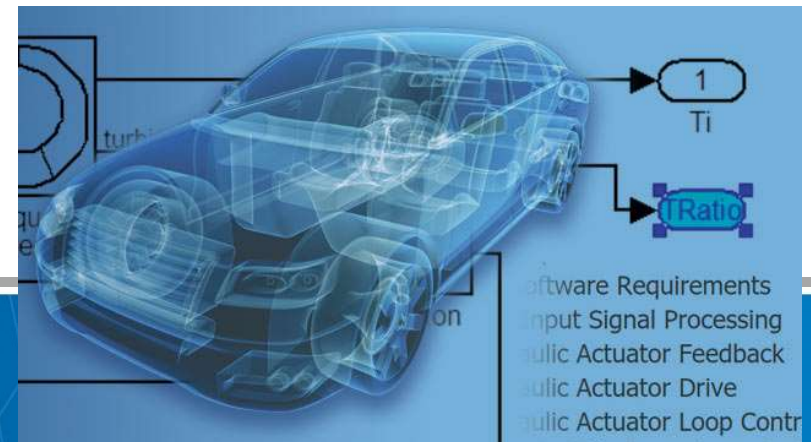
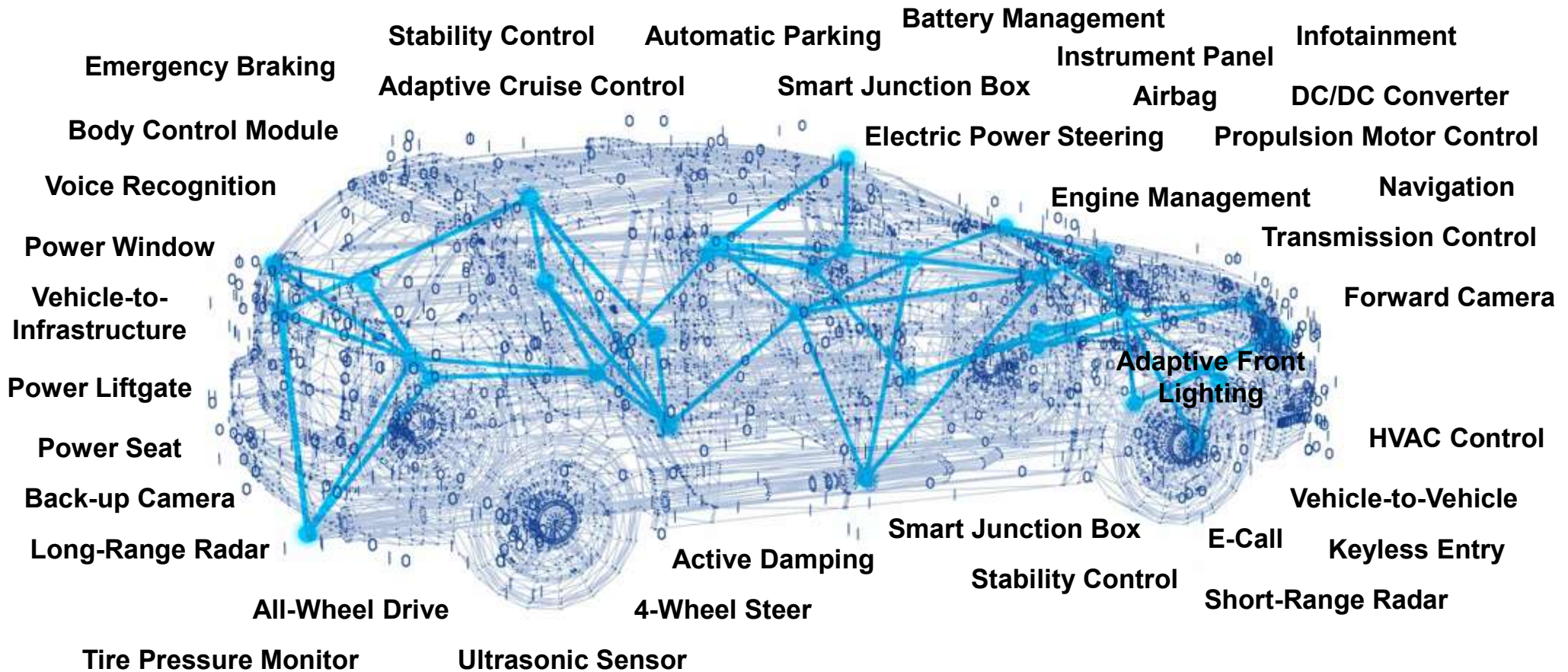


New Requirements, Coverage, and Checking Capabilities with Model-Based Design

Paul Urban
V&V Product Marketing Manager
September, 2017

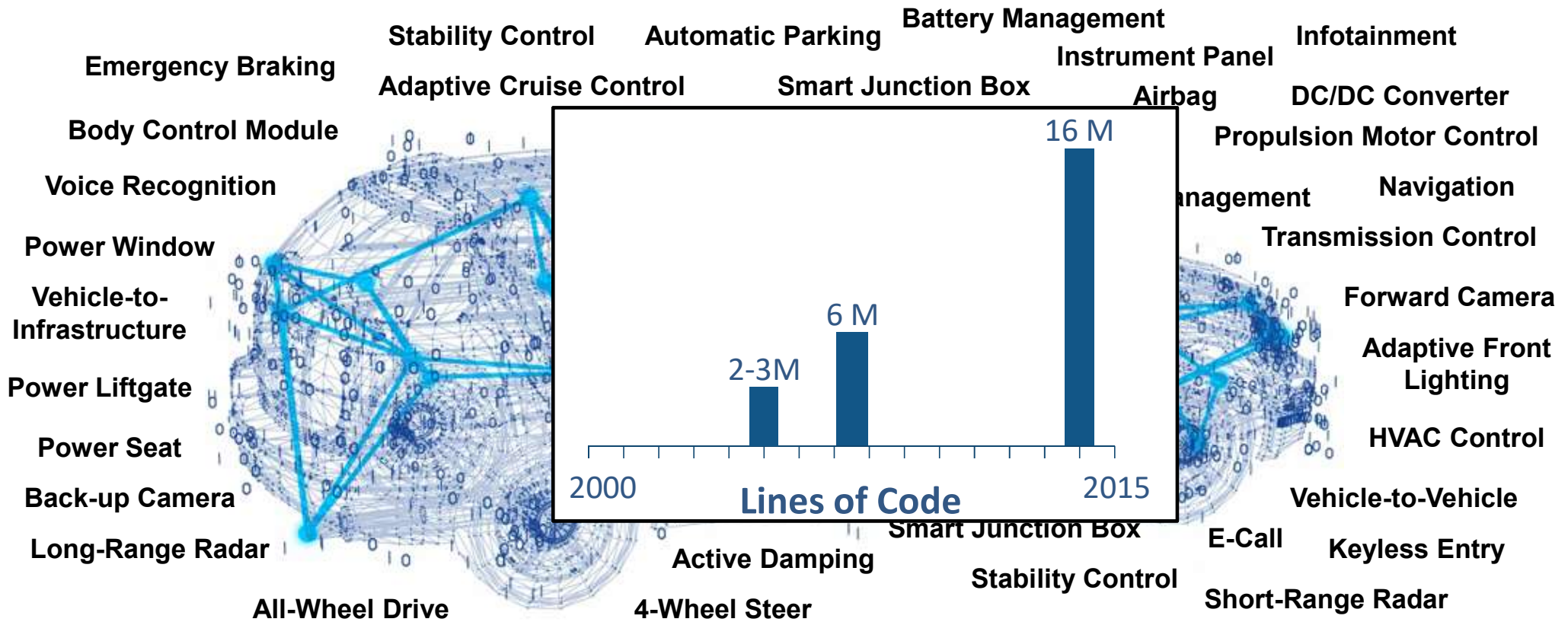


Growing Complexity of Automotive Controls



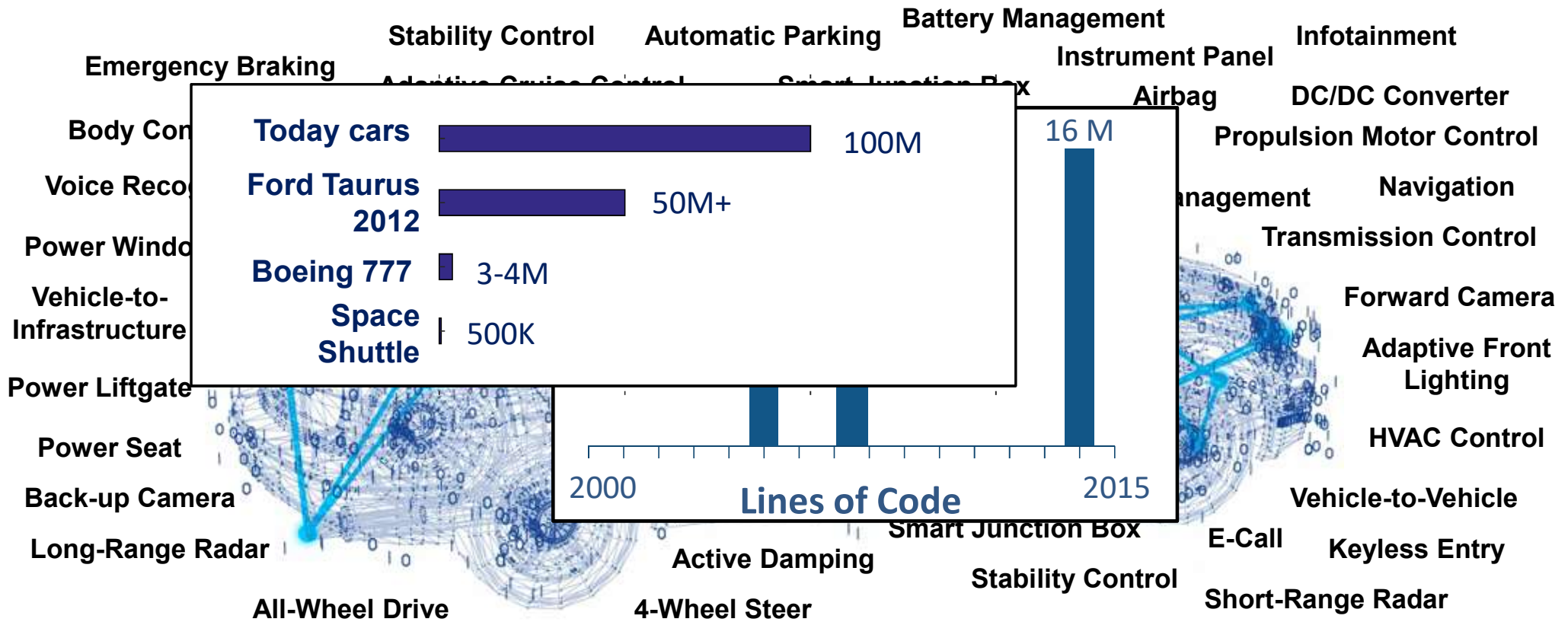
Source of graphic: <http://360.here.com/2013/11/28/putting-firmly-drivers-seat/>

Growing Complexity of Automotive Controls



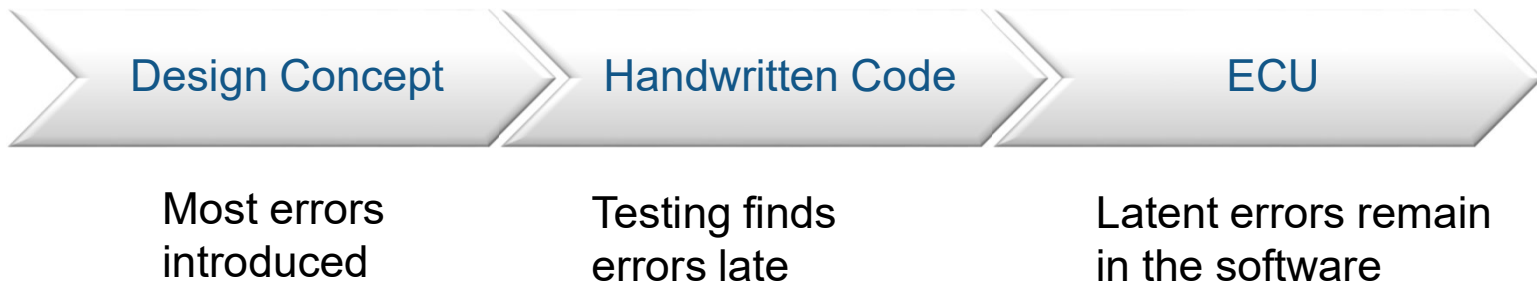
Siemens, "[Ford Motor Company Case Study](#)," Siemens PLM Software, 2014
 McKendrick, J. "[Cars become 'datacenters on wheels', carmakers become software companies.](#)" ZDJNet, 2013

Growing Complexity of Automotive Controls

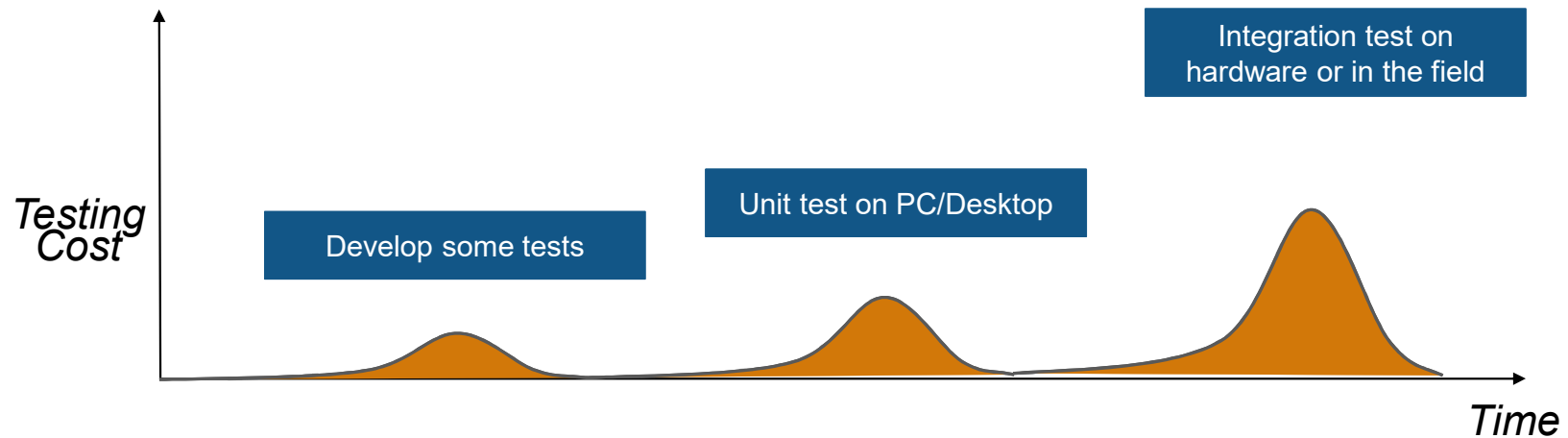


Source:
<https://interact.gsa.gov/sites/default/files/J3061%20JP%20presentation.pdf>

Problems with Traditional Development Process



Cost of finding errors increases over time



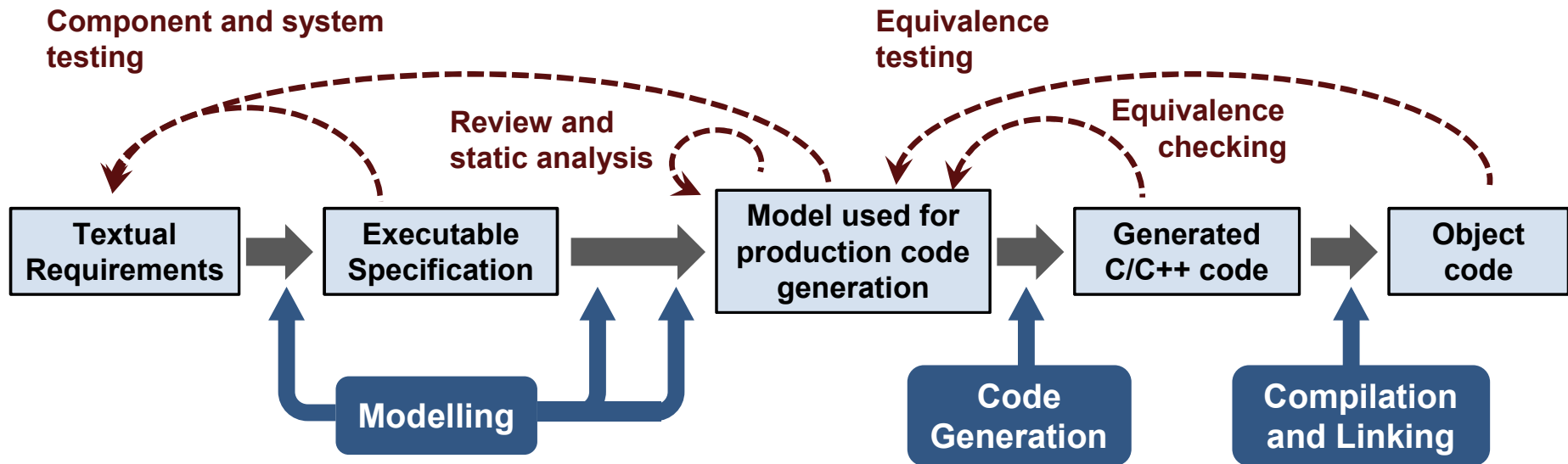
Addressing Challenges with Model Based Design Verification Workflow

