

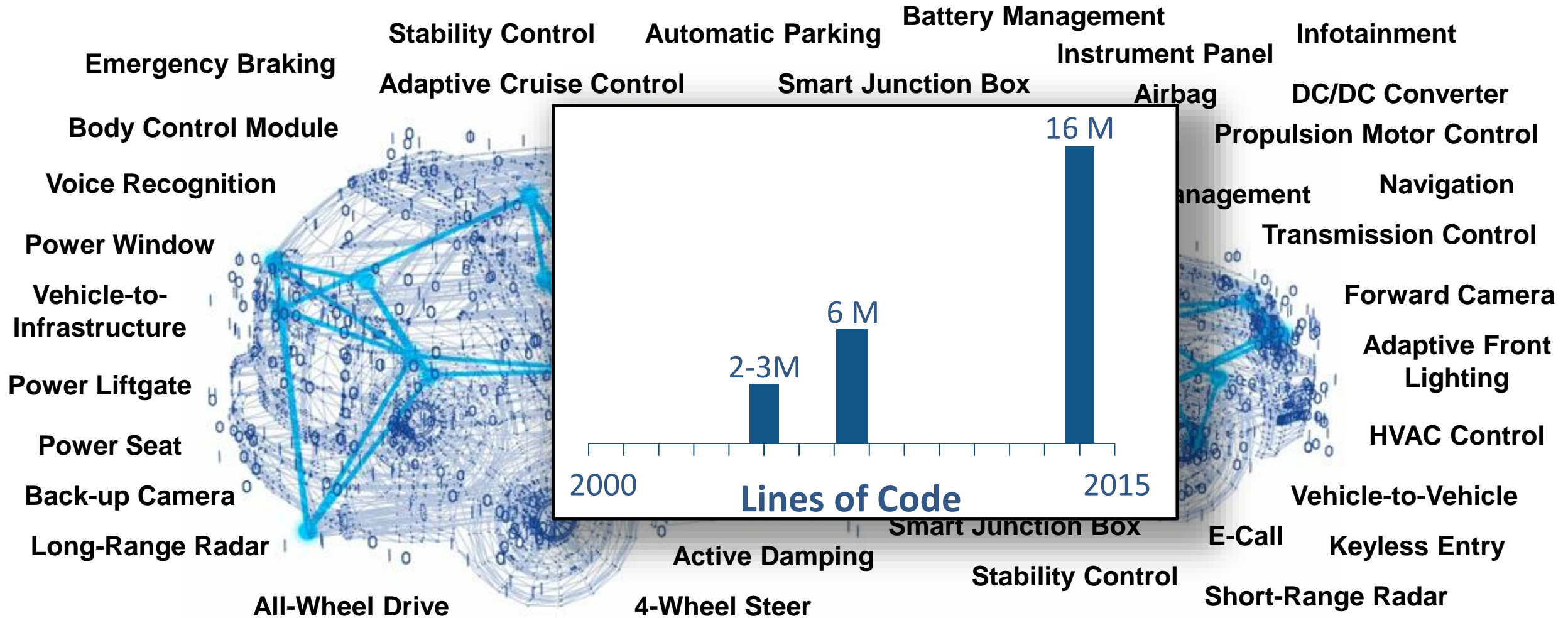
MATLAB EXPO 2018

Verification and Validation of High-Integrity Systems

Chethan CU, MathWorks
Vaishnavi HR, MathWorks

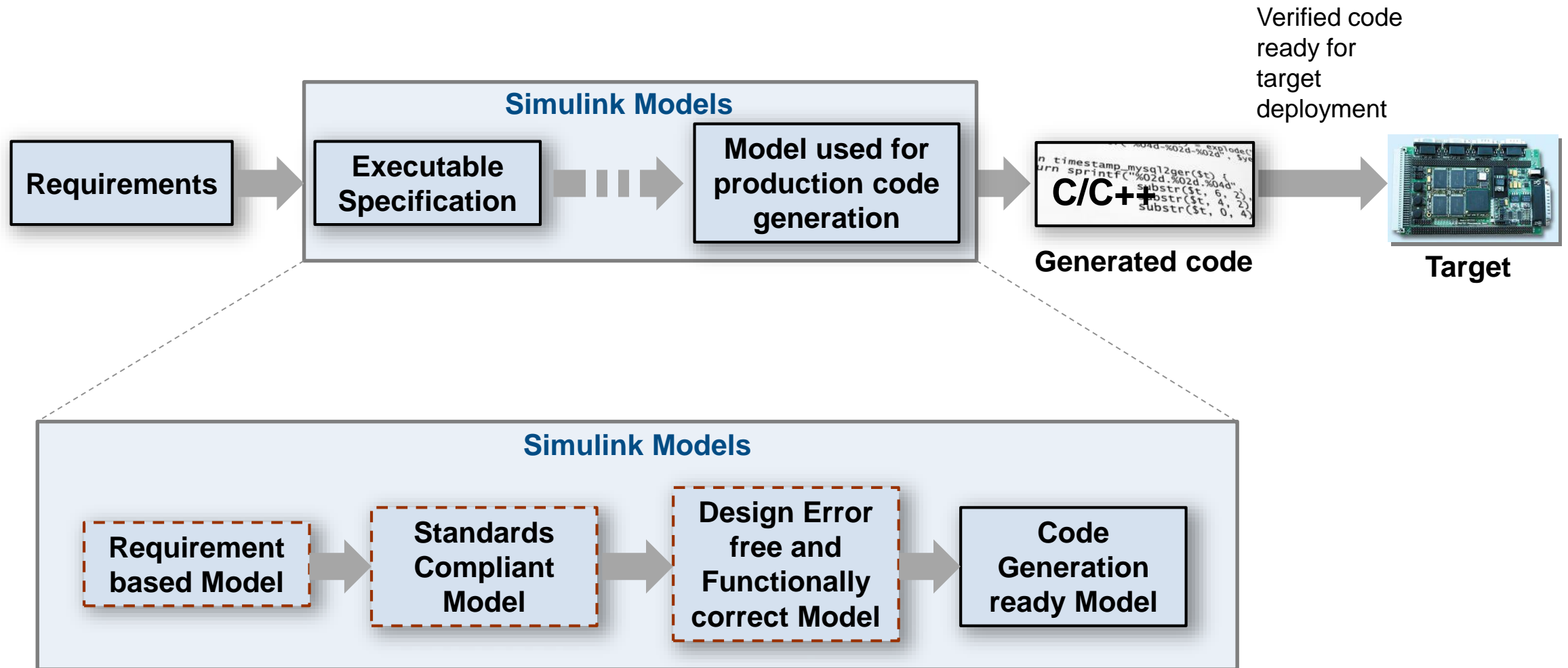


Growing Complexity of Embedded Systems



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

Model-Based Design, Verification and Validation



Key Takeaways

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

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
 90% rise time, left track
 motor speed for 95% rise time, left track
 95% 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

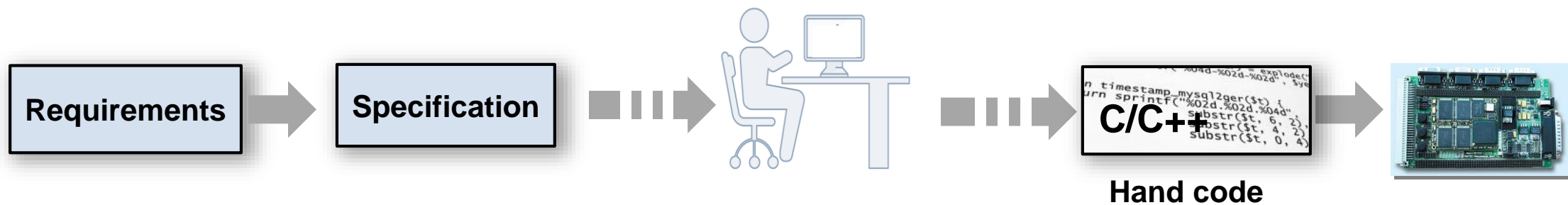
Integration Testing

Why do 71% of Embedded Projects Fail?

