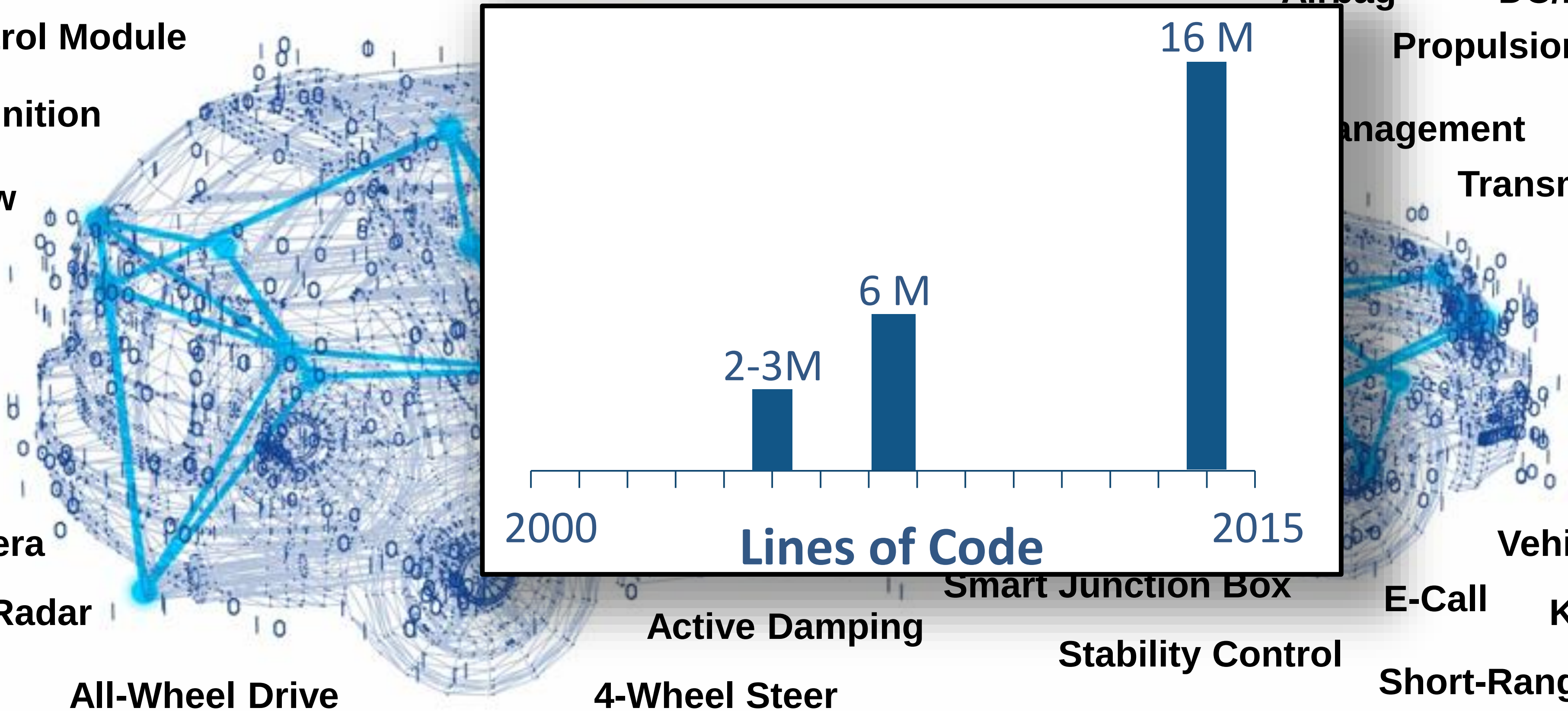
MATLAB EXPO 2018

Automating Best Practices to Improve Design Quality

Adam Whitmill, Senior Application Engineer

Growing Complexity of Embedded Systems



Emergency Braking

Stability Control

Automatic Parking

Battery Management

Instrument Panel

Infotainment

Adaptive Cruise Control

Smart Junction Box

Airbag

DC/DC Converter

Body Control Module

Propulsion Motor Control

Voice Recognition

anagement

Navigation

Power Window

Transmission Control

Vehicle-to-Infrastructure

Forward Camera

Adaptive Front Lighting

Power Liftgate

HVAC Control

Power Seat

Back-up Camera

Vehicle-to-Vehicle

Smart Junction Box

E-Call

Long-Range Radar

Active Damping

Keyless Entry

Stability Control

Short-Range Radar

All-Wheel Drive

4-Wheel Steer

Siemens, “Ford Motor Company Case Study,” Siemens PLM Software, 2014
McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDJNet, 2013

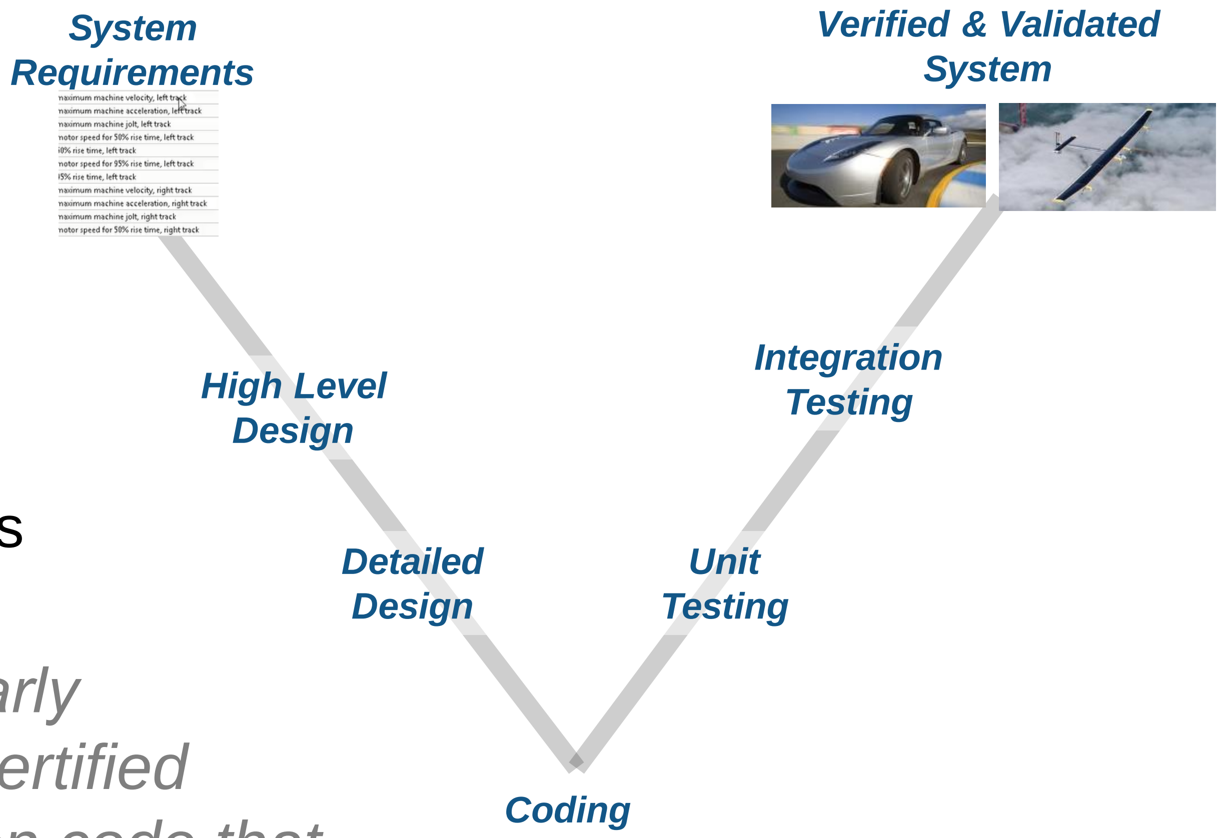
Why do 71% of Embedded Projects Fail?

Poor Requirements Management

Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005

Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate manual verification tasks
- Workflow that conforms to safety standards



naximum machine velocity, left track
naximum machine acceleration, left track
naximum machine jolt, left track
notor speed for 50% rise time, left track
i0% rise time, left track
notor speed for 95% rise time, left track
)5% rise time, left track
naximum machine velocity, right track
naximum machine acceleration, right track
naximum machine jolt, right track
notor speed for 50% rise time, right track

“Reduce costs and project risk through early verification, shorten time to market on a certified system, and deliver high-quality production code that was first-time right”   Michael Schwarz, ITK Engineering

Lear Delivers Quality Body Control Electronics Faster Using Model-Based Design

Challenge

Design, verify, and implement high-quality automotive body control electronics

Solution

Use Model-Based Design to enable early and continuous verification via simulation, SIL, and HIL testing

Results

- Requirements validated early. Over 95% of issues fixed before implementation, versus 30% previously
- Development time cut by 40%. 700,000 lines of code generated and test cases reused throughout the development cycle
- Zero warranty issues reported

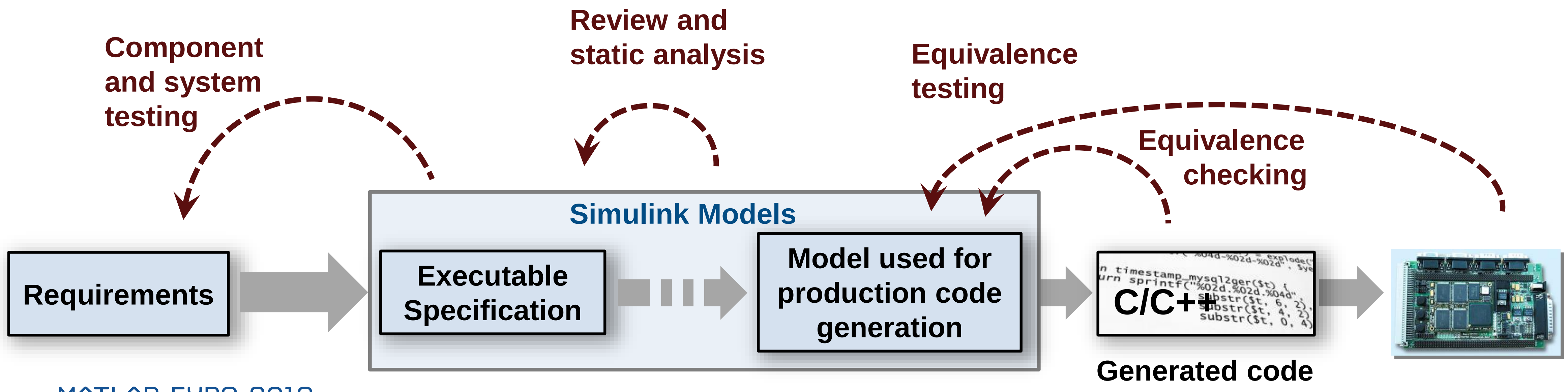
Lear automotive body electronic control unit.

> *"We adopted Model-Based Design not only to deliver better-quality systems faster, but because we believe it is a smart choice. Recently we won a project that several of our competitors declined to bid on because of its tight time constraints. Using Model-Based Design, we met the original delivery date with no problem."*
>
> *- Jason Bauman, Lear Corporation*

[Link to user story]

Model Based Design Verification Workflow

1. Develop functions, perform ad-hoc testing, implement traceability
2. Refine design, Validate and Verify
3. Automatically detect quality issues and run-time error
4. Generate Code & Deploy
5. Auto-execute functional tests, verify product vs specification & Auto-report



Component and system testing

Review and static analysis

Equivalence testing

Equivalence checking

Simulink Models

Requirements

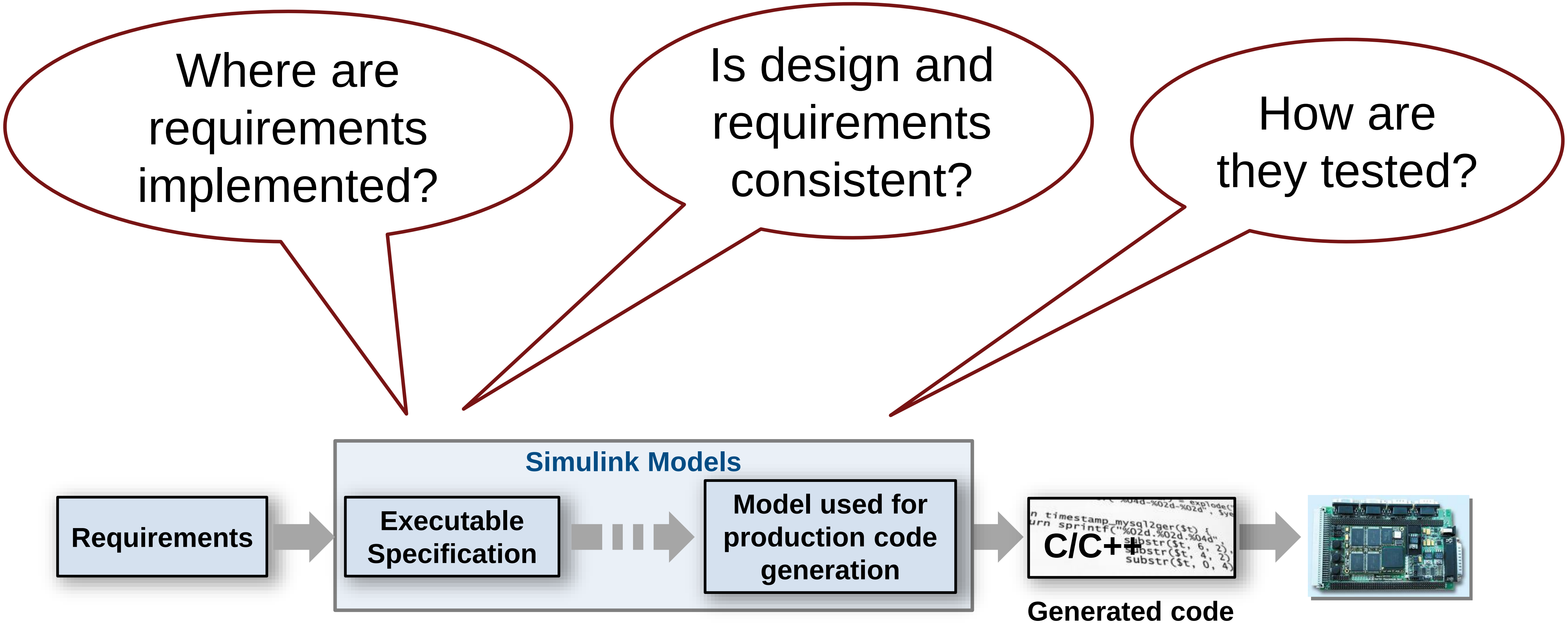
Executable Specification

Model used for production code generation

C/C++

Generated code

Challenges with Requirements



Where are requirements implemented?

Is design and requirements consistent?

How are they tested?