1. Find defects earlier
2. Automate manual verification tasks
3. Learn about reference workflow that conforms to safety standards

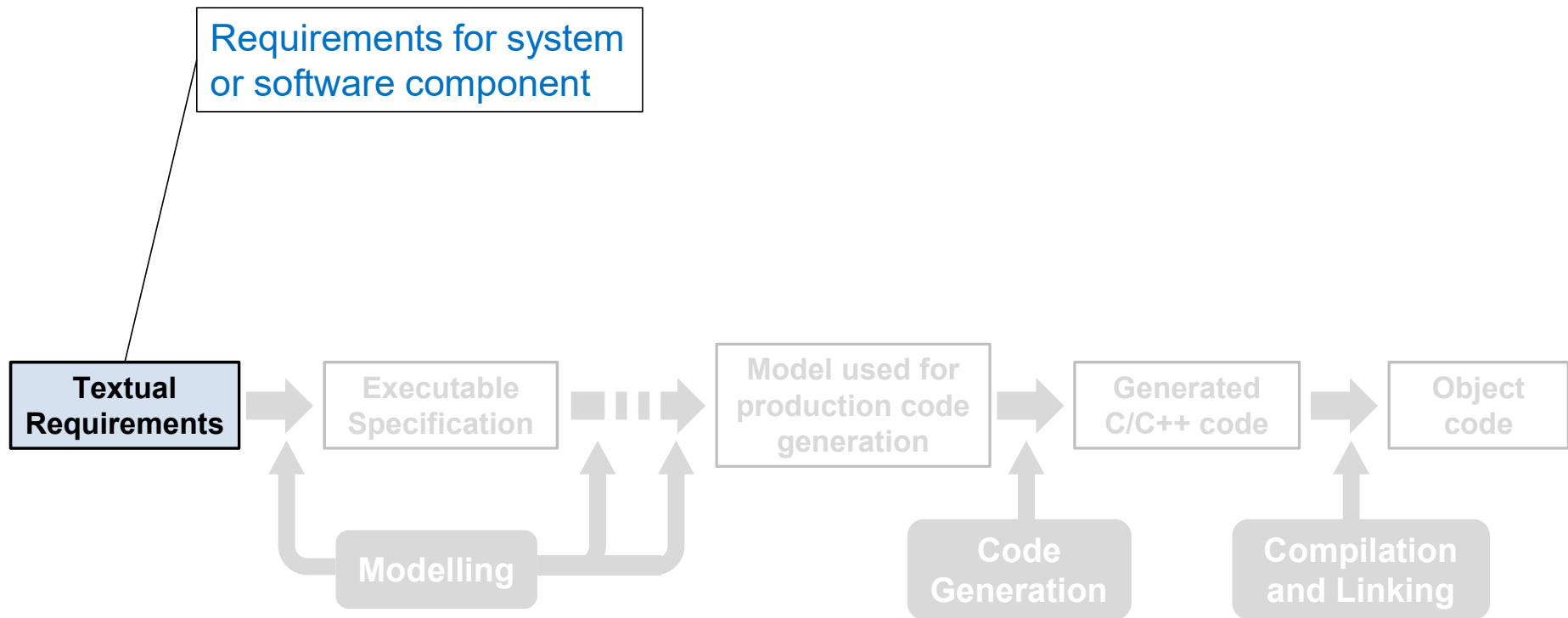
Reference Verification and Validation Workflow

Reference Verification and Validation Workflow

- Certifiable Model-Based Design Workflow to develop critical embedded software
- Reviewed and approved by TÜV SÜD certification authority
- Detailed workflow documented in MathWorks *IEC Certification Kit*

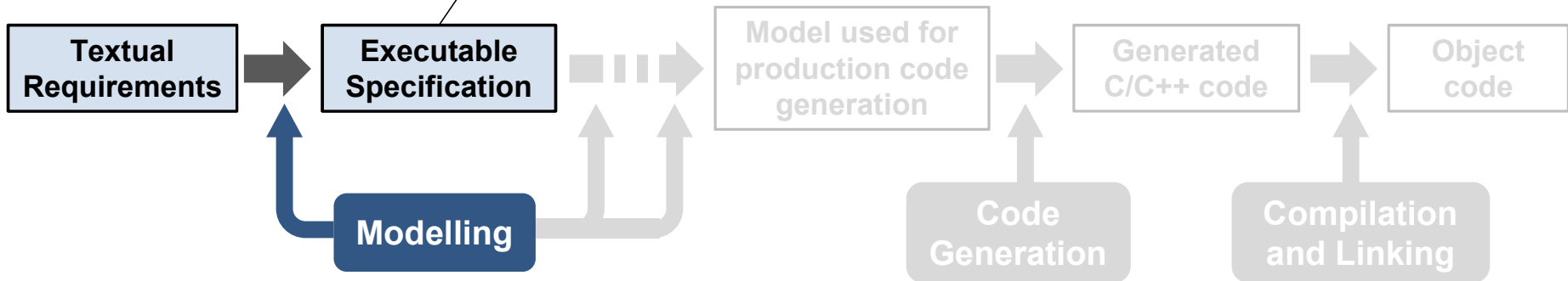


Development Process and Workflow



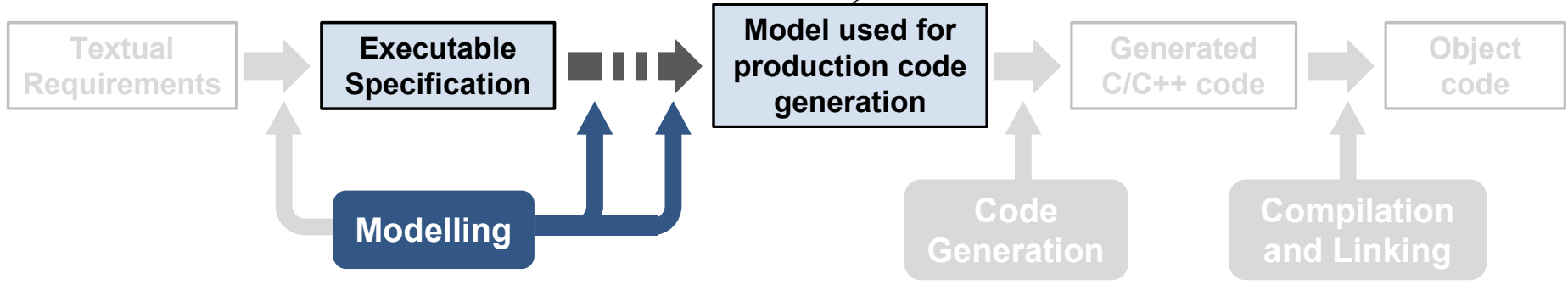
Development Process and Workflow

- Predict dynamic system behavior by simulation
 - System & environment models
 - Precision with floating point
- Use of simulation results for system design
 - Fast What-/If studies
 - Short iteration cycles



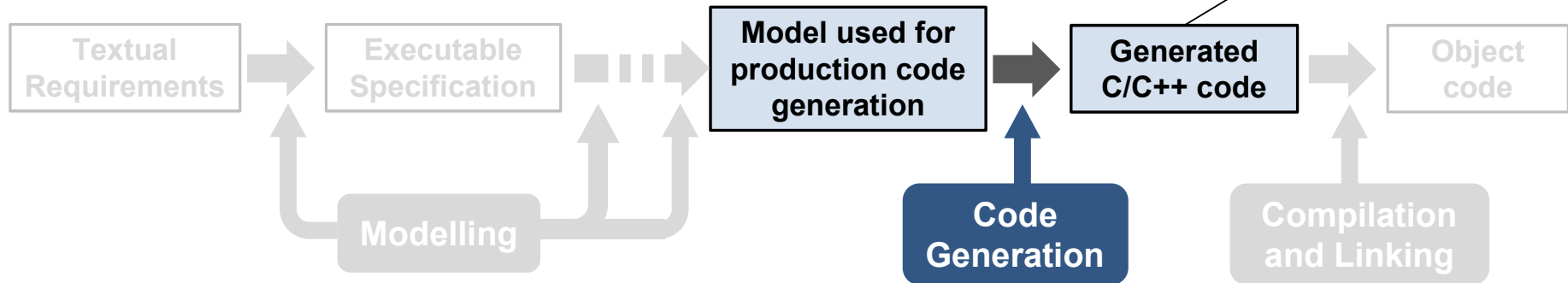
Development Process and Workflow

- Model tuned for target processor
 - Fixed point mathematics, real-time behavior
- Configure for production use
 - Support for standards (e.g. MISRA)

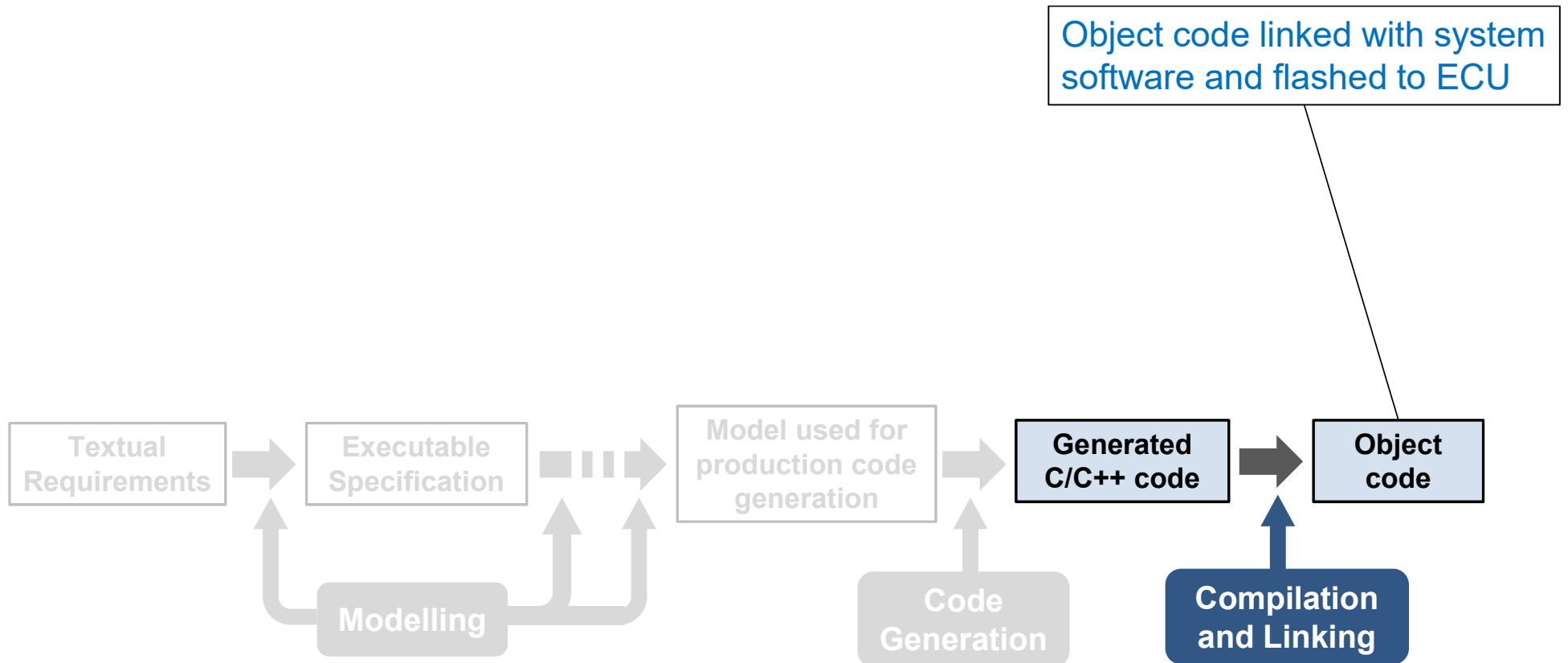


Development Process and Workflow

- Automatically generated code for target processor
 - Optimized, efficient C/C++ code
- Fine grain control of generated code
 - Files, functions, data



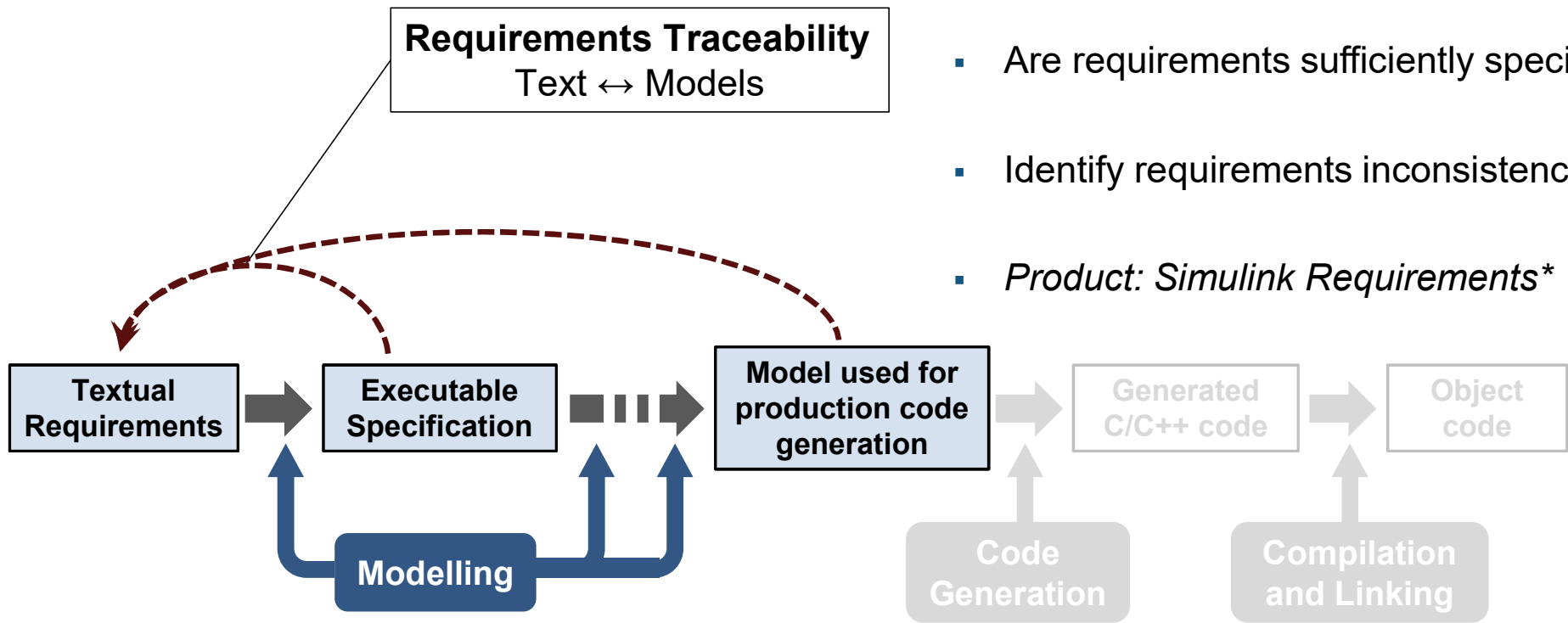
Development Process and Workflow



Verification and Validation Tasks and Activities

Verification and Validation Tasks and Activities

- Find missing or incomplete requirements
- Are requirements sufficiently specified?
- Identify requirements inconsistencies
- *Product: Simulink Requirements**