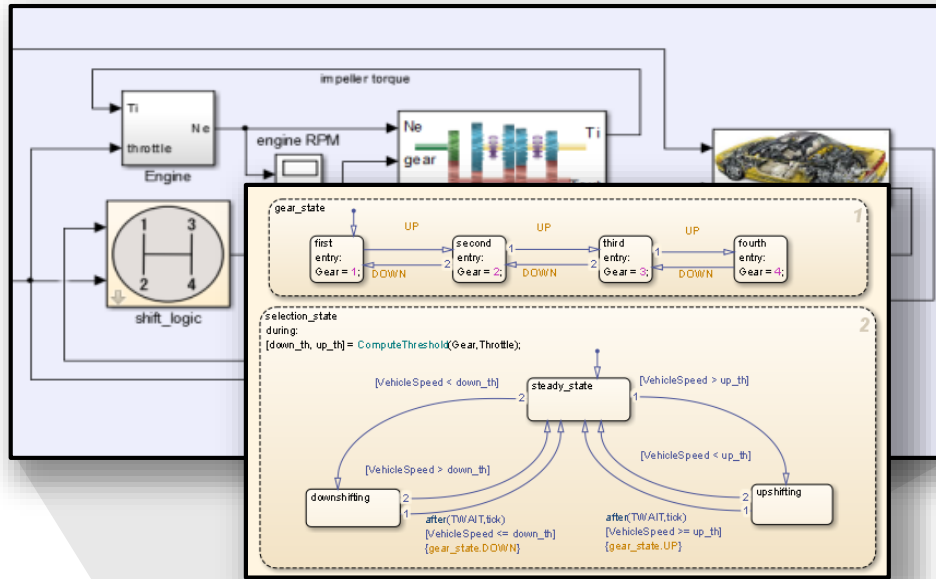
Poor Requirements Management

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

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)
  )
}

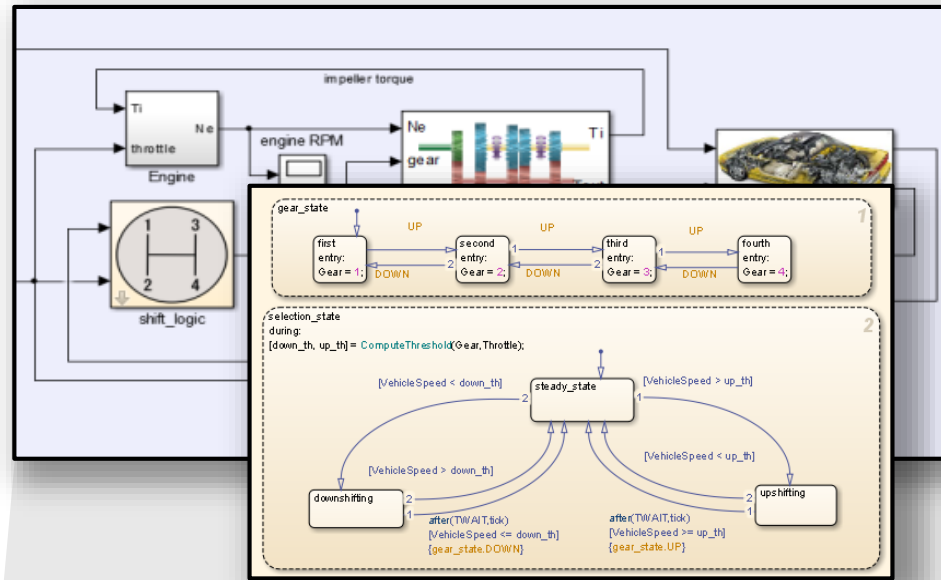
```

C/C++

Hand code



Complete Model Based Design



Code Generation



```

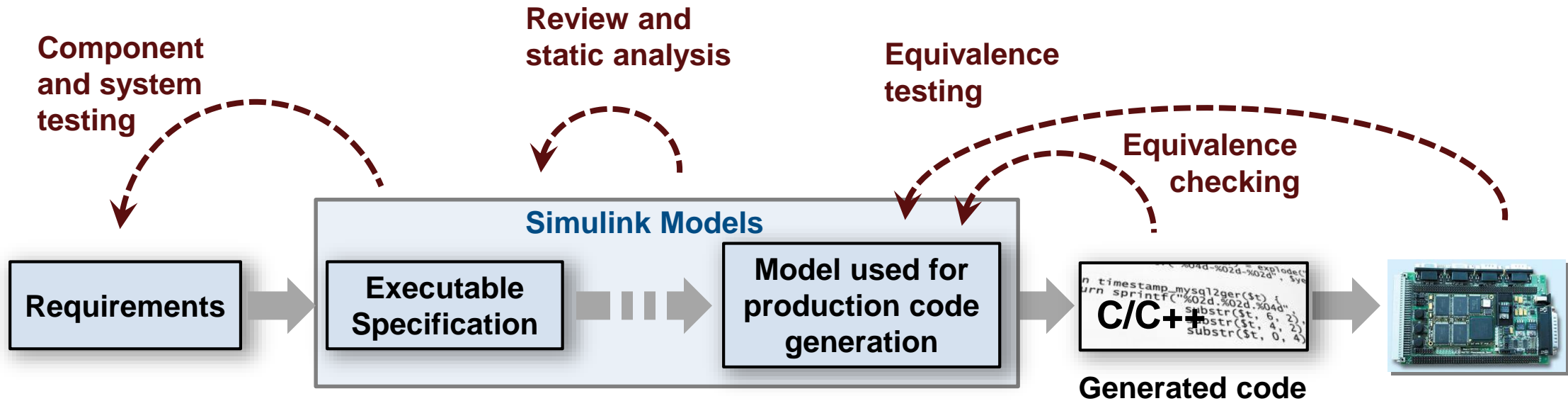
n timestamp mysql2ger($t) {
    urn sprintf("%02d.%02d.%04d",
                substr($t, 6, 2),
                substr($t, 4, 2),
                substr($t, 0, 4)
    )
}
    
