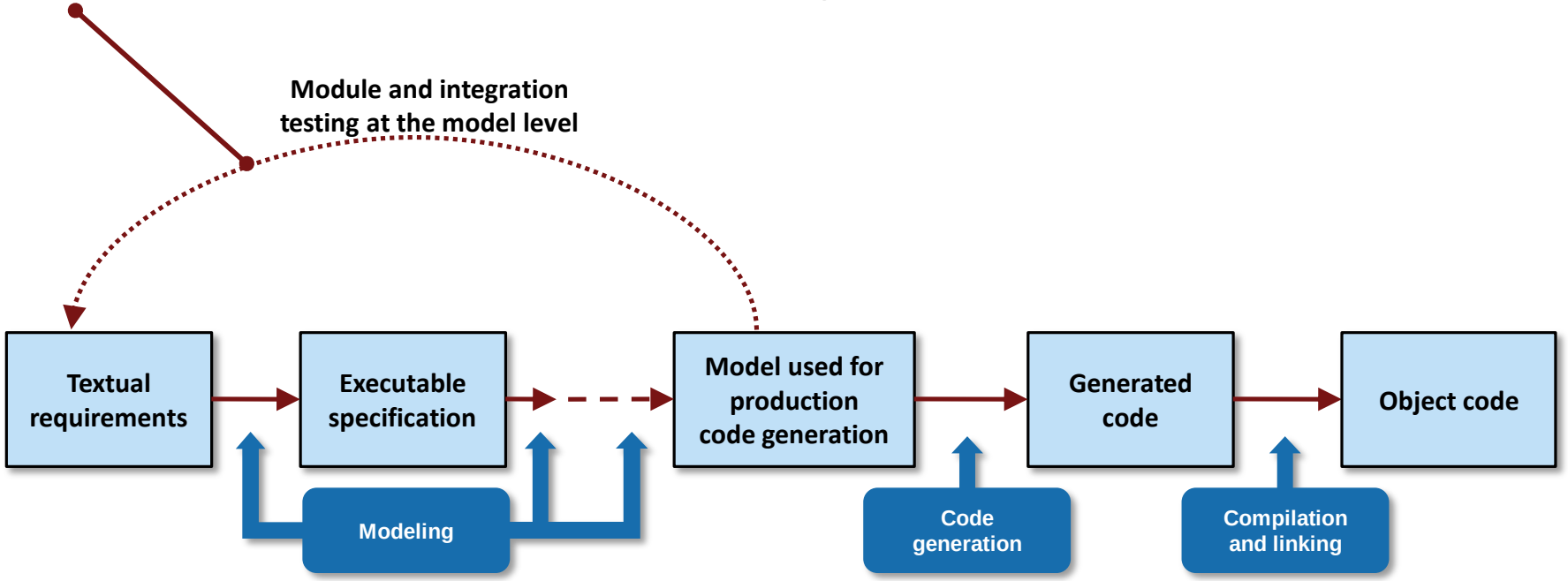
- ✓ Create test cases
- ✓ Group into suites and test files
- ✓ Execute individual or batch
- ✓ View result summary
- ✓ Analyze results
- ✓ Archive, export, report



Measure completeness of testing

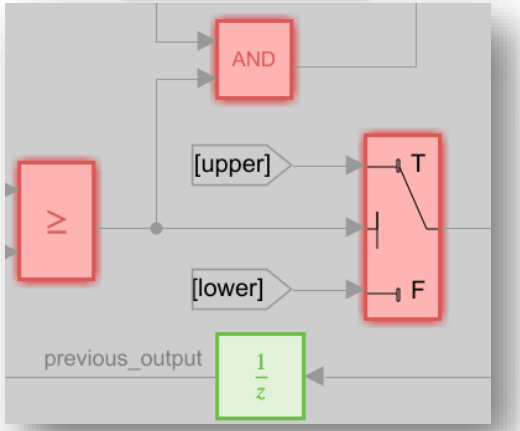
Model Verification

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

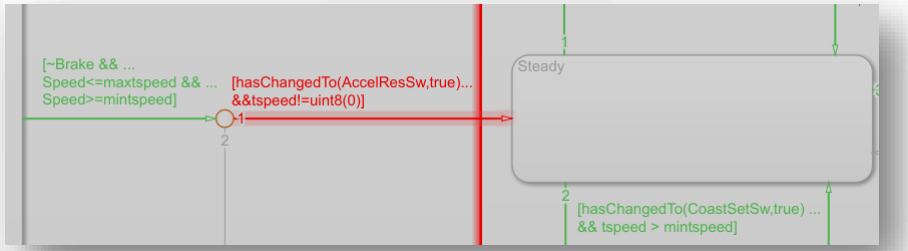


Coverage Analysis to Measure Testing

Simulink®

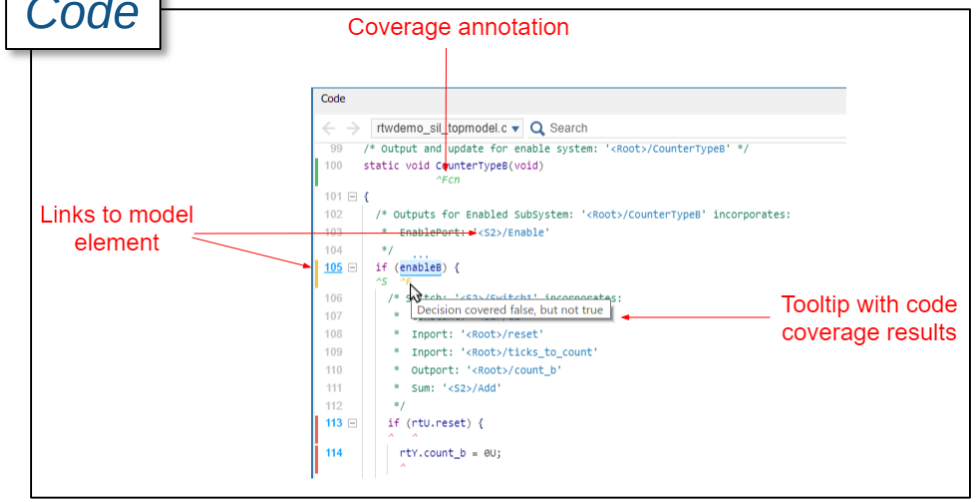


Stateflow™

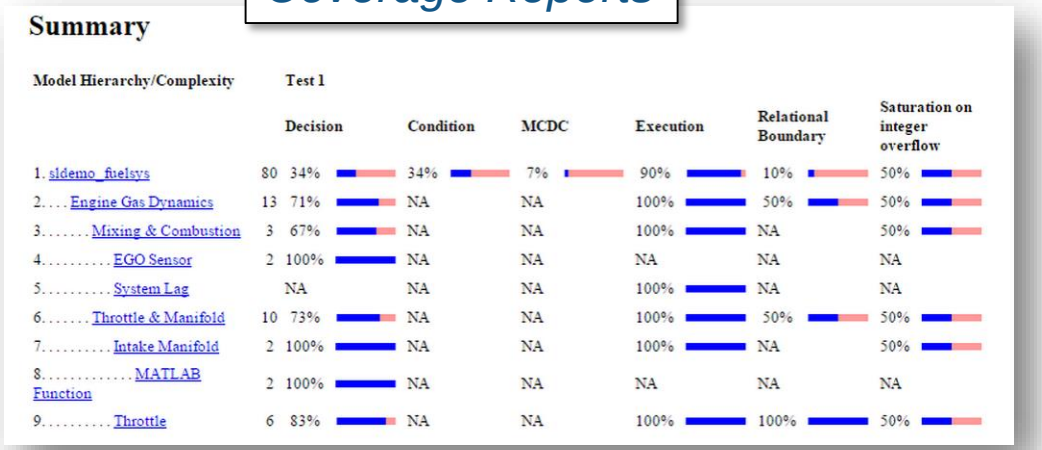


- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors

Code



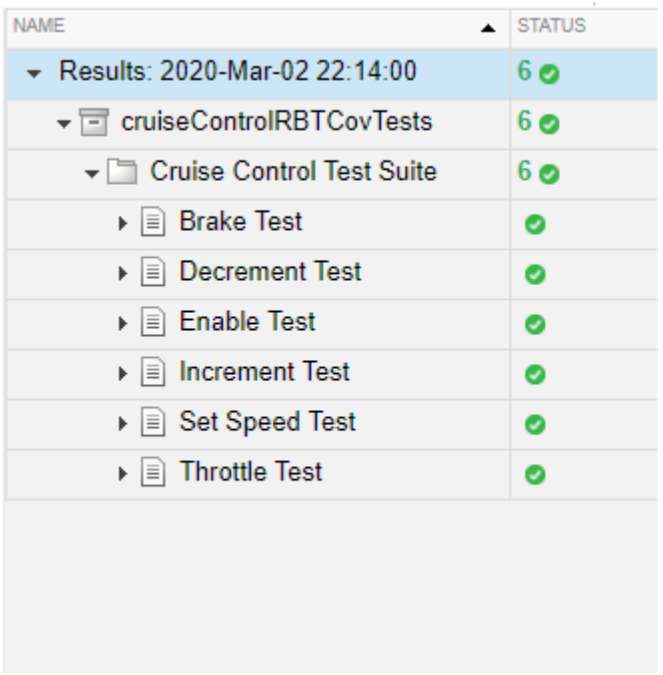
Coverage Reports

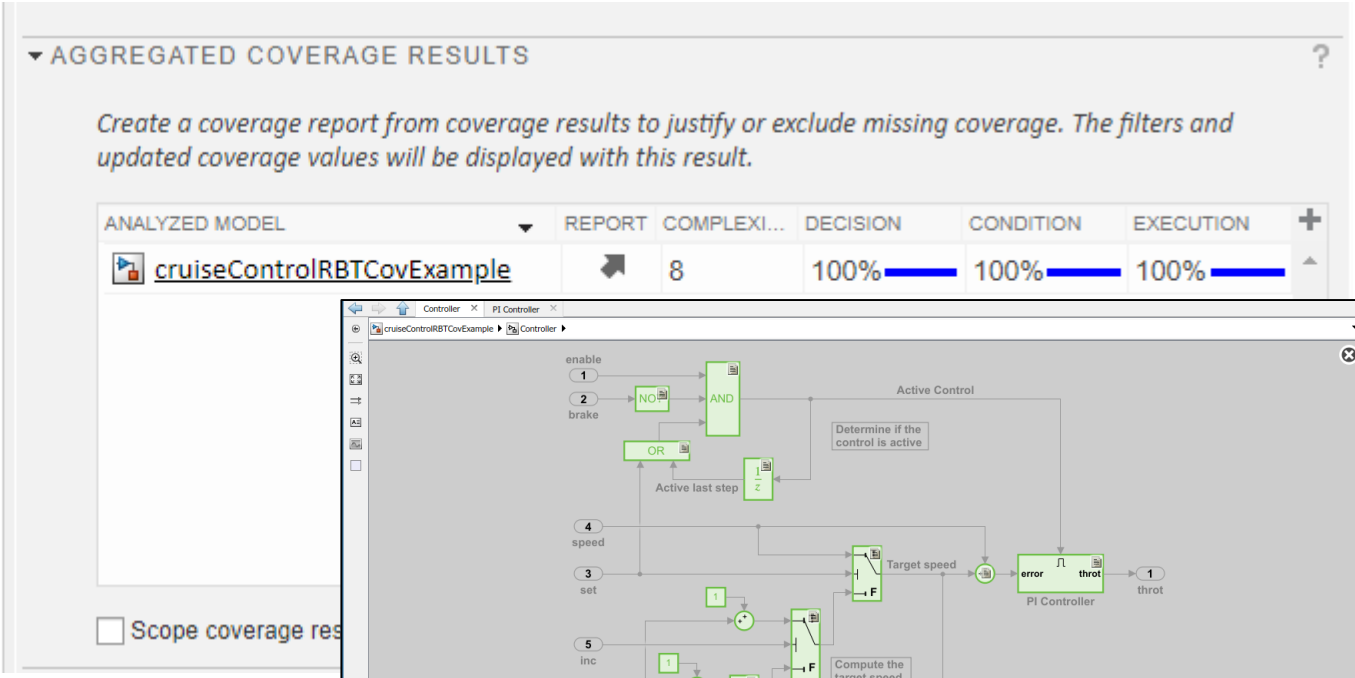


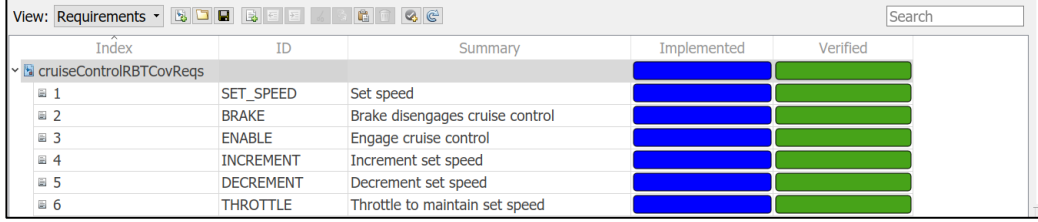
Summary

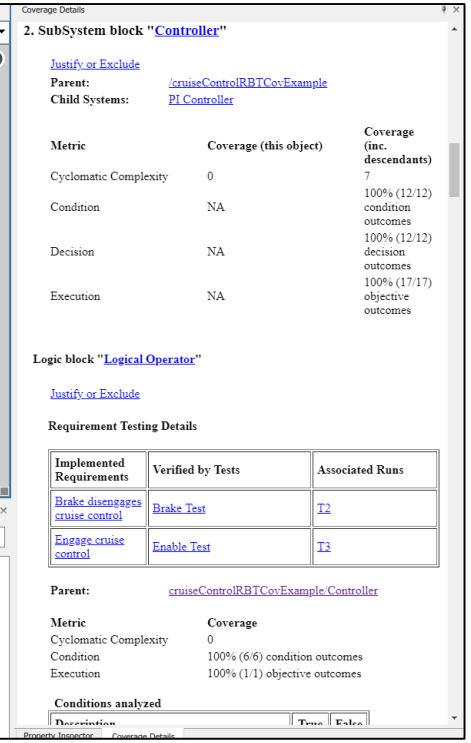
Model Hierarchy/Complexity	Test 1									
	Decision		Condition		MCDC		Execution		Relational Boundary		Saturation on integer overflow	
1. sldemo_fuelsys	80	34%		34%		7%		90%		10%		50%
2. Engine Gas Dynamics	13	71%		NA		NA		100%		50%		50%
3. Mixing & Combustion	3	67%		NA		NA		100%		NA		50%
4. EGO Sensor	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%

Test and Requirements Traceability in Coverage Results









* All tests passed and shows 100% coverage

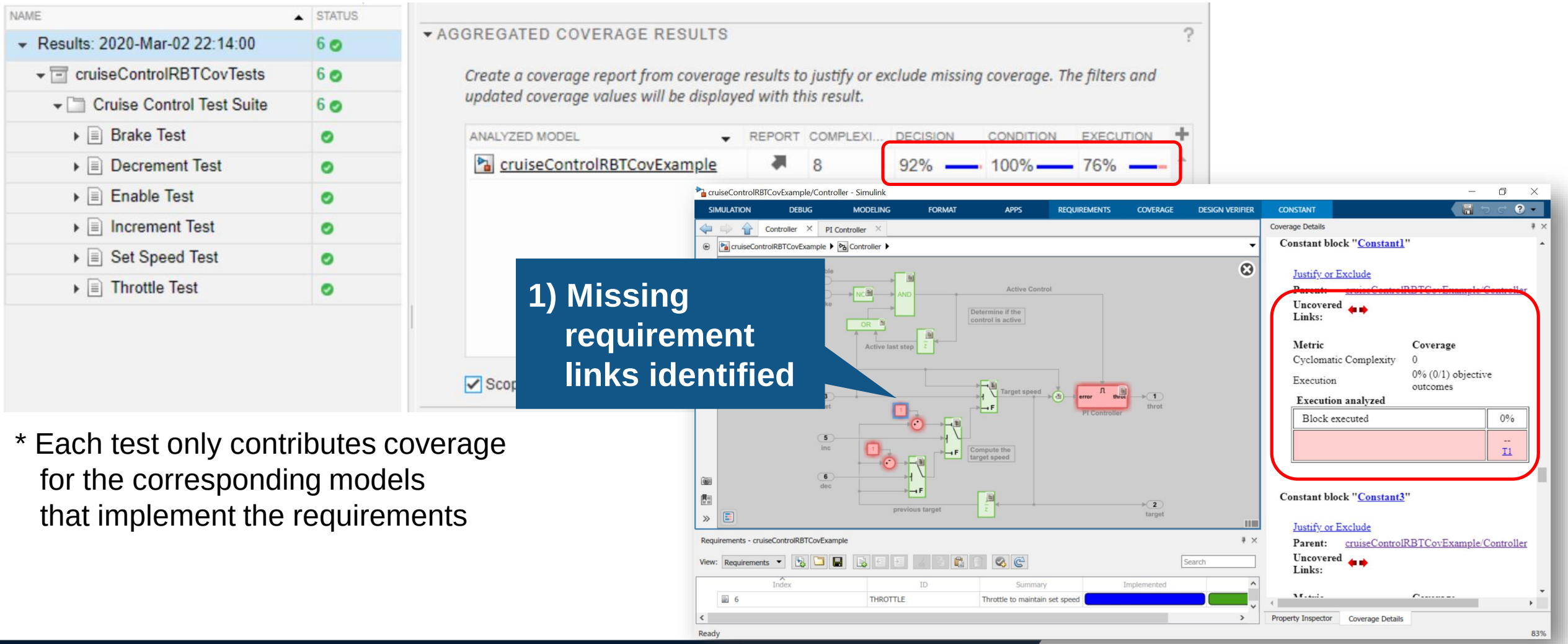
Scoping Model Coverage to Requirements-Based Tests