Simulink Models

Requirements

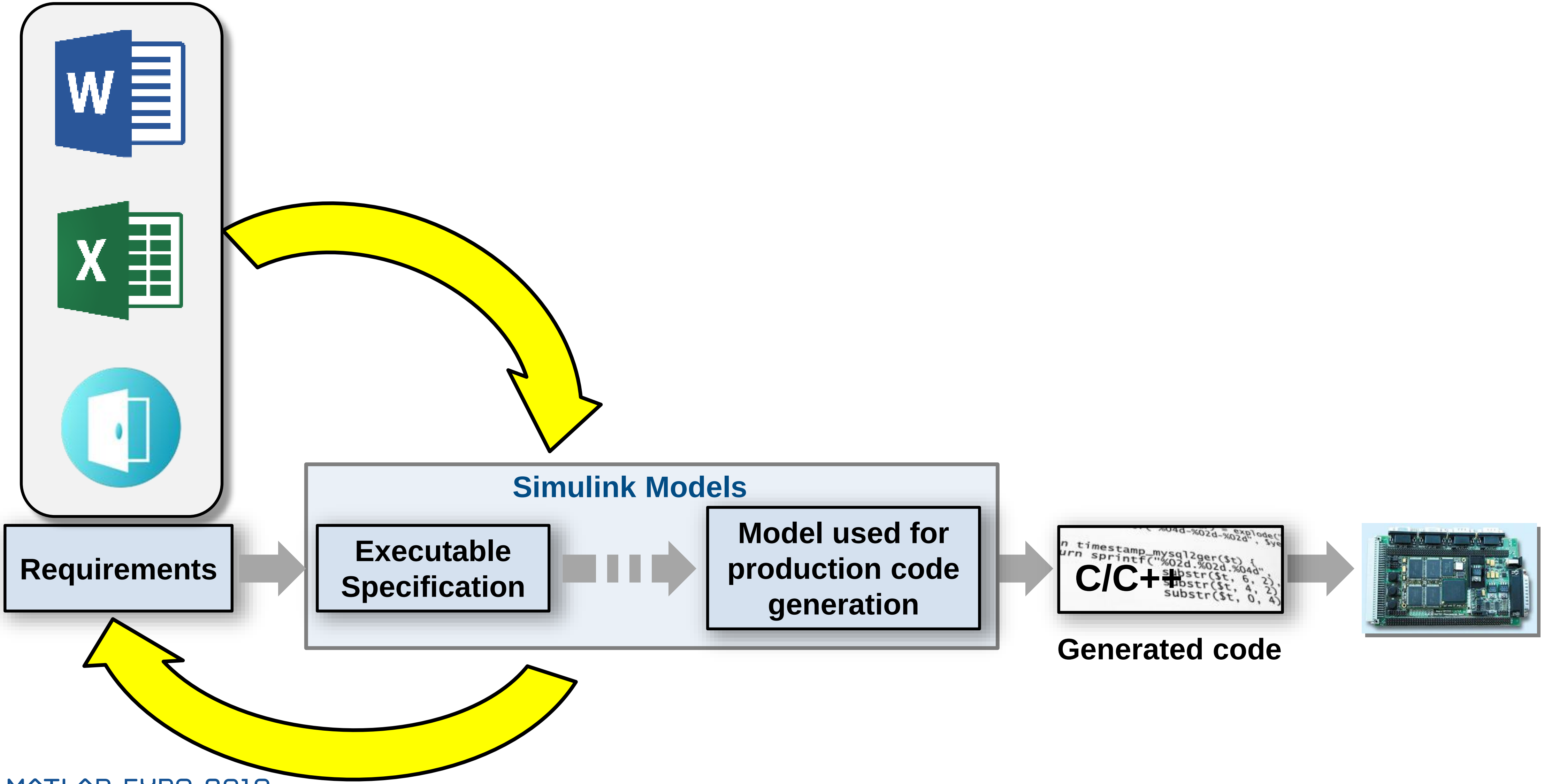
Executable Specification

Model used for production code generation

C/C++

Generated code

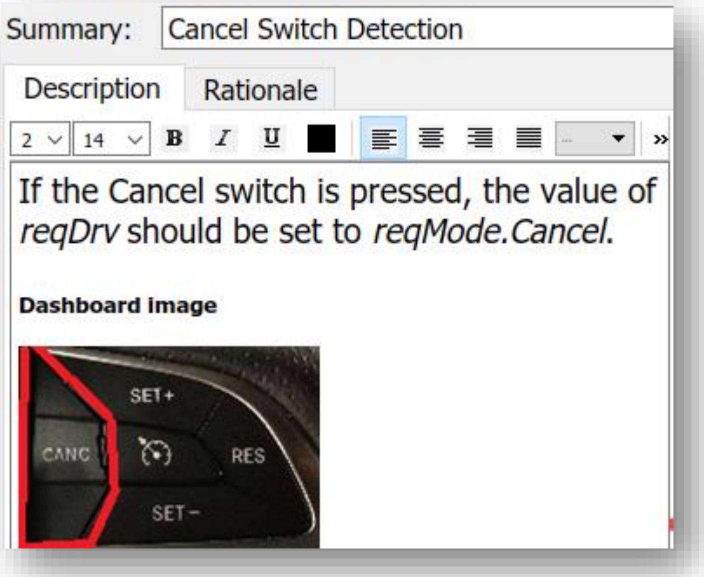
Gap Between Requirements and Design



Simulink Requirements

R2017b

Author

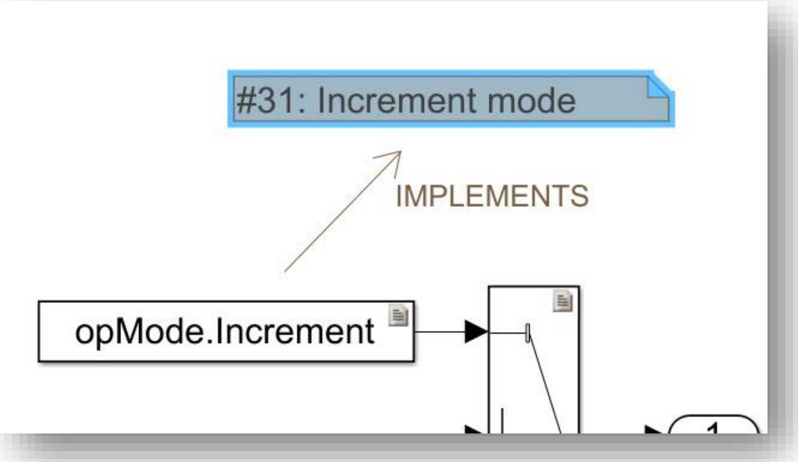


Summary: Cancel Switch Detection

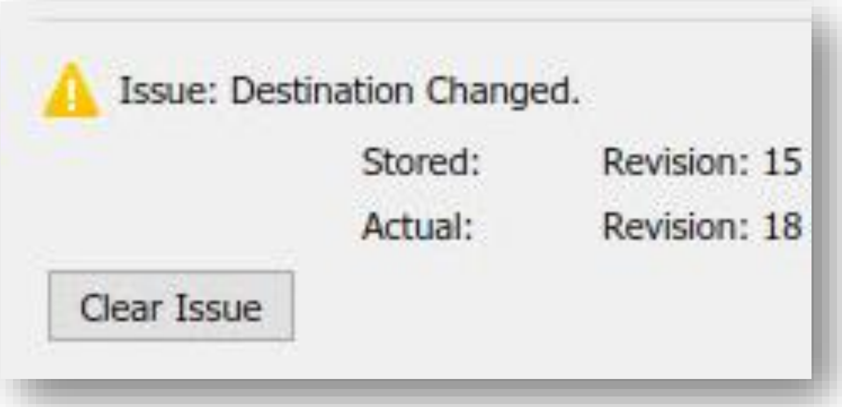
If the Cancel switch is pressed, the value of *reqDrv* should be set to *reqMode.Cancel*.

Dashboard image

Track



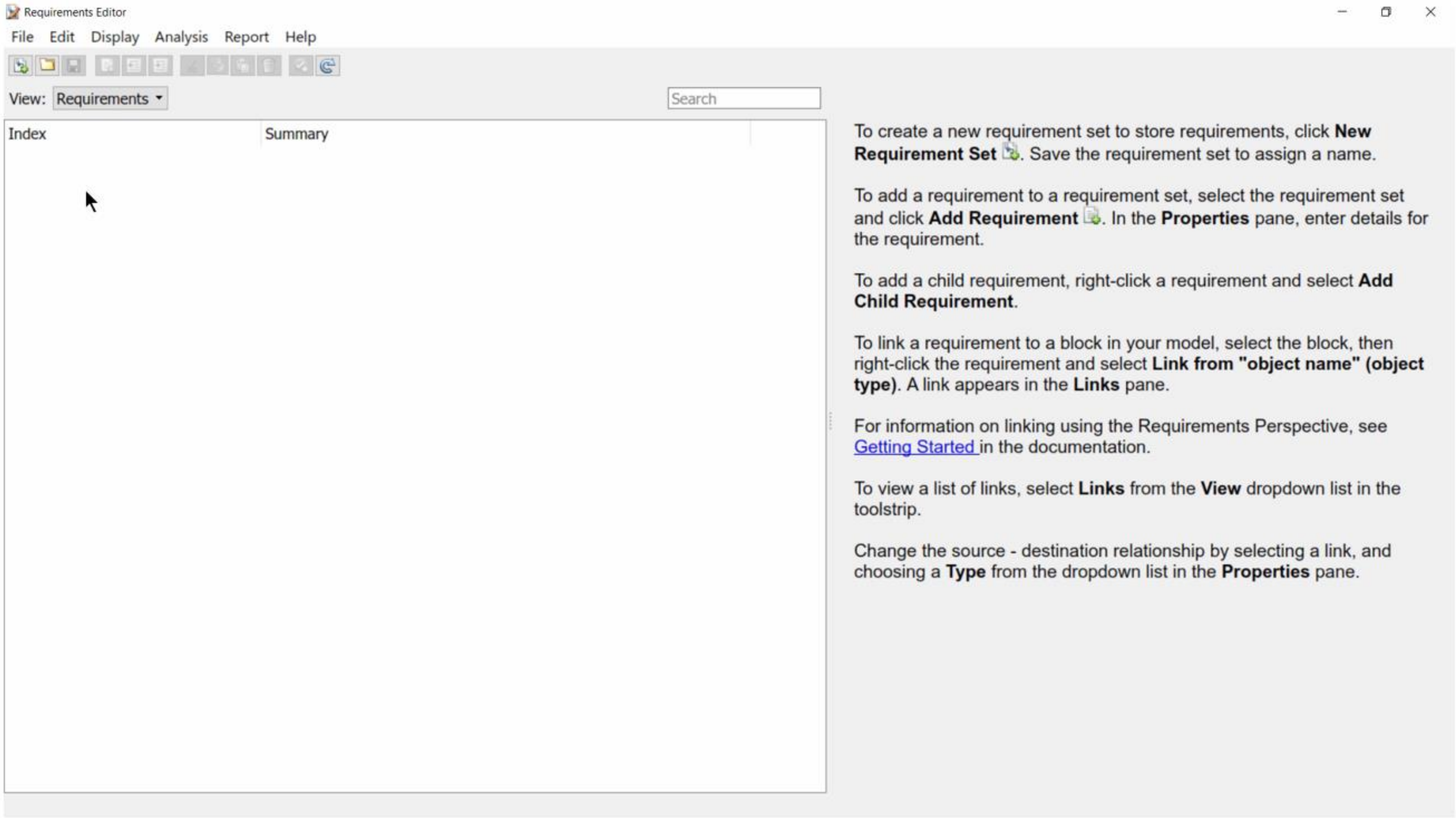
Manage



Issue: Destination Changed.

Stored:	Revision: 15
Actual:	Revision: 18

Requirements Editor



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

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

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

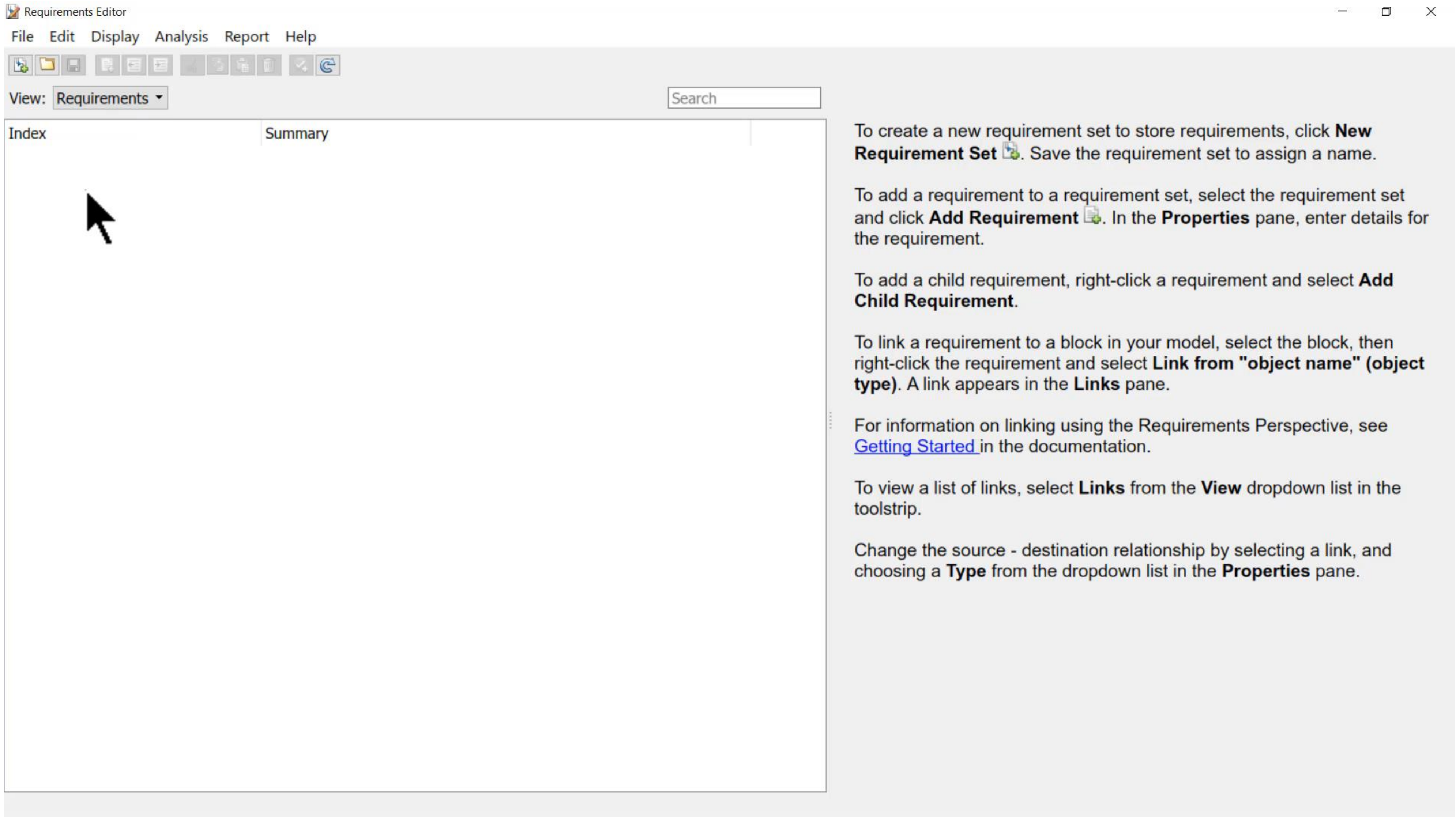
To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see Getting Started in the documentation.

To view a list of links, select **Links** from the **View** dropdown list in the toolstrip.

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

Requirements Editor



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

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

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

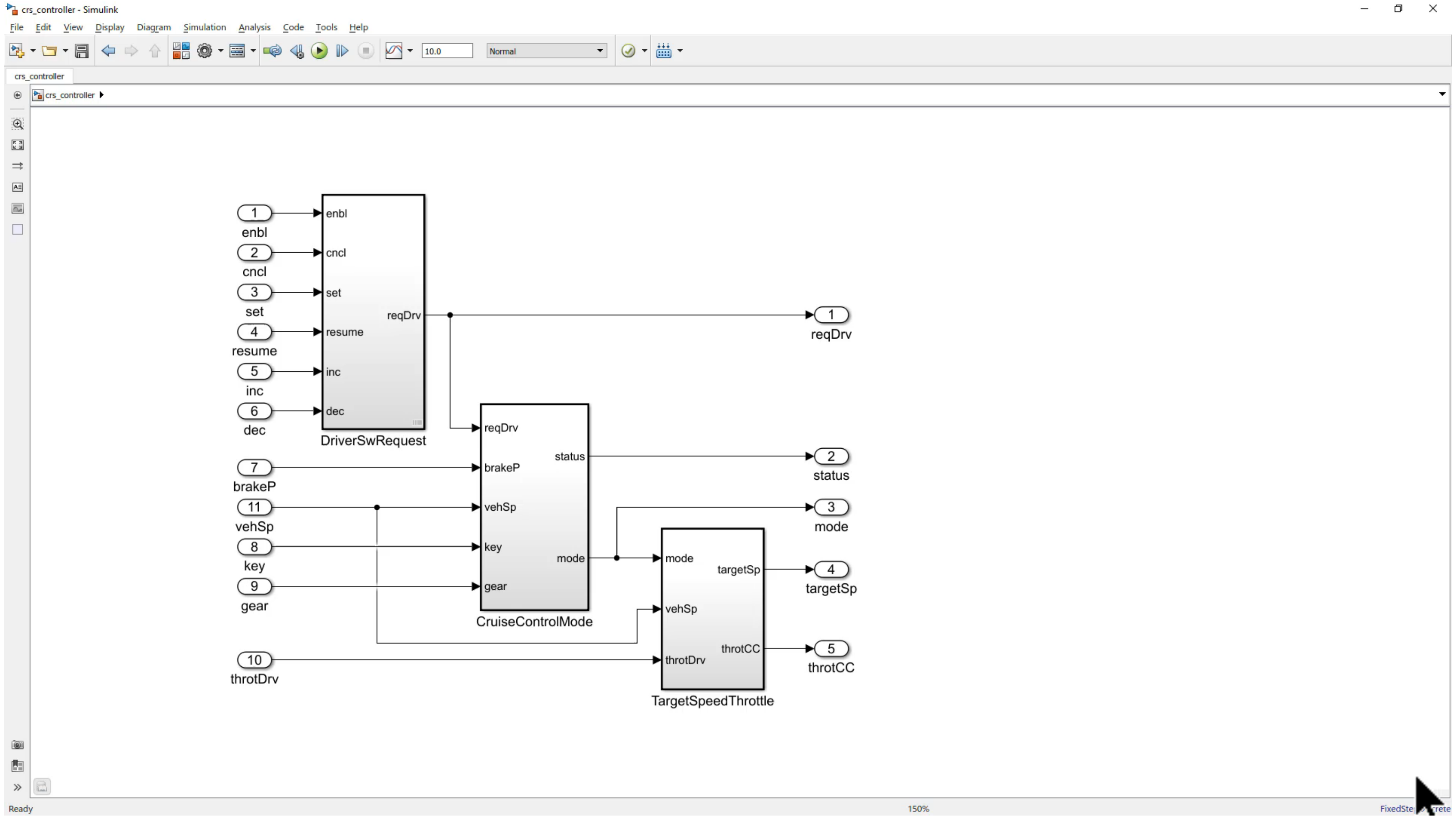
To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see Getting Started in the documentation.

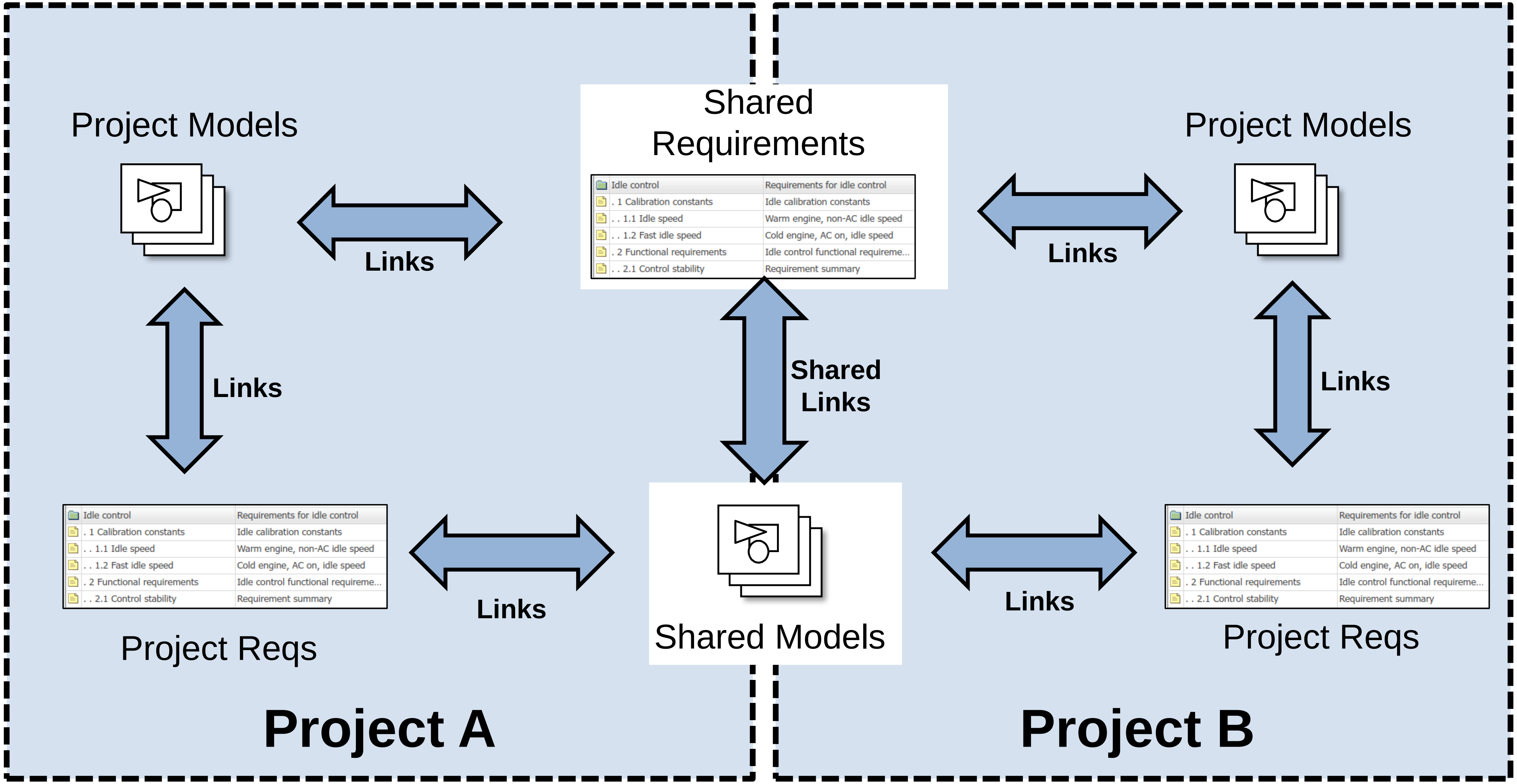
To view a list of links, select **Links** from the **View** dropdown list in the toolstrip.