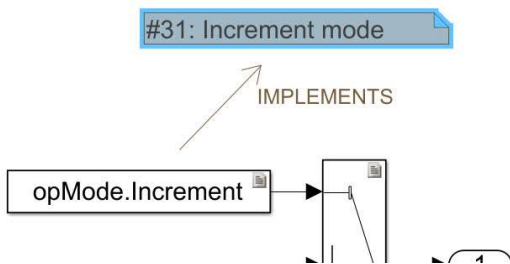
* Customers with Simulink V&V licenses will automatically receive this product

Simulink Requirements

Work with requirements without leaving Simulink

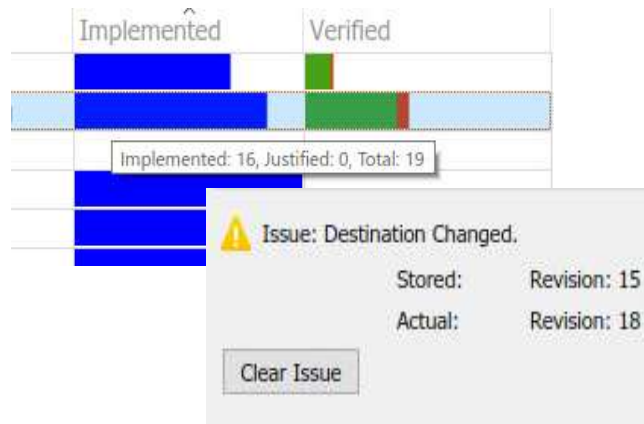
Requirements Capture

- **Author** requirements in Simulink
- **Drag and drop** to create links



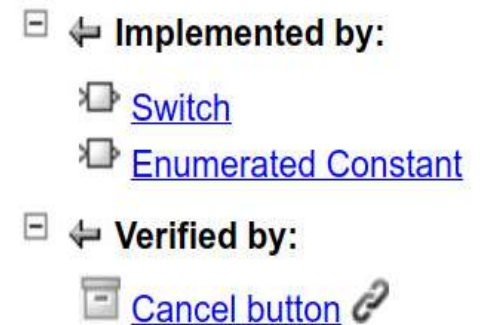
Manage and Analyze Requirements

- **Identify** gaps in design or test
- **Respond** to requirement changes



Requirements Traceability

- **Trace** to design, code and test
- **Understand** impact to design changes



Requirements Perspective

Author, edit and organize requirements

The screenshot displays the Simulink Requirements Perspective interface. It is divided into three main sections:

- Canvas:** Shows a Simulink block diagram with a 'DriverSwRequest' block. A requirement box labeled '#1: Driver Switch Request Handling' is overlaid on the diagram, with an 'IMPLEMENTS' relationship arrow pointing to the 'reqDrv' block. The requirement text reads: 'Handle switch operations by the driver to determine the command for the cruise control system to operate upon.'
- Requirements Browser:** A table listing requirements with their status.

Index	ID	Summary	Implemented	Verified
1	#1	Driver Switch Request Handling	Yes	No
2	#19	Cruise Control Mode	No	No
2.1	#20	Disable Cruise Control system	No	No
2.2	#24	Operation mode determination	No	No
- Property Inspector:** A panel for editing the selected requirement. It shows details for requirement #1, including its index, custom ID, and summary. It also provides a text editor for the description and rationale, and a 'Links' section showing implemented and verified links.

Badges and Markups on canvas

Requirements Property Inspector

Requirements Browser

Requirements Perspective

Explore Requirements and View Status

View Requirement Hierarchy

Create and Open Requirement Sets

Add New Requirements

Implementation Status Roll-up

Verification Status Roll-up

Filter with search

The screenshot shows the Requirements tool interface for a project named 'Requirements - crs_controller'. The interface includes a toolbar with icons for file operations and a search bar. Below the toolbar is a table with columns for Index, ID, Summary, Implemented, and Verified. The table lists several requirements, including 'Driver Switch Request Handling', 'Cruise Control Mode', 'Disable Cruise Control system', and 'Operation mode determination'. The 'Implemented' and 'Verified' columns contain horizontal bars with different colors representing the status of each requirement.

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

Implementation Status

- Blue square: Implemented
- Light blue square: Justified
- White square: No Link

Verification Status

- Green square: Passed
- Red square: Failed
- Orange square: Unknown
- White square: No Test

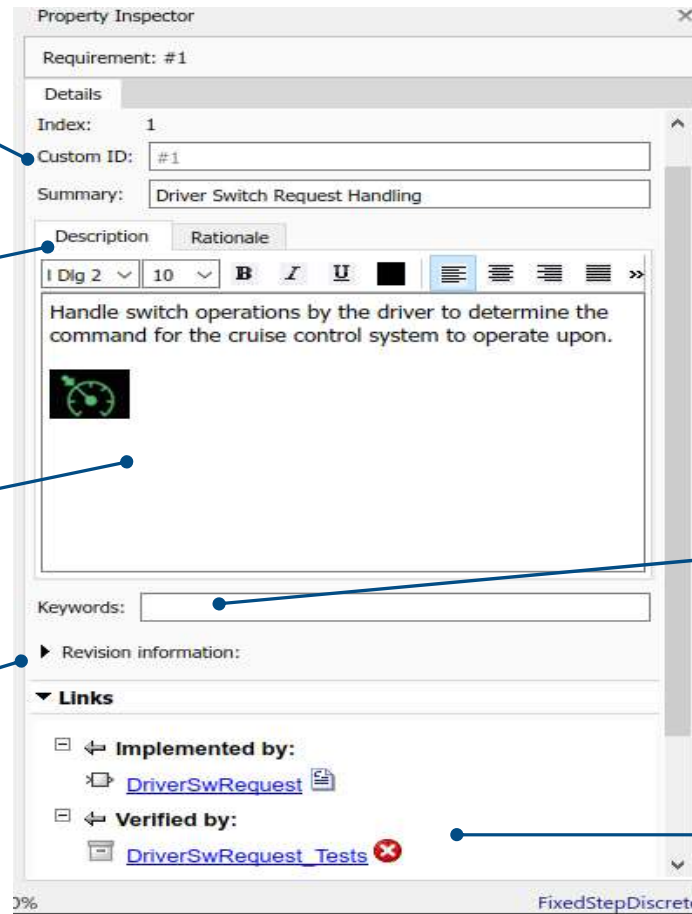
View and Edit Requirement Details

Index, Custom ID & Summary

Description & Rationale

View and Edit with Rich Text

Revision Information



Keywords

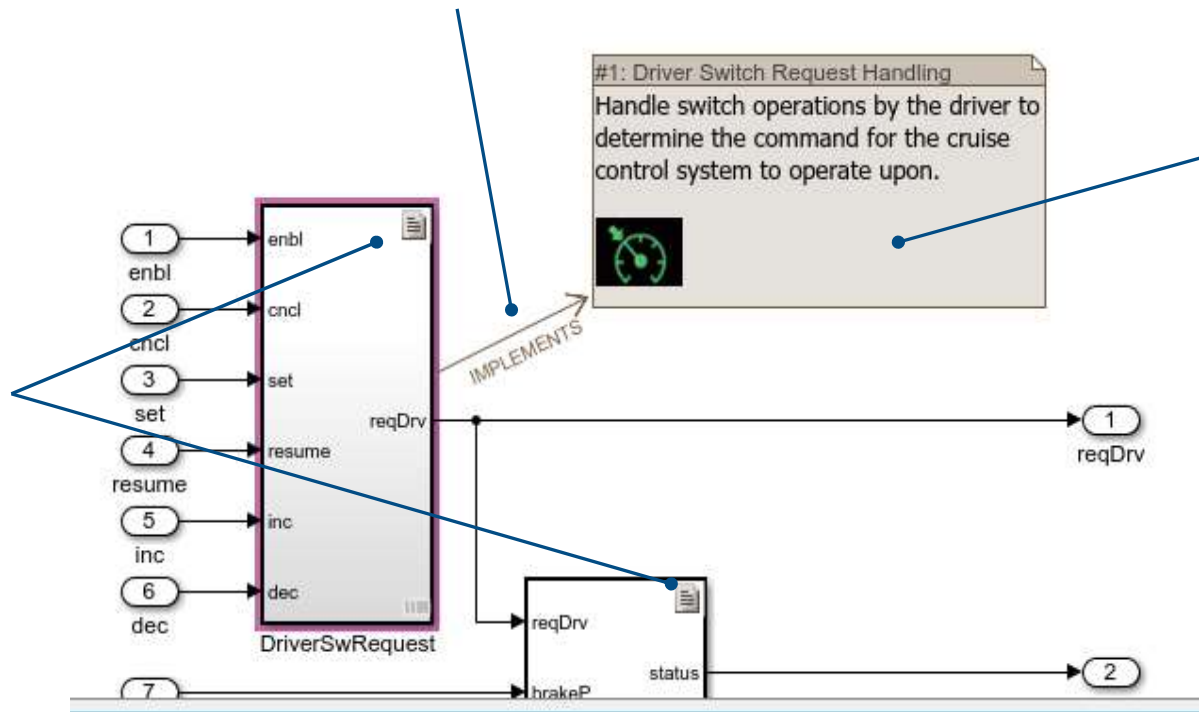
Links Pane with test result

View Requirements and Design Together

Requirement Badges indicate blocks with links

Link Information

Requirement Annotation



- Identify missing traceability
- Communicate requirement details with model

Drag and Drop to Create Links and View Requirements on Diagrams

The screenshot shows the Simulink Requirements tool interface. The top part displays a block diagram with a block named 'reqDrv'. On the left, there are four numbered requirements (1, 2, 3, 4) linked to the block's inputs: 'enbl', 'cncl', 'set', and 'resume'. A requirement labeled '1' is also linked to the 'reqDrv' output. Below the diagram is a 'Requirements - crs_controller' window with a table of requirements.

Index	ID	Summary	Implemented	Verified
crs_req_func_spec*	-	-	<div style="width: 50%; background-color: blue;"></div>	<div style="width: 10%; background-color: green;"></div>
1	#1	Driver Switch Request Handling	<div style="width: 70%; background-color: blue;"></div>	<div style="width: 30%; background-color: green;"></div>
1.1	#2	Switch precedence		

Track Implementation and Verification

Index	Summary	Implemented	Verified
1	Modes of Operation	Blue	Green
2	Establish Communications	Blue	Green
2.1	Crashed or landed time	Blue	Green
3	Initialization	Blue	Green
3.1	Turn off motors	Blue	Green
4	Calibrate the Sensors	Blue	Red
5	Ready for Flight	Blue	Green
6	Track Altitude	Blue	Green
6.1	Ball	Blue	Green
7.1	... modes for entering	Blue	Green
7.2	Track to Land mode af...	Blue	Green
8	Track 3D	Blue	Green
9	Land	Blue	Green
9.1	Landing condition	Blue	Green
10	Crash	Blue	Green

Properties
 Index: 4
 Custom ID: #7
 Summary: Calibrate the Sensors
 Description: The Calibrate Sensors mode shall be entered only from Initialization mode or the Ready for Flight mode when commanded by the ground station. In this mode the quadcopter shall calibrate the gyroscope and accelerometer sensors.

Links
 Implemented by: Calibration
 Verified by: testCalibrationLightOn, testCalibrationLightOff

Implemented by

Verified by

Implemented by:
 Calibration

Verified by:

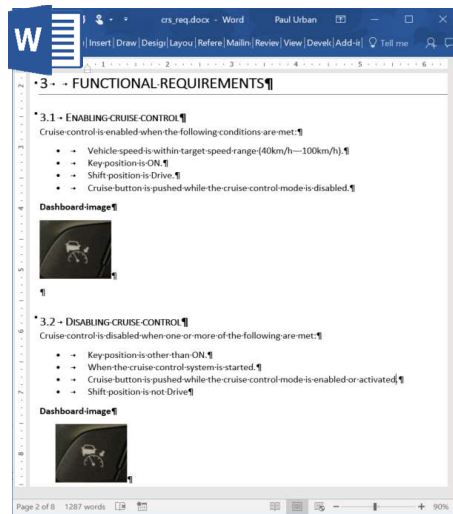
Test Manager

TESTS

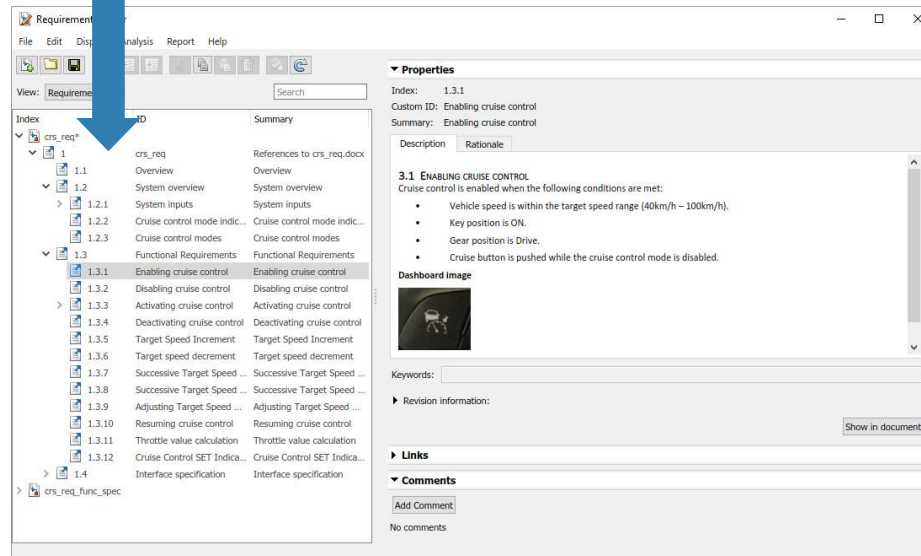
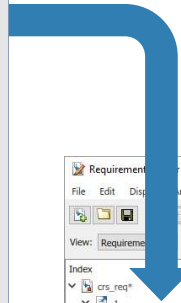
testCalibrationLightOn
 testCalibrationLightOff

Test results

Import External Requirements



Import as reference or for modification



- Access external requirements without leaving Simulink
- Navigate back to source document
- Update synchronizes external document changes

External Requirements

- Word
- Excel
- DOORS

Identify and Respond to Changing Requirements

Requirements with changes are highlighted

3.2_Disablingcruisecont...		#9 Enable Switch Detection
3.3_Activatingcruisecon A...	Derives	#7 Cancel Switch Detection
3.4_Deactivatingcruisec ...	Derives	#8 Set Switch Detection
3.5_TargetSpeedIncreme...	Derives	#8 Set Switch Detection
3.5_TargetSpeedIncreme...	Derives	#11 Increment Switch Detection
3.5_TargetSpeedIncreme...	Derives	#12 Increment Short Switch Detection
3.6_Targetspeeddecreme...	Derives	#15 Decrement Switch Detection
3.6_Targetspeeddecreme...	Derives	#16 Decrement Short Switch Detection
3.7_SuccessiveTargetSpe...	Derives	#13 Increment Long Switch Detection
3.7_SuccessiveTargetSpe...	Derives	#14 Intermediate state
3.8_SuccessiveTargetSpe...	Derives	#17 Decrement Long Switch Detection
3.8_SuccessiveTargetSpe...	Derives	#18 Intermediate state
Changed Source: 0 / 68		Changed Destination: 39 / 68

Take action to address change and clear issue

#12: Increment Short Switch Detecti

Keywords:

Revision information:

Change Information

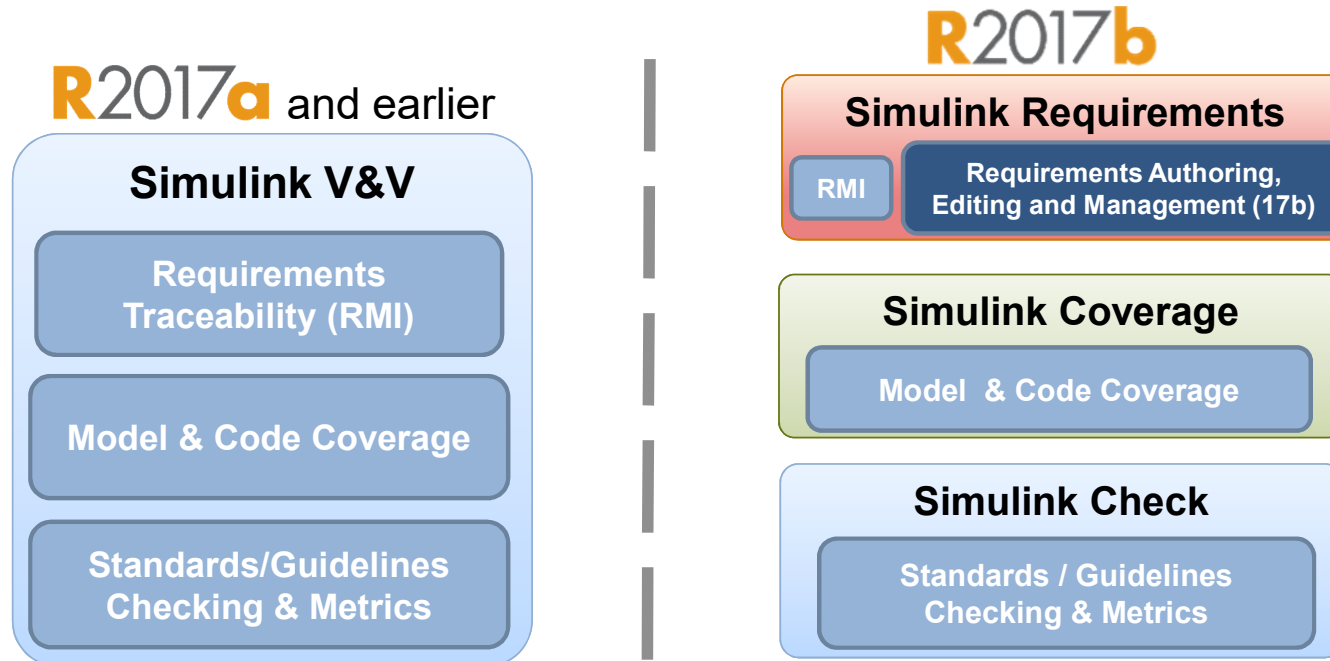
Source: Revision: 1 (Time Stamp: 19-May-2017 18:02:20)

Issue: Destination Changed.