```

C/C++

Generated code



Model Based Design Verification Workflow

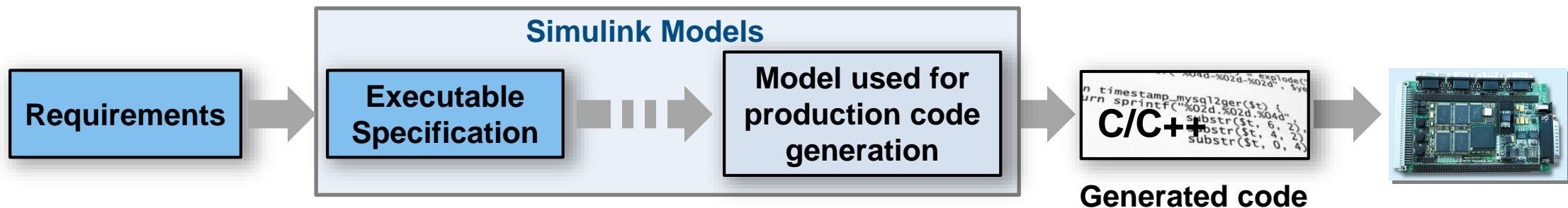


Challenges with Requirements

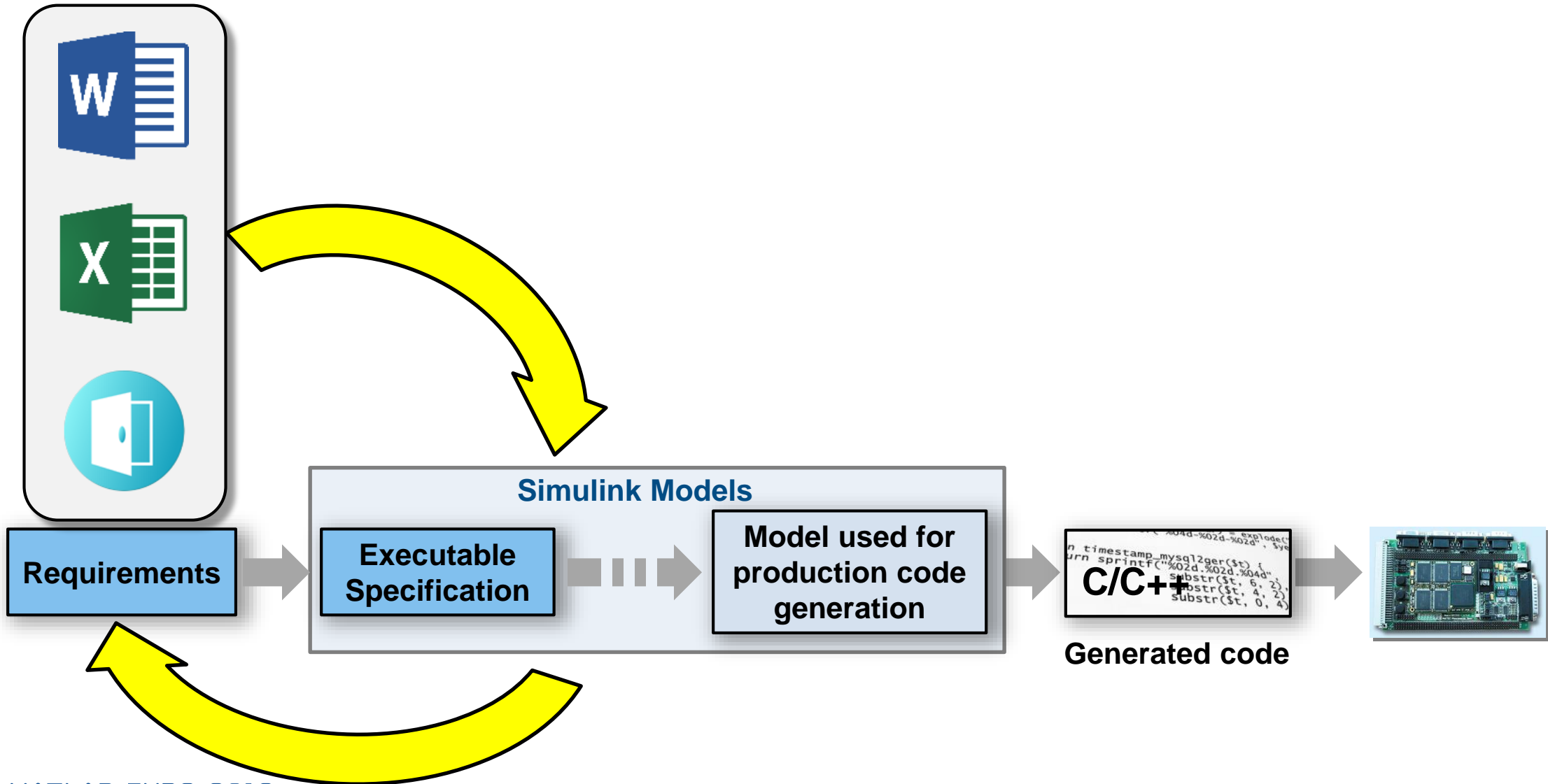
Where are requirements implemented?

Is design and requirements consistent?

How are they tested?



Gap Between Requirements and Design



Simulink Requirements

R2017b

Author


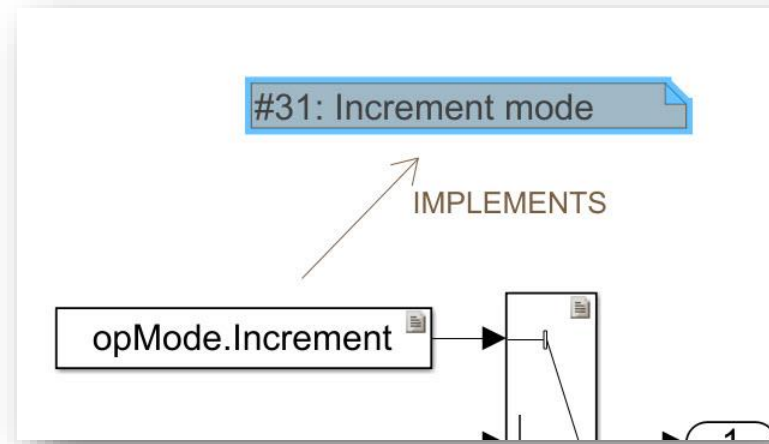
Summary: Cancel Switch Detection

Description Rationale

2 14 B I U [bullet] [list] [list] [list] [list] ... >>


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

Dashboard image

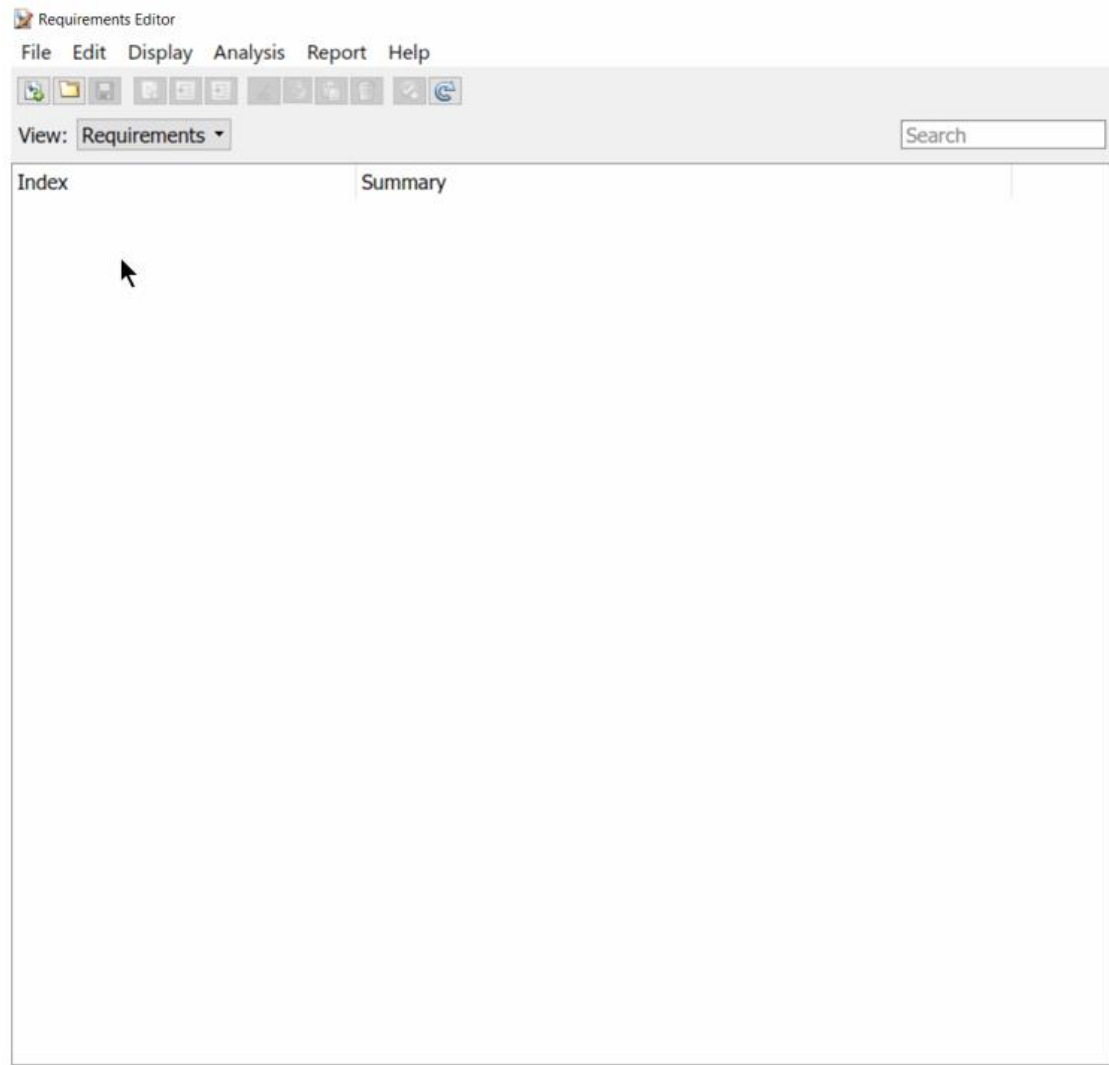
Track

Manage

 Issue: Destination Changed.


Stored:	Revision: 15
Actual:	Revision: 18


Requirements Editor



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

- 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 pane: A large empty area with a mouse cursor.
