MATLAB EXPO

Meet Certification Standards with Automated Requirements Based Testing

Paul Urban
Verification and Validation Product Manager

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 as Suitable for Software Architecture, Design and as basis for Code Generation

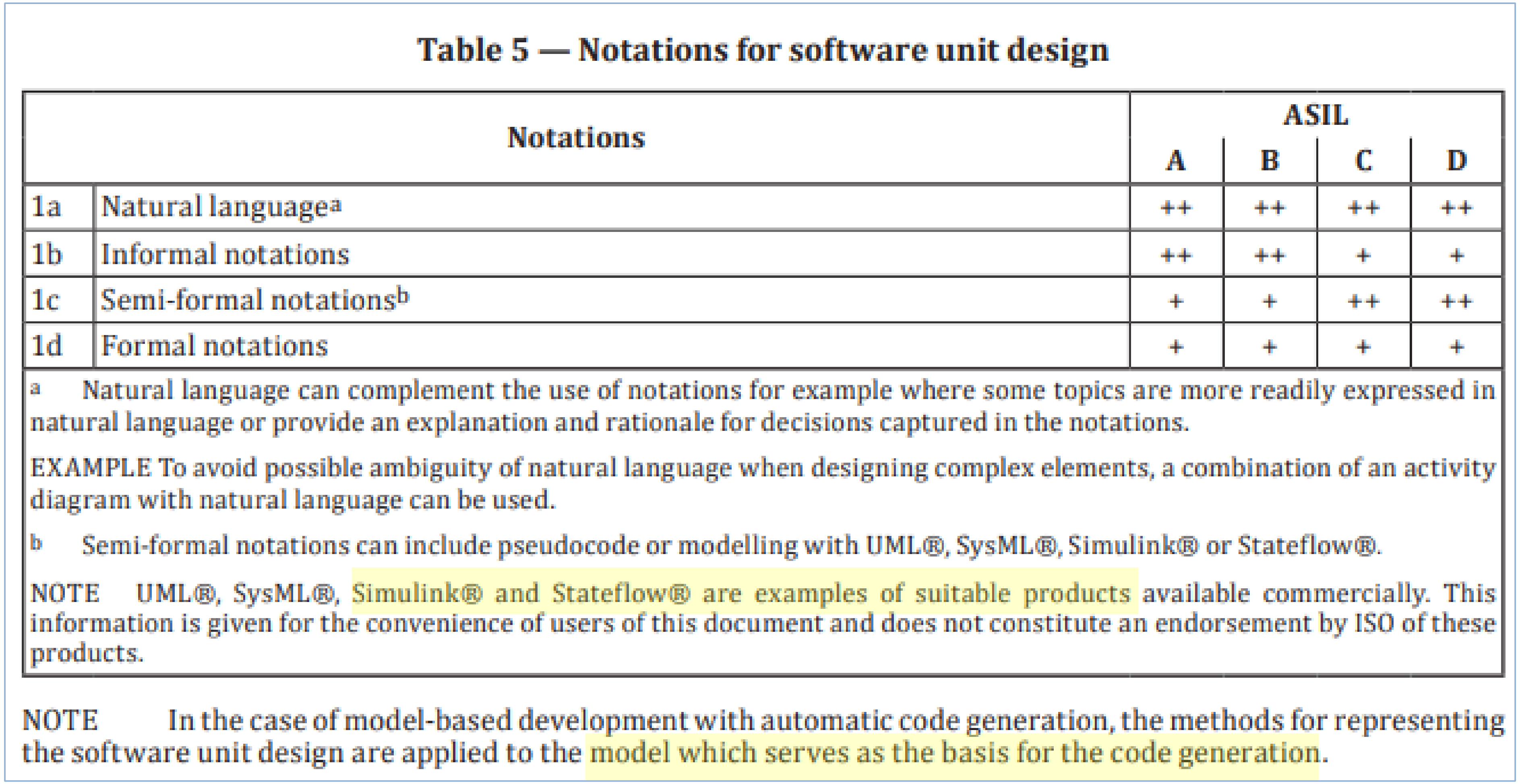


Table 5 — Notations for software unit design

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.

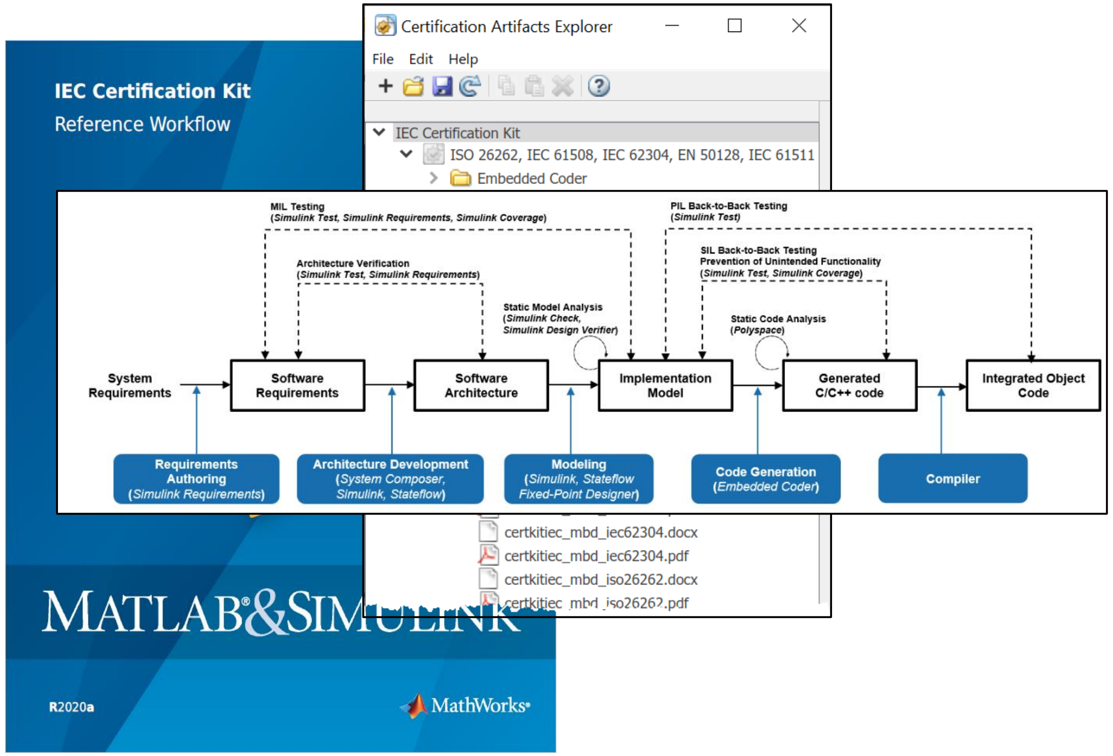
NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design are applied to the model which serves as the basis for the code generation.

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



IEC Certification Kit
Reference Workflow
MATLAB&SIMULINK
R2020a
MathWorks
Certification Artifacts Explorer
File
Edit
Help
IEC Certification Kit
ISO 26262, IEC 61508, IEC 62304, EN 50128, IEC 61511
Embedded Coder
certkitiec_mbd_iec62304.docx
certkitiec_mbd_iec62304.pdf
certkitiec_mbd_iso26262.docx
certkitiec_mbd_iso26262.pdf

MIL Testing (*Simulink Test, Simulink Requirements, Simulink Coverage*)

Architecture Verification (*Simulink Test, Simulink Requirements*)

Static Model Analysis (*Simulink Check, Simulink Design Verifier*)

PIL Back-to-Back Testing (*Simulink Test*)

SIL Back-to-Back Testing Prevention of Unintended Functionality (*Simulink Test, Simulink Coverage*)

Static Code Analysis (*Polyspace*)

System Requirements → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ code → Integrated Object Code

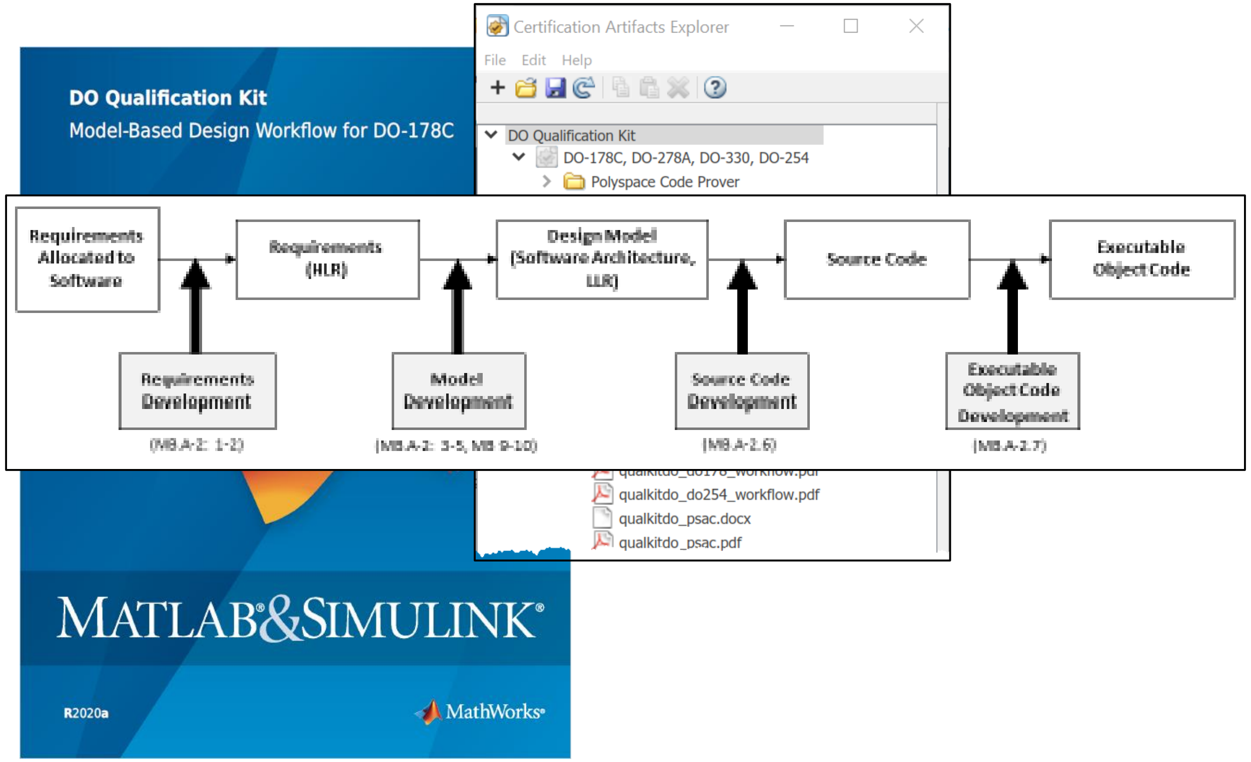
Requirements Authoring (*Simulink Requirements*)

Architecture Development (*System Composer, Simulink, Stateflow*)

Modeling (*Simulink, Stateflow Fixed-Point Designer*)

Code Generation (*Embedded Coder*)

Compiler



DO Qualification Kit
Model-Based Design Workflow for DO-178C
MATLAB&SIMULINK
R2020a
MathWorks
Certification Artifacts Explorer
File
Edit
Help
DO Qualification Kit
DO-178C, DO-278A, DO-330, DO-254
Polyspace Code Prover
qualkitdo_do178_workflow.pdf
qualkitdo_do254_workflow.pdf
qualkitdo_psac.docx
qualkitdo_psac.pdf

Requirements Allocated to Software → Requirements (HLR) → Design Model (Software Architecture, LLR) → Source Code → Executable Object Code

Requirements Development (MB.A-2: 1-2)

Model Development (MB.A-2: 3-5, MB 9-10)

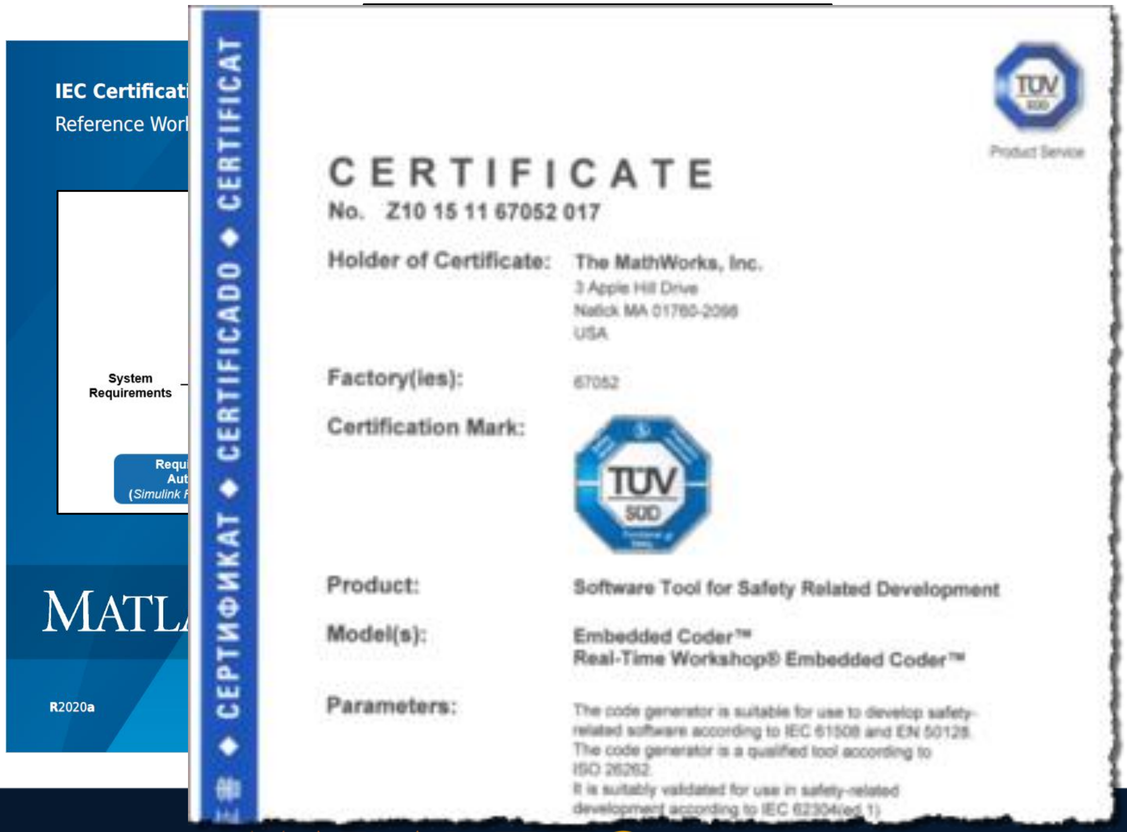
Source Code Development (MB.A-2.6)

Executable Object Code Development (MB.A-2.7)

Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products

- Includes documentation, test cases and procedures



IEC Certificati
Reference Worl
System Requirements
Requi
Aut
(Simulink F
MATL
R2020a
ZERTIFIKAT ◆ CERTIFICATE ◆ CERTIFICADO ◆ CERTIFICAT ◆ СЕРТИФИКАТ ◆ 證書
TÜV SÜD
Product Service

CERTIFICATE

No. Z10 15 11 67052 017

Holder of Certificate: The MathWorks, Inc.
3 Apple Hill Drive
Natick MA 01760-2098
USA

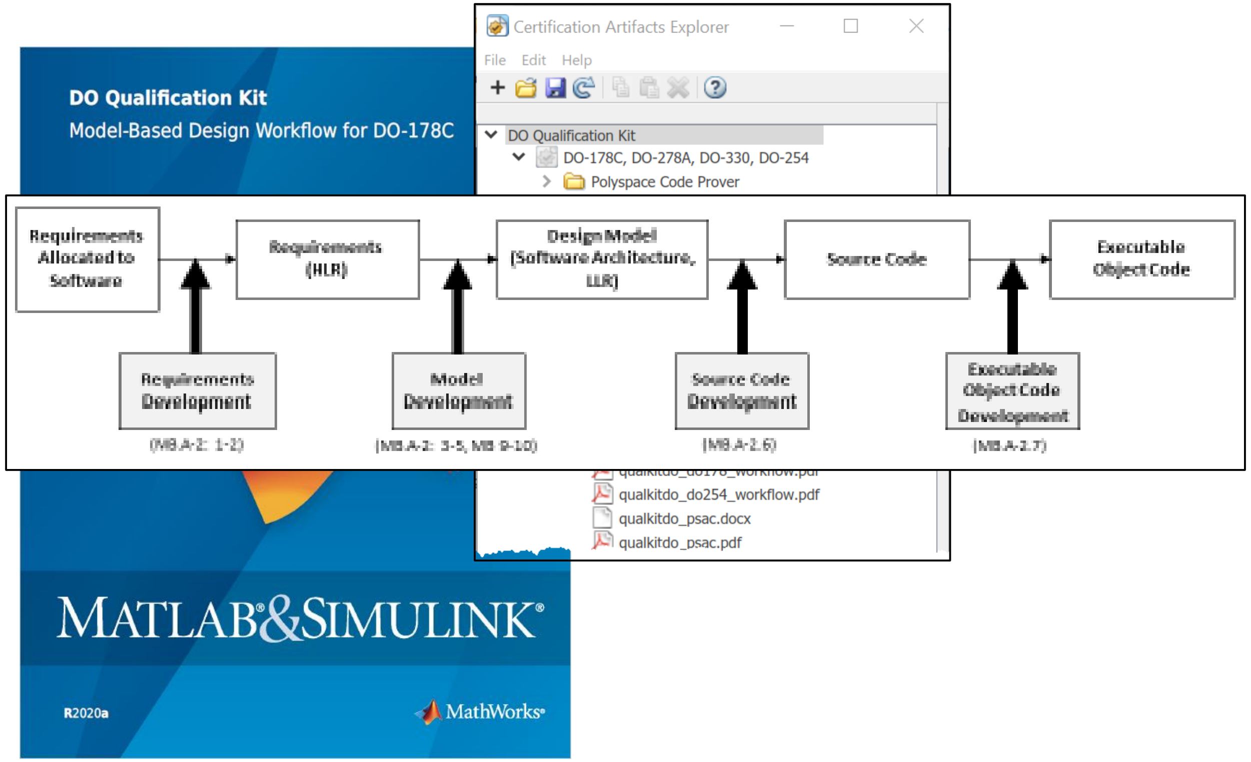
Factory(ies): 67052

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): Embedded Coder™
Real-Time Workshop® Embedded Coder™

Parameters: The code generator is suitable for use to develop safety-related software according to IEC 61508 and EN 50128.
The code generator is a qualified tool according to ISO 26262.
It is suitably validated for use in safety-related development according to IEC 62304(ed.1).

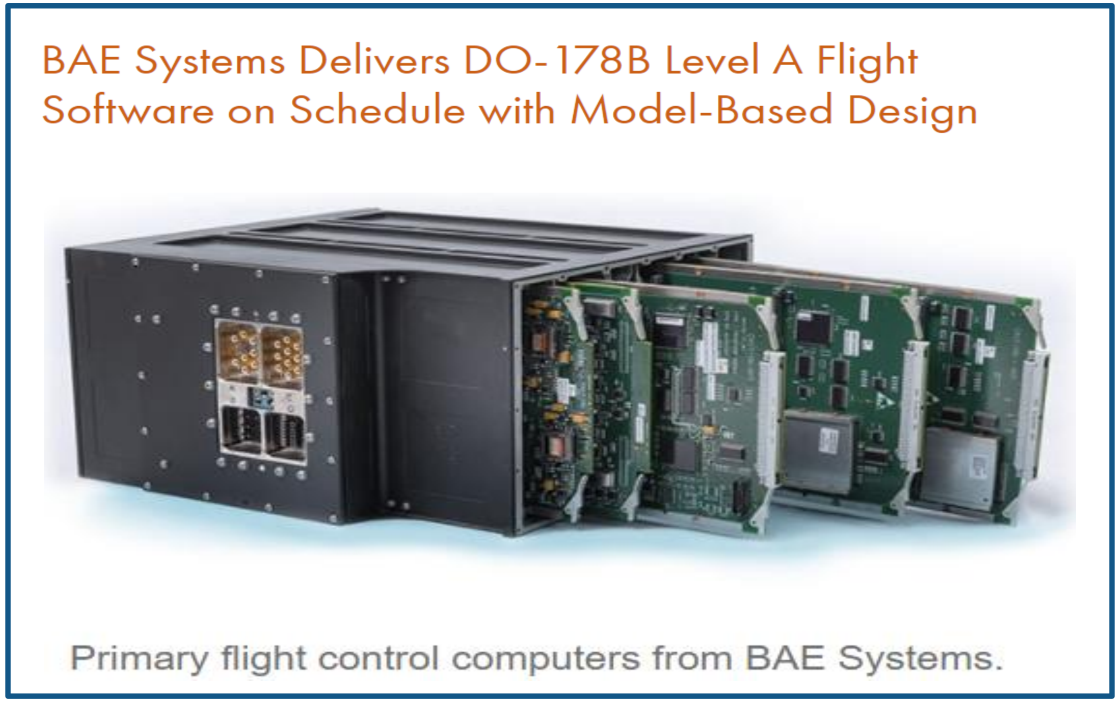


Qualify tools with IEC Certification Kit and DO Qualification Kit

- 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.



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

Primary flight control computers from BAE Systems.

