

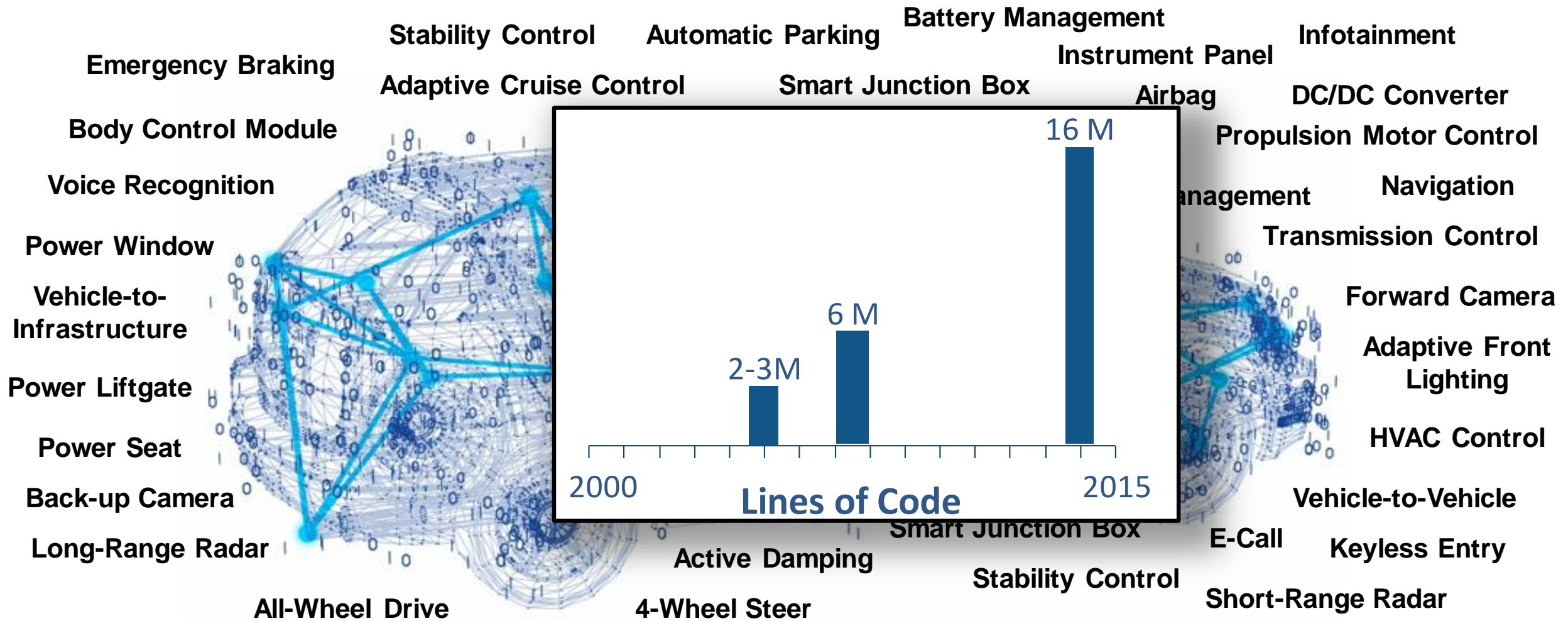
# MATLAB EXPO 2018

Automatización de Métodos y Procesos  
para Mejorar la Calidad del Diseño

Luis López



# 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

# 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

## System Requirements

maximum machine velocity, left track  
 maximum machine acceleration, left track  
 maximum machine jolt, left track  
 motor speed for 50% rise time, left track  
 10% rise time, left track  
 motor speed for 55% rise time, left track  
 15% rise time, left track  
 maximum machine velocity, right track  
 maximum machine acceleration, right track  
 maximum machine jolt, right track  
 motor speed for 50% rise time, right track

## Verified & Validated System



High Level Design

Detailed Design

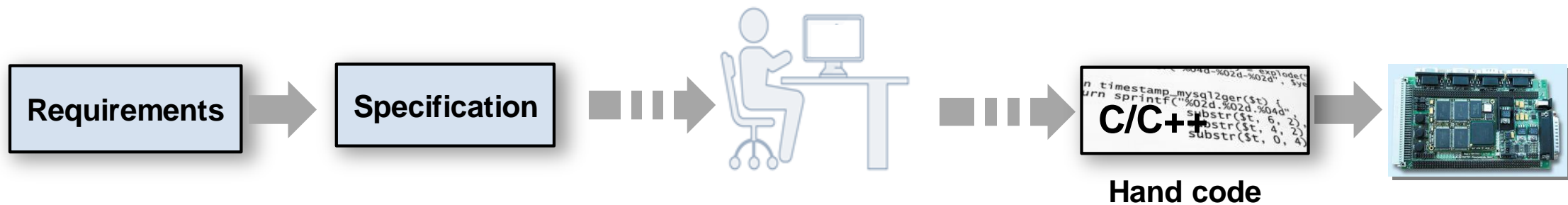
Coding

Unit Testing

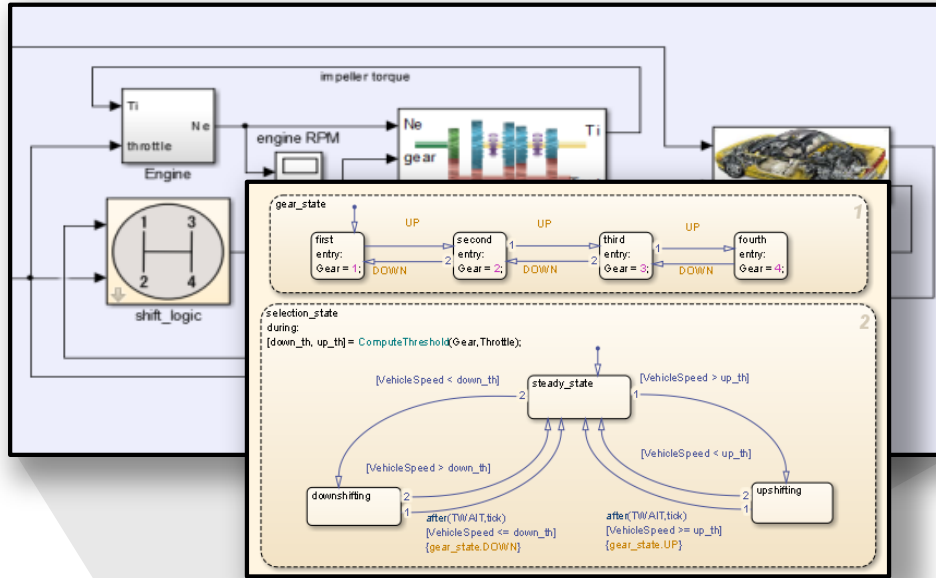
Integration Testing

*“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*

# Challenge with Traditional Development Process



# Simulink Models for Specification



Requirements

Executable Specification



```

n timestamp_mysql2ger($t) {
  urn sprintf("%02d.%02d.%04d",
    substr($t, 6, 2),
    substr($t, 4, 2),
    substr($t, 0, 4)
  )
}

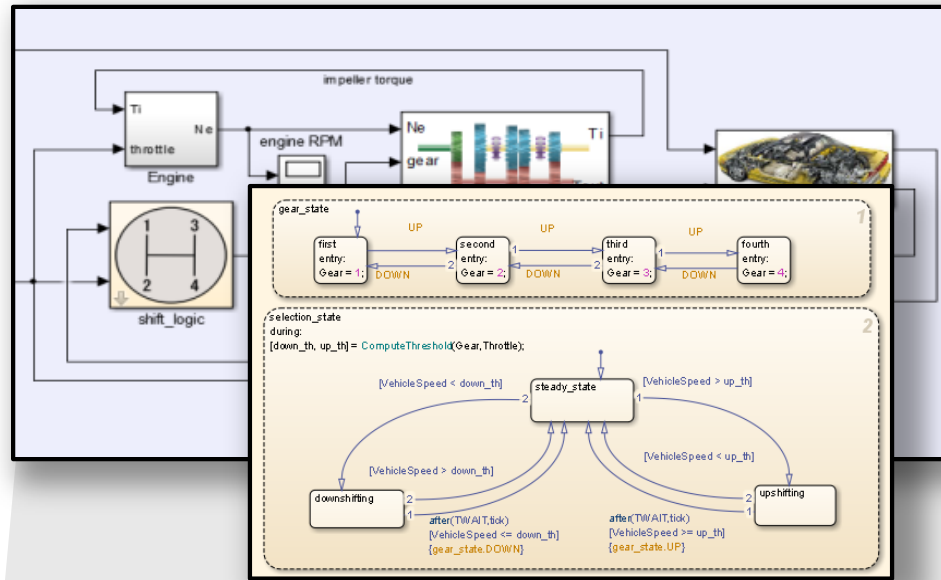
```

C/C++

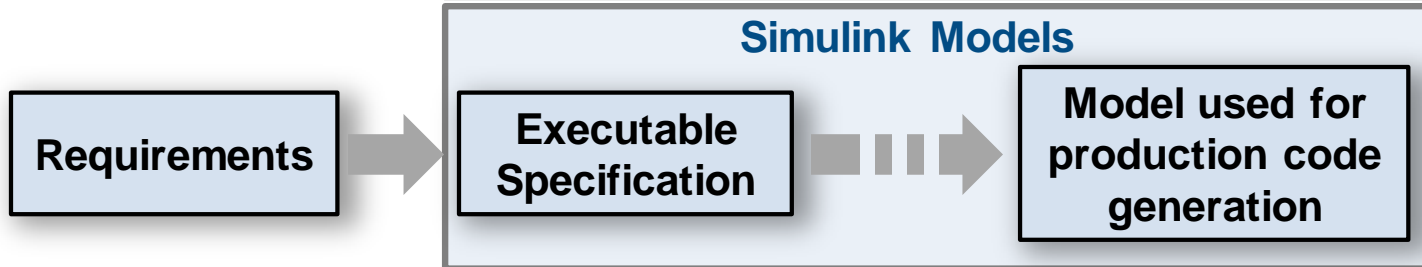
Hand code



# Complete Model Based Design



**Code Generation**



```

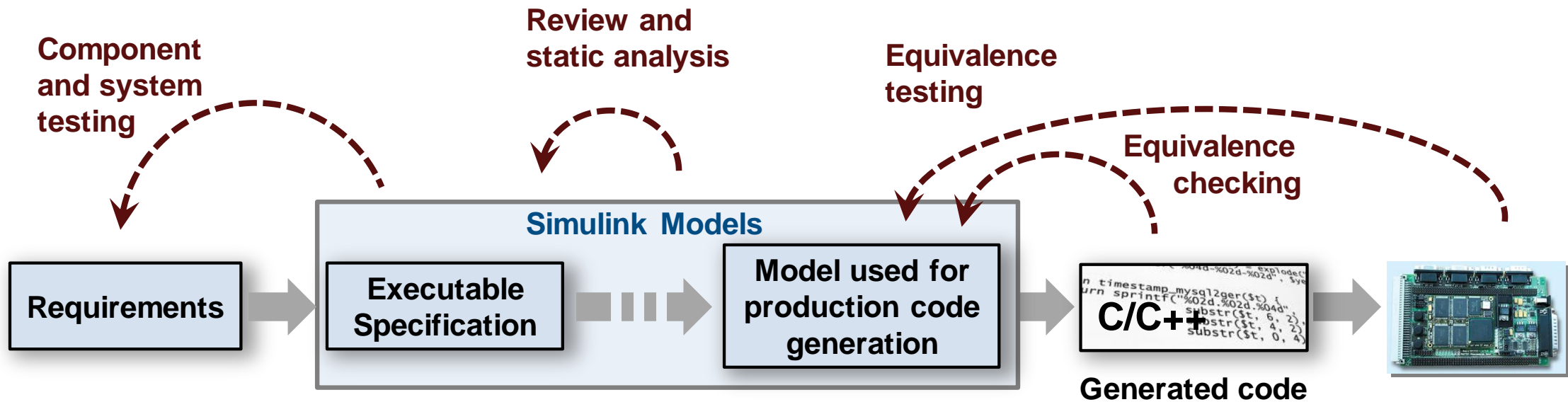
n timestamp,mysql2ger($t) {
  urn sprintf("%02d.%02d.%04d",
    substr($t, 6, 2),
    substr($t, 4, 2),
    substr($t, 0, 4)
  )
}