R2020a

- Checking coverage of tests with the linked requirements
  1) Testing(covered) with missing requirements
  2) Testing(covered) with requirements but not by linked tests
  3) Not testing(not covered)

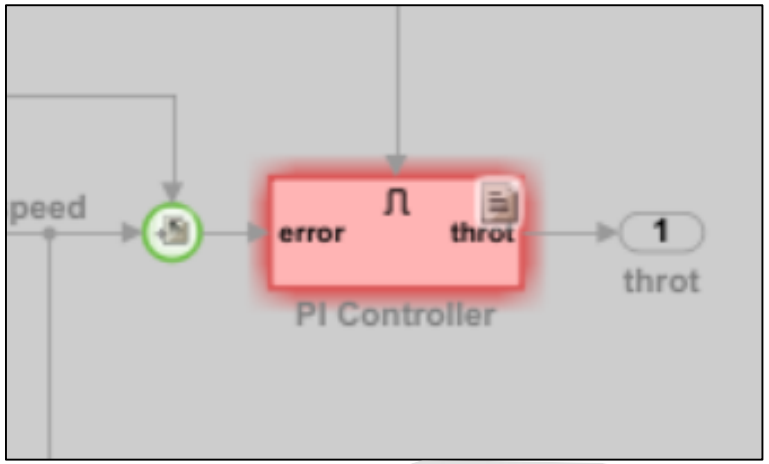
Scoping Model Coverage to Requirements-Based Tests

R2020a



* Each test only contributes coverage for the corresponding models that implement the requirements

Test and Requirements Traceability in Coverage Results

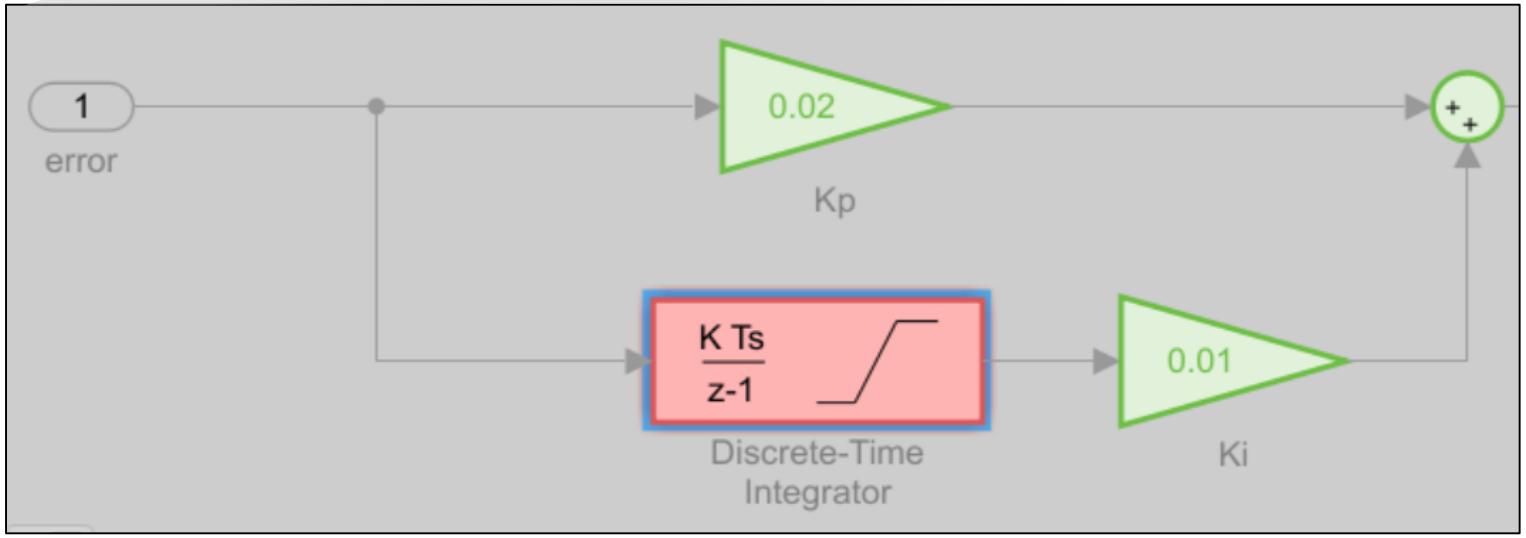


3. SubSystem block "PI Controller"

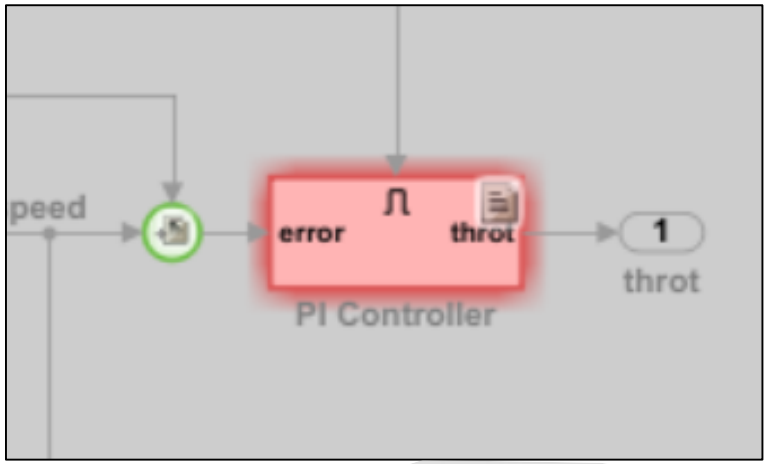
Justify or Exclude

Requirement Testing Details

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



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

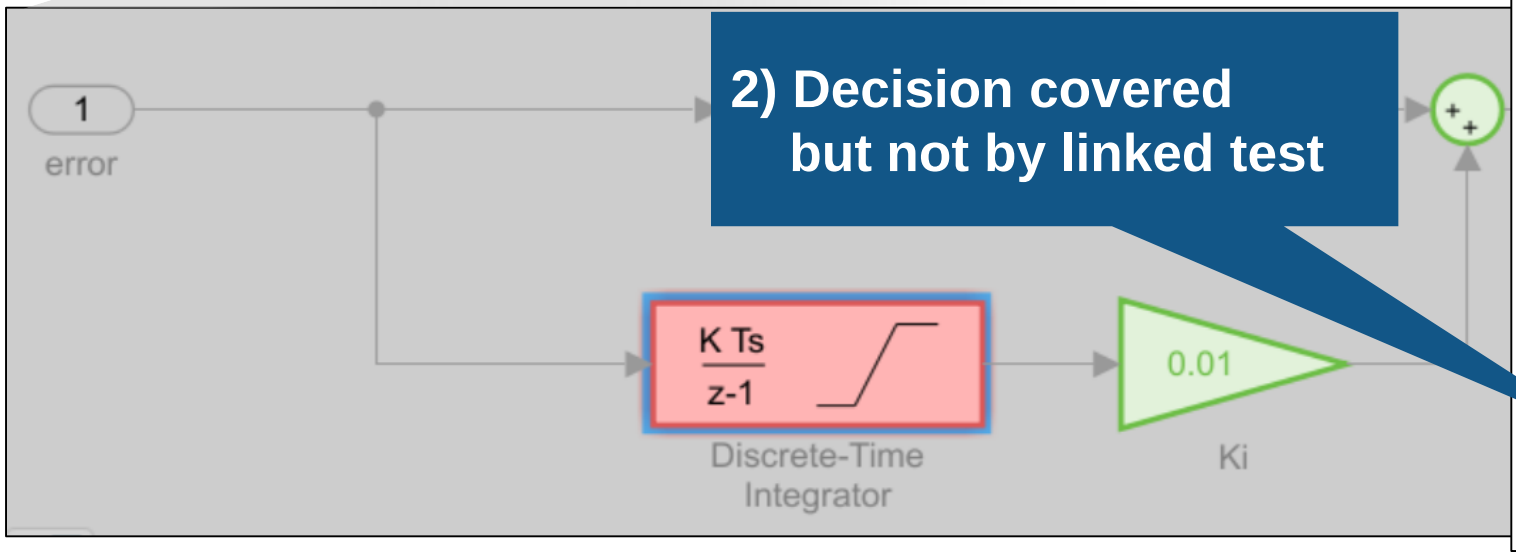
Parent: cruiseControlRBTCovExample/Controller/PI Controller

Uncovered Links:

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

Decisions analyzed

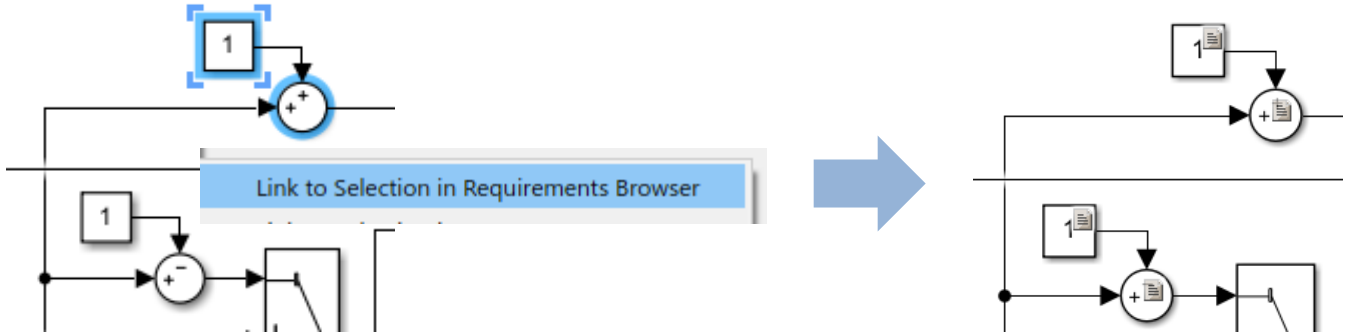
integration result <= lower limit	100%
false	391/801 T6
true	410/801 T6
integration result >= upper limit	50%
false	801/801 T6
true	0/801 T4



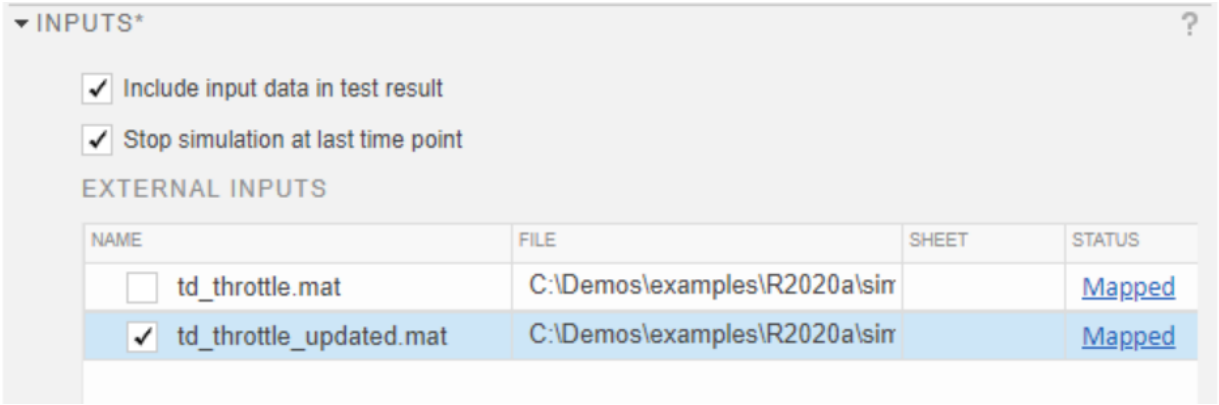
2) Decision covered but not by linked test

Address missing Requirements-Based Test Coverage

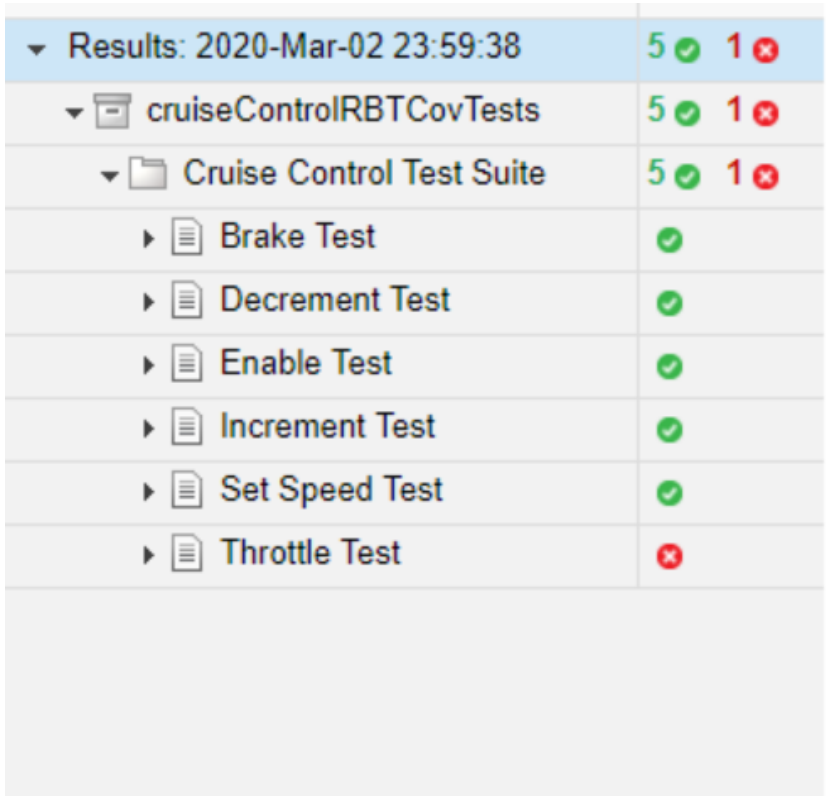
1) Add missing implementation links to requirements