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

Requirements Perspective



Requirement Reuse across Projects



Shared Requirements

Idle control	Requirements for idle control
. 1 Calibration constants	Idle calibration constants
. . 1.1 Idle speed	Warm engine, non-AC idle speed
. . 1.2 Fast idle speed	Cold engine, AC on, idle speed
. 2 Functional requirements	Idle control functional requireme...
. . 2.1 Control stability	Requirement summary

Project Reqs (Project A)

Idle control	Requirements for idle control
. 1 Calibration constants	Idle calibration constants
. . 1.1 Idle speed	Warm engine, non-AC idle speed
. . 1.2 Fast idle speed	Cold engine, AC on, idle speed
. 2 Functional requirements	Idle control functional requireme...
. . 2.1 Control stability	Requirement summary

Project Reqs (Project B)

Idle control	Requirements for idle control
. 1 Calibration constants	Idle calibration constants
. . 1.1 Idle speed	Warm engine, non-AC idle speed
. . 1.2 Fast idle speed	Cold engine, AC on, idle speed
. 2 Functional requirements	Idle control functional requireme...
. . 2.1 Control stability	Requirement summary

Import Requirements from External Sources

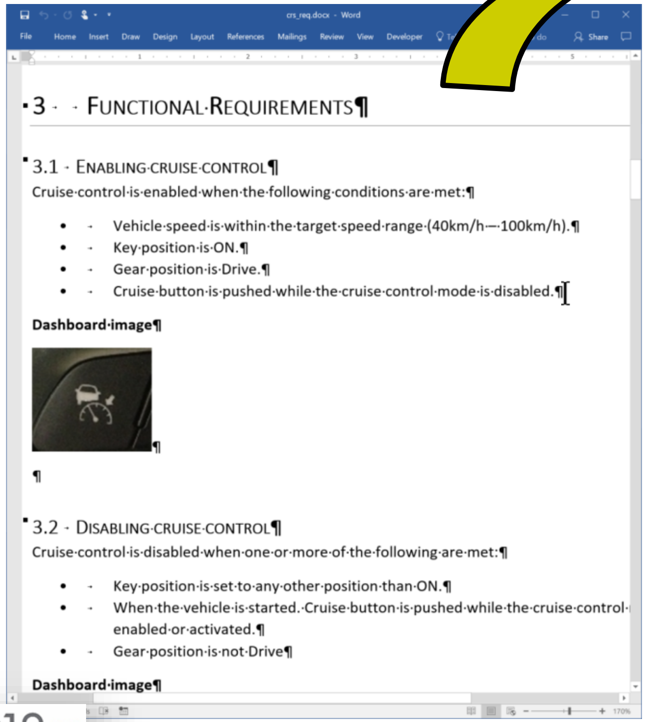
IBM Rational DOORS

R2018a

Microsoft Word

Import

Simulink Requirements Editor



crs_req.docx - Word

3 FUNCTIONAL REQUIREMENTS

3.1 ENABLING CRUISE CONTROL

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h — 100km/h).
- Key position is ON.
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

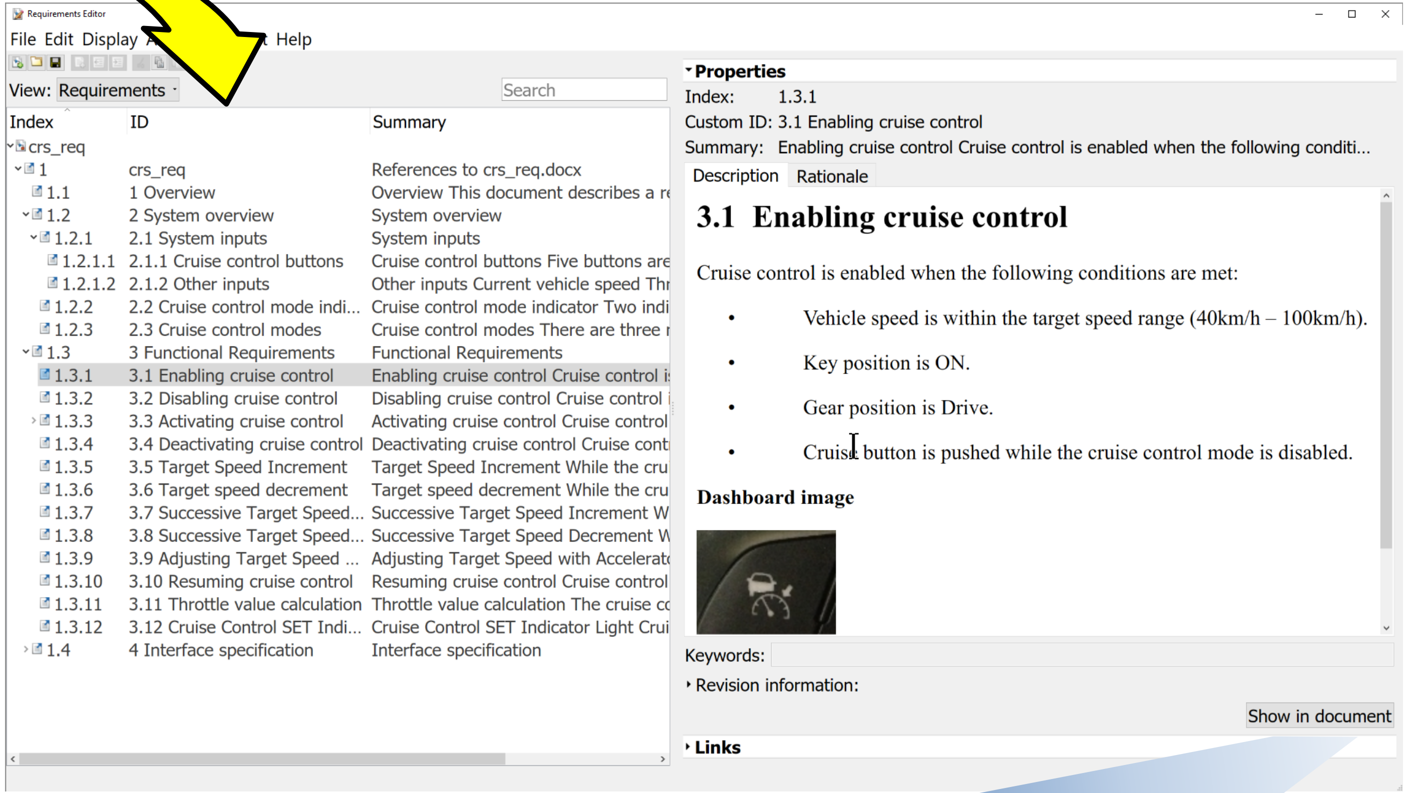
Dashboard image

3.2 DISABLING CRUISE CONTROL

Cruise control is disabled when one or more of the following are met:

- Key position is set to any other position than ON.
- When the vehicle is started. Cruise button is pushed while the cruise control enabled or activated.
- Gear position is not Drive

Dashboard image



Requirements Editor

View: Requirements

Index	ID	Summary
crs_req		
1	crs_req	References to crs_req.docx
1.1	1 Overview	Overview This document describes a re
1.2	2 System overview	System overview
1.2.1	2.1 System inputs	System inputs
1.2.1.1	2.1.1 Cruise control buttons	Cruise control buttons Five buttons are
1.2.1.2	2.1.2 Other inputs	Other inputs Current vehicle speed Thr
1.2.2	2.2 Cruise control mode indi...	Cruise control mode indicator Two indi
1.2.3	2.3 Cruise control modes	Cruise control modes There are three r
1.3	3 Functional Requirements	Functional Requirements
1.3.1	3.1 Enabling cruise control	Enabling cruise control Cruise control is
1.3.2	3.2 Disabling cruise control	Disabling cruise control Cruise control i
1.3.3	3.3 Activating cruise control	Activating cruise control Cruise control
1.3.4	3.4 Deactivating cruise control	Deactivating cruise control Cruise contr
1.3.5	3.5 Target Speed Increment	Target Speed Increment While the cru
1.3.6	3.6 Target speed decrement	Target speed decrement While the cru
1.3.7	3.7 Successive Target Speed...	Successive Target Speed Increment W
1.3.8	3.8 Successive Target Speed...	Successive Target Speed Decrement W
1.3.9	3.9 Adjusting Target Speed ...	Adjusting Target Speed with Accelerat
1.3.10	3.10 Resuming cruise control	Resuming cruise control Cruise control
1.3.11	3.11 Throttle value calculation	Throttle value calculation The cruise cc
1.3.12	3.12 Cruise Control SET Indi...	Cruise Control SET Indicator Light Crui
1.4	4 Interface specification	Interface specification

Properties

Index: 1.3.1

Custom ID: 3.1 Enabling cruise control

Summary: Enabling cruise control Cruise control is enabled when the following conditi...

3.1 Enabling cruise control

Cruise control is enabled when the following conditions are met:

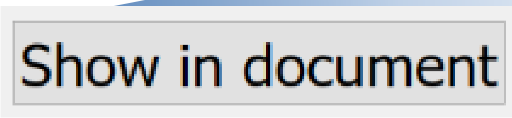
- Vehicle speed is within the target speed range (40km/h – 100km/h).
- Key position is ON.
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

Dashboard image

Keywords:

Revision information:

Links



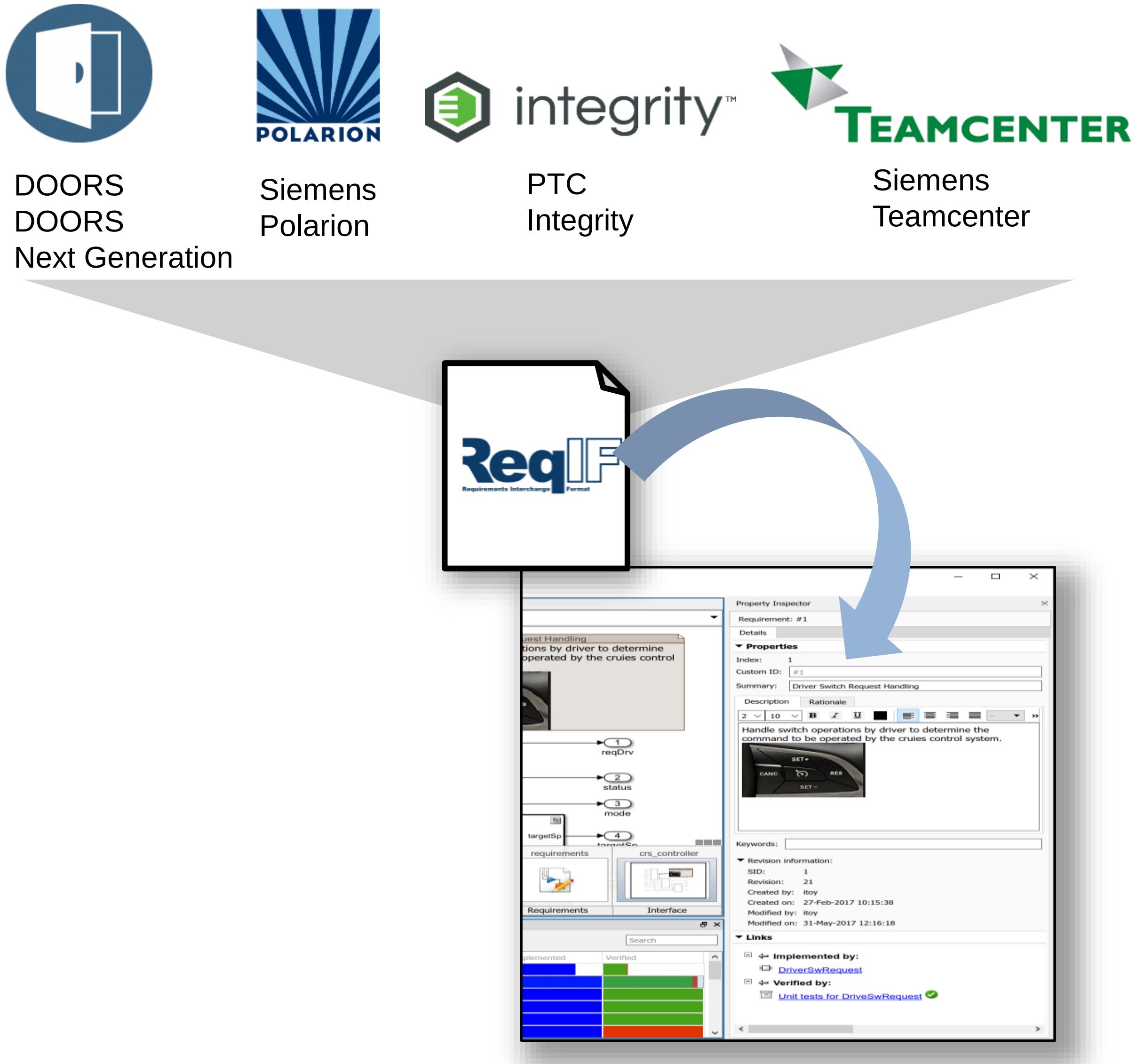
Requirements Import with ReqIF Standard

Allows you to work with requirements from third party tools in Simulink

- Import requirements from third party tools using ReqIF standard ([Requirements Interexchange Format](#))

- Import wizard supports mapping custom attributes

- Tools that support ReqIF standard:
  - IBM DOORS / DOORS Next Generation
  - Siemens Polarion
  - PTC Integrity



DOORS
DOORS
Next Generation

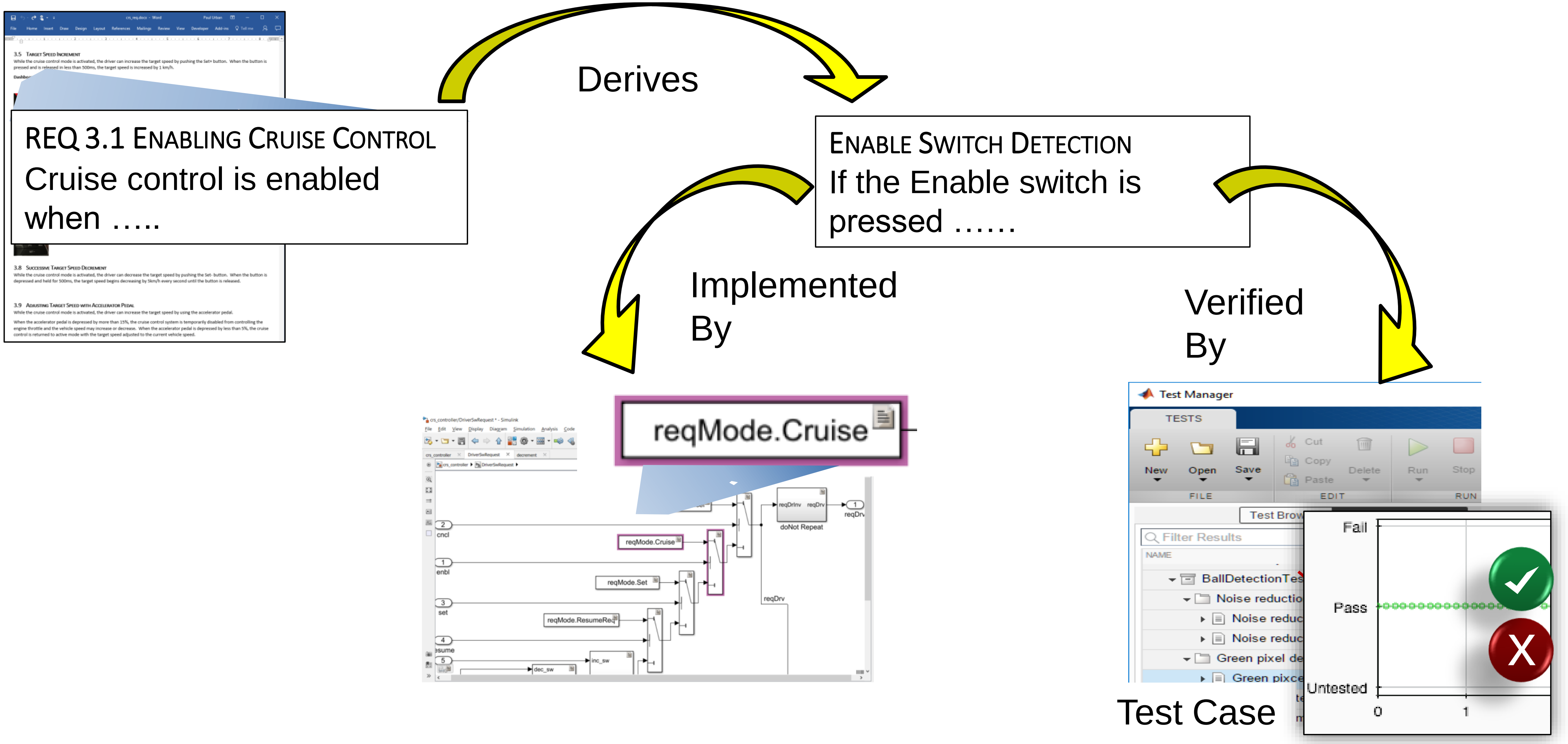
Siemens
Polarion

PTC
Integrity

Siemens
Teamcenter

Property Inspector
Requirement: #1
Details
Properties
Index: 1
Custom ID: #1
Summary: Driver Switch Request Handling
Description | Rationale
Handle switch operations by driver to determine the command to be operated by the cruies control system.
Keywords:
Revision information:
SID: 1
Revision: 21
Created by: itoy
Created on: 27-Feb-2017 10:15:38
Modified by: itoy
Modified on: 31-May-2017 12:16:18
Links
Implemented by:
DriverSwRequest
Verified by:
Unit tests for DriveSwRequest

Link Requirements, Designs and Tests



crs_req.docx - Word

3.5 Target Speed Increment

While the cruise control mode is activated, the driver can increase the target speed by pushing the Set+ button. When the button is pressed and is released in less than 500ms, the target speed is increased by 1 km/h.

3.8 Successive Target Speed Decrement

While the cruise control mode is activated, the driver can decrease the target speed by pushing the Set- button. When the button is depressed and held for 500ms, the target speed begins decreasing by 5km/h every second until the button is released.