Stored:	Revision: 15 (Time Stamp: 20-May-2017 11:28:11)
Actual:	Revision: 18 (Time Stamp: 20-May-2017 13:15:46)

Clear Issue

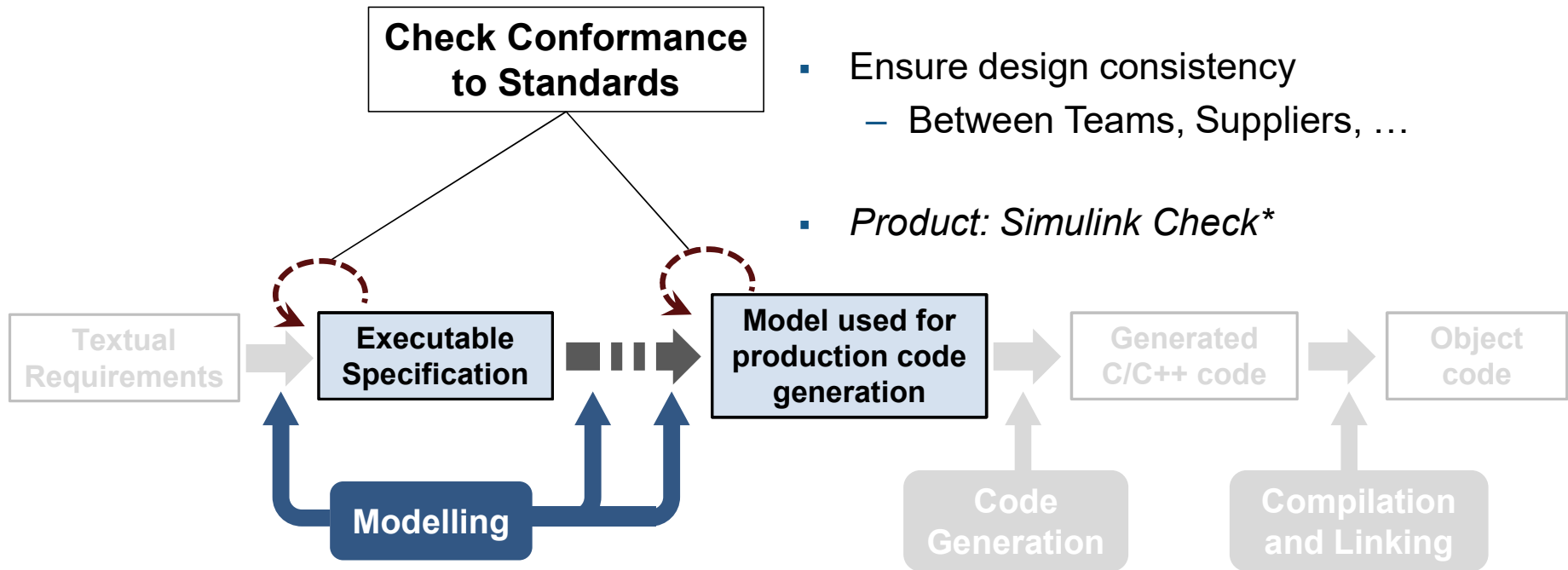
Changes in R2017b aligning products to usage



- New Simulink Requirements product includes RMI
- Model and Code Coverage move to Simulink Coverage
- Static checking, metrics, clone detection move to Simulink Check
- Customers with Simulink V&V licenses will receive all products automatically

Verification and Validation Tasks and Activities

- Check design for various standards
 - MAAB, ISO 26262, MISRA C:2012 ...
- Ensure design consistency
 - Between Teams, Suppliers, ...
- *Product: Simulink Check**



Simulink Check

Automate verification and correct models to improve design

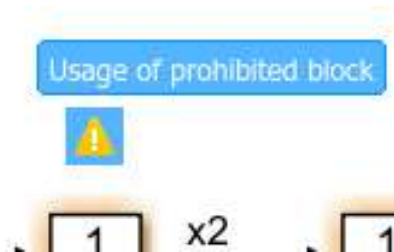
Standards & Guidelines Checks

- **Automate** compliance to standards
- **Customize** checks

- ☑ Modeling Standards for DO-178C/DO-331
- ☑ Modeling Standards for EN 50128
- ☑ Modeling Standards for IEC 61508
- ☑ Modeling Standards for IEC 62304
- ☑ Modeling Standards for ISO 26262
- ☑ Modeling Standards for MAAB

Edit Time Checking

- **Find and fix** compliance issues while you design
- **Avoid** rework later



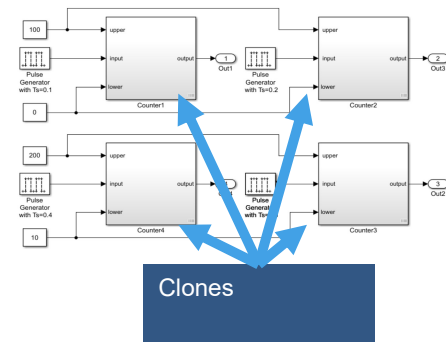
Model Metrics

- **Analyze** complexity, size, reusability
- **Assess** design quality

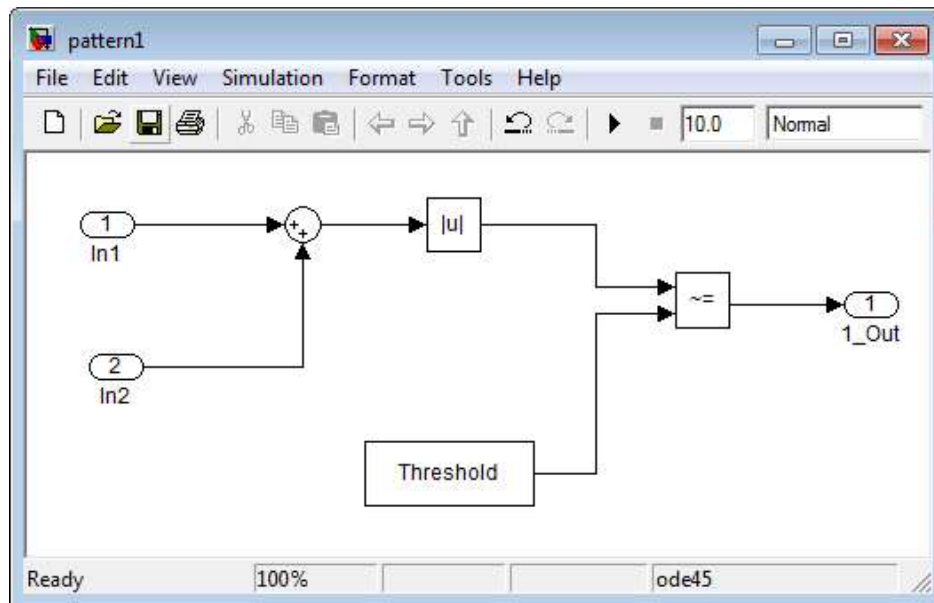


Model Refactoring

- **Find** clones and modeling patterns
- **Refactor** to improve maintainability



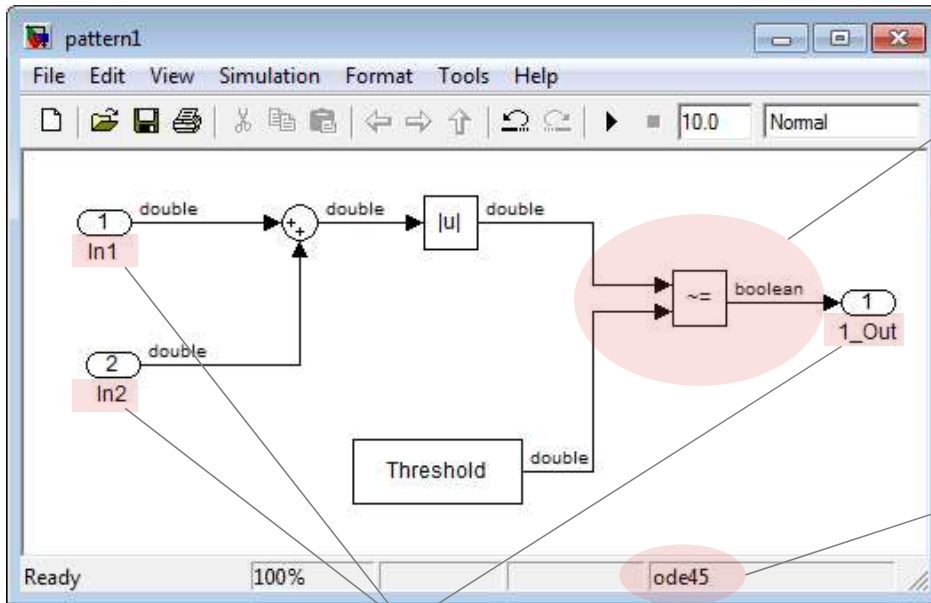
Example



Is there a potential error in this model? It depends...

Example

How about now?



When generating code:

- Floating-point precision issues may lead to incorrect comparison results

Is this a production model?

- Implementation requires a fixed-step, discrete solver

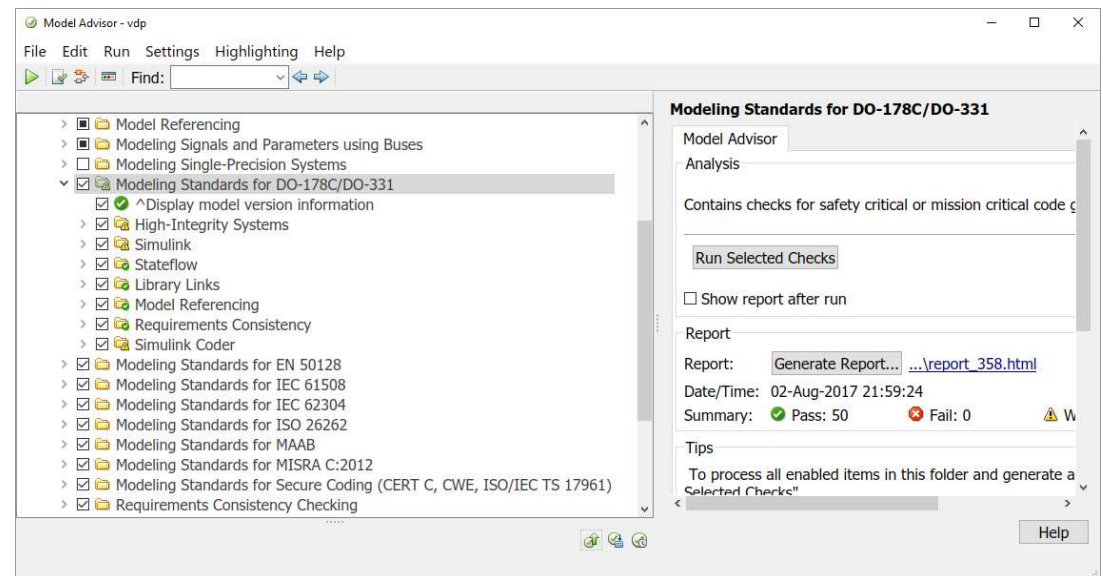
- Ports do not follow established naming conventions

Simulink Check

Automate verification and correct models to improve design

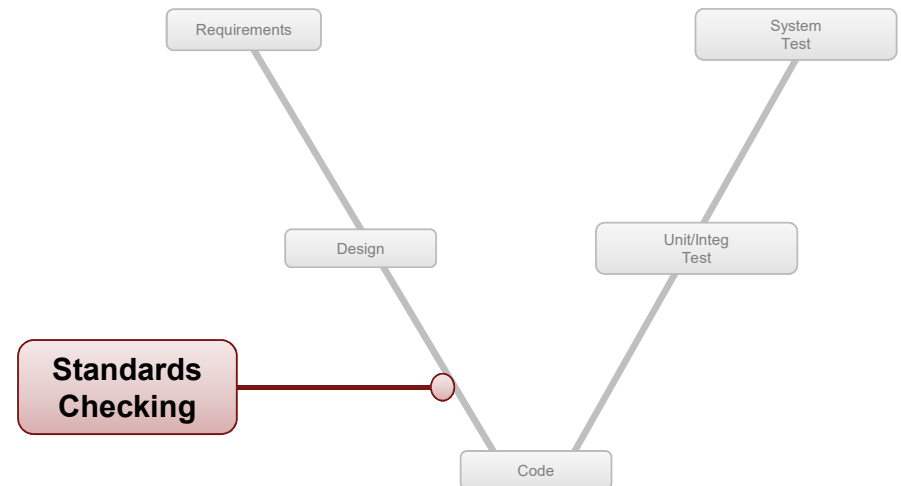
- Automates checking of:
 - Guideline conformance
 - Readability
 - Performance
 - Efficiency
 - Potential errors

- Supports:
 - MAAB Guidelines
 - High Integrity Guidelines
 - Code Generation Guidelines
 - Custom standards creation
 - Standards:
 - ISO 26262, IEC 61508, MISRA C:2012,
 - CERT C, CWE, ISO/IEC TS 17961
 - DO-178/DO-331, IEC 62304 , EN 50128



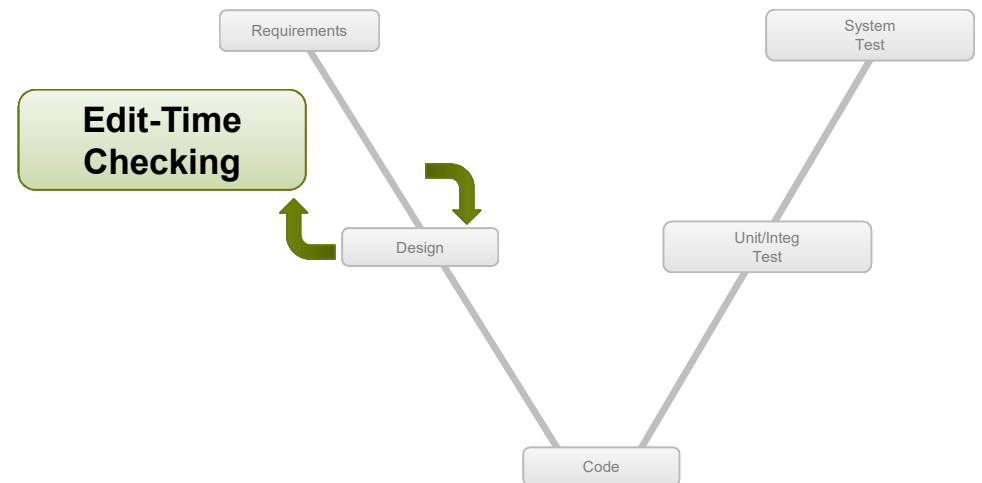
Waiting to Check Standards Leads to Rework