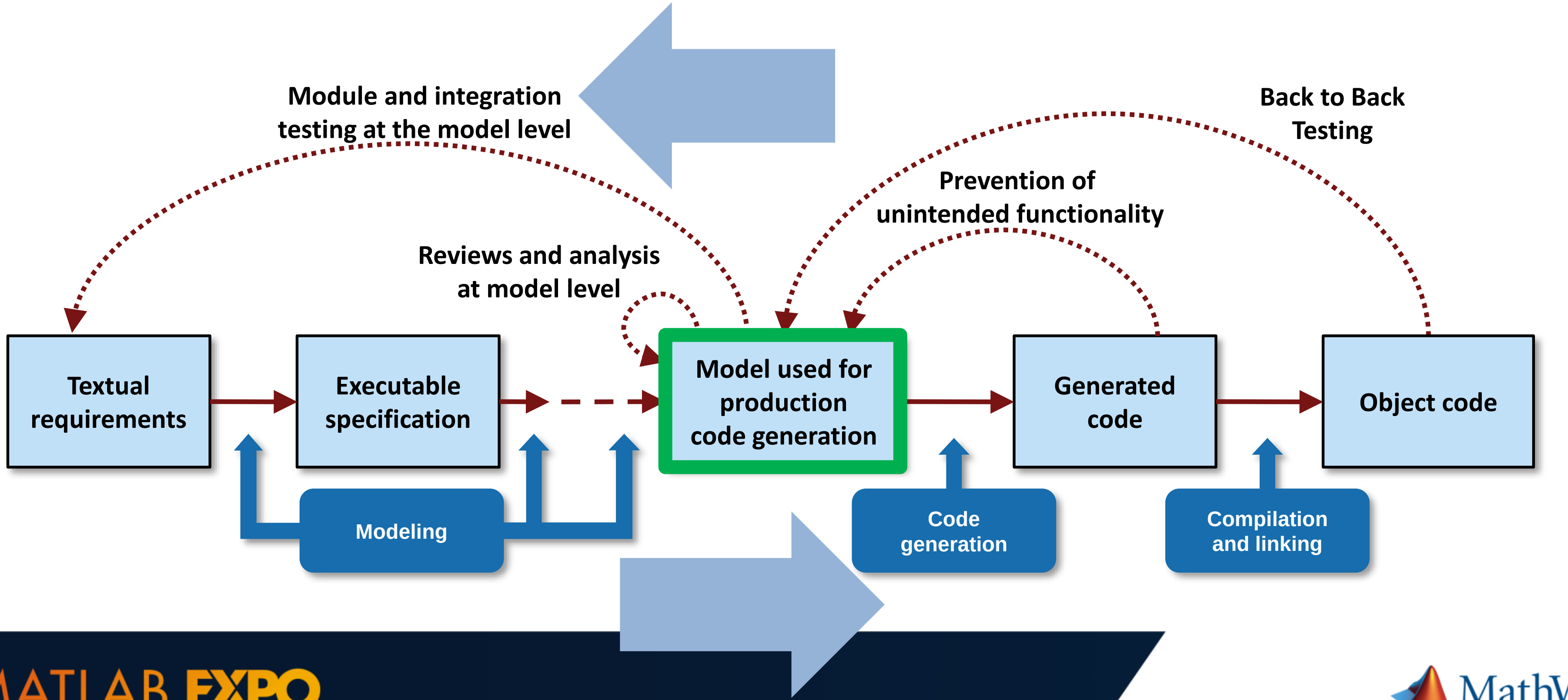
Conform to Certification Standards with Reference Workflow

Model Verification

Discover design errors at design time

Code Verification

Gain confidence in the generated code



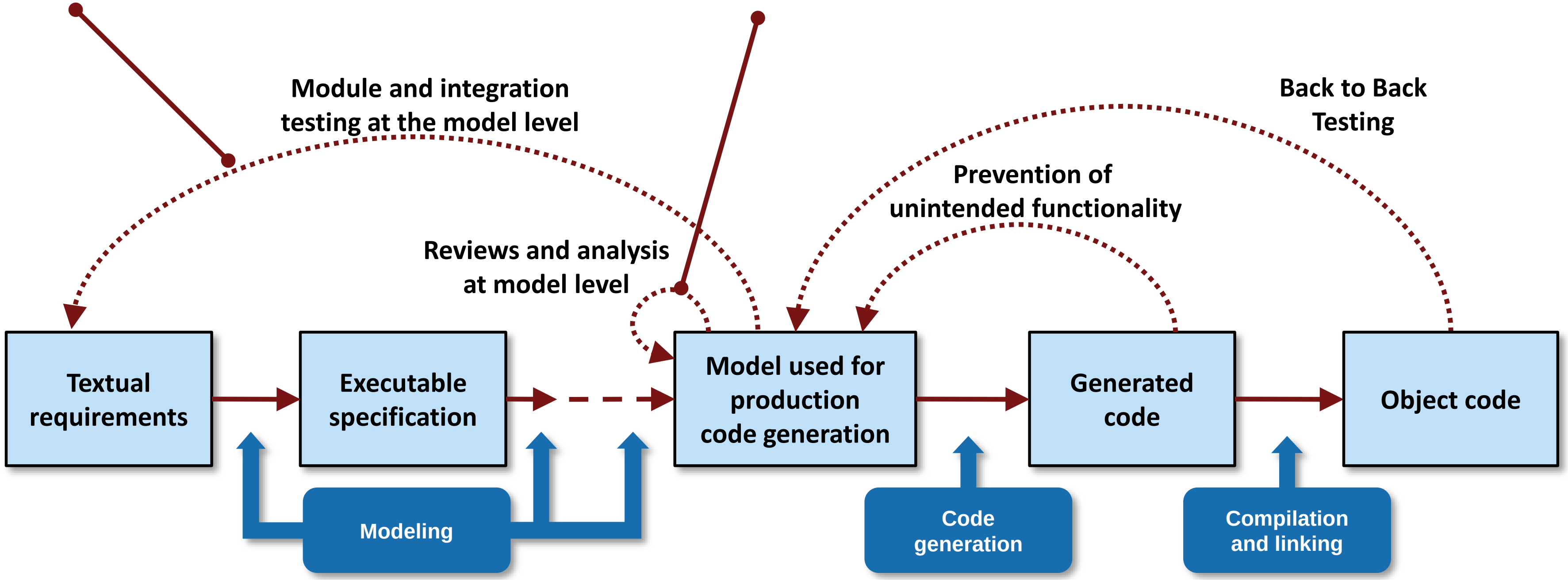
Module and integration testing at the model level
Back to Back Testing
Prevention of unintended functionality
Reviews and analysis at model level
Textual requirements
Executable specification
Model used for production code generation
Generated code
Object code
Modeling
Code generation
Compilation and linking

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



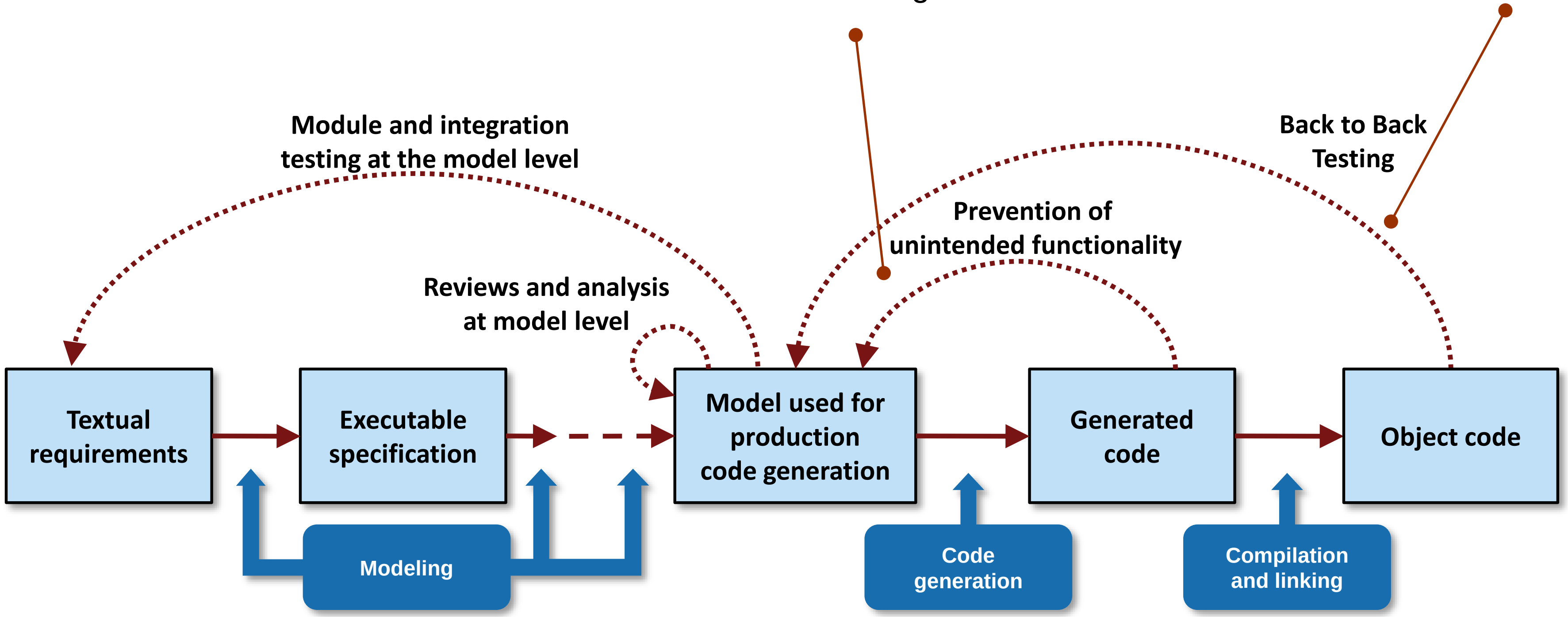
Module and integration testing at the model level
Back to Back Testing
Prevention of unintended functionality
Reviews and analysis at model level
Textual requirements
Executable specification
Model used for production code generation
Generated code
Object code
Modeling
Code generation
Compilation and linking

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

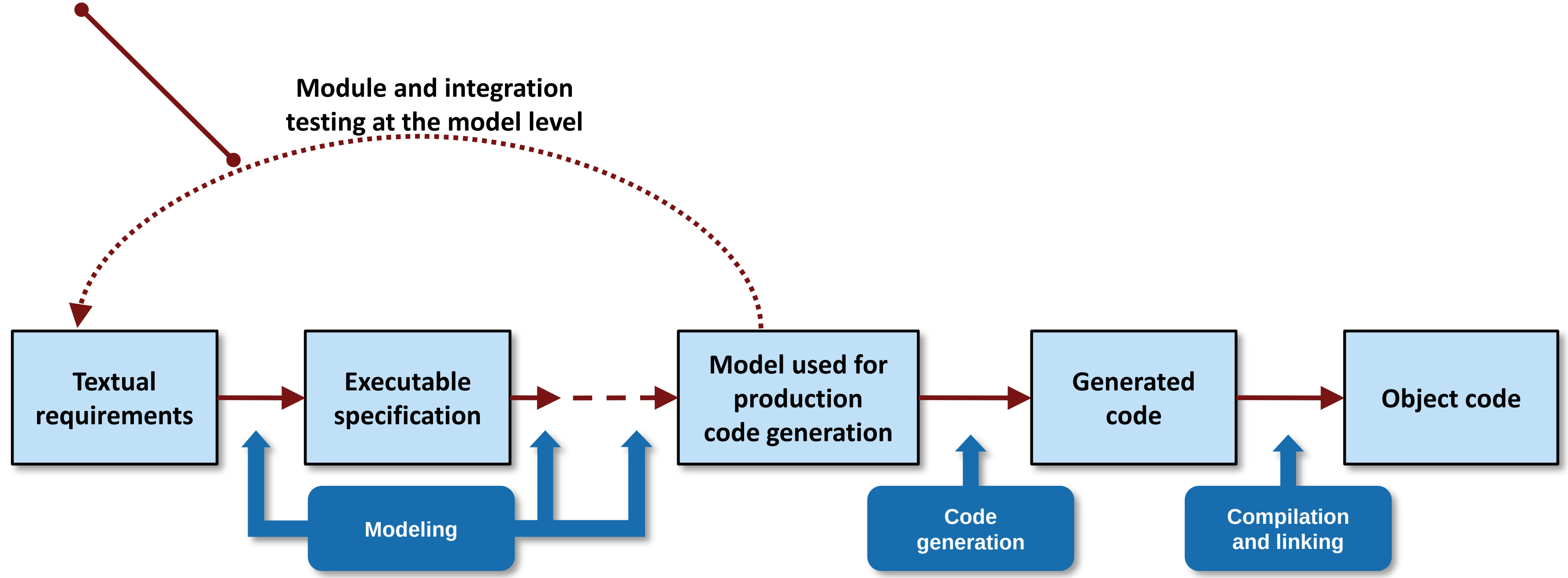


Module and integration testing at the model level
Back to Back Testing
Prevention of unintended functionality
Reviews and analysis at model level
Textual requirements
Executable specification
Model used for production code generation
Generated code
Object code
Modeling
Code generation
Compilation and linking

Manage Requirements

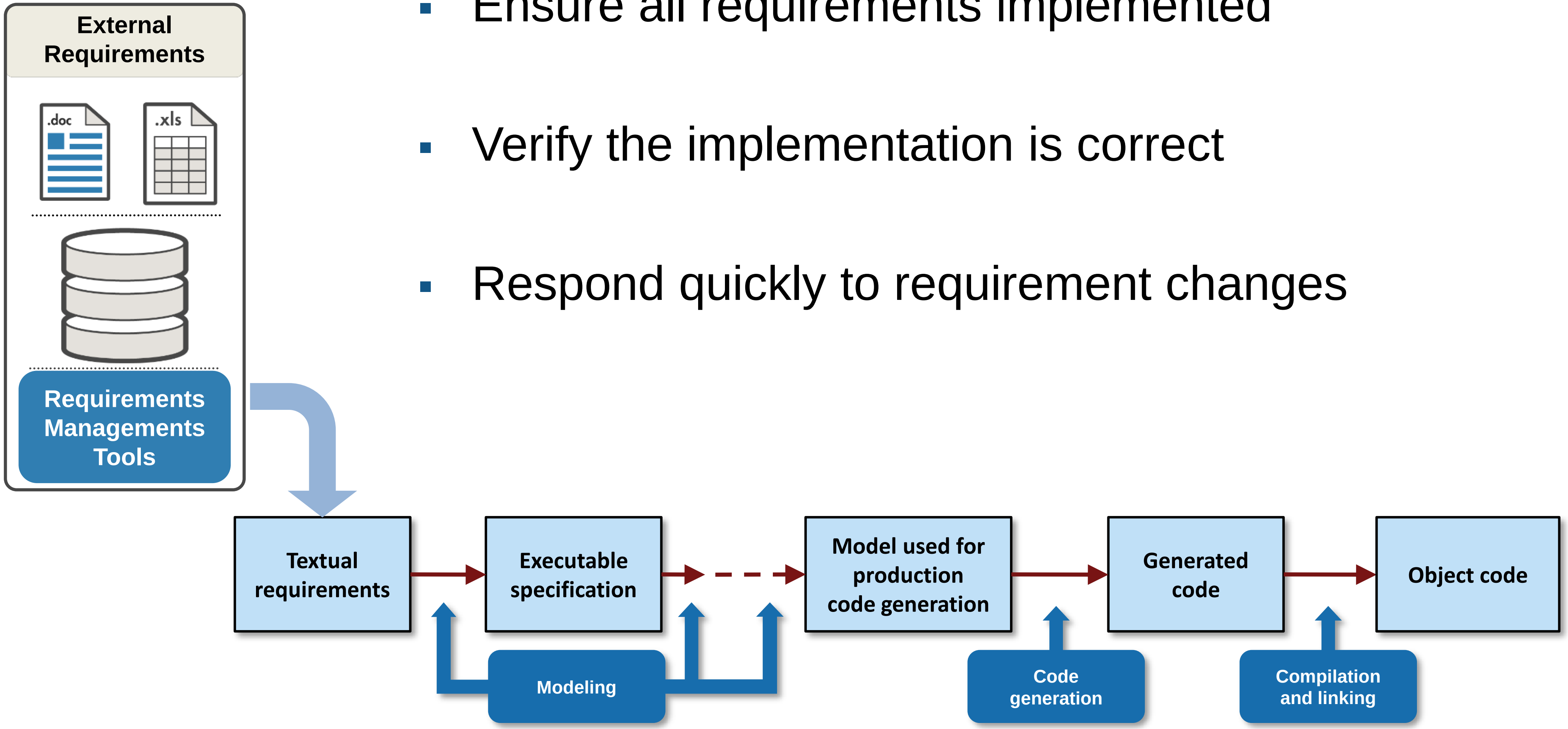
Model Verification

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

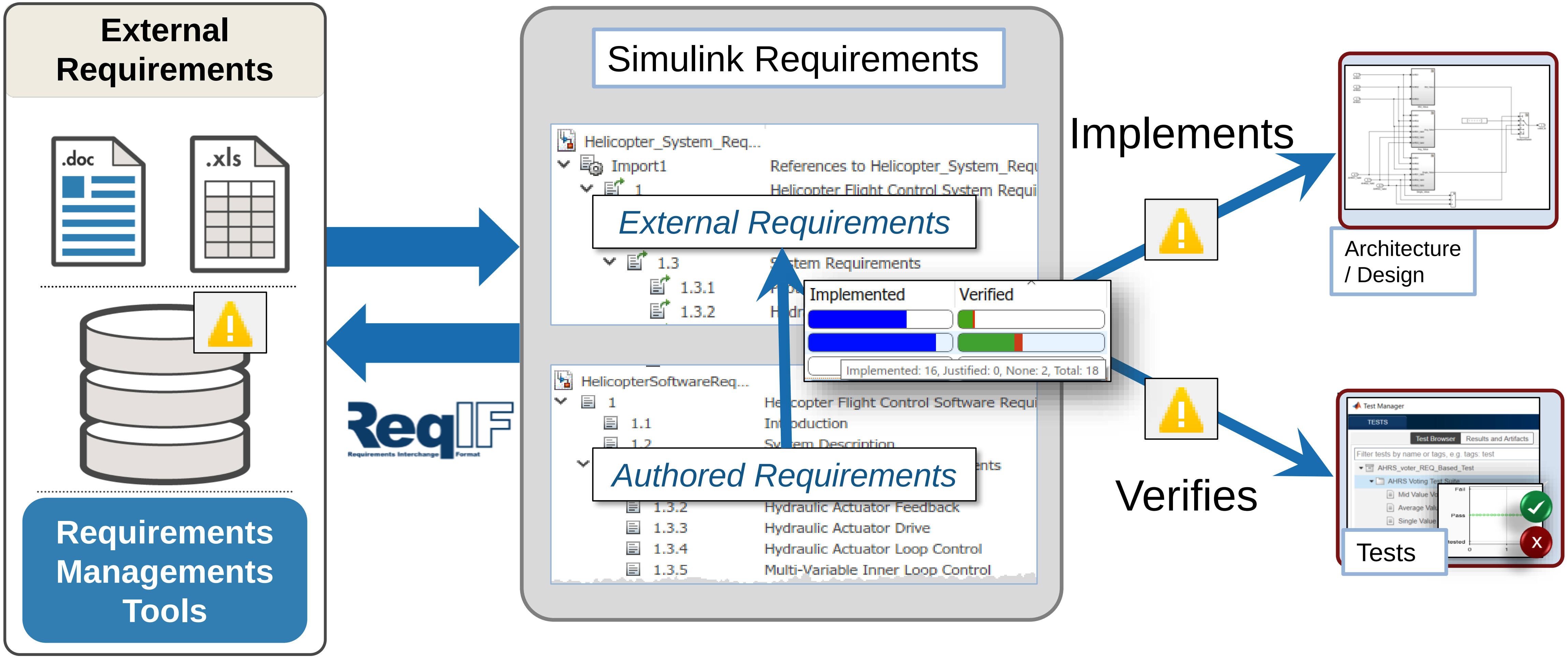


Manage Requirements

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



Work with Requirements, Architecture and Design Together



External Requirements

.doc

.xls

ReqIF

Requirements Managements Tools

Simulink Requirements

Helicopter_System_Req...

Import1 References to Helicopter_System_Requ

1 Helicopter Flight Control System Requi

External Requirements

1.3 S tem Requirements

1.3.1

1.3.2 H dr

Implemented Verified

Implemented: 16, Justified: 0, None: 2, Total: 18

HelicopterSoftwareReq...

1 He copter Flight Control Software Requi

1.1 Int oduction

1.2 Sy tem Description

Authored Requirements

ents

1.3.2 Hydraulic Actuator Feedback

1.3.3 Hydraulic Actuator Drive

1.3.4 Hydraulic Actuator Loop Control

1.3.5 Multi-Variable Inner Loop Control

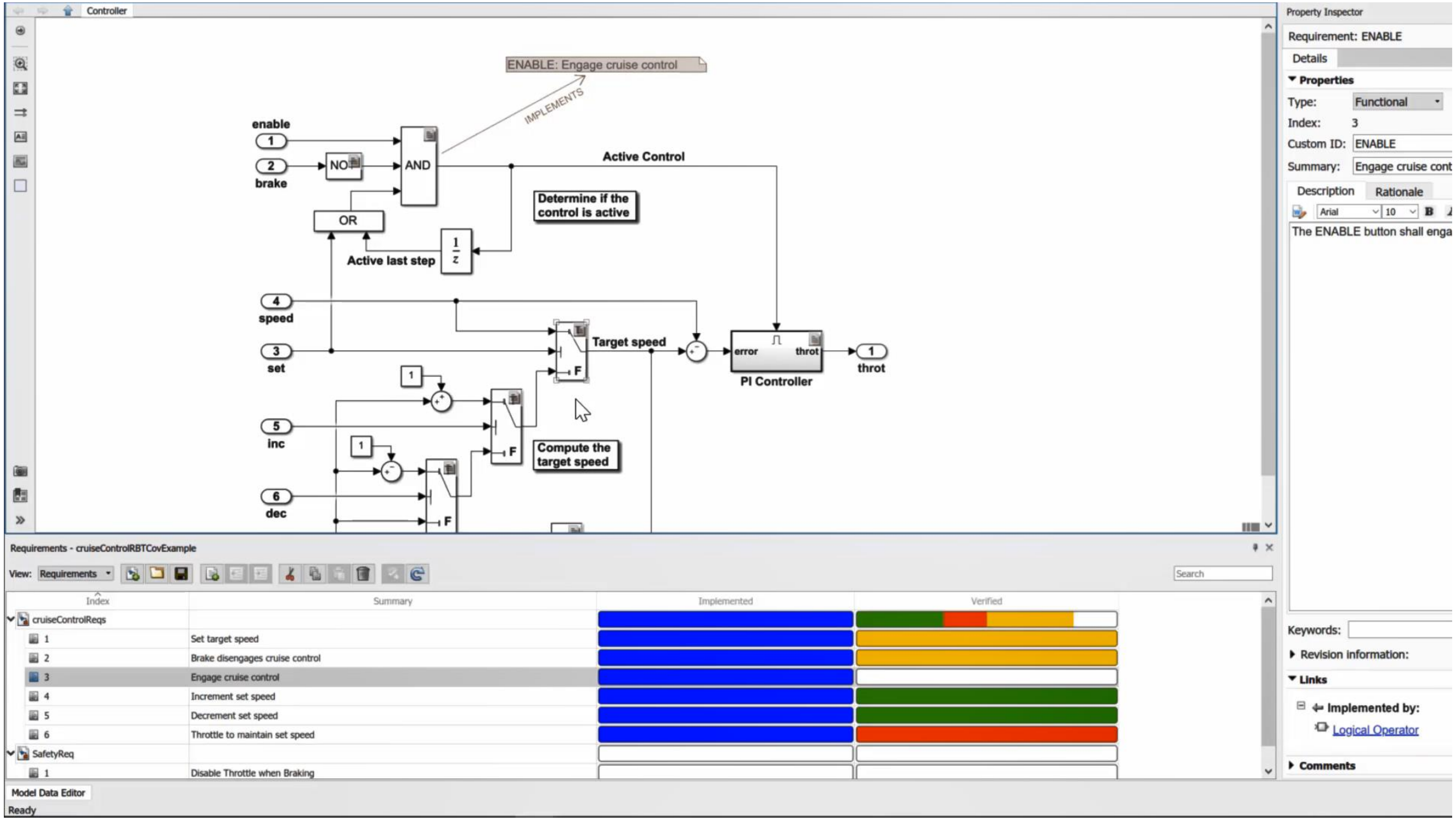
Implements

Architecture / Design

Verifies

Tests

Demo: Requirements Perspective



Requirement: ENABLE

Properties

Type: Functional

Index: 3

Custom ID: ENABLE

Summary: Engage cruise cont

The ENABLE button shall enga

Keywords:

Revision information:

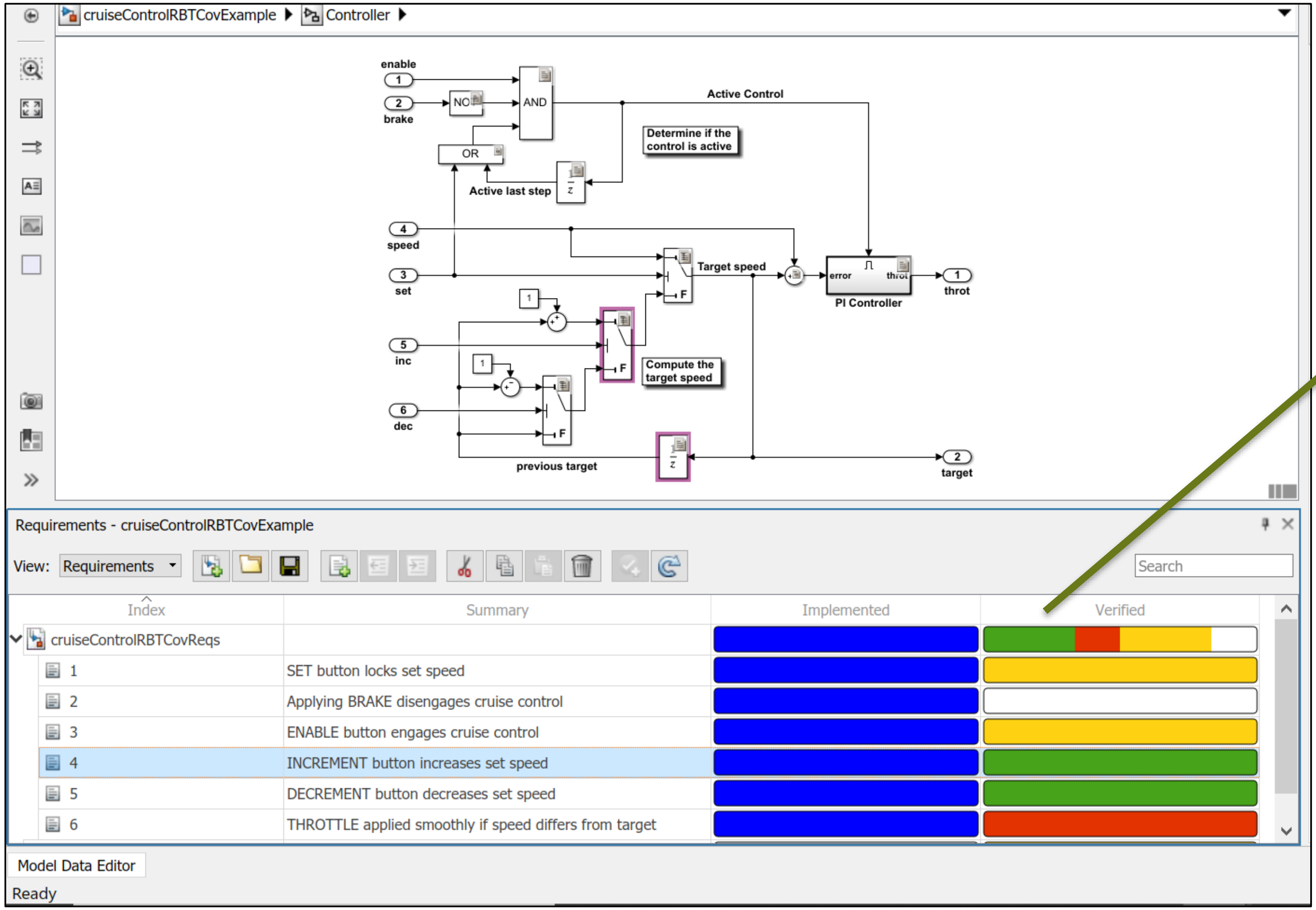
Links

Implemented by: Logical Operator

Comments

Index	Summary	Implemented	Verified
cruiseControlReqs		🟦	🟩🟥🟨⬜
1	Set target speed	🟦	🟨
2	Brake disengages cruise control	🟦	🟨
3	Engage cruise control	🟦	⬜
4	Increment set speed	🟦	🟩
5	Decrement set speed	🟦	🟩
6	Throttle to maintain set speed	🟦	🟥
SafetyReq		⬜	⬜
1	Disable Throttle when Braking	⬜	⬜