3.9 Adjusting Target Speed with Accelerator Pedal

While the cruise control mode is activated, the driver can increase the target speed by using the accelerator pedal.

When the accelerator pedal is depressed by more than 15%, the cruise control system is temporarily disabled from controlling the engine throttle and the vehicle speed may increase or decrease. When the accelerator pedal is depressed by less than 5%, the cruise control is returned to active mode with the target speed adjusted to the current vehicle speed.

REQ 3.1 Enabling Cruise Control
Cruise control is enabled when …..

Derives

Enable Switch Detection
If the Enable switch is pressed ……

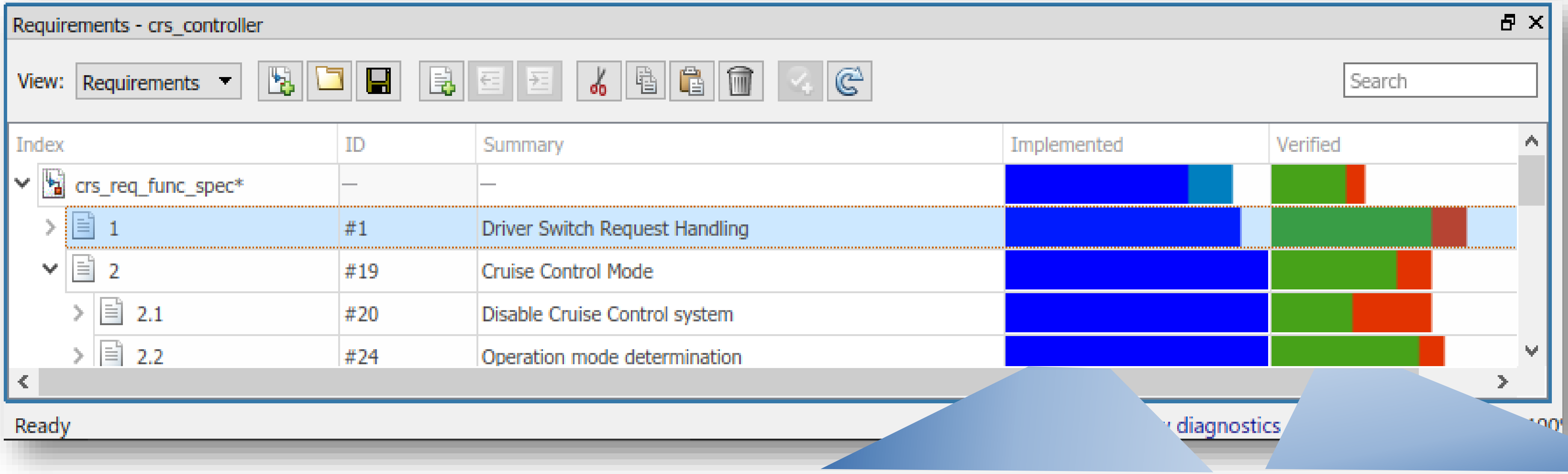
Implemented By

Verified By

reqMode.Cruise

Test Case

Track Implementation and Verification



Requirements - crs_controller

View: Requirements

Search

Index	ID	Summary	Implemented	Verified
crs_req_func_spec*	—	—		
1	#1	Driver Switch Request Handling		
2	#19	Cruise Control Mode		
2.1	#20	Disable Cruise Control system		
2.2	#24	Operation mode determination		

Ready

diagnostics

Implementation Status

- 🟦 Implemented
- ▨ Justified
- □ Missing

Verification Status

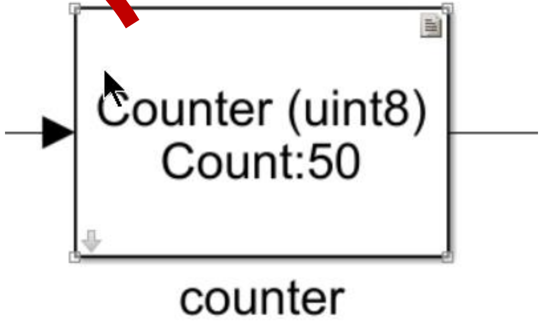
- 🟩 Passed
- 🟥 Failed
- 🟧 No Result
- □ Missing

Respond to Change

Implements

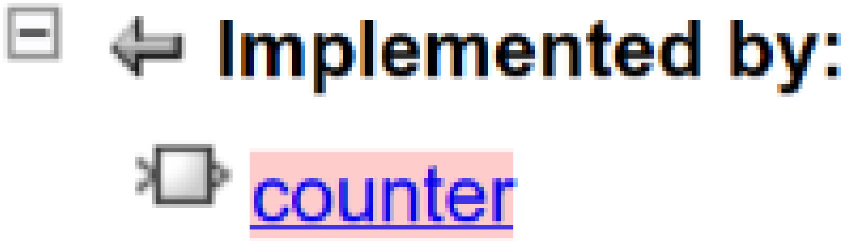
Original Requirement

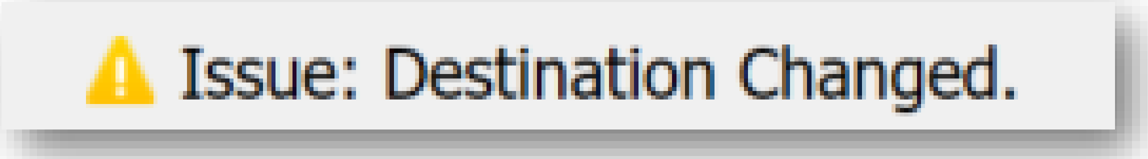
If the switch is pressed and the counter reaches **50** then it shall be recognized as a long press of the tch.



Updated Requirement

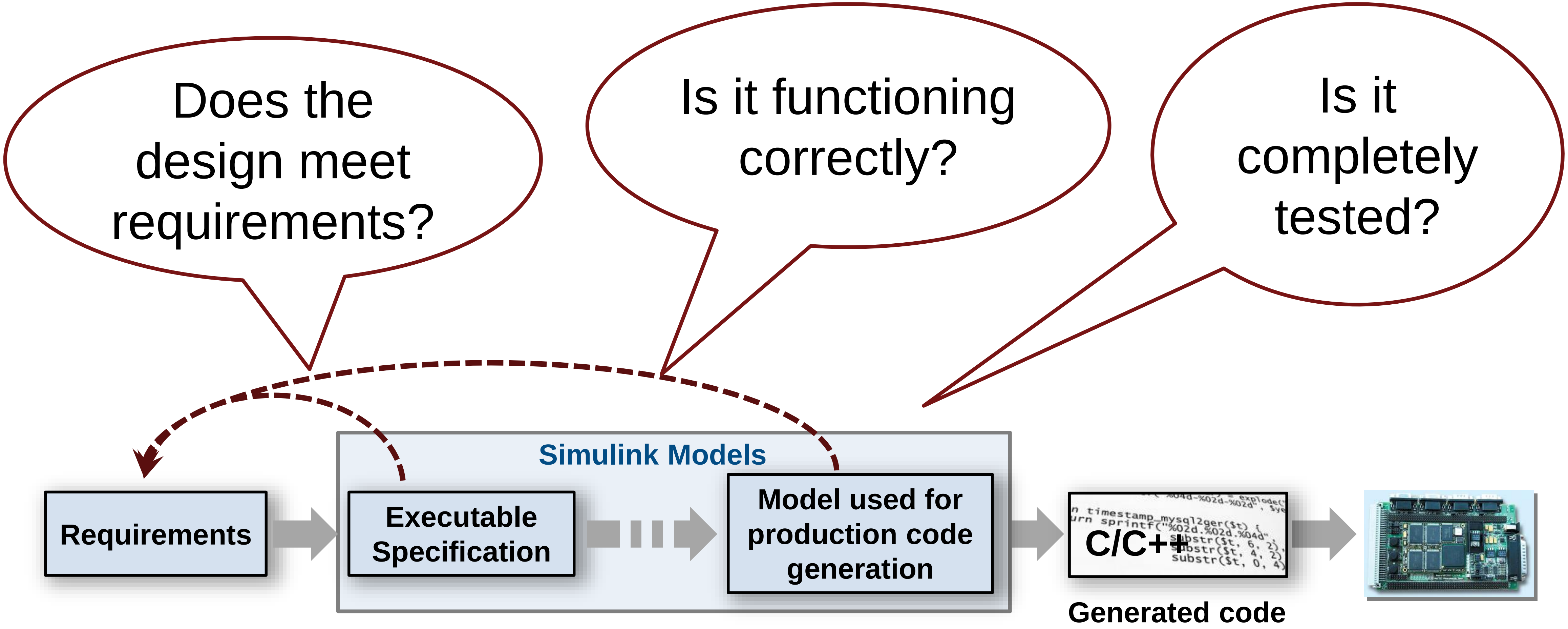
If the switch is pressed and the counter reaches **75** then it shall be recognized as a long press of the switch.





Issue: Destination Changed.

Functional Testing



Does the design meet requirements?

Is it functioning correctly?

Is it completely tested?

Requirements

Simulink Models

Executable Specification

Model used for production code generation

C/C++

Generated code

Systematic Functional Testing

Test Case

Inputs

MAT file (input)

Group 1

Signal 1

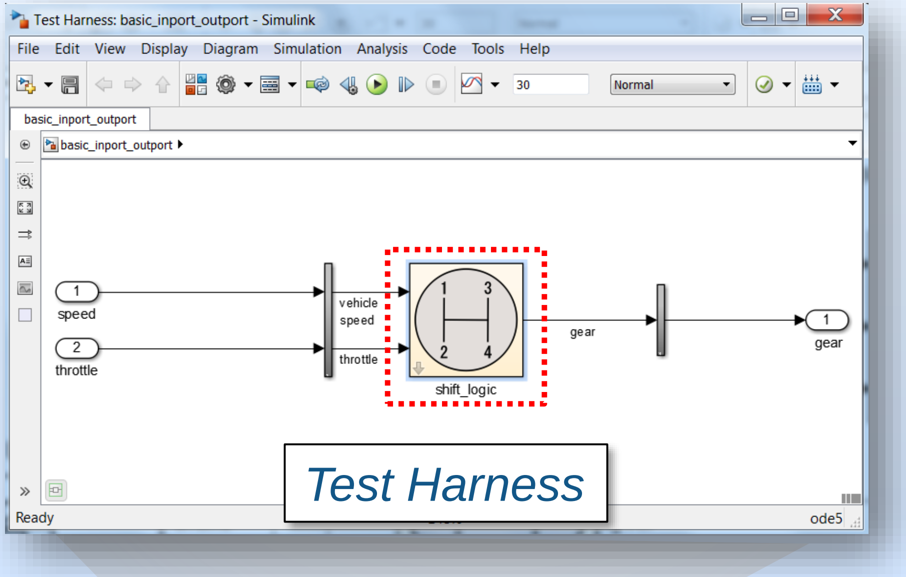
Signal Builder

Test Sequence

and more!

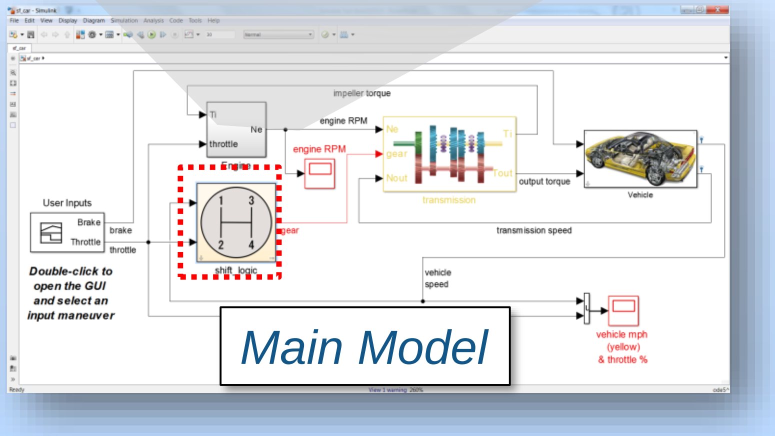
R2017b

Excel file (input)



Test Harness: basic_inport_outport - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
30
Normal
basic_inport_outport
1
speed
2
throttle
vehicle speed
throttle
1 3
2 4
shift_logic
gear
1
gear
Ready
ode5

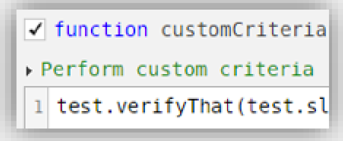
Test Harness



Main Model

Assessments

MAT file (baseline)



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

MATLAB Unit Test

Test Assessment

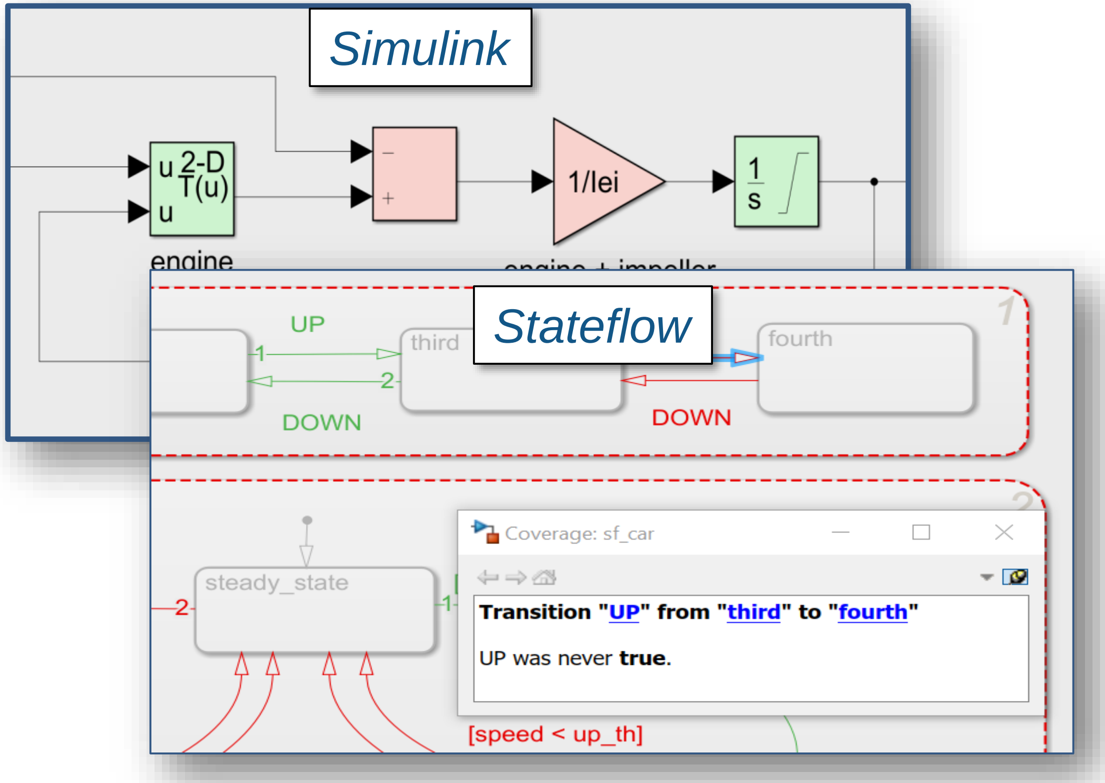
and more!

R2017b

Excel file (baseline)

MATLAB EXPO 2018

Model Coverage Analysis to Measure Testing

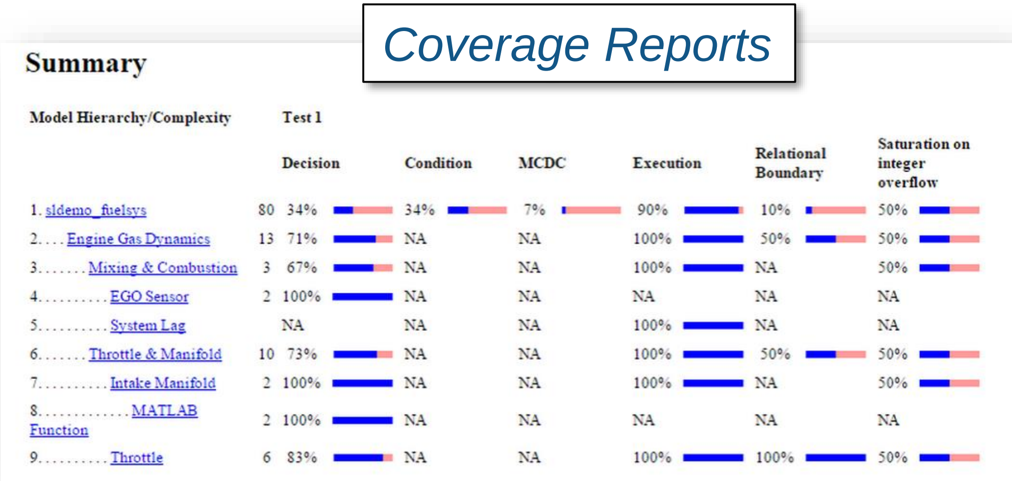


Transition "UP" from "third" to "fourth"

UP was never **true**.

- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Design Errors

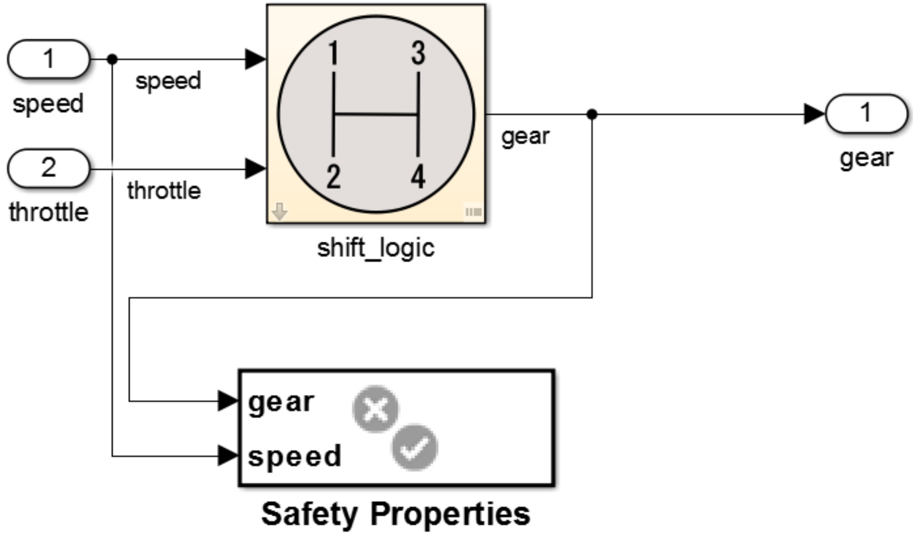
Coverage Reports



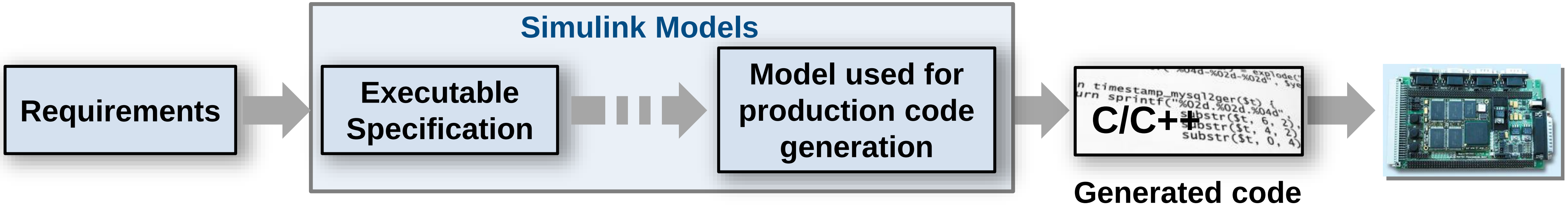
Summary

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

Prove That Design Meets Requirements



- Prove design properties using formal requirement models
- Model functional and safety requirements
- Generates counter example for analysis and debugging

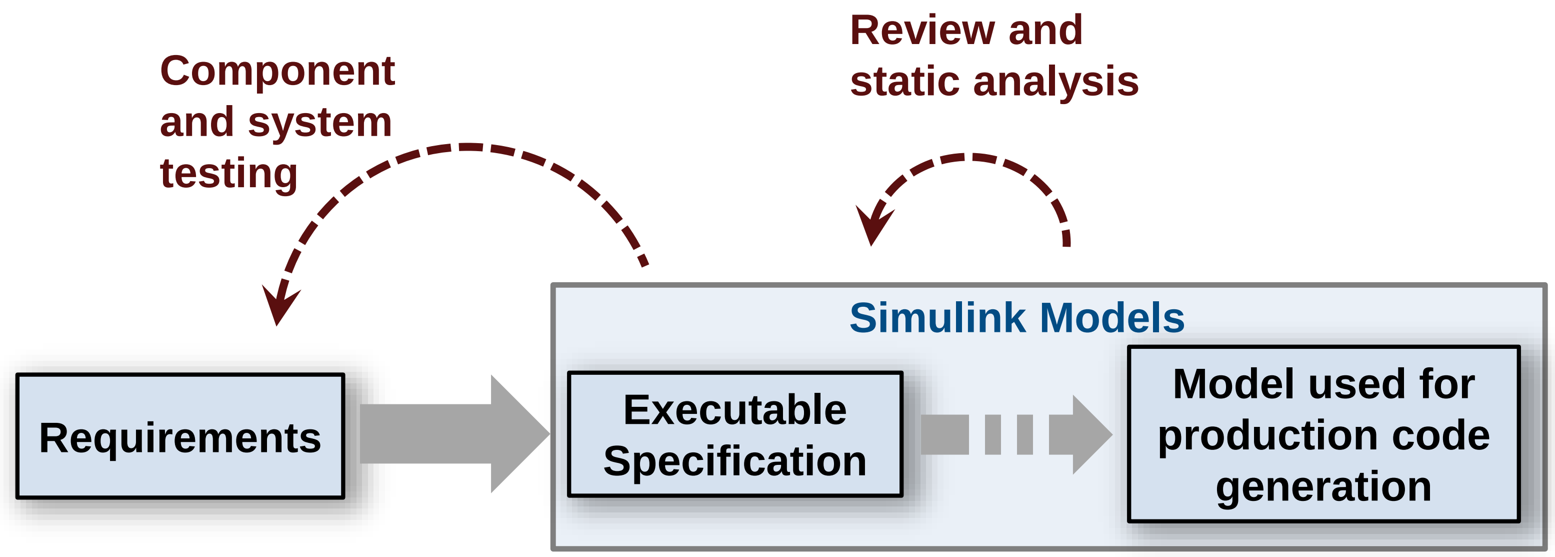


Model Based Design Verification Workflow

1. Develop functions, perform ad-hoc testing, implement traceability

2. Refine design, Validate and Verify

3. Automatically detect quality issues and run-time error



Component and system testing

Review and static analysis

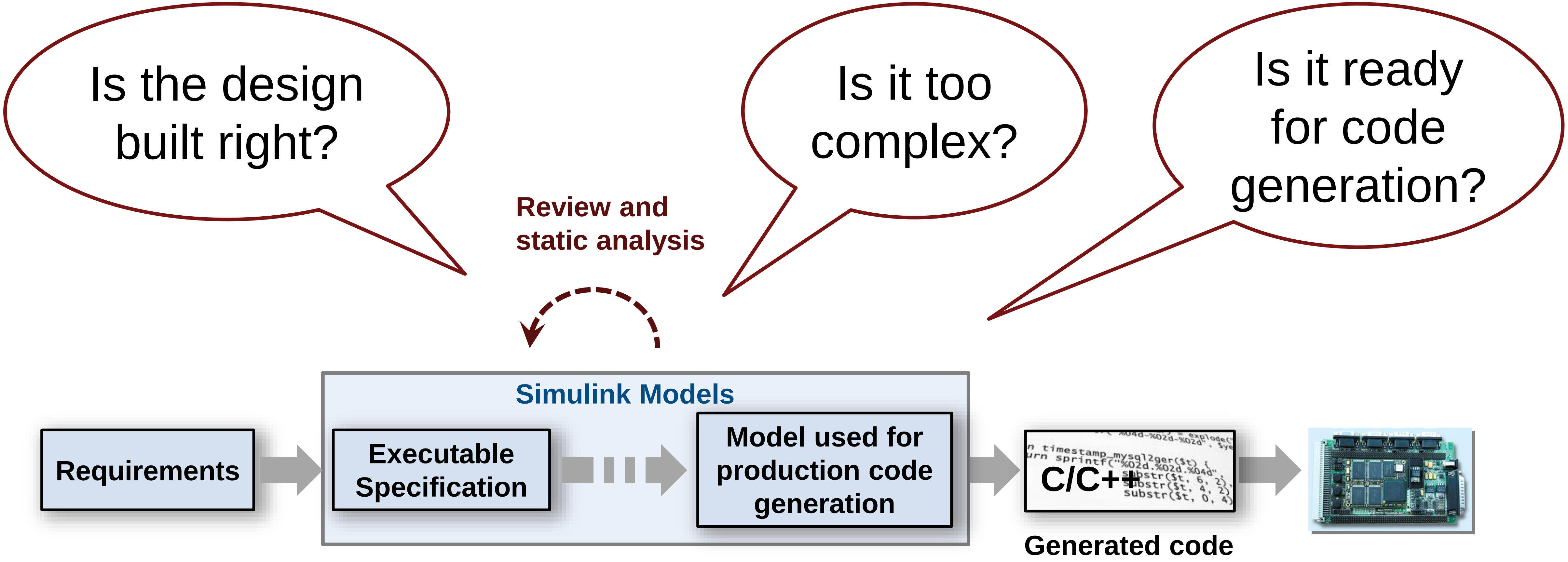
Simulink Models

Requirements

Executable Specification

Model used for production code generation

Verify Design to Guidelines and Standards



Is the design built right?

Is it too complex?

Is it ready for code generation?

Review and static analysis

Simulink Models

Requirements

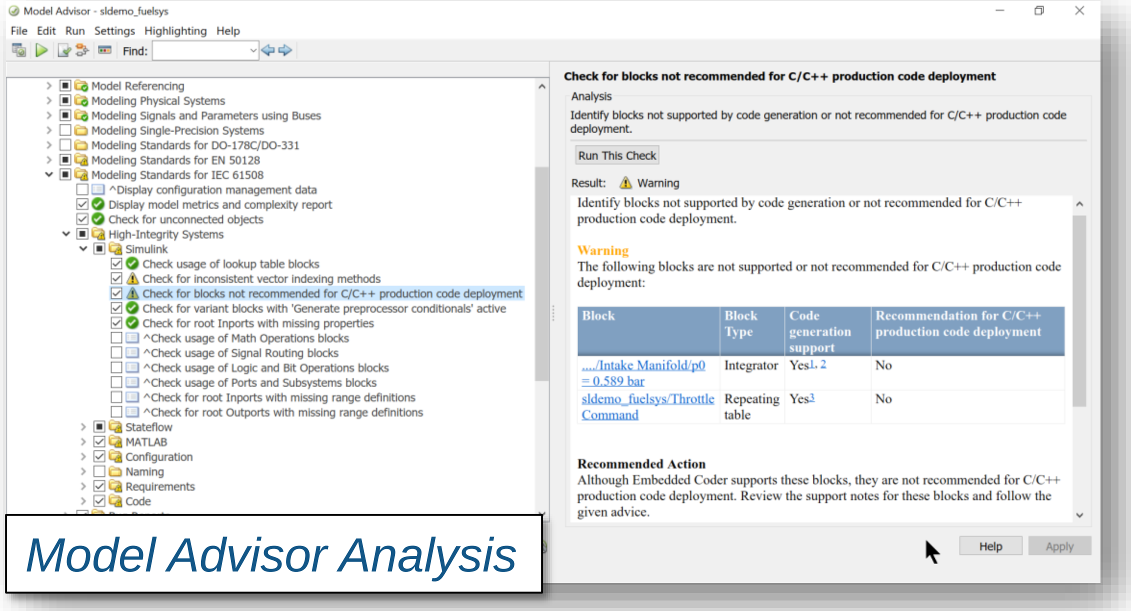
Executable Specification

Model used for production code generation

C/C++

Generated code

Automate verification with static analysis


Model Advisor - sldemo_fuelsys
File Edit Run Settings Highlighting Help
Find:

- › ■ 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
    - › ☑ Requirements
    - › ☑ Code

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

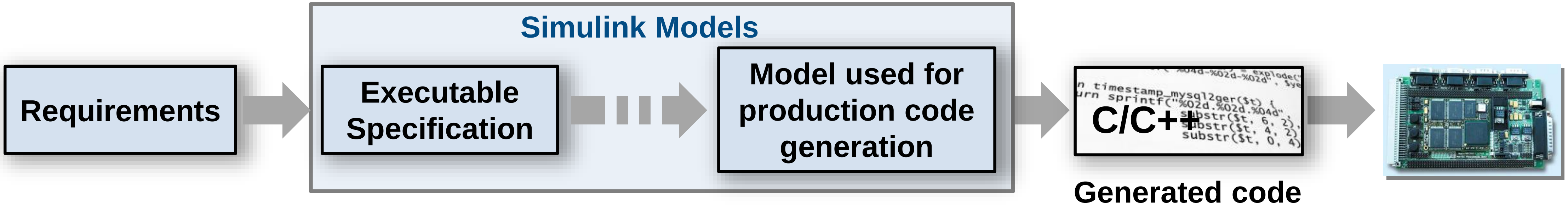
Recommended Action

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

Model Advisor Analysis

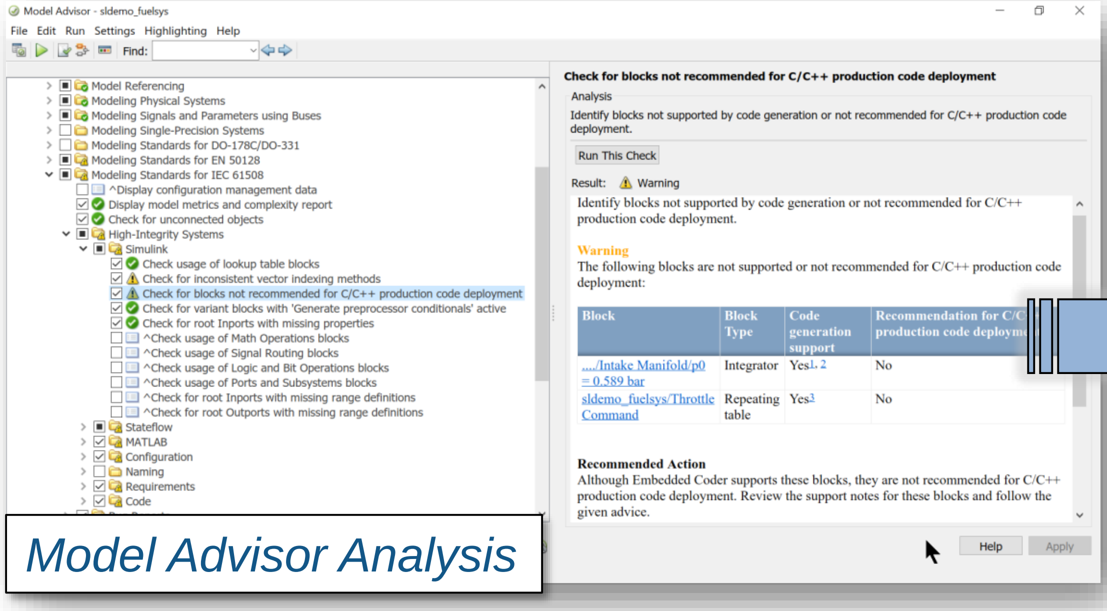
Check for:

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



Generated code

Generate reports for reviews and documentation



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
    - ☑ Requirements
    - ☑ Code

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

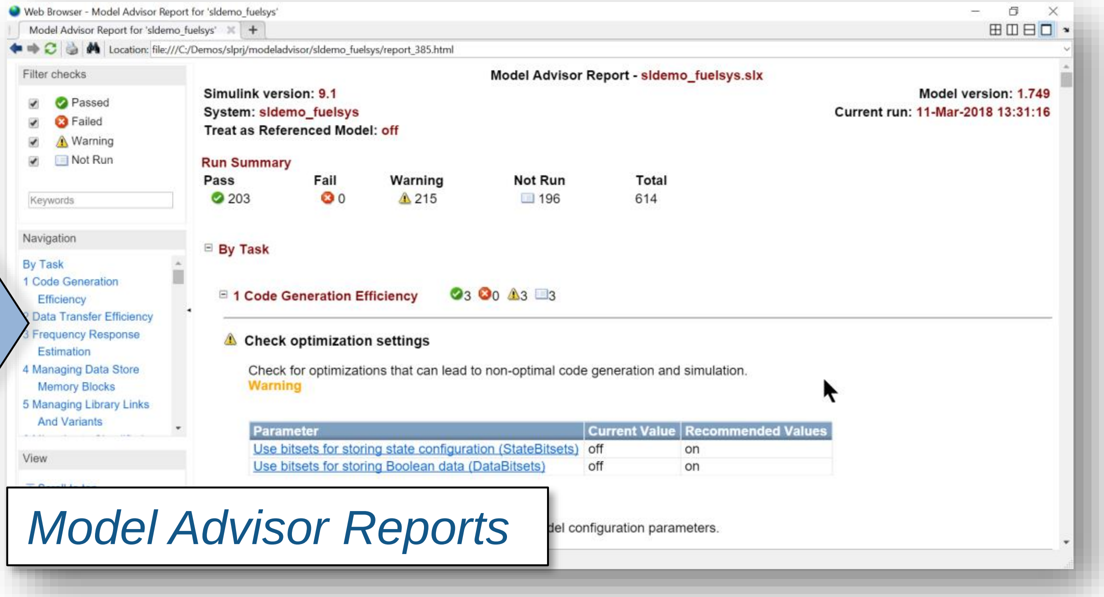
The following blocks are not supported or not recommended for C/C++ production code deployment:

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++ production code deployment. Review the support notes for these blocks and follow the given advice.

Model Advisor Analysis



Web Browser - Model Advisor Report for 'sldemo_fuelsys'

Location: file:///C:/Demos/slprj/modeladvisor/sldemo_fuelsys/report_385.html

Filter checks

- ☑ ✓ Passed
- ☑ ✗ Failed
- ☑ ⚠ Warning
- ☑ Not Run

Navigation

- By Task
- 1 Code Generation Efficiency
- 2 Data Transfer Efficiency
- 3 Frequency Response Estimation
- 4 Managing Data Store Memory Blocks
- 5 Managing Library Links And Variants

View

Model Advisor Report - sldemo_fuelsys.slx

Simulink version: 9.1
System: sldemo_fuelsys
Treat as Referenced Model: off

Model version: 1.749
Current run: 11-Mar-2018 13:31:16

Run Summary

Pass	Fail	Warning	Not Run	Total
✓ 203	✗ 0	⚠ 215	196	614

⊟ By Task

⊟ 1 Code Generation Efficiency ✓3 ✗0 ⚠3 3

⚠ **Check optimization settings**

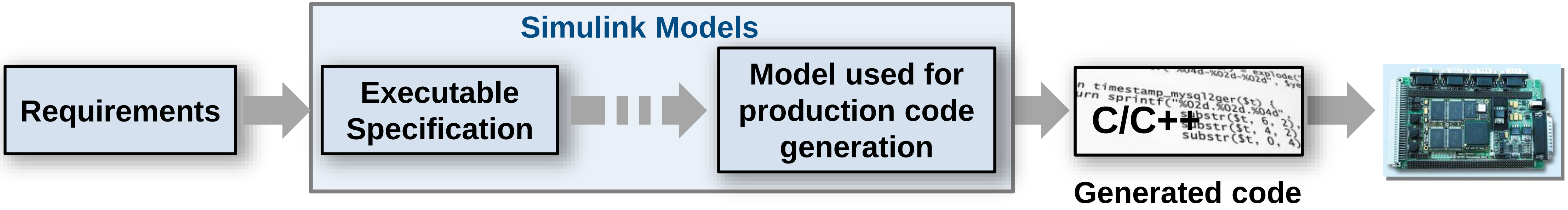
Check for optimizations that can lead to non-optimal code generation and simulation.

Warning

Parameter	Current Value	Recommended Values
Use bitsets for storing state configuration (StateBitsets)	off	on
Use bitsets for storing Boolean data (DataBitsets)	off	on

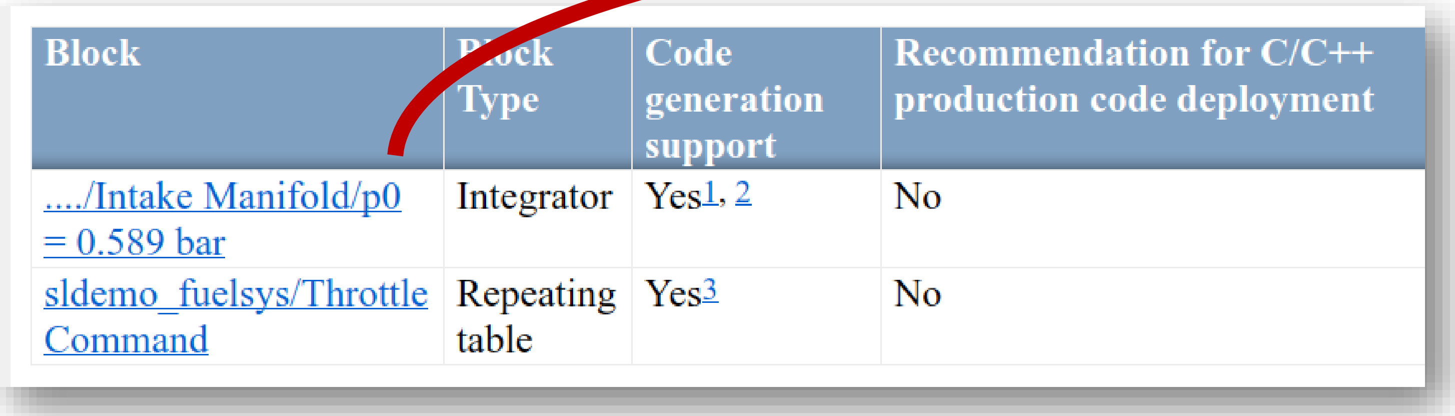
del configuration parameters.

