

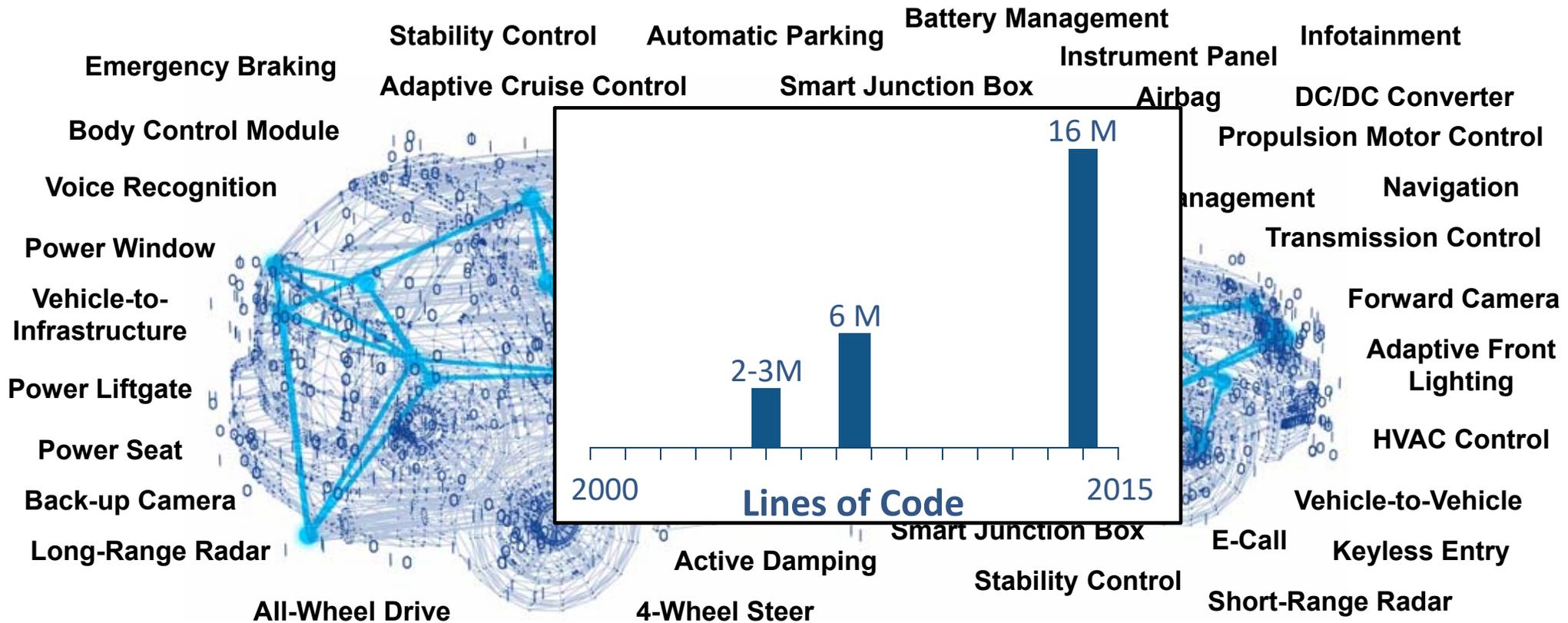
MATLAB EXPO 2018

Verification & Validation:
Automating Best Practices to
Improve Design Quality

Chuck Olosky



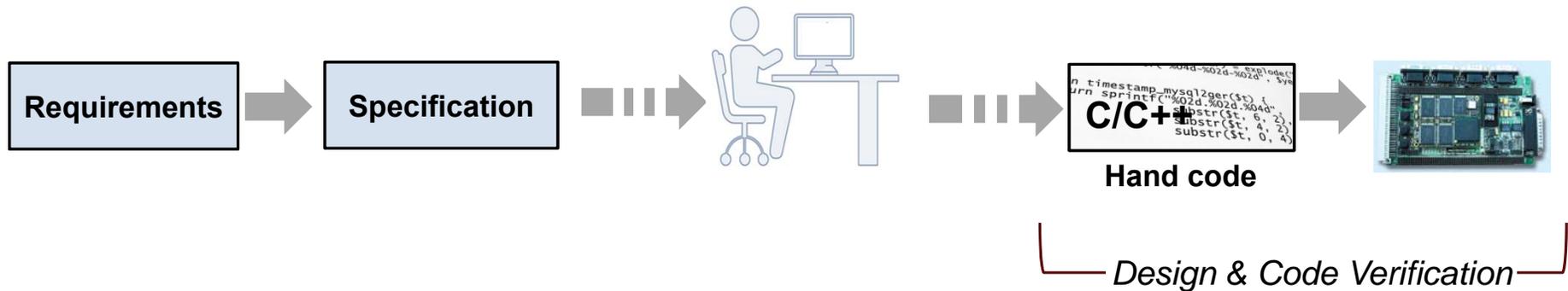
Growing Complexity of Embedded Systems



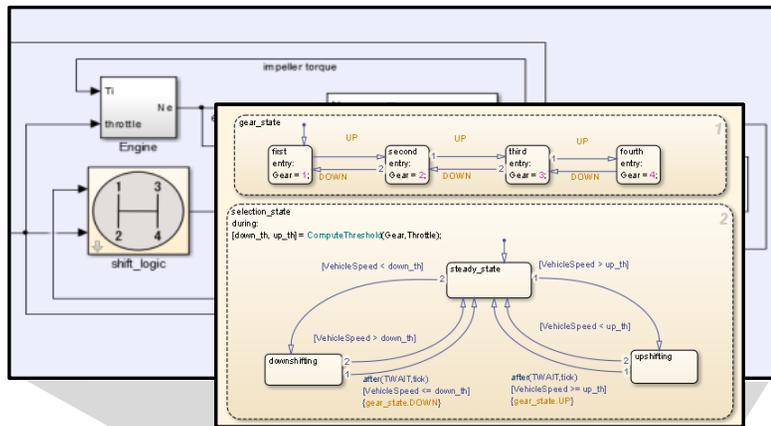
Siemens, "[Ford Motor Company Case Study](#)," Siemens PLM Software, 2014
 McKendrick, J. "[Cars become 'datacenters on wheels', carmakers become software companies.](#)" ZDJNet, 2013

Growing Complexity Challenges the Traditional Development Process

- Find requirements defects later in the process
- Find specification issues later in the process
- Find design issues later in the process



Using Simulink Models for Specification



Requirements

Executable Specification



```

timestamp_mysql2ger($s)
return sprintf("%02d.%02d.%04d",
    substr($t, 6, 2),
    substr($t, 4, 2),
    substr($t, 0, 4))
    
```

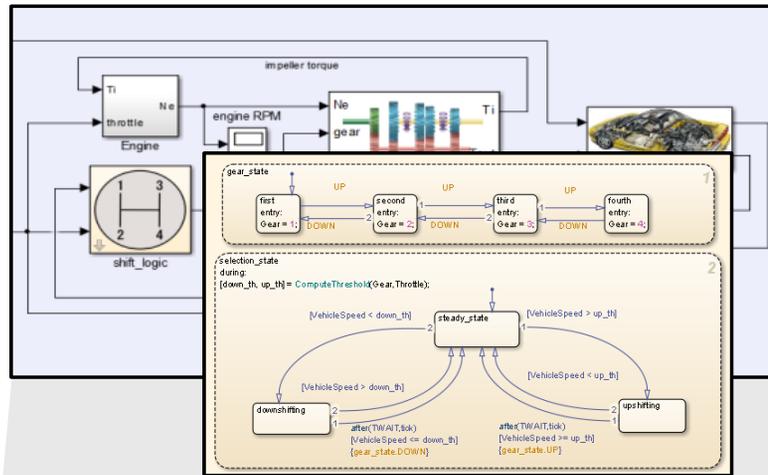
Hand Code



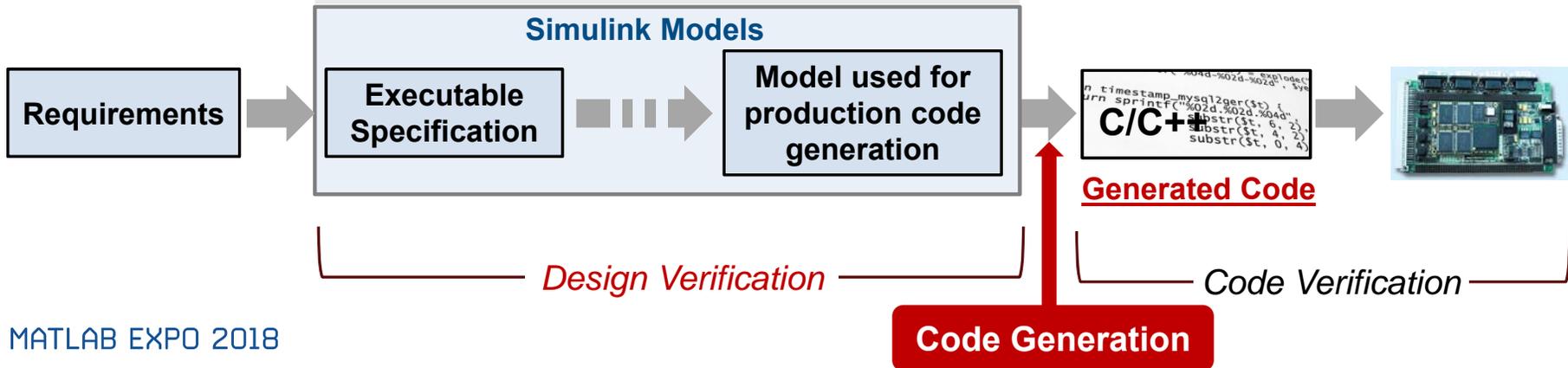
- Find requirements defects earlier in the process
- Find specification issues earlier in the process
- Find design issues later in the process

Design & Code Verification

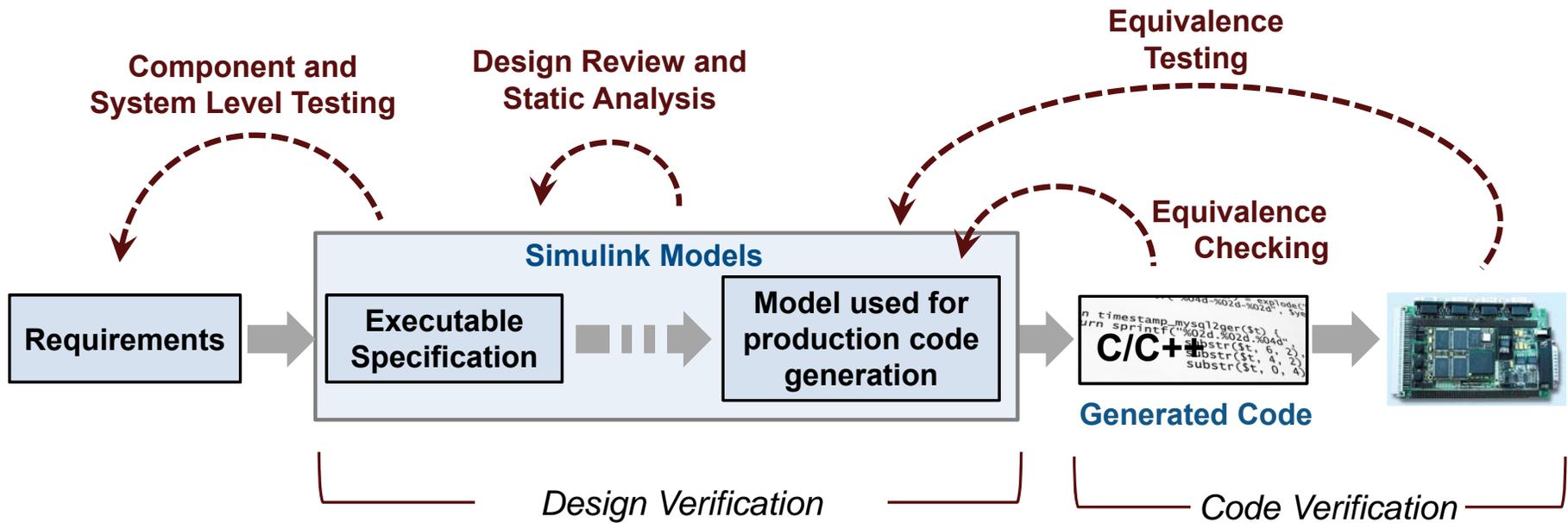
Complete Model Based Design



- Find requirements defects earlier in the process
- Find specification issues earlier in the process
- Find design issues earlier in the process



Model Based Design Verification Workflow



Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate static and dynamic verification
- Workflow that conforms to safety standards

“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

System Requirements

maximum machine velocity, left track
 maximum machine acceleration, left track
 maximum machine pitch, left track
 motor speed for 50% rise time, left track
 20% rise time, left track
 motor speed for 95% rise time, left track
 20% rise time, left track
 maximum machine velocity, right track
 maximum machine acceleration, right track
 maximum machine pitch, right track
 motor speed for 50% rise time, right track

Verified & Validated System



High Level Design

Integration Testing

Detailed Design

Unit Testing

Coding

Why do 71% of Embedded Projects Fail?