Test and Requirements Traceability



Requirements - cruiseControlRBTCovExample

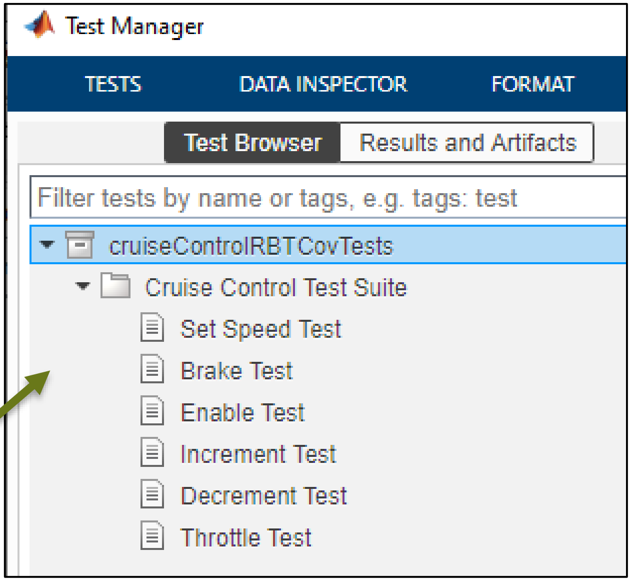
View: Requirements

Search

Index	Summary	Implemented	Verified
cruiseControlRBTCovReqs		🟦	🟩🟥🟨□
1	SET button locks set speed	🟦	🟨
2	Applying BRAKE disengages cruise control	🟦	□
3	ENABLE button engages cruise control	🟦	🟨
4	INCREMENT button increases set speed	🟦	🟩
5	DECREMENT button decreases set speed	🟦	🟩
6	THROTTLE applied smoothly if speed differs from target	🟦	🟥

Model Data Editor

Ready



Filter tests by name or tags, e.g. tags: test

- cruiseControlRBTCovTests
  - Cruise Control Test Suite
    - Set Speed Test
    - Brake Test
    - Enable Test
    - Increment Test
    - Decrement Test
    - Throttle Test

Verification Status

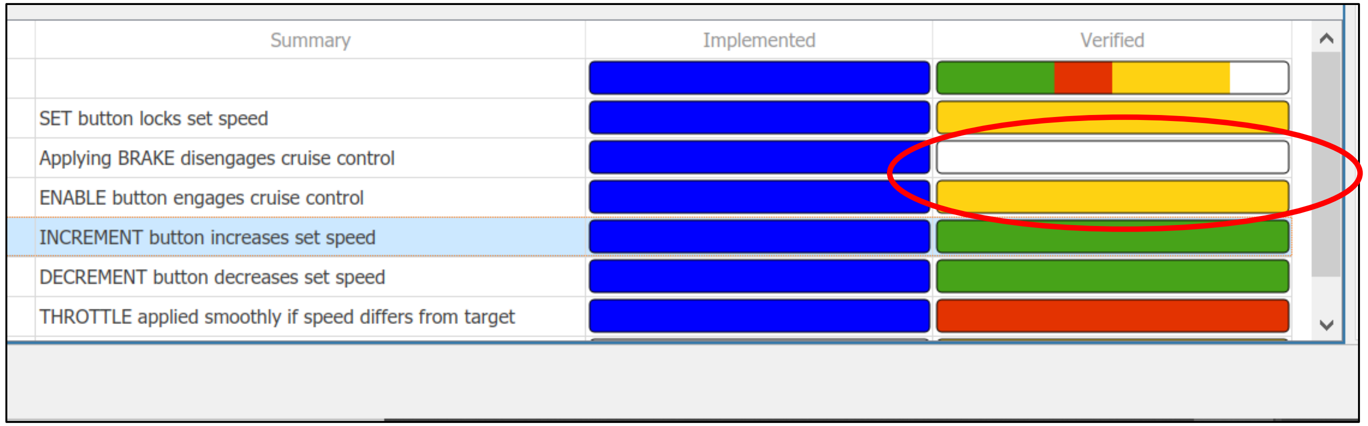
🟩 Passed

🟥 Failed

🟨 Unexecuted

□ Missing

Review and Analyze Traceability with Traceability Matrix

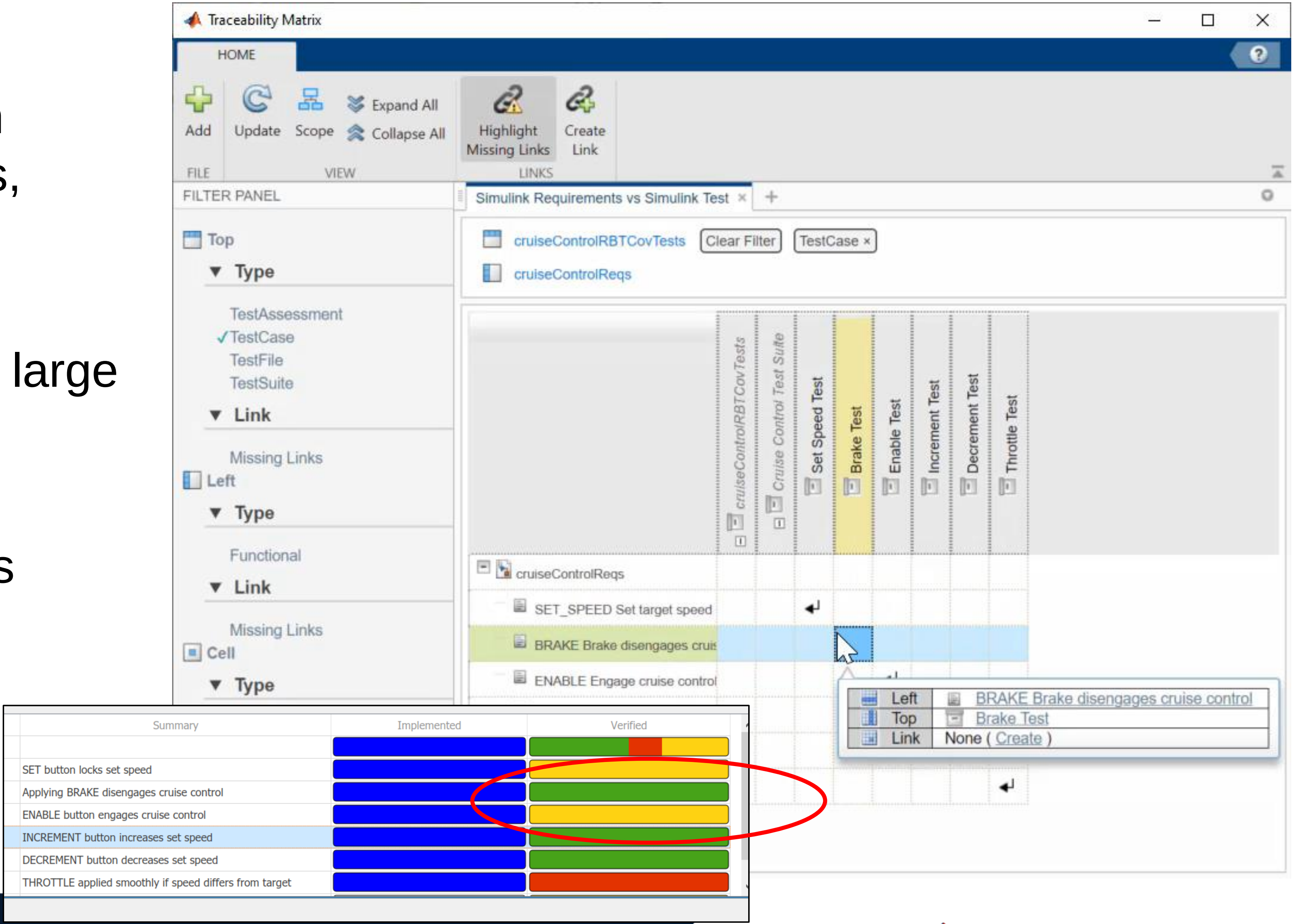


Summary	Implemented	Verified
	🟦	🟩🟥🟨⬜
SET button locks set speed	🟦	🟨
Applying BRAKE disengages cruise control	🟦	⬜
ENABLE button engages cruise control	🟦	🟨
INCREMENT button increases set speed	🟦	🟩
DECREMENT button decreases set speed	🟦	🟩
THROTTLE applied smoothly if speed differs from target	🟦	🟥

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



Left	BRAKE Brake disengages cruise control	
Top	Brake Test	
Link	None (Create)	

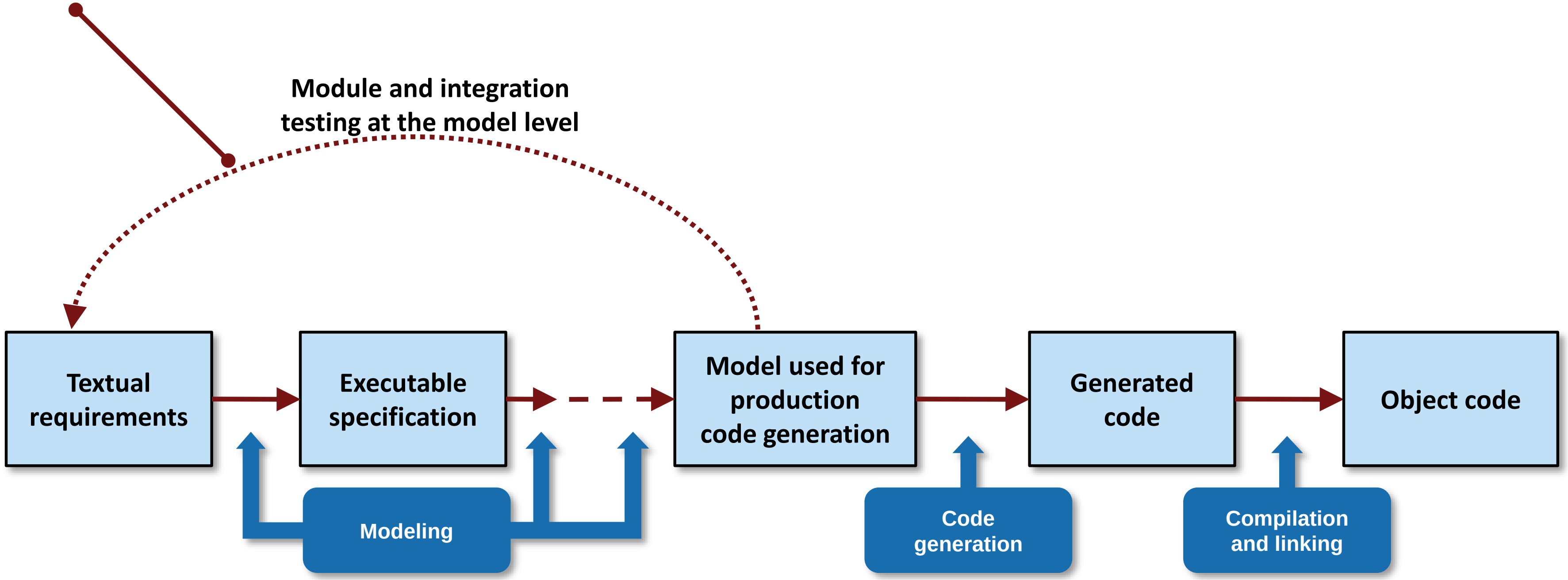
Summary	Implemented	Verified
	🟦	🟩🟥🟨
SET button locks set speed	🟦	🟨
Applying BRAKE disengages cruise control	🟦	🟩
ENABLE button engages cruise control	🟦	🟨
INCREMENT button increases set speed	🟦	🟩
DECREMENT button decreases set speed	🟦	🟩
THROTTLE applied smoothly if speed differs from target	🟦	🟥

Systematic Functional Testing of Model

Model Verification

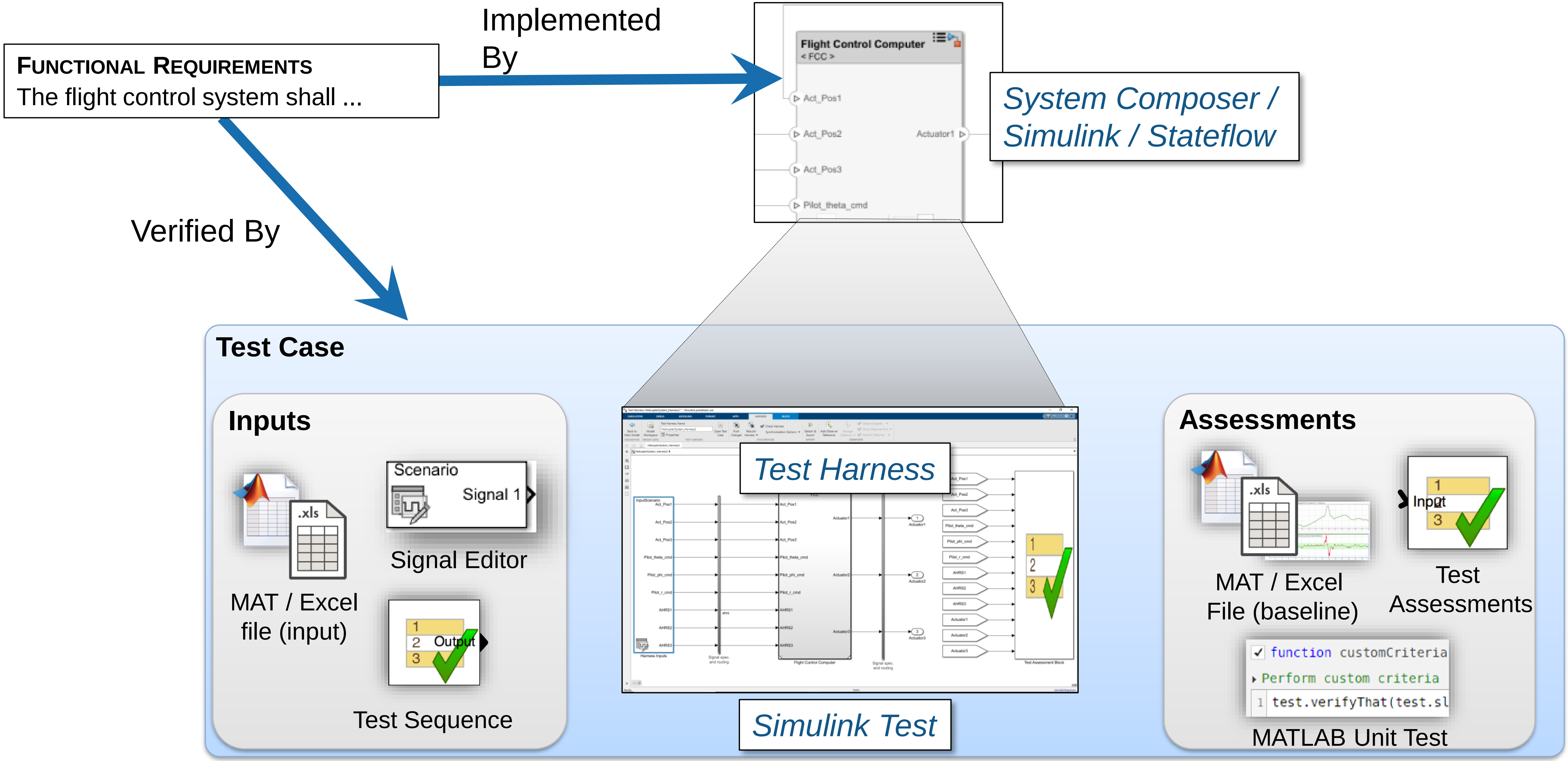
- Manage requirements
- **Systematically test**
- Measure model coverage

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



Module and integration testing at the model level

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

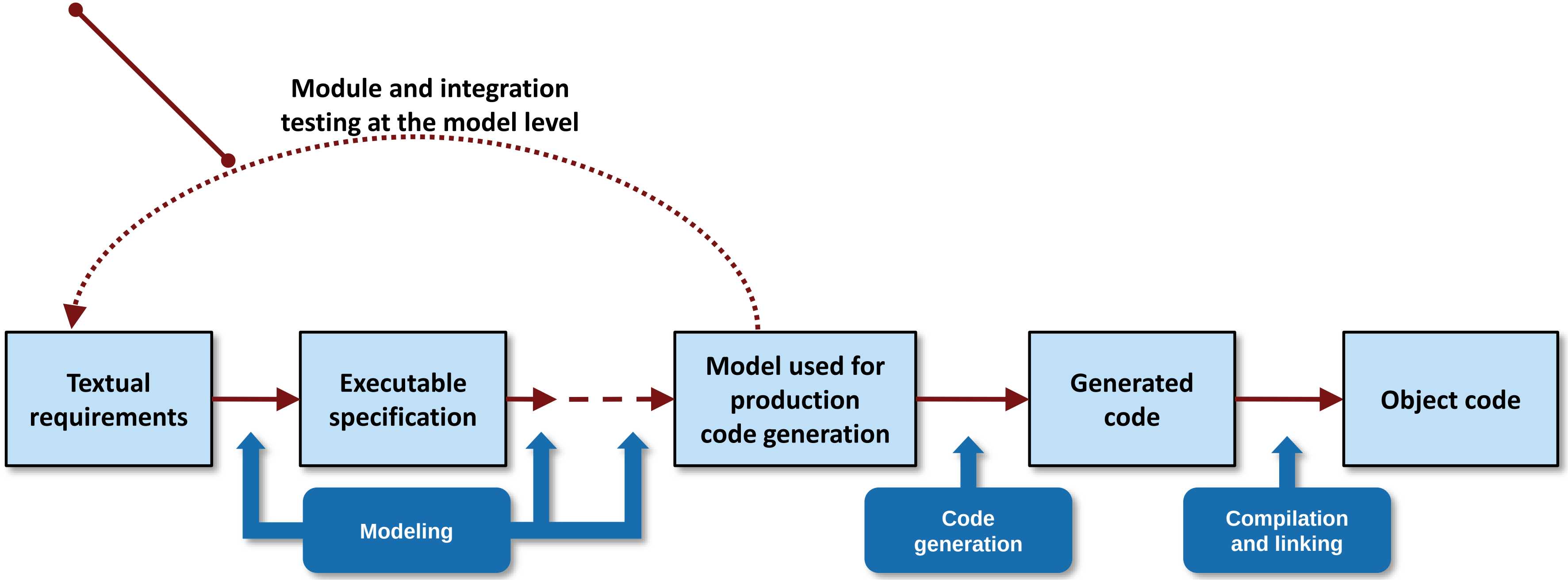
```
function customCriteria
Perform custom criteria
test.verifyThat(test.sl
```

MATLAB Unit Test

Measure completeness of testing

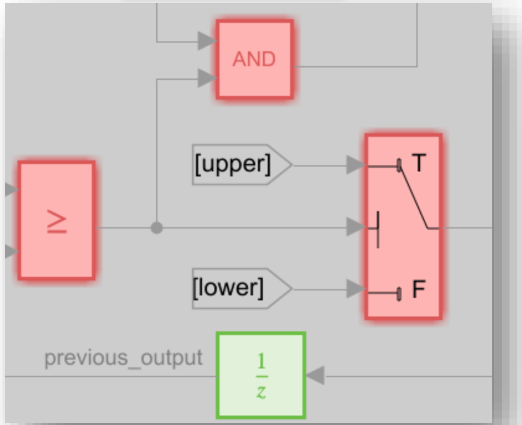
Model Verification

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

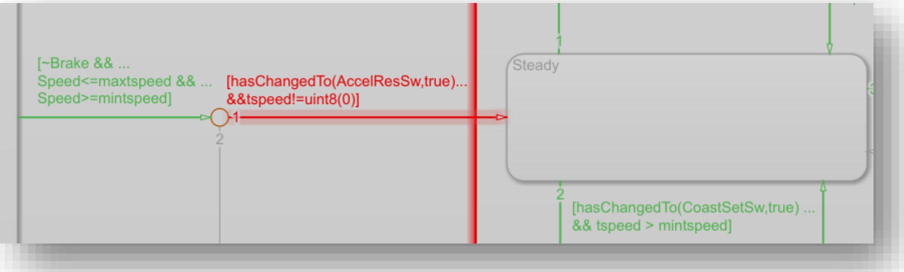


Coverage Analysis to Measure Testing

Simulink



Stateflow

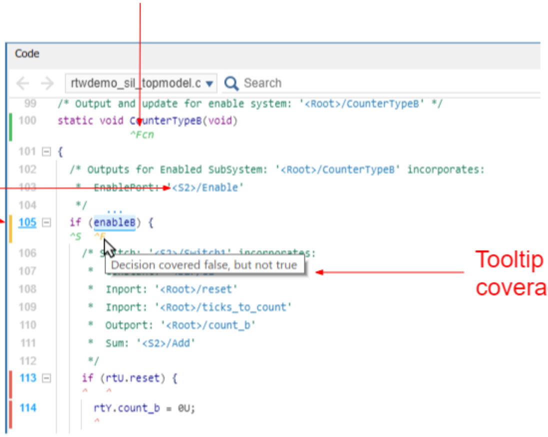


[~Brake && ... Speed<=maxtspeed && ... Speed>=mintspeed]
[hasChangedTo(AccelResSw,true)... &&tspeed!=uint8(0)]
1
2
Steady
1
2
[hasChangedTo(CoastSetSw,true) ... && tspeed > mintspeed]

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

Code

Coverage annotation



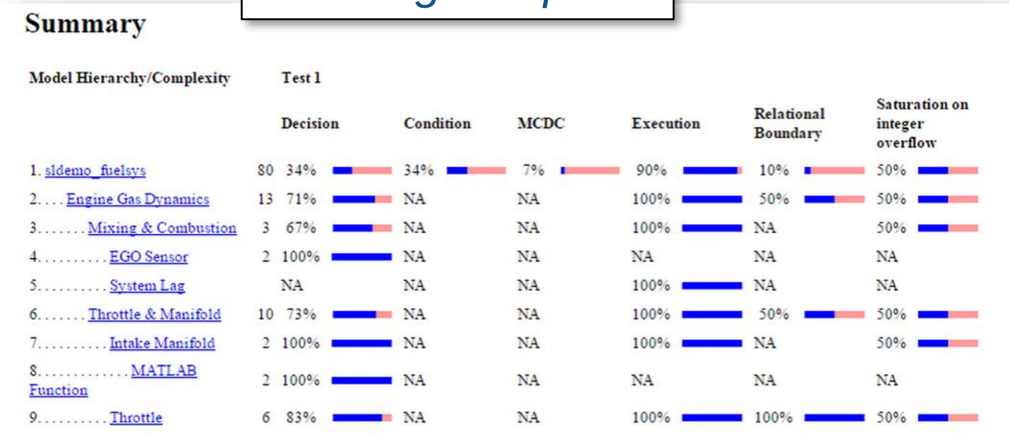
```
/* Output and update for enable system: '<Root>/CounterTypeB' */
static void CounterTypeB(void)
              ^Fcn
{
  /* Outputs for Enabled SubSystem: '<Root>/CounterTypeB' incorporates:
   *  EnablePort: '<S2>/Enable'
   */
  if (enableB) {
  ^S  ^F
    /* Switch: '<S2>/Switch1' incorporates:
     *
     *  Inport: '<Root>/reset'
     *  Inport: '<Root>/ticks_to_count'
     *  Outport: '<Root>/count_b'
     *  Sum: '<S2>/Add'
     */
    if (rtU.reset) {
    ^   ^
      rtY.count_b = 0U;
      ^
```