```

**Generated code**



# Model Based Design Verification Workflow



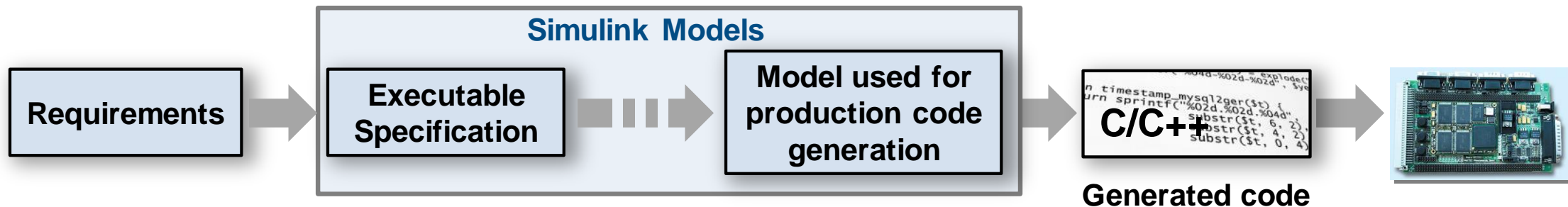


# Challenges with Requirements

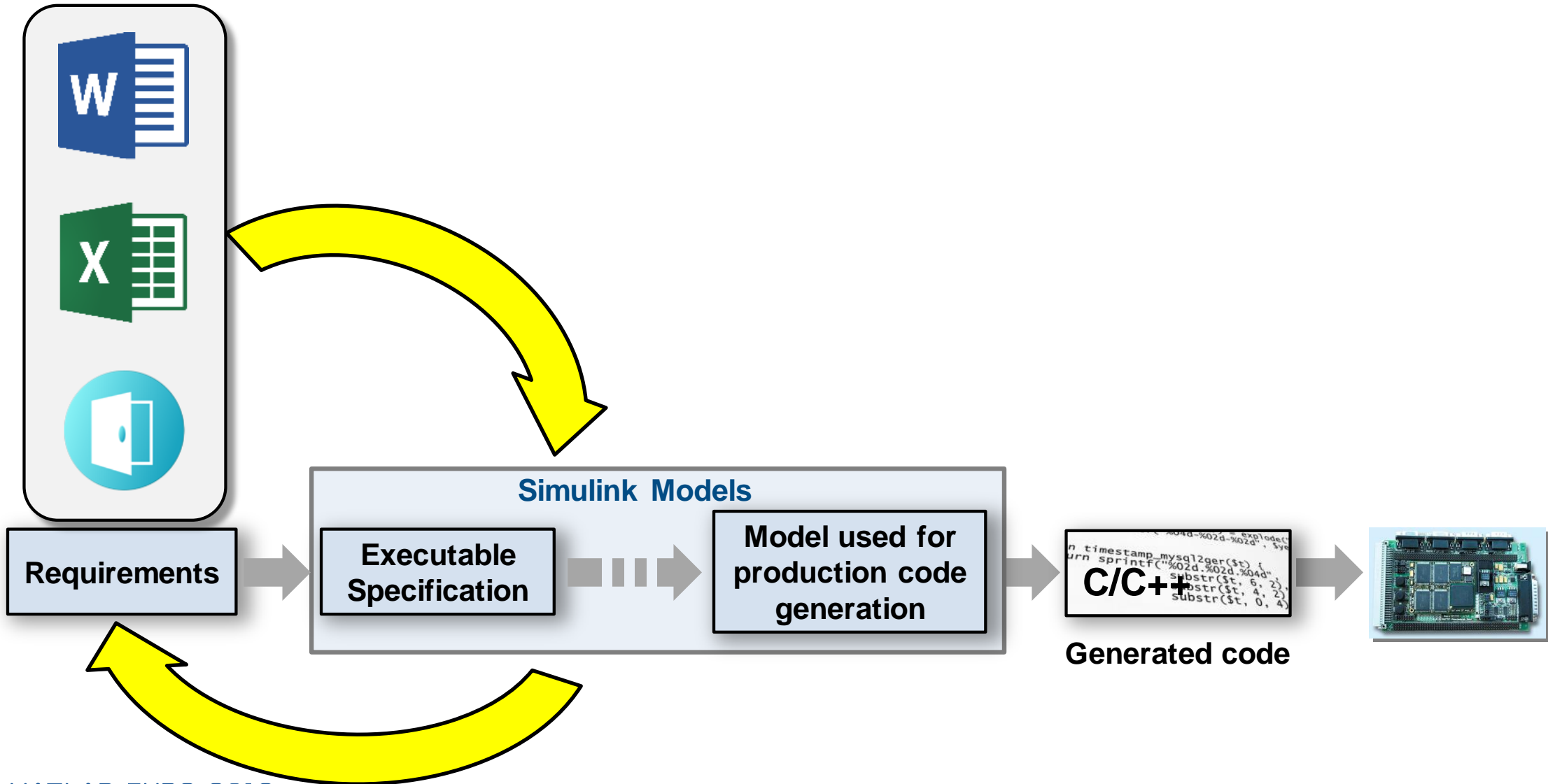
Where are requirements implemented?

Is design and requirements consistent?

How are they tested?



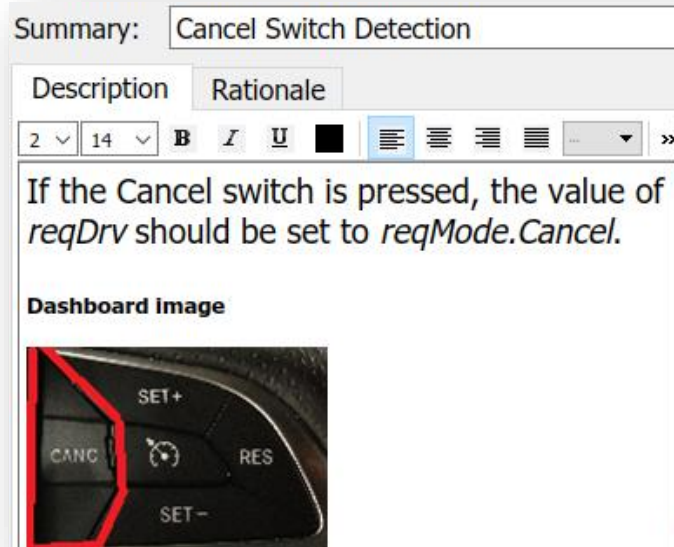
# Gap Between Requirements and Design



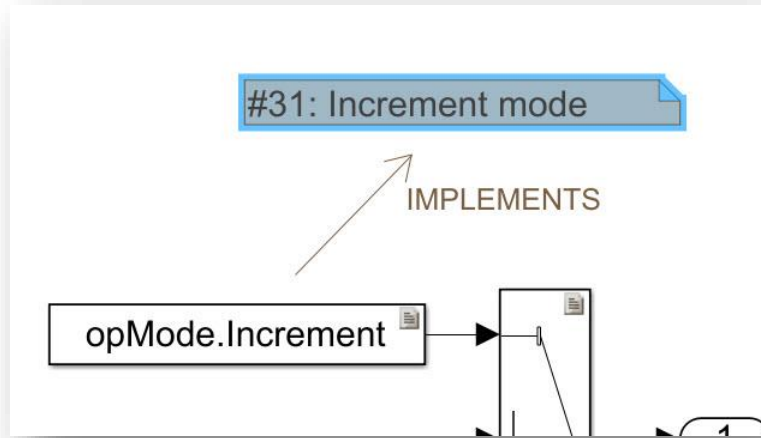
# Simulink Requirements

R2017b

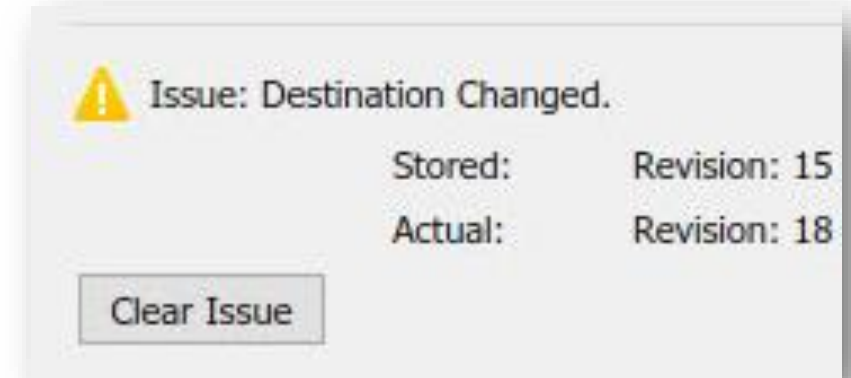
## Author



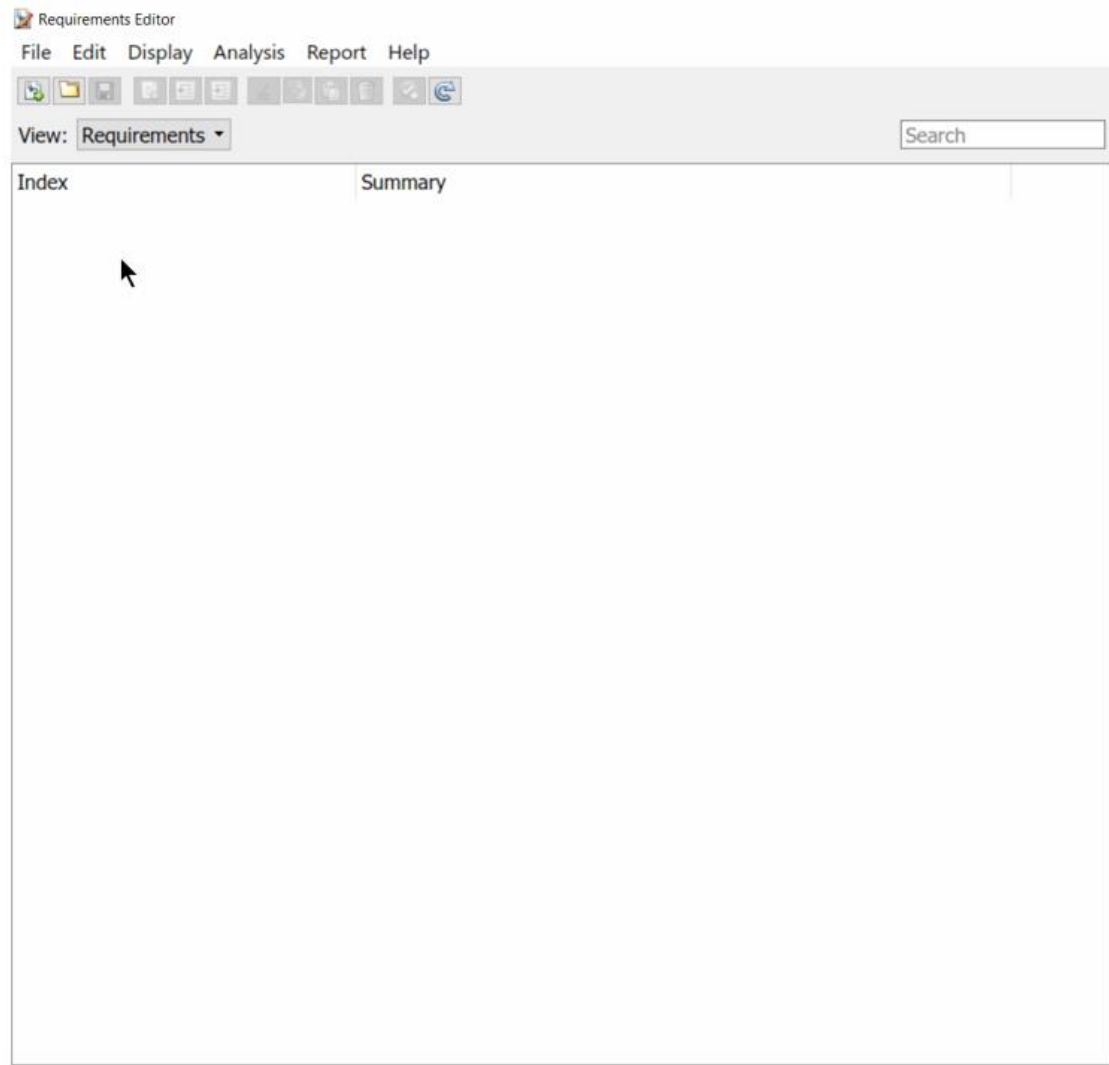
## Track



## Manage





# Requirements Editor



The screenshot shows the Requirements Editor window with the following components:

- Title bar: Requirements Editor
- Menu bar: File Edit Display Analysis Report Help
- Toolbar: Contains icons for file operations and editing.
- View: Requirements (dropdown menu)
- Search: Search input field
- Index: A large empty pane with a mouse cursor.
- Summary: A smaller pane to the right of the Index.

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**.