Model Advisor Reports

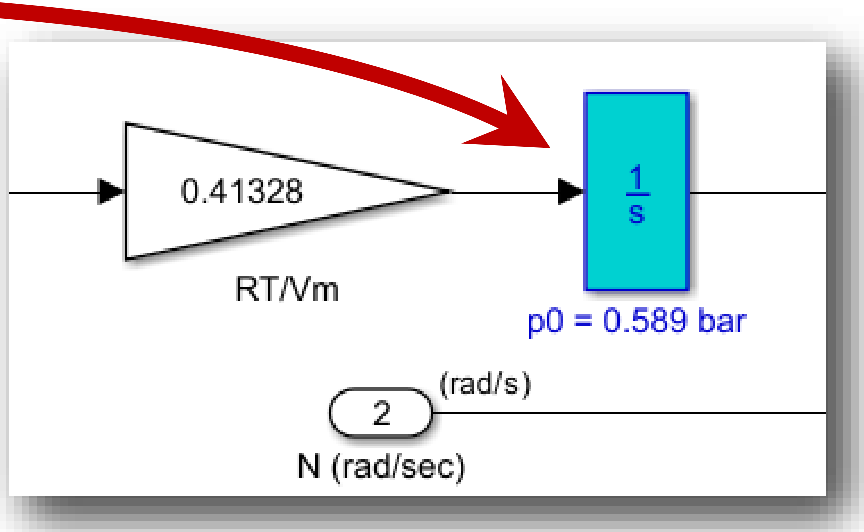


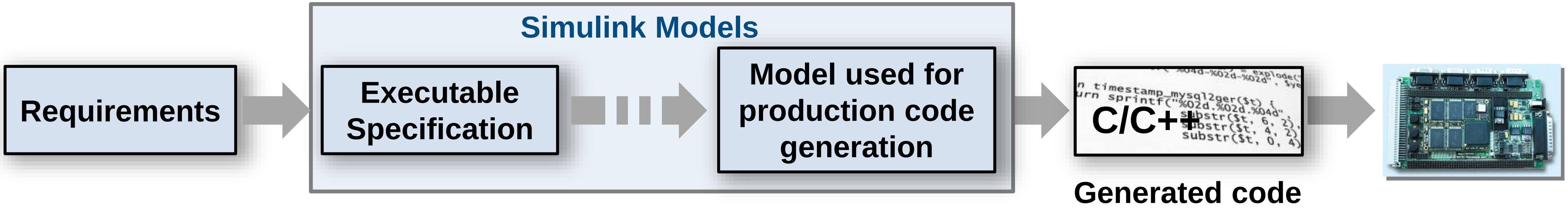
Generated code

Navigate to Problematic Blocks



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





Requirements

Simulink Models

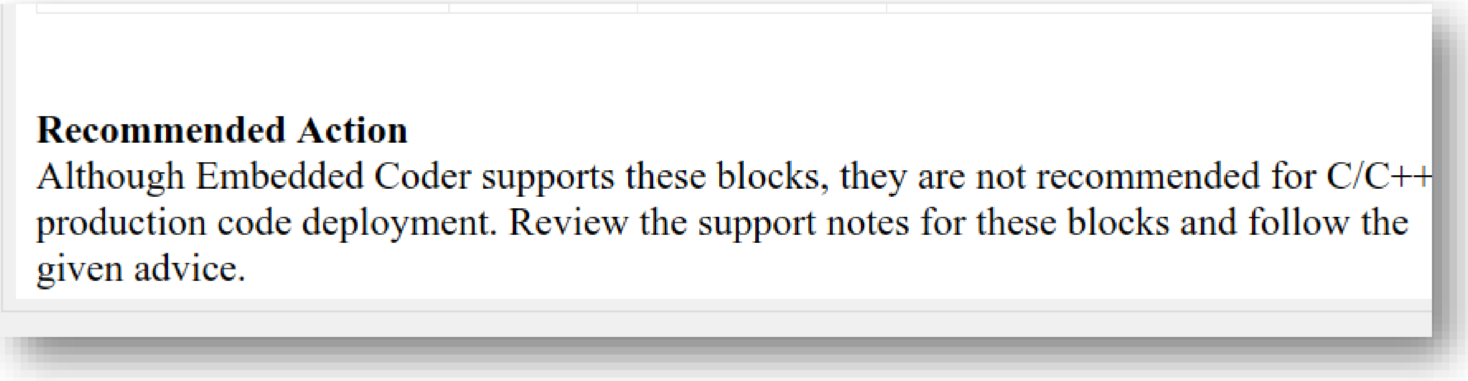
Executable Specification

Model used for production code generation

C/C++

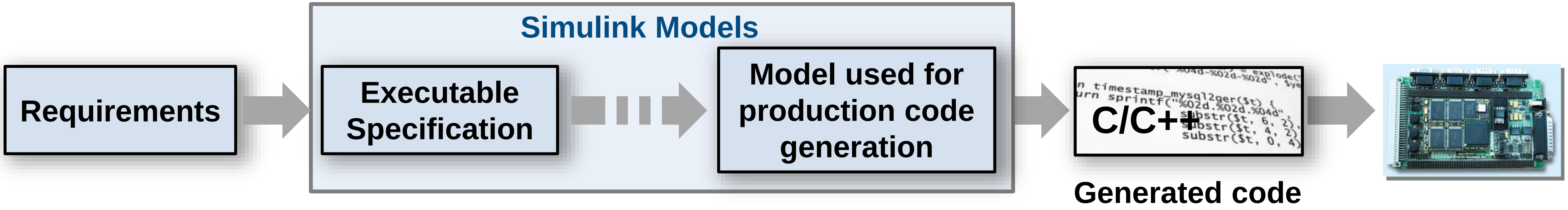
Generated code

Guidance Provided to Address Issues or Automatically Correct



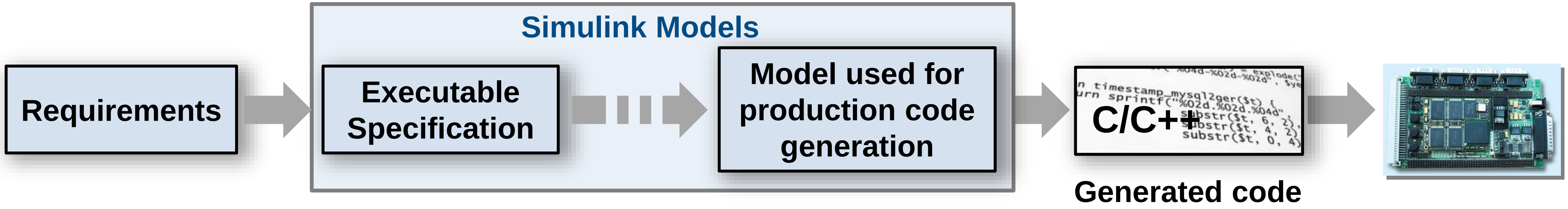
Recommended Action

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



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
- MAAB (MathWorks Automotive Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)



Requirements

Simulink Models

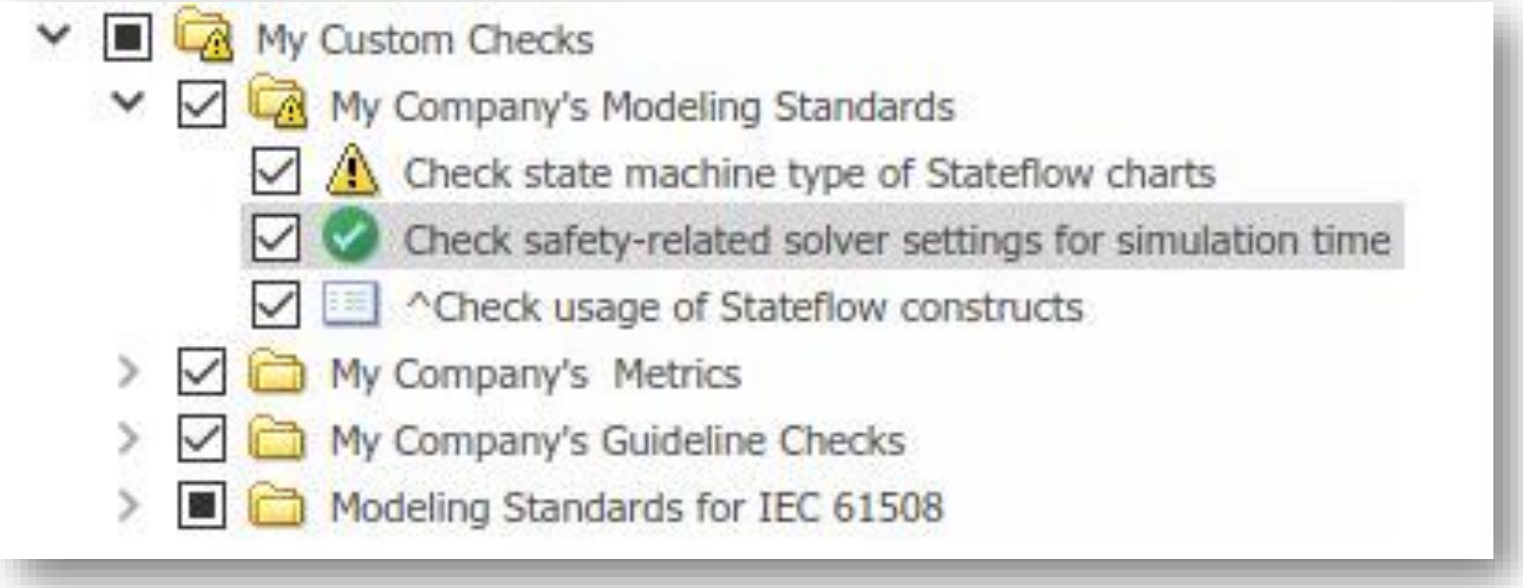
Executable Specification

Model used for production code generation

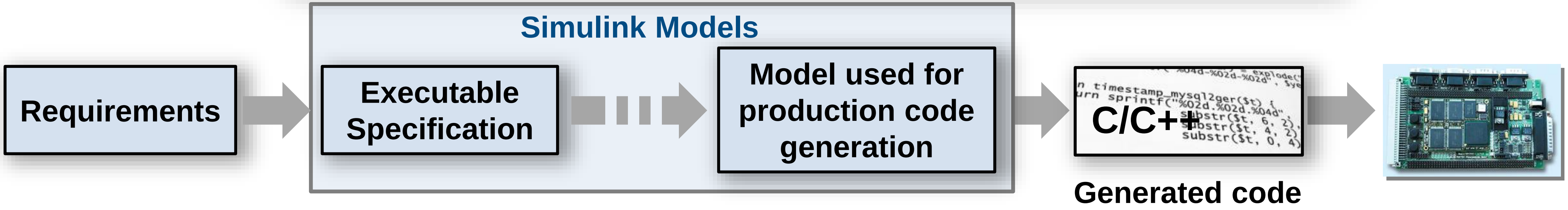
C/C++

Generated code

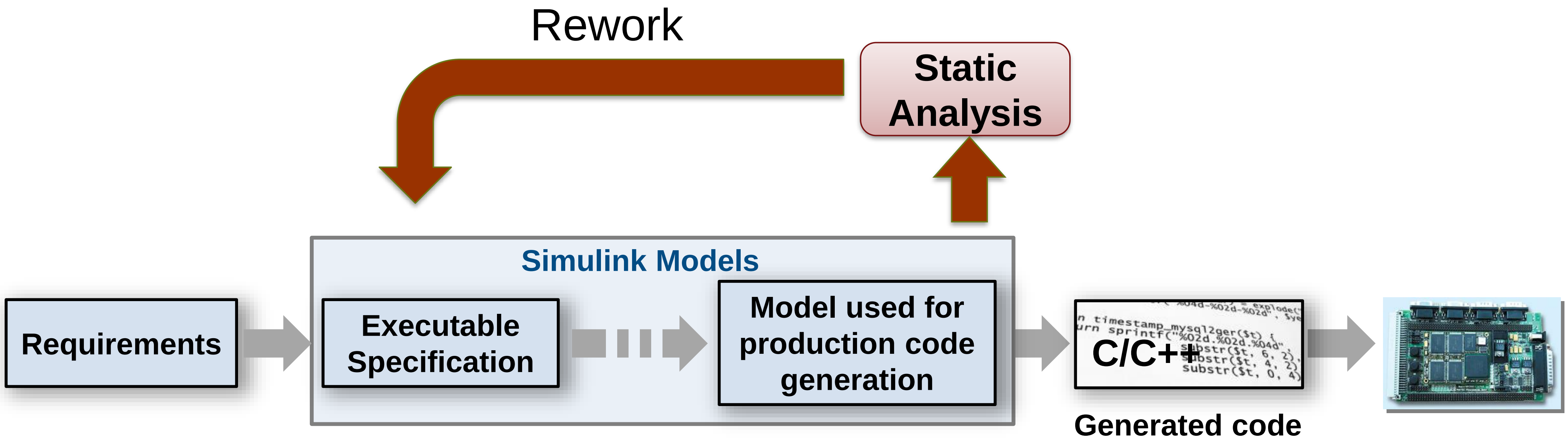
Configure and customize analysis



- ■ My Custom Checks
  - ☑ My Company's Modeling Standards
    - ☑ ⚠ Check state machine type of Stateflow charts
    - ☑ ✓ Check safety-related solver settings for simulation time
    - ☑ ^Check usage of Stateflow constructs
  - ☑ My Company's Metrics
  - ☑ My Company's Guideline Checks
  - ■ Modeling Standards for IEC 61508