Decision covered false, but not true

Links to model element

Tooltip with code coverage results

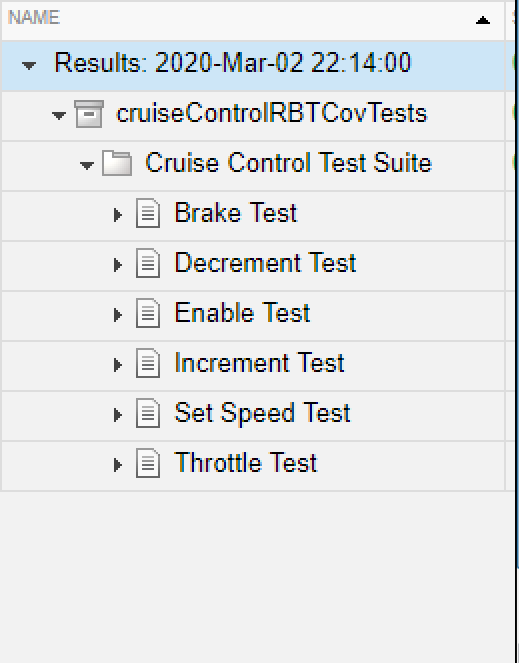
Coverage Reports



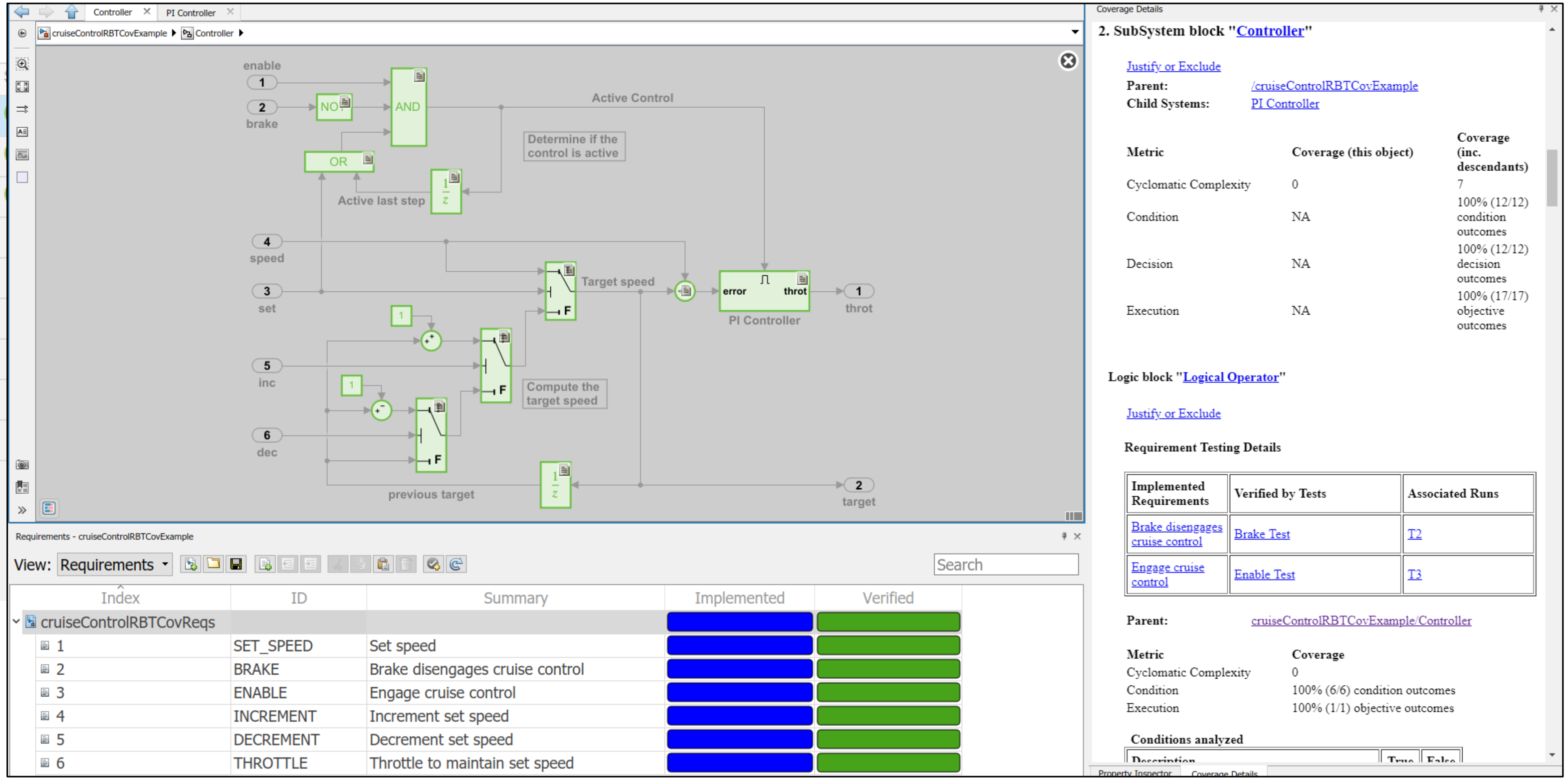
Summary

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



- Results: 2020-Mar-02 22:14:00
  - cruiseControlRBTCovTests
    - Cruise Control Test Suite
      - Brake Test
      - Decrement Test
      - Enable Test
      - Increment Test
      - Set Speed Test
      - Throttle Test



Index	ID	Summary	Implemented	Verified
cruiseControlRBTCovReqs				
1	SET_SPEED	Set speed		
2	BRAKE	Brake disengages cruise control		
3	ENABLE	Engage cruise control		
4	INCREMENT	Increment set speed		
5	DECREMENT	Decrement set speed		
6	THROTTLE	Throttle to maintain set speed		

2. SubSystem block "Controller"

Justify or Exclude

Parent: /cruiseControlRBTCovExample

Child Systems: PI Controller

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	0	7
Condition	NA	100% (12/12) condition outcomes
Decision	NA	100% (12/12) decision outcomes
Execution	NA	100% (17/17) objective outcomes

Logic block "Logical Operator"

Justify or Exclude

Requirement Testing Details

Implemented Requirements	Verified by Tests	Associated Runs
Brake disengages cruise control	Brake Test	T2
Engage cruise control	Enable Test	T3

Parent: cruiseControlRBTCovExample/Controller

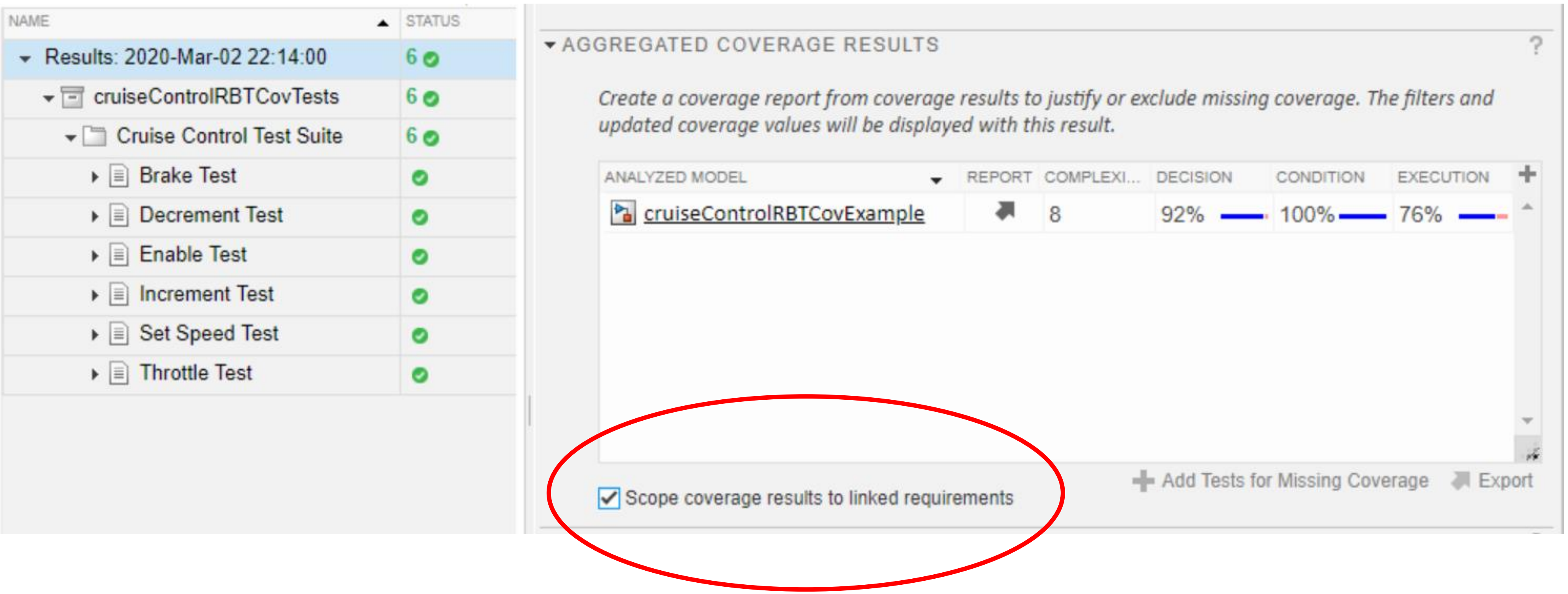
Metric	Coverage
Cyclomatic Complexity	0
Condition	100% (6/6) condition outcomes
Execution	100% (1/1) objective outcomes

Conditions analyzed

Description	True	False

Scoping Model Coverage to Requirements-Based Tests

R2020a



NAME	STATUS
Results: 2020-Mar-02 22:14:00	6 ✓
cruiseControlRBTCovTests	6 ✓
Cruise Control Test Suite	6 ✓
Brake Test	✓
Decrement Test	✓
Enable Test	✓
Increment Test	✓
Set Speed Test	✓
Throttle Test	✓

AGGREGATED COVERAGE RESULTS

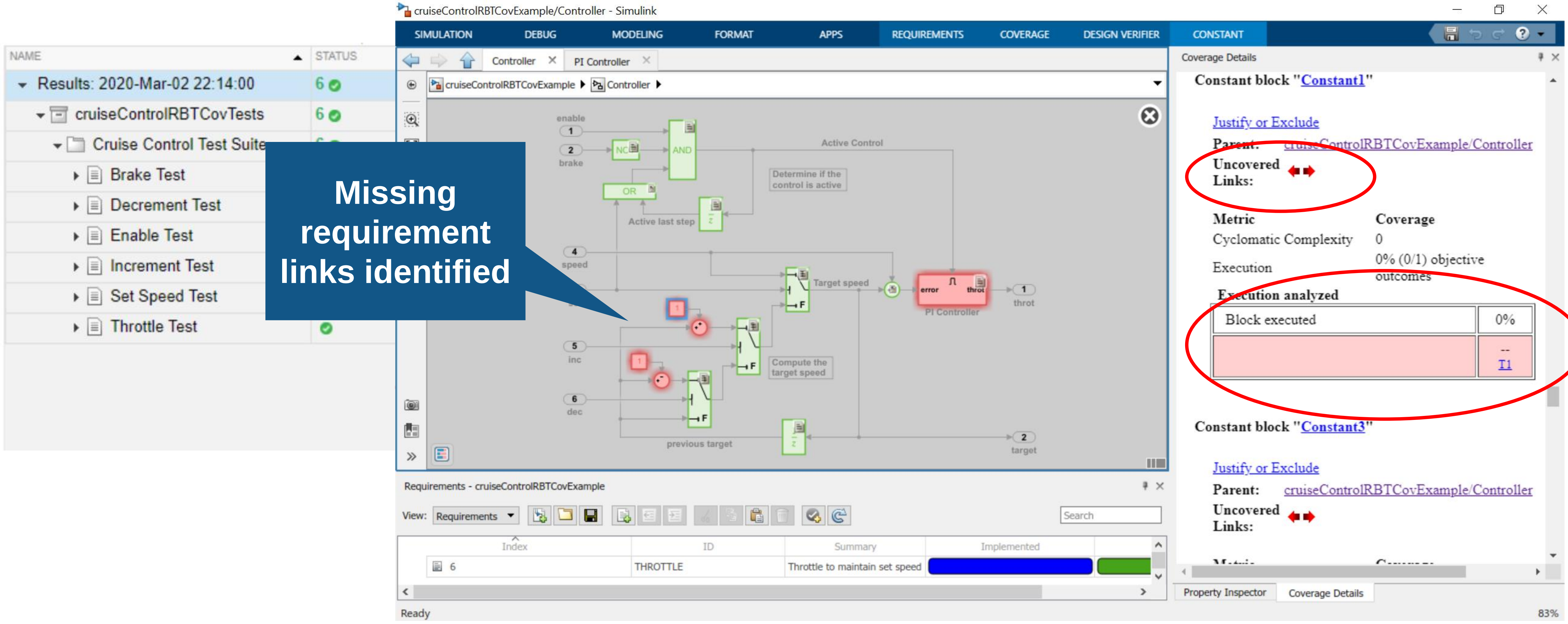
Create a coverage report from coverage results to justify or exclude missing coverage. The filters and updated coverage values will be displayed with this result.

ANALYZED MODEL	REPORT	COMPLEXI...	DECISION	CONDITION	EXECUTION
cruiseControlRBTCovExample		8	92%	100%	76%

☑ Scope coverage results to linked requirements

Scoping Model Coverage to Requirements-Based Tests

R2020a



Missing requirement links identified

NAME	STATUS
Results: 2020-Mar-02 22:14:00	6 ✓
cruiseControlRBTCovTests	6 ✓
Cruise Control Test Suite	
Brake Test	
Decrement Test	
Enable Test	
Increment Test	
Set Speed Test	
Throttle Test	✓

cruiseControlRBTCovExample/Controller - Simulink

Coverage Details

Constant block "Constant1"

Justify or Exclude

Parent: cruiseControlRBTCovExample/Controller

Uncovered Links: ⬅➡

Metric	Coverage
Cyclomatic Complexity	0
Execution	0% (0/1) objective outcomes

Execution analyzed

Block executed	0%
	-- T1

Constant block "Constant3"

Justify or Exclude

Parent: cruiseControlRBTCovExample/Controller

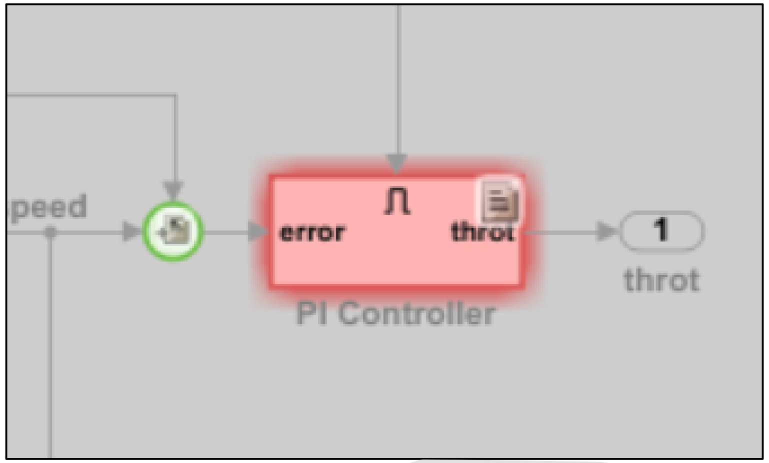
Uncovered Links: ⬅➡

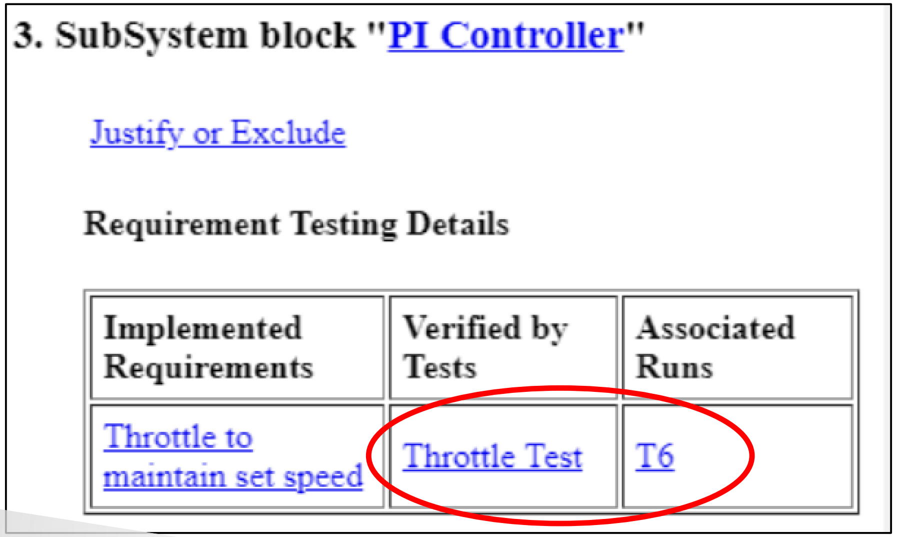
Requirements - cruiseControlRBTCovExample