- Summary pane: A smaller pane to the right of the Index pane.

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**.

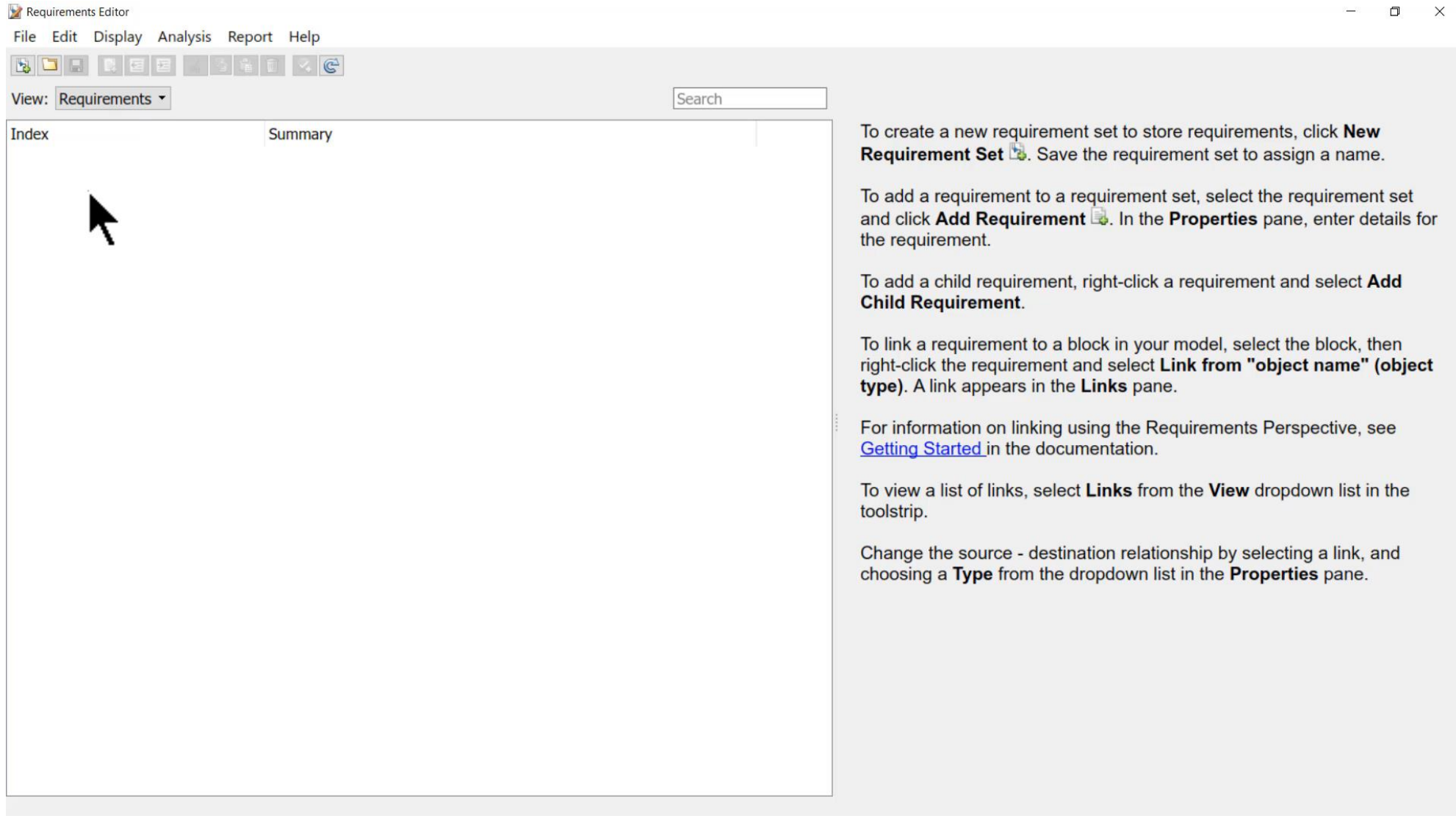
To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

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.

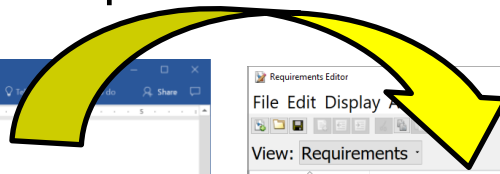
For information on linking using the Requirements Perspective, see [Getting Started](#) in the documentation.

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

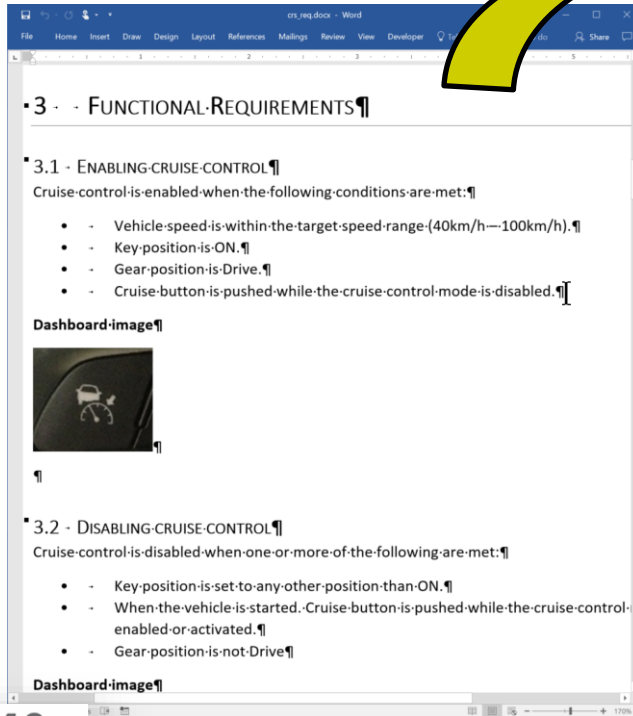
Import



IBM Rational DOORS

ReqIF
Requirements Interchange Format

Microsoft Word



Simulink Requirements Editor

Requirements Editor

View: Requirements

Index	ID	Summary
crs_req		
1	crs_req	References to crs_req.docx
1.1	1 Overview	Overview This document describes a 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 Accelerat