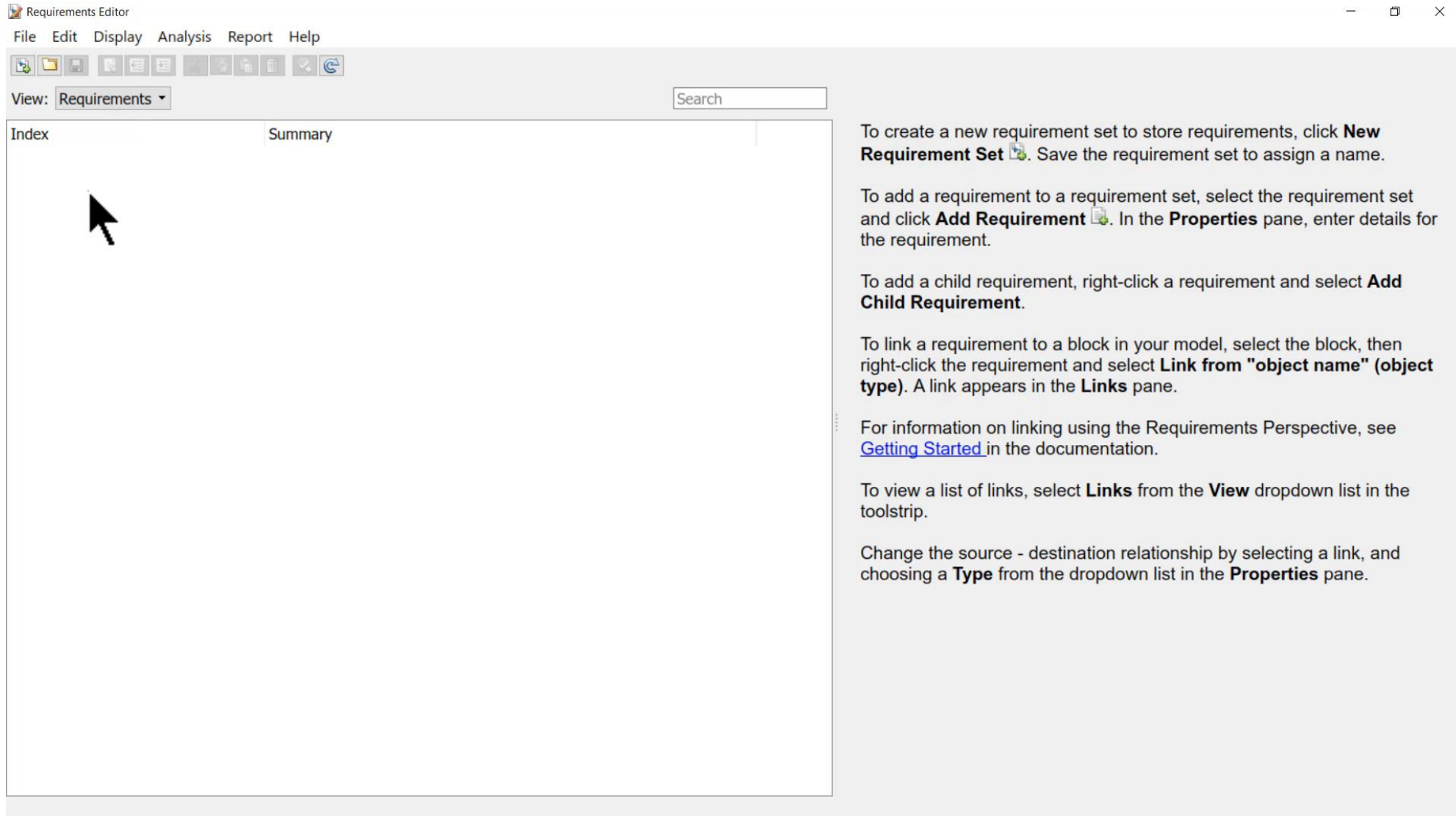
To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

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.

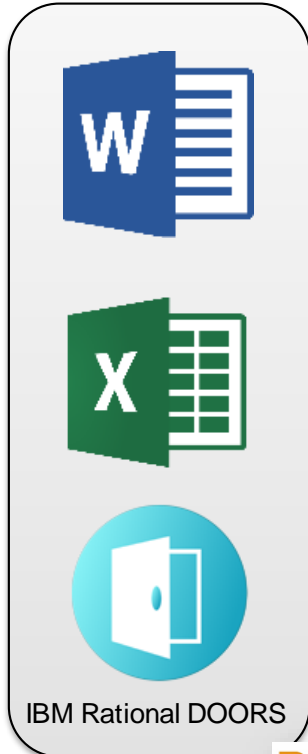
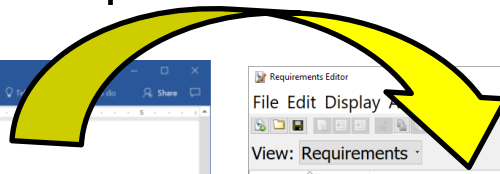
For information on linking using the Requirements Perspective, see [Getting Started](#) in the documentation.

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



R2018a

Microsoft Word

crs\_req.docx - Word

File Home Insert Draw Design Layout References Mailings Review View Developer

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 ¶

Simulink Requirements Editor

Requirements Editor

File Edit Display Help

View: Requirements Search

Index	ID	Summary
crs_req	crs_req	References to crs_req.docx
1	1 Overview	Overview This document describes a r
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 Th
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 i
1.3.2	3.2 Disabling cruise control	Disabling cruise control Cruise control
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 cont
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 Accelerate
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 c
1.3.12	3.12 Cruise Control SET Indi...	Cruise Control SET Indicator Light Cru
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 condi...

Description Rationale

### 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

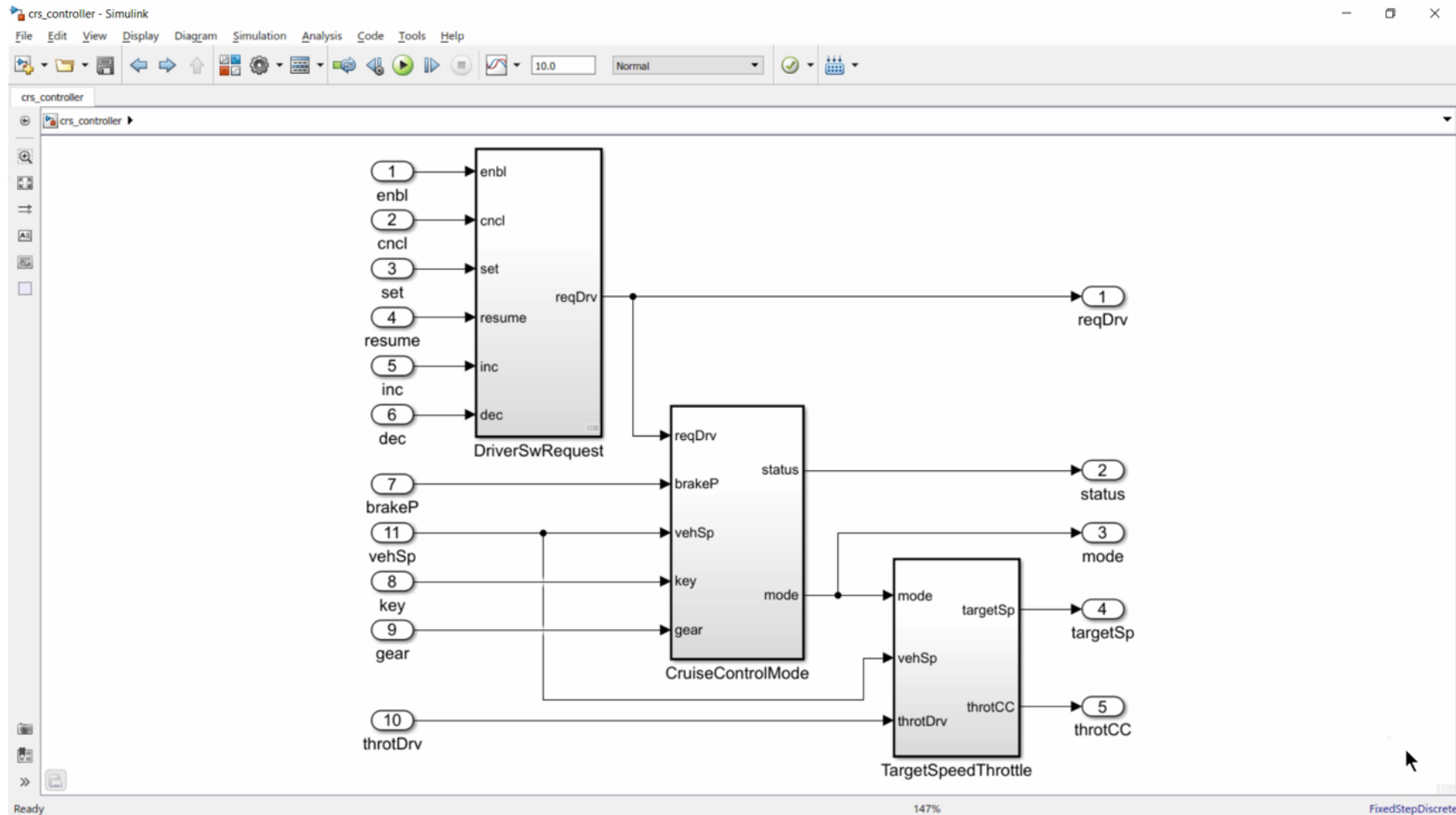
Keywords:

Revision information: Show in document

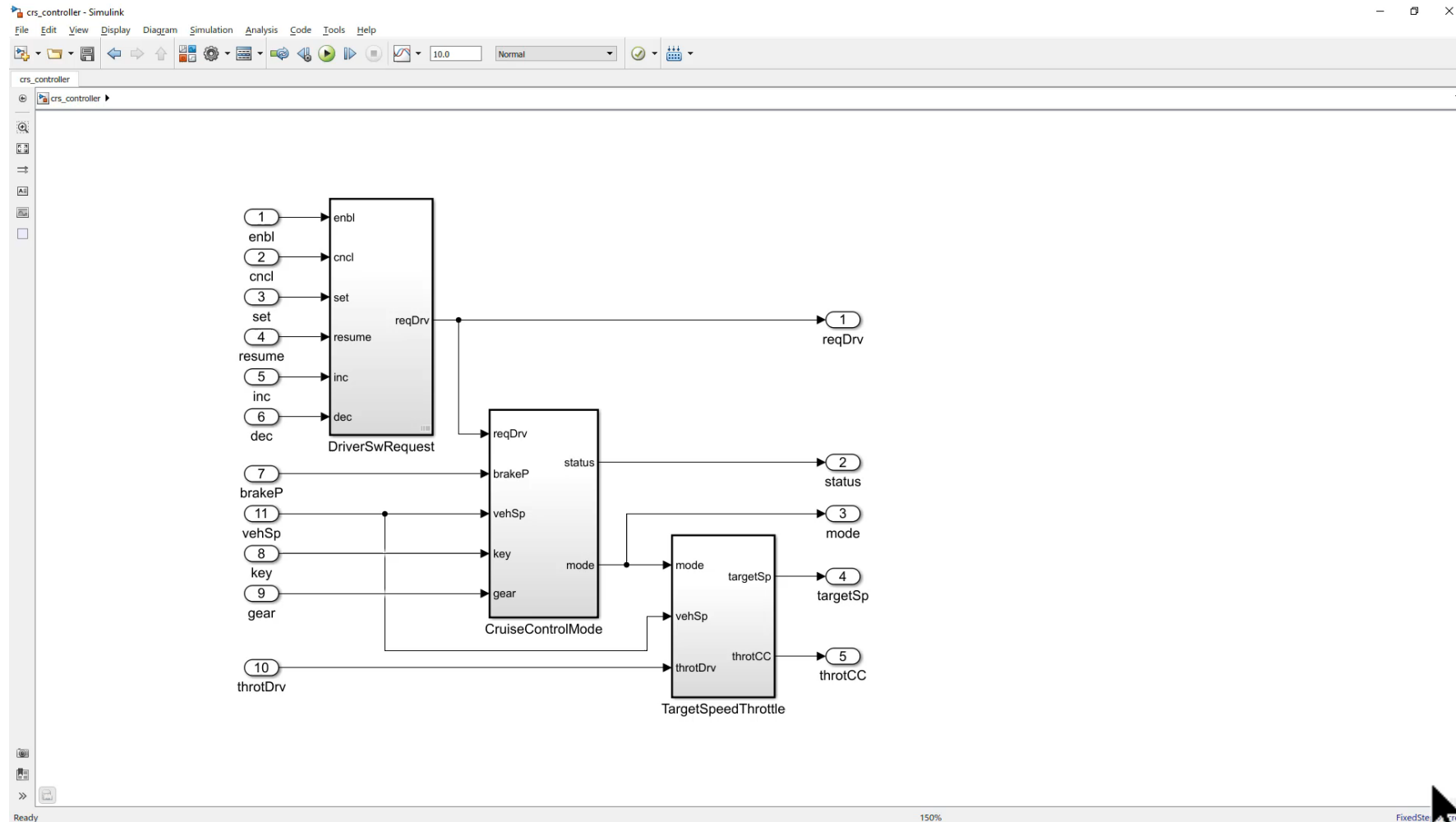
Links

Show in document

# Requirements Perspective

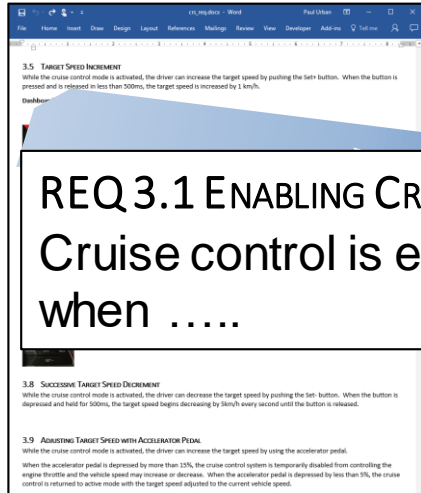


# Requirements Perspective



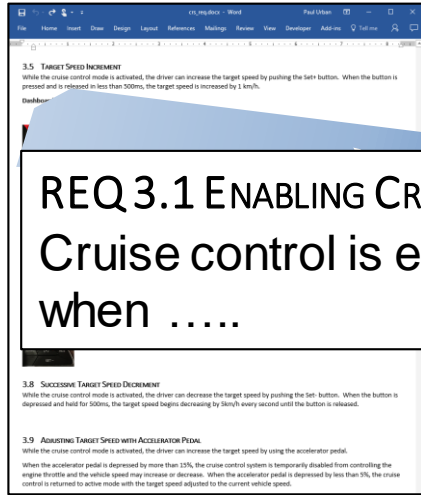


# Link Requirements, Designs and Tests



REQ 3.1 ENABLING CRUISE CONTROL  
Cruise control is enabled  
when .....