View: Requirements

Index	ID	Summary	Implemented
6	THROTTLE	Throttle to maintain set speed	

Test and Requirements Traceability in Coverage Results

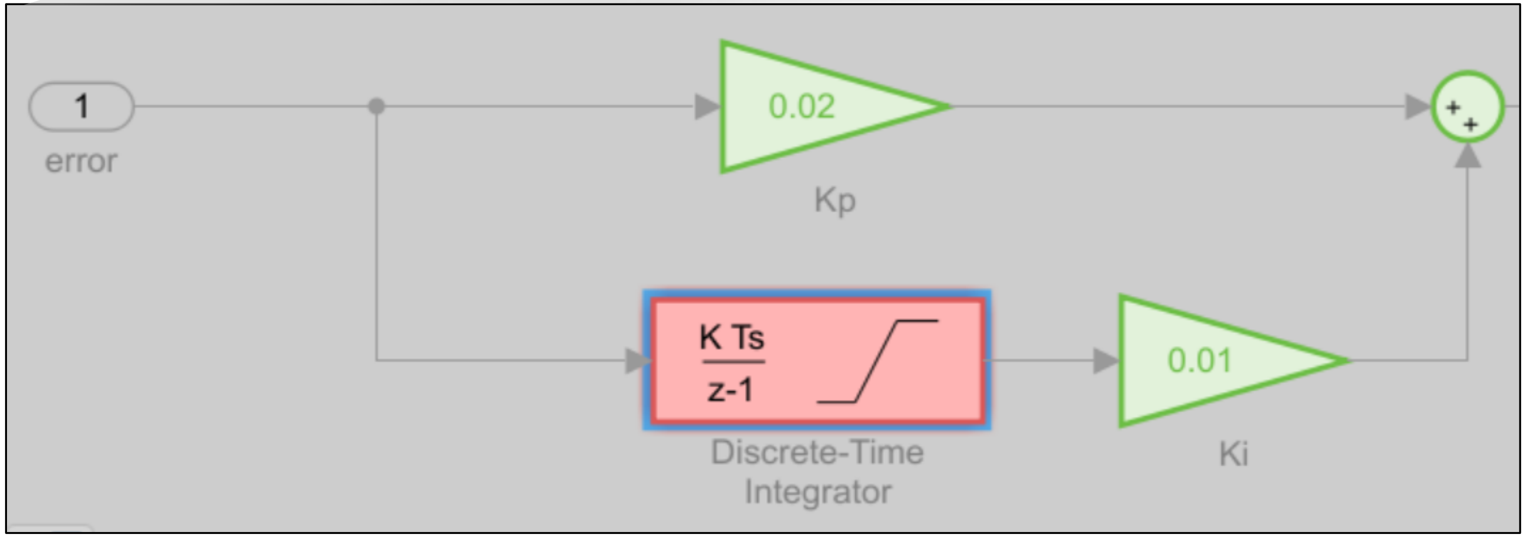



3. SubSystem block "PI Controller"

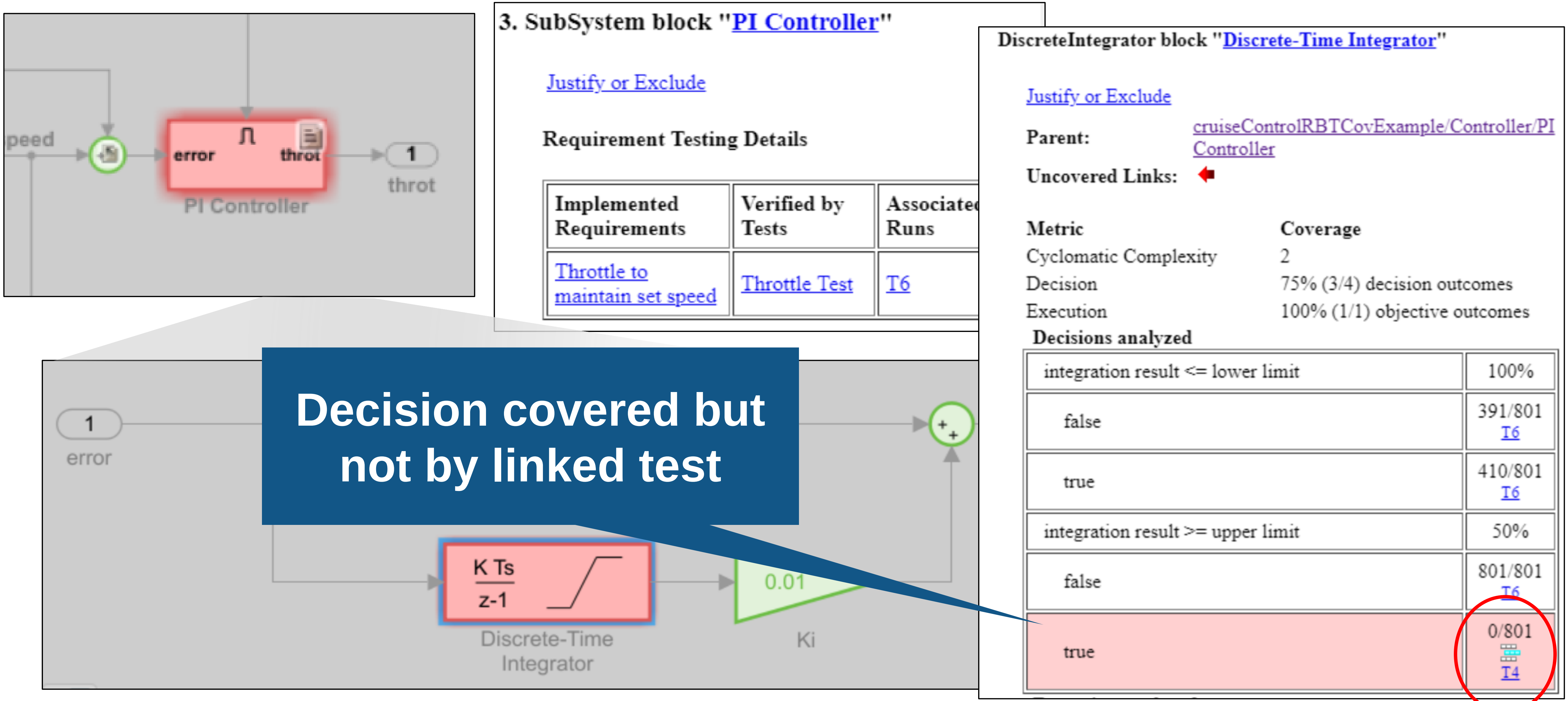
Justify or Exclude

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

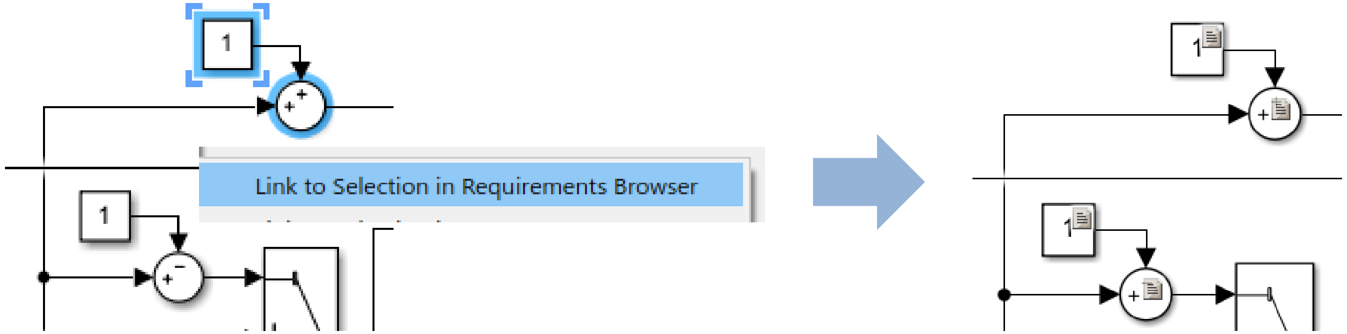
Decisions analyzed

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

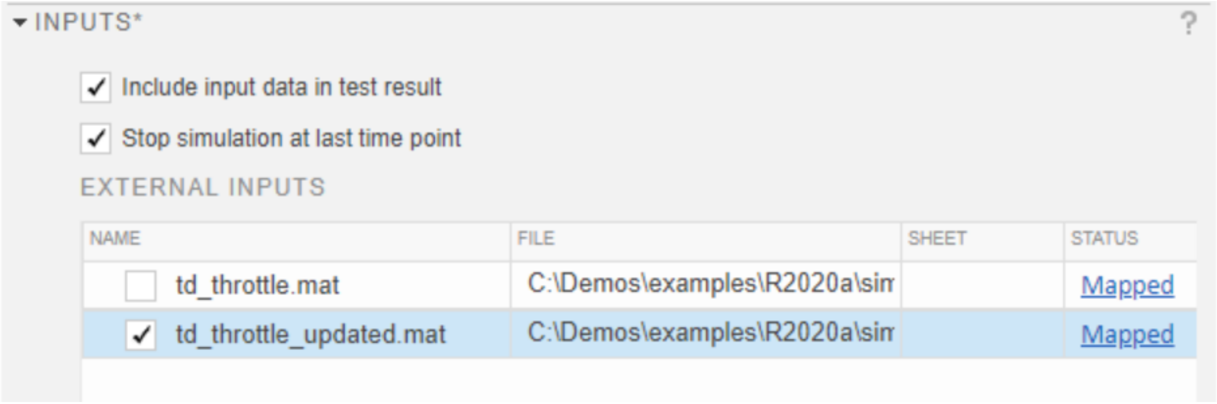
Decision covered but not by linked test

Address missing Requirements Based Test Coverage

- Add missing implementation links to requirements



- Update test to increase target speed



▾ INPUTS*

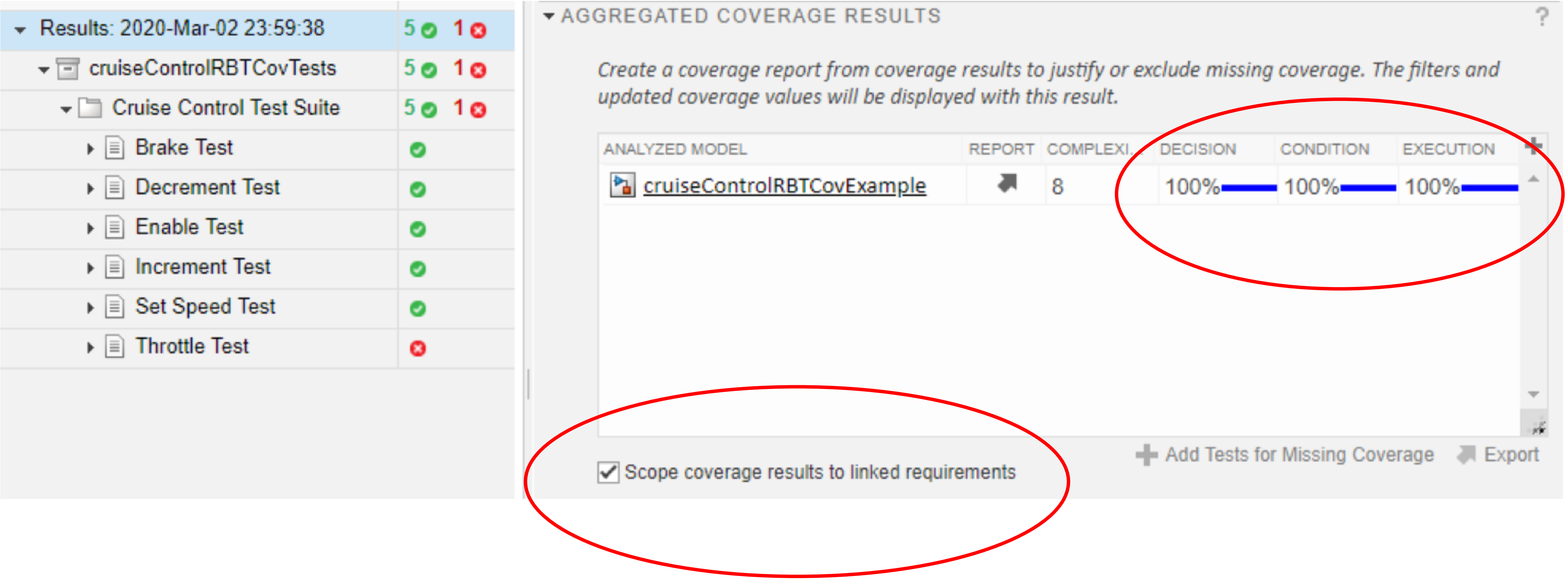
☑ Include input data in test result

☑ Stop simulation at last time point

EXTERNAL INPUTS

	NAME	FILE	SHEET	STATUS
□	td_throttle.mat	C:\Demos\examples\R2020a\sir		Mapped
☑	td_throttle_updated.mat	C:\Demos\examples\R2020a\sir		Mapped

100% Coverage but Testing Identified Error in Implementation



Results: 2020-Mar-02 23:59:38	5 ✓	1 ✗
cruiseControlRBTCovTests	5 ✓	1 ✗
Cruise Control Test Suite	5 ✓	1 ✗
Brake Test	✓	
Decrement Test	✓	
Enable Test	✓	
Increment Test	✓	
Set Speed Test	✓	
Throttle Test	✗	

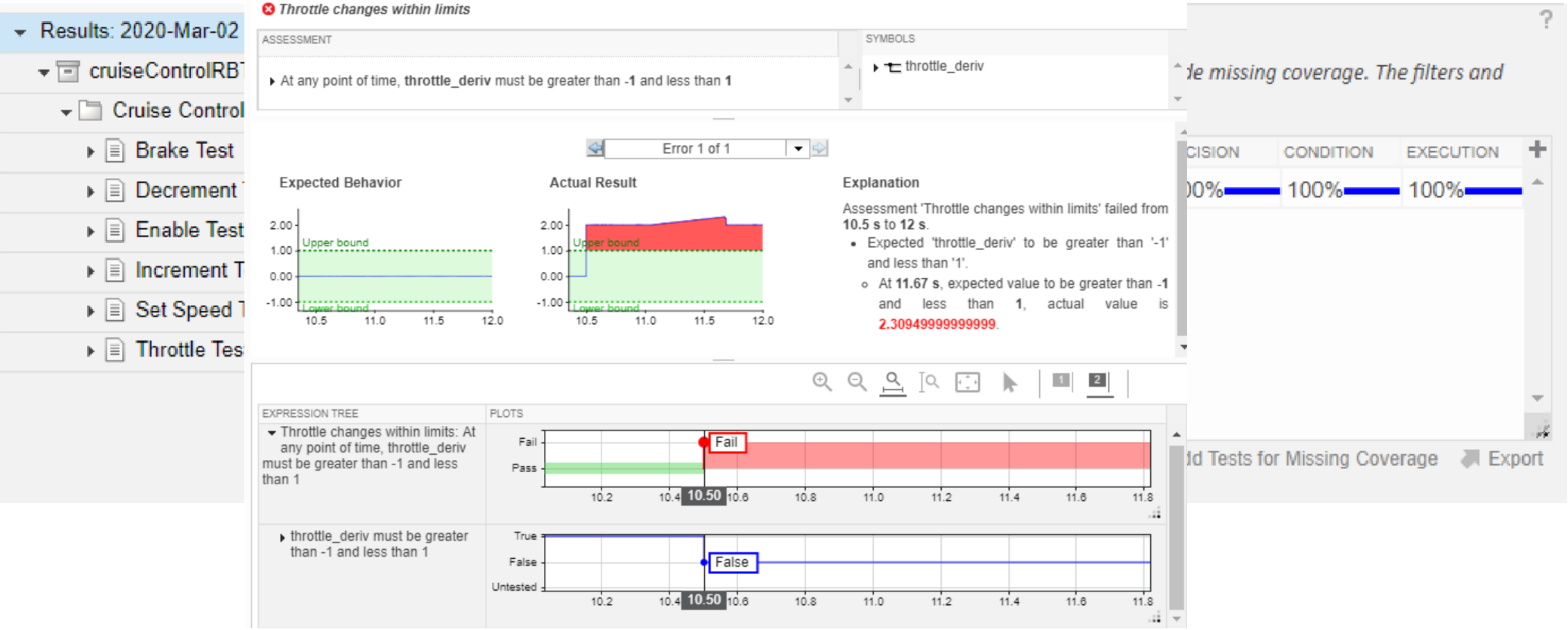
AGGREGATED COVERAGE RESULTS

Create a coverage report from coverage results to justify or exclude missing coverage. The filters and updated coverage values will be displayed with this result.

ANALYZED MODEL	REPORT	COMPLEXI...	DECISION	CONDITION	EXECUTION
cruiseControlRBTCovExample		8	100%	100%	100%

☑ Scope coverage results to linked requirements

Additional Testing Identified Error in Implementation



Results: 2020-Mar-02
- cruiseControlRBT
  - Cruise Control
    - Brake Test
    - Decrement
    - Enable Test
    - Increment T
    - Set Speed T
    - Throttle Tes

Throttle changes within limits

ASSESSMENT

At any point of time, **throttle_deriv** must be greater than **-1** and less than **1**

SYMBOLS

throttle_deriv

Error 1 of 1

Expected Behavior

Actual Result

Explanation

Assessment 'Throttle changes within limits' failed from **10.5** s to **12** s.

- Expected 'throttle_deriv' to be greater than '-1' and less than '1'.
  - At **11.67** s, expected value to be greater than **-1** and less than **1**, actual value is 2.30949999999999.

EXPRESSION TREE

Throttle changes within limits: At any point of time, throttle_deriv must be greater than -1 and less than 1

throttle_deriv must be greater than -1 and less than 1

PLOTS

ide missing coverage. The filters and

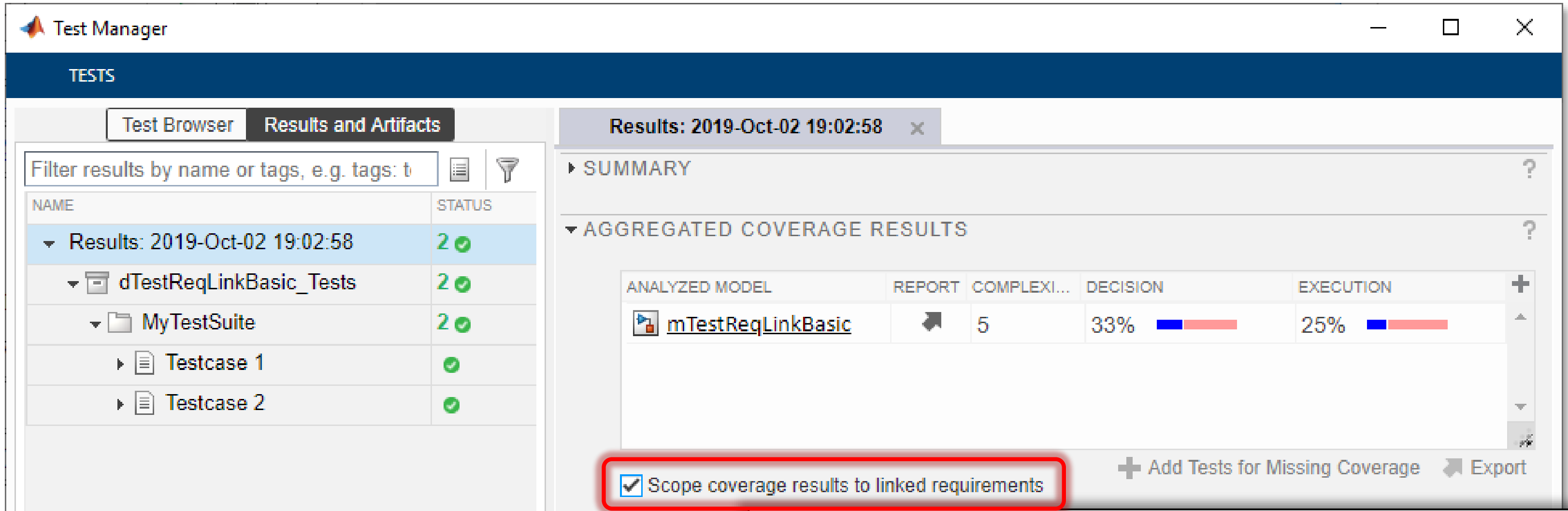
CISION	CONDITION	EXECUTION
00%	100%	100%

dd Tests for Missing Coverage

Export

Scoped Model Coverage to Requirements-Based Tests

R2020a



Test Manager

TESTS

Test Browser | Results and Artifacts

Results: 2019-Oct-02 19:02:58

Filter results by name or tags, e.g. tags: t

NAME	STATUS
Results: 2019-Oct-02 19:02:58	2 ✓
dTestReqLinkBasic_Tests	2 ✓
MyTestSuite	2 ✓
Testcase 1	✓
Testcase 2	✓

SUMMARY

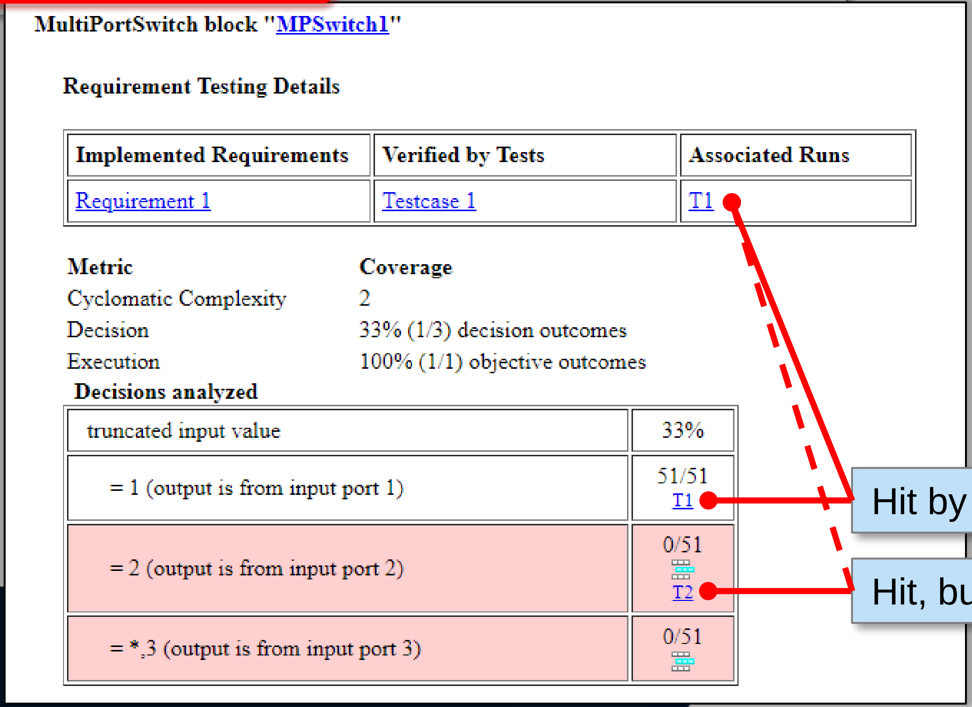
AGGREGATED COVERAGE RESULTS

ANALYZED MODEL	REPORT	COMPLEXI...	DECISION	EXECUTION
mTestReqLinkBasic		5	33%	25%

☑ Scope coverage results to linked requirements

Add Tests for Missing Coverage

Export



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**

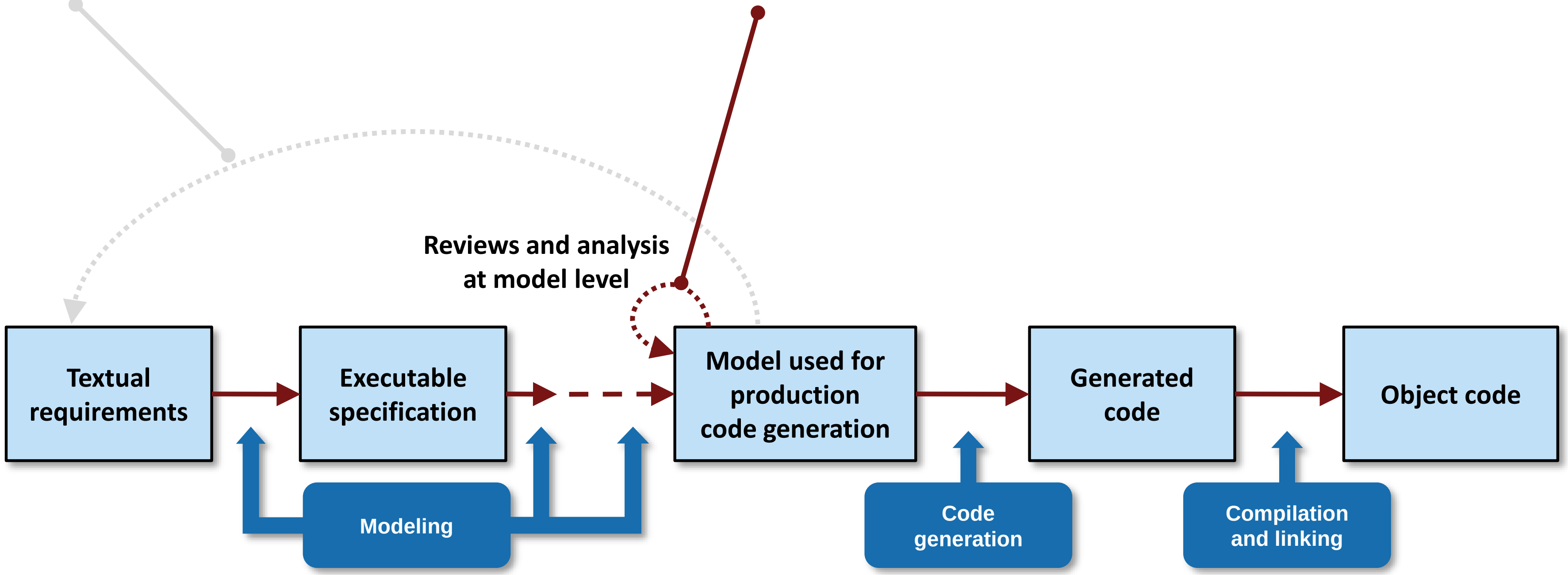
DO-178C 6.4.4.2

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

Check standard compliance

Model Verification

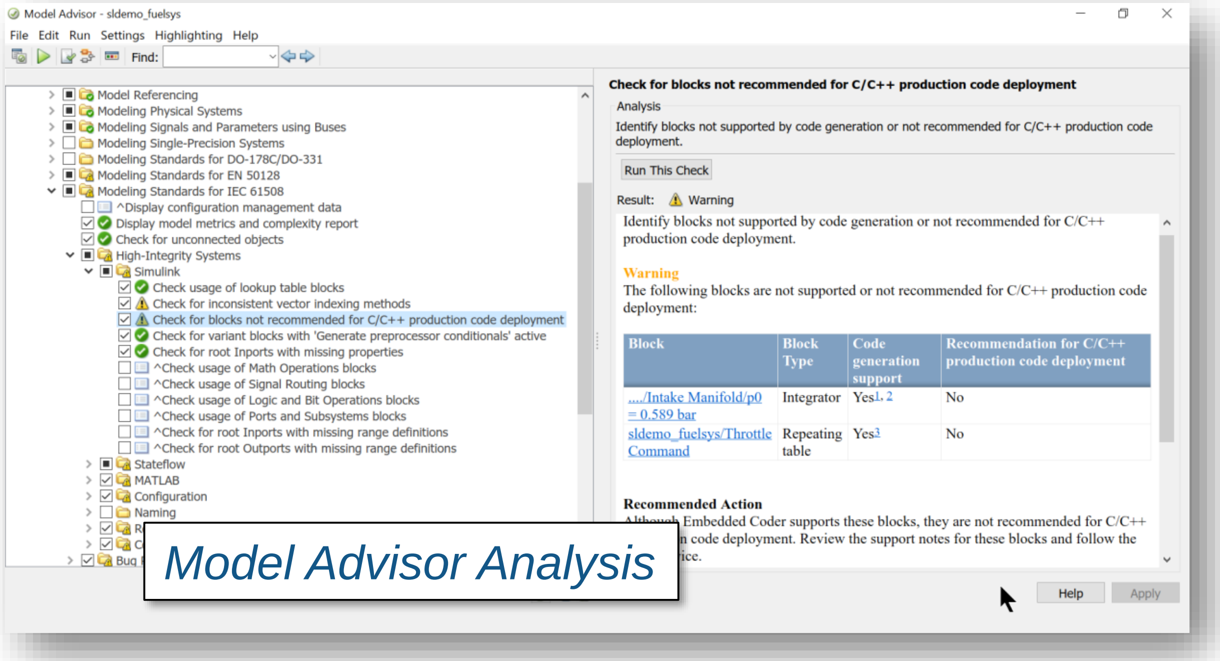
- Manage requirements
- Systematically test
- Measure model coverage
- **Check standard compliance**
- Detect design errors
- Prove model behavior compliance



Verify Design to Guidelines and Standards

Check for:

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



Model Advisor - sldemo_fuelsys

- ■ Model Referencing
- ■ Modeling Physical Systems
- ■ Modeling Signals and Parameters using Buses
- □ Modeling Single-Precision Systems
- □ Modeling Standards for DO-178C/DO-331
- ■ Modeling Standards for EN 50128
- ■ Modeling Standards for IEC 61508
  - □ ^Display configuration management data
  - ☑ ✓ Display model metrics and complexity report
  - ☑ ✓ Check for unconnected objects
  - ■ High-Integrity Systems
    - ■ Simulink
      - ☑ ✓ Check usage of lookup table blocks
      - ☑ ⚠ Check for inconsistent vector indexing methods
      - ☑ ⚠ Check for blocks not recommended for C/C++ production code deployment
      - ☑ ✓ Check for variant blocks with 'Generate preprocessor conditionals' active
      - ☑ ✓ Check for root Inports with missing properties
      - □ ^Check usage of Math Operations blocks
      - □ ^Check usage of Signal Routing blocks
      - □ ^Check usage of Logic and Bit Operations blocks
      - □ ^Check usage of Ports and Subsystems blocks
      - □ ^Check for root Inports with missing range definitions
      - □ ^Check for root Outports with missing range definitions
    - ■ Stateflow
    - ☑ MATLAB
    - ☑ Configuration
    - □ Naming
    - ☑ R
    - ☑ C
  - ☑ Bug

Check for blocks not recommended for C/C++ production code deployment

Analysis

Identify blocks not supported by code generation or not recommended for C/C++ production code deployment.

Result: ⚠ Warning

Identify blocks not supported by code generation or not recommended for C/C++ production code deployment.