1.3.10	3.10 Resuming cruise control	Resuming cruise control Cruise control
1.3.11	3.11 Throttle value calculation	Throttle value calculation The cruise 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:

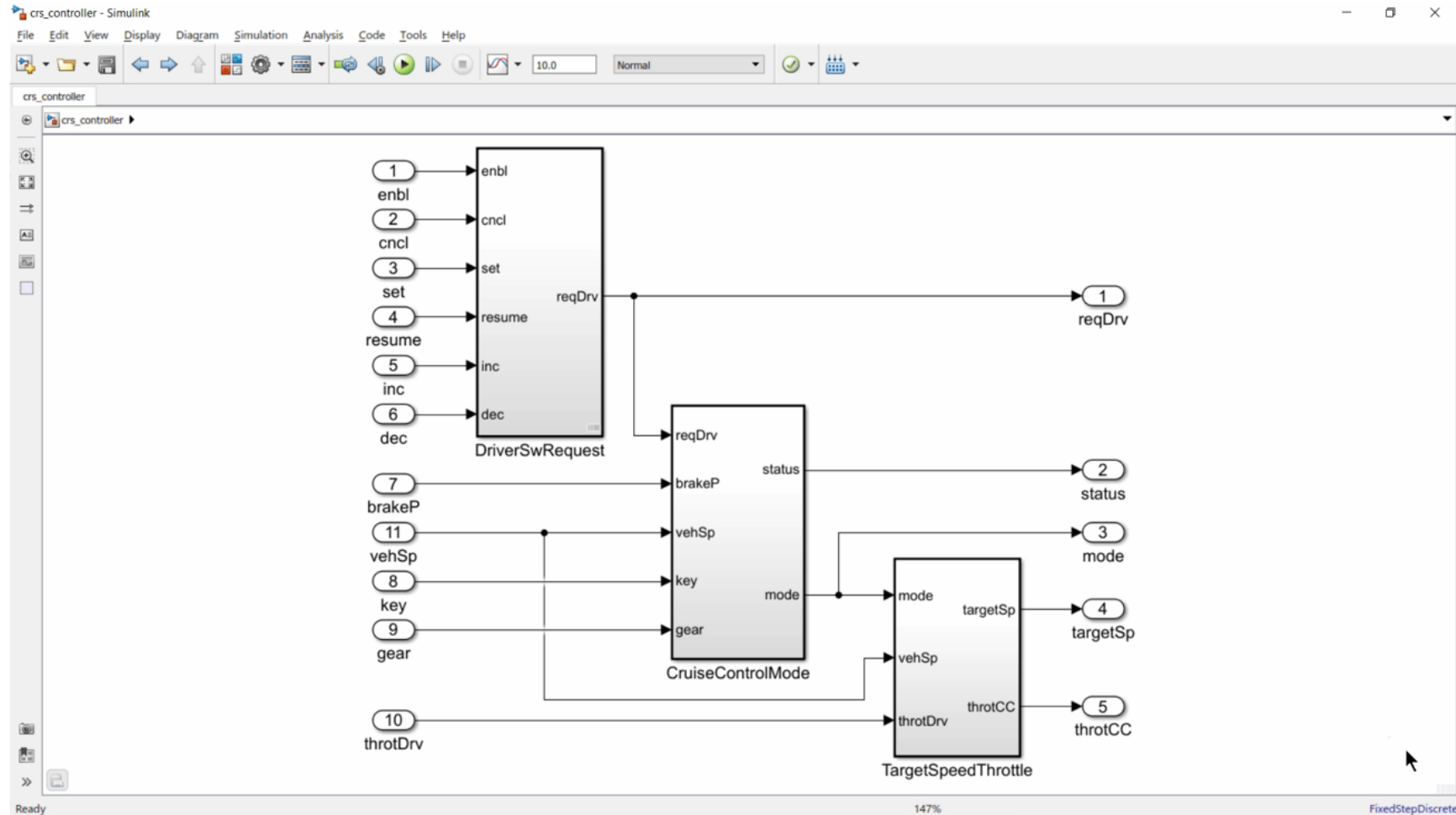
Revision information:

Show in document

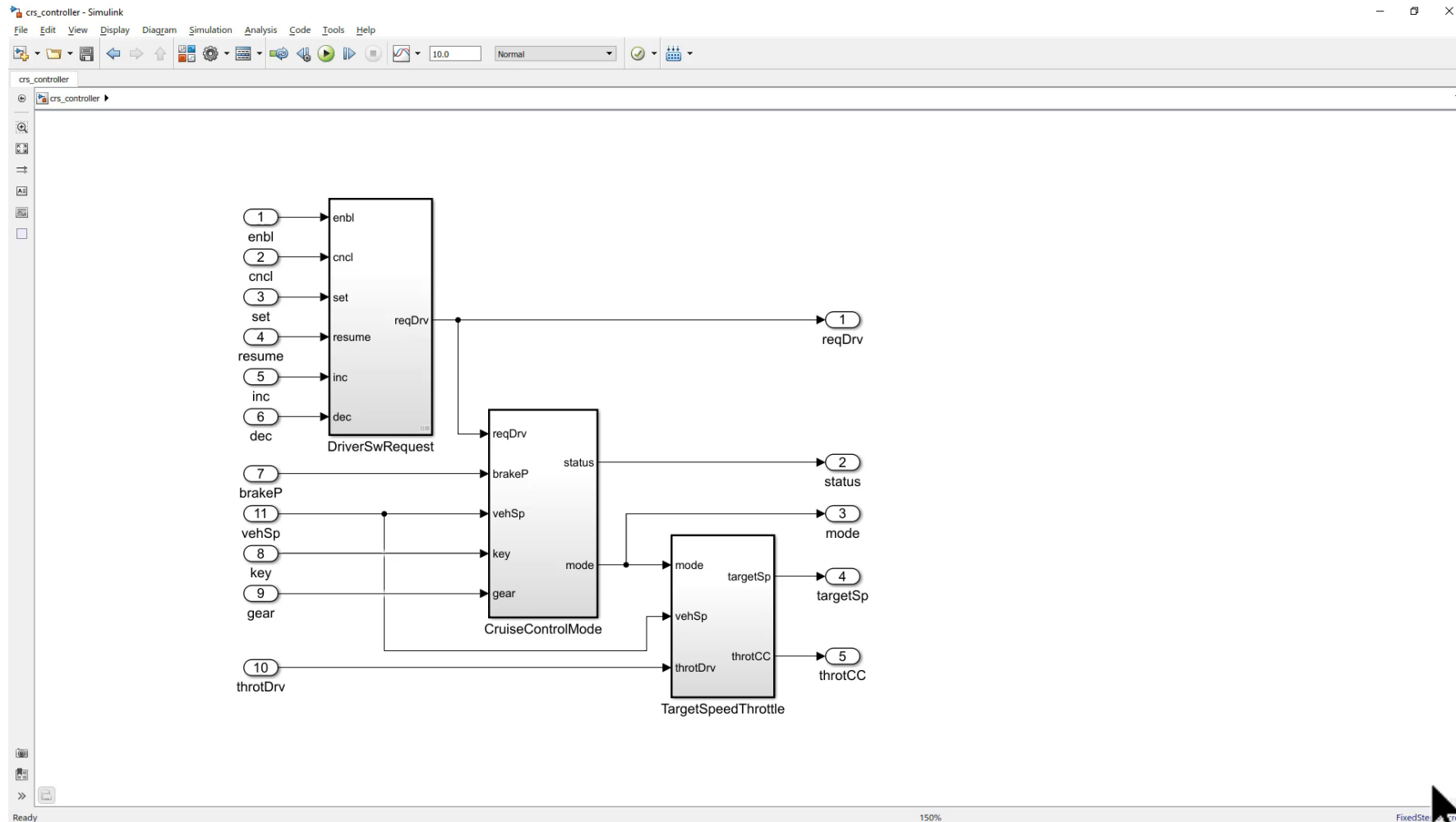
Links

Show in document

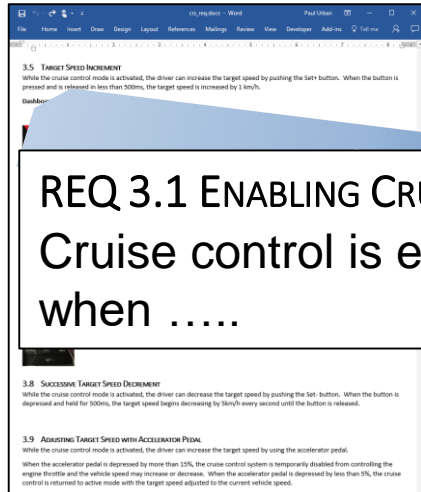
Requirements Perspective



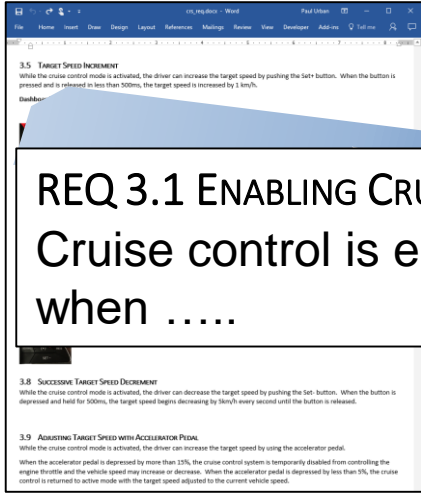
Requirements Perspective



Link Requirements, Designs and Tests



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 INCREMENT
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.

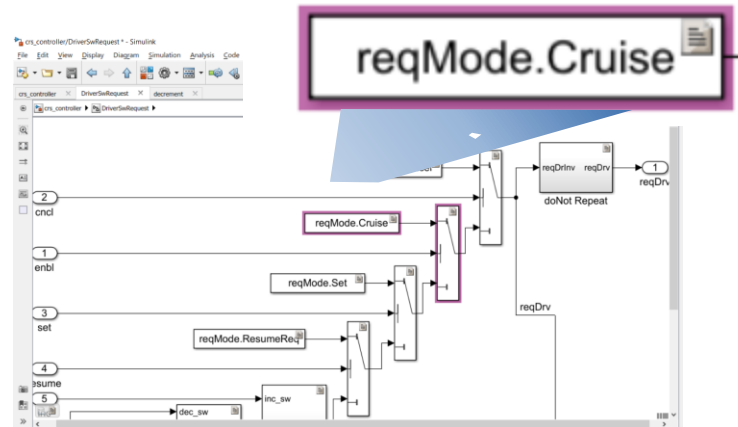
3.9 ADJUSTING TARGET SPEED WITH ACCELERATION PEDAL
While the cruise control mode is activated, the driver can increase the target speed by using the accelerator pedal. When the accelerator