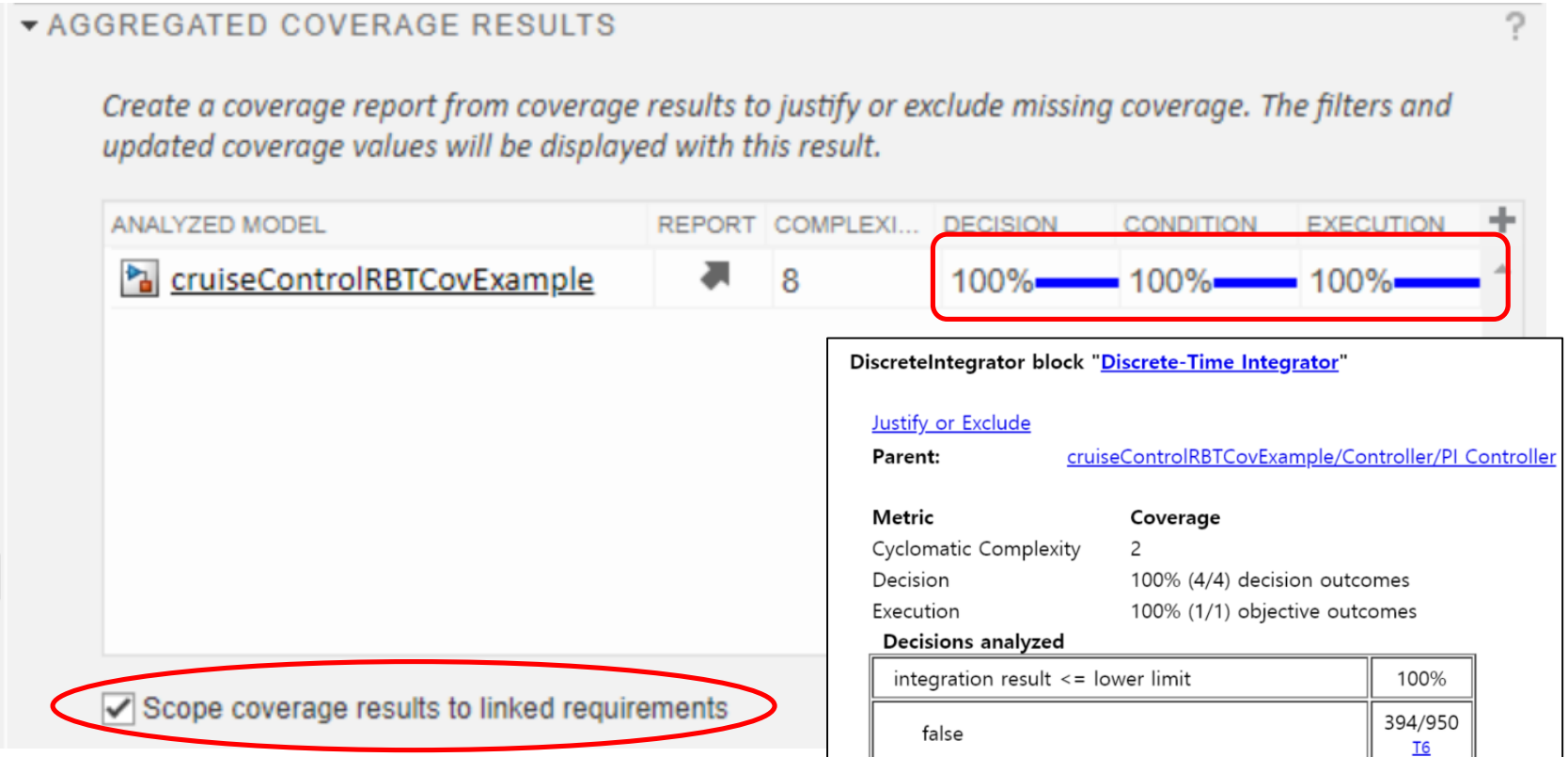


2) Update test to increase target speed to be covered by the linked test

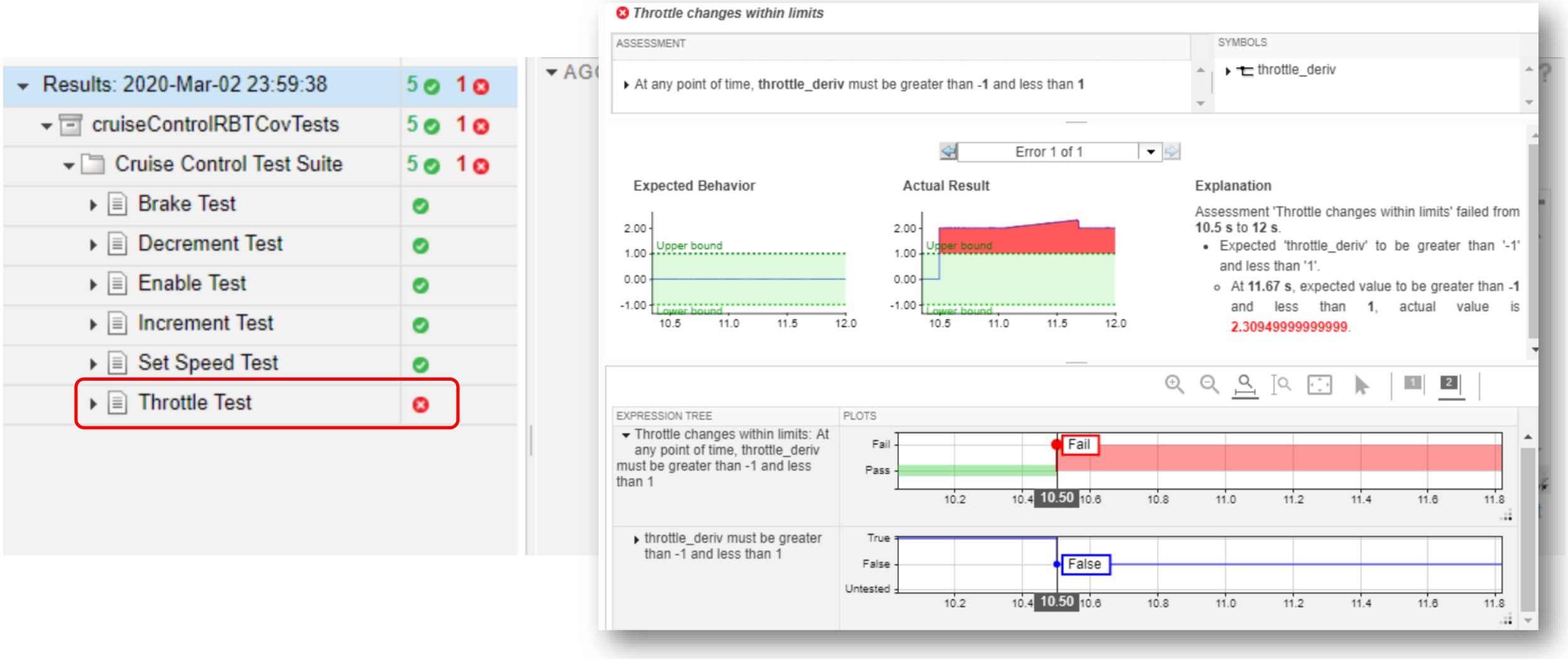


100% Coverage but Testing Identified Error in Implementation





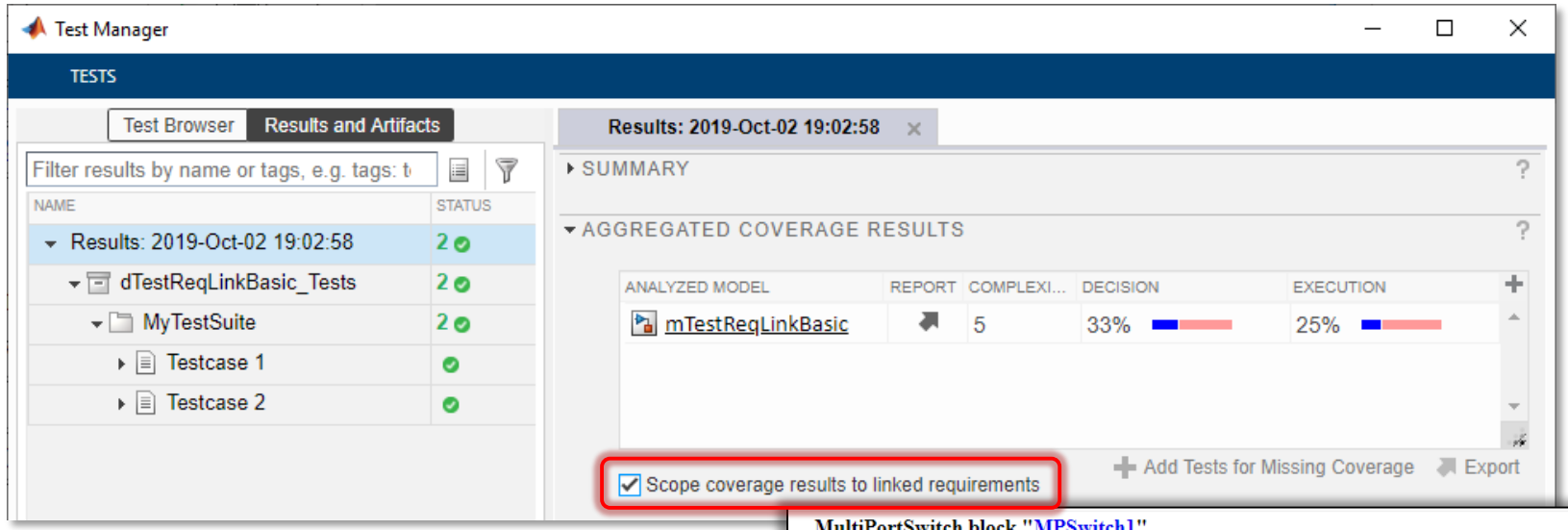
Additional Testing Identified Error in Implementation



https://kr.mathworks.com/help/sltest/ug/test-coverage-for-requirements-based-testing.html?s_tid=srchtitle

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**

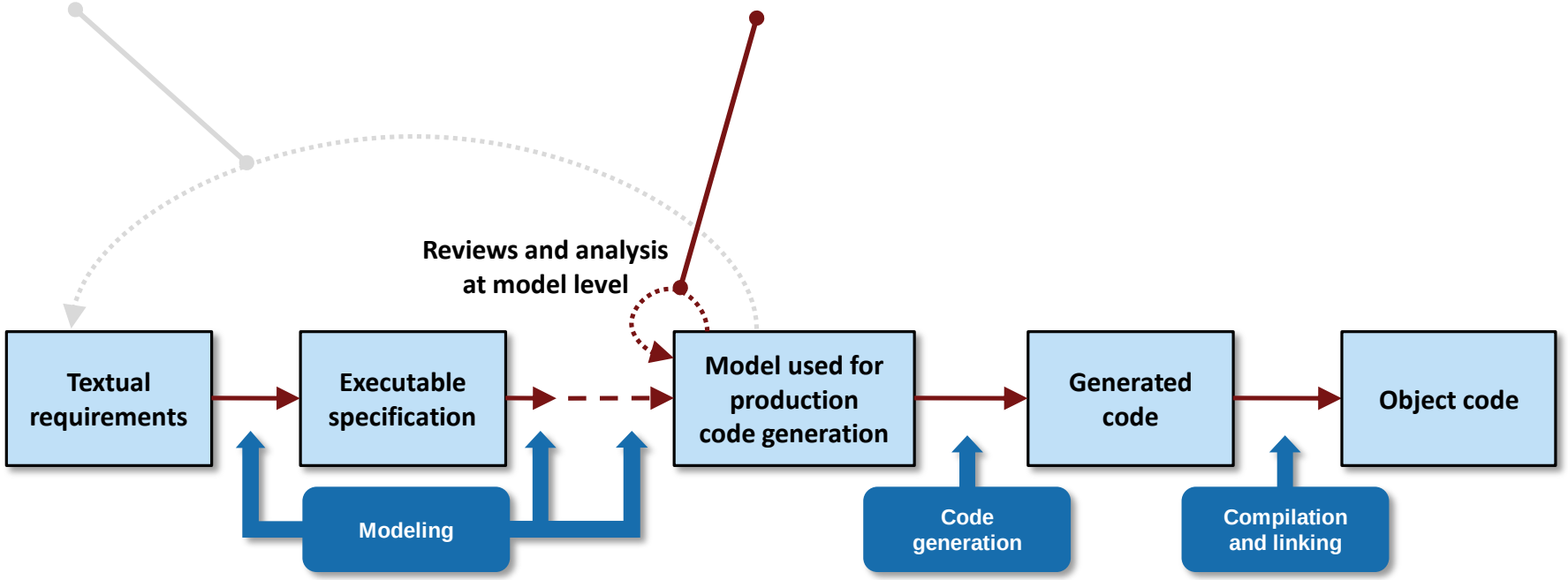
RBT (Requirements-Based Tests)

Check Standard Compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage

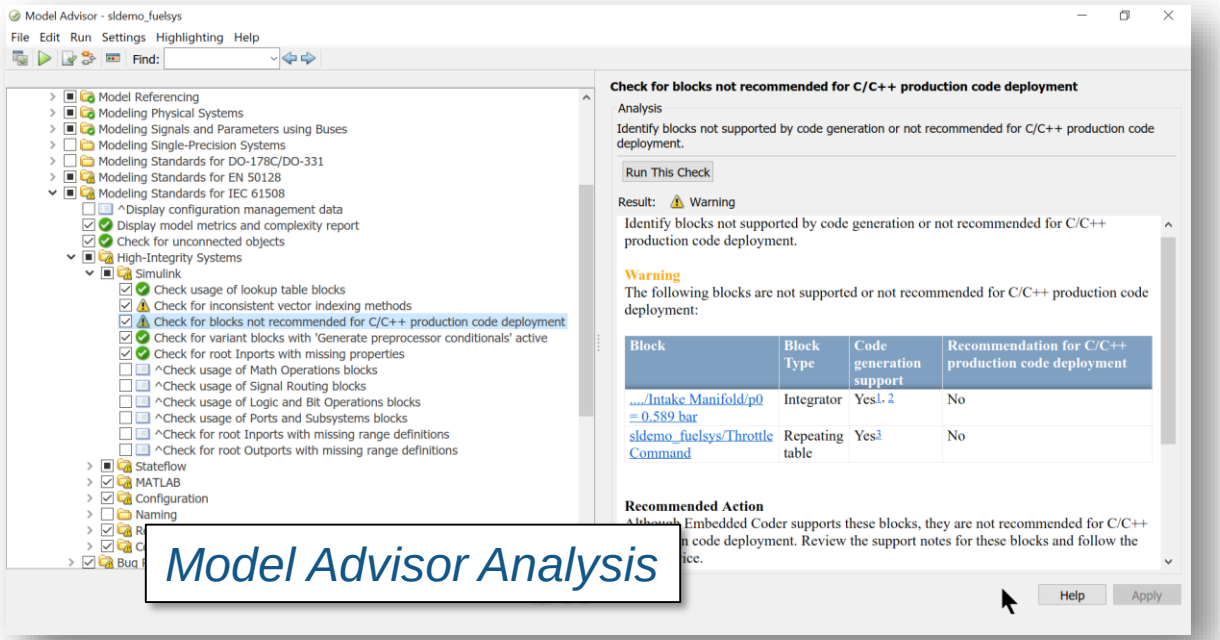
- **Check standard compliance**
- Detect design errors
- Prove model behavior compliance

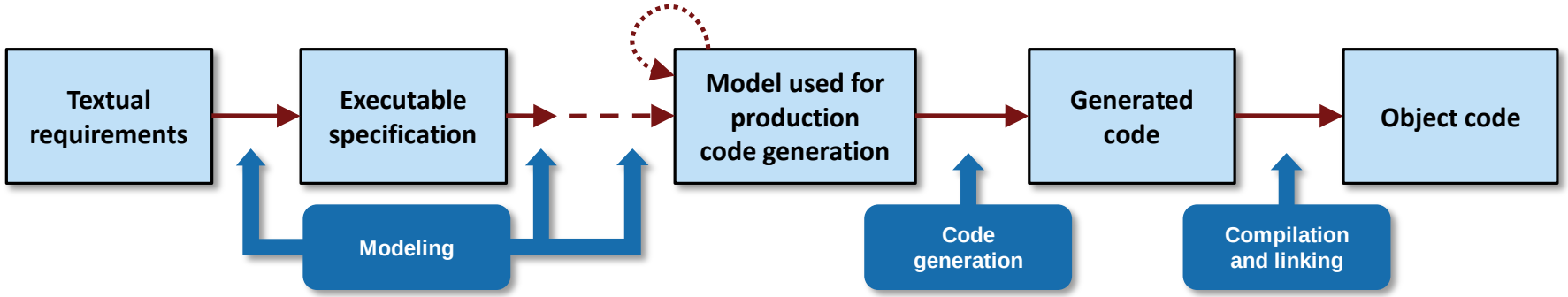


Verify Design to Guidelines and Standards

Check for:

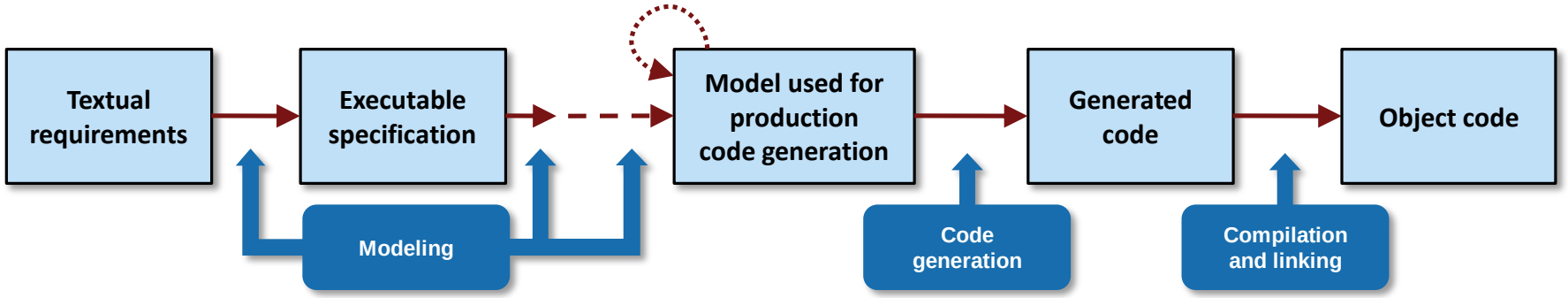
- Readability and Semantics
- Performance and Efficiency
- Clones (for optimization)
- And more……





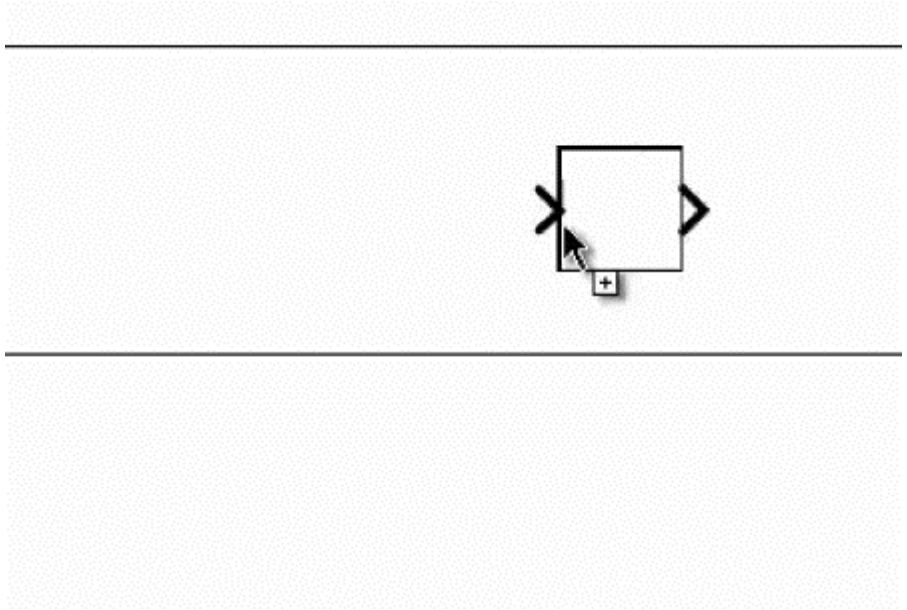
Built in Checks for Industry Standards and Guidelines