# Link Requirements, Designs and Tests



Derives

**REQ 3.1 ENABLING CRUISE CONTROL**  
Cruise control is enabled when .....

**ENABLE SWITCH DETECTION**  
If the Enable switch is pressed .....

# Link Requirements, Designs and Tests

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 5 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.

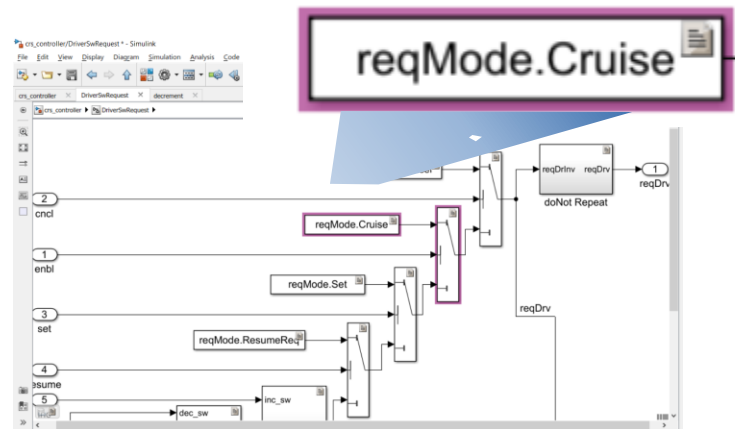
3.9 ADJUSTING TARGET SPEED WITH ACCELERATOR PEDAL  
While the cruise control mode is activated, the driver can increase the target speed by using the accelerator pedal. When the accelerator pedal is depressed by more than 15%, the cruise control system is temporarily disabled from controlling the engine 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



# Link Requirements, Designs and Tests

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 1km/h every second until the button is released.

3.9 ADJUSTING TARGET SPEED WITH 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 .....

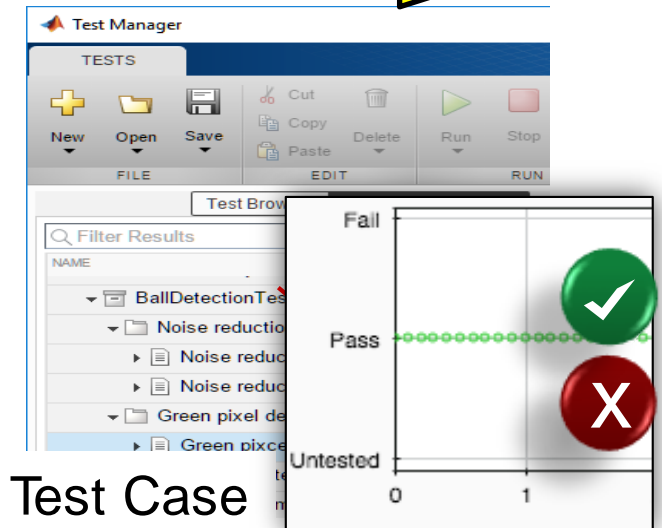
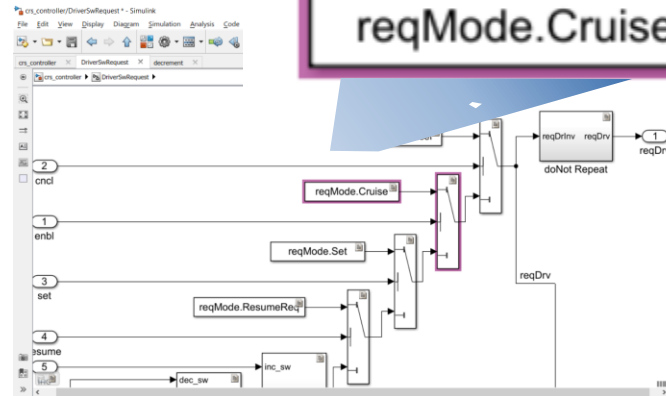
Derives

**ENABLE SWITCH DETECTION**  
If the Enable switch is pressed .....

Implemented  
By

Verified  
By

**reqMode.Cruise**



# Track Implementation and Verification

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

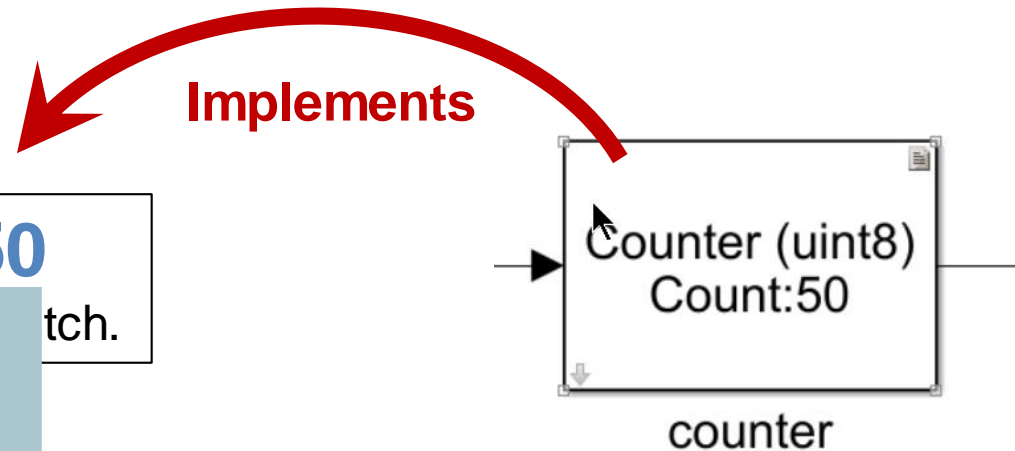
# Respond to Change

## 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

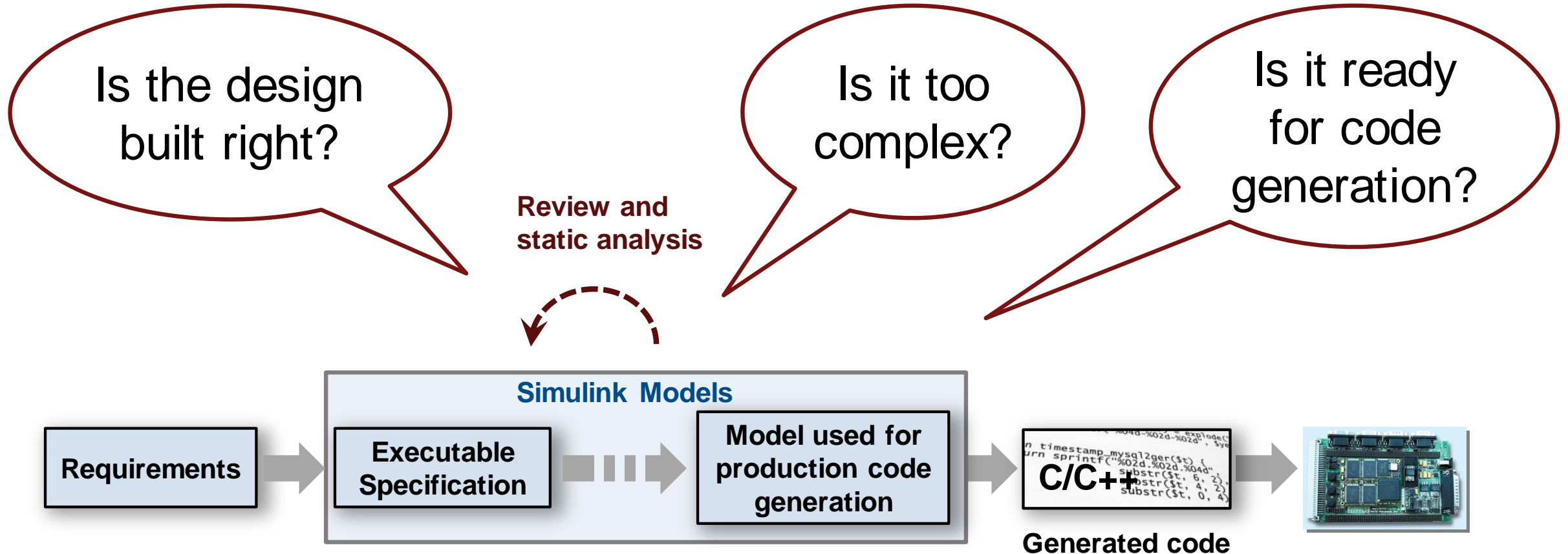
If the switch is pressed and the counter reaches **75** then it shall be recognized as a long press of the switch.



← **Implemented by:**  
counter

 **Issue: Destination Changed.**

# Verify Design to Guidelines and Standards



# Automate verification with static analysis

**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.

Run This Check

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

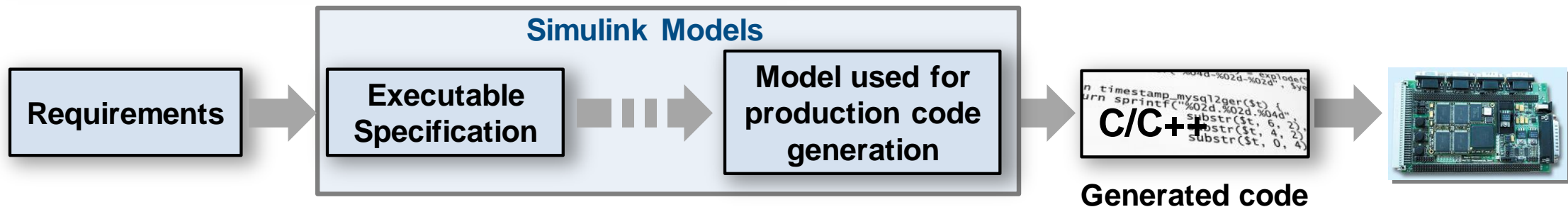
**Warning**  
The following blocks are not supported or not recommended for C/C++ production code deployment:

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

Check for:

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





# 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/p0 = 0.589 bar	Integrator	Yes <sup>1,2</sup>	No
sldemo_fuelsys/Throttle Command	Repeating table	Yes <sup>3</sup>	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

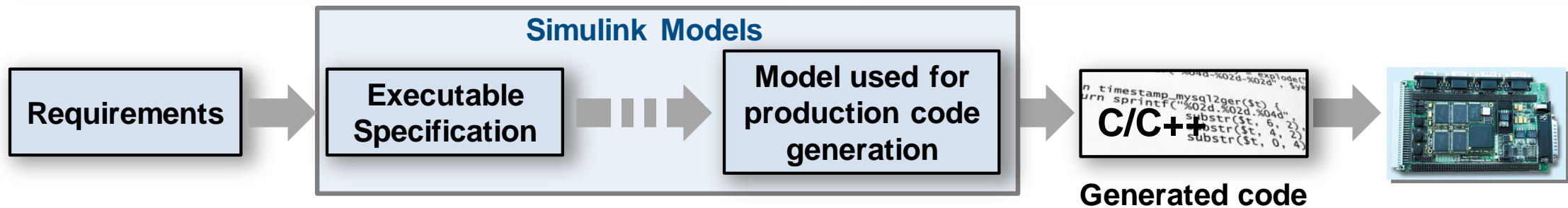
**By Task**

- 1 Code Generation Efficiency: 3 Pass, 0 Fail, 3 Warning

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

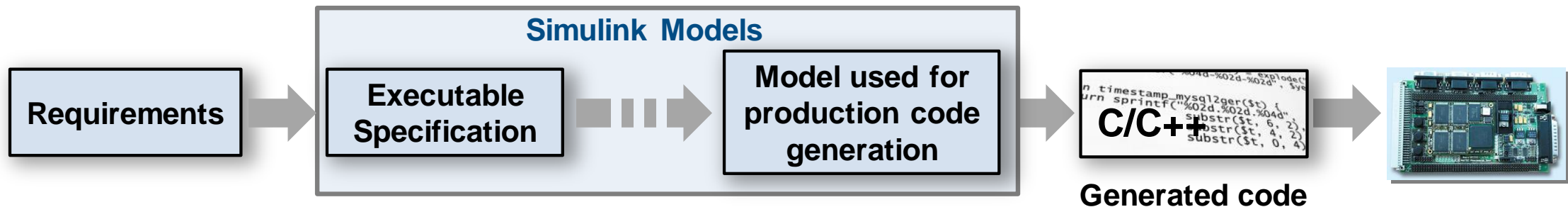
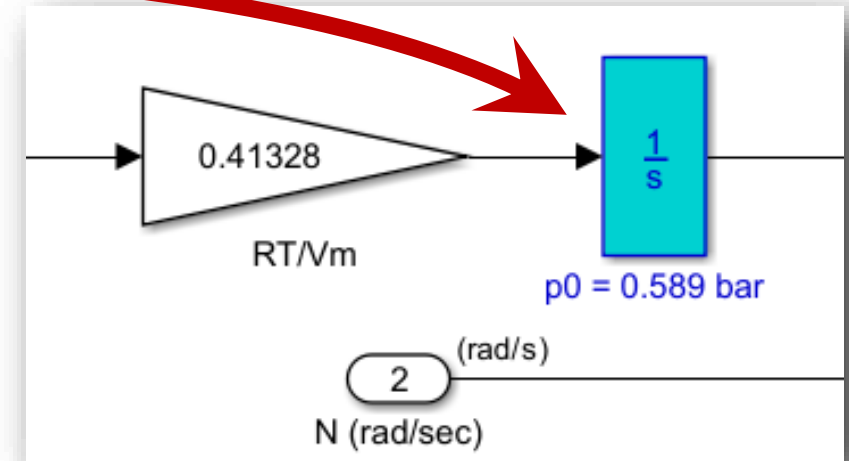
**Warning**