pedal is depressed by more than 15%, the cruise control system is temporarily disabled from controlling the engine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

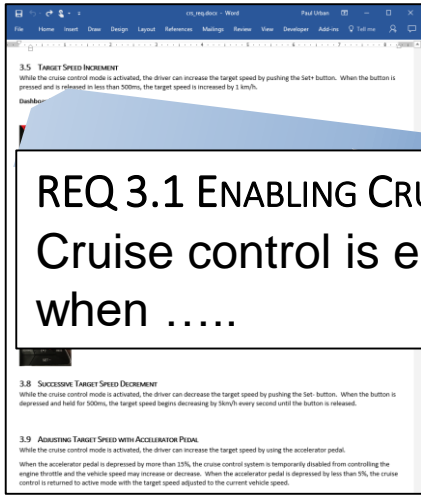
Derives

ENABLE SWITCH DETECTION
If the Enable switch is pressed

Implemented
By



Link Requirements, Designs and Tests



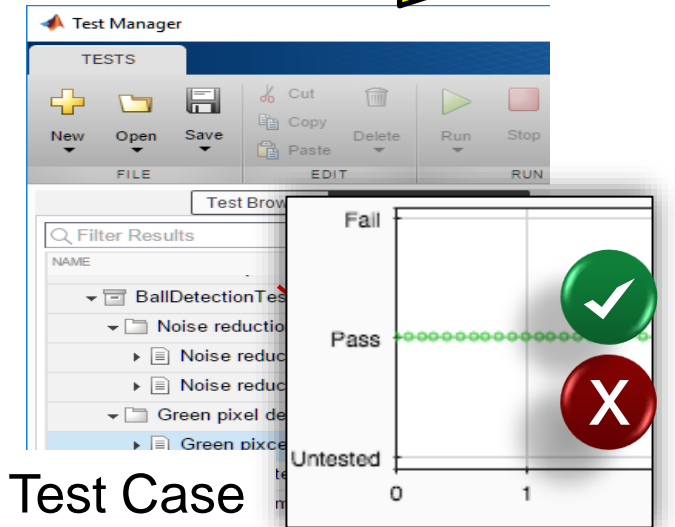
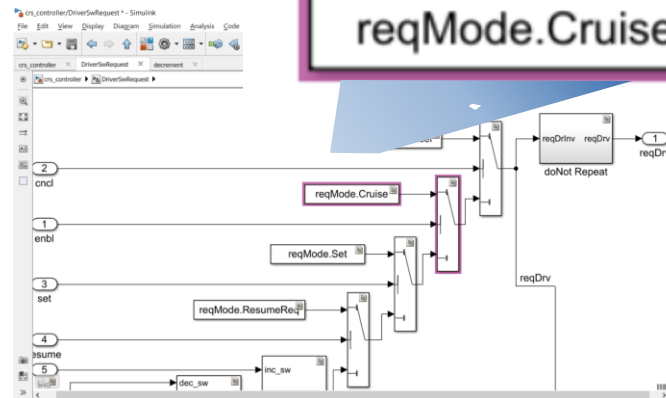
Derives

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

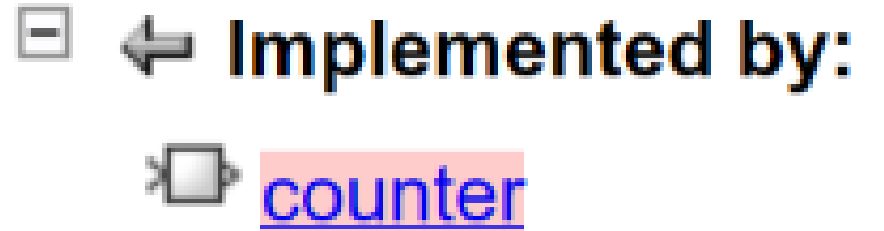
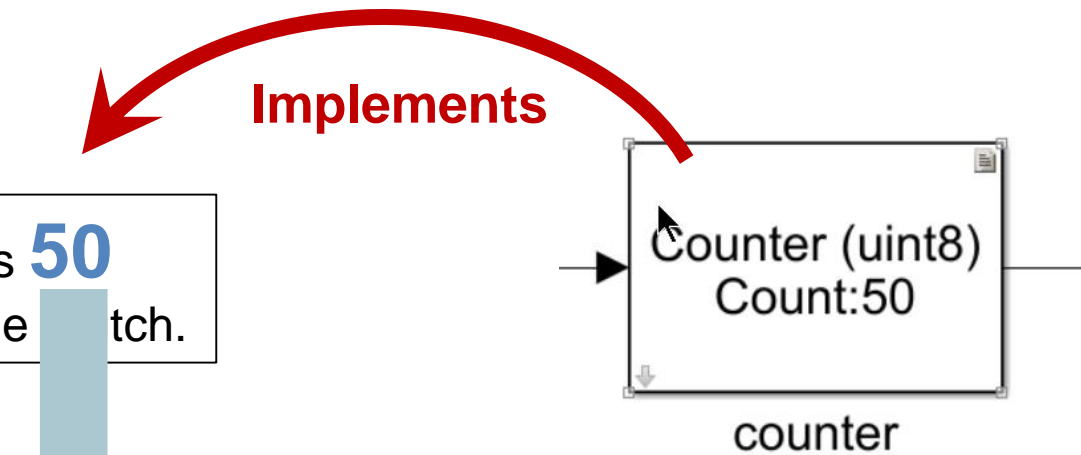
Respond to Change

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.



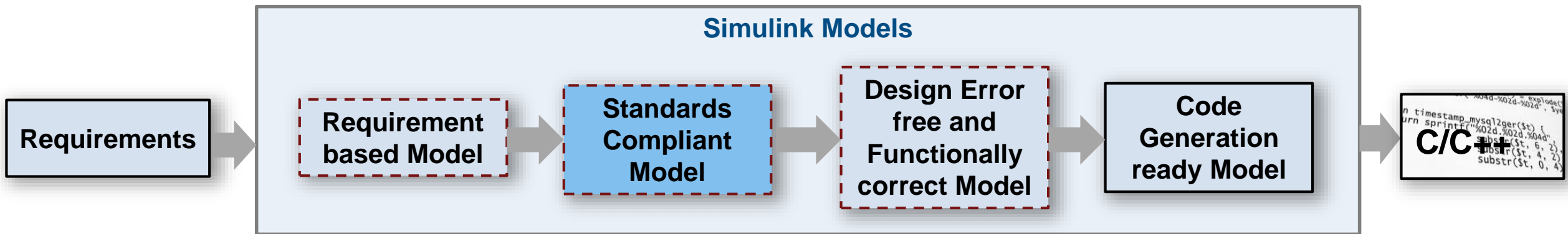
Issue: Destination Changed.

Verify Design to Guidelines and Standards

Is the design built right?

Is it too complex?

Is it ready for code generation?



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.