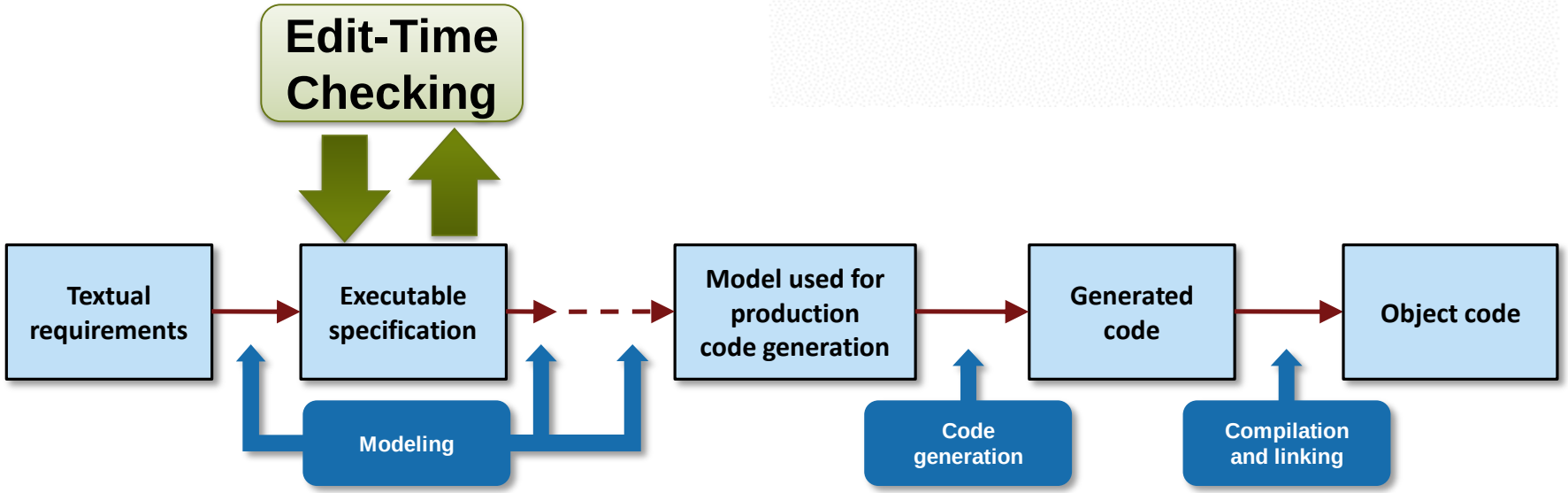
- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128
- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAB (MathWorks Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)



Shift Verification Earlier With Edit-Time Checking

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

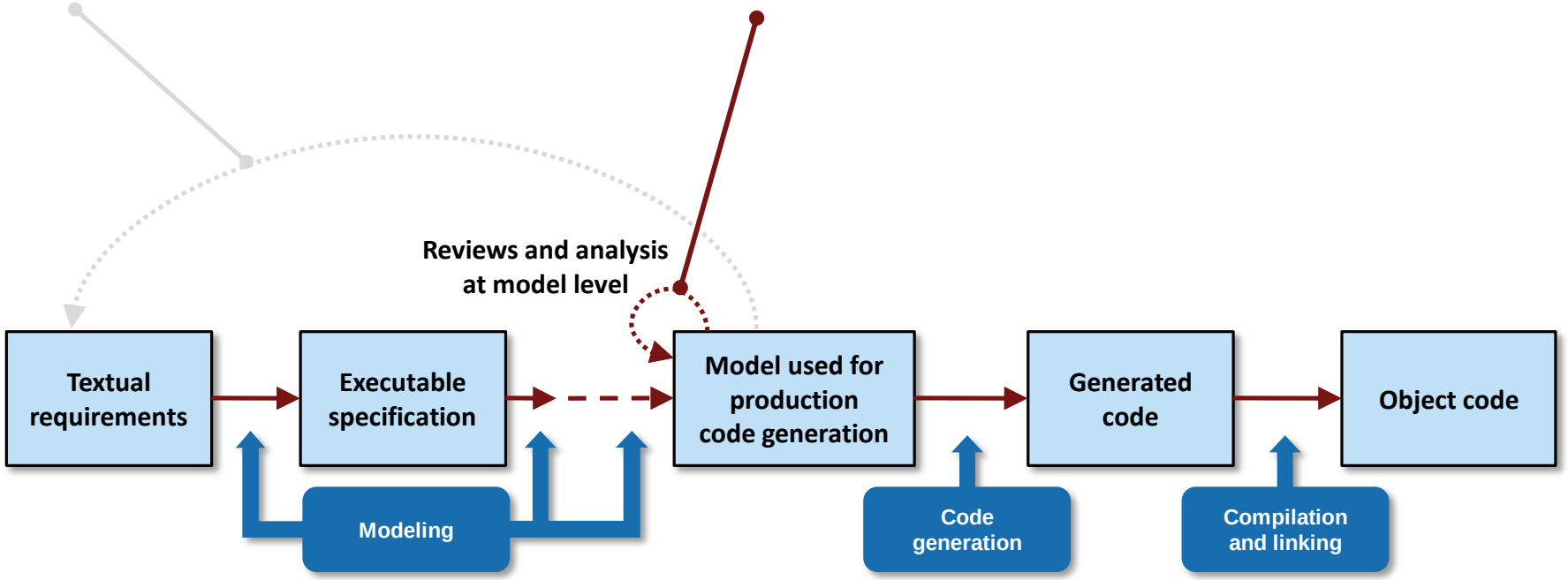




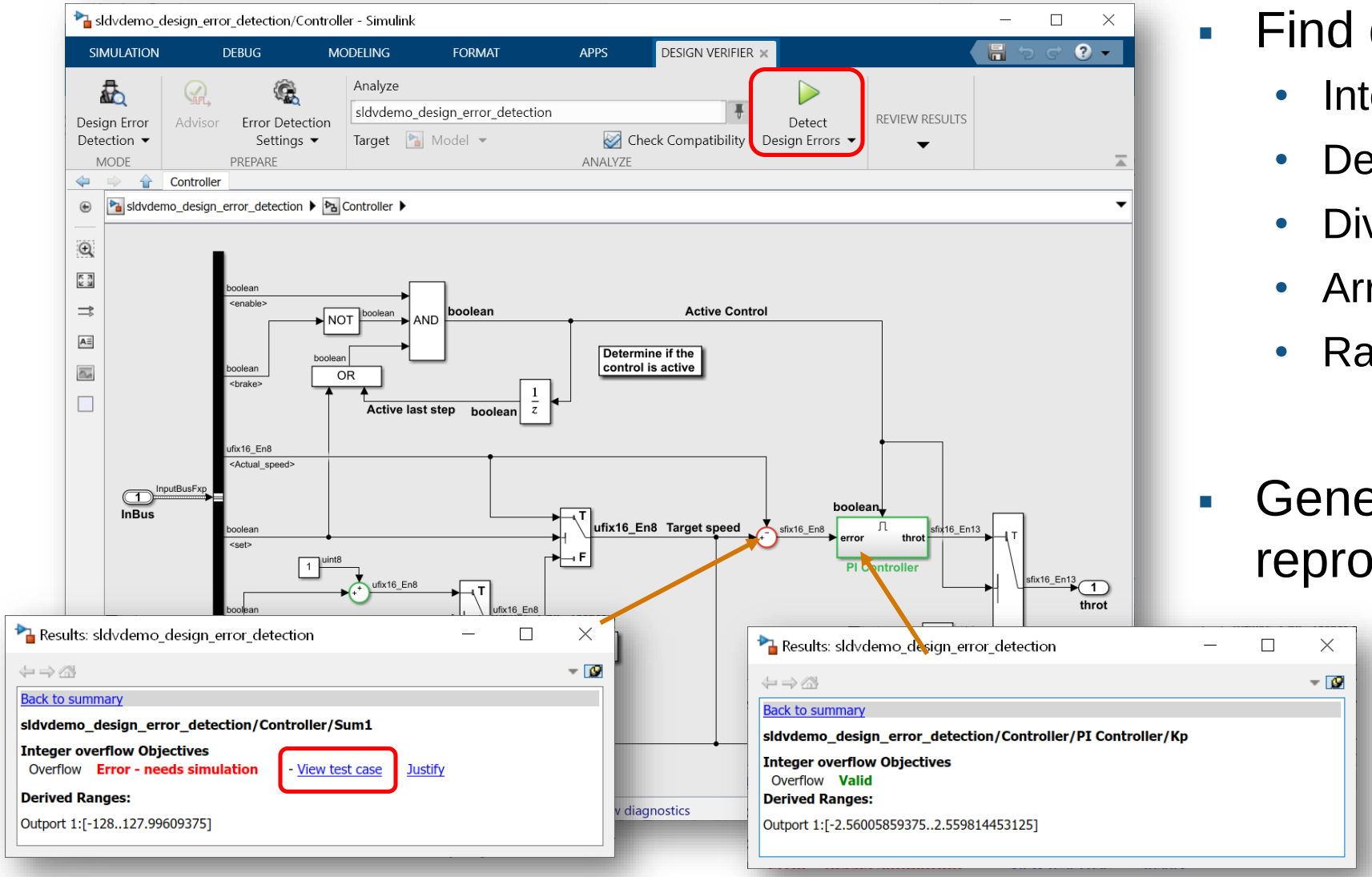
Detect Design Errors with Formal Methods

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

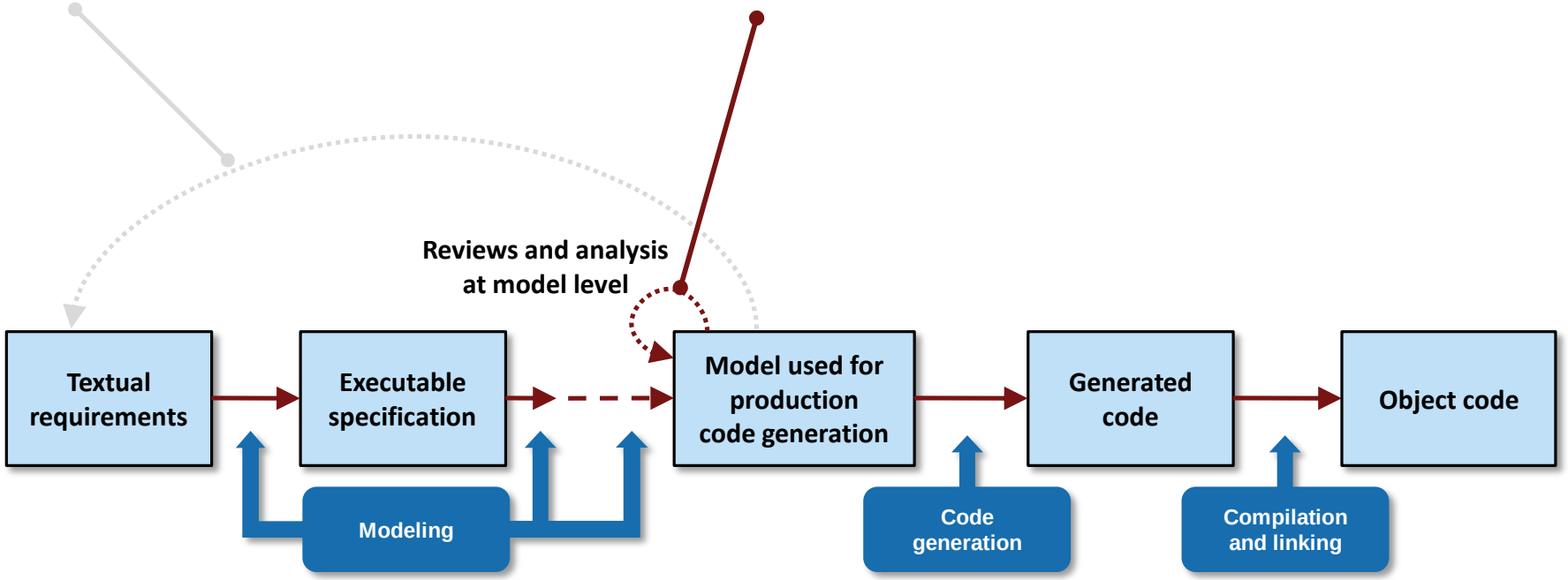


- Find design errors
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations
- Generate counter example to reproduce error

Prove Model Behavior Compliance

Model Verification

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

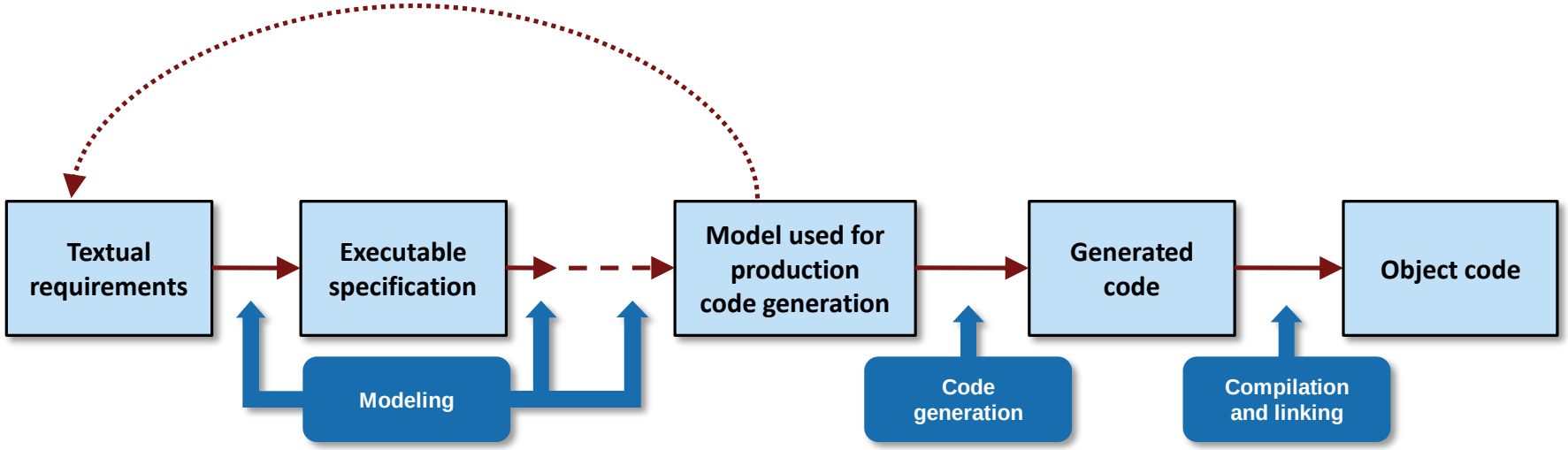


Proving Model Meets Requirements

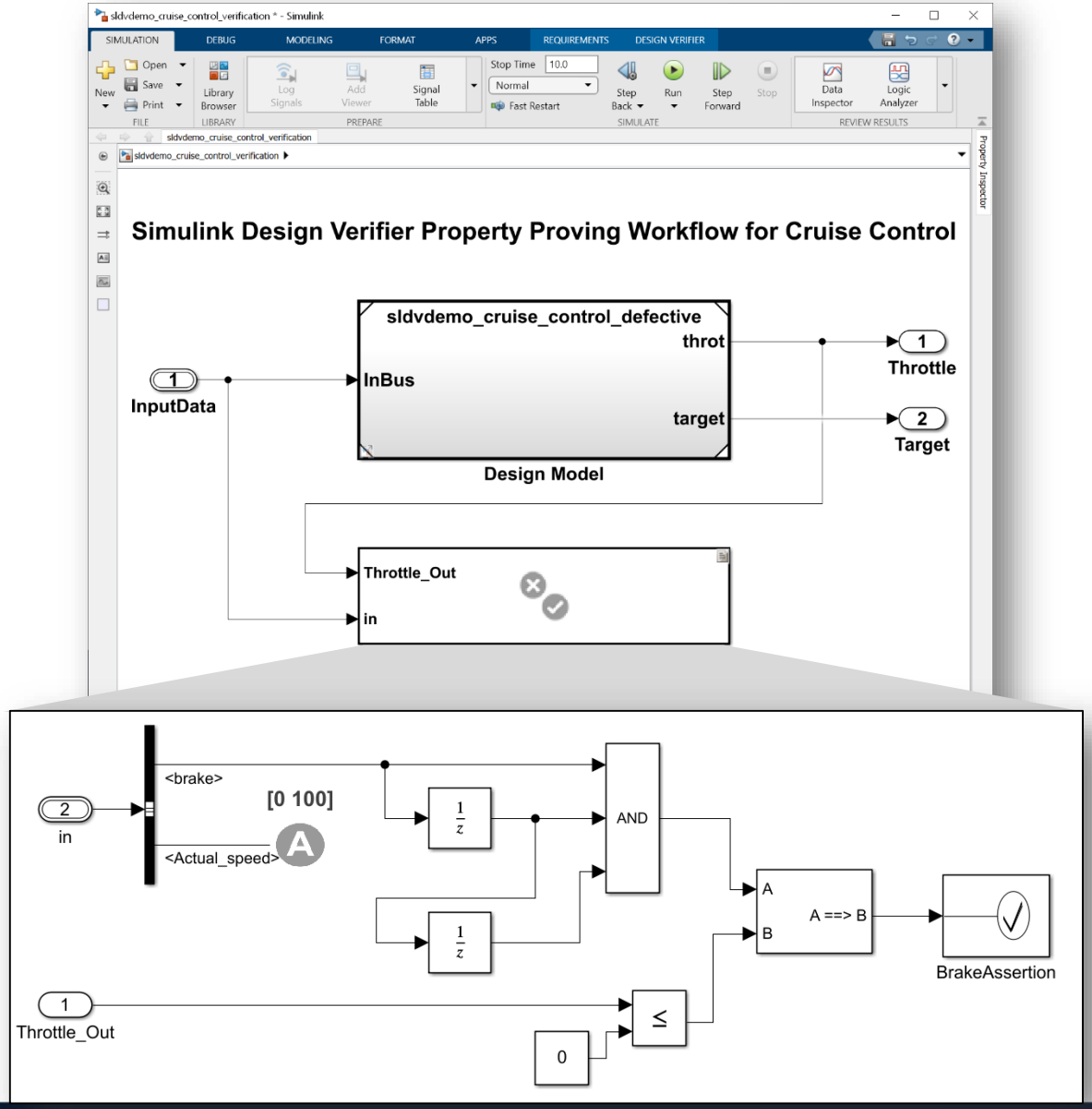
- Need to ensure the design performs correctly