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



# Navigate to Problematic Blocks

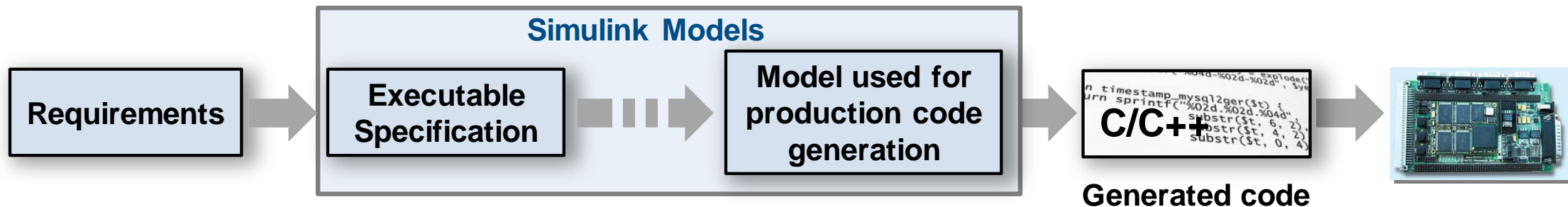
Block	Block Type	Code generation support	Recommendation for C/C++ production code deployment
<a href="#">.../Intake Manifold/p0</a> = 0.589 bar	Integrator	Yes <sup>1, 2</sup>	No
<a href="#">sldemo_fuelsys/Throttle Command</a>	Repeating table	Yes <sup>3</sup>	No



# Guidance Provided to Address Issues or Automatically Correct

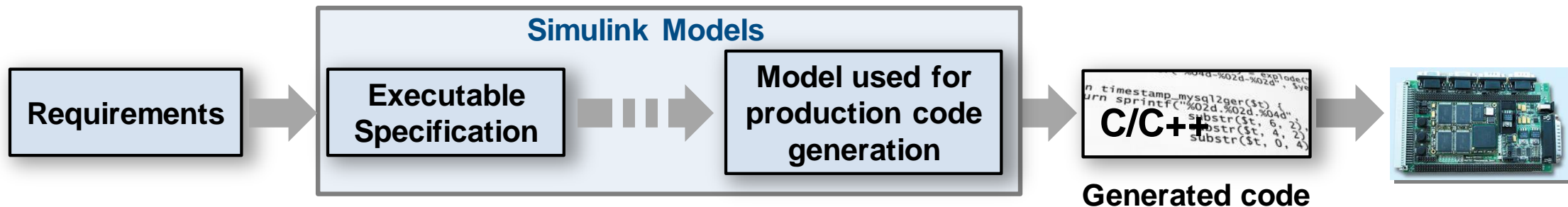
## Recommended Action

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.

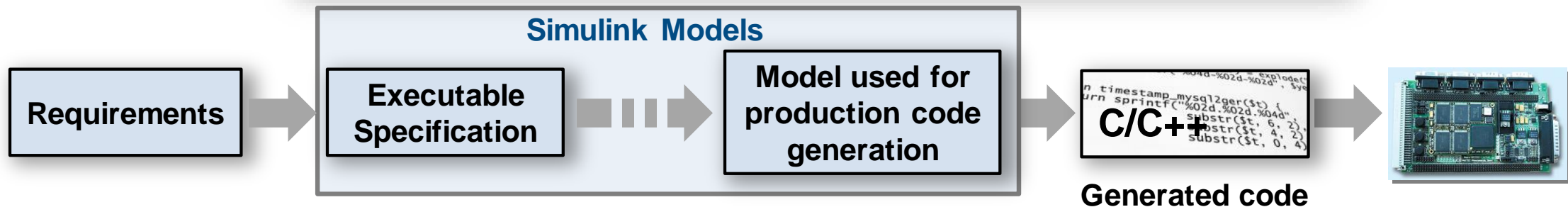


## Built in checks for industry standards and guidelines

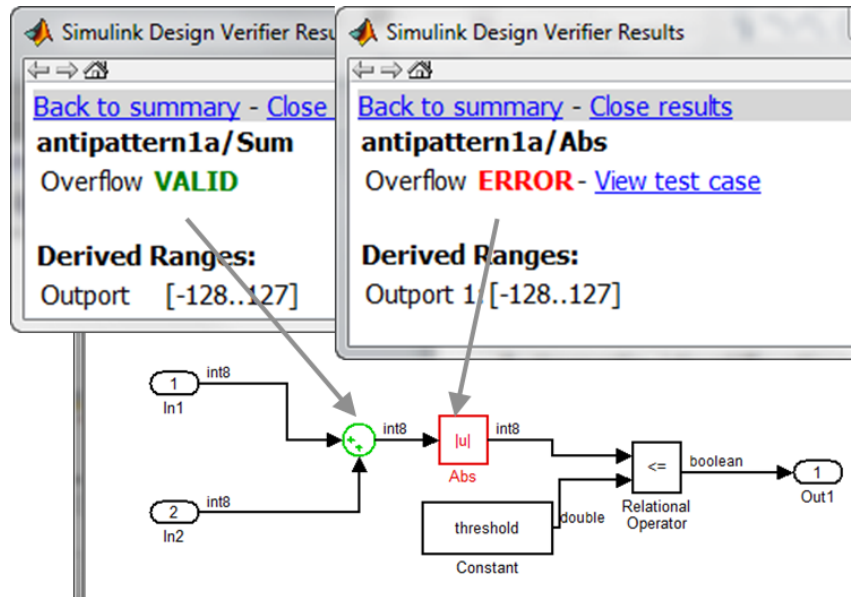
- DO-178/DO-331
- MISRA C:2012
- ISO 26262
- CERT C, CWE, ISO/IEC TS 17961
- IEC 61508
- MAAB (MathWorks Automotive Advisory Board)
- IEC 62304
- JMAAB (Japan MATLAB Automotive Advisory Board)
- EN 50128



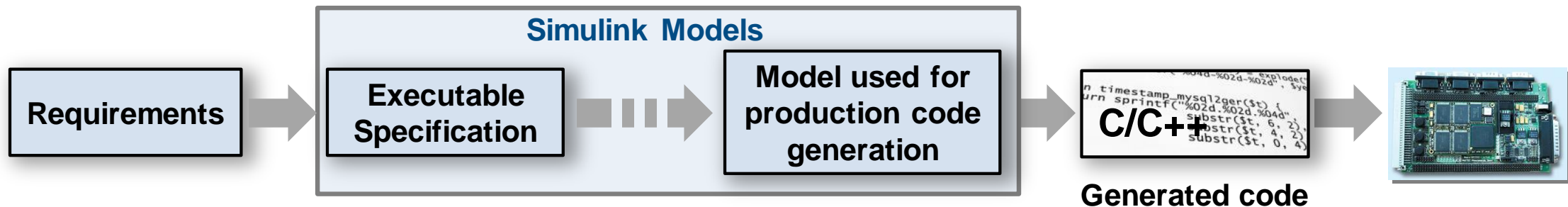
# Configure and customize analysis



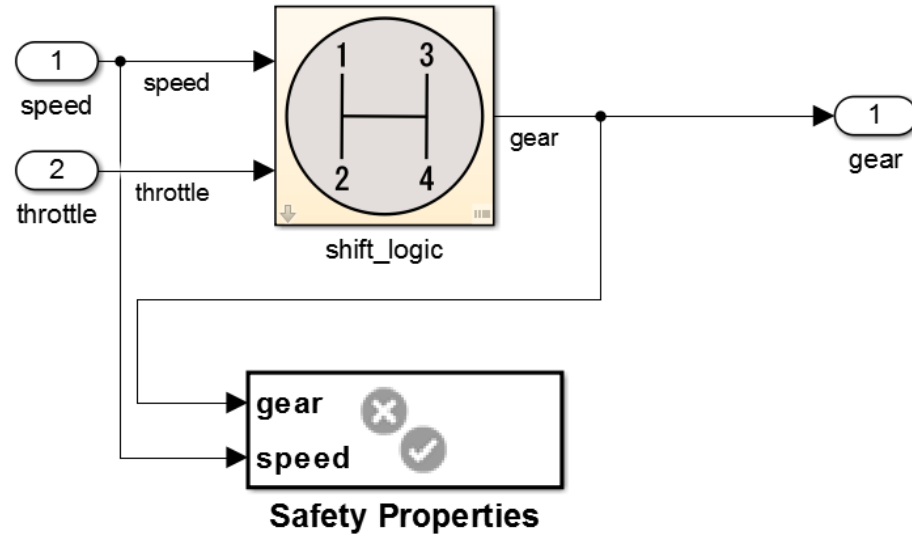
# Detect Design Errors with Formal Methods



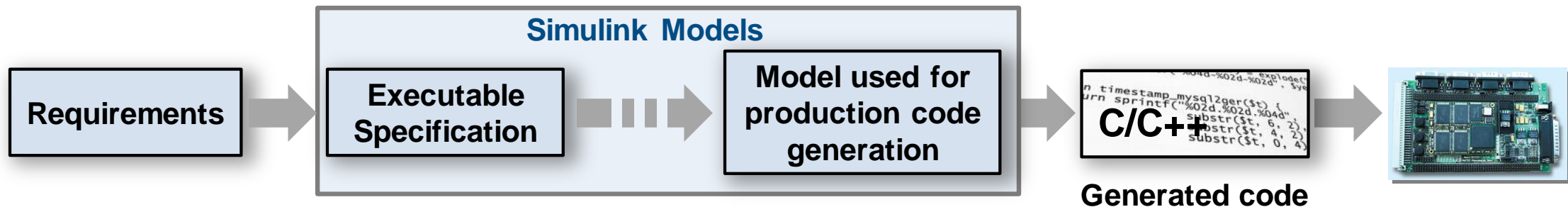
- Find run-time design errors:
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations
  
- Generate counter example to reproduce error



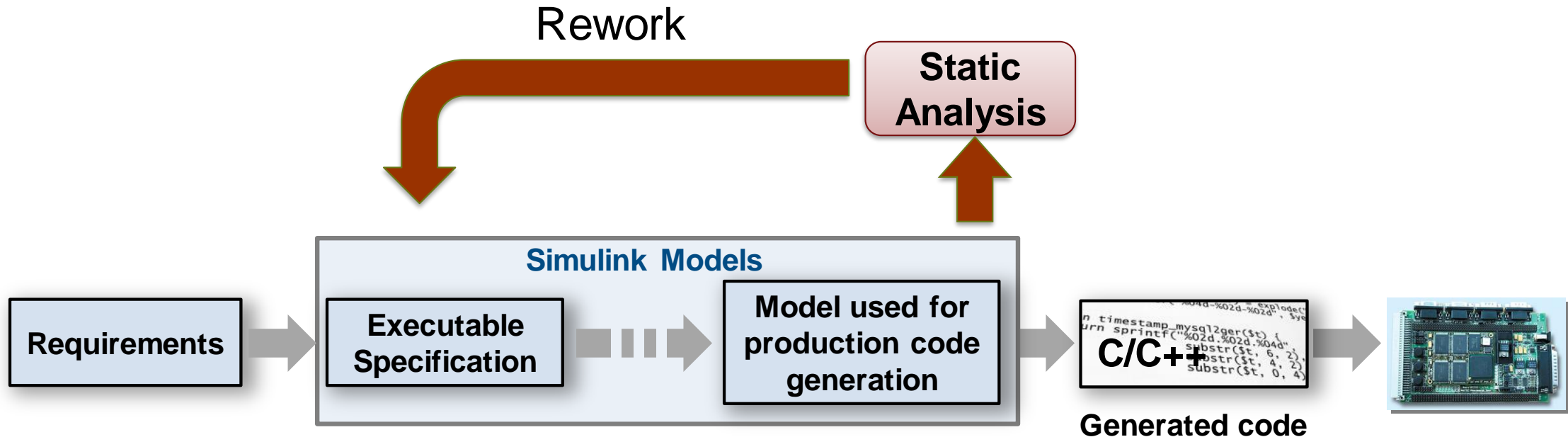
# Prove That Design Meets Requirements



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



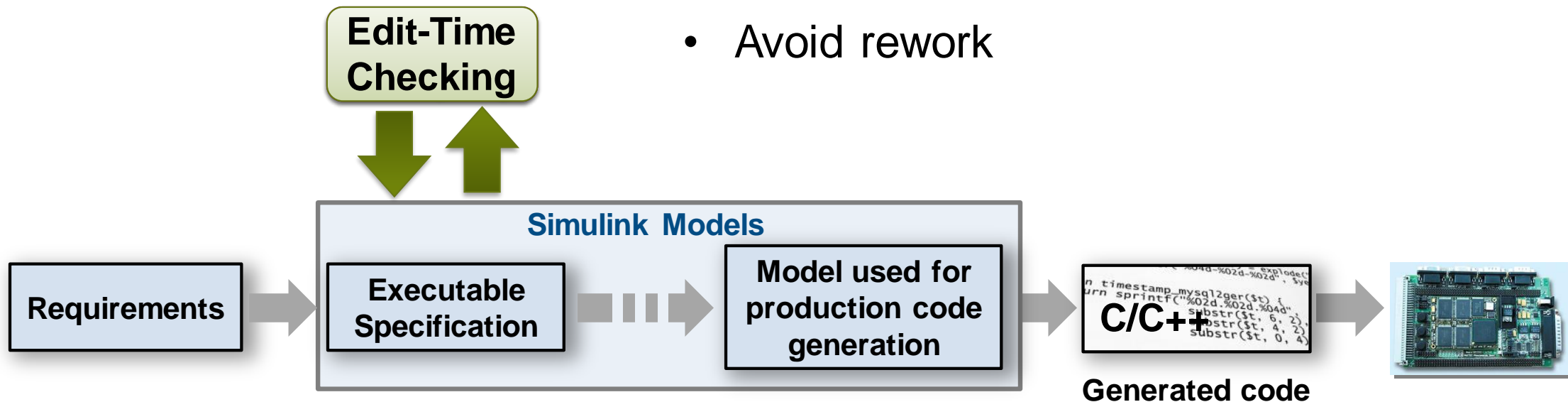
# Checks for standards and guidelines are often performed late