Poor Requirements Management

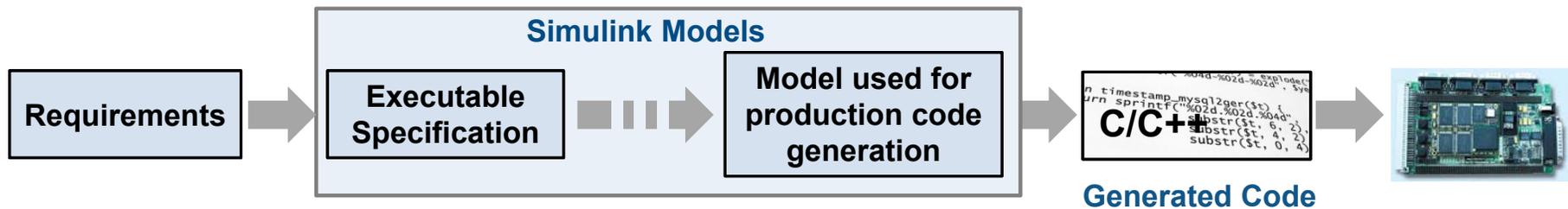
Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005

Challenges with Requirements

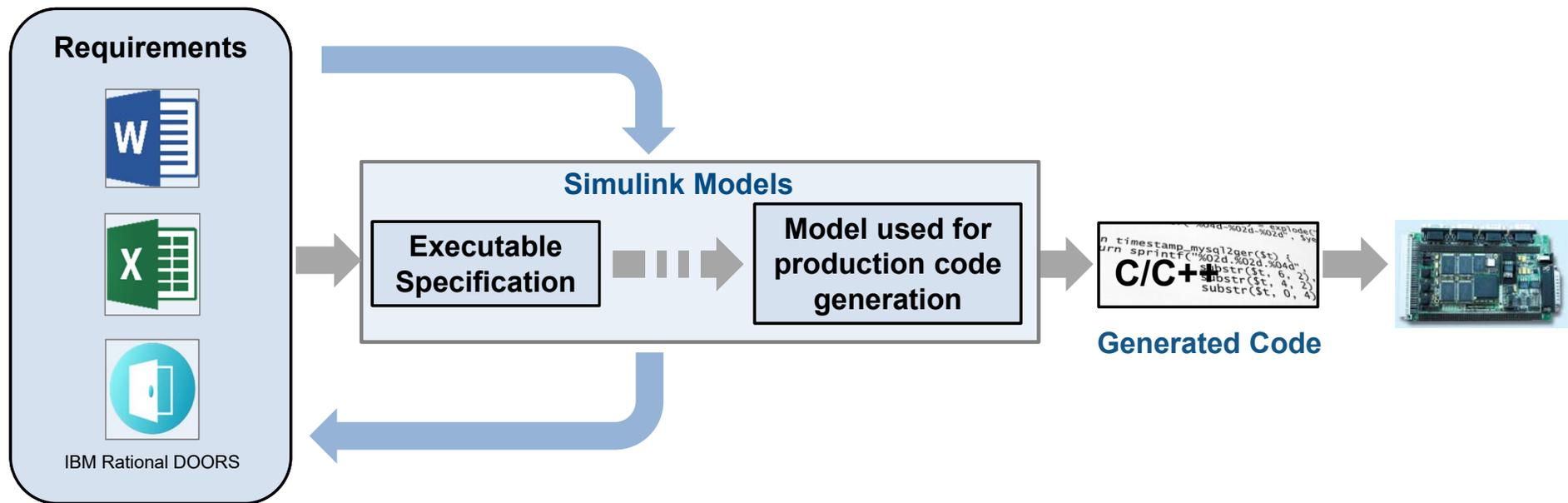
Where are they implemented?

Are they consistent with the design?

How are they tested?



Gap Between Requirements and Design



Simulink Requirements

R2017b

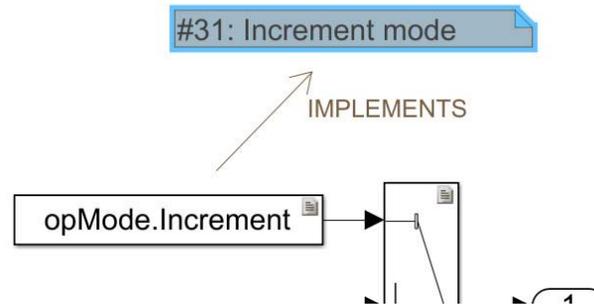
Link & Track

Author

Summary: Cancel Switch Detection

Description	Rationale
If the Cancel switch is pressed, the value of <i>reqDrv</i> should be set to <i>reqMode.Cancel</i> .	

Dashboard Image

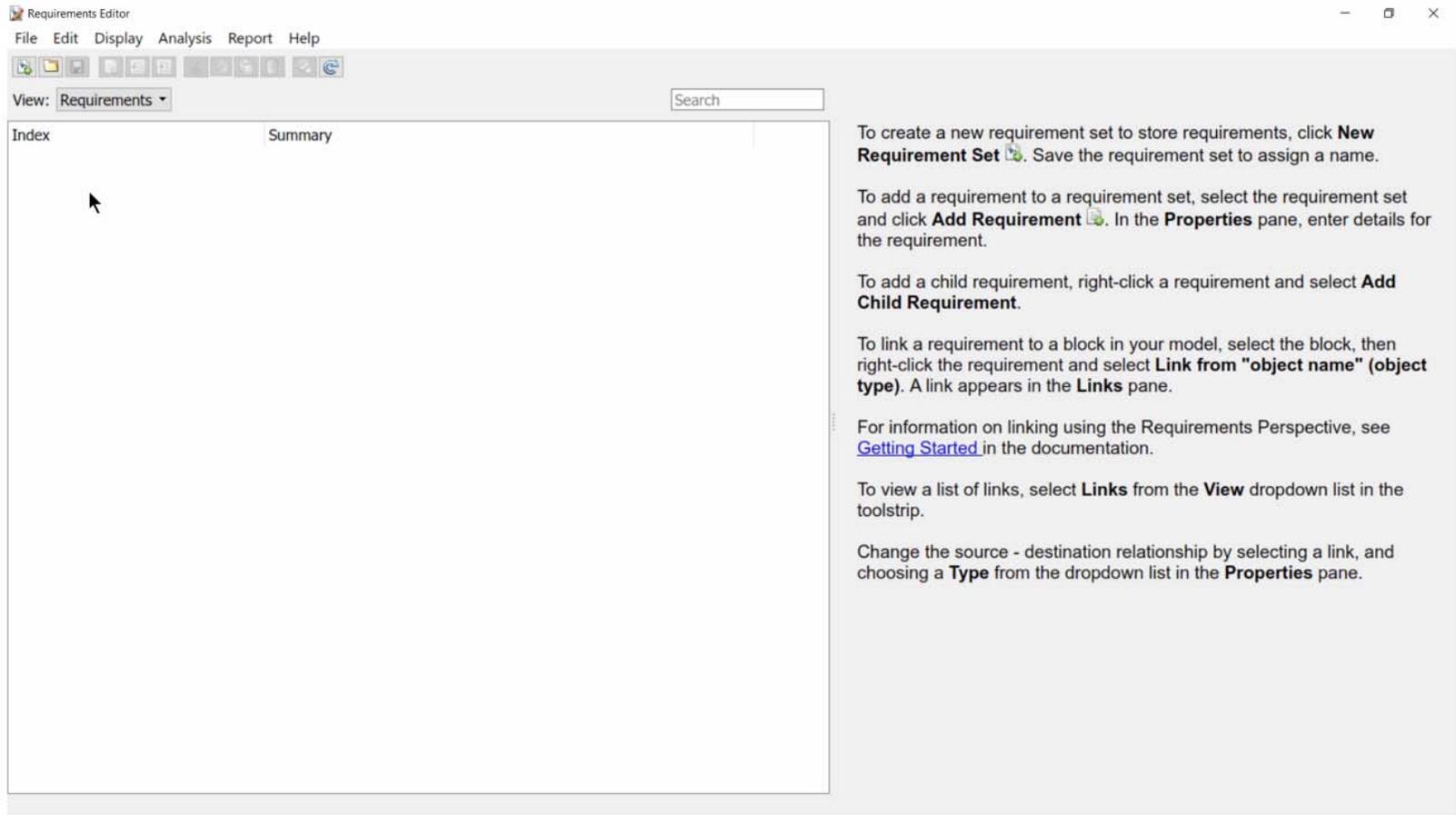
Manage Updates

Issue: Destination Changed.

Stored:	Revision: 15
Actual:	Revision: 18

Clear Issue

Requirements Editor



Requirements Editor

File Edit Display Analysis Report Help

View: Requirements Search

Index Summary

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**.

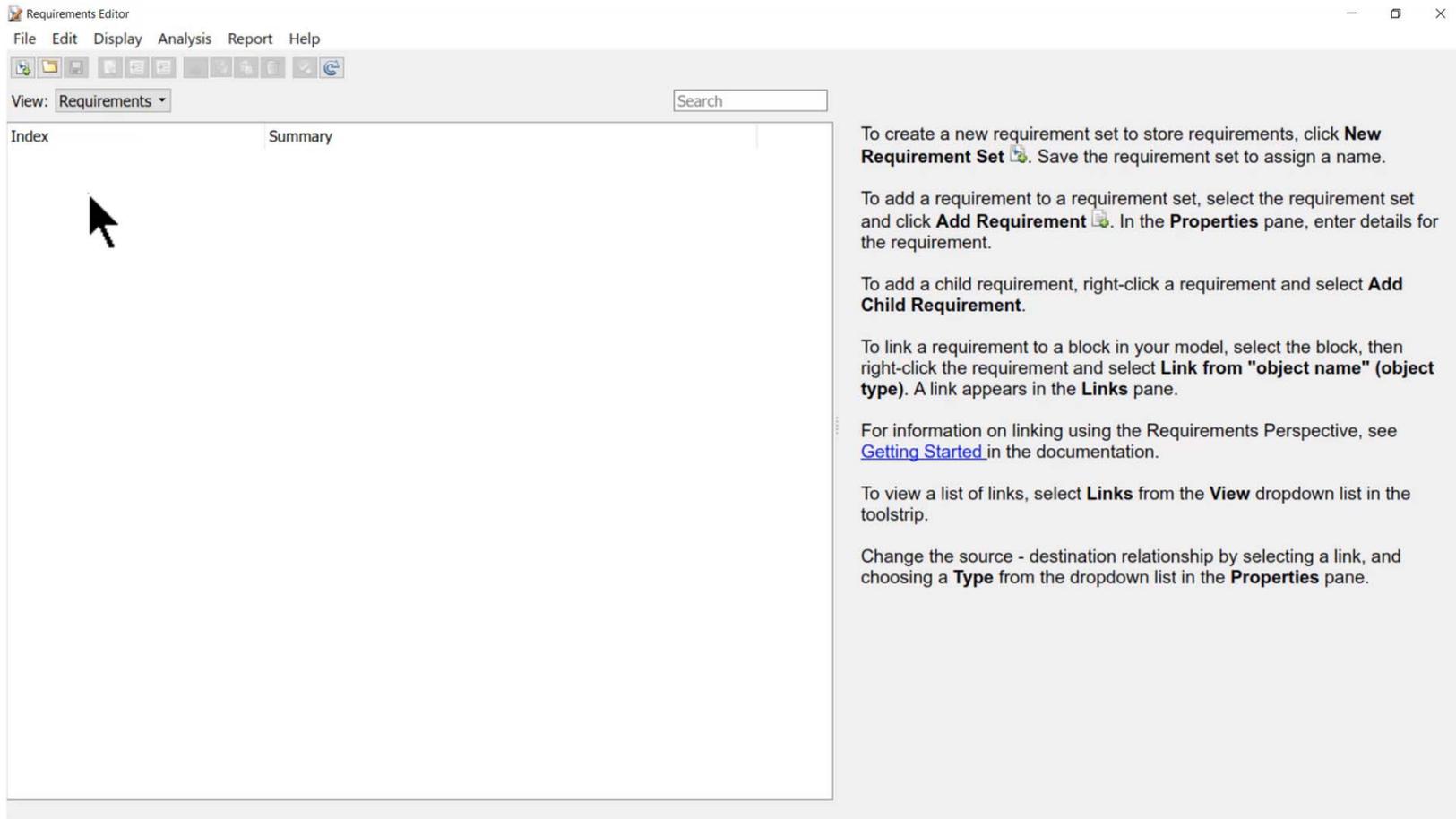
To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

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 toolbar.

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

Requirements Editor



Requirements Editor

File Edit Display Analysis Report Help

View: Requirements Search

Index Summary

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 toolbar.