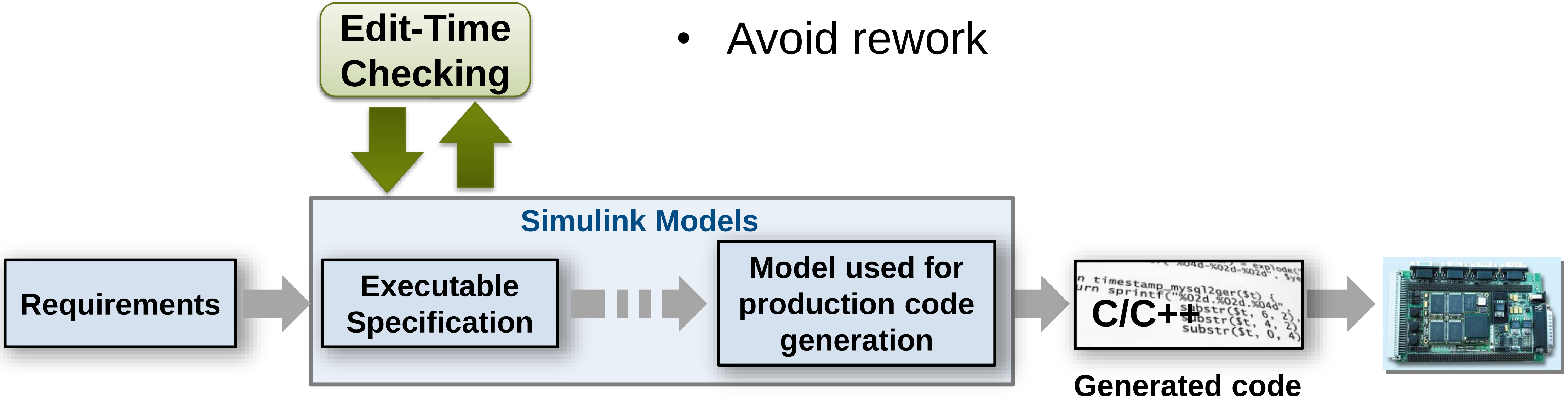


Checks for standards and guidelines are often performed late



Shift Verification Earlier With Edit-Time Checking

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



Edit-Time Checking

Simulink Models

Requirements

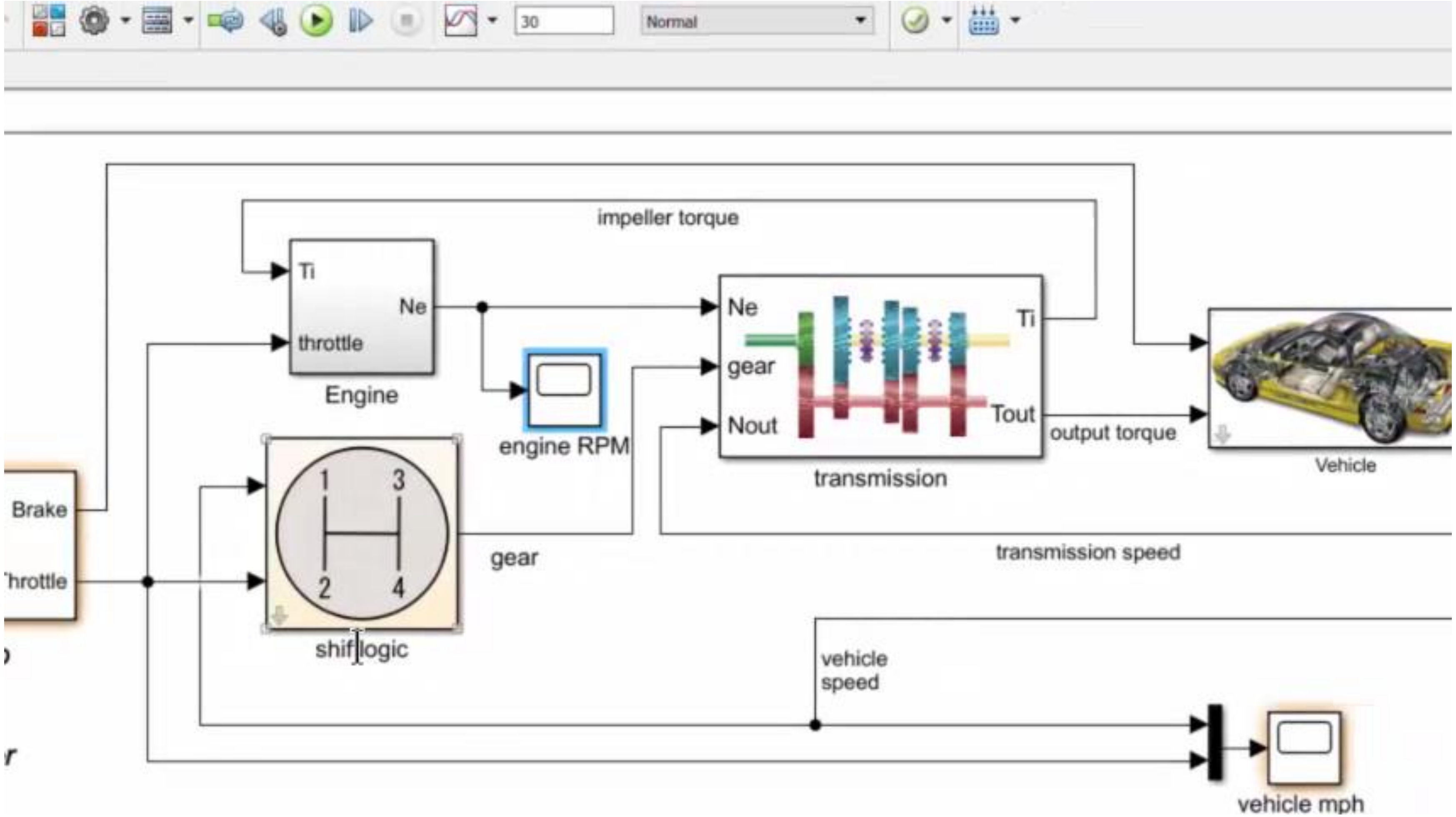
Executable Specification

Model used for production code generation

C/C++

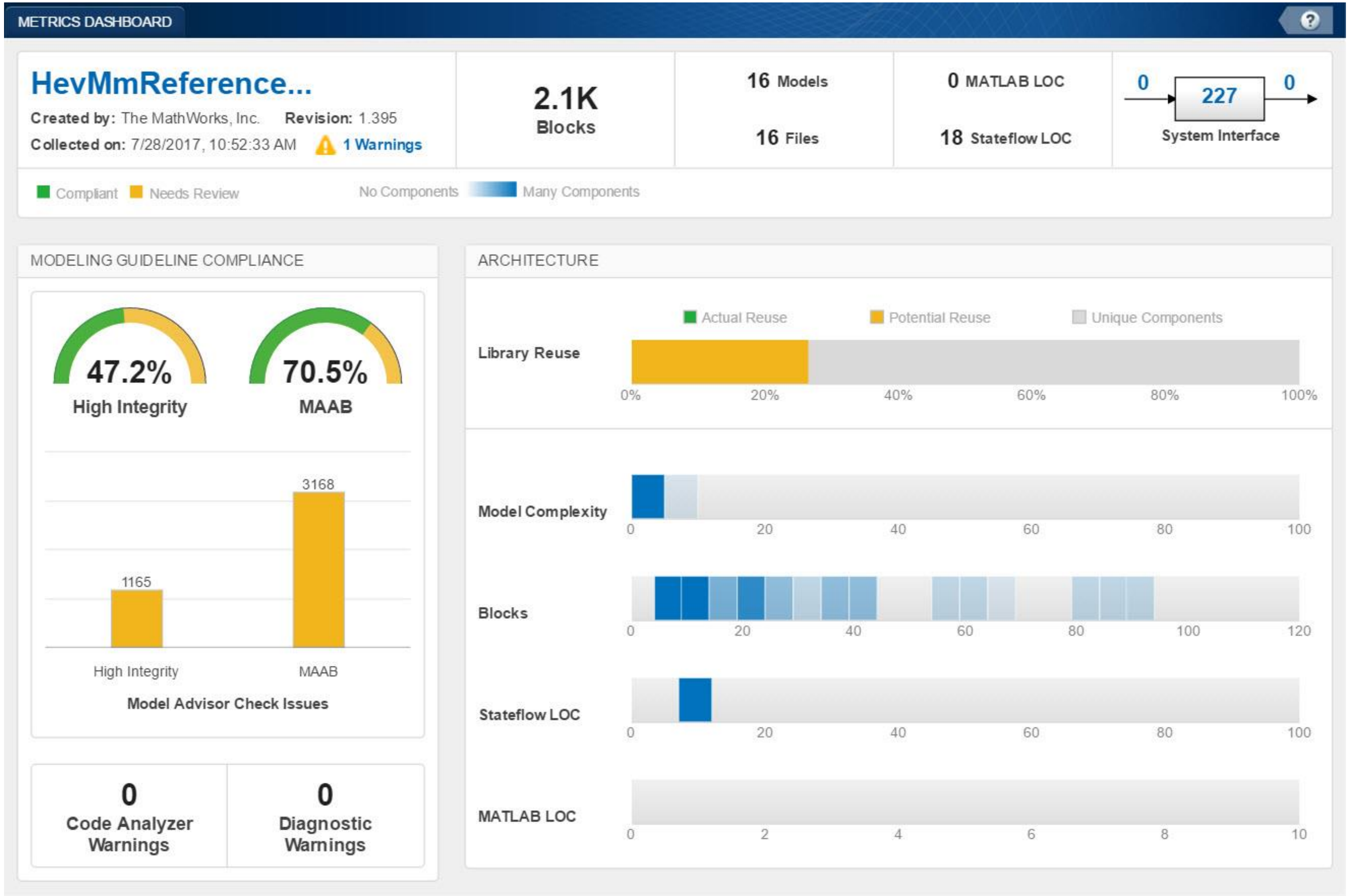
Generated code

Find Compliance Issues as you Edit with Edit-Time Checking



MATLAB EXPO 2018

Assess Quality with Metrics Dashboard



METRICS DASHBOARD

HevMmReference...

Created by: The MathWorks, Inc. Revision: 1.395

Collected on: 7/28/2017, 10:52:33 AM ⚠ 1 Warnings

2.1K Blocks

16 Models

16 Files

0 MATLAB LOC

18 Stateflow LOC

0 227 0

System Interface

🟩 Compliant 🟨 Needs Review No Components Many Components

MODELING GUIDELINE COMPLIANCE

47.2% High Integrity

70.5% MAAB

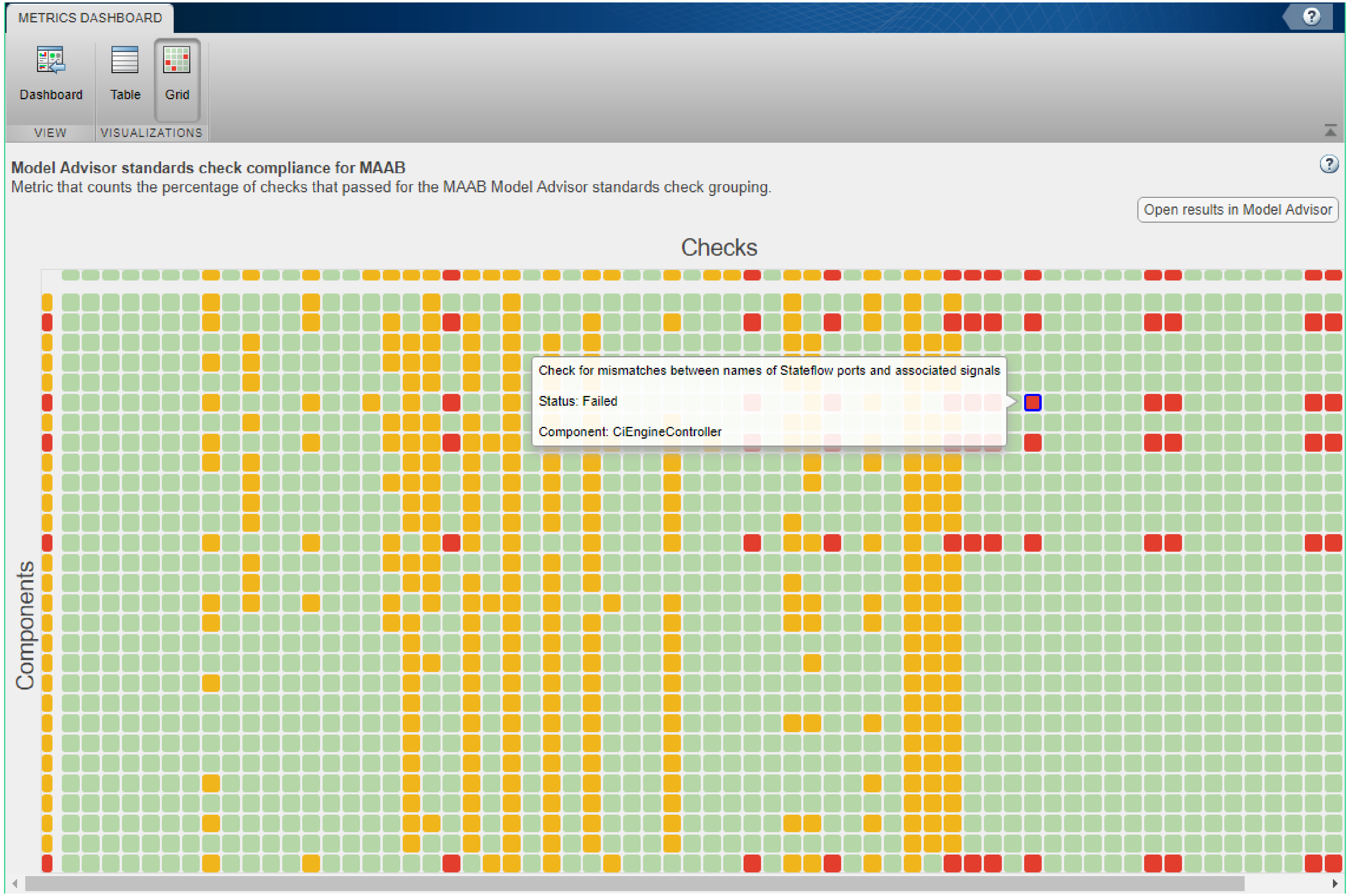
0 Code Analyzer Warnings

0 Diagnostic Warnings

ARCHITECTURE

- Consolidated view of metrics
  - Size
  - Compliance
  - Complexity
- Identify where problem areas may be

Grid Visualization for Metrics



Model Advisor standards check compliance for MAAB

Metric that counts the percentage of checks that passed for the MAAB Model Advisor standards check grouping.

Check for mismatches between names of Stateflow ports and associated signals

Status: Failed

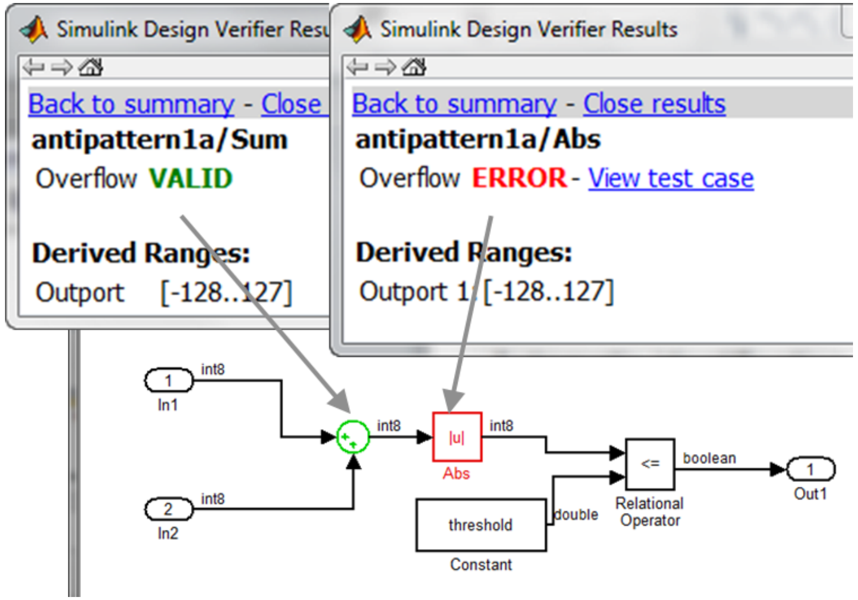
Component: CiEngineController

- Visualize Standards Check Compliance
  - Find Issues
  - Identify patterns
  - See hot spots

Legend:

🟥	Red:	Fail
🟧	Orange:	Warning
🟩	Green:	Pass
▨	Gray:	Not run

Detect Design Errors with 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

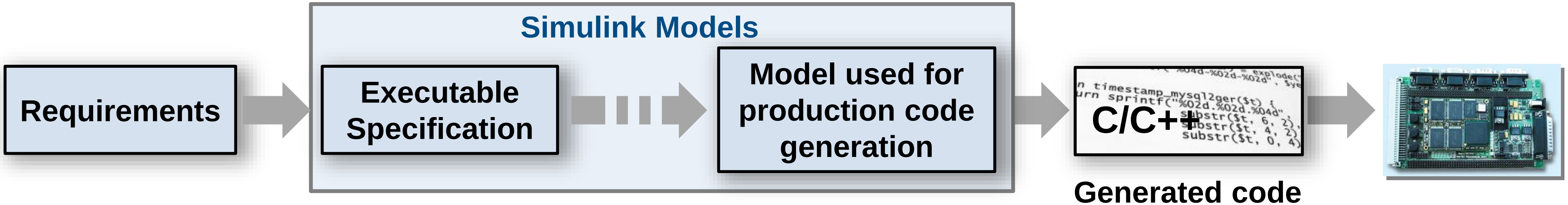
Overflow **ERROR** - View test case

Derived Ranges:

Outport 1 [-128..127]

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

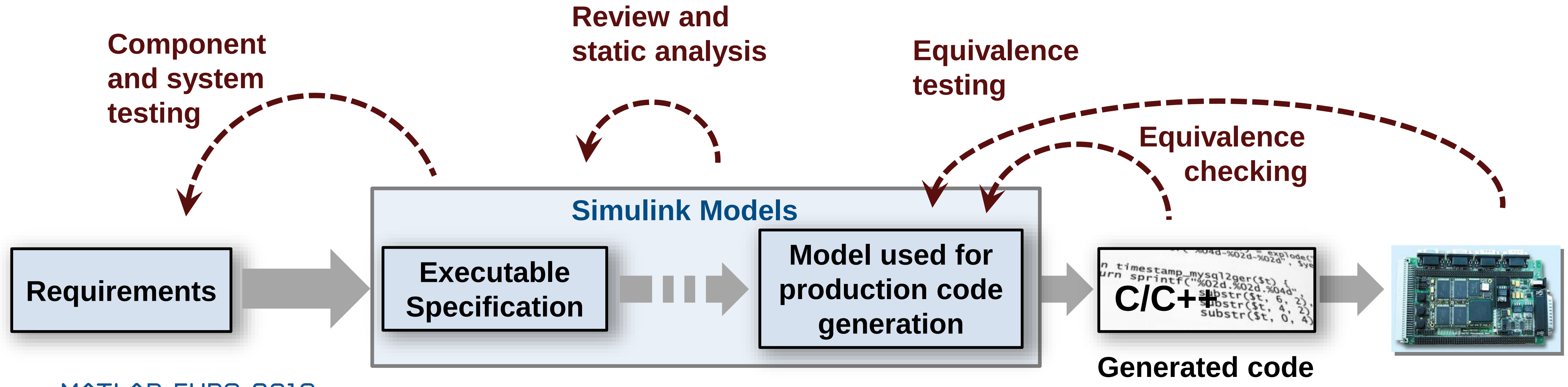
- Generate counter example to reproduce error



Generated code

Model Based Design Verification Workflow

1. Develop functions, perform ad-hoc testing, implement traceability
2. Refine design, Validate and Verify
3. Automatically detect quality issues and run-time error
4. Generate Code & Deploy
5. Auto-execute functional tests, verify product vs specification & Auto-report



Component and system testing

Review and static analysis

Equivalence testing

Equivalence checking

Simulink Models

Requirements

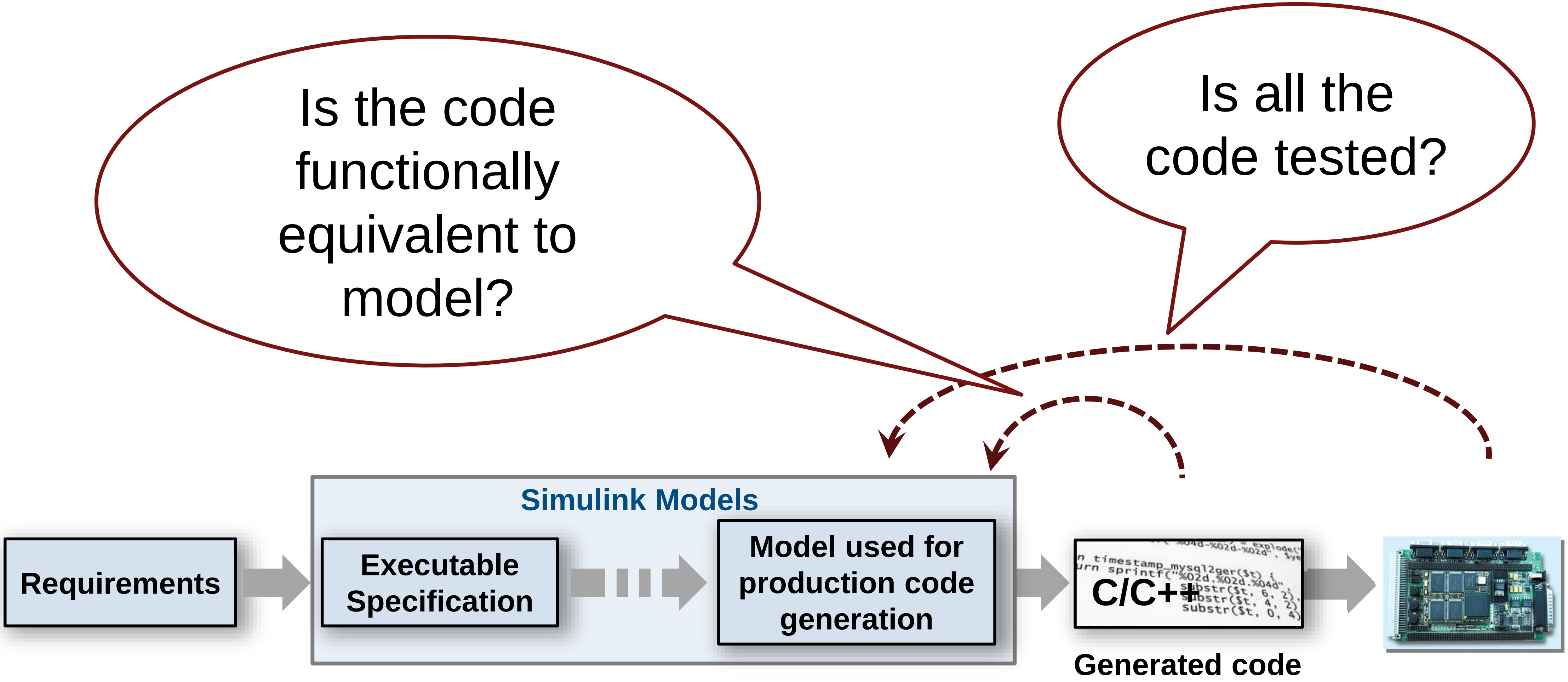
Executable Specification

Model used for production code generation

C/C++

Generated code

Equivalence Testing



Is the code functionally equivalent to model?