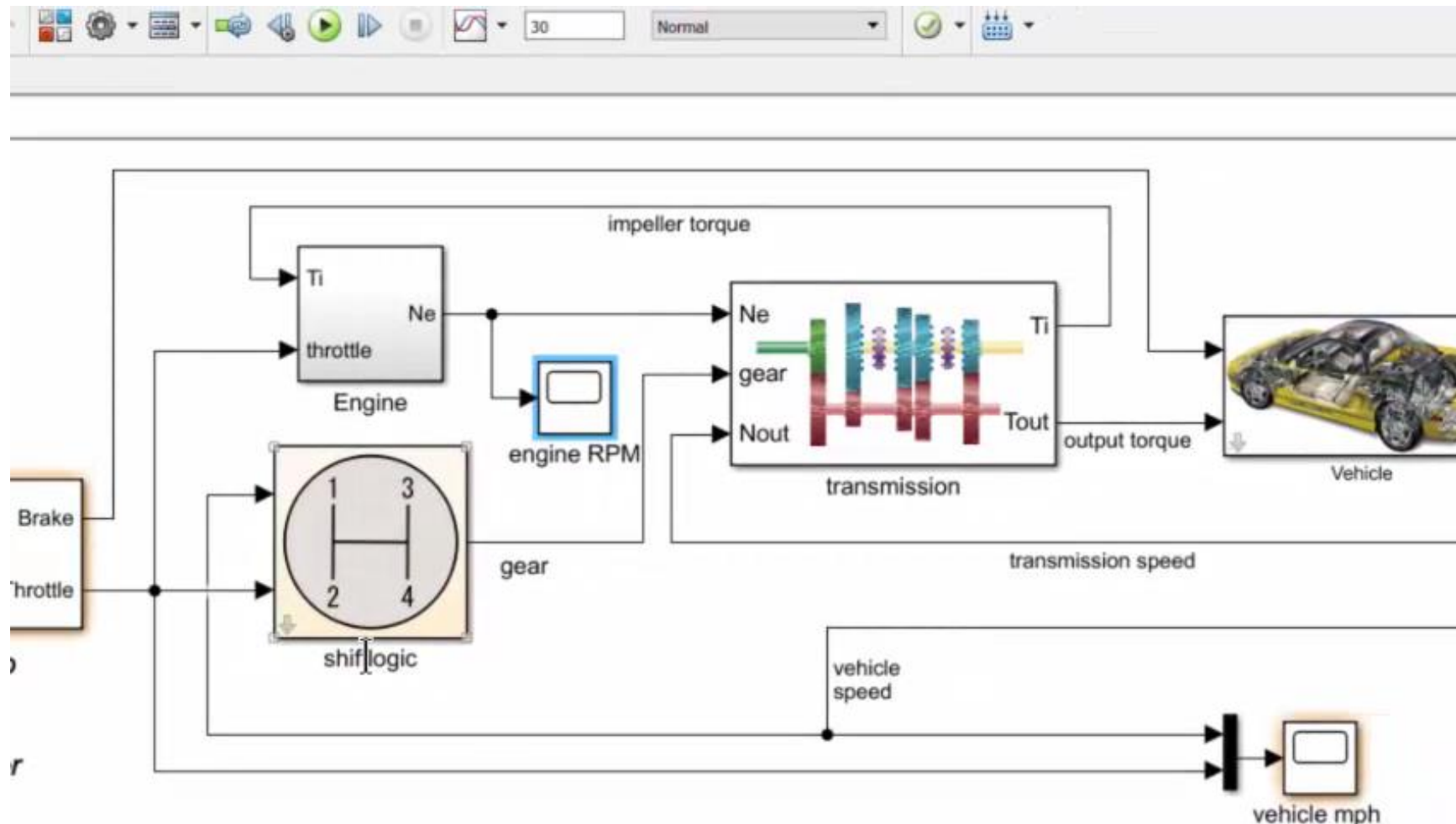


# Shift Verification Earlier With Edit-Time Checking

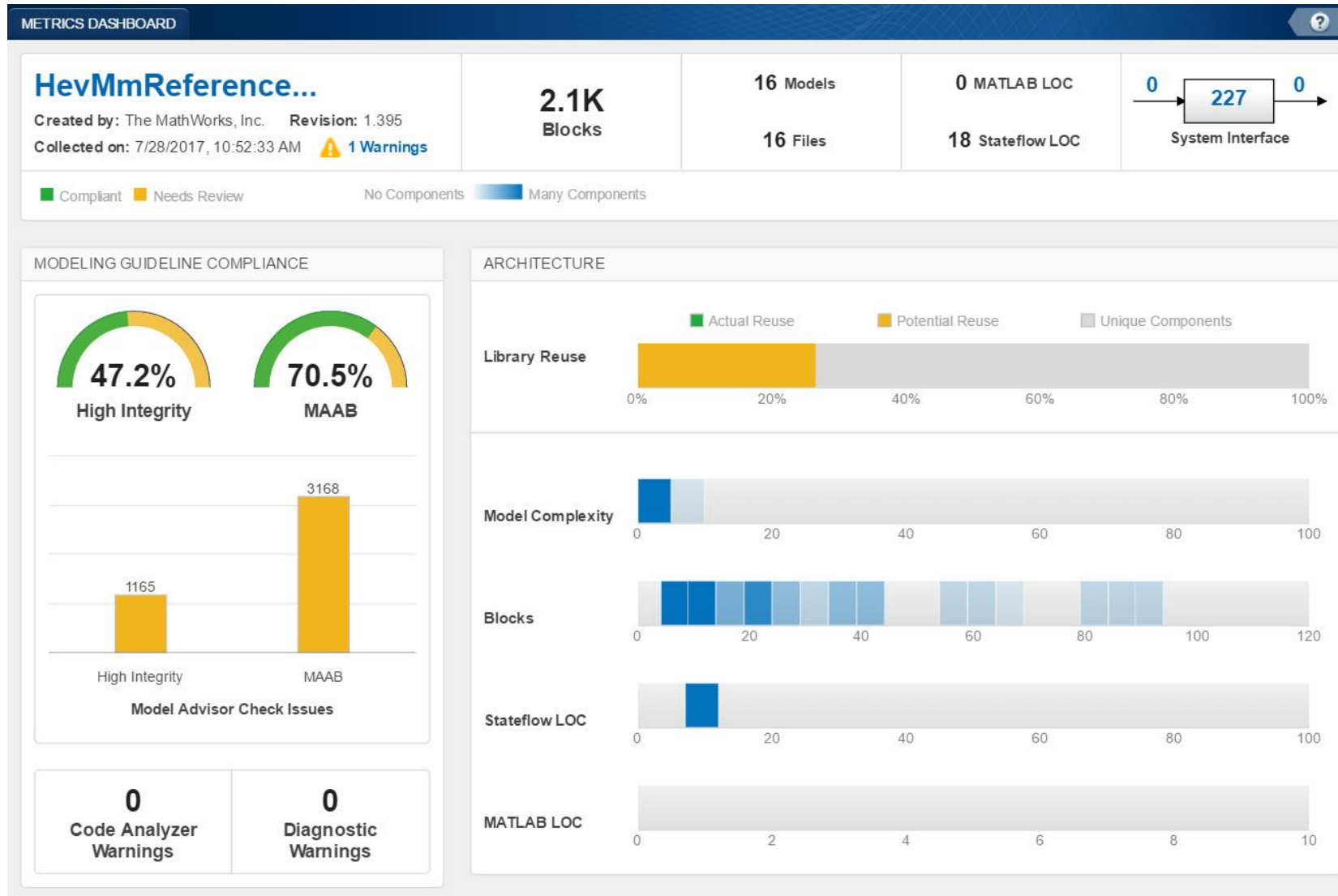
- Highlight violations as you edit
- Fix issues earlier
- Avoid rework