Safety Requirement:

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

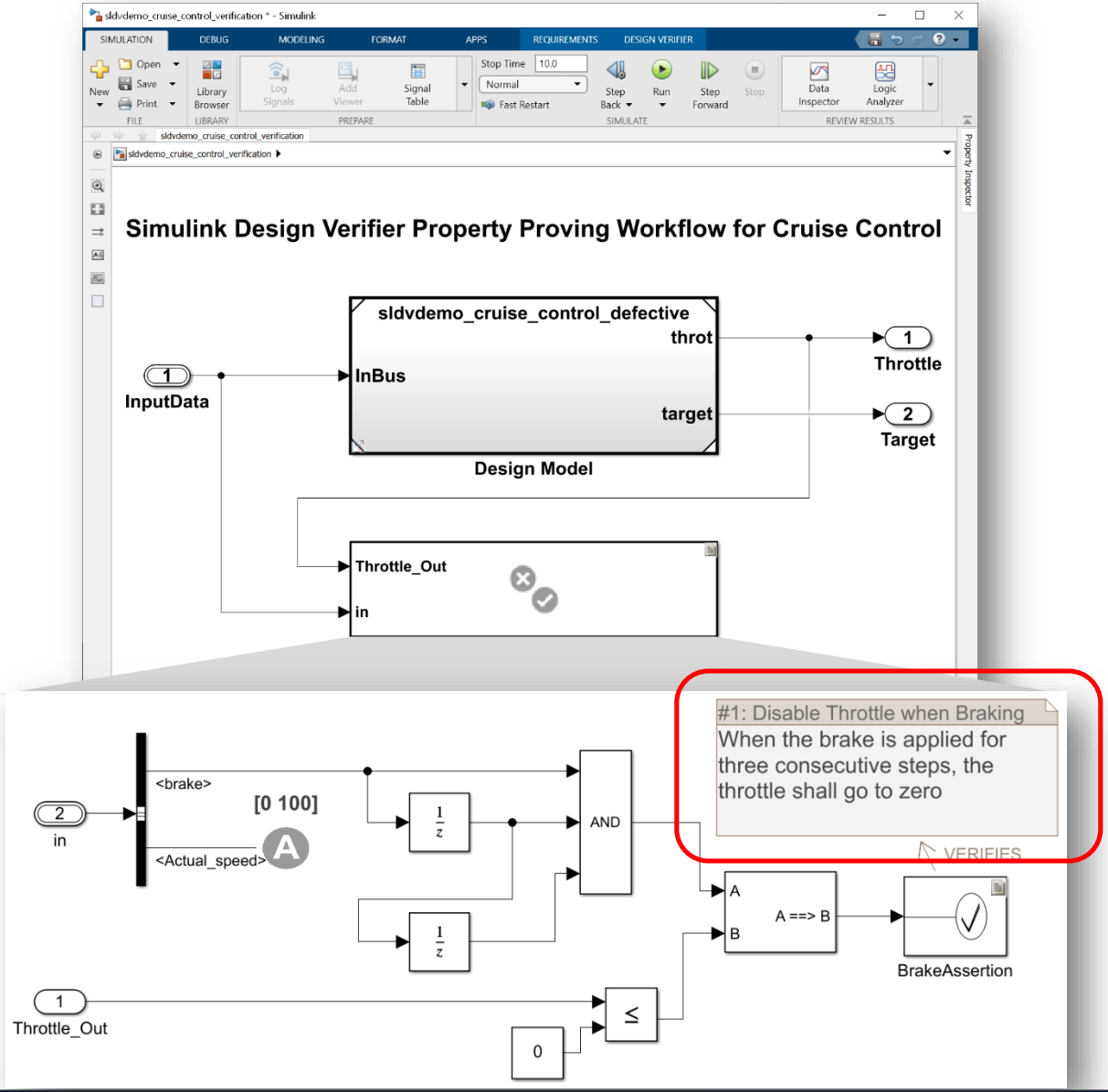


Model Functional and Safety Requirements

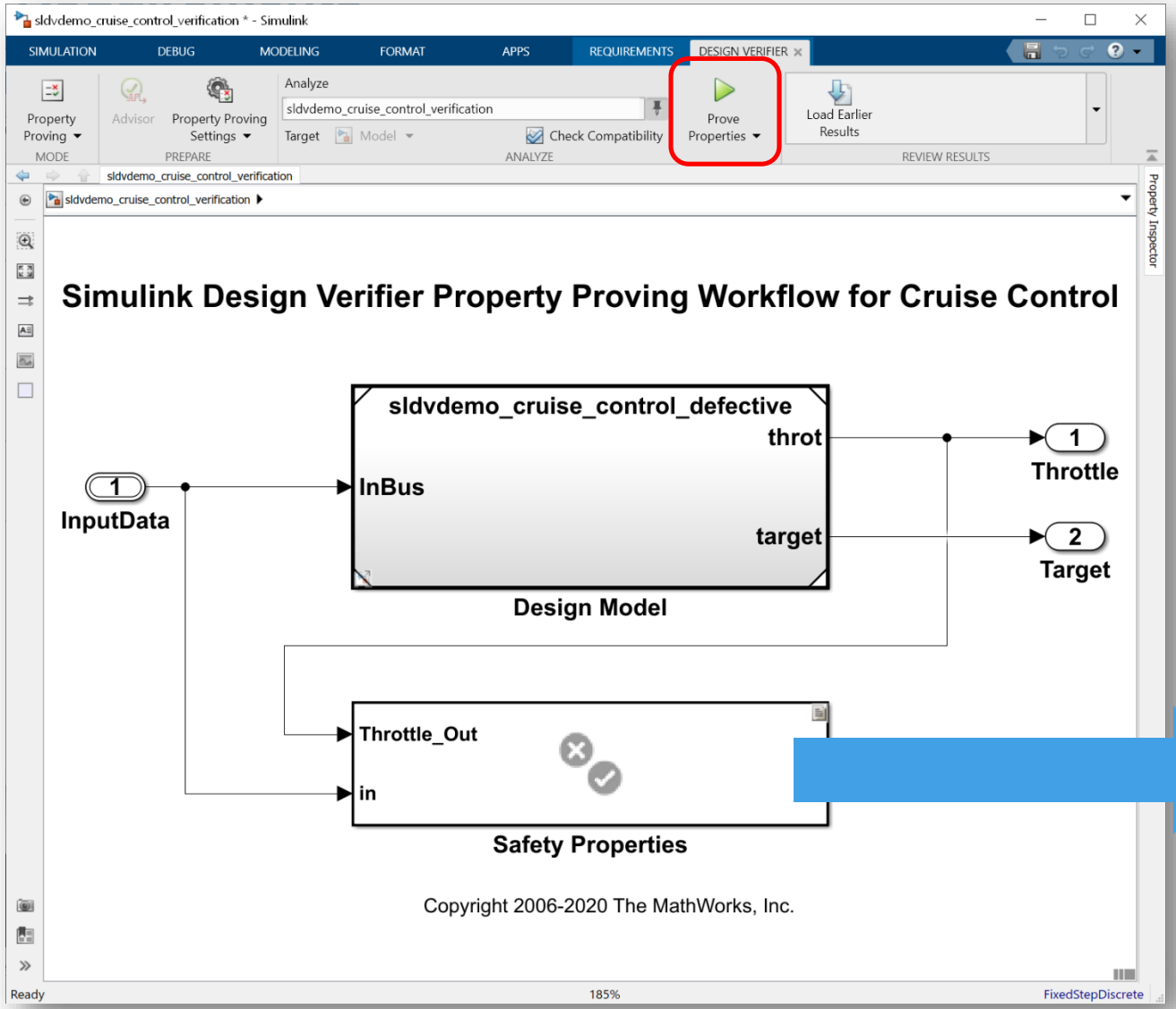


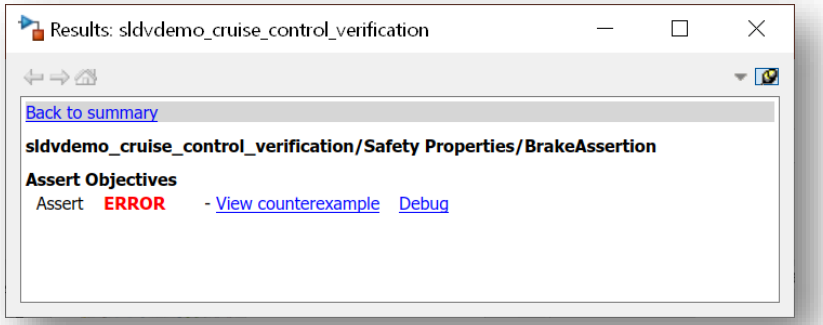
https://kr.mathworks.com/help/sldv/ug/property-proving-workflow-for-cruise-control.html