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

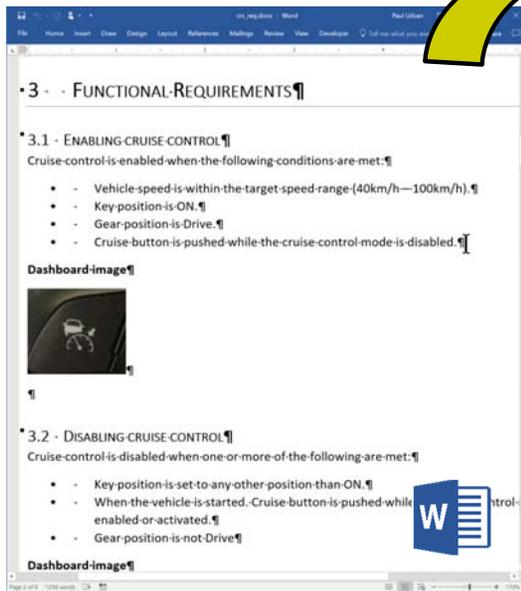
Import Requirements from External Sources

Import

Simulink Requirements Editor

IBM Rational DOORS

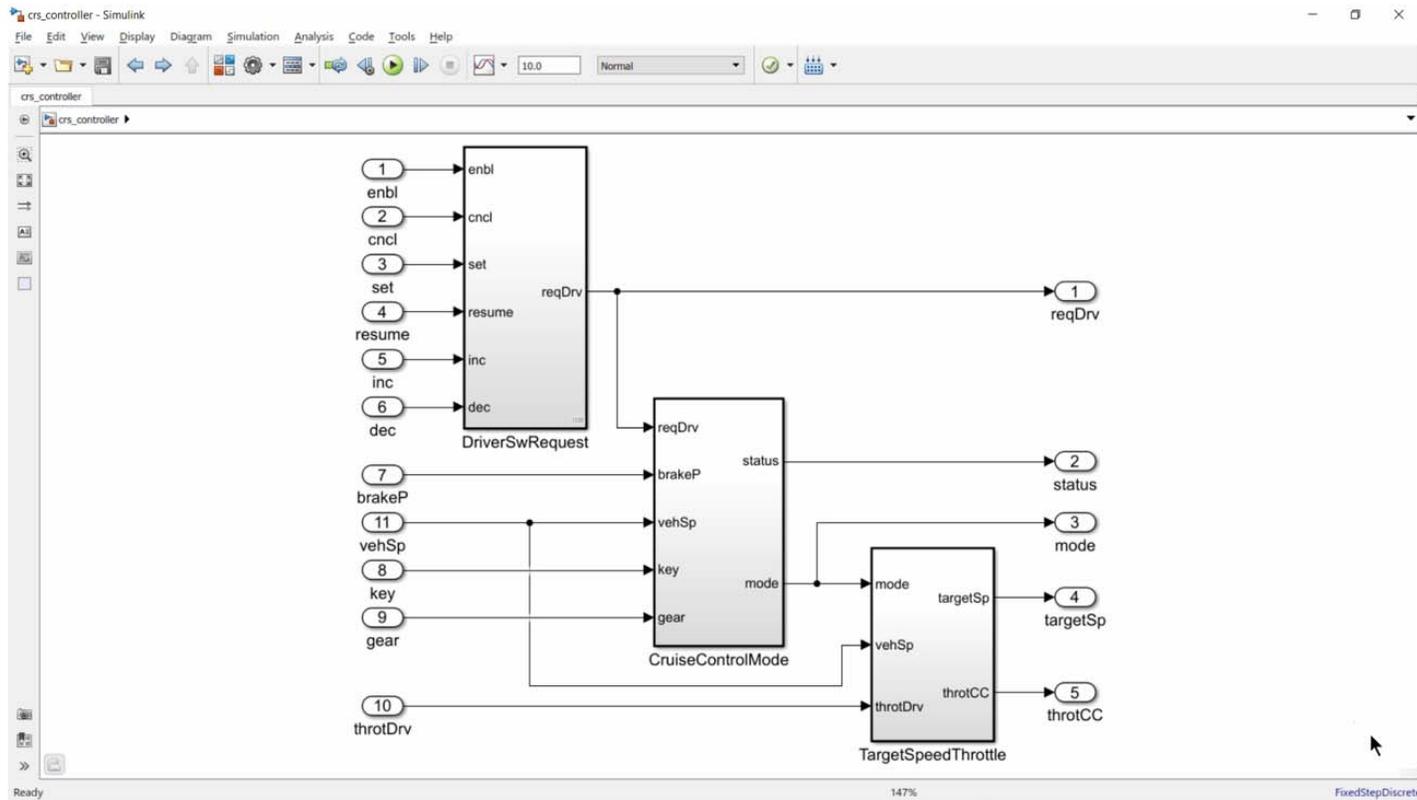
ReqIF R2018a



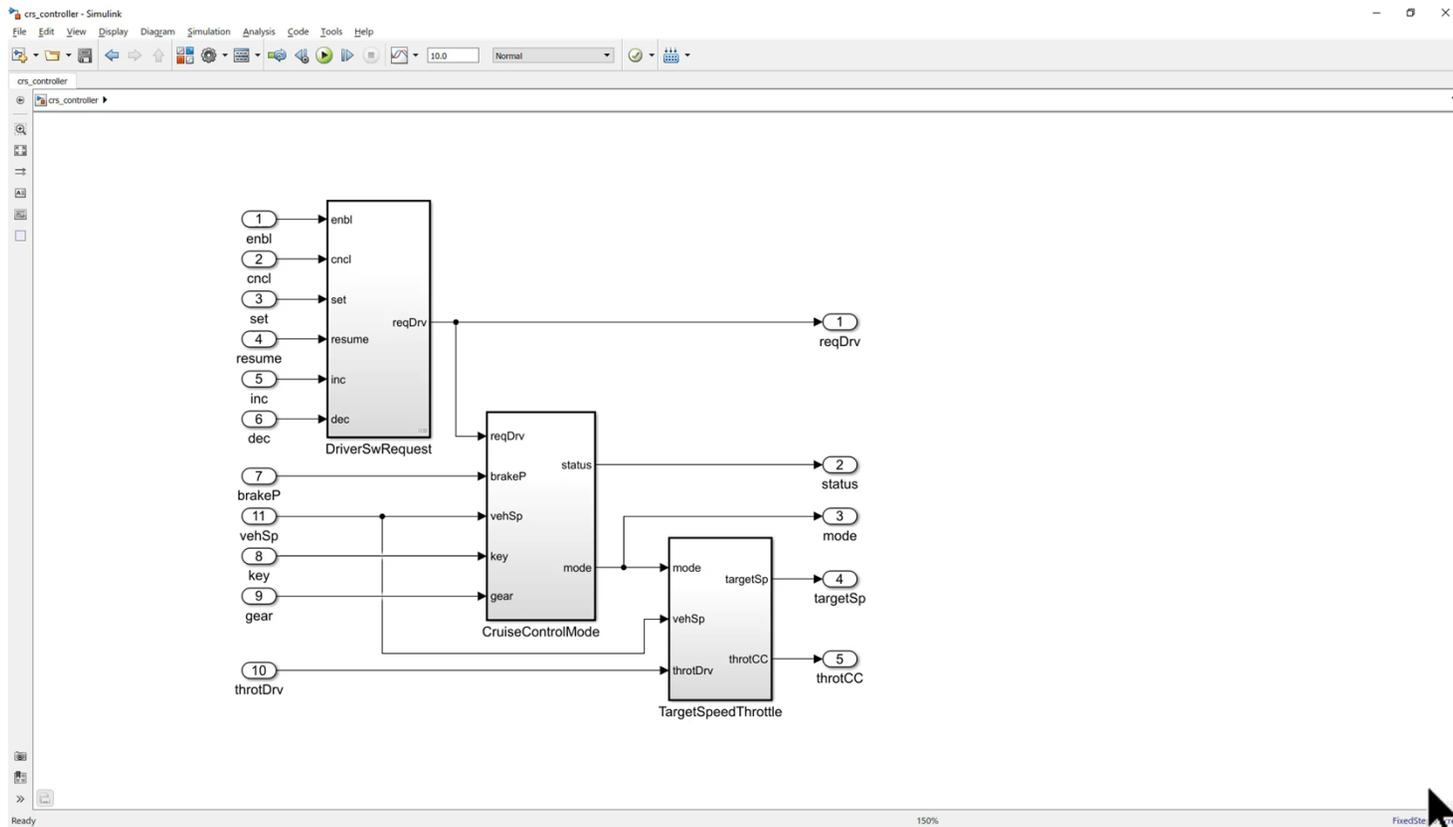
Index	ID	Summary
crs_req		References to crs_req.docx
1	1	Overview
1.1	1	Overview This document describes a
1.2	2	System overview
1.2.1	2.1	System inputs
1.2.1.1	2.1.1	Cruise control buttons
1.2.1.2	2.1.2	Other inputs
1.2.2	2.2	Cruise control mode indi...
1.2.3	2.3	Cruise control modes
1.3	3	Functional Requirements
1.3.1	3.1	Enabling cruise control
1.3.2	3.2	Disabling cruise control
1.3.3	3.3	Activating cruise control
1.3.4	3.4	Deactivating cruise control
1.3.5	3.5	Target Speed Increment
1.3.6	3.6	Target speed decrement
1.3.7	3.7	Successive Target Speed...
1.3.8	3.8	Successive Target Speed...
1.3.9	3.9	Adjusting Target Speed ...
1.3.10	3.10	Resuming cruise control
1.3.11	3.11	Throttle value calculation
1.3.12	3.12	Cruise Control SET Indi...
1.4	4	Interface specification