# Find Compliance Issues as you Edit with Edit-Time Checking

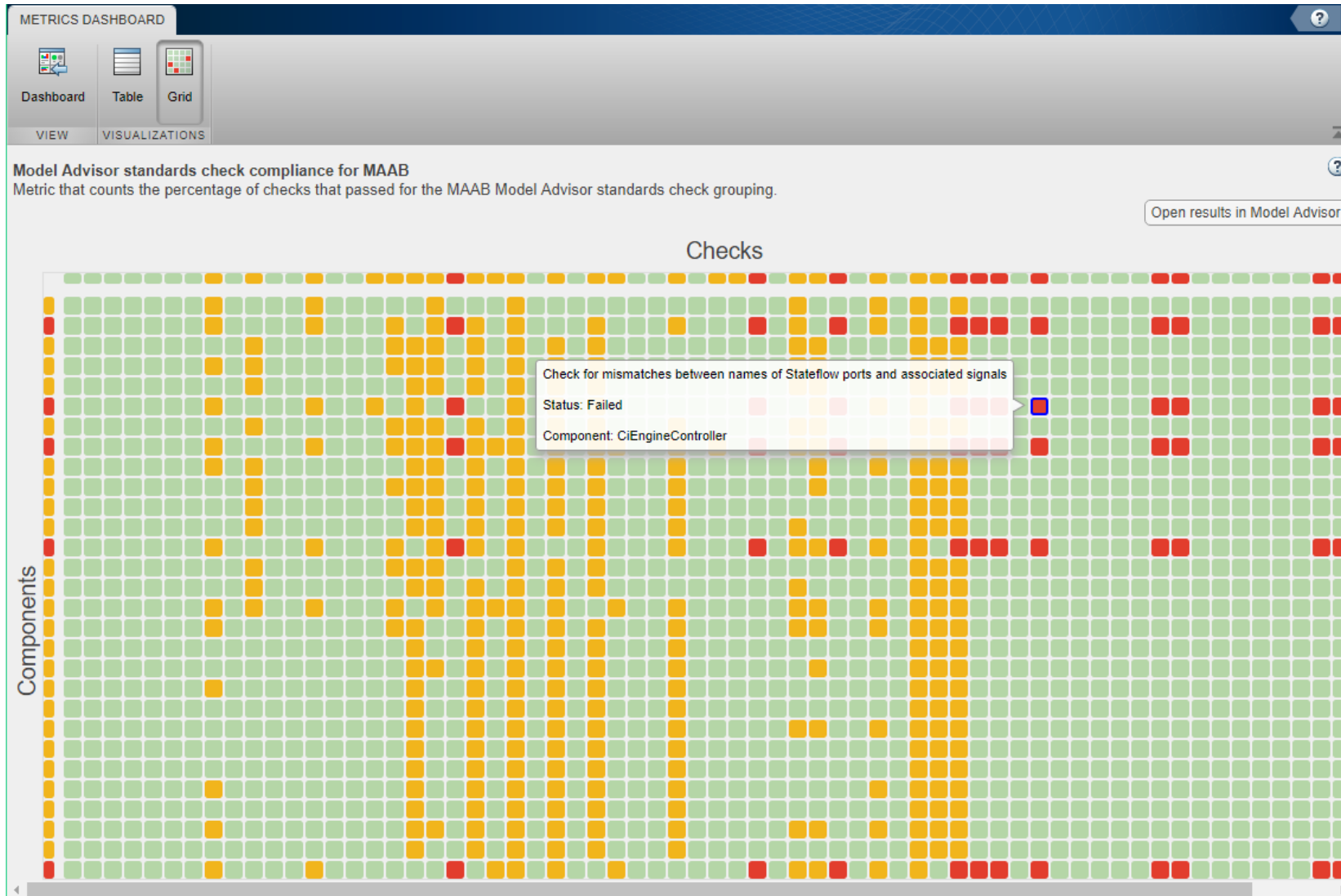


# Assess Quality with Metrics Dashboard



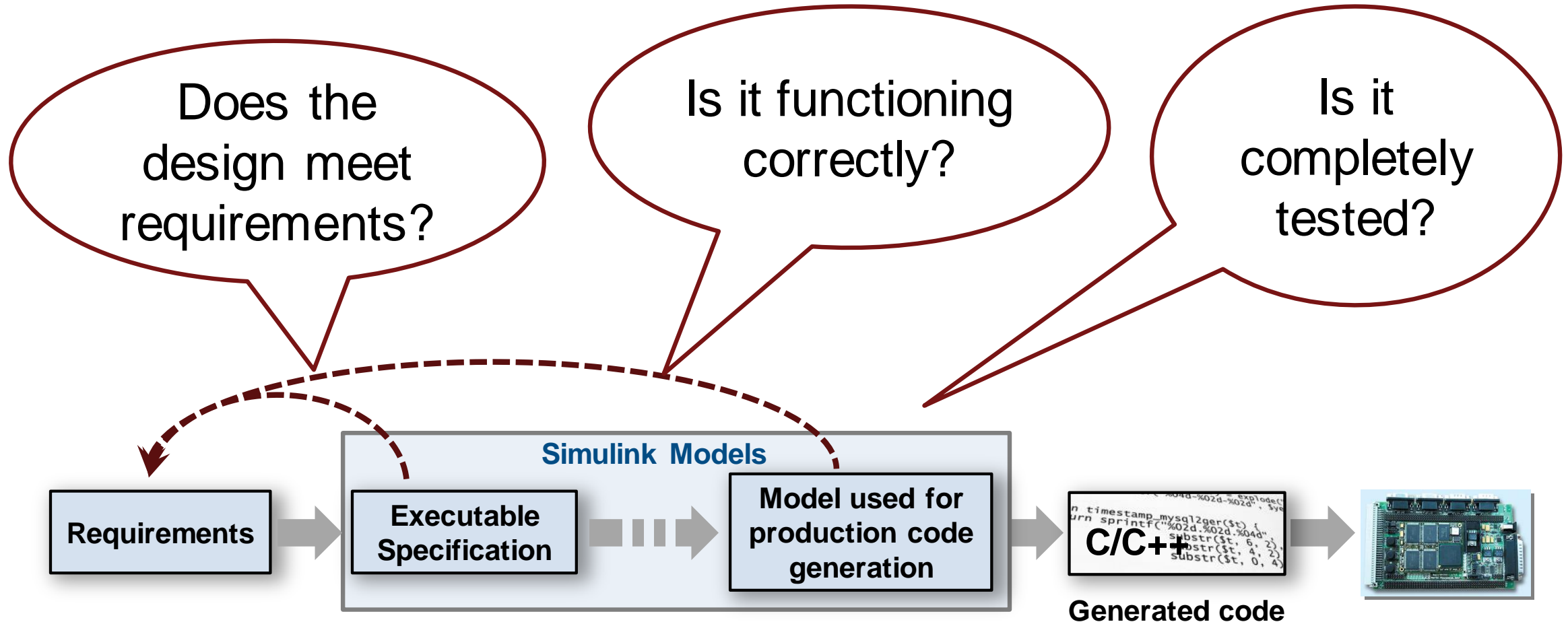
- Consolidated view of metrics
  - Size
  - Compliance
  - Complexity
  
- Identify where problem areas may be

# Grid Visualization for Metrics



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

# Functional Testing



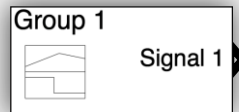
# Systematic Functional Testing

## Test Case

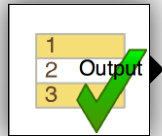
### Inputs



MAT file (input)



Signal Builder



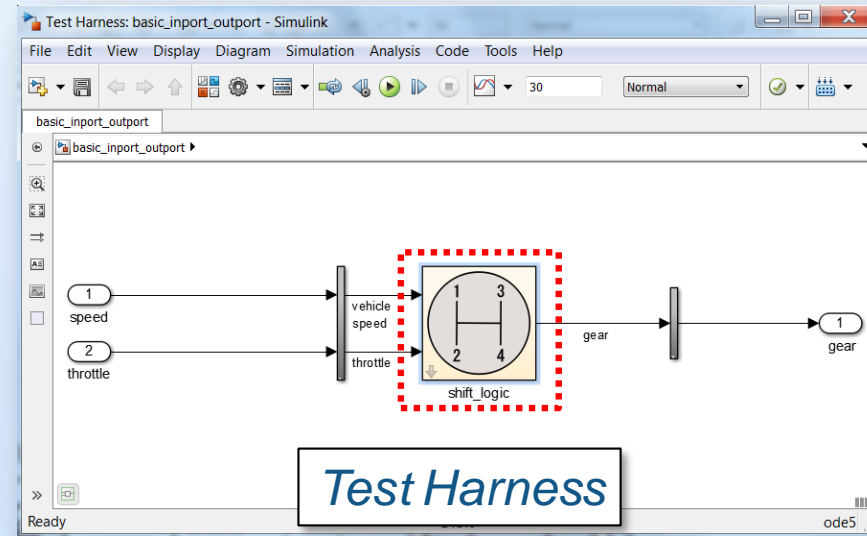
Test Sequence

and more!

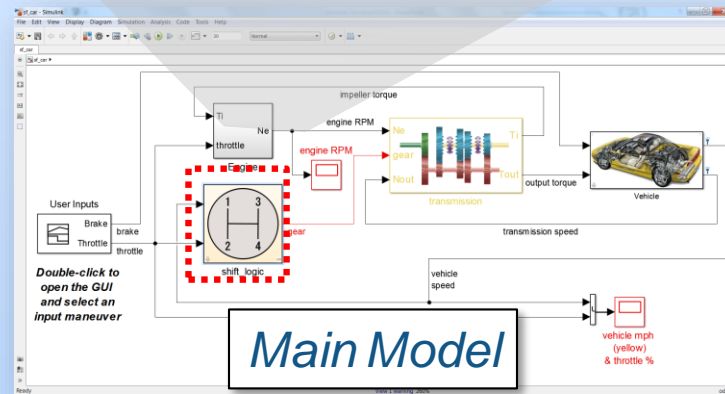


Excel file (input)

R2017b

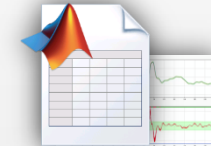


Test Harness



Main Model

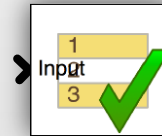
## Assessments



MAT file (baseline)

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

MATLAB Unit Test



Test Assessment

and more!



Excel file (baseline)

R2017b

# Manage Testing and Test Results

**Test Manager**

TESTS

FILE EDIT RUN RESULTS RESOURCES

Test Browser Results and Artifacts

Start Page x Slow Accel x

Filter Tests

- 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\monell\Desktop\...
Enabled	<input checked="" type="checkbox"/>
Hierarchy	ComponentTesting > Fu...
Model	st_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

Start Page x Slow Accel x Comparison x

Filter Results

NAME	STATUS
Results : 2015-Jan-12 17:35:31	2 <input checked="" type="checkbox"/> 1 <input checked="" type="checkbox"/>
Signal Builder Baseline examples	2 <input checked="" type="checkbox"/> 1 <input checked="" type="checkbox"/>
Slow Accel	<input checked="" type="checkbox"/>
Fast Accel	<input checked="" type="checkbox"/>
Baseline Criteria Result	<input checked="" type="checkbox"/>
gear	<input checked="" type="checkbox"/>
throttle	<input checked="" type="checkbox"/>
vehicle speed	<input checked="" type="checkbox"/>
Sim Output (sf_car : normal)	<input checked="" type="checkbox"/>
Decel	<input checked="" type="checkbox"/>

PROPERTY VALUE

Name	gear
Status	<input checked="" type="checkbox"/>
Absolute Tolerance	0
Relative Tolerance	0.00 %
Block Path	SigBdriven/shift_logic

Comparison Plot

Legend: Baseline (Yellow), Compare To (Red)

Y-axis: fourth, third, second, first, None

X-axis: 0 to 30

Tolerance Plot

Legend: Tolerance (Green), Difference (Red)

Y-axis: 0 to 1.0

X-axis: 0 to 30

# Coverage Analysis to Measure Testing

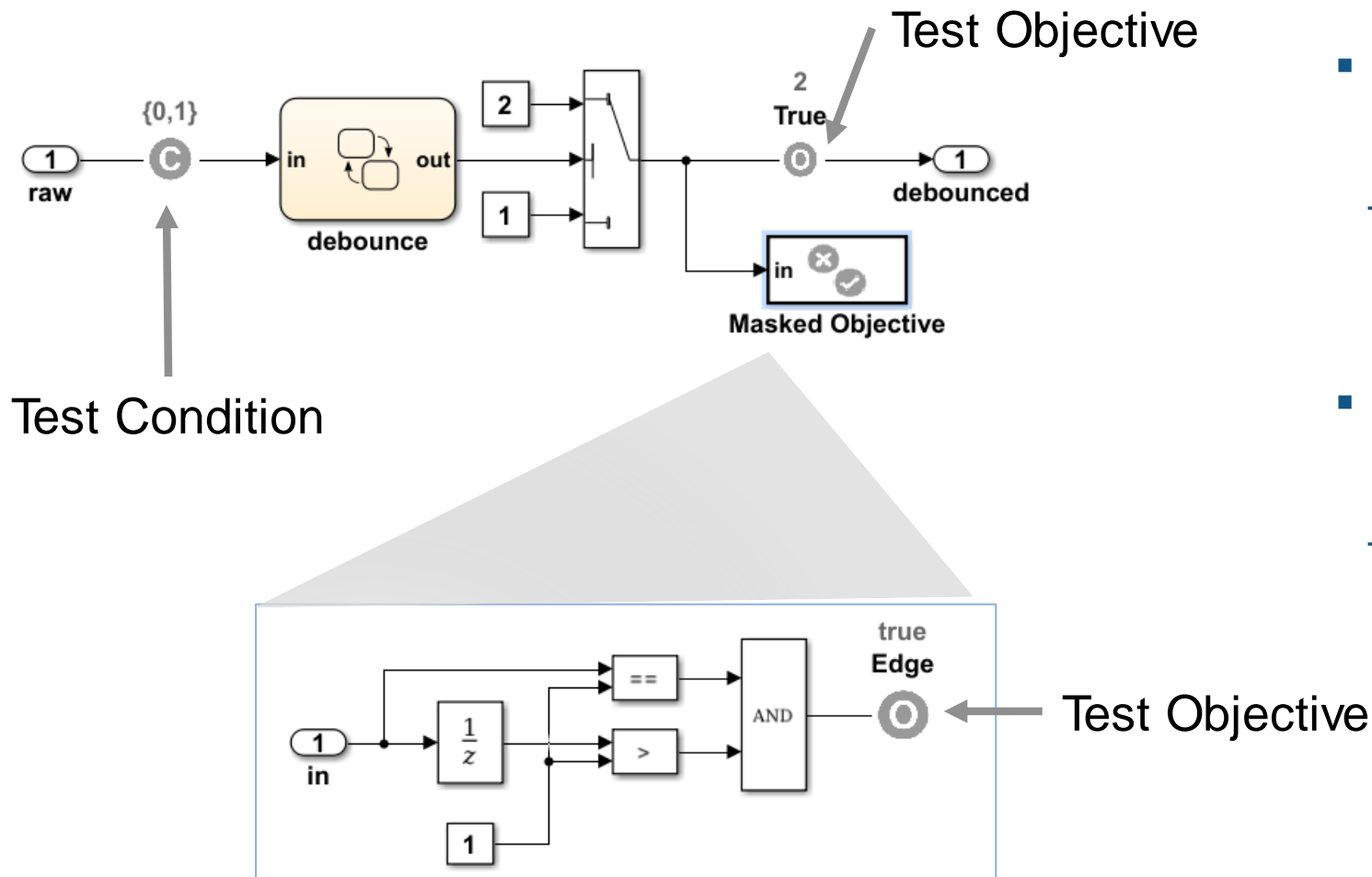
- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Design Errors