- Checks for standards and guidelines are often performed late
- Typically after design complete and before code generation
- Results in finding many errors
- Leads to last minute rework

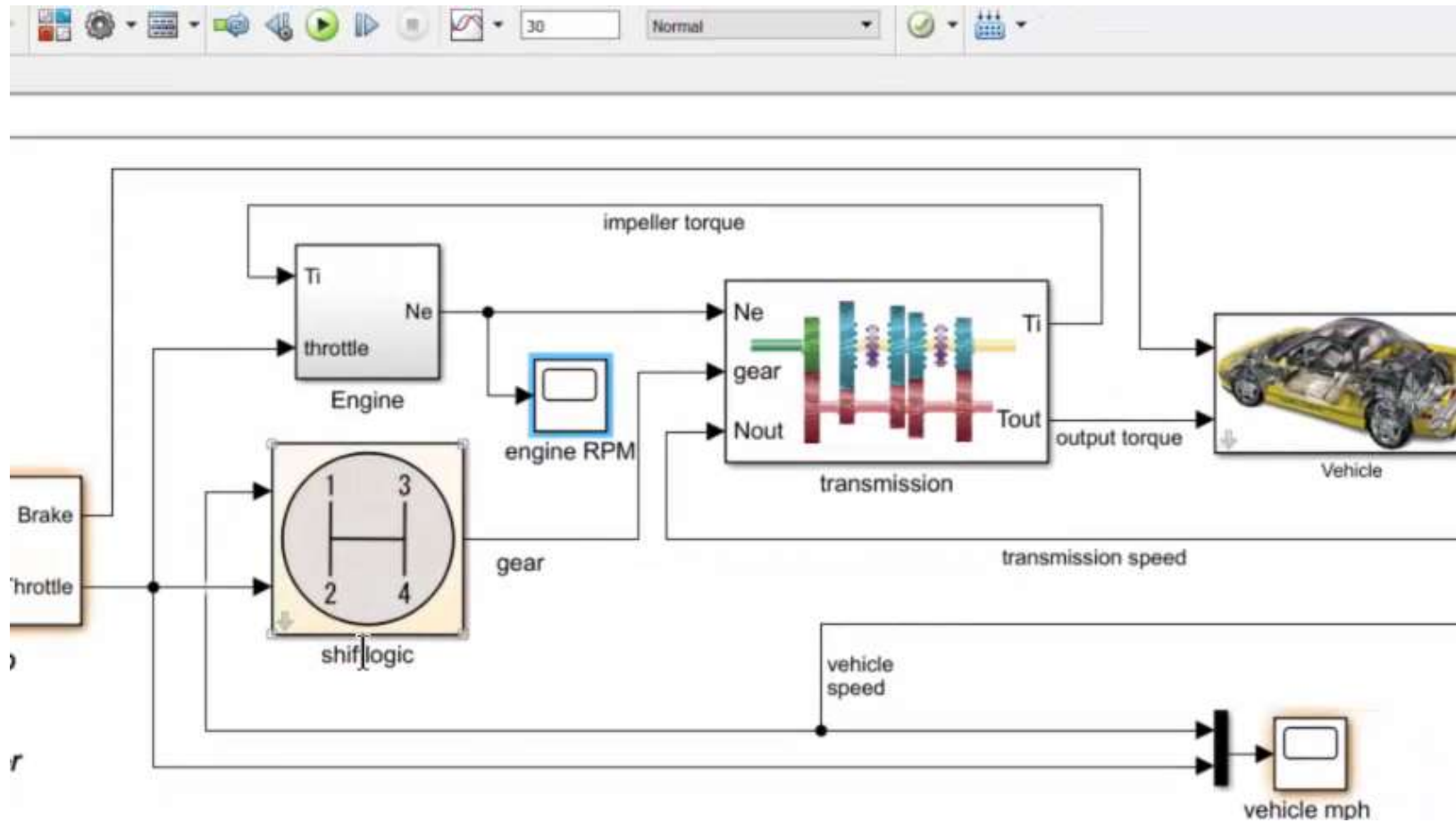


Fix Errors As You Go With Edit-Time Checking

- Violations are highlighted as you edit
 - Similar to grammar checking in text editor
- Quickly address compliance and modeling standard issues
- Avoid rework later to meet standards



Find Compliance Issues as you Edit with Edit-Time Checking



How to measure MBD quality?

Direct measures

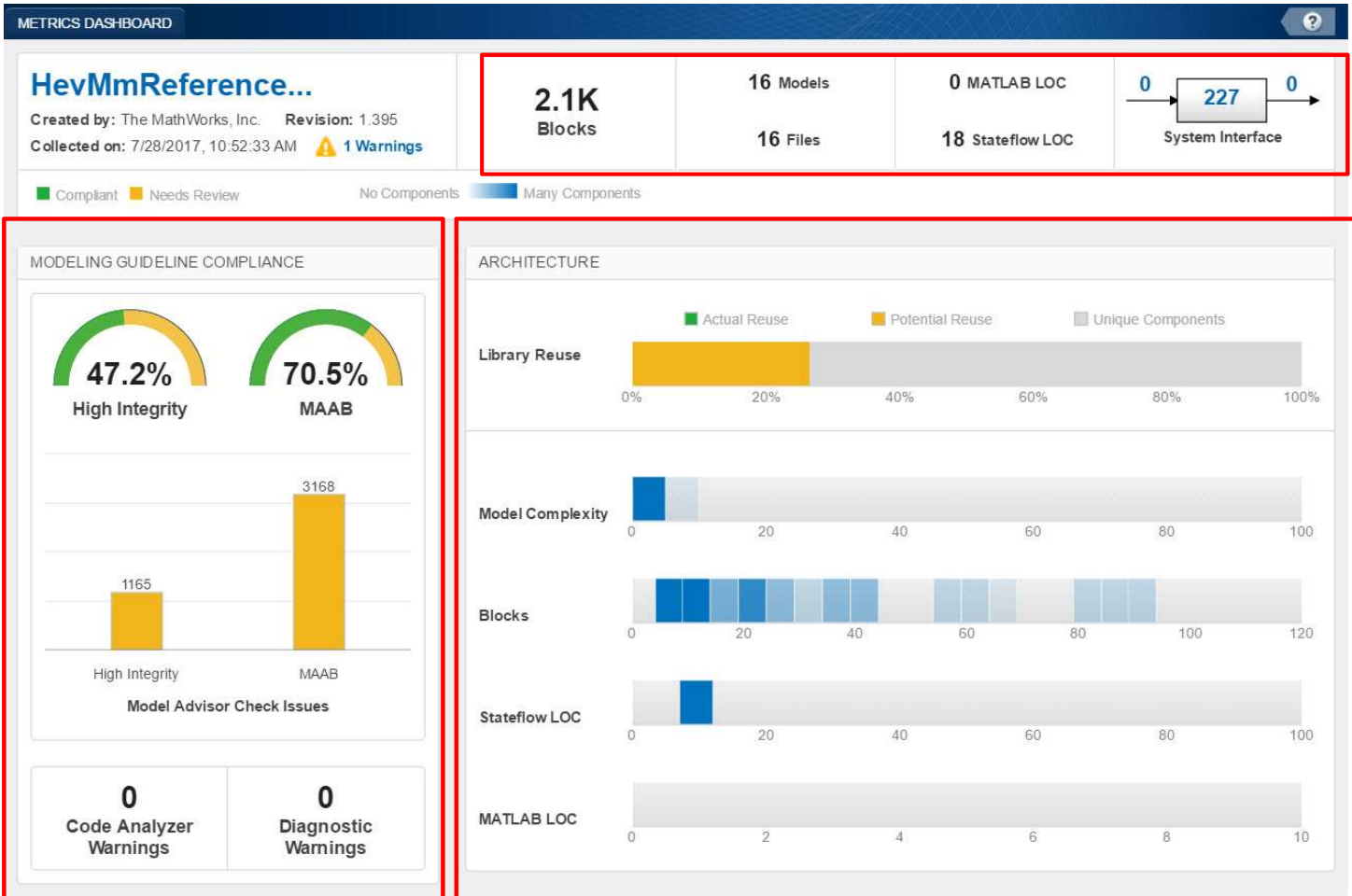
- Number of blocks
- Number of data objects
- McCabe model complexity
- Signal coupling
- Number of line crossings
- Number of requirements
- ...

Indirect measures

- Number of passed Model Advisor checks
- Number of passed test cases
- Model decision coverage
- Number of design errors
- ...

View Metrics with Metrics Dashboard

R2017b



- Consolidated view of metrics
 - Size
 - Compliance
 - Complexity
- Assess quality of the model
- Identify where problem areas may be

Explore Metric Details

METRICS DASHBOARD

HevMmReference...
Created by: The MathWorks, Inc. Revisor: 1.395
Collected on: 7/28/2017, 10:52:33 AM 1 Warnings

2.1K Blocks
16 Models
0 MATLAB LOC
16 Files
18 Stateflow LOC

Modeling Guideline Compliance: 47.2% High Integrity, 70.5% MAAB

Architecture: Library Reuse, Model Complexity, Blocks, Stateflow LOC, MATLAB LOC

Simulink Clone Count

Component / Clone Patterns	Path	Clones	Clones (incl. Descendants)
HevMmReferenceApplication		0	18
HevMmReferenceApplication/Passenger Car		0	16
HevMmReferenceApplication/Passenger Car/Engine		0	14
SIEngine		3	5
SIEngineCore		3	4
SIMappedEngine		2	3
CMappedEngine		2	2
HevMmReferenceApplication/.../senger Car/Electric Plant		0	2
HevMmReferenceApplication/.../rain Control Module (PCM)		0	2
HevMmReferenceApplication/.../gine Control Module (ECM)		0	2
HevMmReferenceApplication/Controllers		0	2
HevMmReferenceApplication/.../Car/Electric Plant/Motor		0	1
GeDynamic		1	1
SIEngineCore		1	1
MoDynamic		1	1
SIMappedEngine/Three-Way Catalyst		0	1
SIEngine/Three-Way Catalyst		0	1
SIEngineController		0	1
SIEngineCore		0	1

Cyclomatic complexity

Type	Component	Path	Qty	Model Complexity	Model Complexity (incl. Descendants)
SubSystem	Longitudinal Driver	HevMmReferenceApplication/Longitudinal Driver	1	8	8
SubSystem	Emission Calculations	HevMmReferenceApplication/.../Emission Calculations	1	4	4
SubSystem	Performance Calculations	HevMmReferenceApplication/.../Performance Calculations	1	4	4
Model	HevMmReferenceApplication		1	2	18
SubSystem	Engine Plant Input	HevMmReferenceApplication/.../er Car/Engine Plant Input	1	0	0
SubSystem	Powertrain Control Module (PCM)	HevMmReferenceApplication/.../rain Control Module (PCM)	1	0	0
SubSystem	Drivetrain Output	HevMmReferenceApplication/.../ger Car/Drivetrain Output	1	0	0
SubSystem	Drivetrain Plant Input	HevMmReferenceApplication/.../Drivetrain Plant Input	1	0	0
SubSystem	Electric Plant	HevMmReferenceApplication/.../senger Car/Electric Plant	1	0	0
SubSystem	Generator	HevMmReferenceApplication/.../Electric Plant/Generator	1	0	0
SubSystem	Motor	HevMmReferenceApplication/.../Car/Electric Plant/Motor	1	0	0
SubSystem	Controllers	HevMmReferenceApplication/Controllers	1	0	0
SubSystem	Electric Plant Output	HevMmReferenceApplication/.../Car/Electric Plant/Output	1	0	0
SubSystem	Passenger Car	HevMmReferenceApplication/Passenger Car	1	0	0

Model Advisor Check Issues: 0 Code Analyzer Warnings, 0 Diagnostic Warnings

Guidelines: % of checks passed

Clones

Complexity

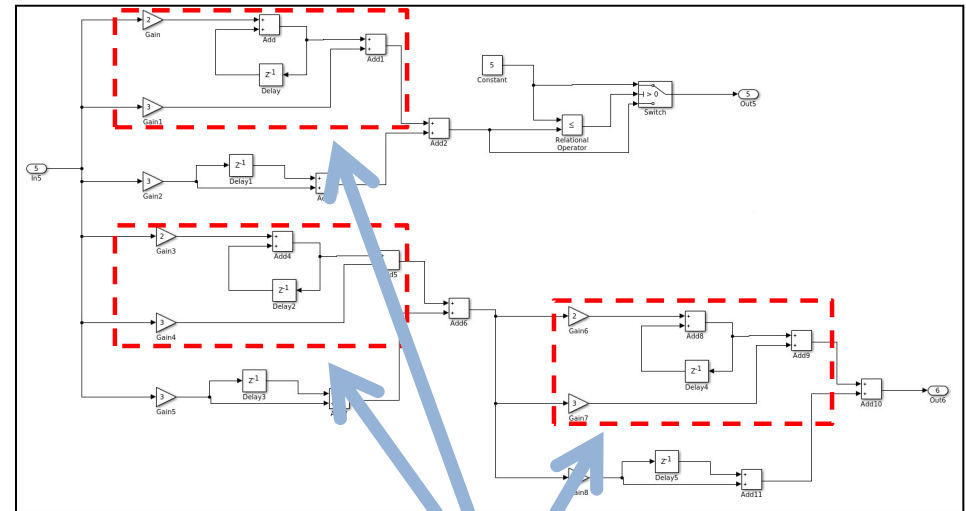
Guidelines

- Drill down in dashboard for detailed metrics
- Uncover errors earlier

Detecting Copy and Paste (Clone) Errors

- Copy and pasting model content is common practice
 - Results in subsystem and model clones

- Risks of cloned subsystems
 - Bugs may propagate across design
 - Difficult to find all occurrences to fix
 - Code size may be increased

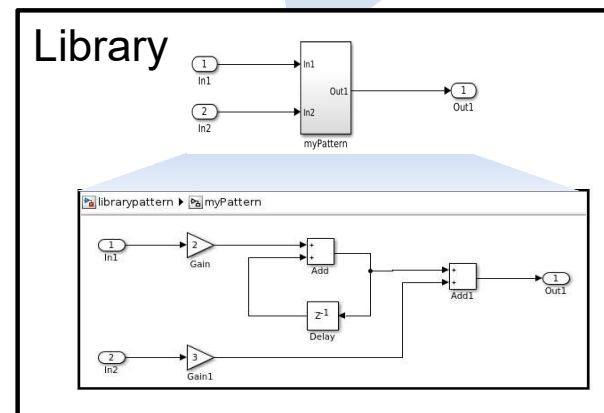
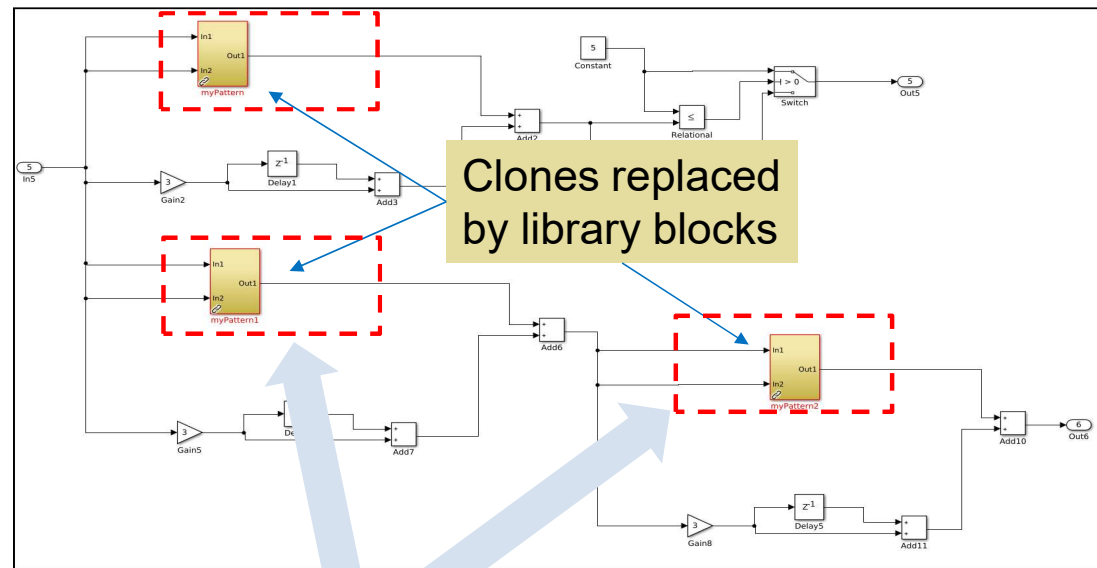