Show in document

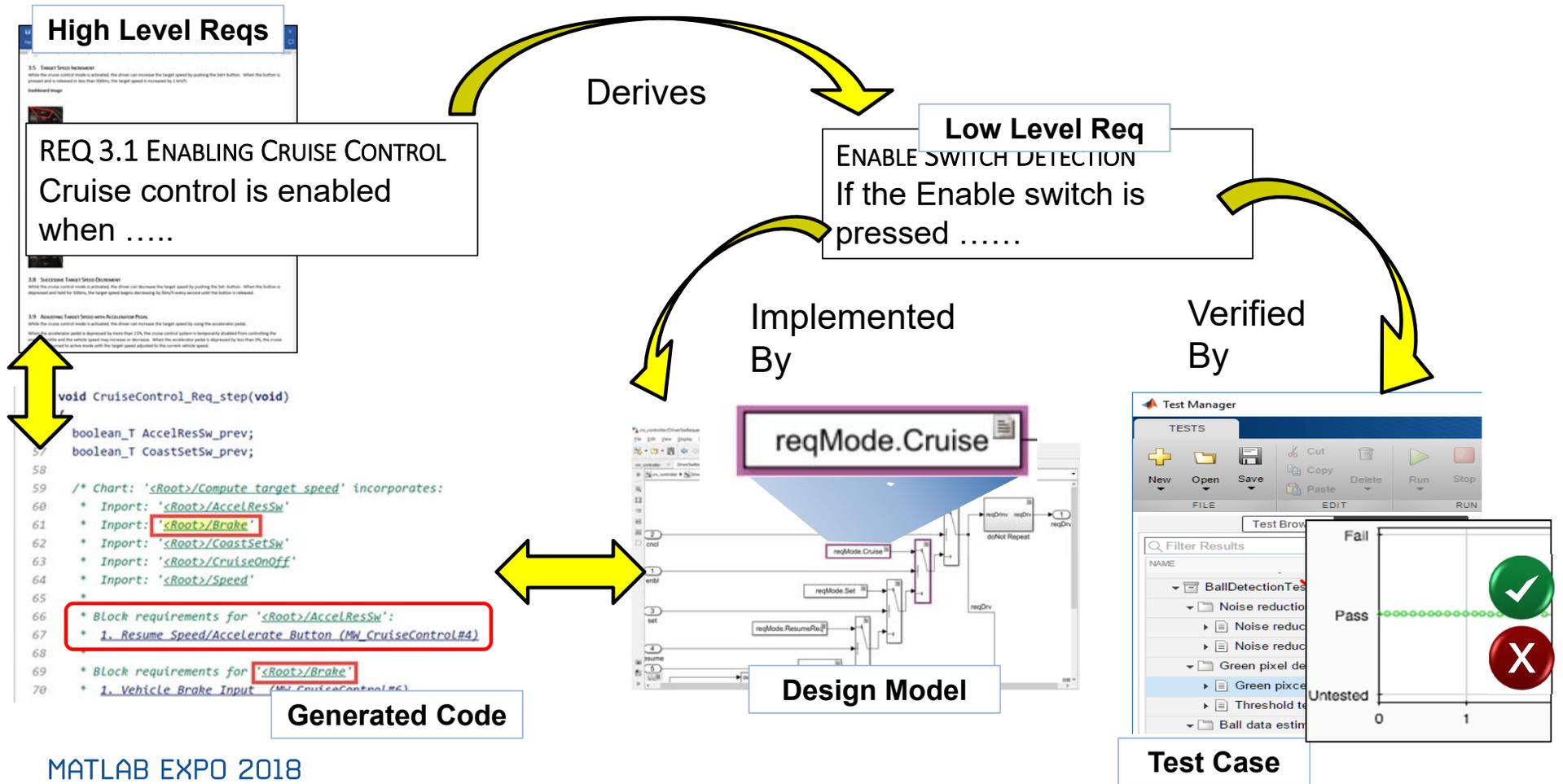
Requirements Perspective View of Model



Requirements Perspective View of Model



Requirements Traceability



Track Implementation and Verification Status

Requirements - crs_controller

View: Requirements

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

Implementation Status

- Implemented
- Justified
- Missing

Verification Status

- Passed
- Failed
- No Result
- Missing

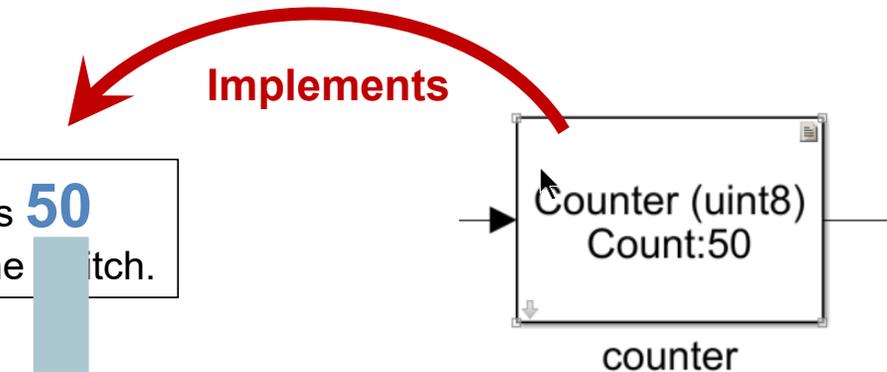
Respond to Requirements Change

Original Requirement

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

Updated Requirement

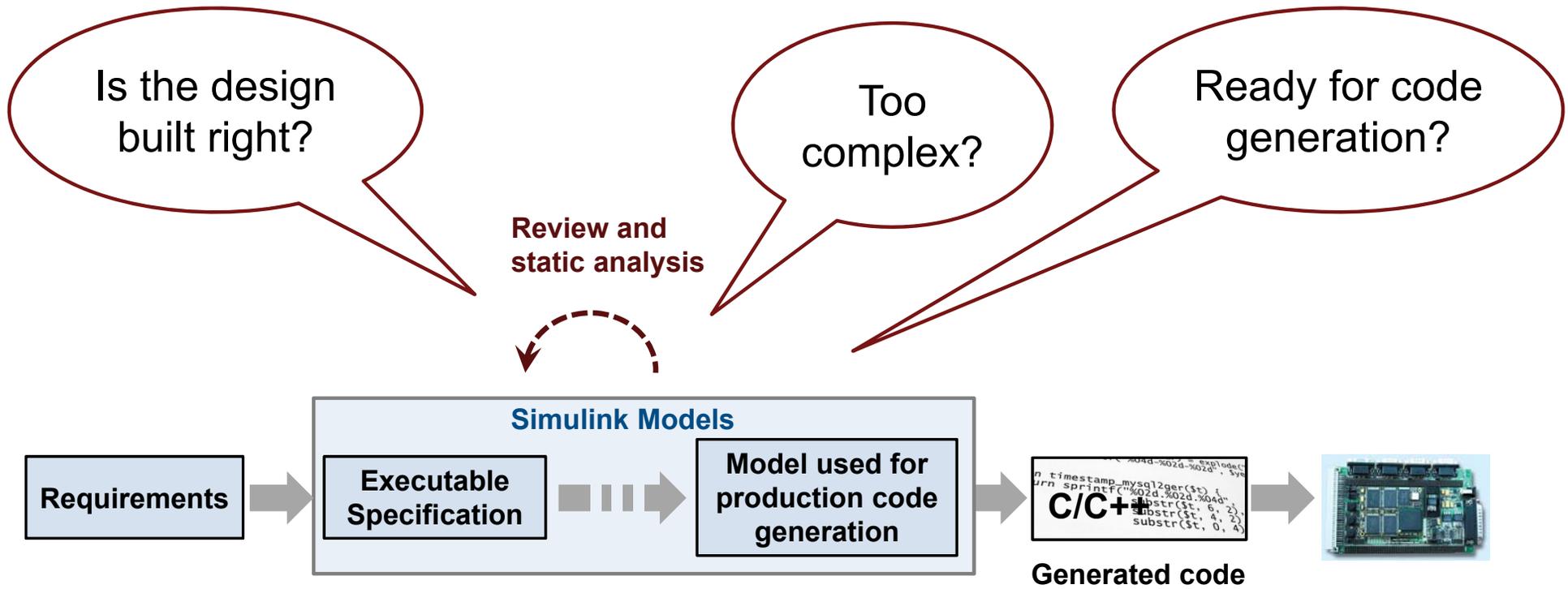
If the switch is pressed and the counter reaches **75** then it shall be recognized as a long press of the switch.



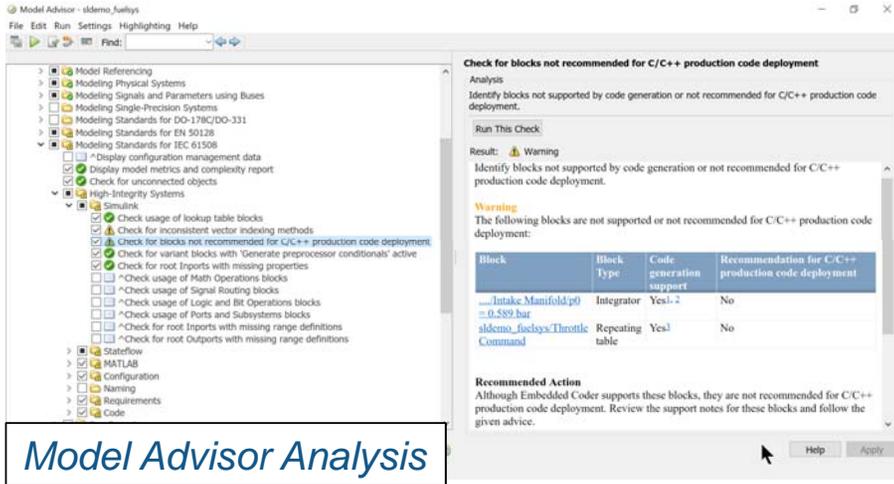
← **Implemented by:**
 counter

 **Issue: Destination Changed.**

Verify Design to Guidelines and Standards

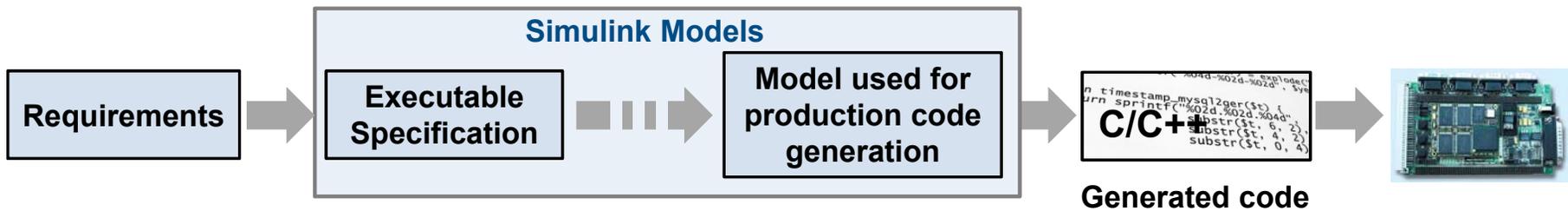


Automated Static Analysis of the Design



Check the model for

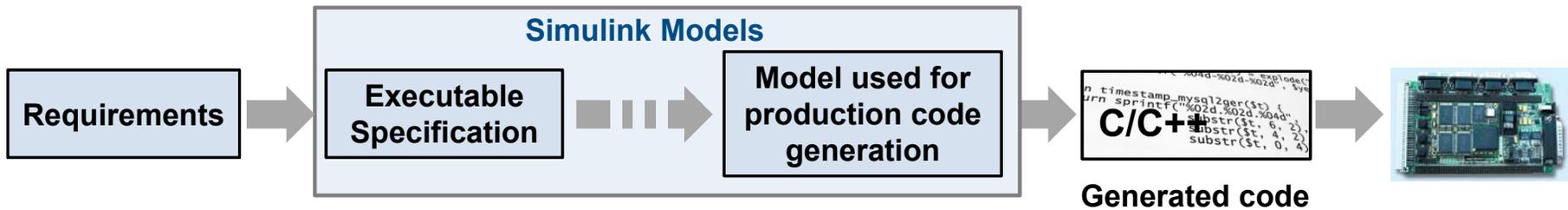
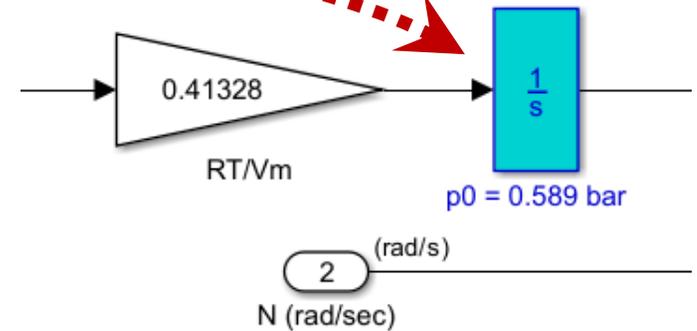
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more.....



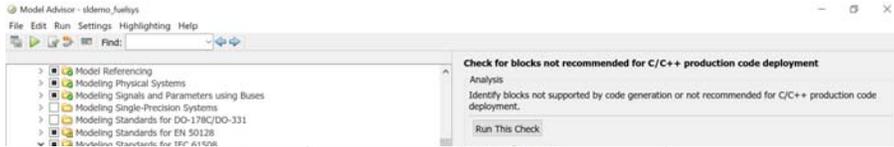
Navigate to Problematic Blocks

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

Model Advisor Analysis



Guidance Provided to Address Issues w/ Auto-Correct

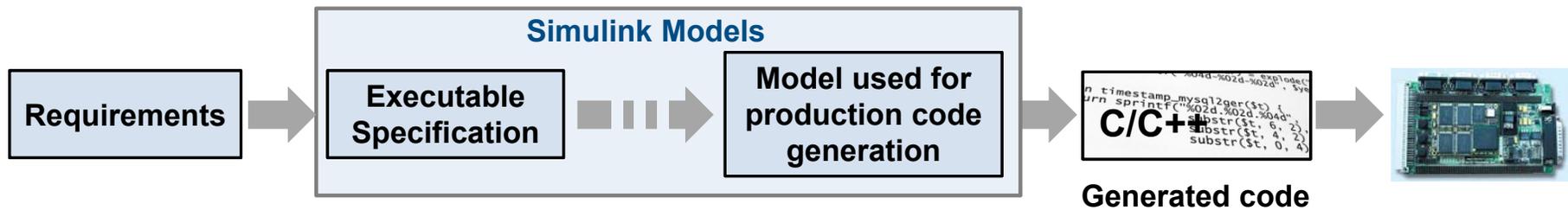


Recommended Action

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.