**Summary Coverage Reports**

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

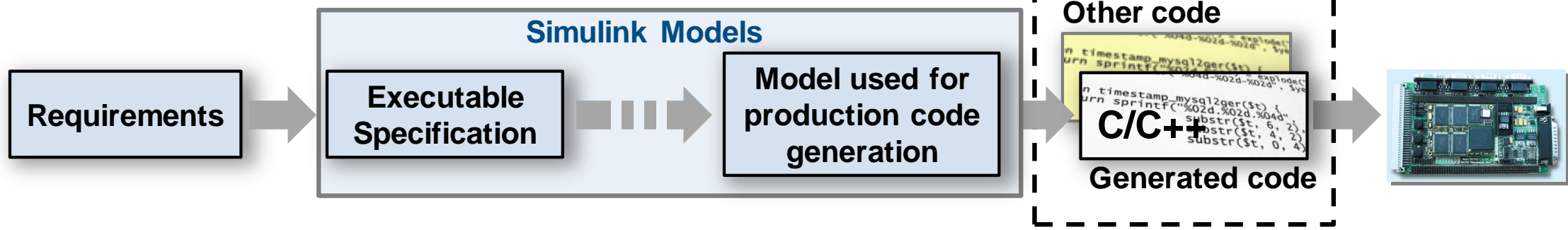
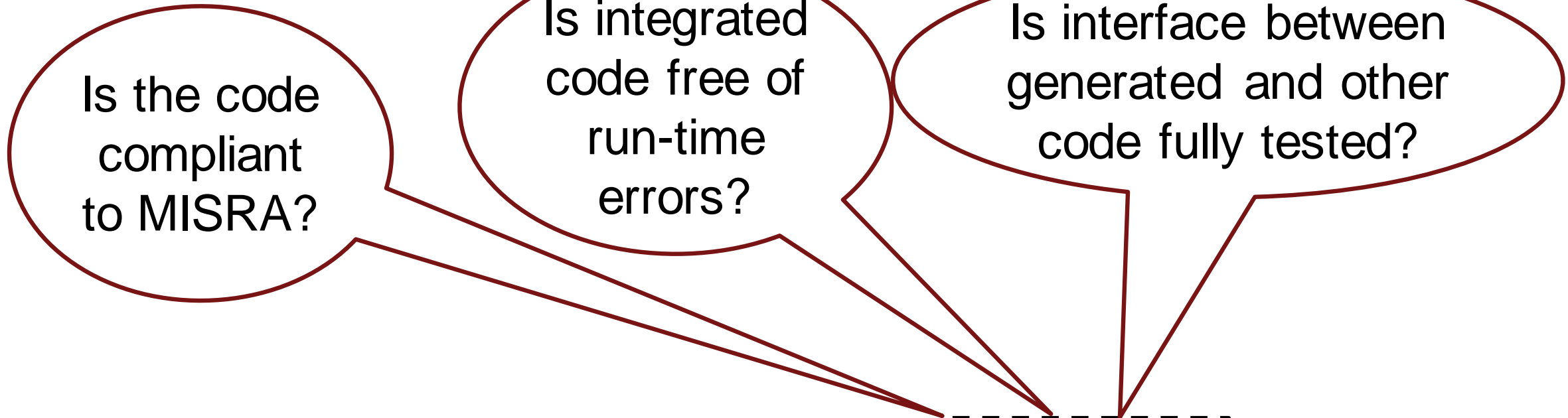


# Test Case Generation for Functional Testing



- 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

# Static Code Analysis



*The Generated Code is integrated with Other Code (Handwritten)*

# 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++;
        }
    }

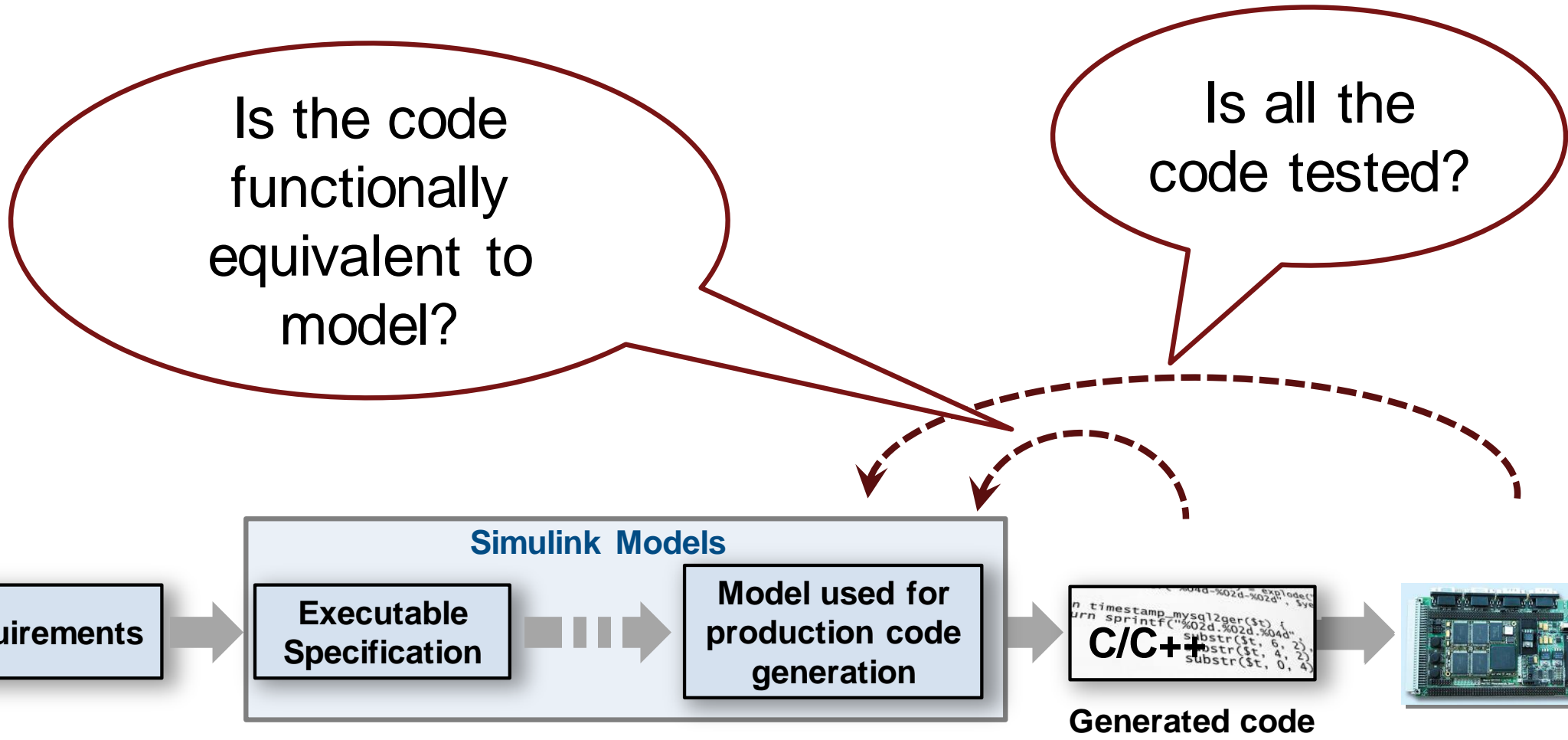
    i = get_bus_status();

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

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

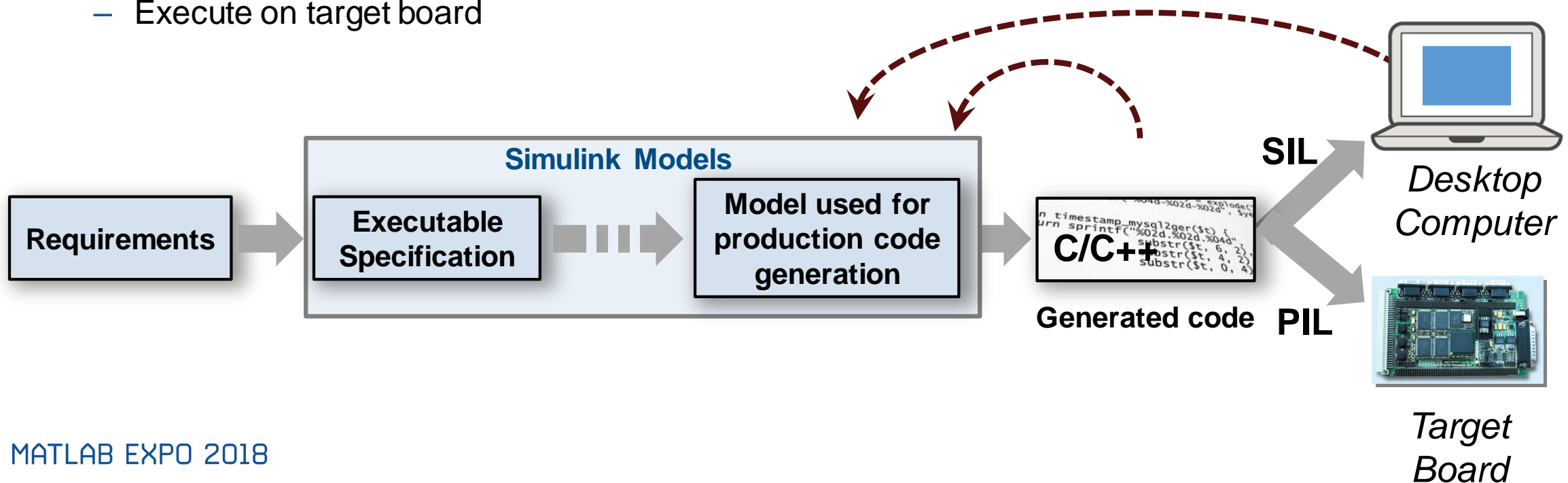
Results from Polyspace Code Prover

# 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
- Collect code coverage



# 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.

# 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*

# Customer References and Applications



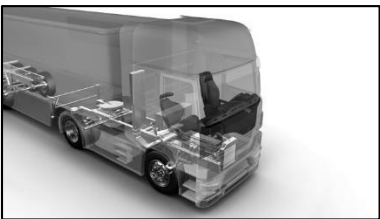
Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design

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

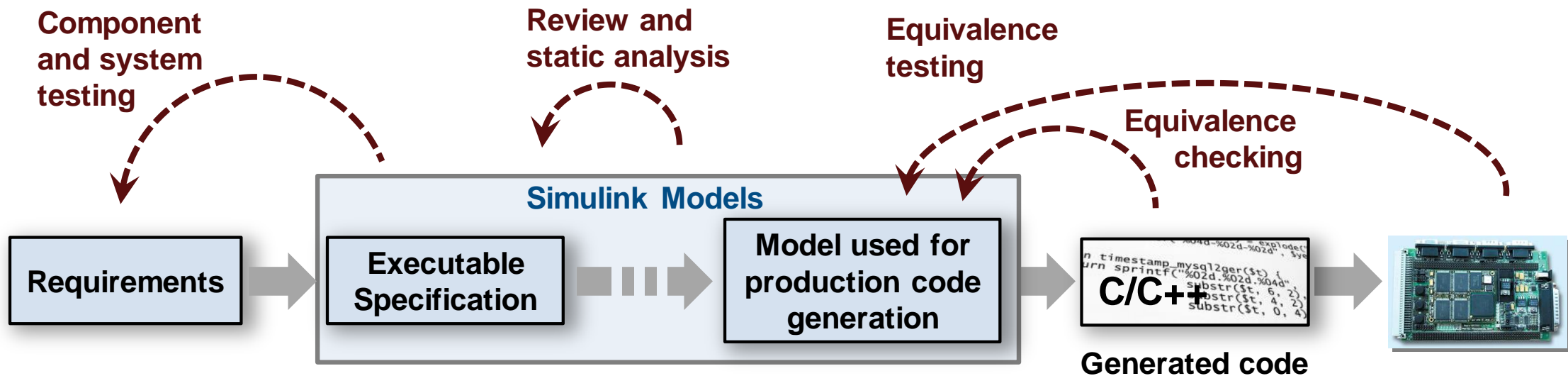
Verification time cut by up to 50 percent

More User Stories: [www.mathworks.com/company/user\\_stories.html](http://www.mathworks.com/company/user_stories.html)



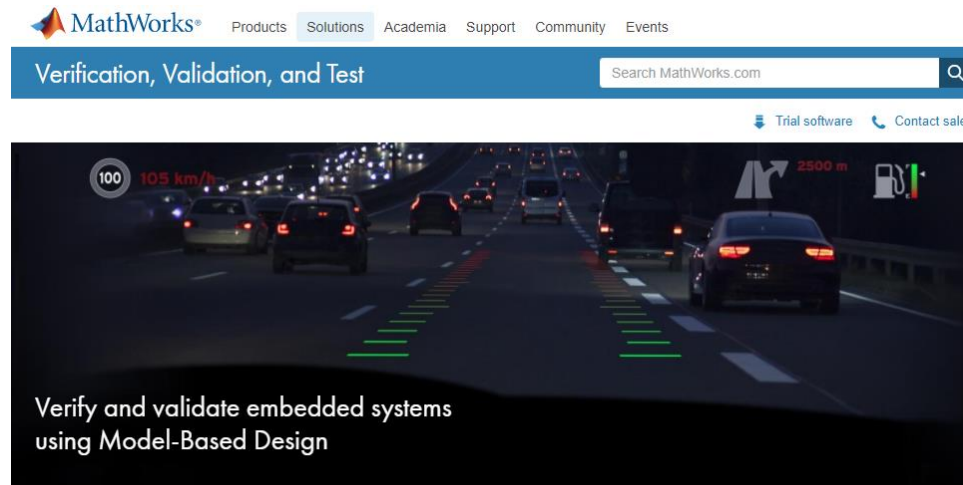
# Summary

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



# Learn More

Visit MathWorks Verification, Validation and Test Solution Page:  
[mathworks.com/solutions/verification-validation.html](https://mathworks.com/solutions/verification-validation.html)



MathWorks® Products Solutions Academia Support Community Events

Verification, Validation, and Test Search MathWorks.com

Trial 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).

# Thank You!