Warning

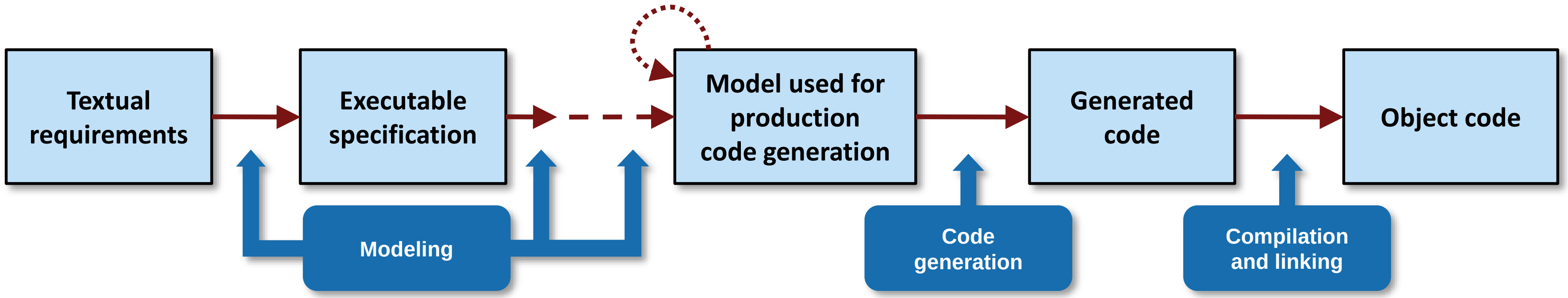
The following blocks are not supported or not recommended for C/C++ production code deployment:

Block	Block Type	Code generation support	Recommendation for C/C++ production code deployment
..../Intake Manifold/p0 = 0.589 bar	Integrator	Yes[1, 2]	No
sldemo_fuelsys/Throttle Command	Repeating table	Yes[3]	No

Recommended Action

Although Embedded Coder supports these blocks, they are not recommended for C/C++ n code deployment. Review the support notes for these blocks and follow the ice.

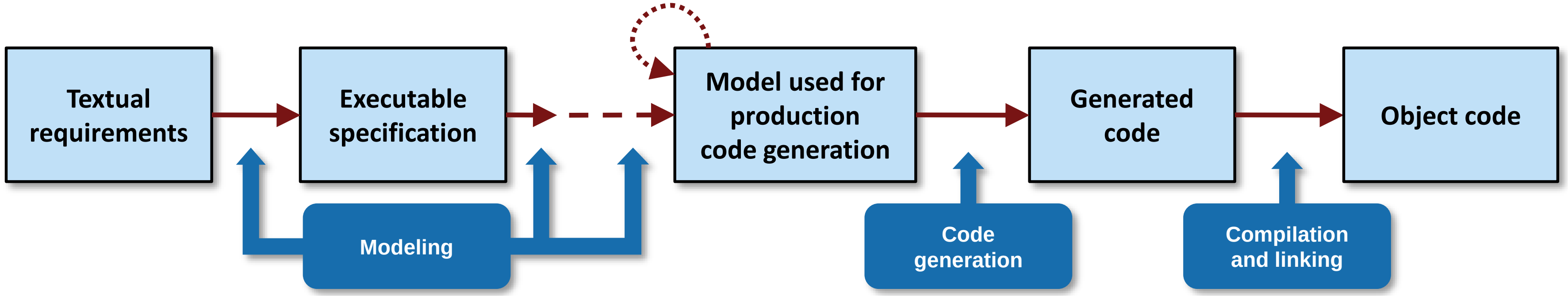
Model Advisor Analysis



Built in checks for industry standards and guidelines

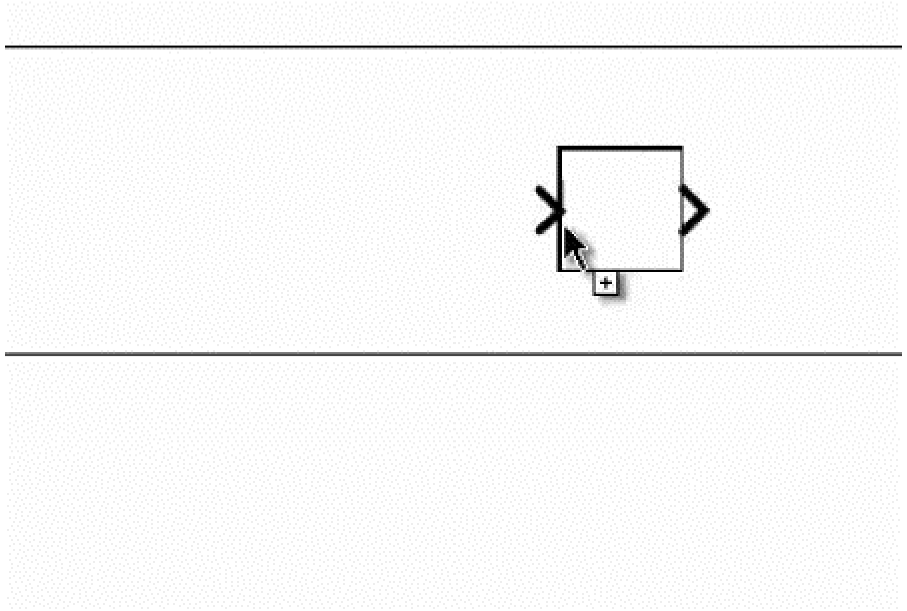
- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128

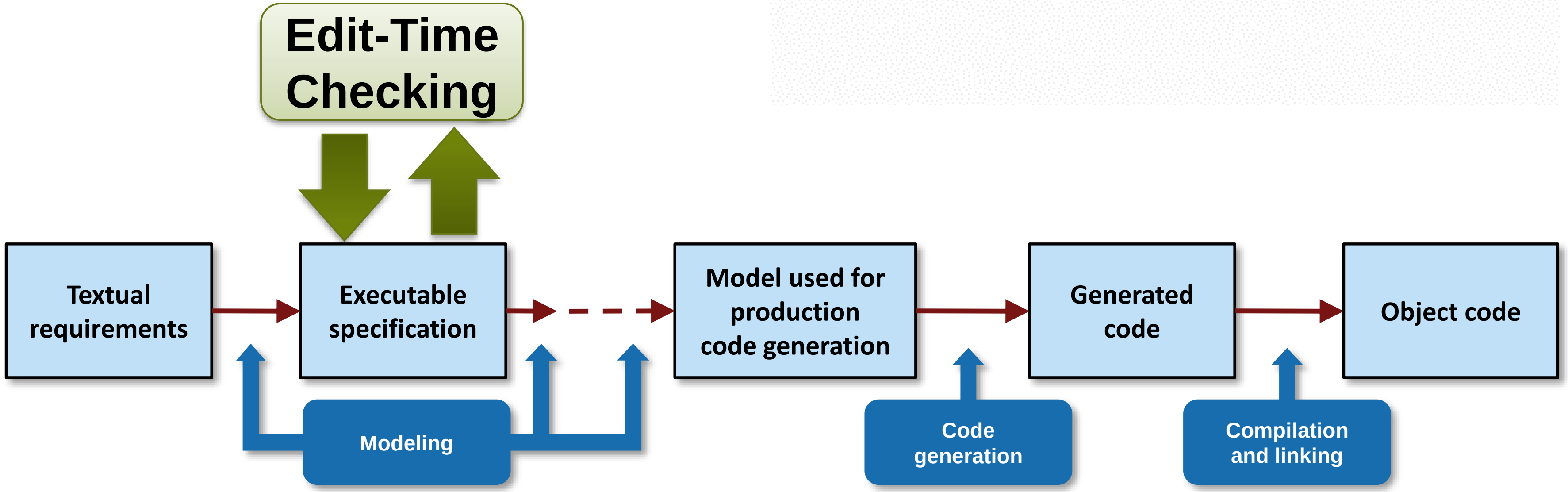
- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAB (MathWorks Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)



Shift Verification Earlier With Edit-Time Checking

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



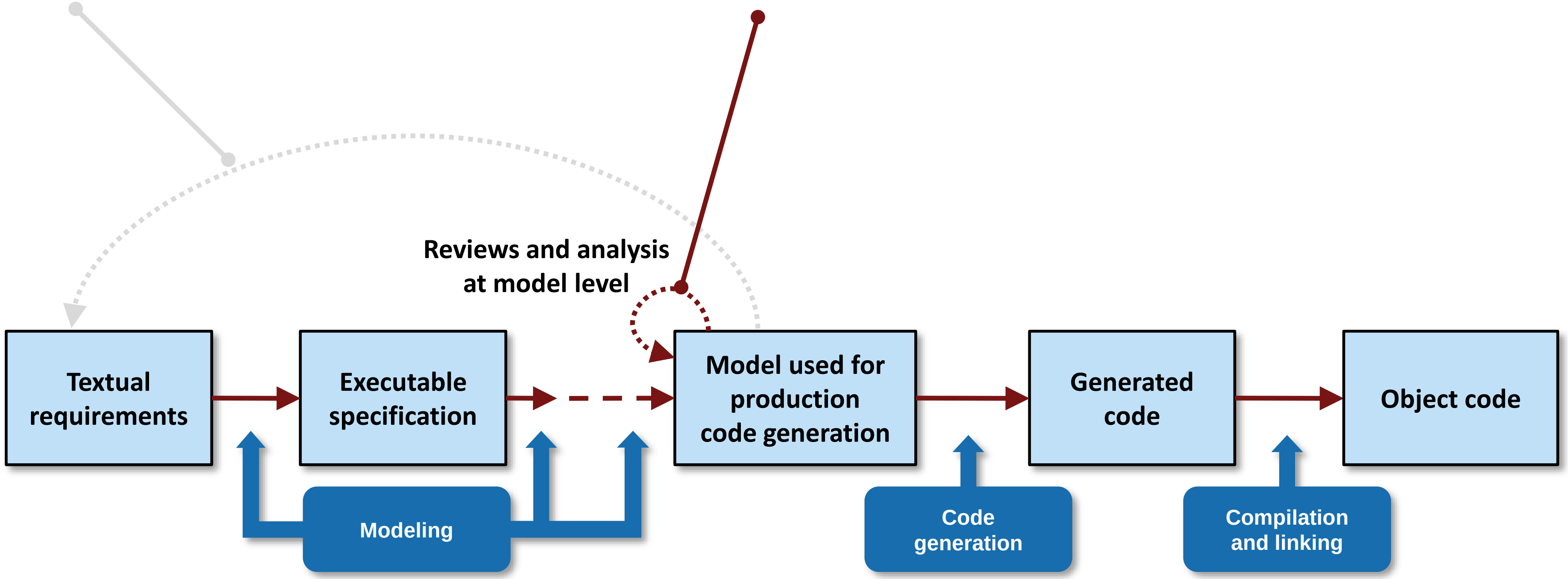


Detect Design Errors with Formal Methods

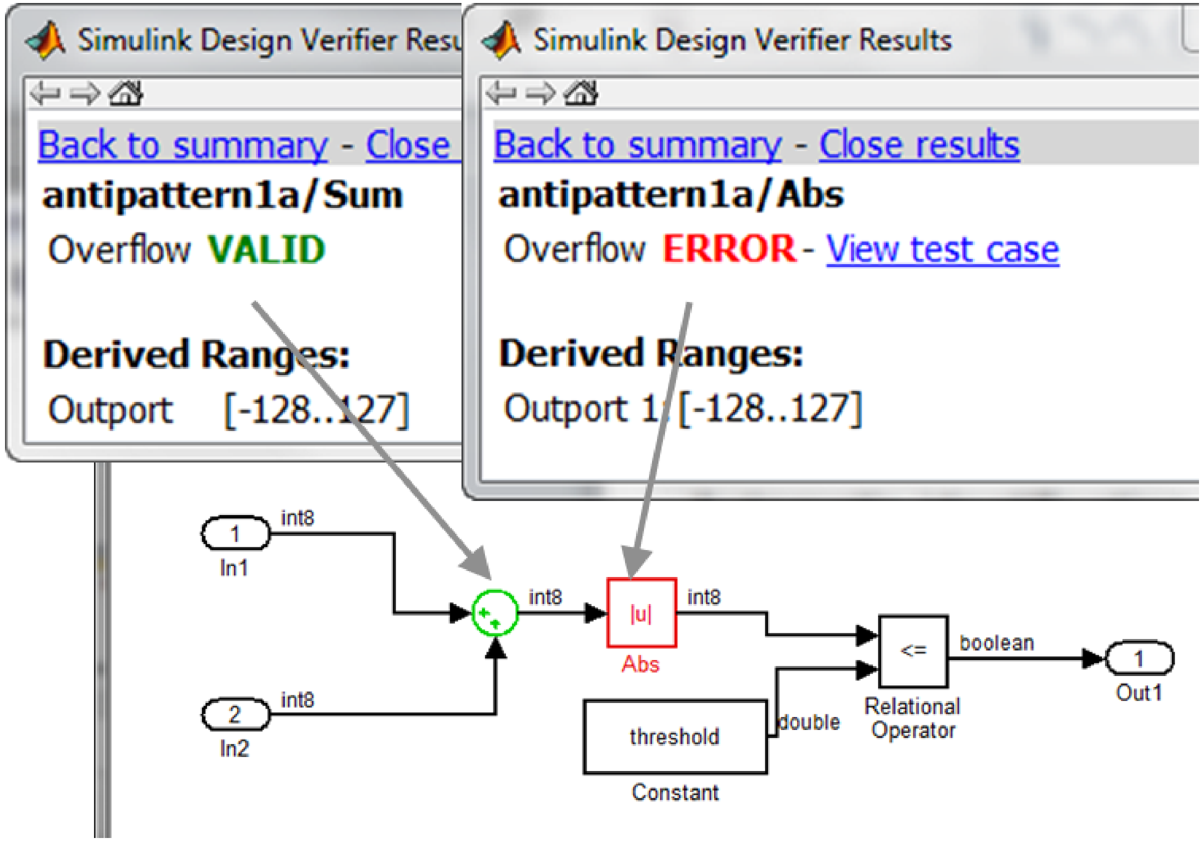
Model Verification

- Manage requirements
- Systematically test
- Measure model coverage

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



Detect Design Errors Using Formal Methods



Simulink Design Verifier Resu

Back to summary - Close

antipattern1a/Sum

Overflow **VALID**

Derived Ranges:

Outport [-128..127]

Simulink Design Verifier Results

Back to summary - Close results

antipattern1a/Abs

Overflow **ERROR** - View test case

Derived Ranges:

Outport 1: [-128..127]

- Find design errors
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations

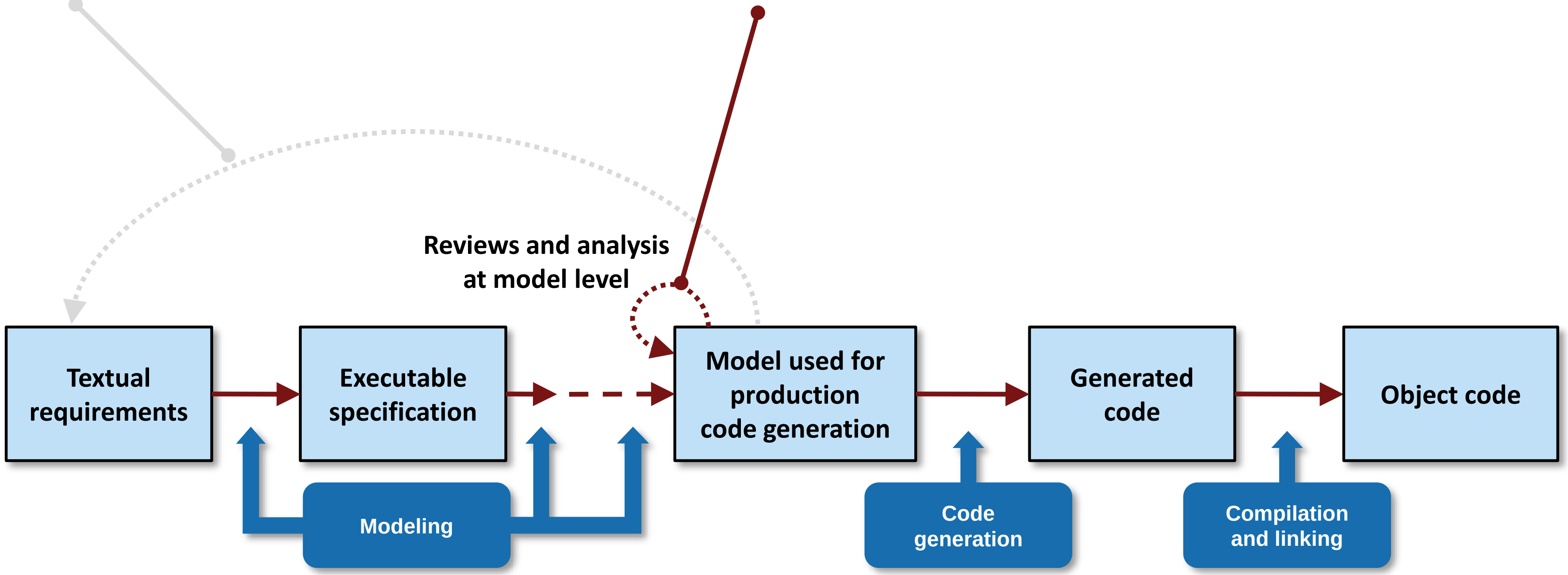
- Generate counter example to reproduce error

Prove Model Behavior Compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage

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

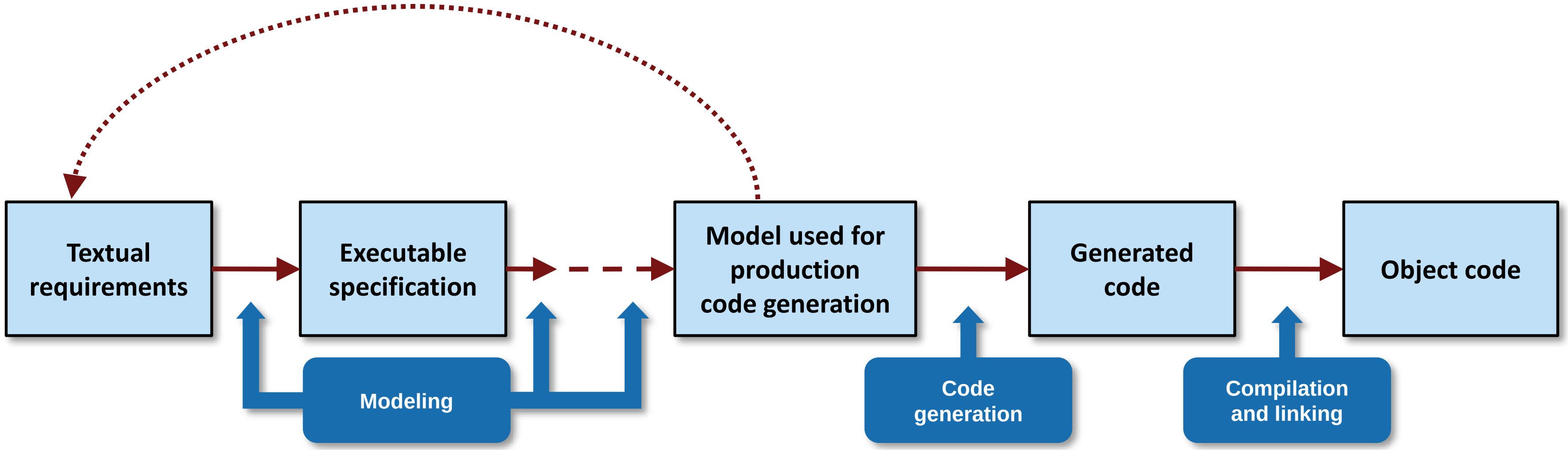


Proving Model Meets Requirements

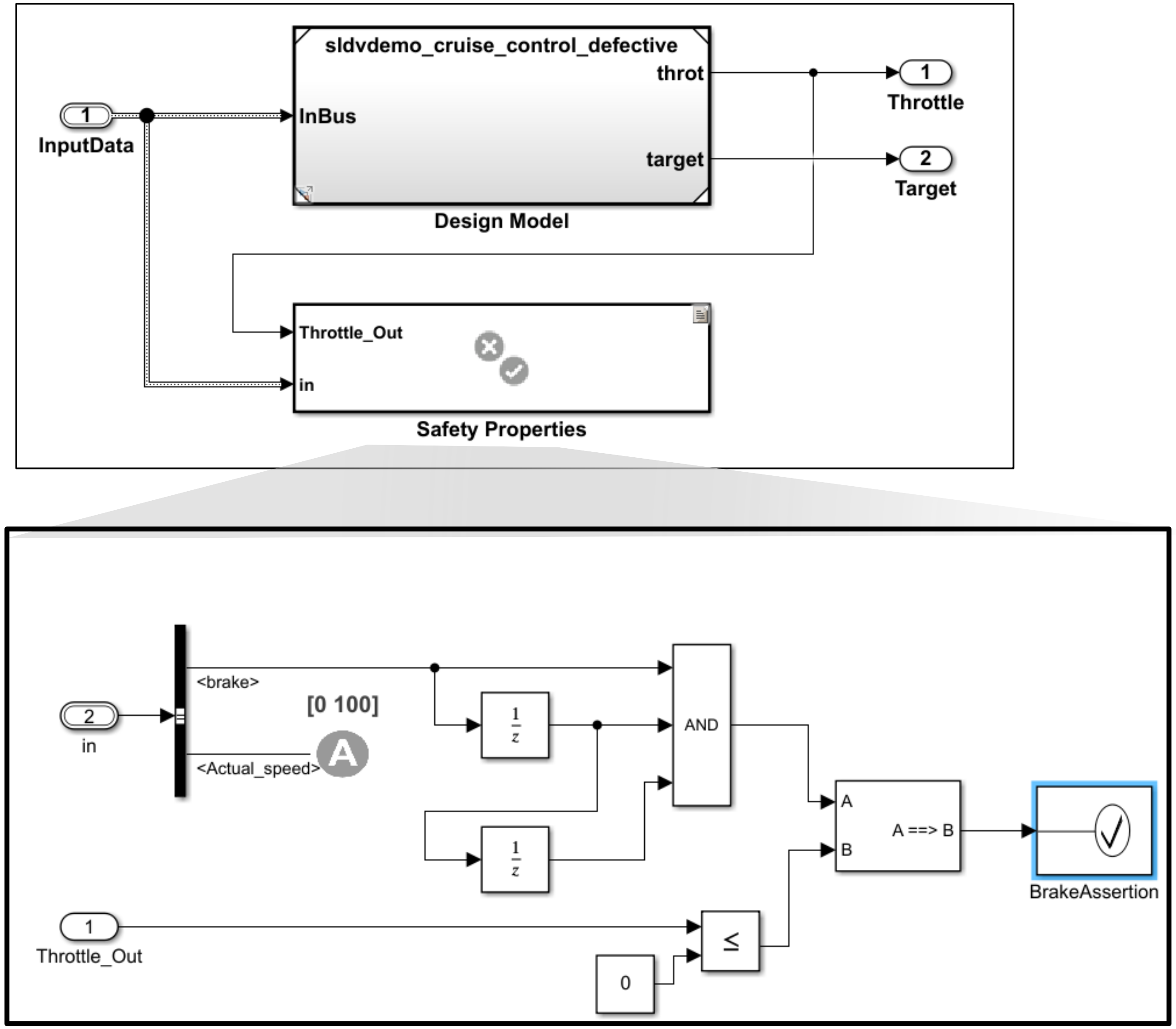
Safety Requirement:

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

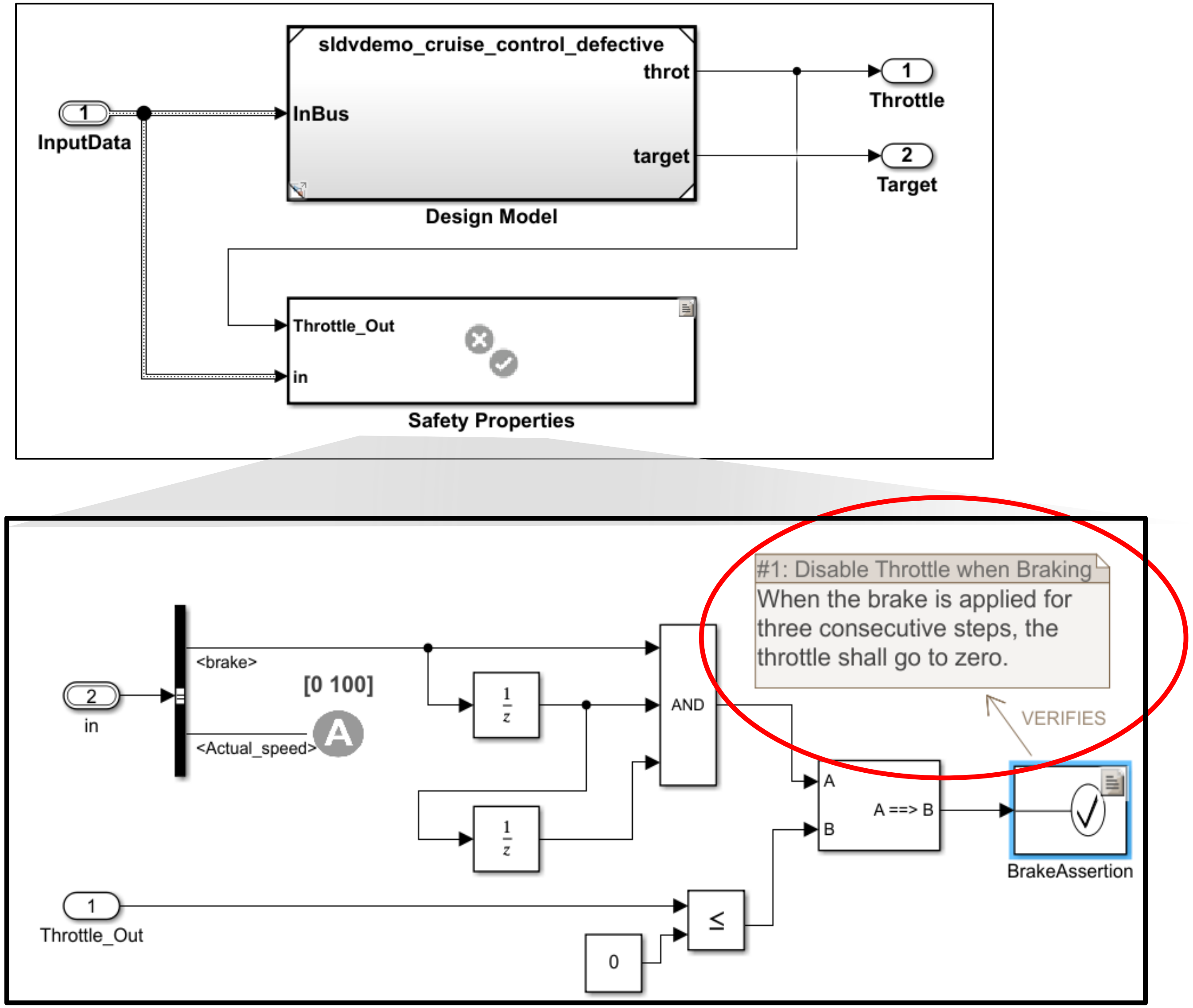
- Need to ensure the design performs correctly



Model functional and safety requirements



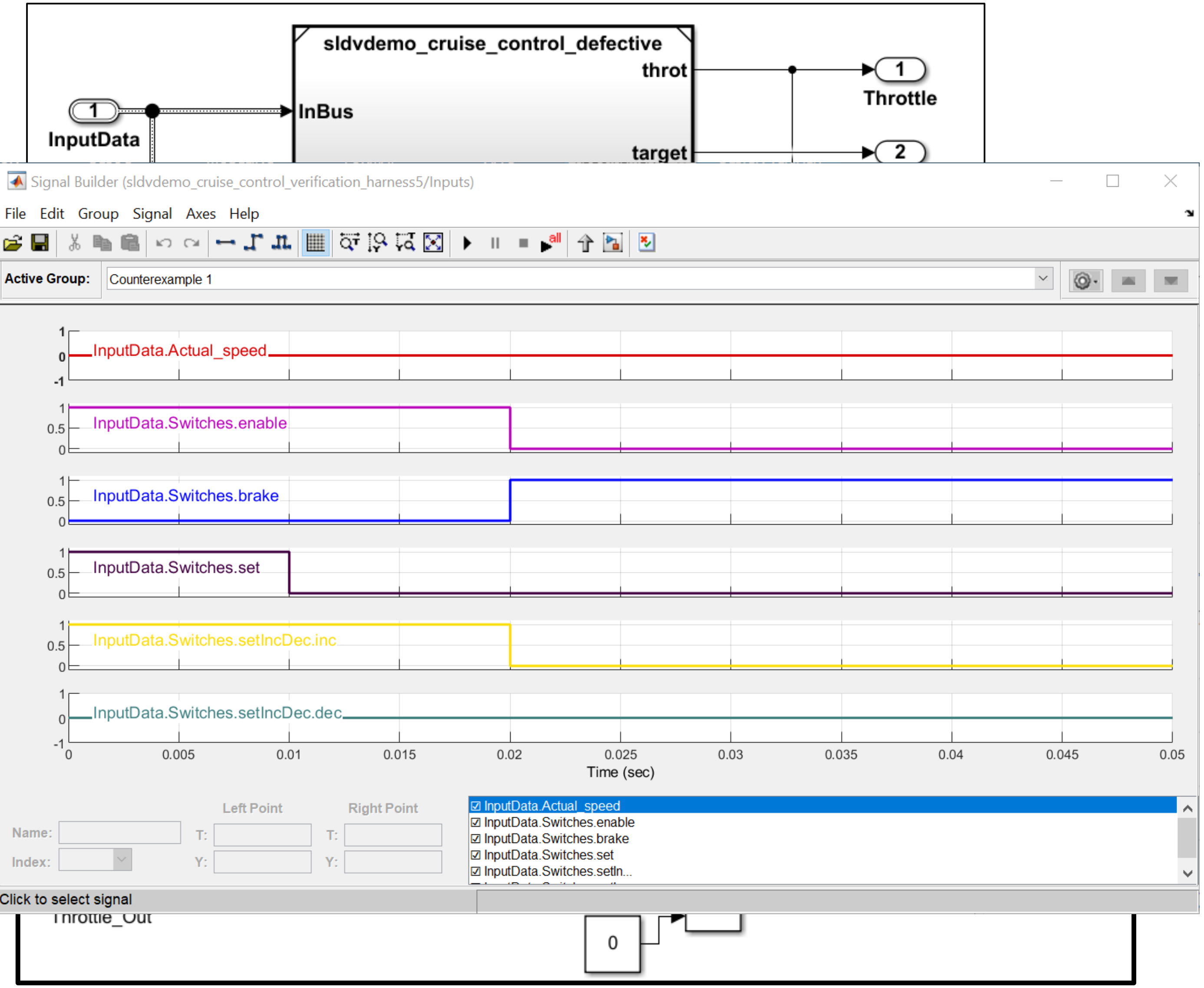
Link requirements to properties

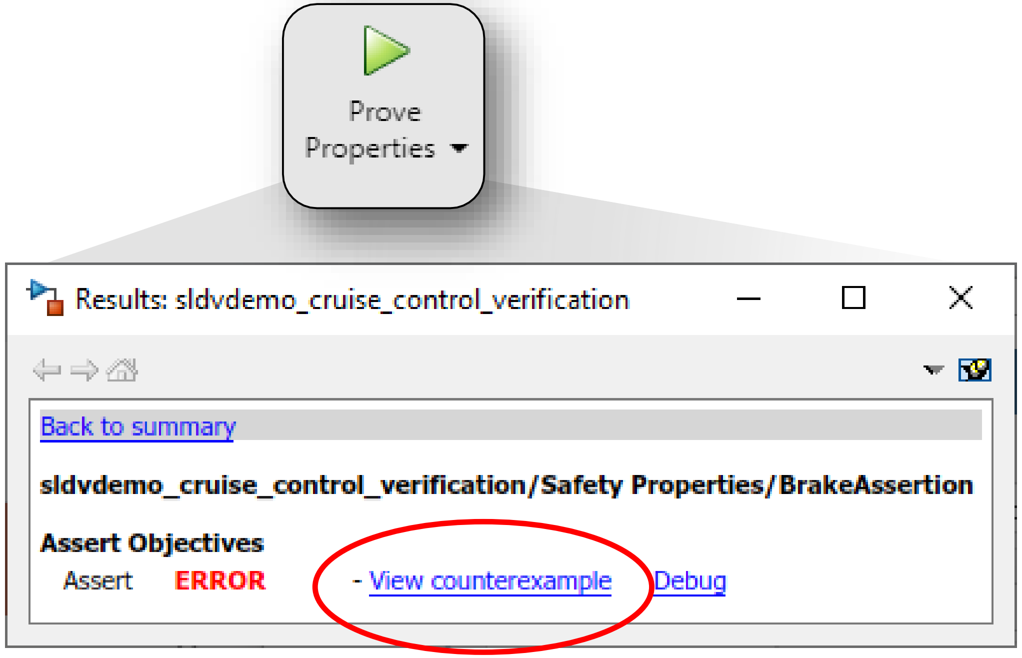


#1: Disable Throttle when Braking

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

Prove That Design Meets Requirements


sldvdemo_cruise_control_defective
throt
InBus
target
1
InputData
1
Throttle
2
Signal Builder (sldvdemo_cruise_control_verification_harness5/Inputs)
File Edit Group Signal Axes Help
Active Group: Counterexample 1
InputData.Actual_speed
InputData.Switches.enable
InputData.Switches.brake
InputData.Switches.set
InputData.Switches.setIncDec.inc
InputData.Switches.setIncDec.dec
1
0
-1
0.5
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
Time (sec)
Left Point
Right Point
Name:
Index:
T:
Y:
☑ InputData.Actual_speed
☑ InputData.Switches.enable
☑ InputData.Switches.brake
☑ InputData.Switches.set
☑ InputData.Switches.setIn...
Click to select signal
Throttle_Out
0



Results: sldvdemo_cruise_control_verification

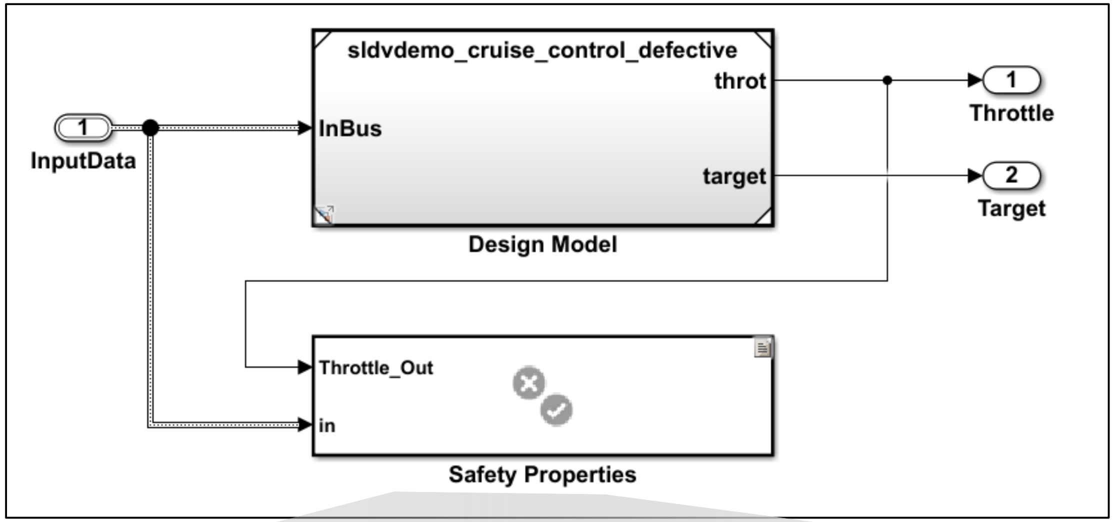
Back to summary

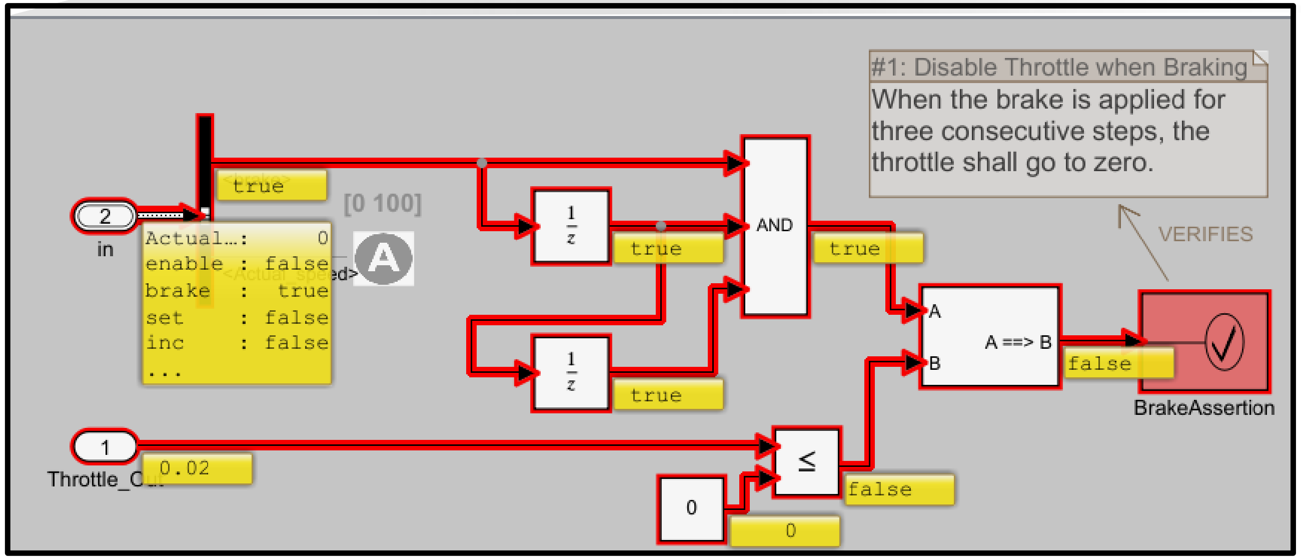
sldvdemo_cruise_control_verification/Safety Properties/BrakeAssertion

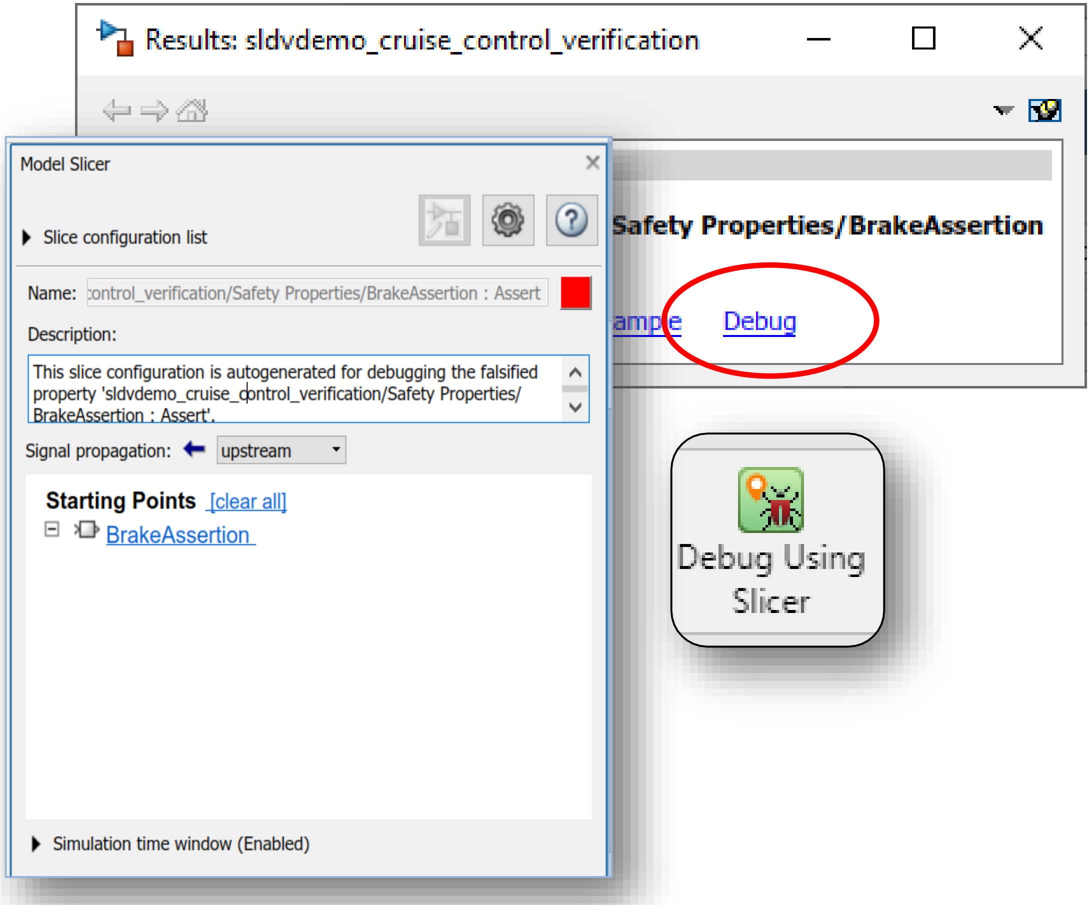
Assert Objectives

Assert **ERROR** - View counterexample Debug

Debugging Property Proving Violations






Results: sldvdemo_cruise_control_verification
Safety Properties/BrakeAssertion
ample
Debug
Model Slicer
Slice configuration list
Debug Using Slicer

Name: control_verification/Safety Properties/BrakeAssertion : Assert 🟥

Description: This slice configuration is autogenerated for debugging the falsified property 'sldvdemo_cruise_control_verification/Safety Properties/BrakeAssertion : Assert'.

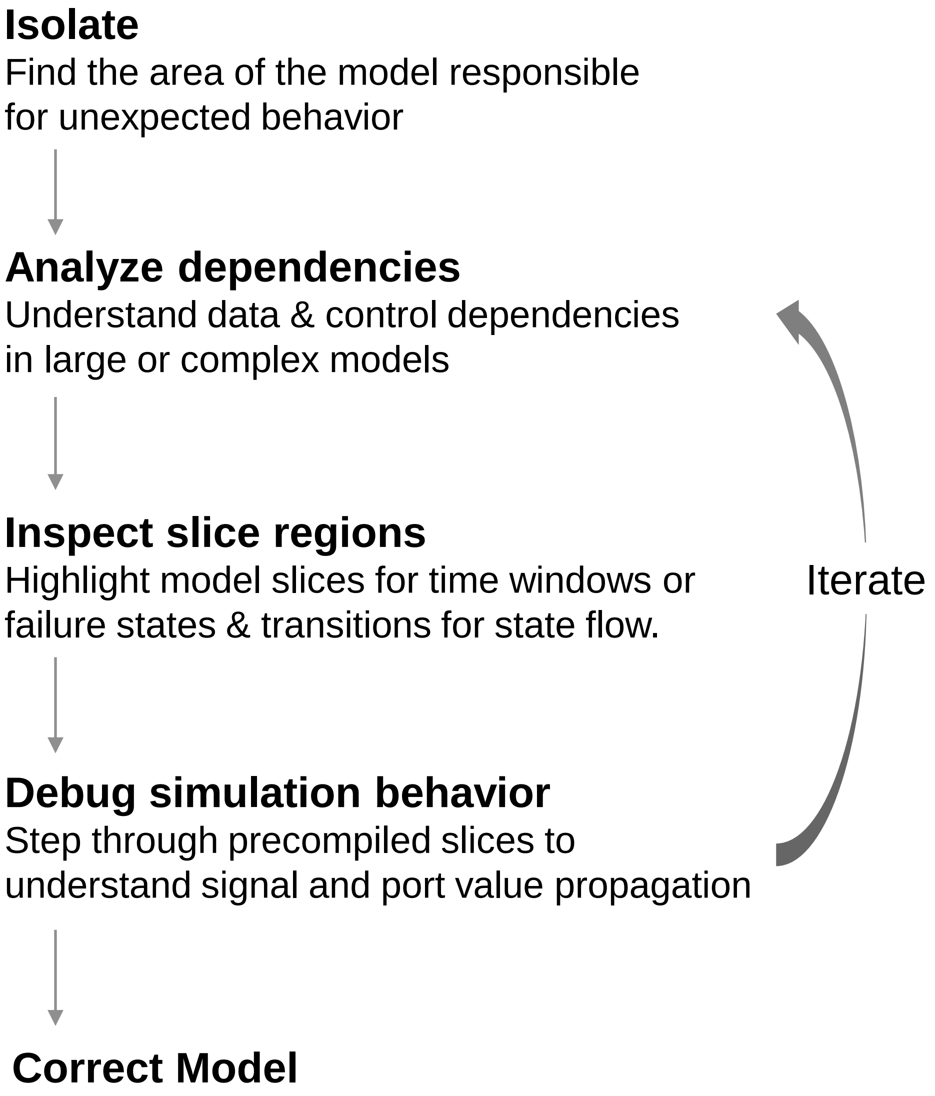
Signal propagation: upstream

Starting Points [clear all]

BrakeAssertion

Simulation time window (Enabled)

Resolve unexpected behavior in a model with Model Slicer



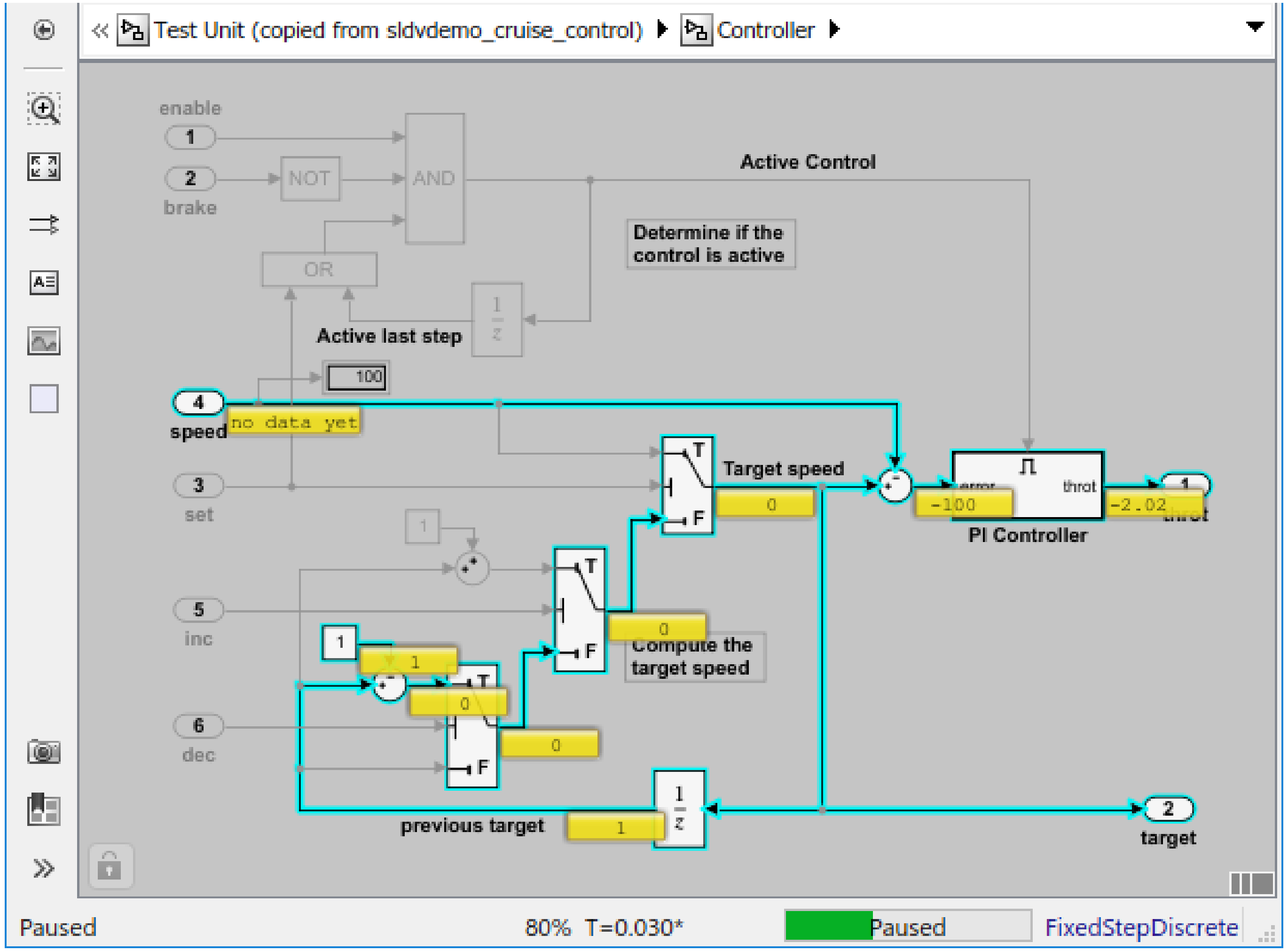
Isolate
Find the area of the model responsible for unexpected behavior