Is all the code tested?

Simulink Models

Requirements

Executable Specification

Model used for production code generation

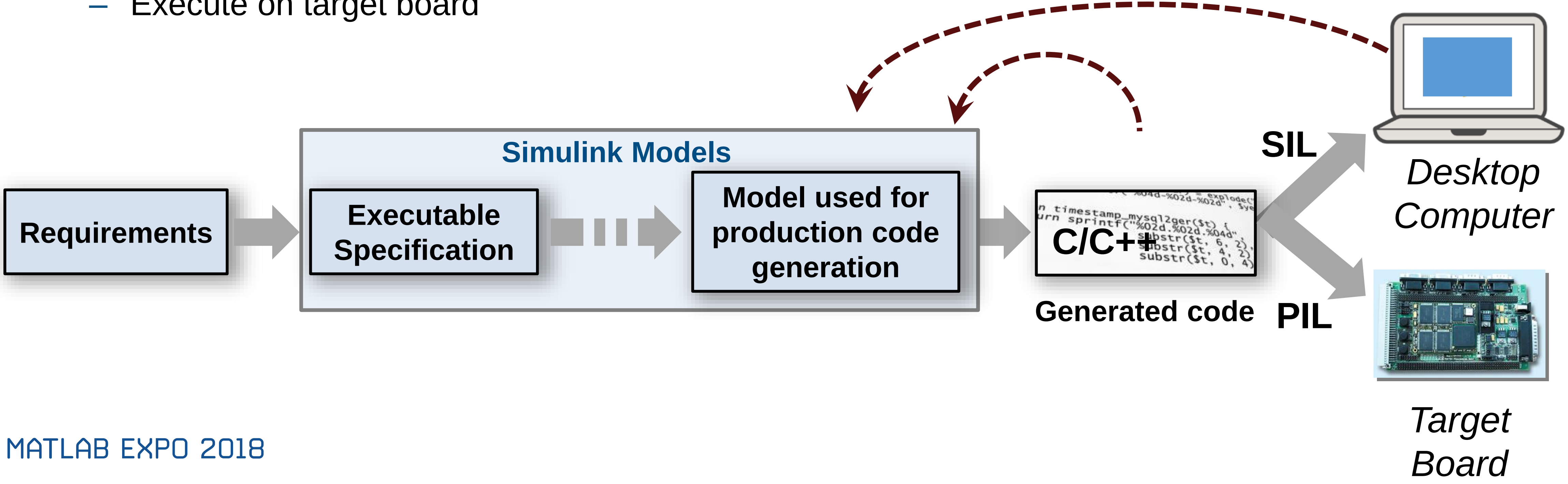
C/C++

Generated code

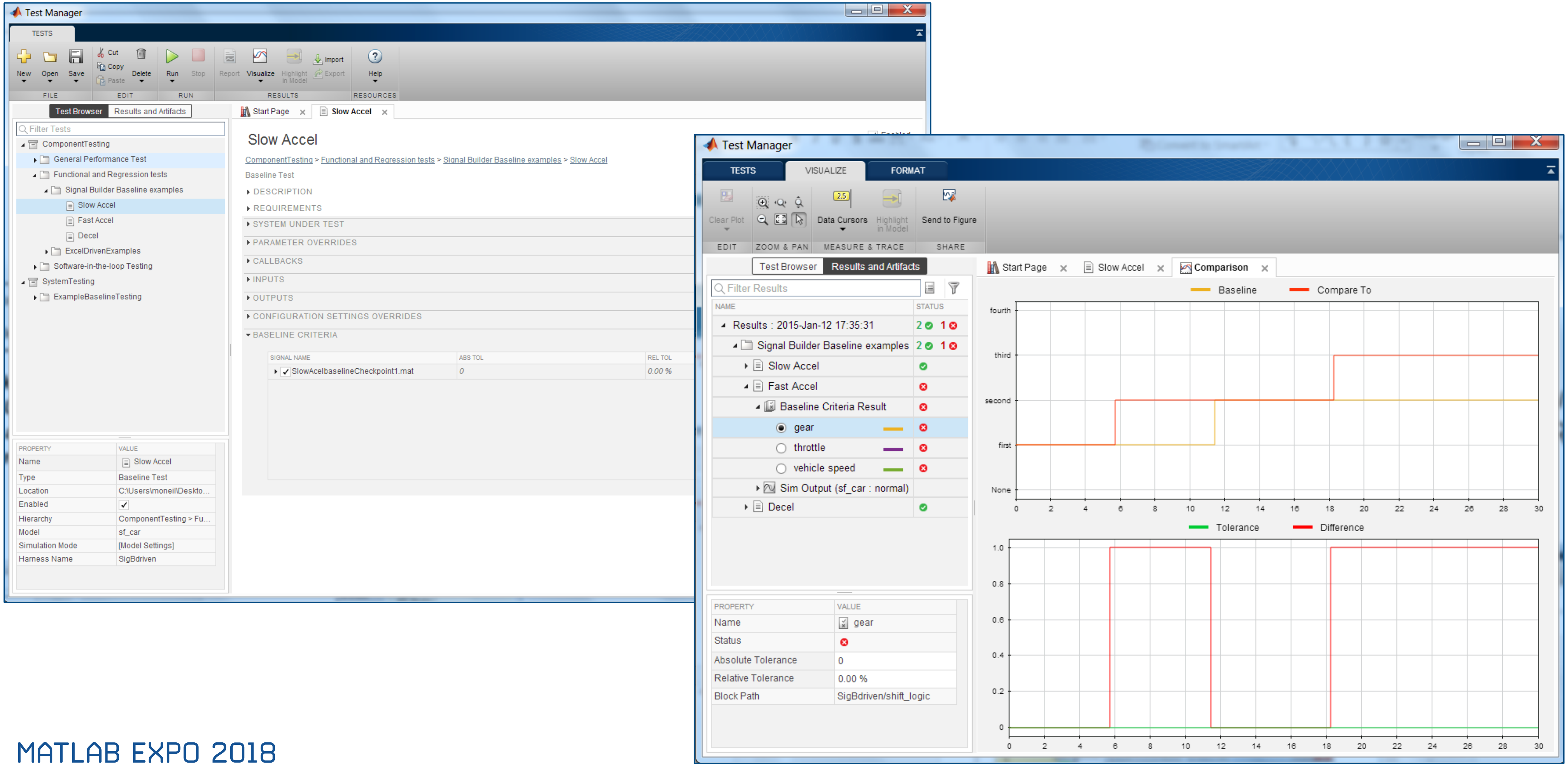
Equivalence Testing

- Software in the Loop (SIL)
  - Show functional equivalence, model to code
  - Execute on desktop / laptop computer
- Processor in the Loop (PIL)
  - Numerical equivalence, model to target code
  - Execute on target board

- Re-deploy model based tests on source copiled or compiled object
- Collect code coverage



Manage Testing and Test Results



- ComponentTesting
  - General Performance Test
  - Functional and Regression tests
    - Signal Builder Baseline examples
      - Slow Accel
      - Fast Accel
      - Decel
    - ExcelDrivenExamples
  - Software-in-the-loop Testing
- SystemTesting
  - ExampleBaselineTesting

Slow Accel

ComponentTesting > Functional and Regression tests > Signal Builder Baseline examples > Slow Accel

Baseline Test

- DESCRIPTION
- REQUIREMENTS
- SYSTEM UNDER TEST
- PARAMETER OVERRIDES
- CALLBACKS
- INPUTS
- OUTPUTS
- CONFIGURATION SETTINGS OVERRIDES
- BASELINE CRITERIA

SIGNAL NAME	ABS TOL	REL TOL
✓ SlowAccelbaselineCheckpoint1.mat	0	0.00 %

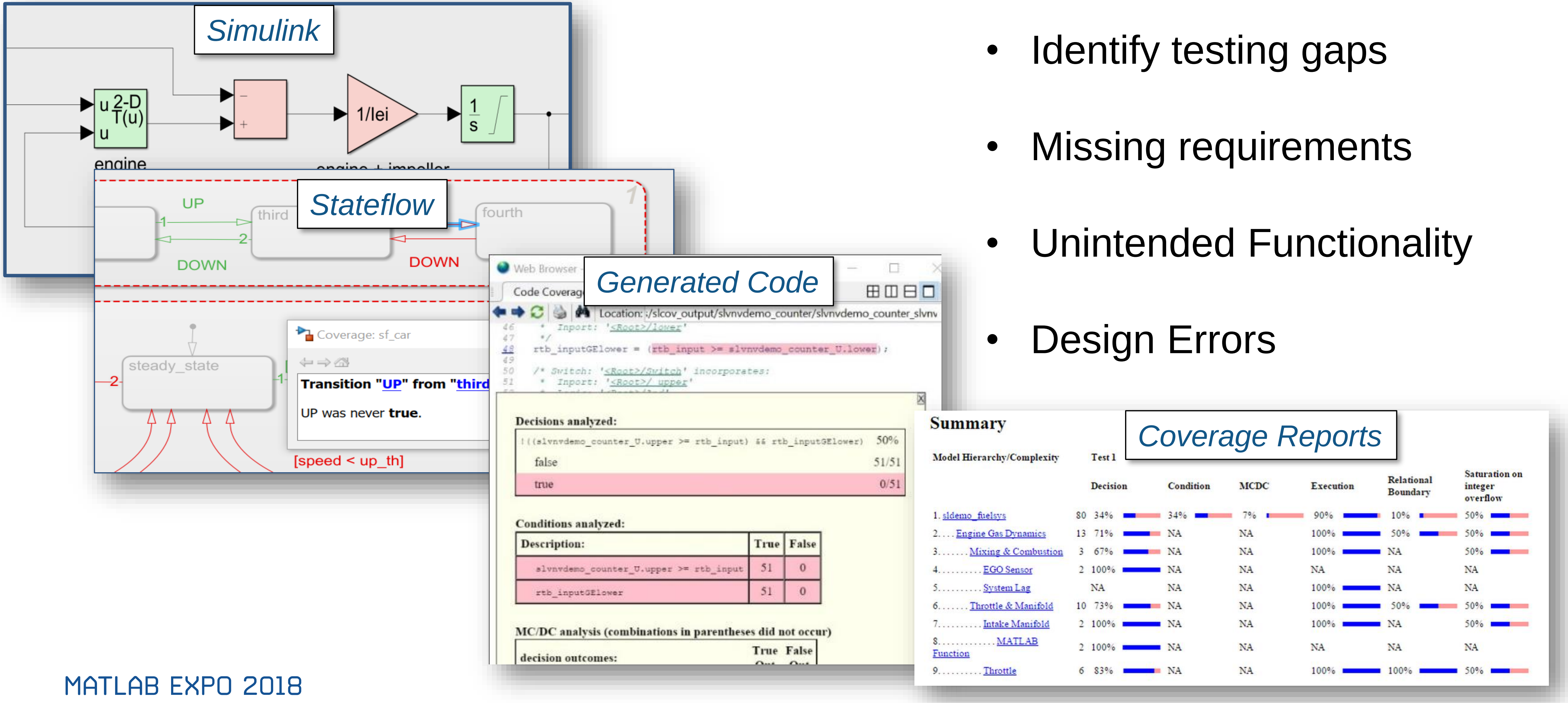
PROPERTY	VALUE
Name	Slow Accel
Type	Baseline Test
Location	C:\Users\moneil\Deskto...
Enabled	✓
Hierarchy	ComponentTesting > Fu...
Model	sf_car
Simulation Mode	[Model Settings]
Harness Name	SigBdriven

NAME	STATUS
Results : 2015-Jan-12 17:35:31	2 ✓ 1 ✗
Signal Builder Baseline examples	2 ✓ 1 ✗
Slow Accel	✓
Fast Accel	✗
Baseline Criteria Result	✗
● gear	✗
○ throttle	✗
○ vehicle speed	✗
Sim Output (sf_car : normal)	
Decel	✓

PROPERTY	VALUE
Name	gear
Status	✗
Absolute Tolerance	0
Relative Tolerance	0.00 %
Block Path	SigBdriven/shift_logic

MATLAB EXPO 2018

Source Code Coverage Measurement & Comparison



Coverage: sf_car

Transition "UP" from "third

UP was never **true**.

```
 *  Inport: '<Root>/lower'
 */
rtb_inputGElower = (rtb_input >= slvnvdemo_counter_U.lower);

/* Switch: '<Root>/Switch' incorporates:
 *  Inport: '<Root>/ upper'
```

Decisions analyzed:

!((slvnvdemo_counter_U.upper >= rtb_input) && rtb_inputGElower)	50%
false	51/51
true	0/51

Conditions analyzed:

Description:	True	False
slvnvdemo_counter_U.upper >= rtb_input	51	0
rtb_inputGElower	51	0

MC/DC analysis (combinations in parentheses did not occur)

decision outcomes:	True Out	False Out

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

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.

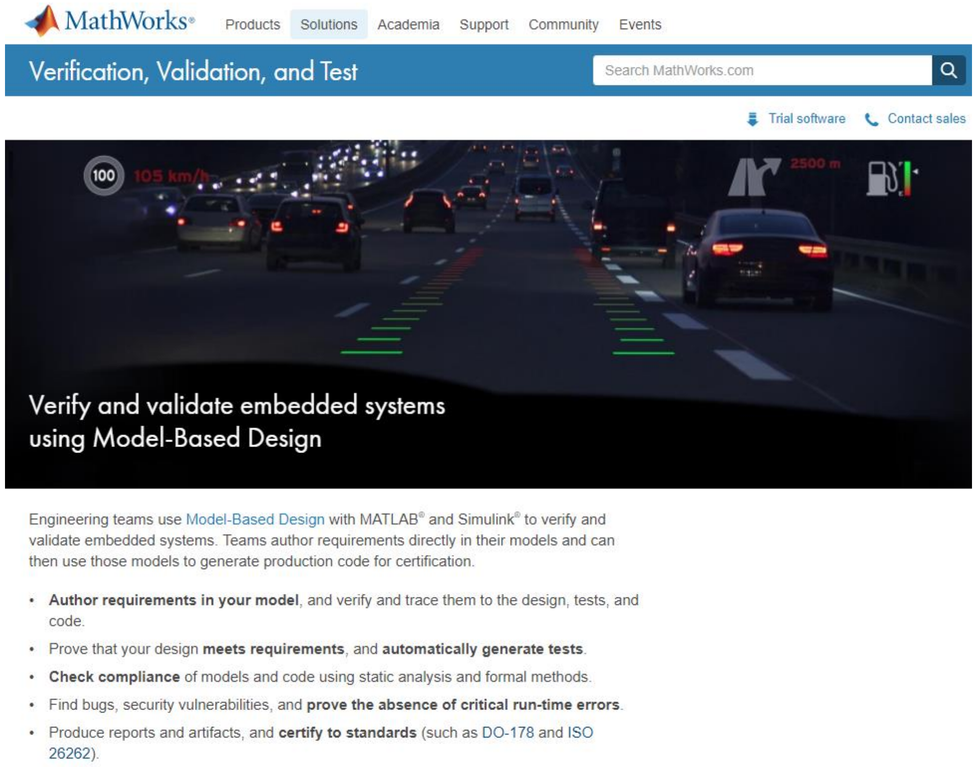
BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Primary flight control computers from BAE Systems.

Learn More

Visit MathWorks Verification, Validation and Test Solution Page:

mathworks.com/solutions/verification-validation.html



Verification, Validation, and Test

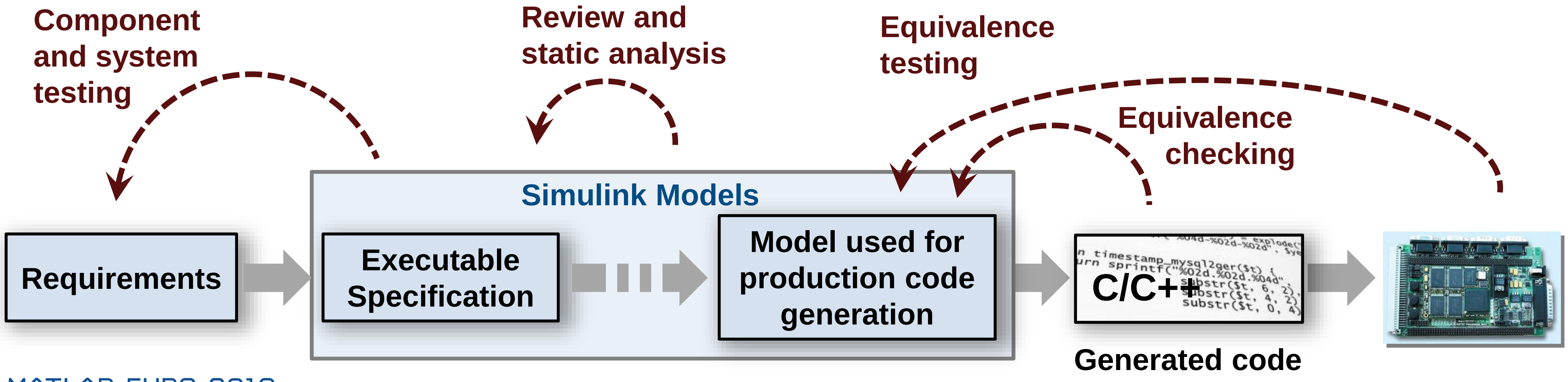
Verify and validate embedded systems using Model-Based Design

Engineering teams use Model-Based Design with MATLAB® and Simulink® to verify and validate embedded systems. Teams author requirements directly in their models and can then use those models to generate production code for certification.

- **Author requirements in your model**, and verify and trace them to the design, tests, and code.
- Prove that your design **meets requirements**, and **automatically generate tests**.
- **Check compliance** of models and code using static analysis and formal methods.
- Find bugs, security vulnerabilities, and **prove the absence of critical run-time errors**.
- Produce reports and artifacts, and **certify to standards** (such as DO-178 and ISO 26262).

Summary

1. Author and manage requirements within Simulink
2. Find defects earlier
3. Automate manual verification tasks
4. Reference workflow that conforms to safety standards


Component and system testing
Review and static analysis
Equivalence testing
Equivalence checking
Simulink Models
Requirements
Executable Specification
Model used for production code generation
C/C++
Generated code