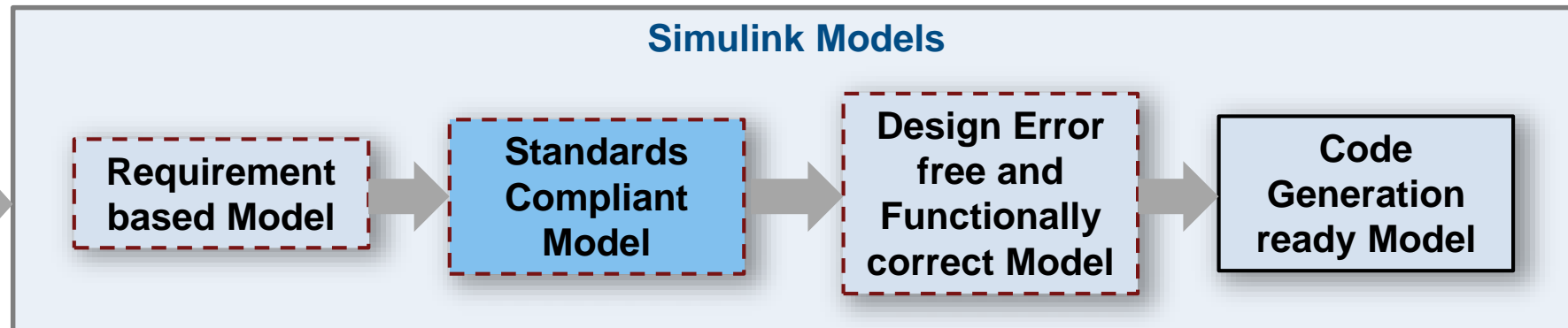
Warning
The following blocks are not supported or not recommended for C/C++ production code deployment:

Block	Block Type	Code generation support	Recommendation for C/C++ production code deployment
../Intake Manifold/p0 = 0.589 bar	Integrator	Yes ^{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.

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 ^{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

Run Summary

Pass	Fail	Warning	Not Run	Total
203	0	215	196	614

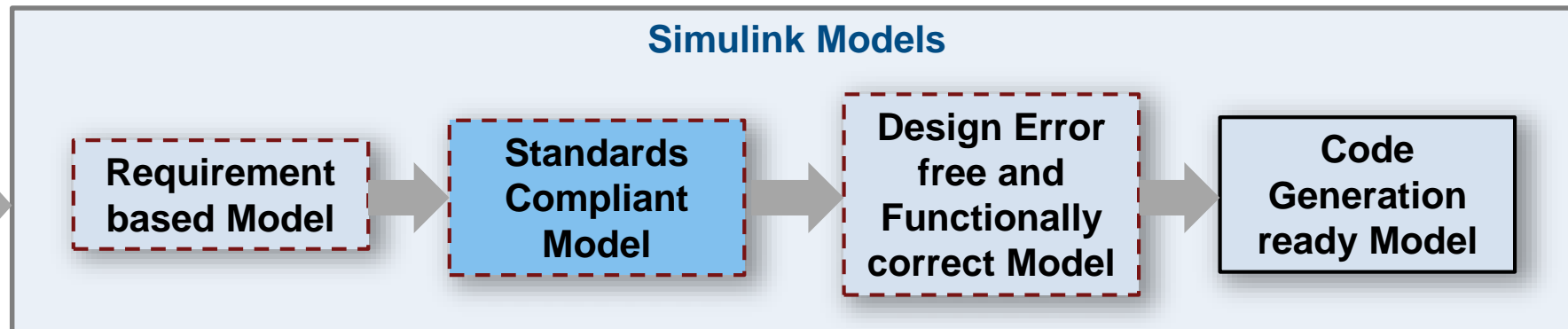
By Task

- 1 Code Generation Efficiency: 3 Pass, 0 Fail, 3 Warning, 3 Not Run

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

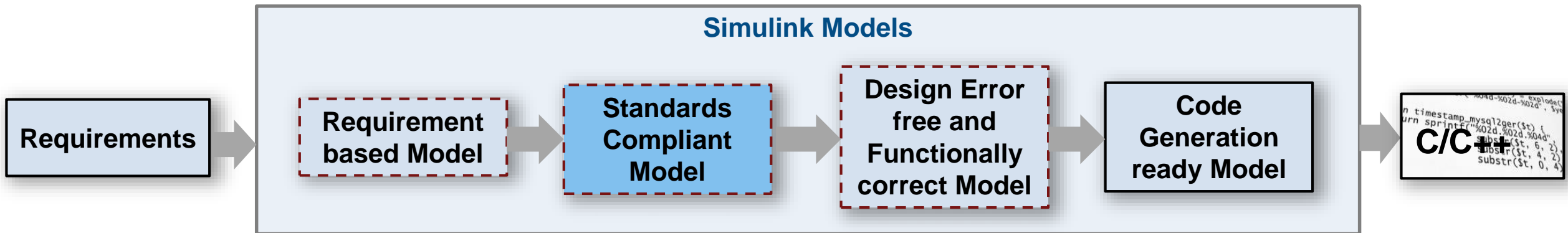
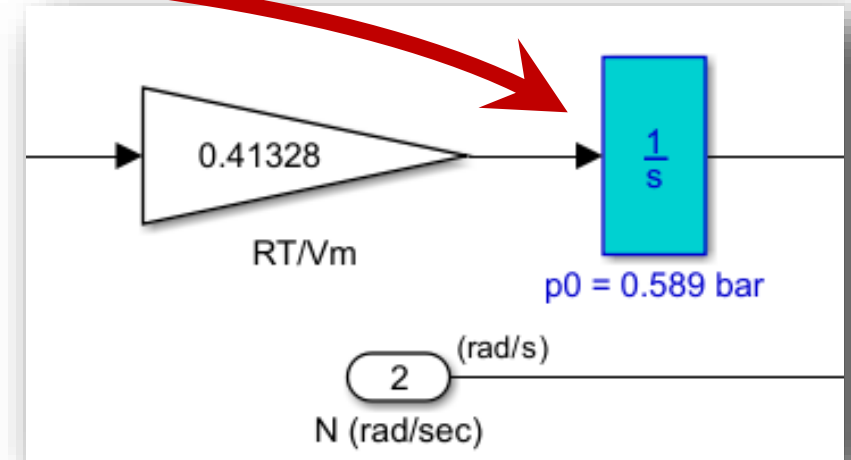
Warning

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

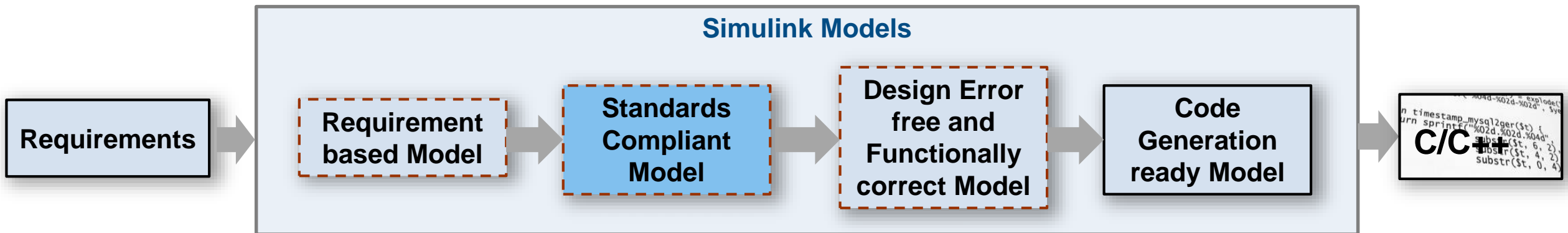