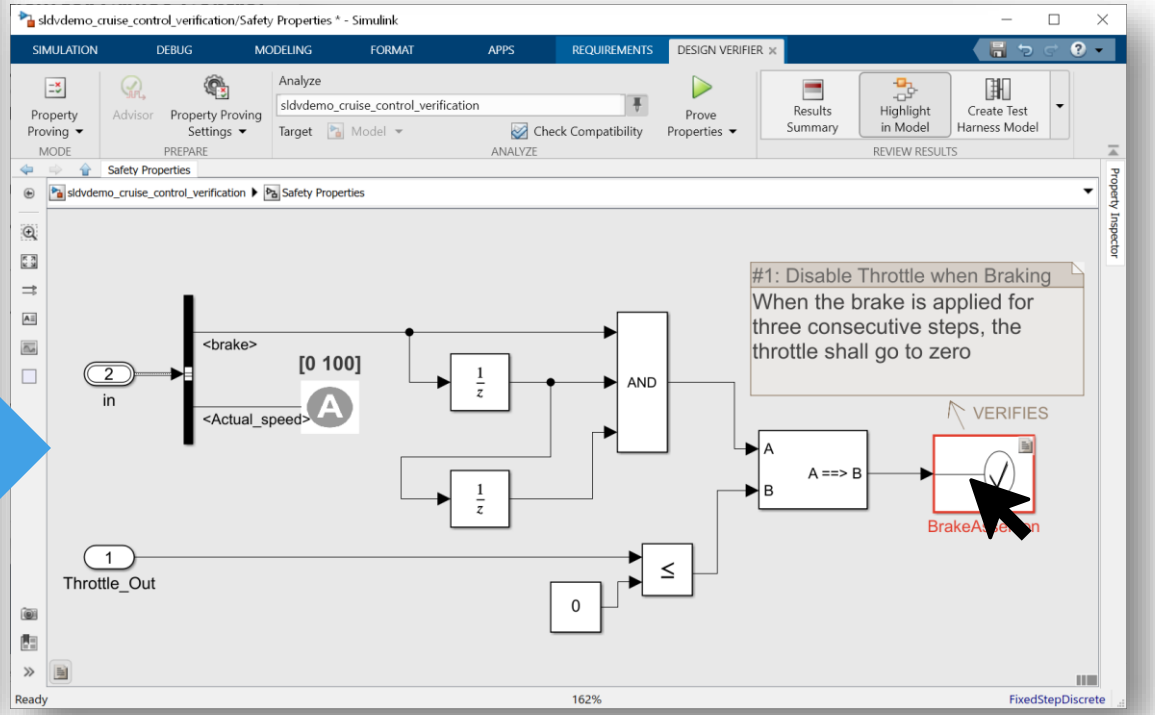
Link Requirements to Properties



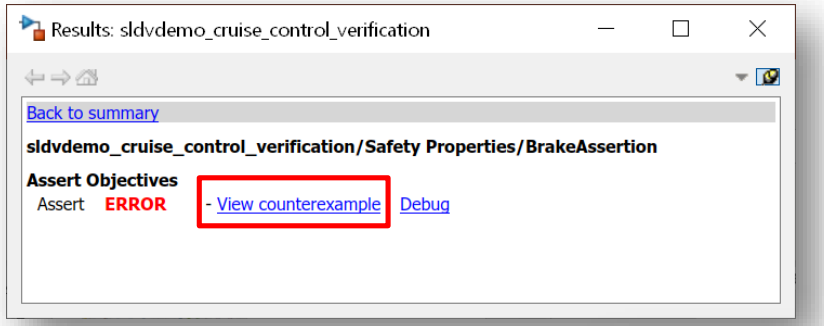
Prove That Design Meets Requirements

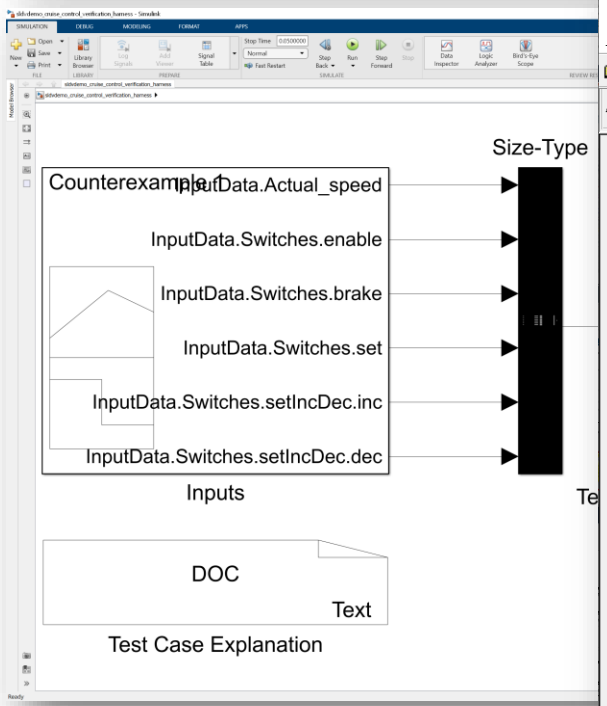


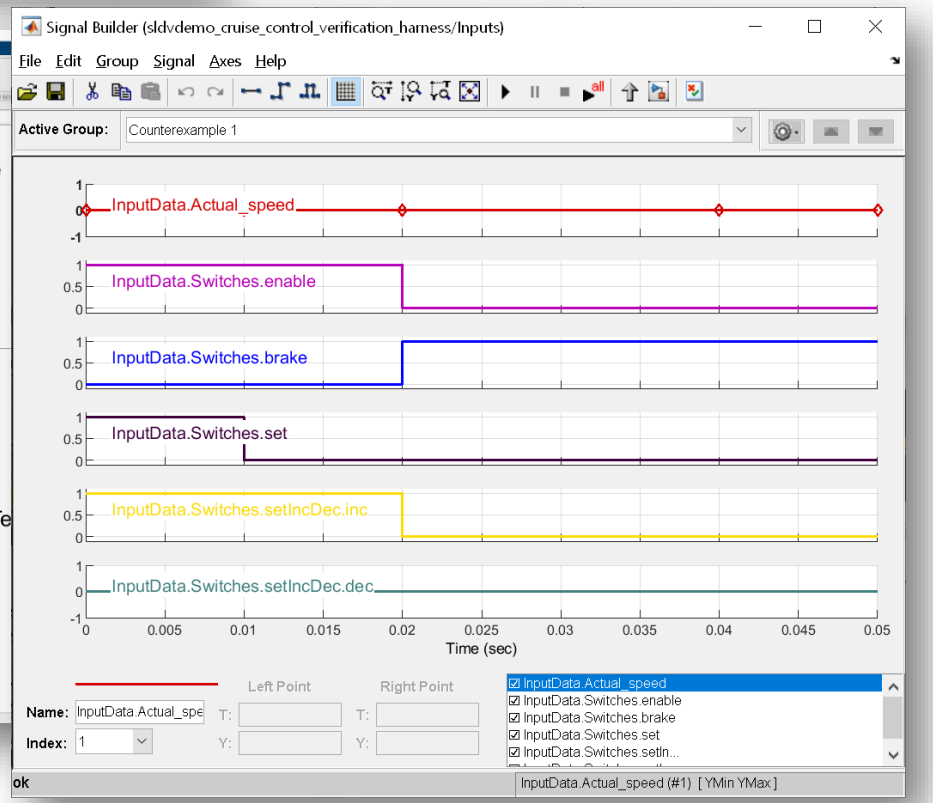


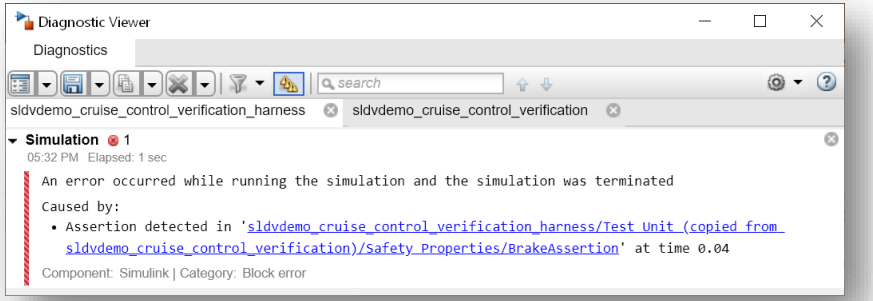


Prove That Design Meets Requirements

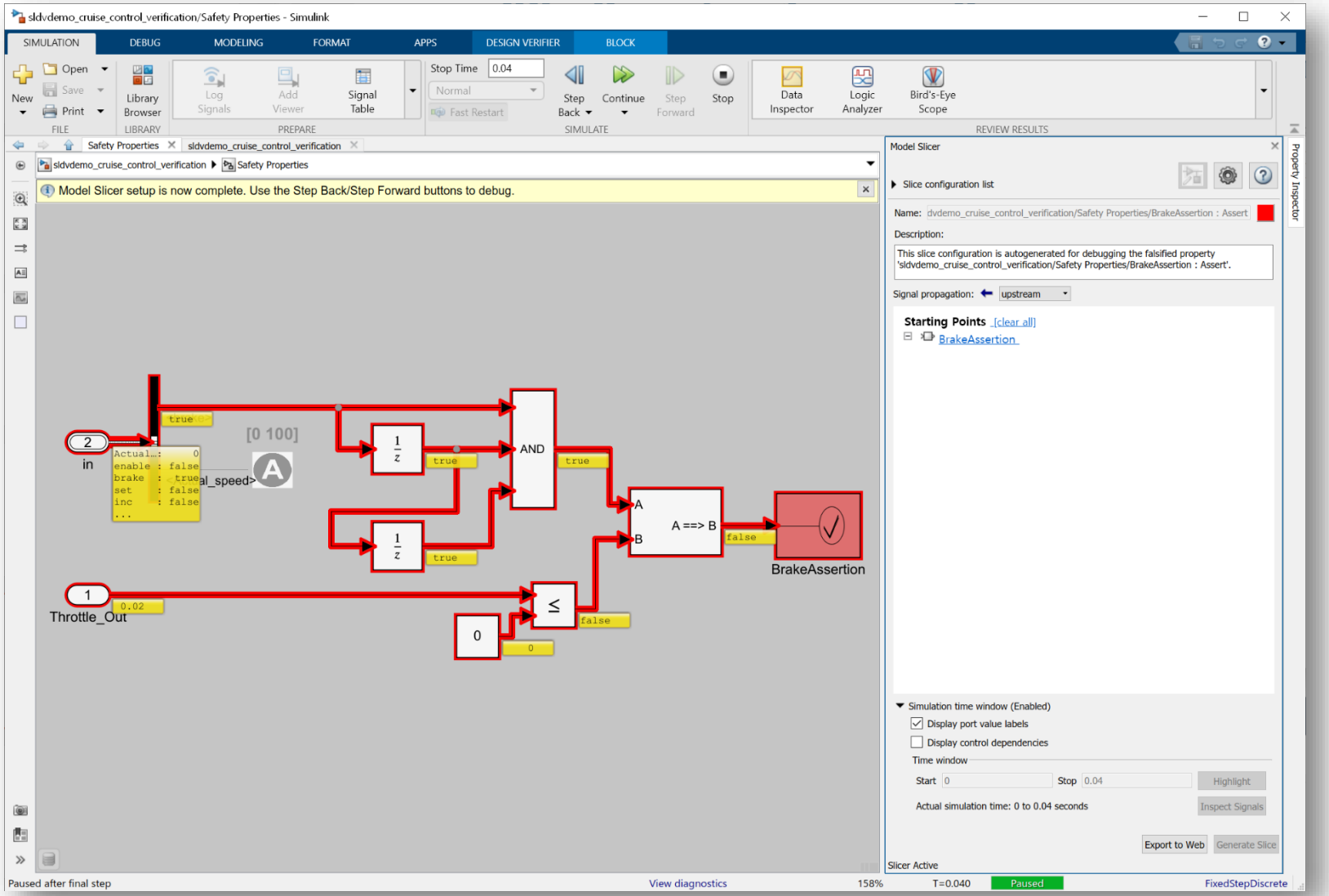


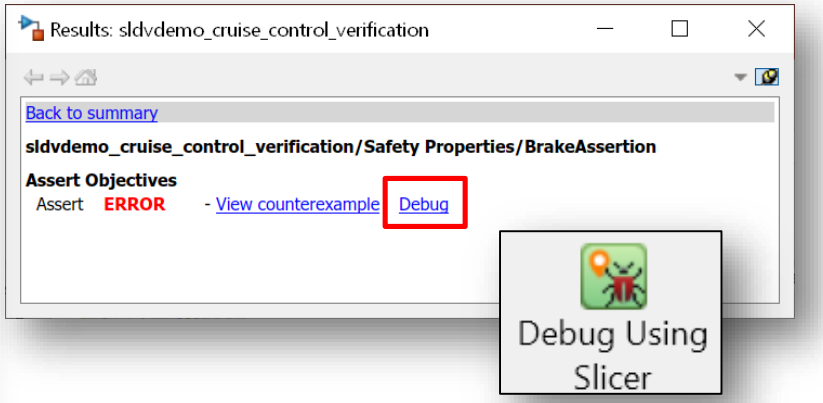






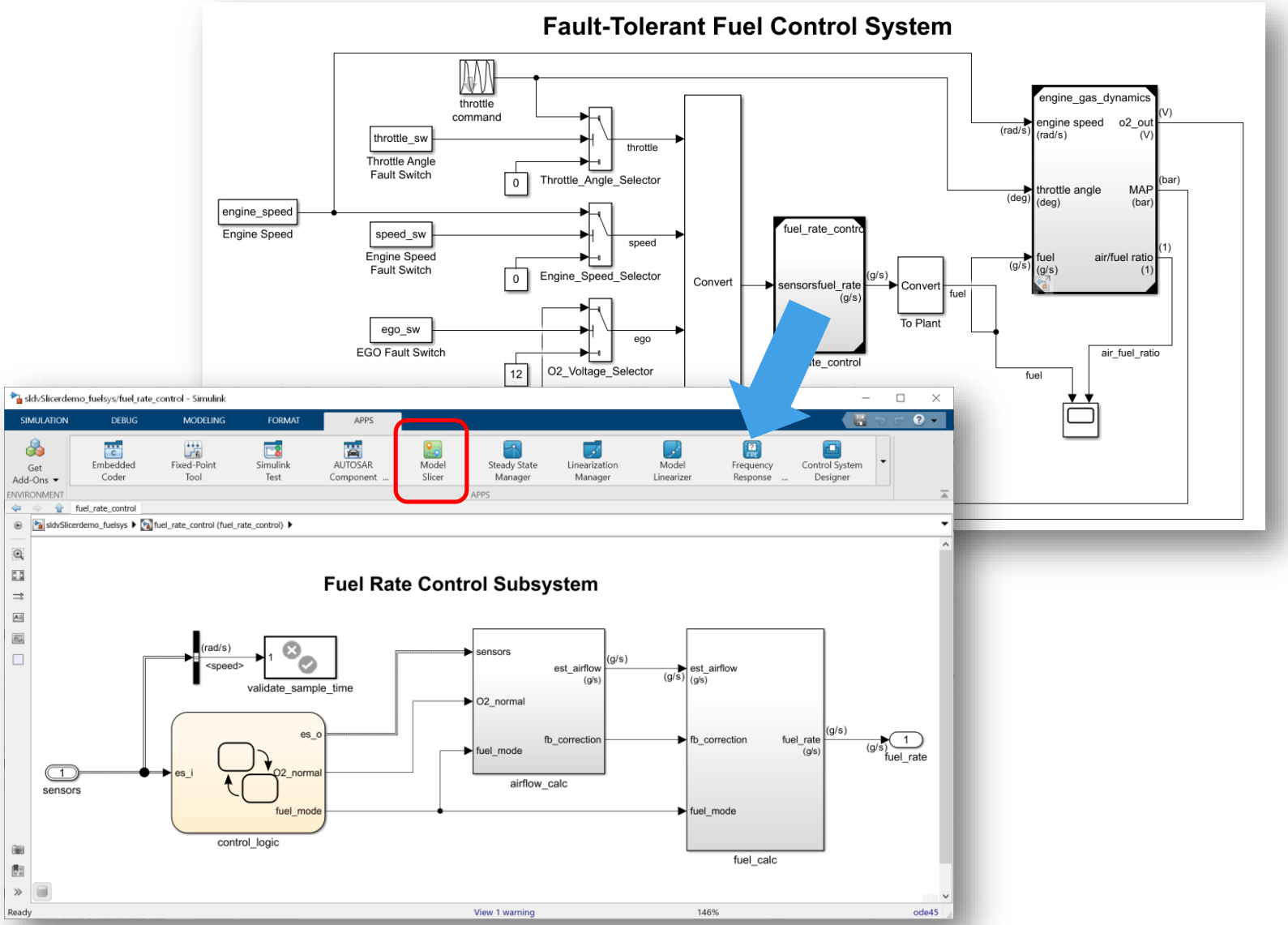
Debugging Property Proving Violations





Resolve Unexpected Behavior in a Model with Model Slicer

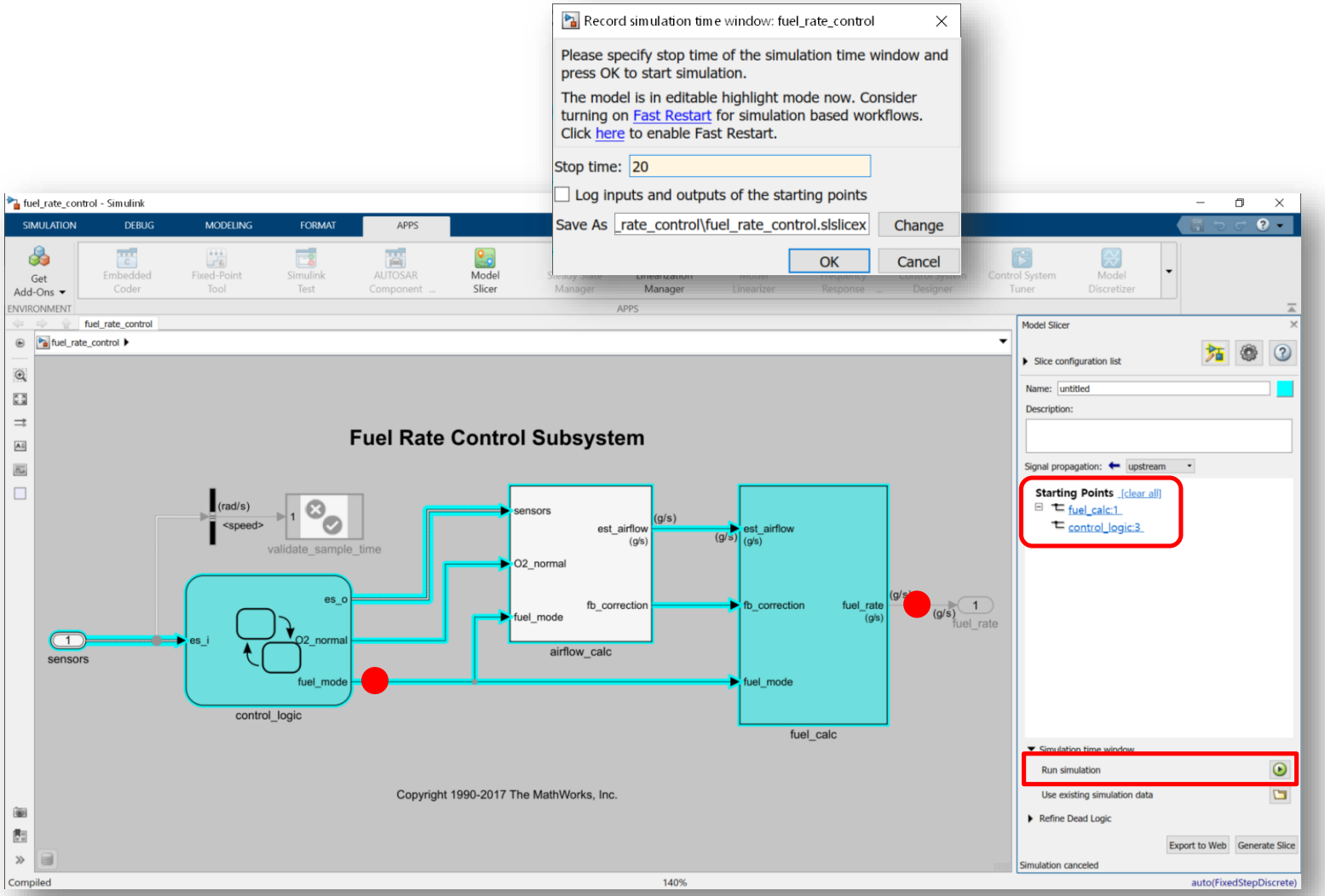
Isolate
Find the area of the model responsible for unexpected behavior



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

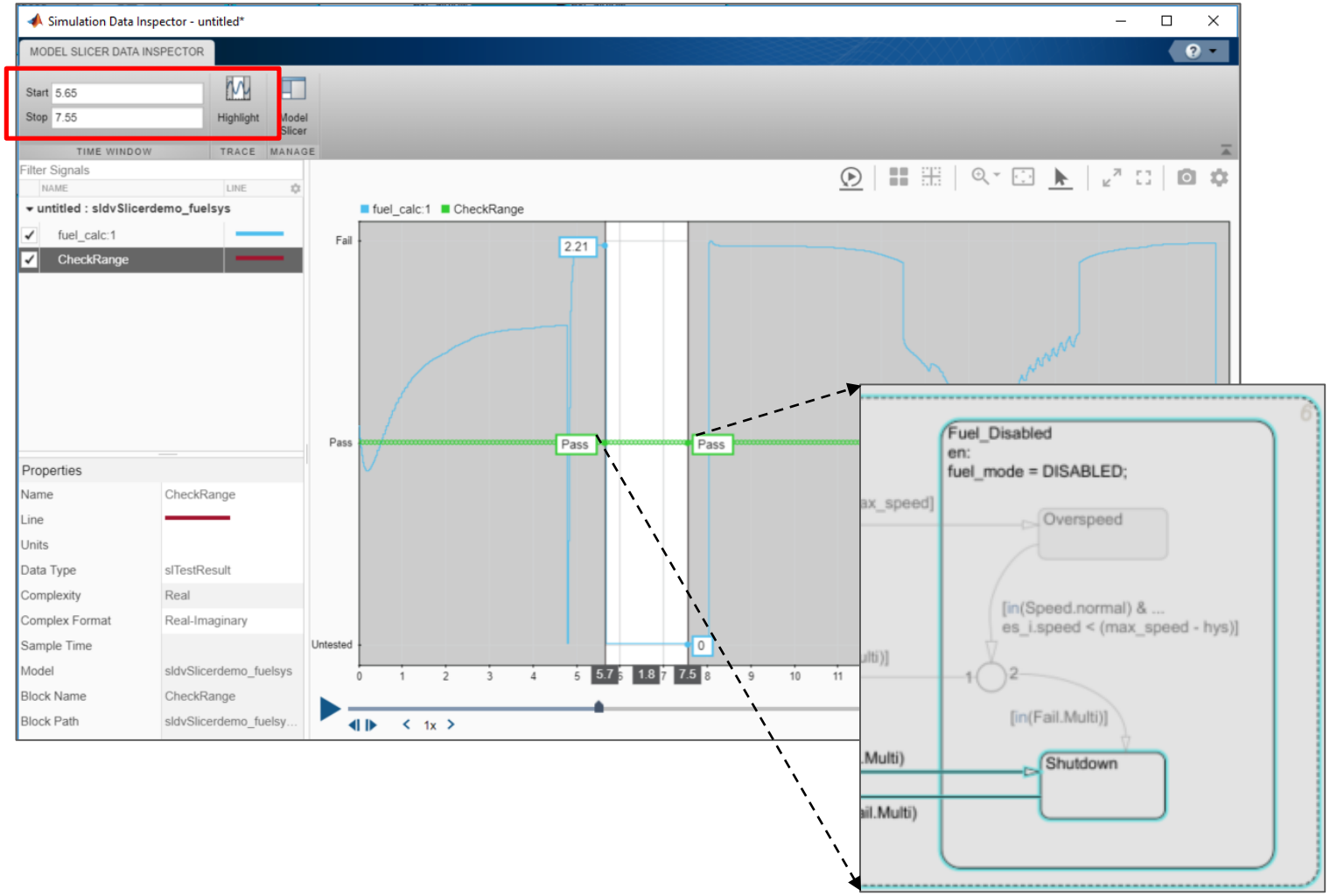


Resolve Unexpected Behavior in a Model with Model Slicer

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.