Model Advisor Analysis



Modeling Guidelines for High-Integrity Systems

- Leverage industry-best practices and MathWorks tool expertise when developing high-integrity systems
- Modeling Guidelines with corresponding Model Advisor checks
- Mapped to the modeling standards and guidelines objectives of industry standards like ISO 26262 and MISRA-C

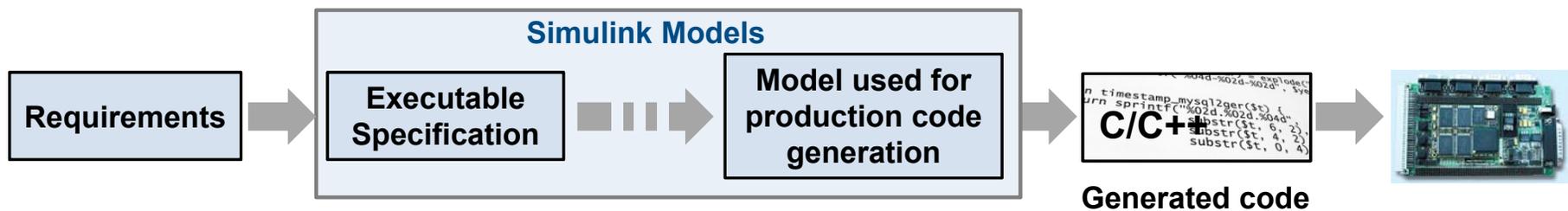
Simulink®
Modeling Guidelines for High-Integrity Systems

hisl_0007: Usage of For Iterator or While Iterator subsystems

ID: Title	hisl_0007: Usage of For Iterator or While Iterator subsystems
Description	To support unambiguous behavior, when using For Iterator Subsystem or While Iterator Subsystem, avoid using sample time-dependent blocks, such as integrators, filters, and transfer functions within the subsystems.
Rationale	Avoid ambiguous behavior from the subsystem.
Model Advisor Checks	<ul style="list-style-type: none"> • By Task > Modeling Standards for DO-178C/DO-331 > High-Integrity Systems > Simulink > Check sample time-dependent blocks • By Task > Modeling Standards for IEC 61508 > High-Integrity Systems > Simulink > Check sample time-dependent blocks • By Task > Modeling Standards for IEC 62304 > High-Integrity Systems > Simulink > Check sample time-dependent blocks • By Task > Modeling Standards for ISO 26262 > High-Integrity Systems > Simulink > Check sample time-dependent blocks • By Task > Modeling Standards for EN 50128 > High-Integrity Systems > Simulink > Check sample time-dependent blocks <p>For check details, see Check sample time-dependent blocks.</p>
References	<ul style="list-style-type: none"> • DO-331, Section MB.6.3.1.e 'High-level requirements conform to standards' • DO-331, Section MB.6.3.2.e 'Low-level requirements conform to standards' • Sections MB.6.3.1.g and MB.6.3.2.g 'Algorithms are accurate' • IEC 61508-3, Table A.3 (3) 'Language subset' • IEC 61508-3, Table A.4 (3) 'Defensive programming' • IEC 62304, 5.5.3 - Software Unit acceptance criteria • ISO 26262-6, Table 1 (1b) 'Use of language subsets' • ISO 26262-6, Table 1 (1d) 'Use of defensive implementation techniques' • EN 50128, Table A.4 (11) 'Language Subset' • EN 50128, Table A.3 (1) 'Defensive Programming' • MISRA C:2012, Rule 14.2 • MISRA C:2012, Rule 16.4 • MISRA C:2012, Dir 4.1
Last Changed	R2018b

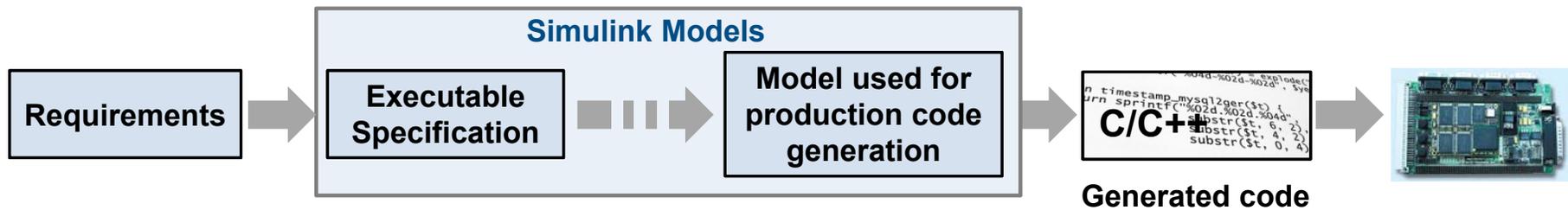
Built in Checks for Industry Standards and Guidelines

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



Configure and Customize Static 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



Generate Reports for Reviews and Documentation

Model Advisor Analysis

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 (ps)	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 Reports

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

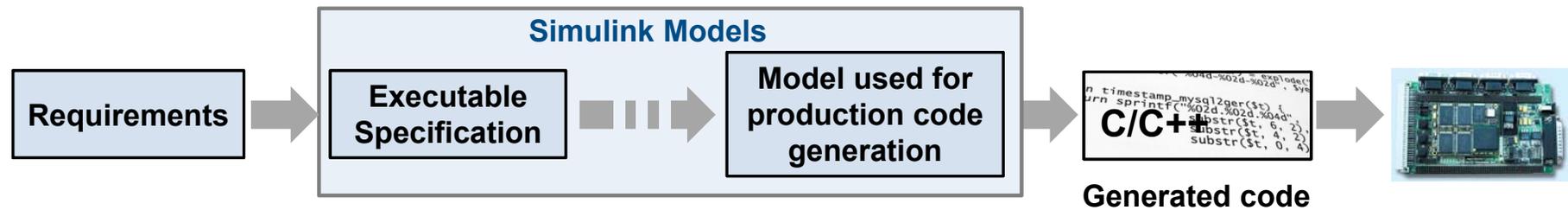
By Task

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

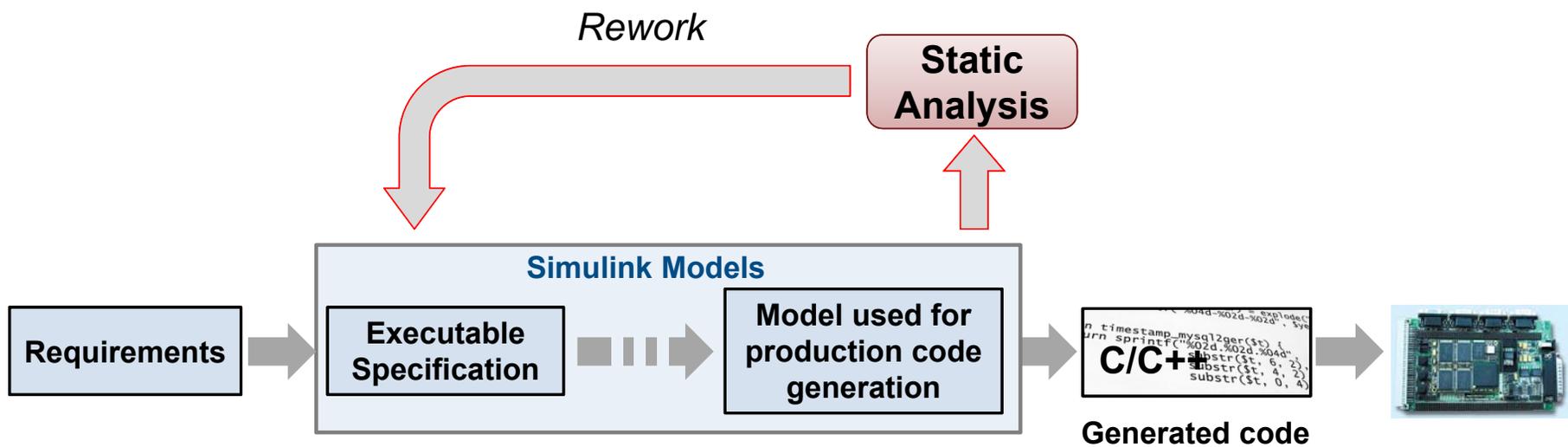
Check optimization settings

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

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

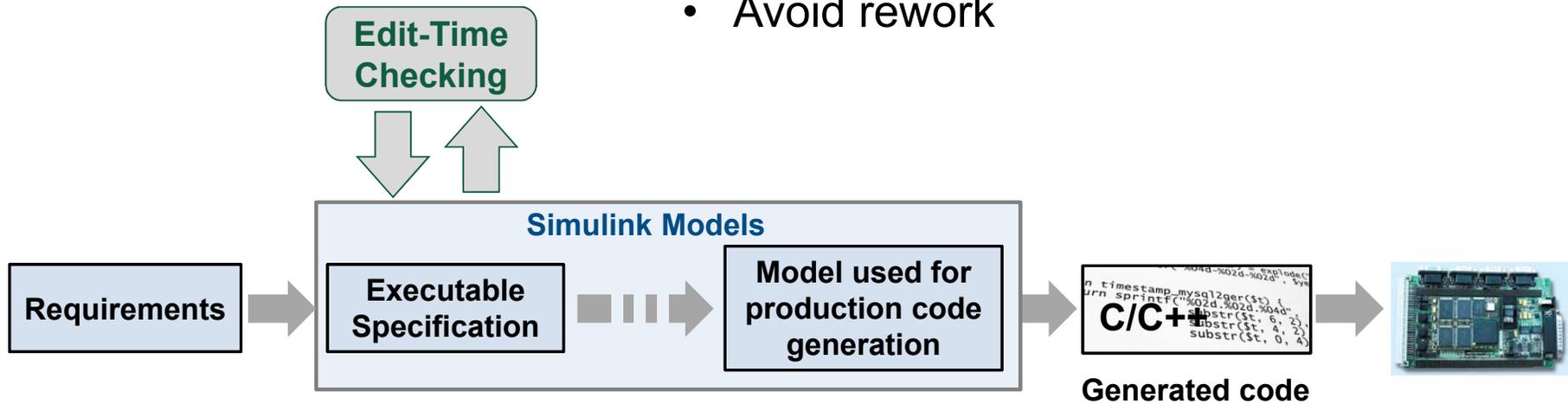


Checks for Standards and Guidelines are often Performed Late

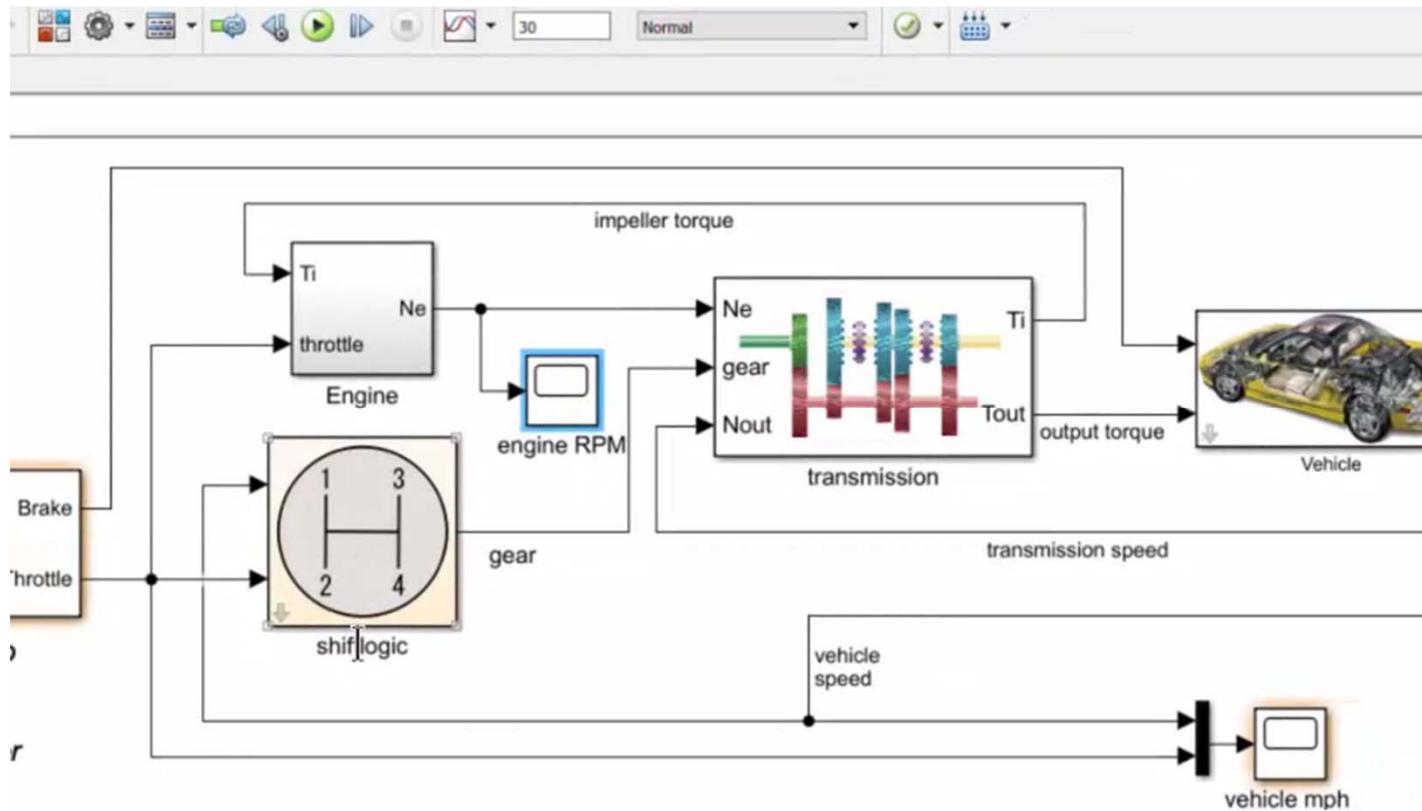


Shift Verification Earlier with Edit-Time Checking

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



Find Compliance Issues while you Design



Modeling Standards Checking with Simulink Check

Advisor Report - CruiseControl_Check.slx

Model version: 1.455
 System: CruiseControl_Check
 Current run: 09-Oct-2018 19:46:10
 Treat as Referenced Model: off

Run Summary

Pass	Fail	Warning	Not Run	Total
0	0	1	0	1

Warning
 Check Stateflow charts for strong data typing
 Identify expressions with variables and parameters of different data types in Stateflow objects.