**Copied
blocks**

Clone Detection and Refactor

Identify modeling patterns and refactor to simplify model

- Find duplicate model content in your design for reuse
- Replace exact clones with library blocks to improve reuse
- Find patterns stored in a library and link to library



Library pattern

Summary of experimental results with customer models

118

- Customer projects tested

80%

- Projects that contained at least one subsystem clone group

140

- Max number of clone groups in a given model

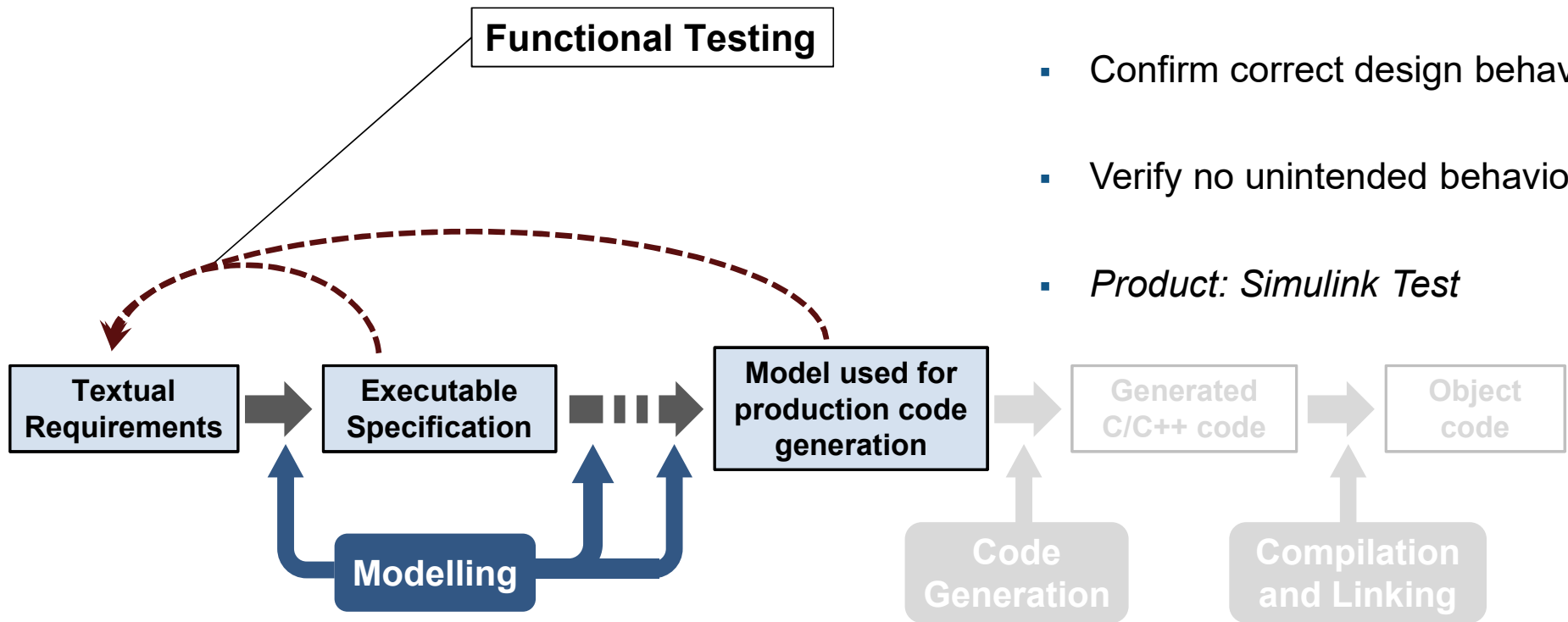
~2000

- Largest number blocks in a single clone group

-54%

- Largest reduction in generated lines of code in one model

Verification and Validation Tasks and Activities

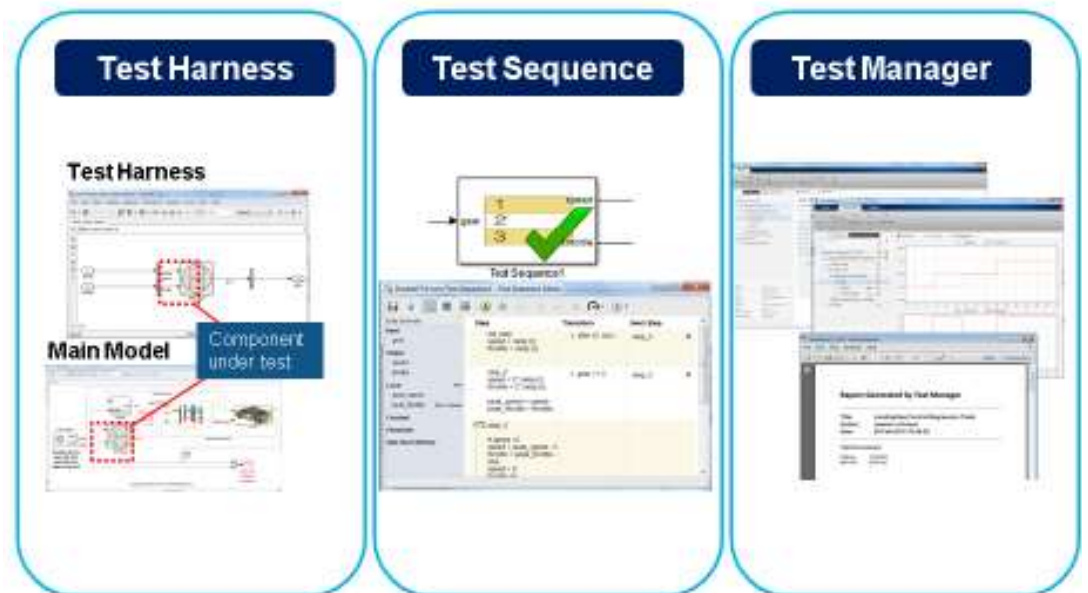


- Does design meet requirements?
- Confirm correct design behavior
- Verify no unintended behavior
- *Product: Simulink Test*

Functional Testing Process

- Author test-cases that are derived from requirements
 - Use test harness to isolate component under test
 - Test Sequence to create complex test scenarios

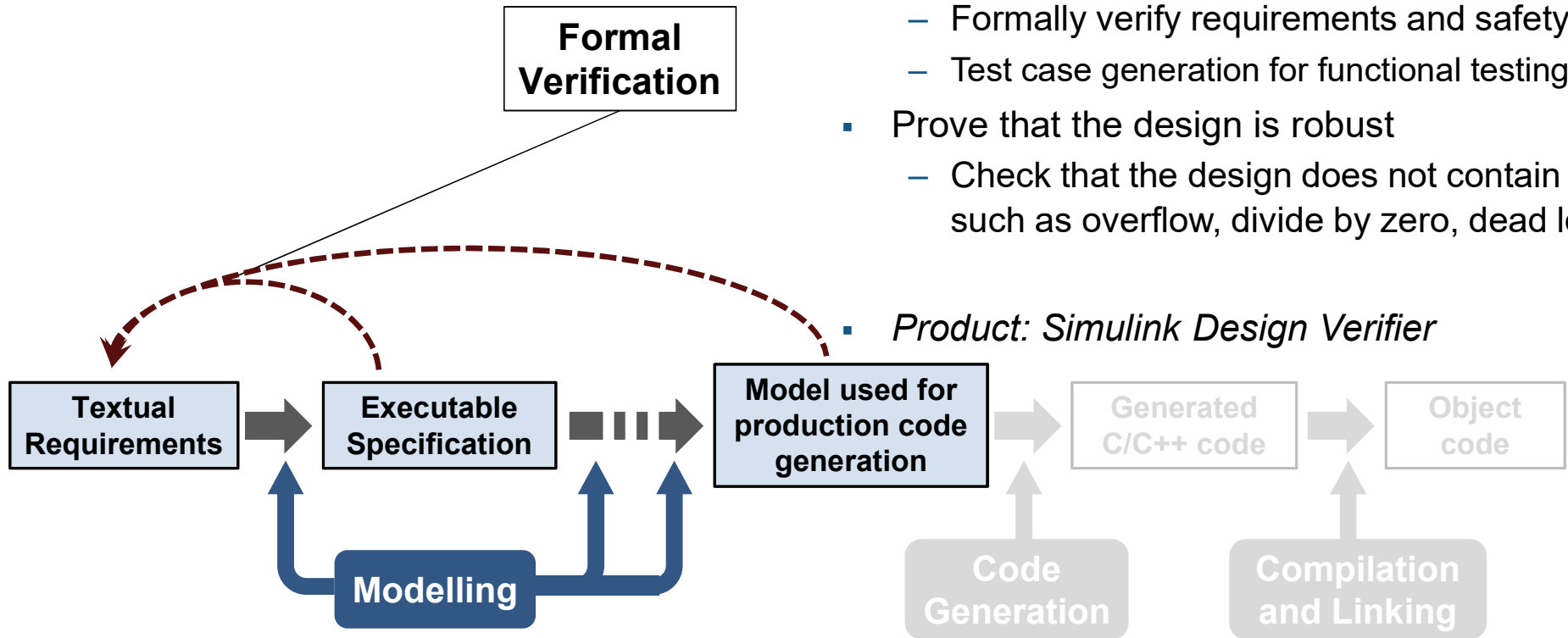
- Manage tests, execution, results
 - Re-use tests for regression
 - Automate in Continuous Integration systems such as Jenkins



Verification and Validation Tasks and Activities

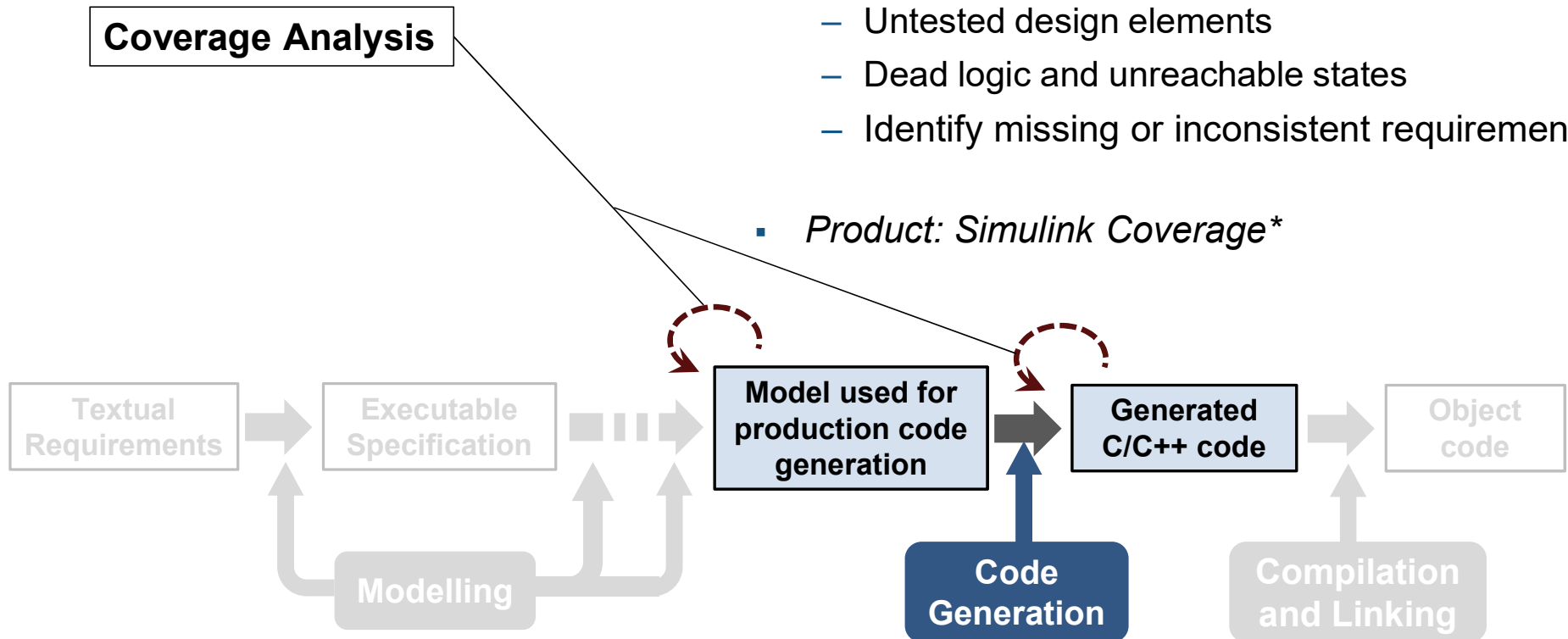
- Prove design meets requirements
 - Formally verify requirements and safety
 - Test case generation for functional testing
- Prove that the design is robust
 - Check that the design does not contain errors such as overflow, divide by zero, dead logic, ...

▪ *Product: Simulink Design Verifier*



Verification Task

- How much of software has been tested?
- Identify testing gaps to find
 - Untested design elements
 - Dead logic and unreachable states
 - Identify missing or inconsistent requirements
- *Product: Simulink Coverage**



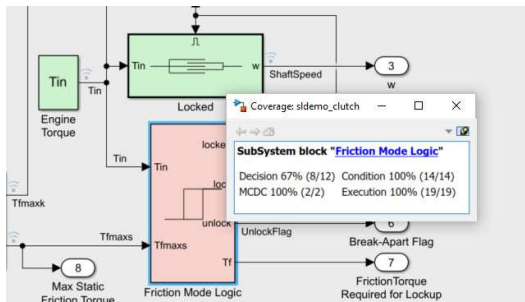
* Customers with Simulink V&V licenses will automatically receive this product

Simulink Coverage

Measure test coverage in models and generated code

Model Coverage

- **Measure** test completeness
- **Identify** missing tests or unintended functionality



Generated Code Coverage

- **Find** untested generated code
- **Map** results from code to model object

Code Coverage Report for slvndemo_counter

```

46 /* Import: 'sRoot/sLower'
47 */
48 rtb_inputdLower = (rtb_input >= slvndemo_counter_U.lower);
49
50 /* SWITCH: 'sRoot/switch' incorporates:
51 * Import: 'sRoot/wupper'

```

Decisions analyzed:

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

Conditions analyzed:

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

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

decision outcomes:	True	False
	51	0

Highlighting and Reporting

- **View** coverage results on diagrams
- **Manage** accumulated coverage results

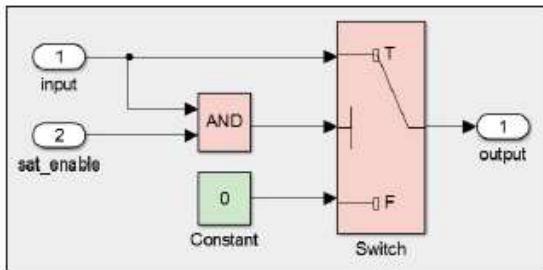
Summary

Model Hierarchy Complexity	Test 1 Decision	Conditions	MCDC	TBL	Execution
1. s_Lin	32 79%	75%	50%	27%	100%
2. Engine	NA	NA	NA	11%	100%
3. Vehicle	NA	NA	NA	NA	100%
4. s_slid_logic	26 78%	75%	50%	17%	100%
5. s_slid_logic	25 78%	75%	50%	17%	100%
6. s_slid_logic	9 69%	NA	NA	NA	NA
7. s_slid_logic	16 86%	75%	50%	17%	100%
8. s_slid_logic	0 0%	NA	NA	17%	100%
9. s_slid_logic	6 83%	NA	NA	17%	100%
10. s_slid_logic	5 80%	NA	NA	NA	NA
11. s_slid_logic	5 80%	NA	NA	91%	100%
12. Torque Converter	NA	NA	NA	91%	100%
13. transmission_ump	5 80%	NA	NA	NA	100%
14. Lock-Up Table	5 80%	NA	NA	NA	NA