Resolve Unexpected Behavior in a Model with Model Slicer

Isolate
Find the area of the model responsible for unexpected behavior

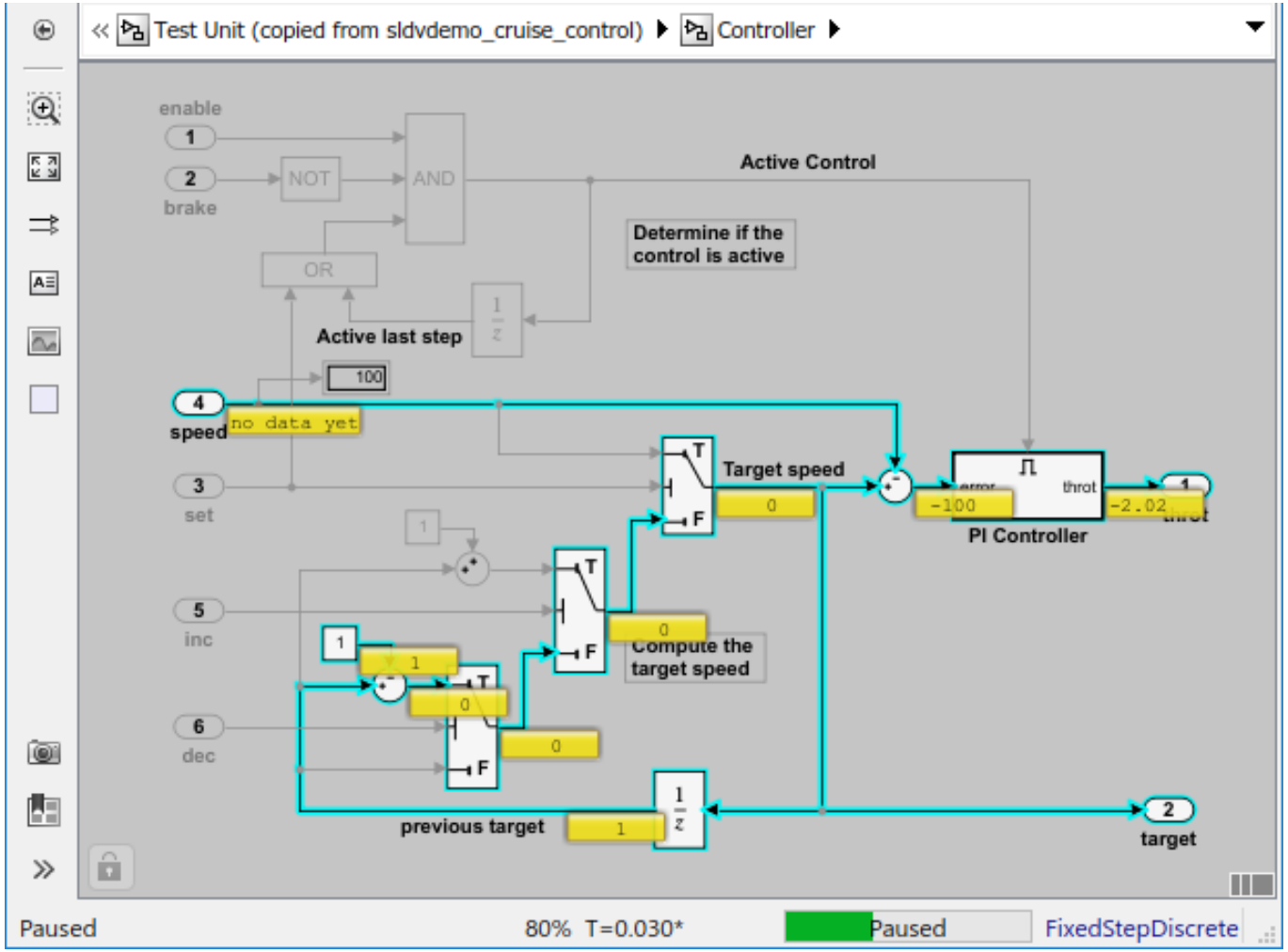
Analyze dependencies
Understand data & control dependencies in large or complex models

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

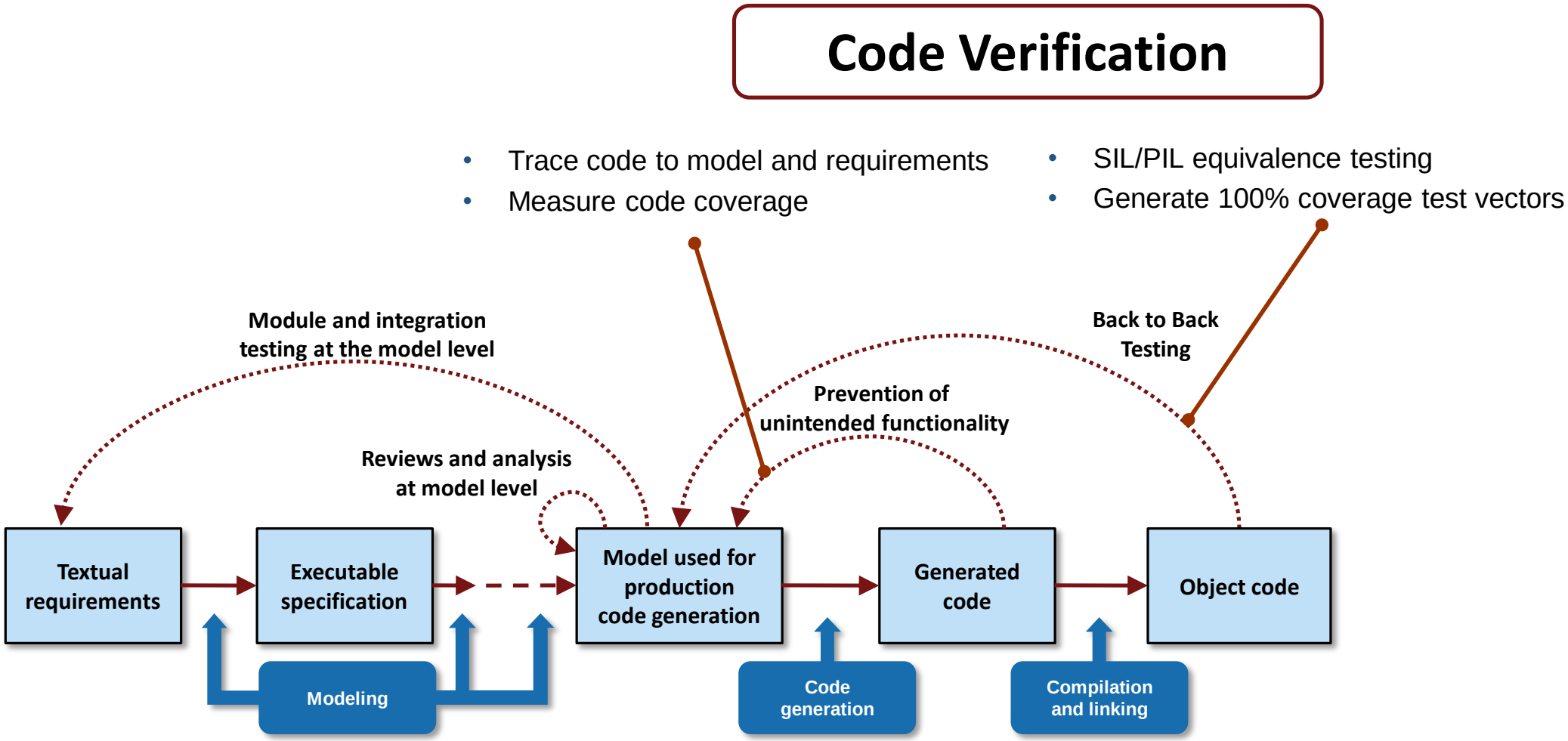
Debug simulation behavior
Step through precompiled slices to understand signal and port value propagation