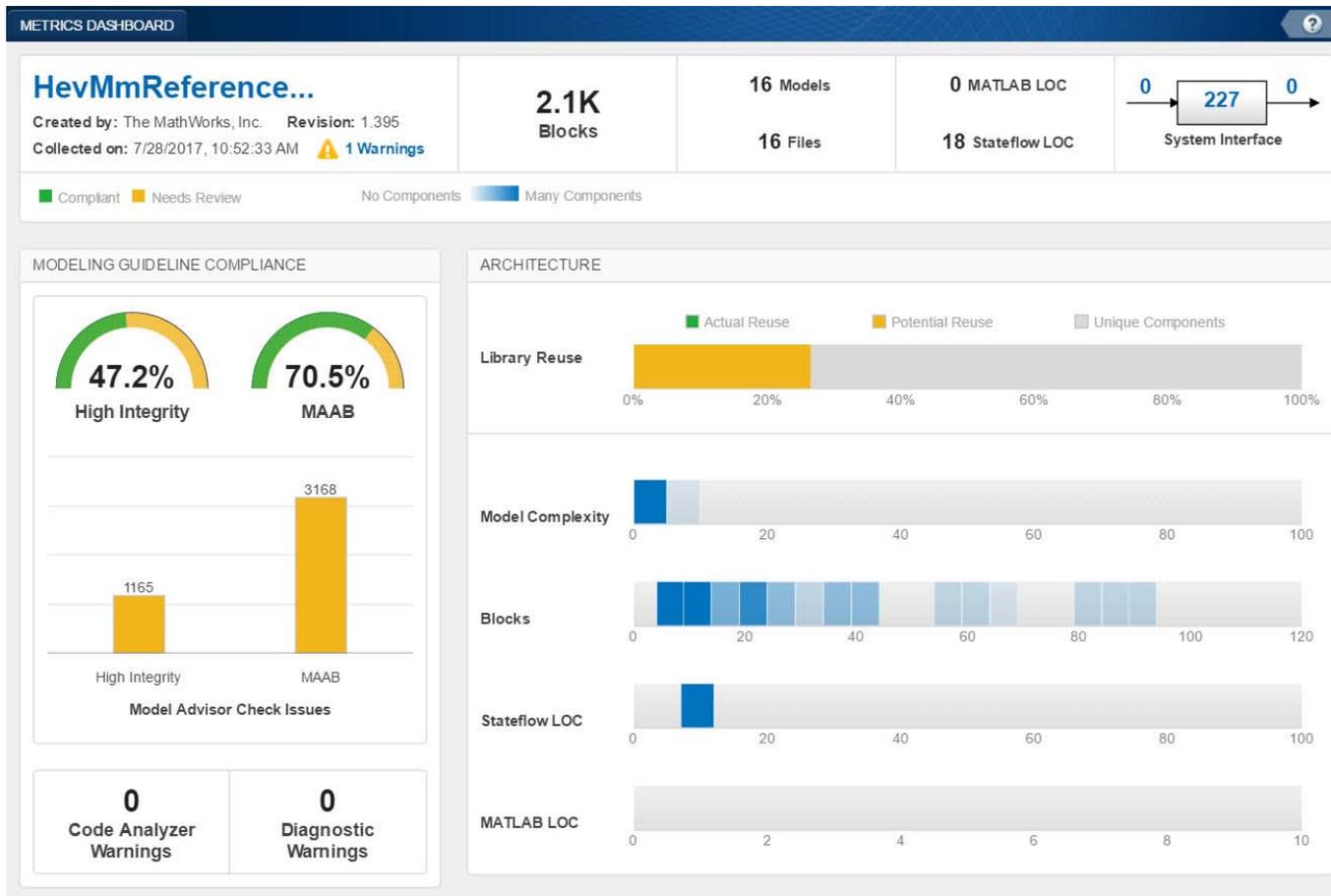
The following expressions consist of variables and parameters of different datatypes.

Expressions	State or Transition
tspeed = 0;	CruiseControl_Check/Compute target speed/OFF
&& tspeed!=0	CruiseControl_Check/Compute target speed/CRUISE

Recommended Action
 Revisit expressions listed above to avoid operations with different data types.

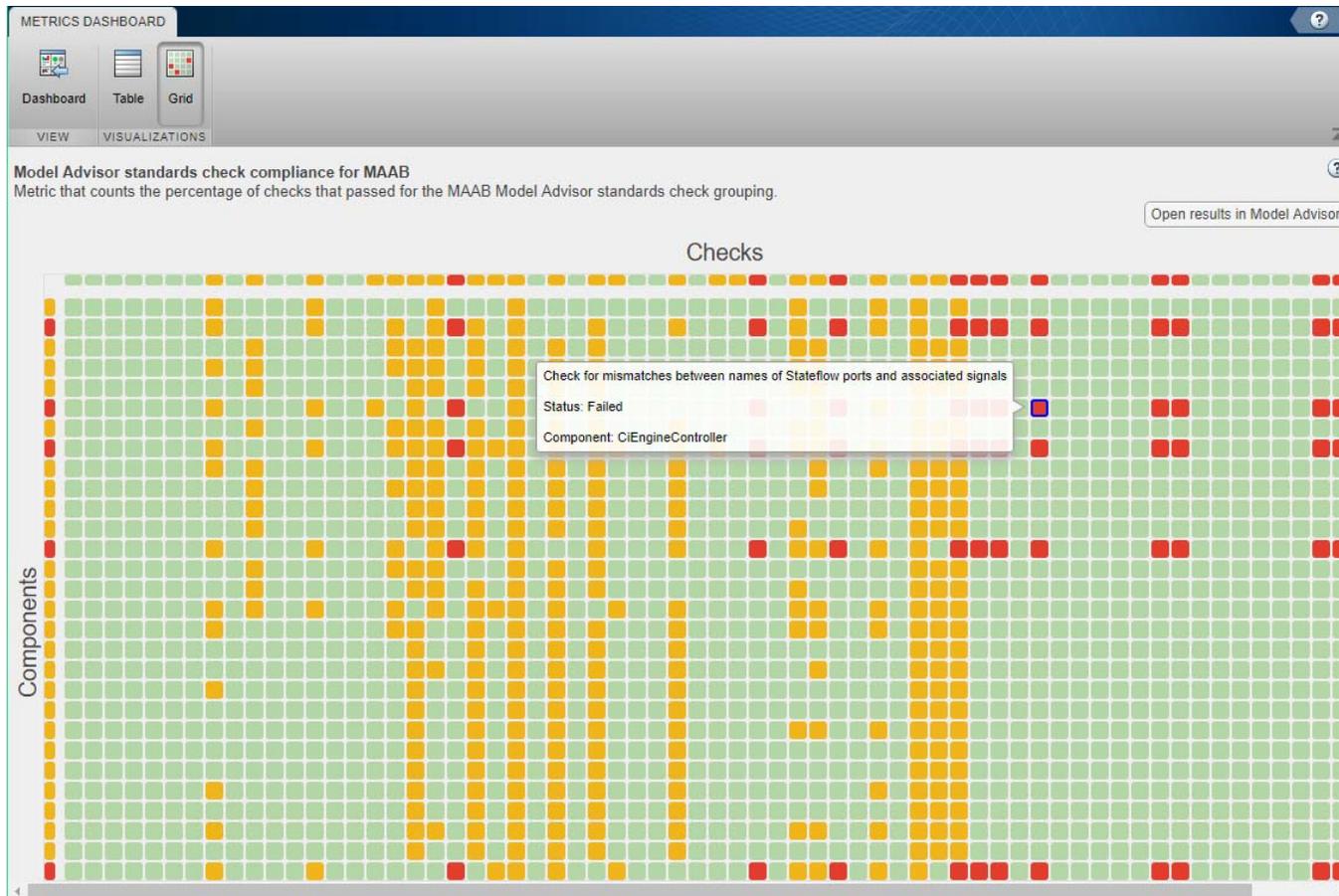
- Static analysis of models against a set of checks
- Modeling Standards Checks
 - MAAB Style Guidelines V3.0
 - ISO 26262
 - MISRA C:2012
- Additional Checks
 - Model Metrics
 - Tool Bug Reports (Cert Kit)
 - Requirements Consistency

Assess Quality with Metrics Dashboard



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

Grid View for Metrics Analysis

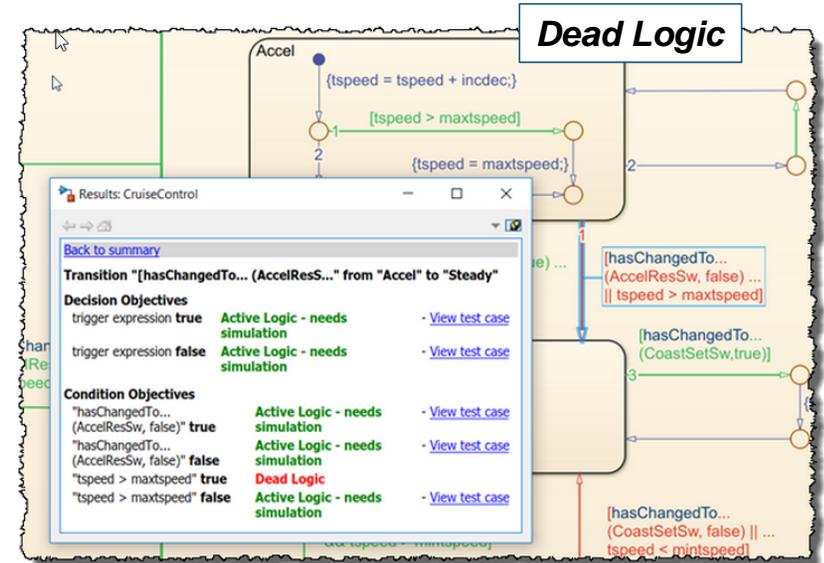
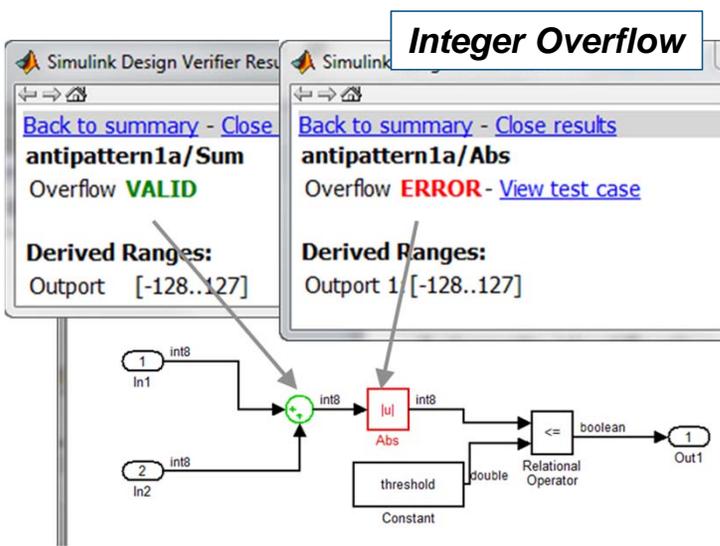


- Visualize Standards Check Compliance
 - Find Issues
 - Identify patterns
 - See hot spots

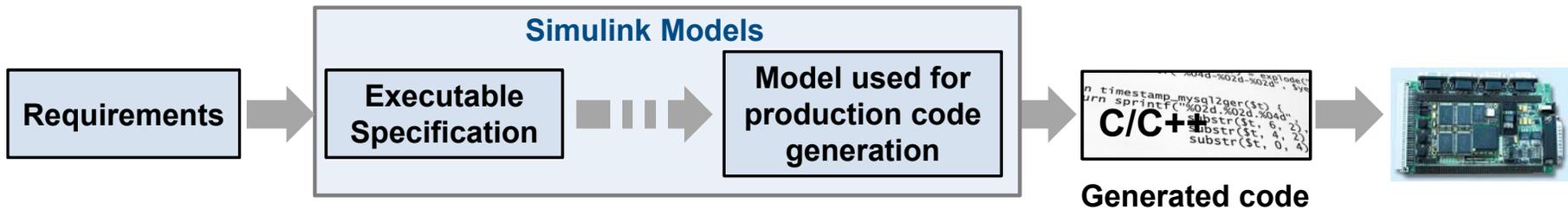
Legend:

- Red: Fail
- Orange: Warning
- Green: Pass
- Gray: Not run

Static Analysis for Detecting Design Errors



Static Analysis



- Find run-time design errors
- Generates a test case to reproduce the issue for debugging

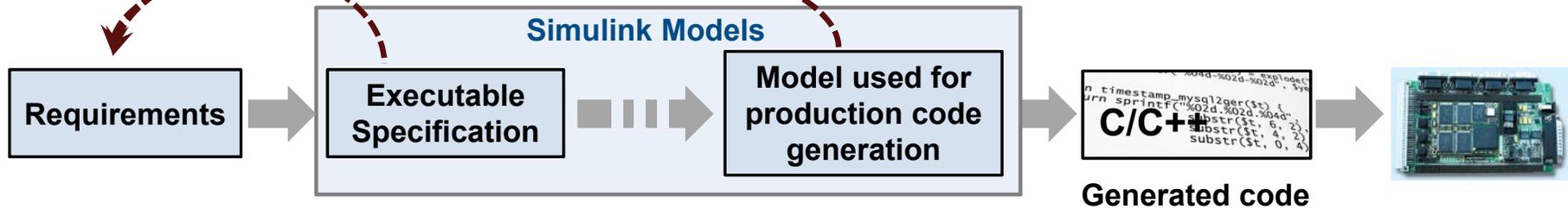
Functional Testing

Does the design meet requirements?