Model Elements That Receive Coverage

R2017b

Simulink models

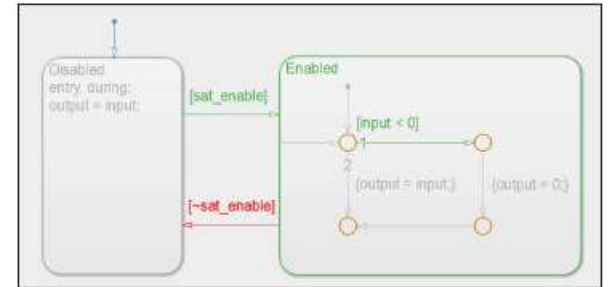


MATLAB function blocks

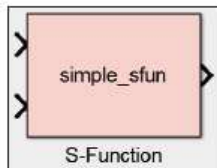
```

1 function output = myFcn(input, sat_enable)
2 %#codegen
3
4 if (input<0 && sat_enable)
5     output = 0;
6 else
7     output = input;
8 end
    
```

Stateflow charts



C/C++ code S-Functions



Decisions analyzed:

<code>!(w[0]<0) && w[1]</code>	50%
false	101/101
true	0/101

Conditions analyzed:

Description:	True	False
<code>w[0]<0</code>	50	51
<code>w[1]</code>	0	50

Generated code

```

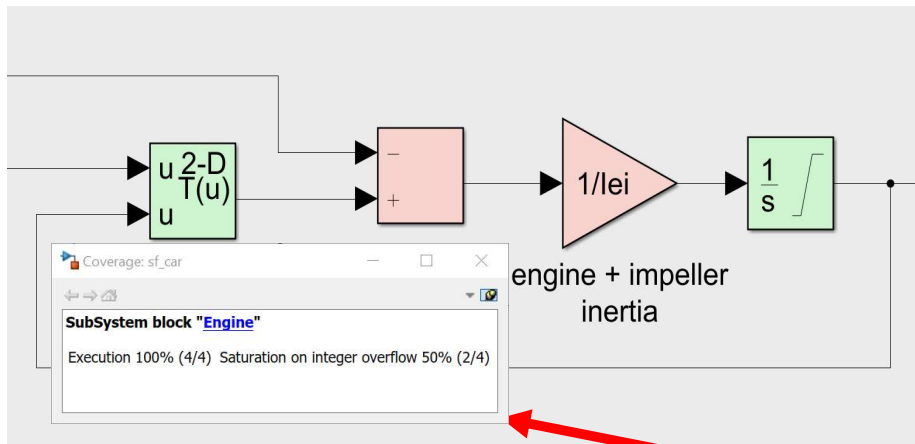
47 */
48 rtb_inputGElower = (rtb_input >= slvndemo_counter_U.lower);
49
50 /* Switch: '<Root>/Switch' incorporates:
51  * Inport: '<Root>/upper'
52  * Logic: '<Root>/And'
53  * RelationalOperator: '<Root>/upper_GE_input'
54  */
55 if (!(slvndemo_counter_U.upper >= rtb_input) && rtb_inputGElower) {
    
```

Decisions analyzed:

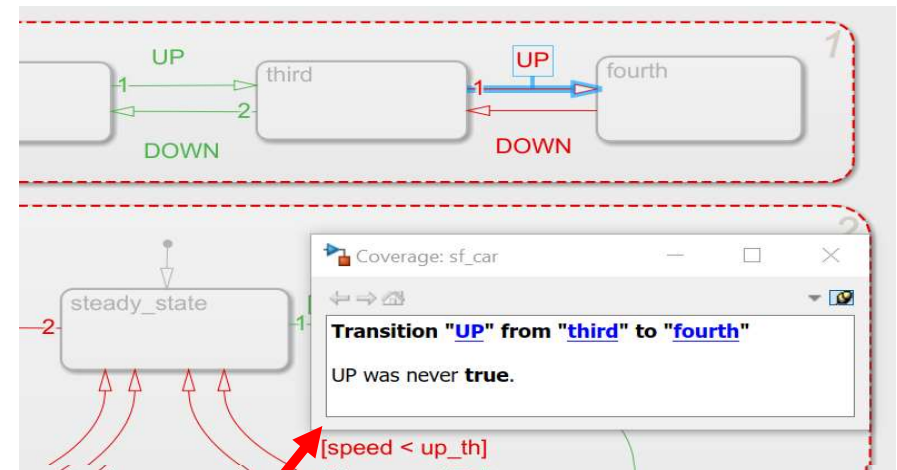
<code>!(slvndemo_counter_U.upper >= rtb_input) && rtb_inputGElower</code>	50%
false	51/51
true	0/51

Highlight And Explore Coverage Results On Model

Simulink



Stateflow



Highlighting shows you missing coverage

- **Green** is full coverage
- **Red** shows missing coverage

Coverage Display Window

- Quickly see detailed coverage results for object

Coverage Reports For Analysis, Reviews And Documentation

Summary

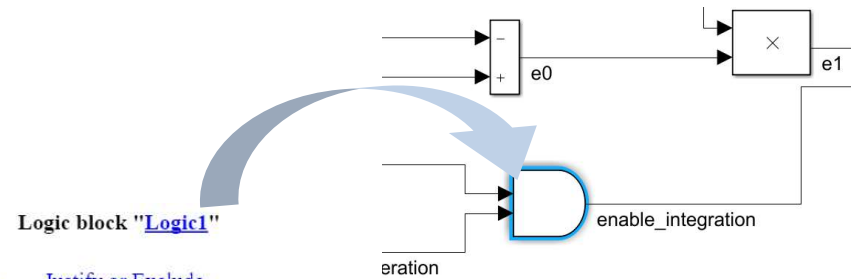
Model Hierarchy/Complexity	Test 1	Decision	Condition	MCDC	Execution	Relational Boundary	Saturation on integer
1. sldemo_fuelsys	80	34%	34%	NA			
2. Engine Gas Dynamics	13	71%	NA				
3. Mixing & Combustion	3	67%	NA				
4. EGO Sensor	2	100%	NA				
5. System Lag	NA		NA				
6. Throttle & Manifold	10	73%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

Details

1. Model "sldemo_fuelsys"

Child Systems: [Engine Gas Dynamics](#), [Throttle Command](#), [To Controller](#), [T](#)

Metric	Coverage (this object)	Coverage (inc. descen)
Cyclomatic Complexity	1	80
Condition	NA	34% (11/32) condition
Decision	NA	34% (41/122) decision
MCDC	NA	7% (1/14) conditions
Lookup Table	NA	1% (13/1511)interpol
Execution	NA	90% (64/71) objective
Relational Boundary	NA	10% (5/50) objective
Saturation on integer overflow	NA	50% (10/20) objective



Logic block "[Logic1](#)"

[Justify or Exclude](#)

Parent: [sldemo_fuelsys/fuel_rate_control/airflow_calc](#)

Uncovered Links:

Metric	Coverage
Cyclomatic Complexity	0
Condition	75% (3/4) condition outcomes
MCDC	50% (1/2) conditions reversed the outcome
Execution	100% (1/1) objective outcomes

Conditions analyzed

Description	True	False
input port 1	199521	480
input port 2	200001	0

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

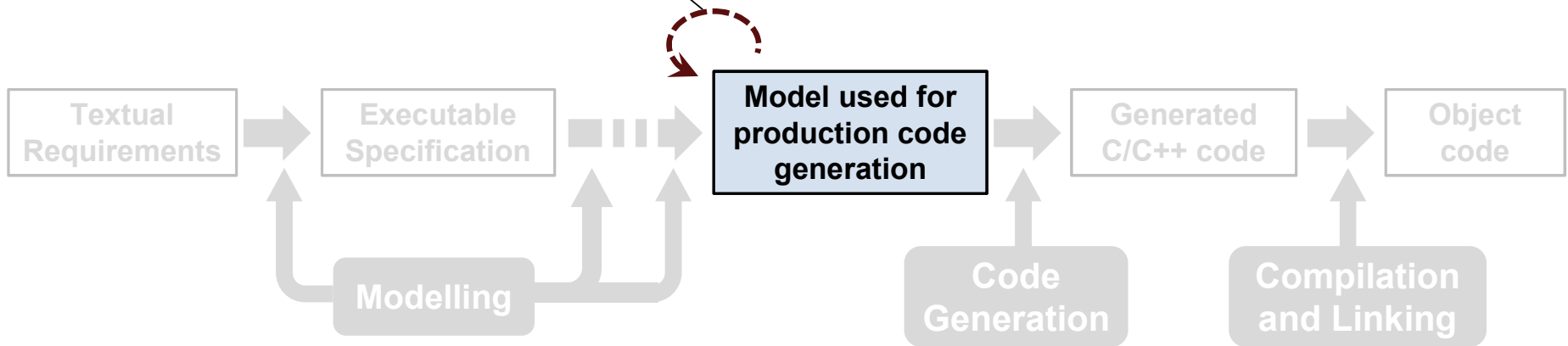
Decision/Condition	True Out	False Out
expression for output		
input port 1	TT	FT
input port 2	TT	(TF)

- Summary section provides quick overview of results
- Detailed metrics are reported for each object including Decision, Condition, MCDC and more
- Easily navigate to model or code from report

Verification and Validation Tasks and Activities

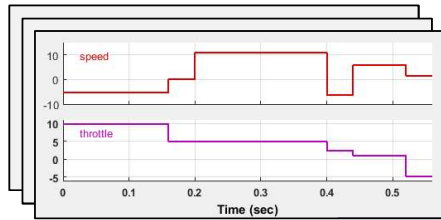
Test Generation for Coverage

- Automate manual task of writing test-cases and test inputs
 - Intelligent determination of input combinations for high coverage
- Formal methods based test generation
 - Analyze design, states, logic paths in the design model
- *Product: Simulink Design Verifier*

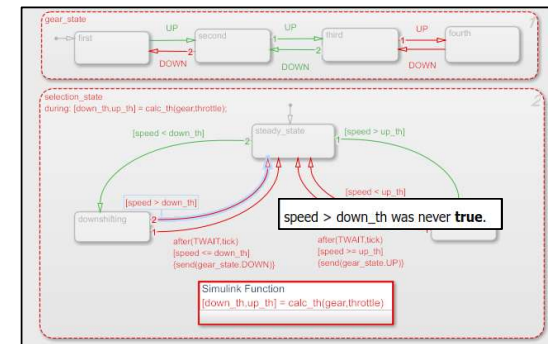
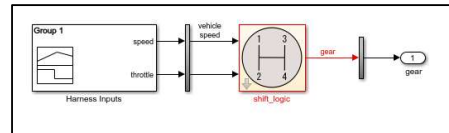


Addressing Missing Coverage

Partial Coverage

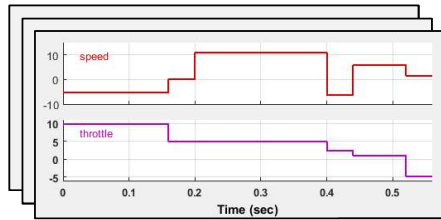


Test Cases

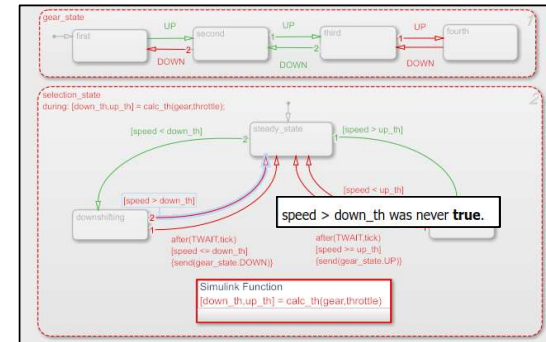
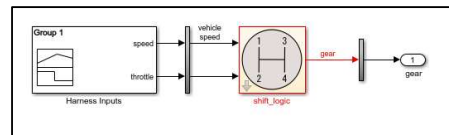