Correct Model

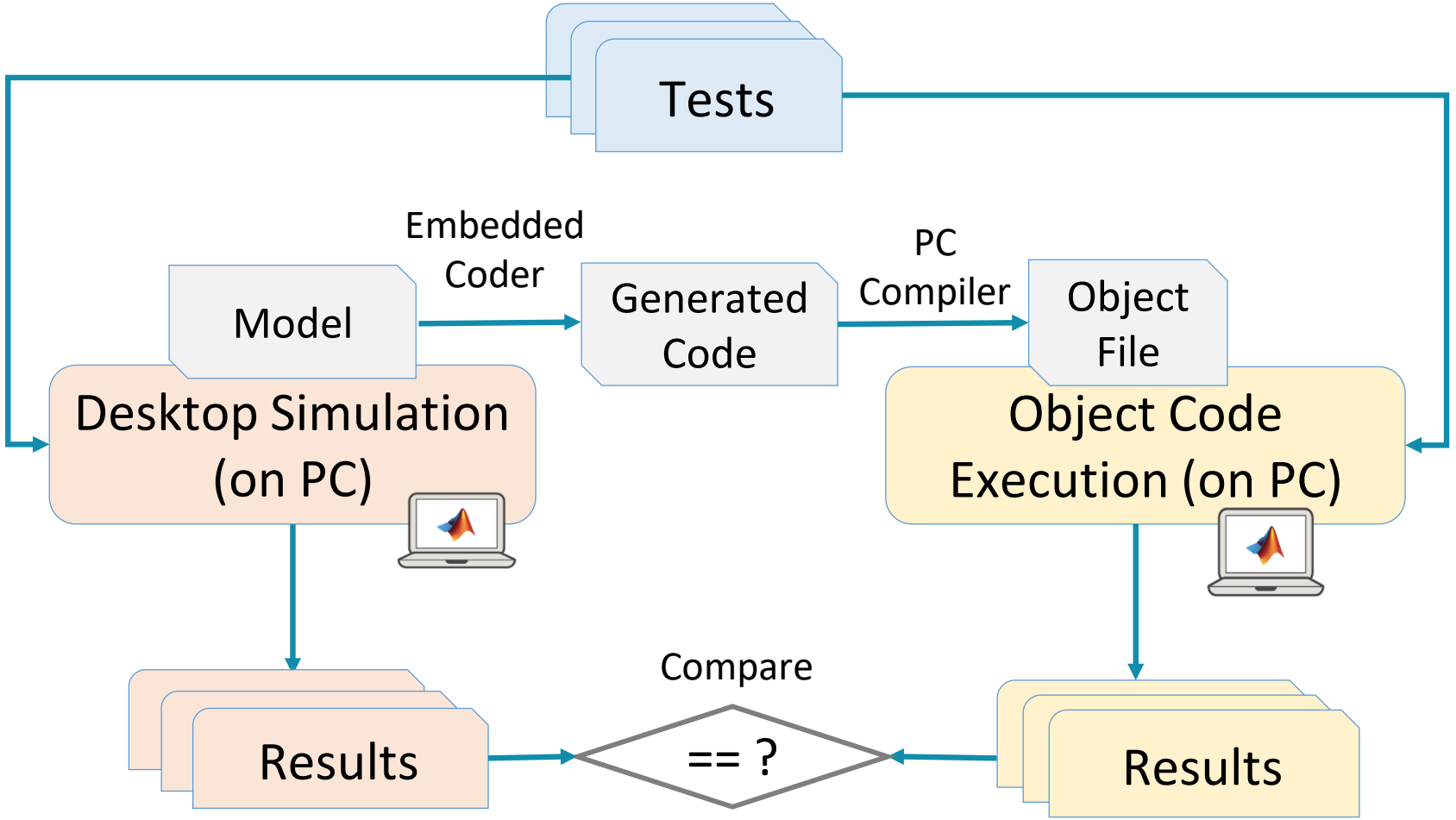
Iterate



Code Verification: Gain Confidence in the Generated Code

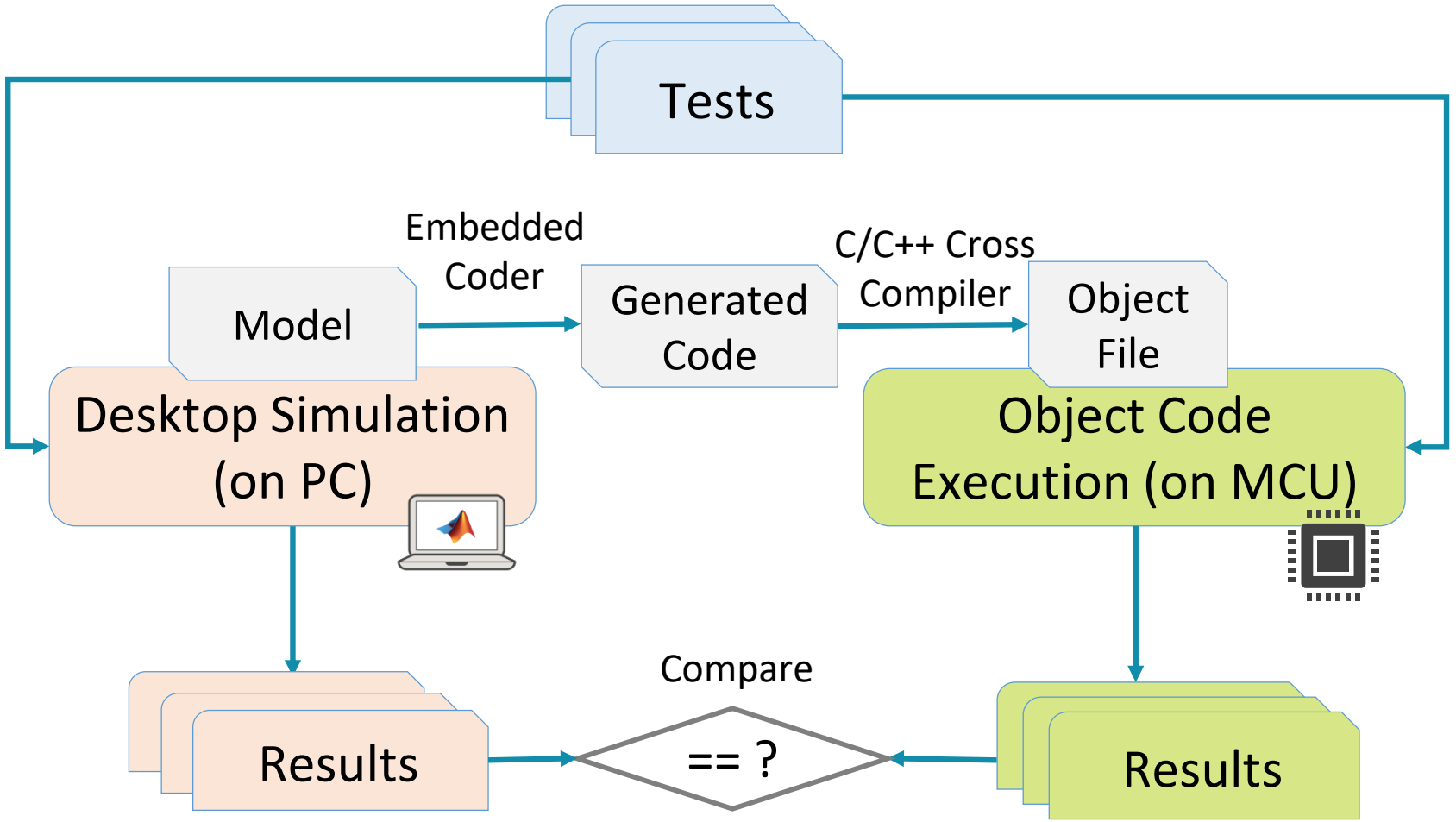


Back-to-Back Testing: Simulation-In-the-Loop



- Automate SIL testing using Simulink Test
- Testing across releases

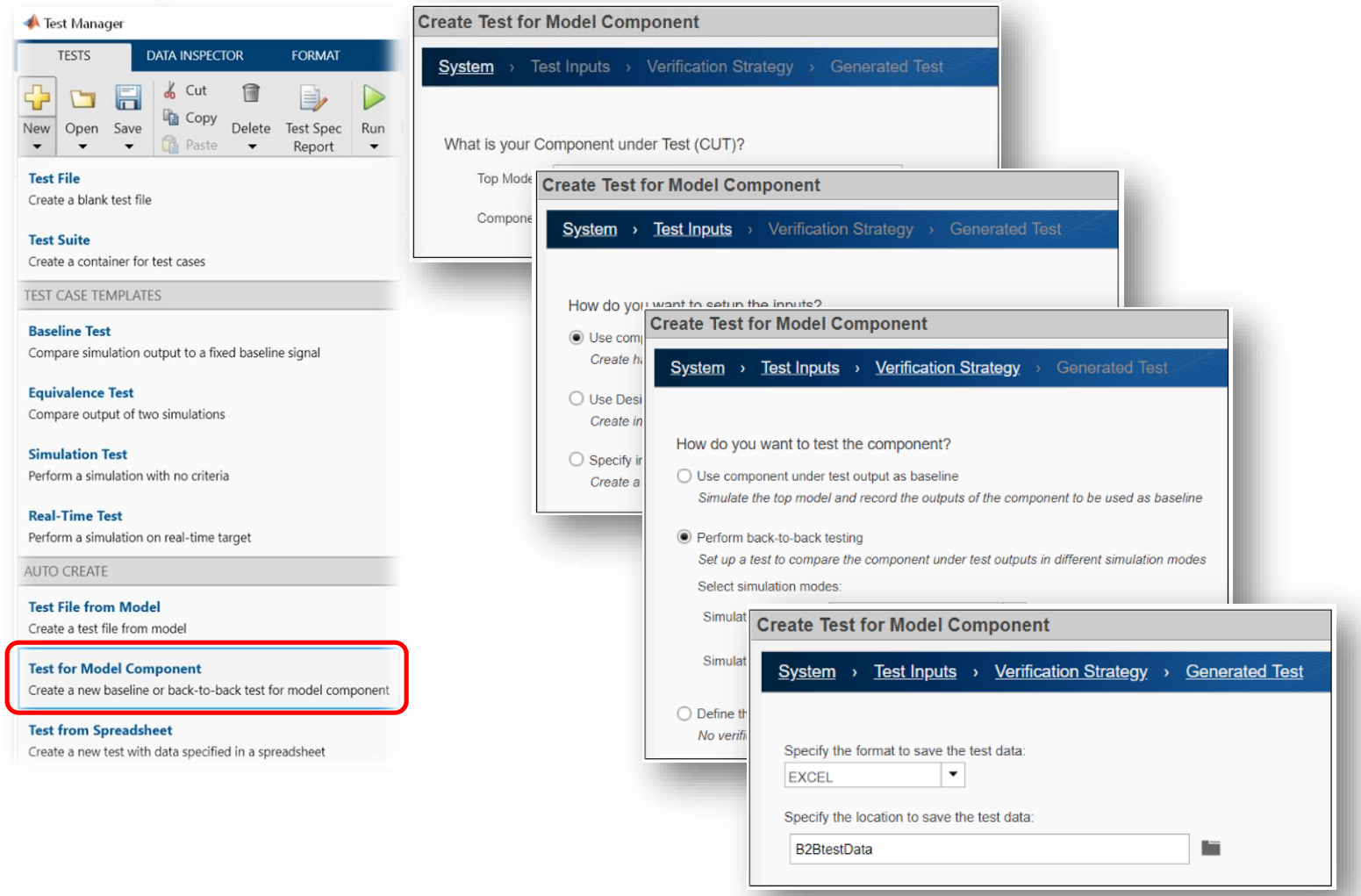
Back-to-Back Testing: Processor-In-the-Loop



- Automate PIL testing using Simulink Test
- Testing across releases
- Hardware support package

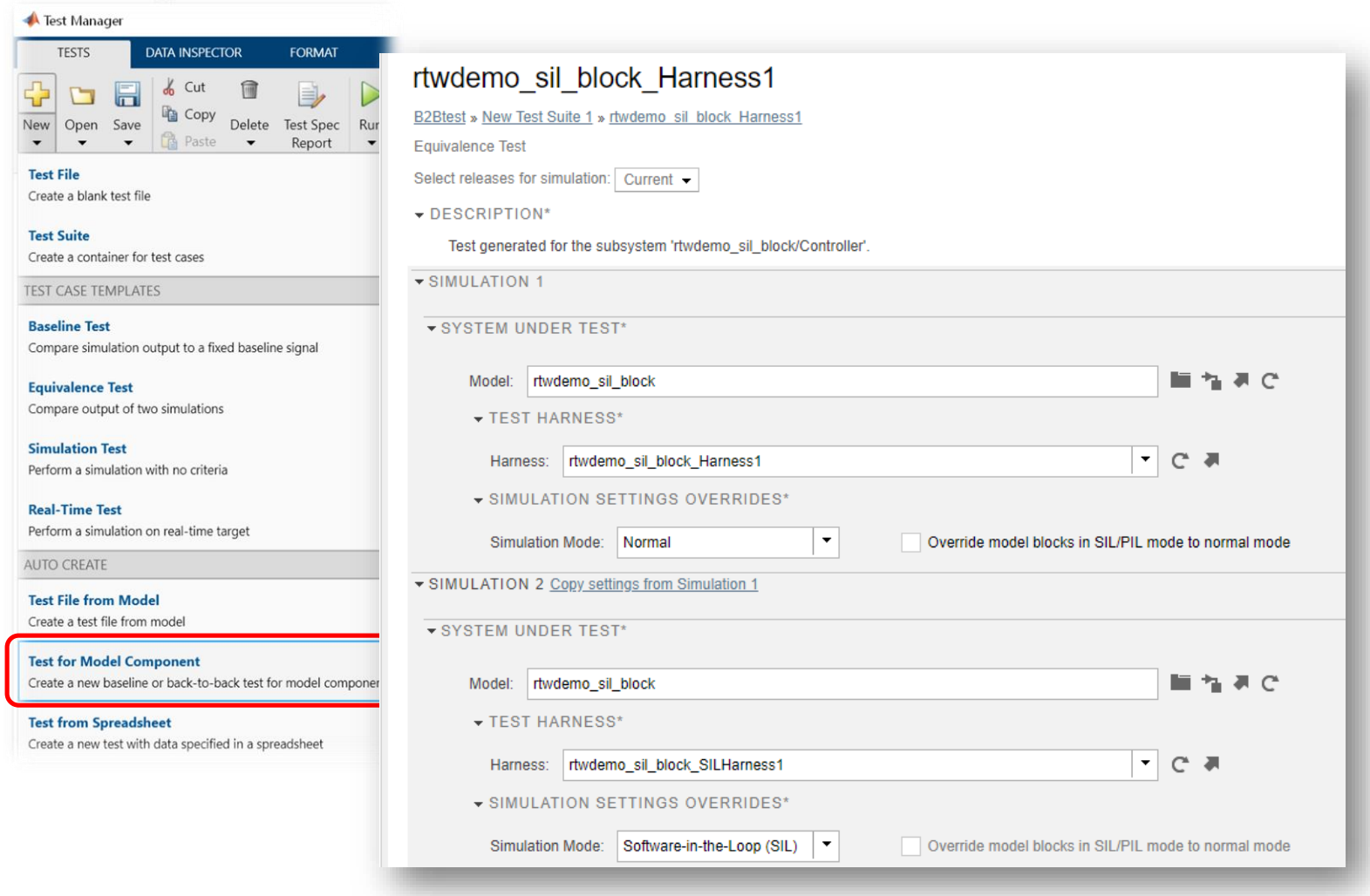
Automate Test Creation using Test Manager Wizard

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



Automate Test Creation using Test Manager Wizard

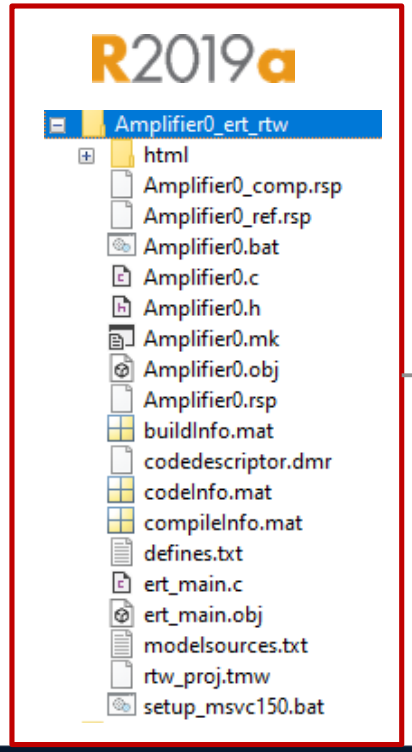
- Guided steps to define component to test, inputs, type of test and format for output
- Wizard generates required test harness
- Auto generate tests using Simulink Design Verifier

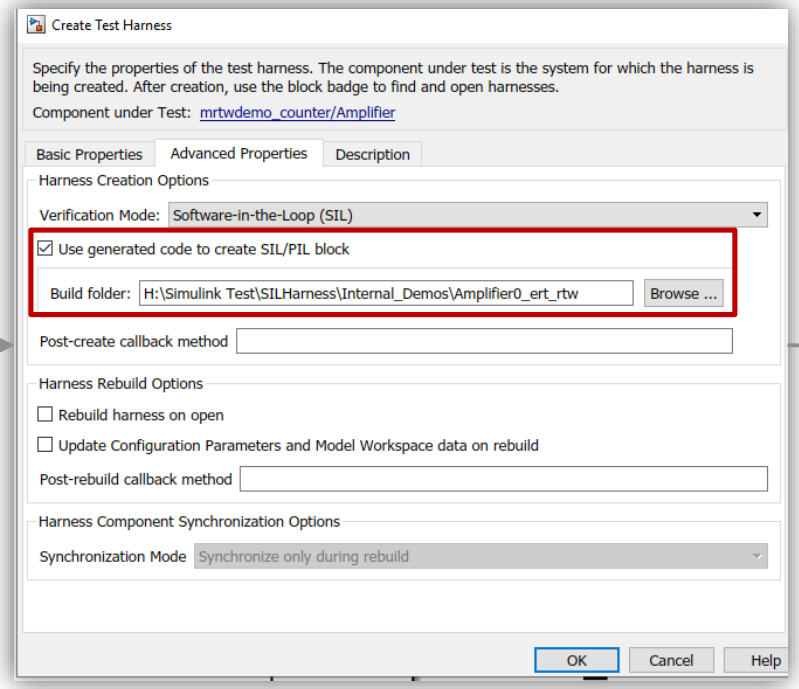


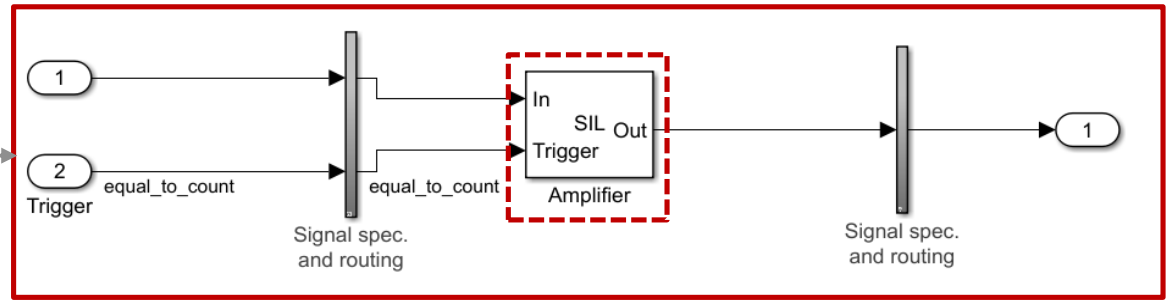
Cross Release SIL/PIL Test Harness Generation

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

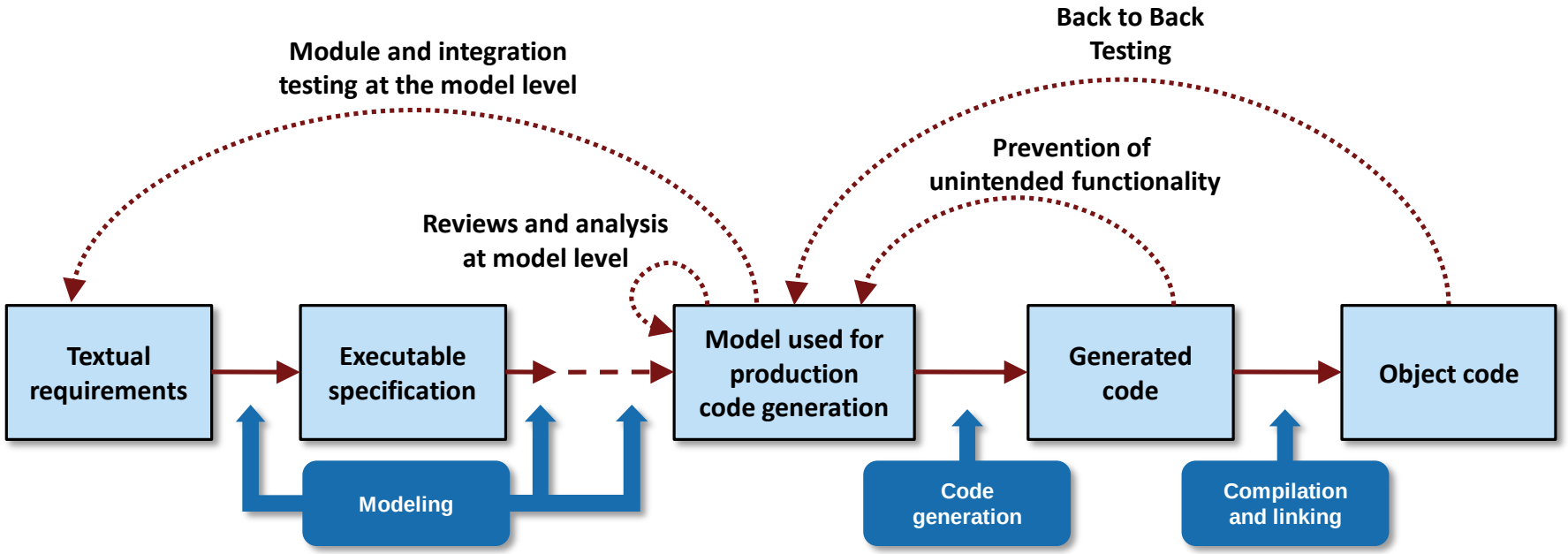






Use Reference Workflow to Conform to Standards

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



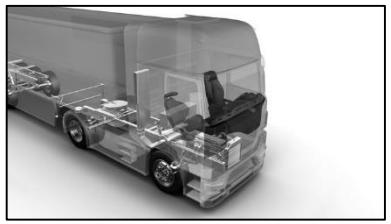
Customer References and Applications

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

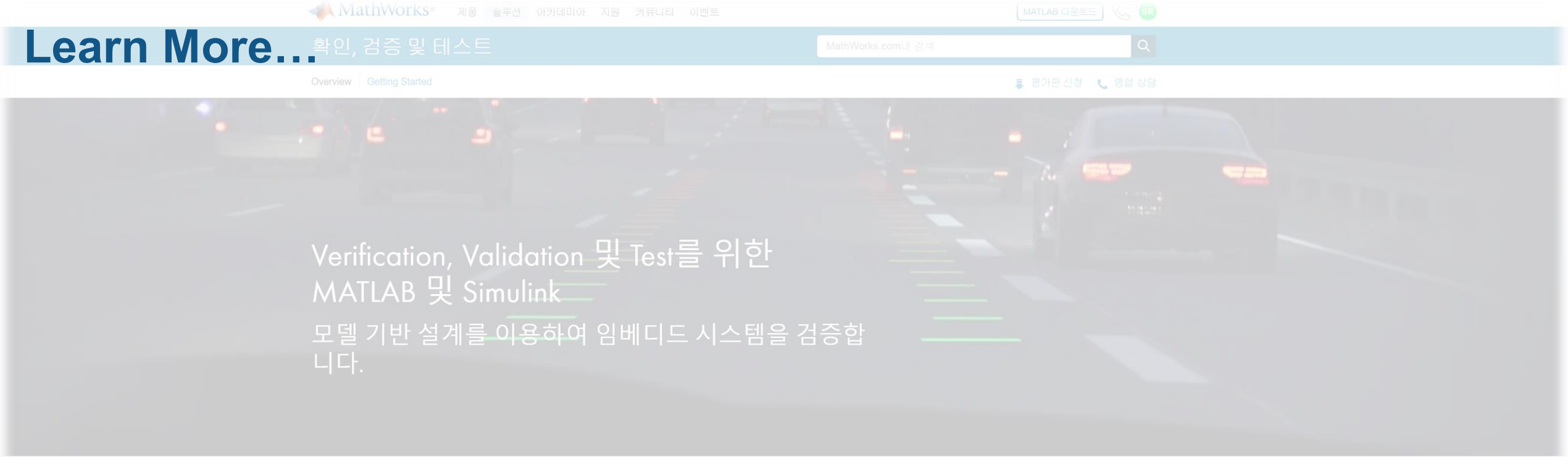


Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks

Verification time cut by up to 50 percent

More User Stories: https://kr.mathworks.com/company/user_stories.html

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 !!