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

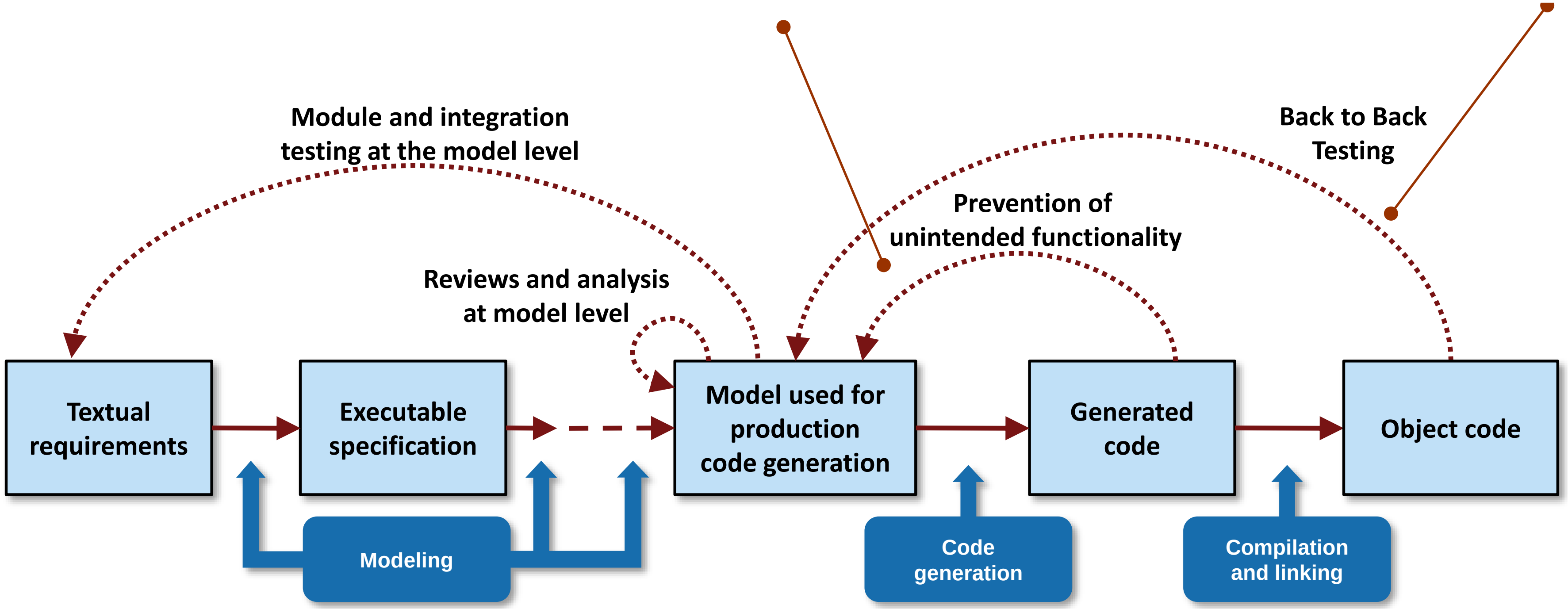


Code Verification: Gain Confidence in the Generated Code

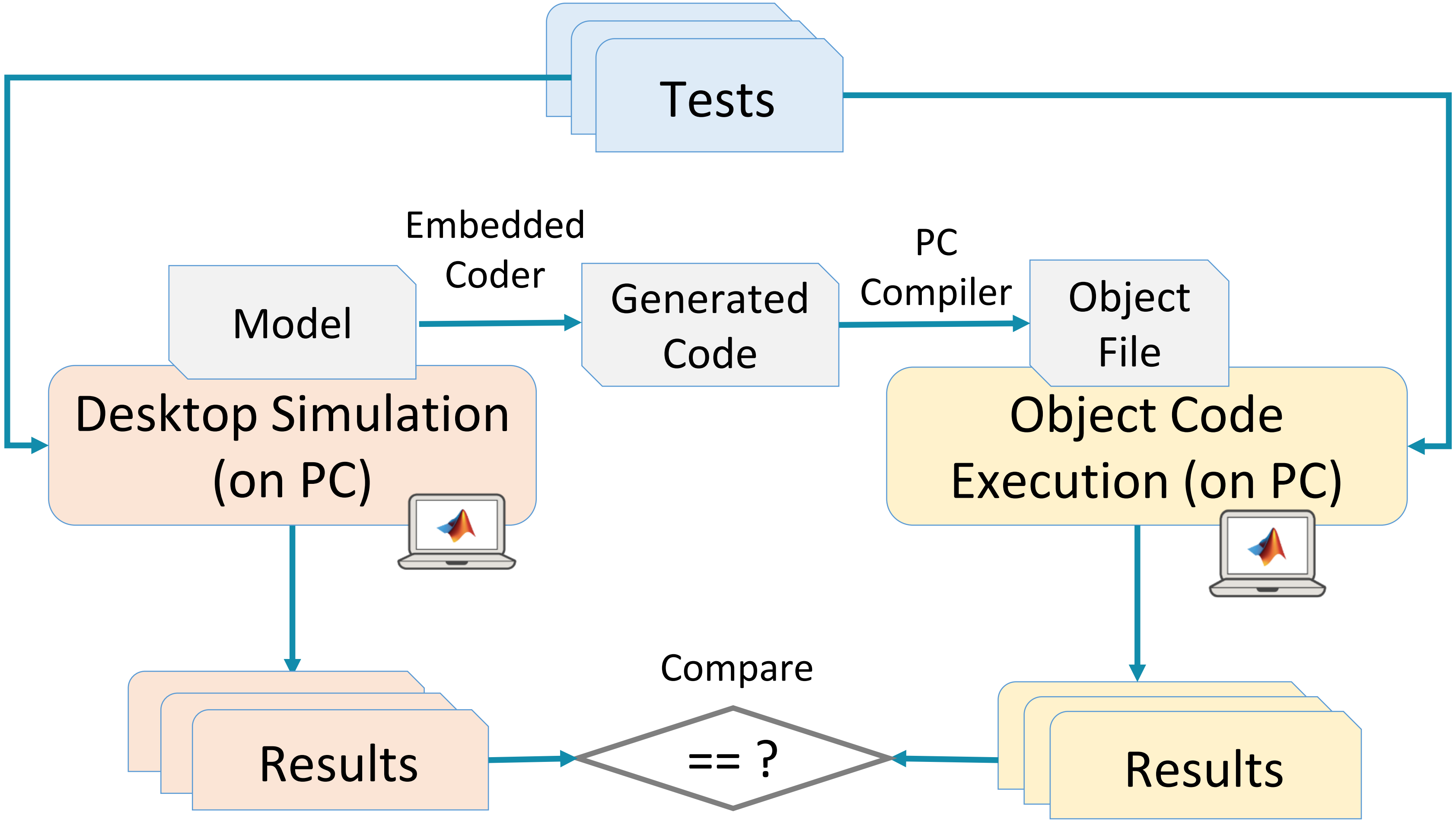
Code Verification

- Trace code to model and requirements
- Measure code coverage

- SIL/PIL equivalence testing
- Generate 100% coverage test vectors


Module and integration testing at the model level
Back to Back Testing
Prevention of unintended functionality
Reviews and analysis at model level
Textual requirements
Executable specification
Model used for production code generation
Generated code
Object code
Modeling
Code generation
Compilation and linking

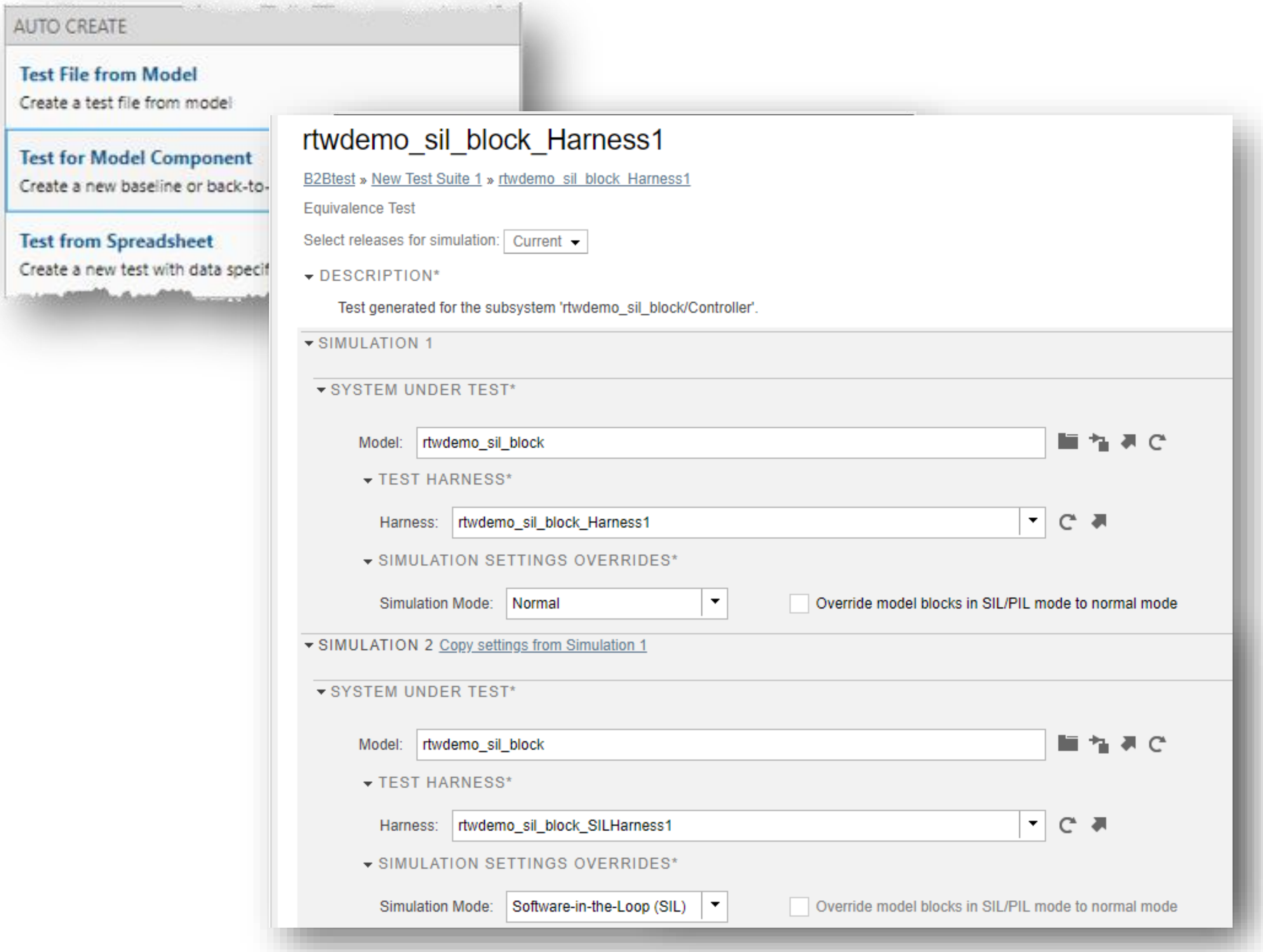
Back-to-Back Testing



- Automate SIL testing using Simulink Test

- Testing across releases

Automate Test Creation using Test Manager Wizard



AUTO CREATE

Test File from Model
Create a test file from model

Test for Model Component
Create a new baseline or back-to-

Test from Spreadsheet
Create a new test with data specif

rtwdemo_sil_block_Harness1

B2Btest » New Test Suite 1 » rtwdemo_sil_block_Harness1

Equivalence Test

Select releases for simulation: Current

DESCRIPTION*

Test generated for the subsystem 'rtwdemo_sil_block/Controller'.

SIMULATION 1

SYSTEM UNDER TEST*

Model: rtwdemo_sil_block

TEST HARNESS*

Harness: rtwdemo_sil_block_Harness1

SIMULATION SETTINGS OVERRIDES*

Simulation Mode: Normal

□ Override model blocks in SIL/PIL mode to normal mode

SIMULATION 2 Copy settings from Simulation 1

SYSTEM UNDER TEST*

Model: rtwdemo_sil_block

TEST HARNESS*

Harness: rtwdemo_sil_block_SILHarness1

SIMULATION SETTINGS OVERRIDES*

Simulation Mode: Software-in-the-Loop (SIL)

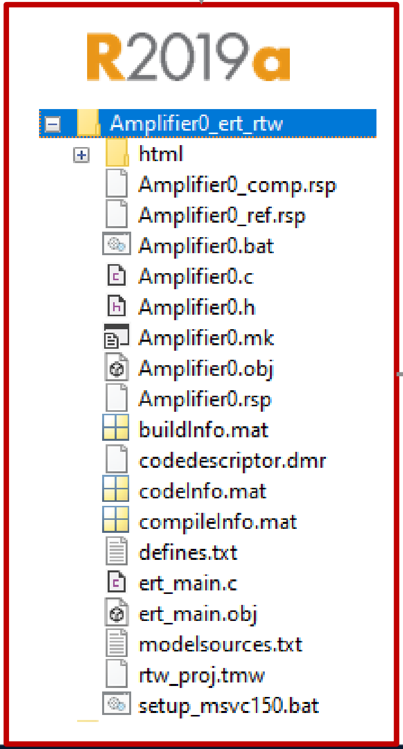
□ Override model blocks in SIL/PIL mode to normal mode

- 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

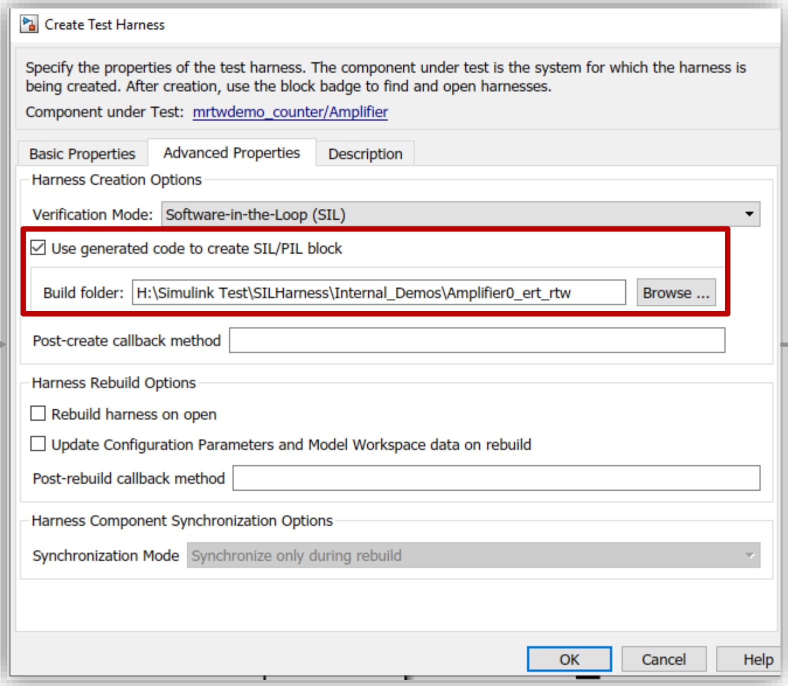
R2020a

- 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



R2019a

- Amplifier0_ert_rtw
  - html
  - Amplifier0_comp.rsp
  - Amplifier0_ref.rsp
  - Amplifier0.bat
  - Amplifier0.c
  - Amplifier0.h
  - Amplifier0.mk
  - Amplifier0.obj
  - Amplifier0.rsp
  - buildInfo.mat
  - codedescriptor.dmr
  - codeInfo.mat
  - compileInfo.mat
  - defines.txt
  - ert_main.c
  - ert_main.obj
  - modelsources.txt
  - rtw_proj.tmw
  - setup_msvc150.bat



Create Test Harness

Specify the properties of the test harness. The component under test is the system for which the harness is being created. After creation, use the block badge to find and open harnesses.

Component under Test: mrtwdemo_counter/Amplifier

Basic Properties | Advanced Properties | Description

Harness Creation Options

Verification Mode: Software-in-the-Loop (SIL)

☑ Use generated code to create SIL/PIL block

Build folder: H:\Simulink Test\SILHarness\Internal_Demos\Amplifier0_ert_rtw [Browse ...]

Post-create callback method

Harness Rebuild Options

☐ Rebuild harness on open

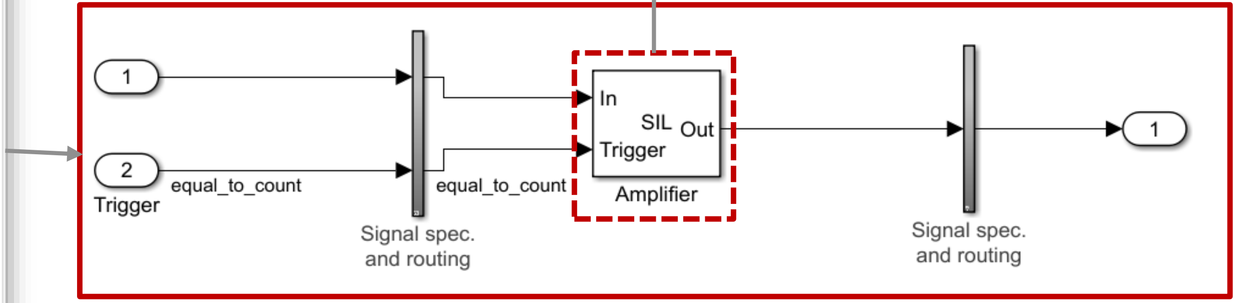
☐ Update Configuration Parameters and Model Workspace data on rebuild

Post-rebuild callback method

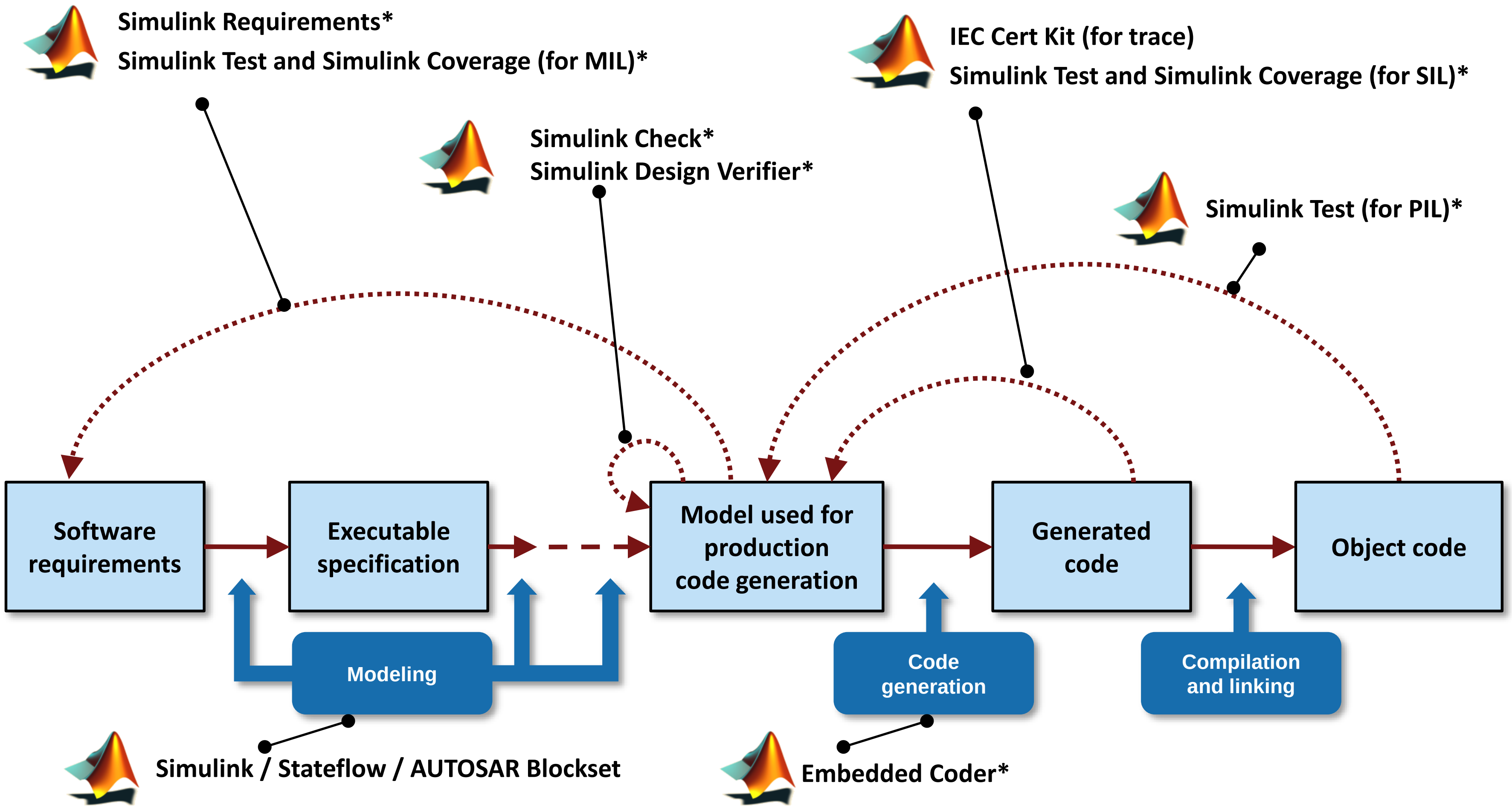
Harness Component Synchronization Options

Synchronization Mode: Synchronize only during rebuild

[OK] [Cancel] [Help]



Reference Workflow for Generated Code



*Qualifiable

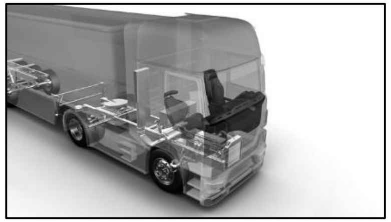
Customer References and Applications

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



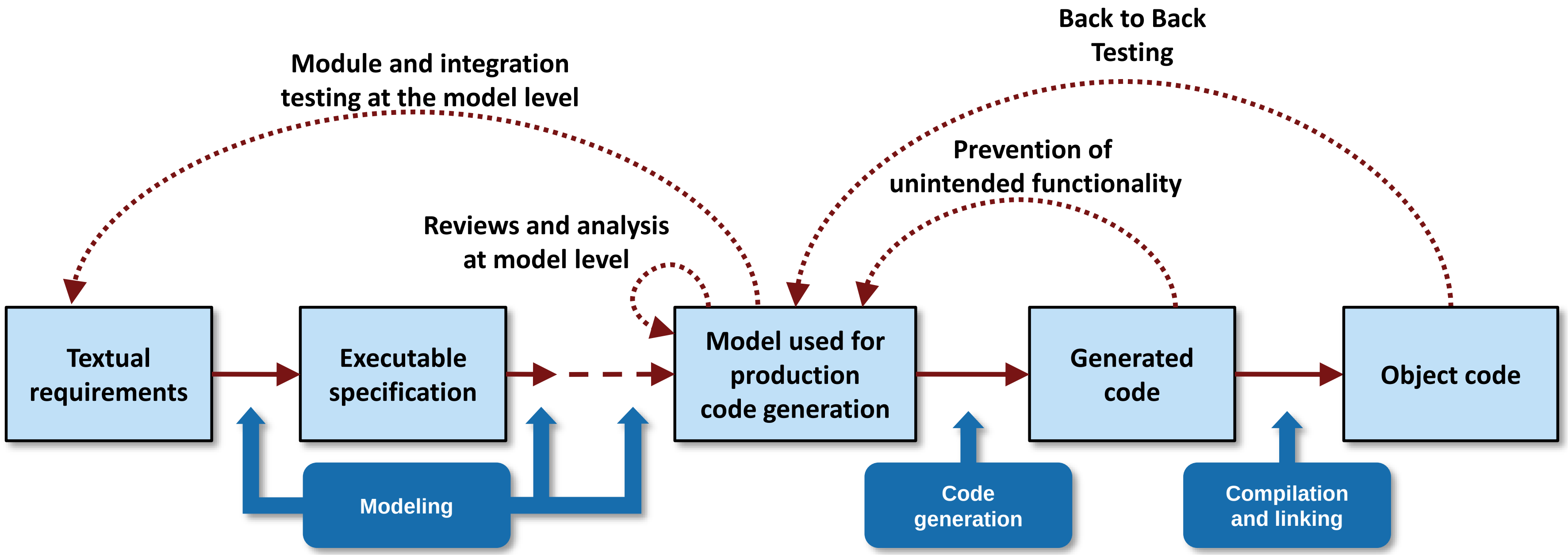
Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks

Verification time cut by up to 50 percent

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

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


Back to Back Testing
Module and integration testing at the model level
Prevention of unintended functionality
Reviews and analysis at model level
Textual requirements
Executable specification
Model used for production code generation
Generated code
Object code
Modeling
Code generation
Compilation and linking

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!