Addressing Missing Coverage

Partial Coverage



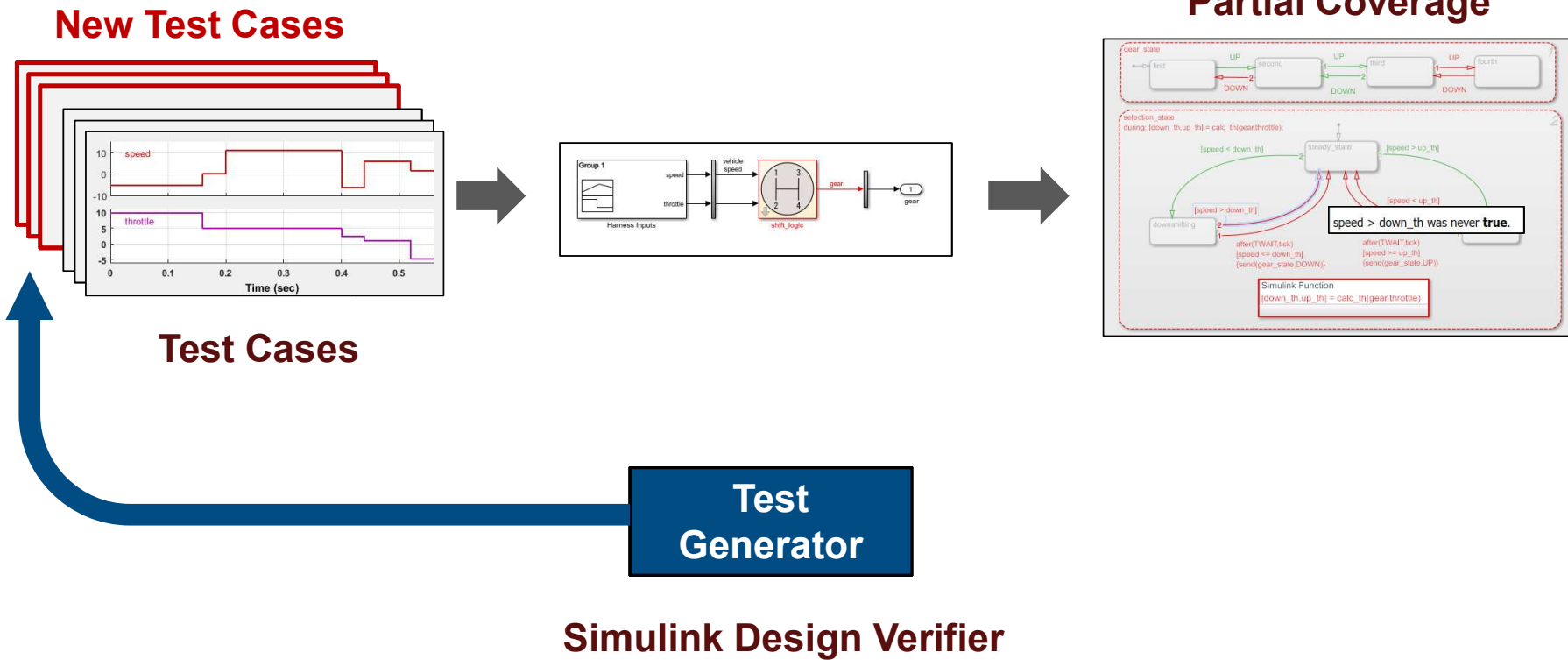
Test Cases



Test Generator

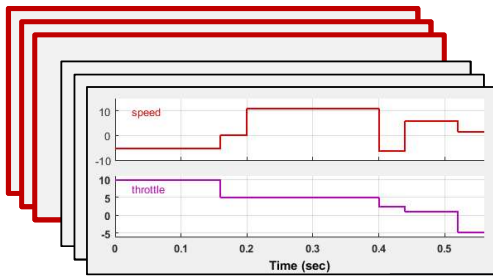
Simulink Design Verifier

Addressing Missing Coverage

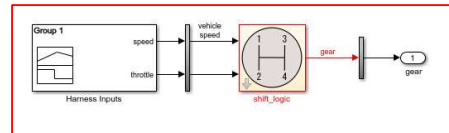


Addressing Missing Coverage

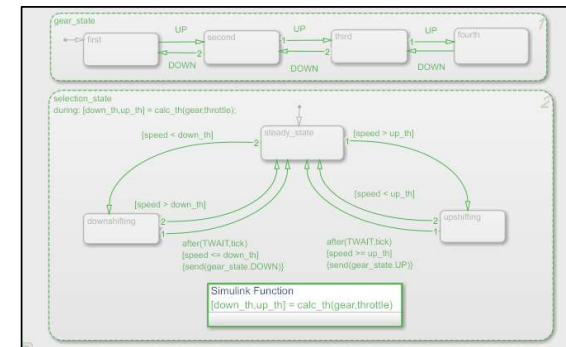
New Test Cases



Test Cases








Full Coverage




Automatically Address Missing Coverage

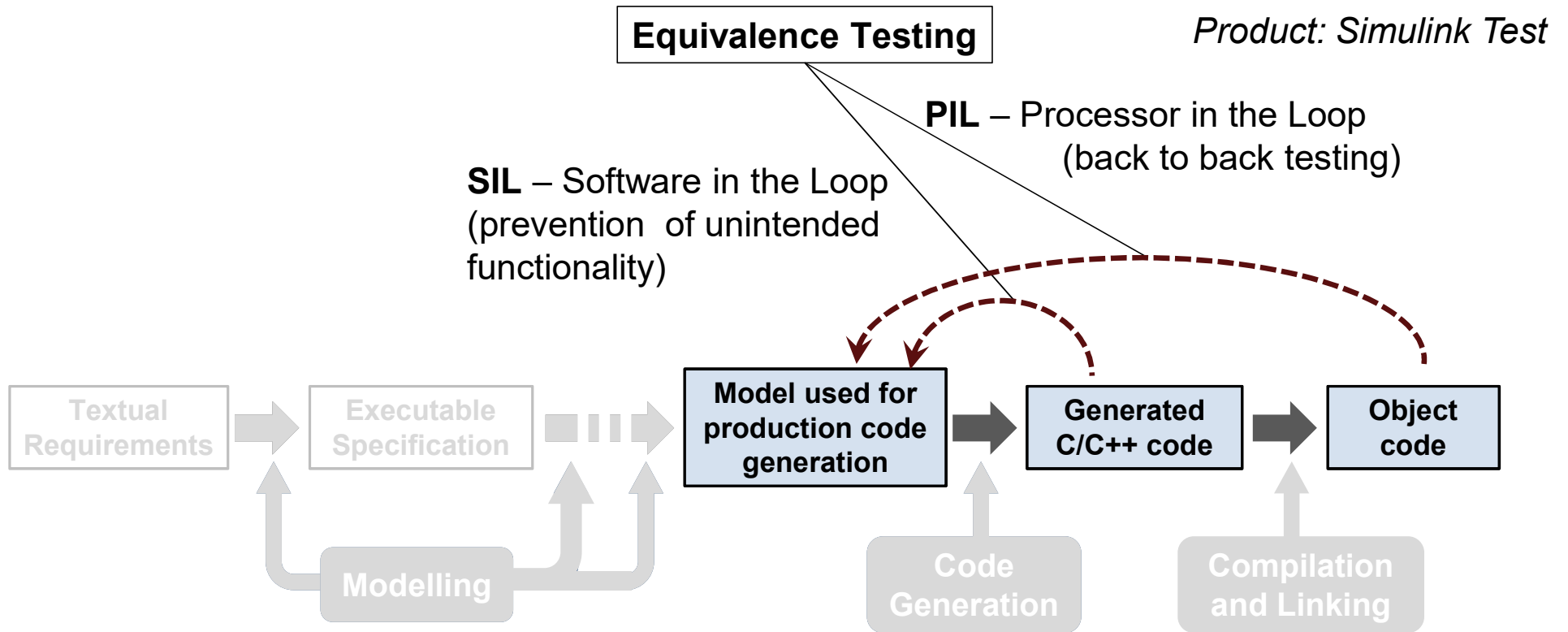
Generate additional tests automatically using Simulink Design Verifier from within the Test Manager to increase coverage

▼ AGGREGATED COVERAGE RESULTS ?

ANALYZED MODEL	REPORT CO...	DECISION	CONDITION	MCDC
 simulinkCruiseAddReqExample	 31	50% 	41% 	25% 

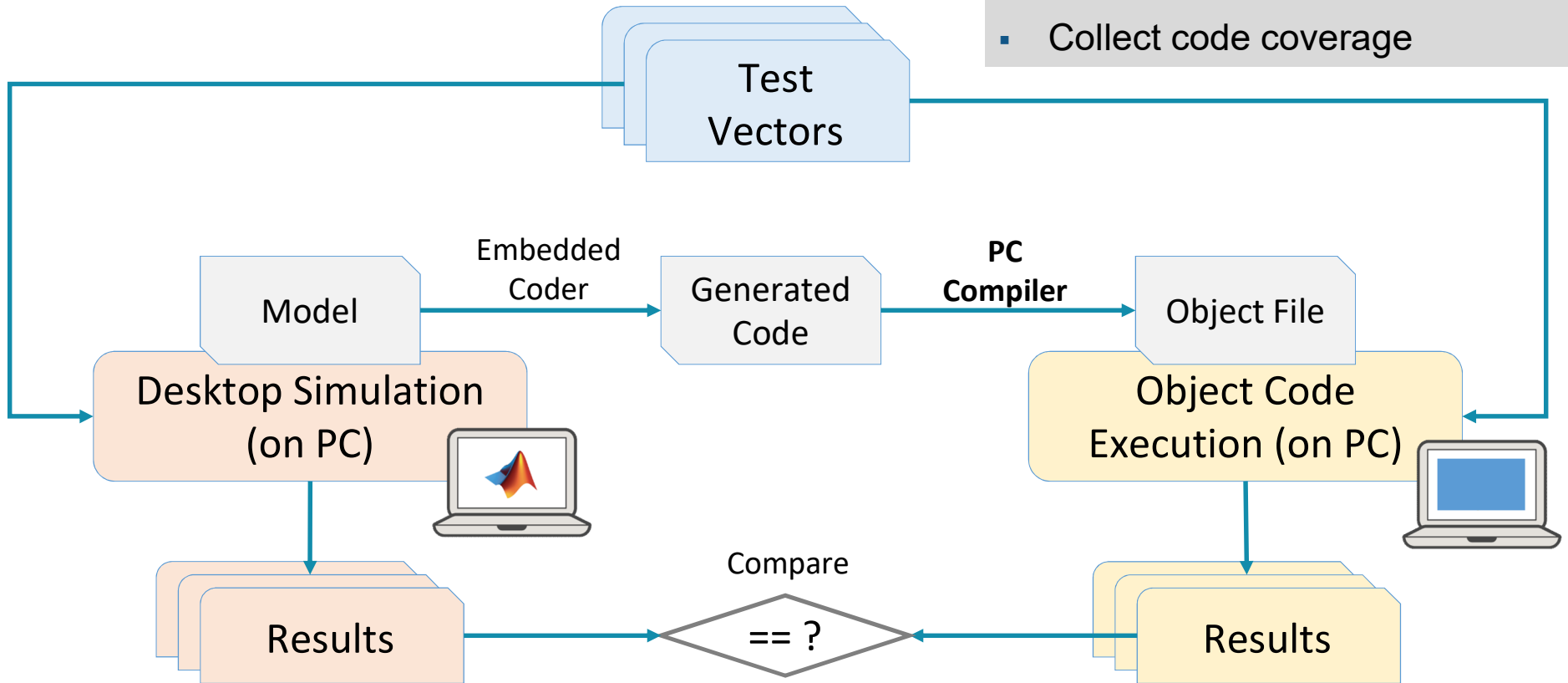
+ Add Tests for Missing Coverage  Export

Verification and Validation Tasks and Activities



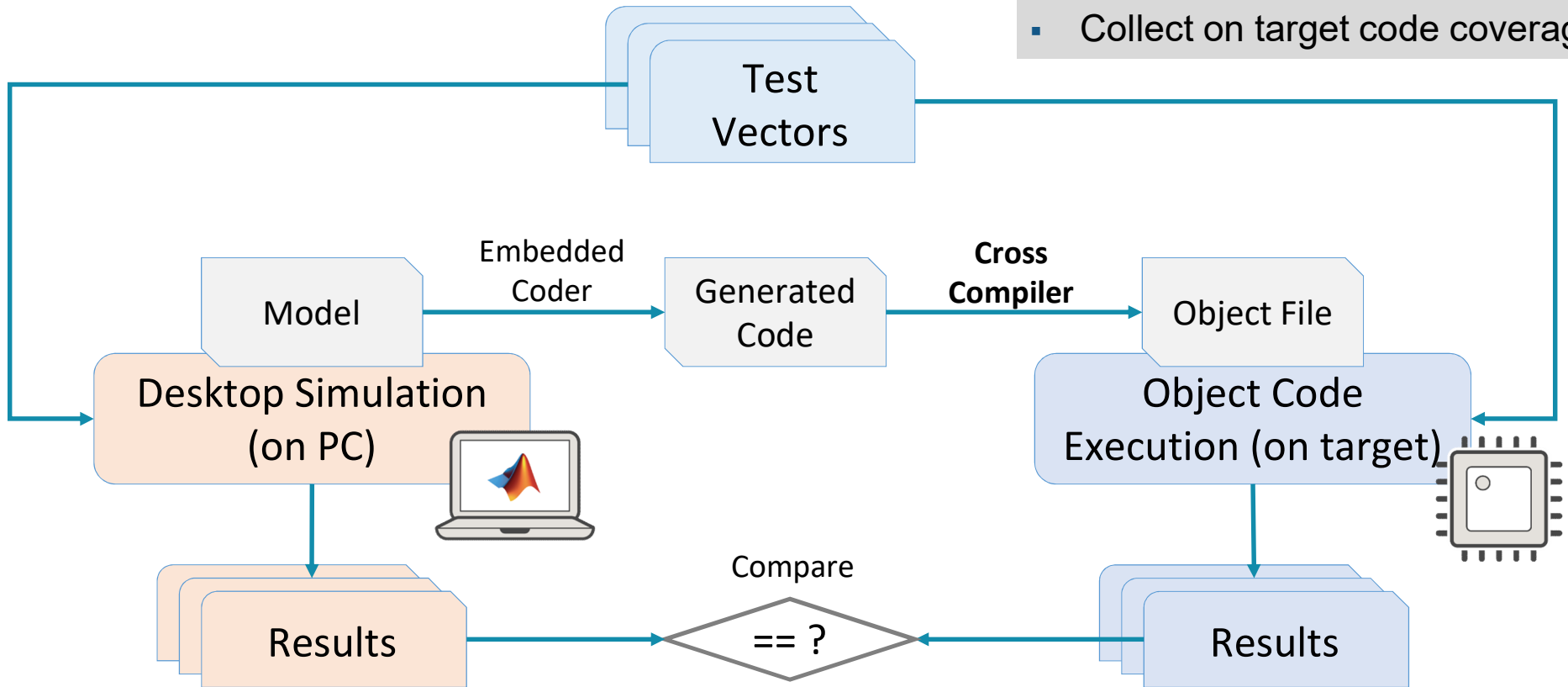
Software In the Loop (SIL) Testing

- Show equivalence, model to code
- Assess code execution time
- Collect code coverage



Processor In the Loop (PIL) Testing

- Verify numerical equivalence
- Assess target execution time
- Collect on target code coverage



MathWorks V&V Solution Summary

Requirements

Standards Compliance

Testing

Formal Verification

Coverage Analysis

Static Code Analysis

SIL, PIL

MathWorks V&V Product Capabilities

Requirements	Simulink Requirements* (<i>New in R2017b</i>)
Standards Compliance	Simulink Check* (<i>New in R2017b</i>)
Testing	Simulink Test
Formal Verification	Simulink Design Verifier
Coverage Analysis	Simulink Coverage* (<i>New in R2017b</i>)
Static Code Analysis	Polyspace Bug Finder, Polyspace Code Prover
SIL, PIL	Simulink Test

* Customers with Simulink V&V licenses will automatically receive these new products

Qualify Tools using IEC Certification Kit for ISO 26262, IEC 61508, and related standards



- Qualify tools, including
 - Embedded Coder
 - Simulink Check
 - Simulink Coverage
 - Simulink Design Verifier
 - Simulink Test
 - Polyspace Bug Finder
 - Polyspace Code Prover
- Support standards, including
 - ISO 26262 (Automotive)
 - IEC 61508 (Industrial)
 - EN 50128 (Rail)
 - IEC 62304 (Medical)

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.

Customer References and Applications



Korean Air Speeds UAV Flight Control Software Development and Verification with Model-Based Design

Korean Air designed and simulated flight control laws and operational logic, generated and verified production code, and conducted HIL tests.



Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks

Continental AG used MathWorks software to design a level and roll control system for heavy-duty, 40-ton trucks.



Chery Enables In-House Development of Engine Management System Software with Model-Based Design

Chery decreased cost and accelerated development time by modeling, simulating, testing, and generating production code for its EMS software in-house.



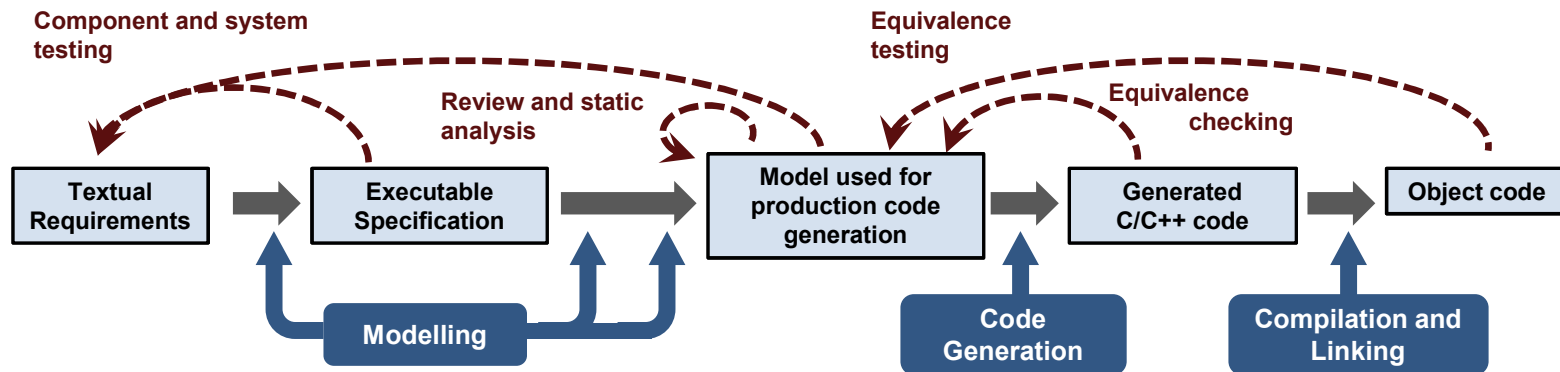
Lear Delivers Quality Body Control Electronics Faster Using Model-Based Design

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

Key Takeaways

1. Find defects earlier
2. Automate manual verification tasks
3. Reference workflow is supported by V&V tools and conforms to safety standards

After Lunch:
 SW 신뢰성 향상을 위한
 테스트 솔루션
 : **Simulink Test**
 이제훈 차장



Thank You!

고맙습니다!

Appendix

Learn More:

[MathWorks Release 2017b: Prerelease Information \(June 2017\)](#)



The screenshot shows the top navigation bar of the MathWorks website. On the left is the MathWorks logo. In the center, the page title "MathWorks Release 2017b: Prerelease Information (June 2017)" is displayed. To the right of the title is a search bar with the placeholder text "Search MathWorks.com" and a magnifying glass icon. Below the navigation bar are two menu items: "Trial software" with a downward arrow icon and "Contact sales" with a telephone handset icon.

One product has a license-related change in R2017b. To use the latest version of this product, you must have a subscription to MathWorks Software Maintenance Service as of R2017b.

Simulink Verification and Validation

As of R2017b, Simulink Verification and Validation™ will transition into three products: Simulink Check™, Simulink Coverage™, and Simulink Requirements™.

- Requirements traceability and Requirements Management Interface (RMI) functionality have moved to the Simulink Requirements product.
- Model and generated code coverage functionality, and component verification functions such as `slvnmakeharness`, have moved to the Simulink Coverage product.
- Compliance checking, model metrics, clone detection and refactoring, and model transformer functionality have moved to the Simulink Check product.

Simulink Check, Simulink Coverage, and Simulink Requirements will each require MATLAB® and Simulink®, but do not have a dependency on each other. Simulink Design Verifier™, which required Simulink Verification and Validation, requires both Simulink Check and Simulink Coverage.

Learn More: [MathWorks Release 2017b: Prerelease Information \(June 2017\)](#)

If you are:

Subscribed to Software Maintenance Service as of R2017b for Simulink Verification and Validation	License(s) will be updated to include the following products at no initial cost: Simulink Check Simulink Coverage Simulink Requirements They will appear on future Software Maintenance Service renewal invoices.
Not subscribed to Software Maintenance Service as of R2017b for Simulink Verification and Validation	License(s) will be updated to include the following products at no initial cost: Simulink Check Simulink Coverage Simulink Requirements You will need to renew your Software Maintenance Service subscription to access the updated products.