Is it functioning correctly?

Is it completely tested?

Component and System Level Testing



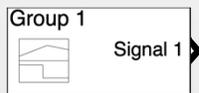
Systematic Functional Testing

Test Case

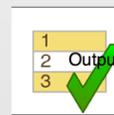
Inputs



MAT file (input)



Signal Builder



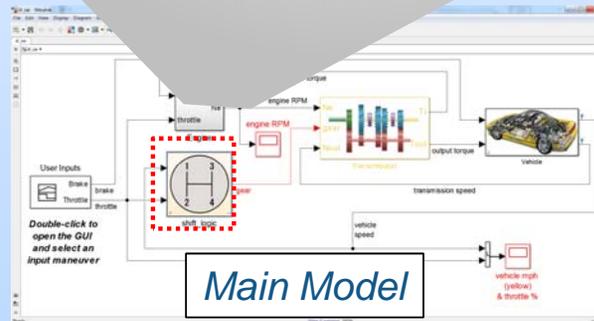
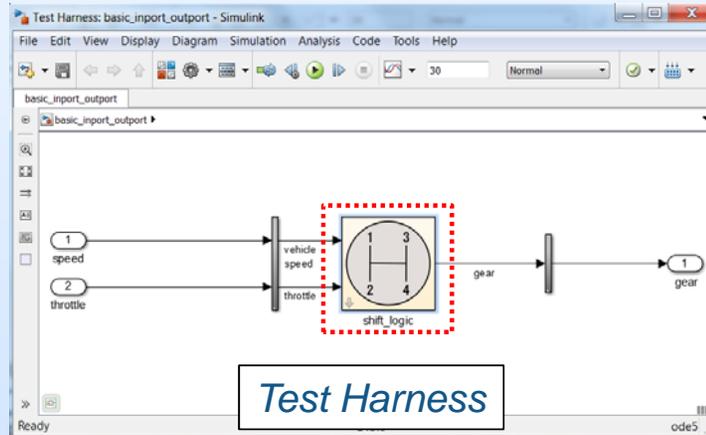
Test Sequence

and more!



Excel file (input)

R2017b



Assessments



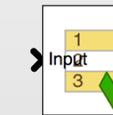
MAT file (baseline)

function customCriteria

Perform custom criteria

```
1 test.verifyThat(test.sl
```

MATLAB Unit Test



Test Assessment

and more!



Excel file (baseline)

R2017b

Test Execution and Results Analysis

Test Manager

TESTS

FILE EDIT RUN RESULTS RESOURCES

Test Browser Results and Artifacts

Filter Tests

- Component Testing
 - 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

BASILINE CRITERIA

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

PROPERTY VALUE

Name	Slow Accel
Type	Baseline Test
Location	C:\Users\moneh\Deskto...
Enabled	<input checked="" type="checkbox"/>
Hierarchy	ComponentTesting > Fu...
Model	sf_car
Simulation Mode	[Model Settings]
Harness Name	SigBdriven

Test Manager

TESTS VISUALIZE FORMAT

Clear Plot Data Cursors Highlight in Model Send to Figure

EDIT ZOOM & PAN MEASURE & TRACE SHARE

Test Browser Results and Artifacts

Filter Results

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

Start Page Slow Accel Comparison

Baseline Compare To

fourth
third
second
first
None

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

Tolerance Difference

1.0
0.8
0.6
0.4
0.2
0

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

Coverage Analysis to Measure Test Completeness

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

Simulink

Stateflow

Generated Code

Transition "UP" from "third"

UP was never true.

[speed < up_th]

Decisions analyzed:

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

Conditions analyzed:

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

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

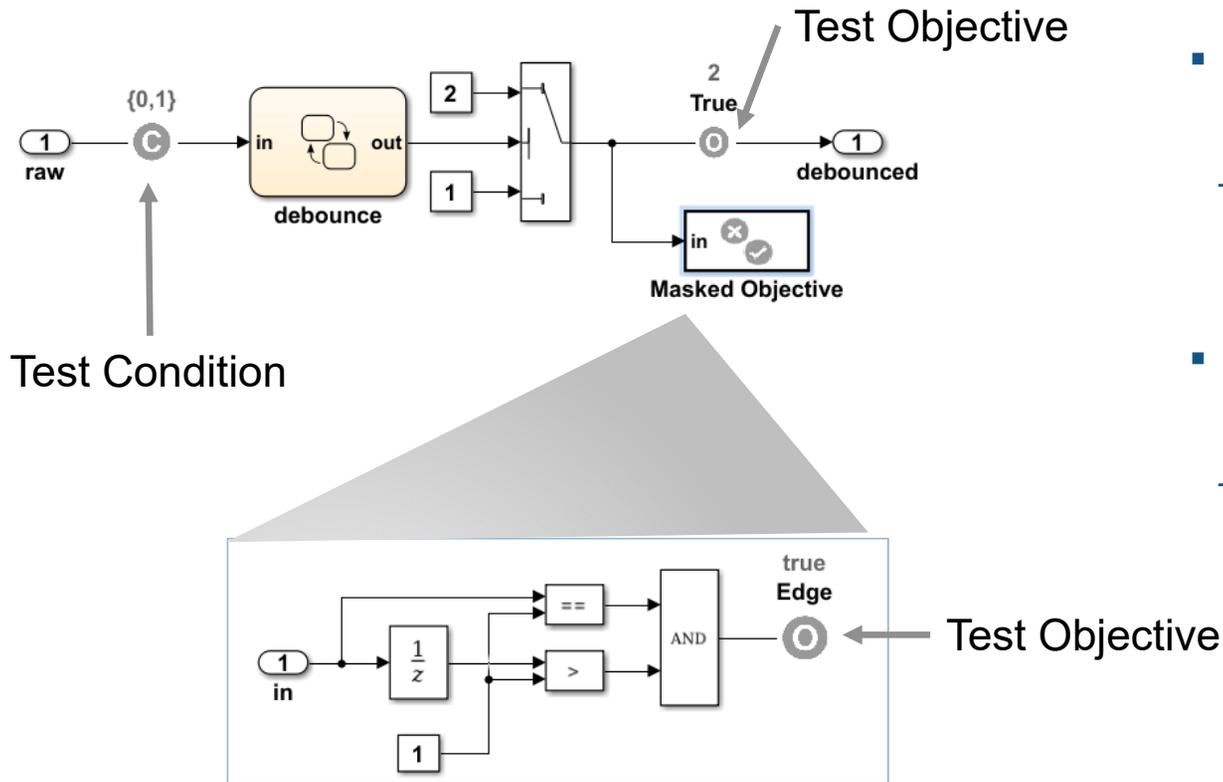
decision outcomes:	True	False
	Out	Out

Summary

Coverage Reports

Model Hierarchy/Complexity	Test 1	Decision	Condition	MCDC	Execution	Relational Boundary	Saturation on integer overflow
1. sldemo_fuellys	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%

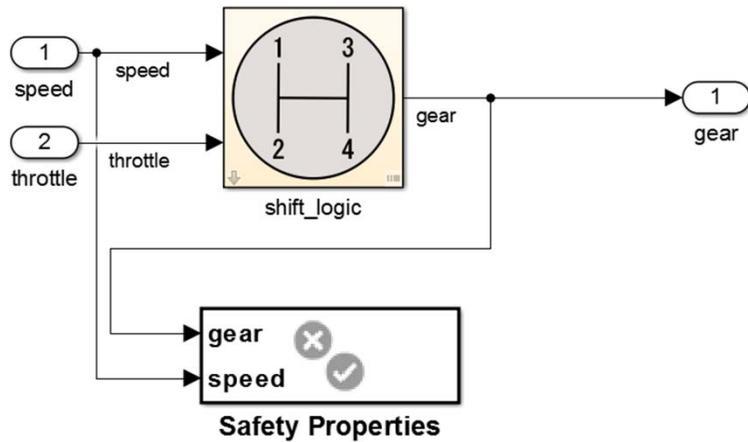
Test Case Generation for Functional Testing



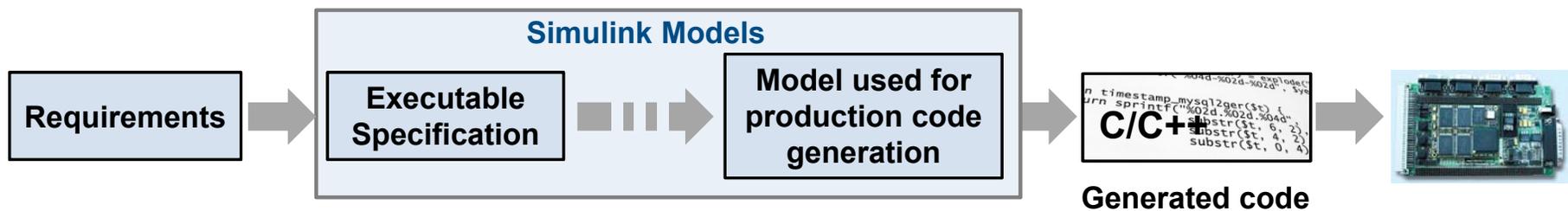
- Specify functional test objectives
 - Define custom objectives that signals must satisfy in test cases

- Specify functional test conditions
 - Define constraints on signal values to constrain test generator

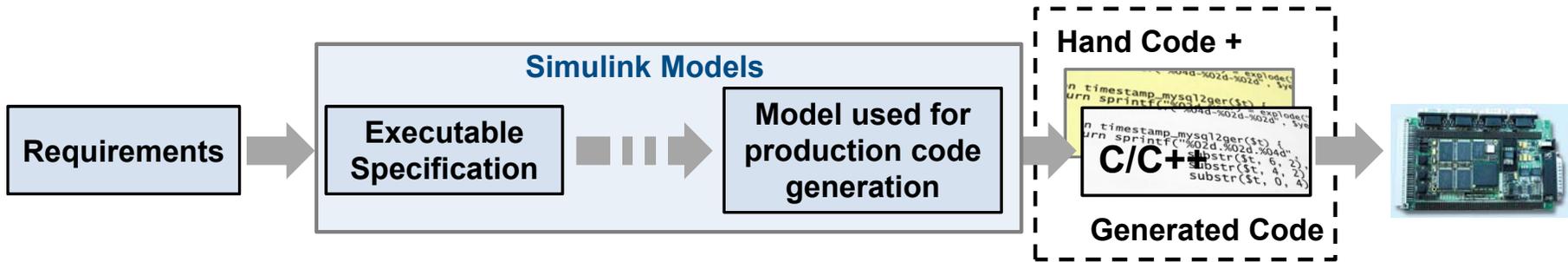
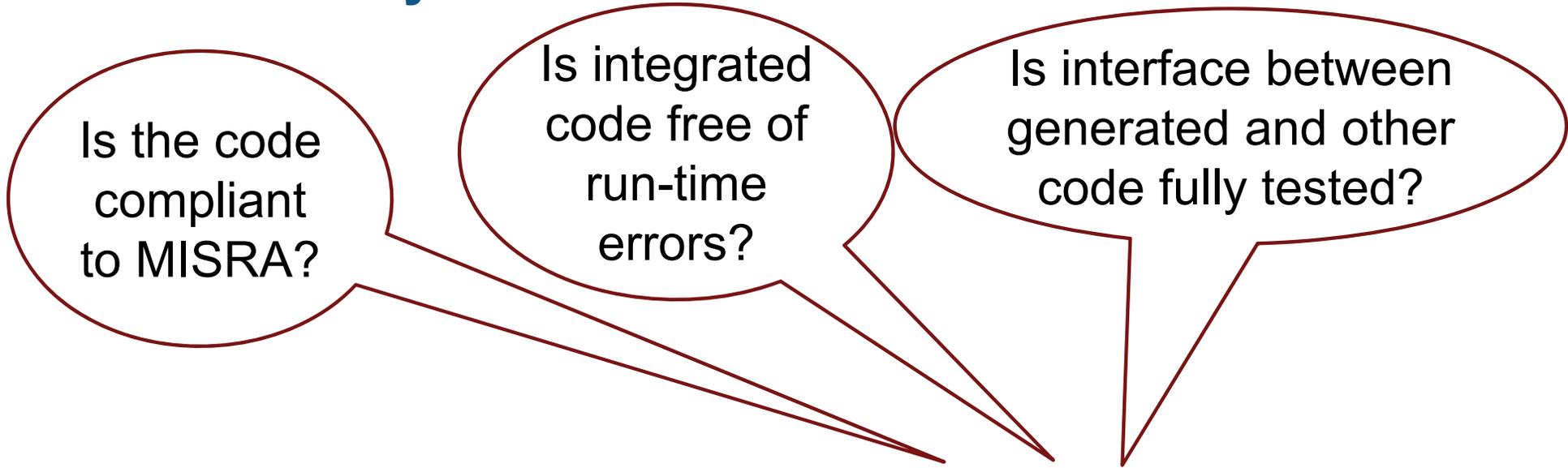
Prove Design Meets Requirements



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



Static Code Analysis

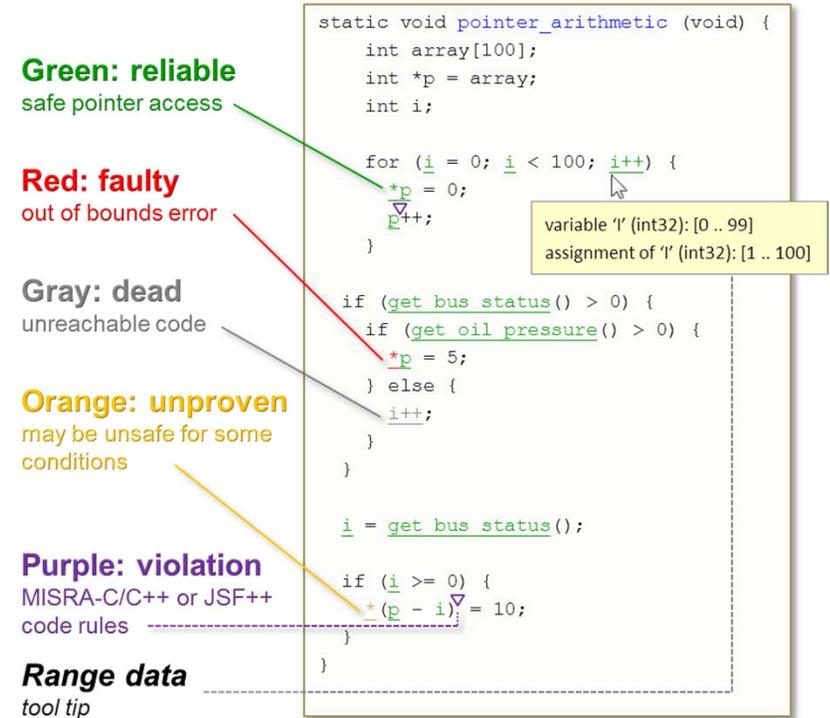


Generated Code is integrated with Hand Code

Static Code Analysis with Polyspace

- Code metrics and standards
 - Comment density, cyclomatic complexity,...
 - MISRA and Cybersecurity standards
 - Support for DO-178, ISO 26262,

- Bug finding and code proving
 - Check data and control flow of software
 - Detect bugs and security vulnerabilities
 - Prove absence of runtime errors



Green: reliable
safe pointer access

Red: faulty
out of bounds error

Gray: dead
unreachable code

Orange: unproven
may be unsafe for some conditions

Purple: violation
MISRA-C/C++ or JSF++ code rules

Range data
tool tip

```

static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
        *p = 0;
        p++;
    }

    if (get_bus_status() > 0) {
        if (get_oil_pressure() > 0) {
            *p = 5;
        } else {
            i++;
        }
    }

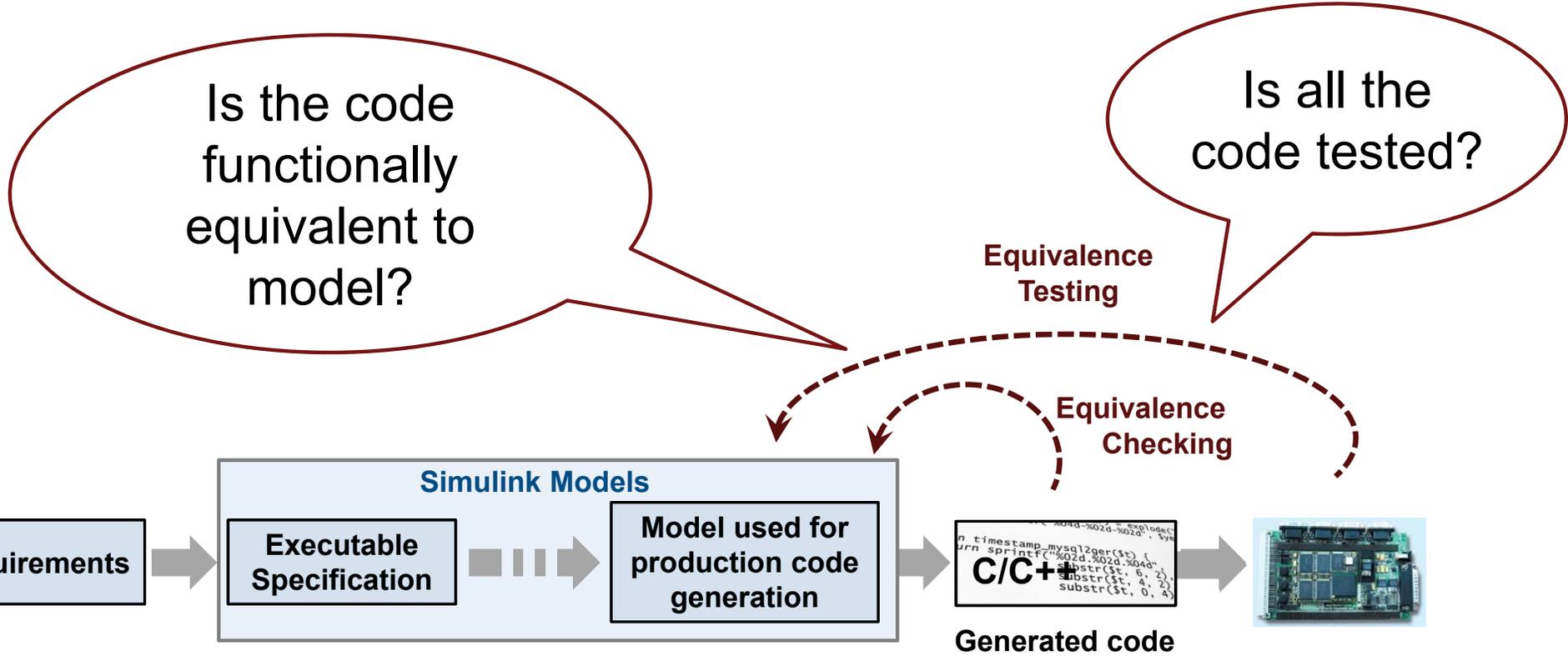
    i = get_bus_status();

    if (i >= 0) {
        *(p - i) = 10;
    }
}
    
```

variable 'i' (int32): [0 .. 99]
assignment of 'i' (int32): [1 .. 100]

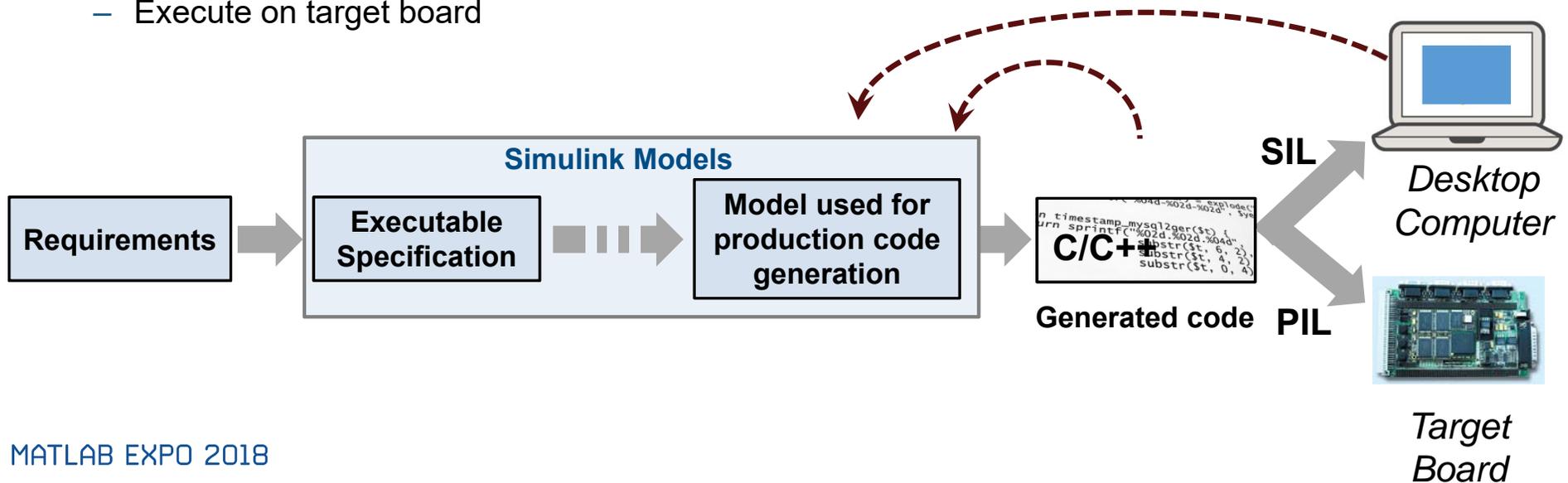
Results from Polyspace Code Prover

Equivalence Testing



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-use tests developed for model to test code
- Check for equivalent outputs model to code
- Collect code coverage, compare to model coverage



Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification tools
- 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.

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

MATLAB EXPO 2018

[Link to user story](#)



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

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



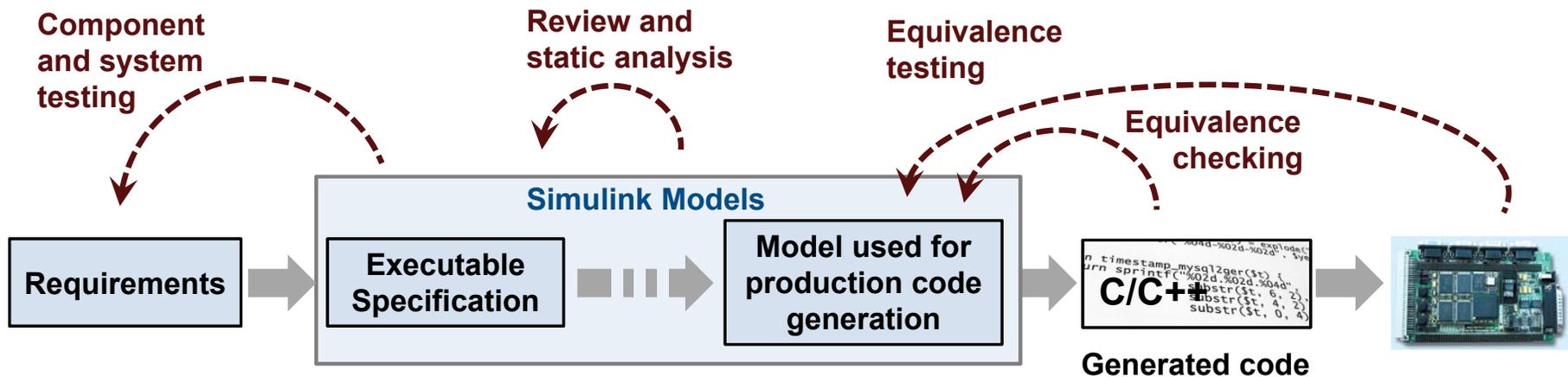
Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks

Verification time cut by up to 50 percent

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

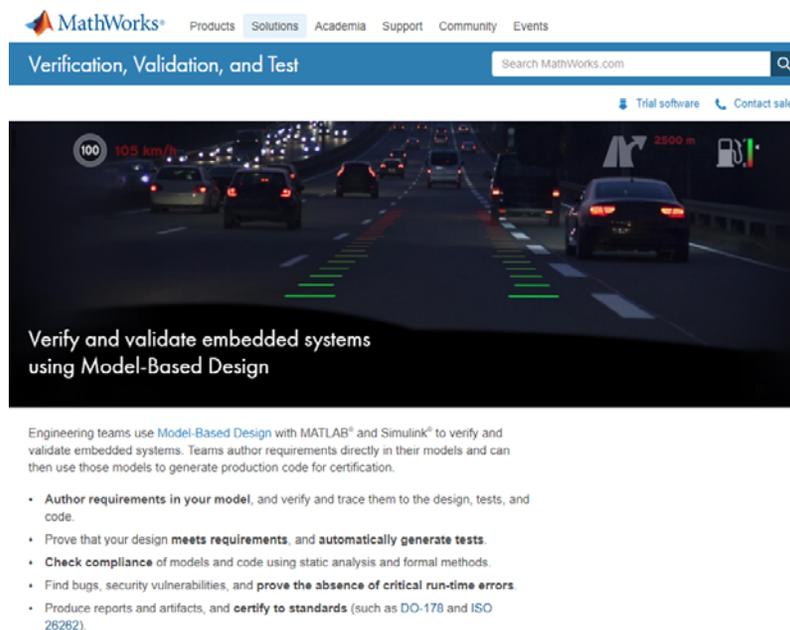
Summary

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



Learn More

Visit MathWorks Verification, Validation and Test Solution Page:
mathworks.com/solutions/verification-validation.html



MathWorks® Products Solutions Academia Support Community Events

Verification, Validation, and Test Search MathWorks.com

Try software Contact sales

100 105 km/h 2500 m

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).

MATLAB EXPO 2018

Thank You!