Block	Block Type	Code generation support	Recommendation for C/C++ production code deployment
.../Intake Manifold/p0 = 0.589 bar	Integrator	Yes ^{1, 2}	No
sldemo_fuelsys/Throttle Command	Repeating table	Yes ³	No



Guidance Provided to Address Issues or Automatically Correct

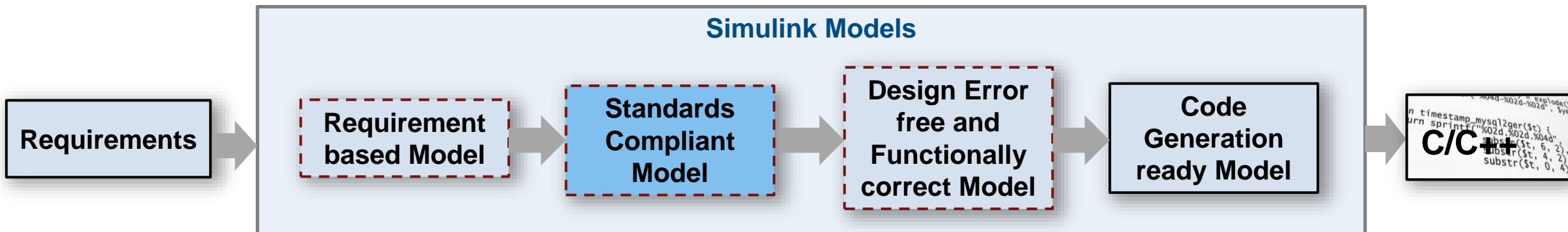
Recommended Action

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.

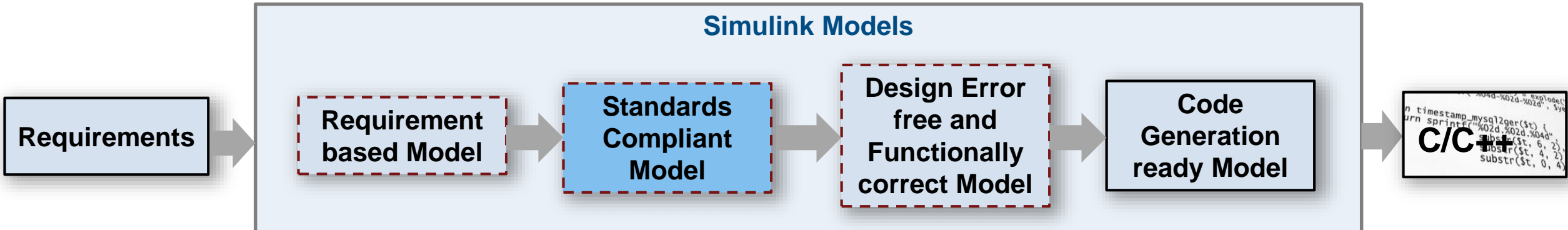


Built in checks for industry standards and guidelines

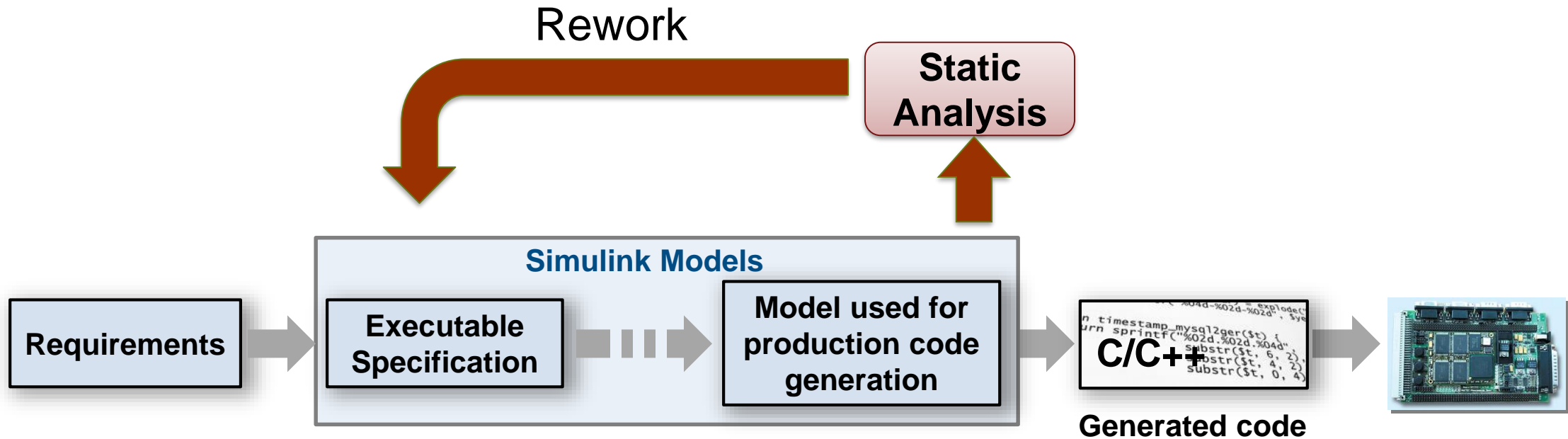
- DO-178/DO-331
- MISRA C:2012
- ISO 26262
- CERT C, CWE, ISO/IEC TS 17961
- IEC 61508
- MAAB (MathWorks Automotive Advisory Board)
- IEC 62304
- JMAAB (Japan MATLAB Automotive Advisory Board)
- EN 50128



Configure and customize analysis

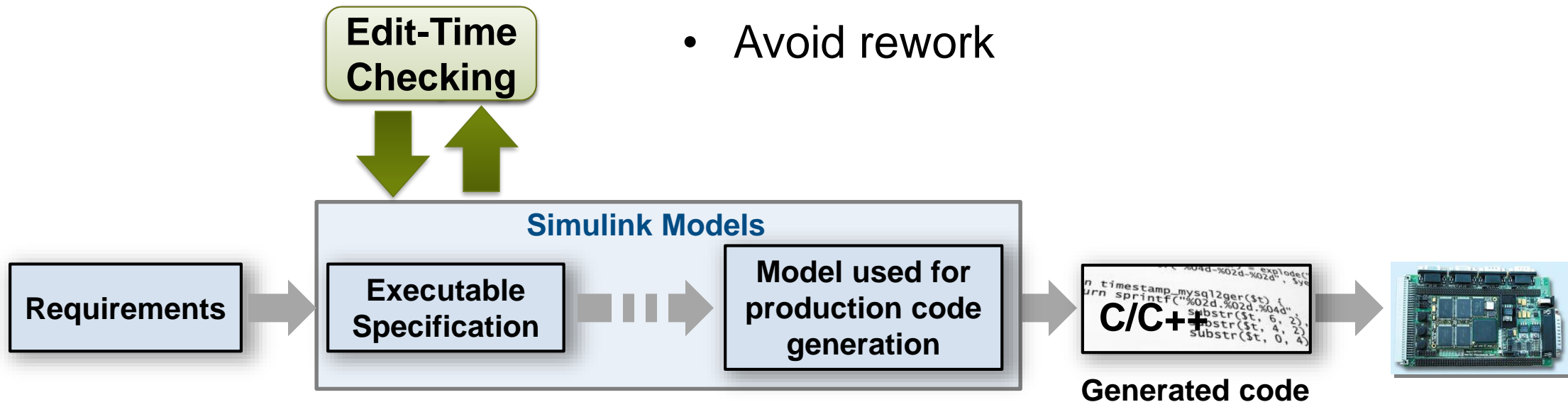


Checks for standards and guidelines are often performed late

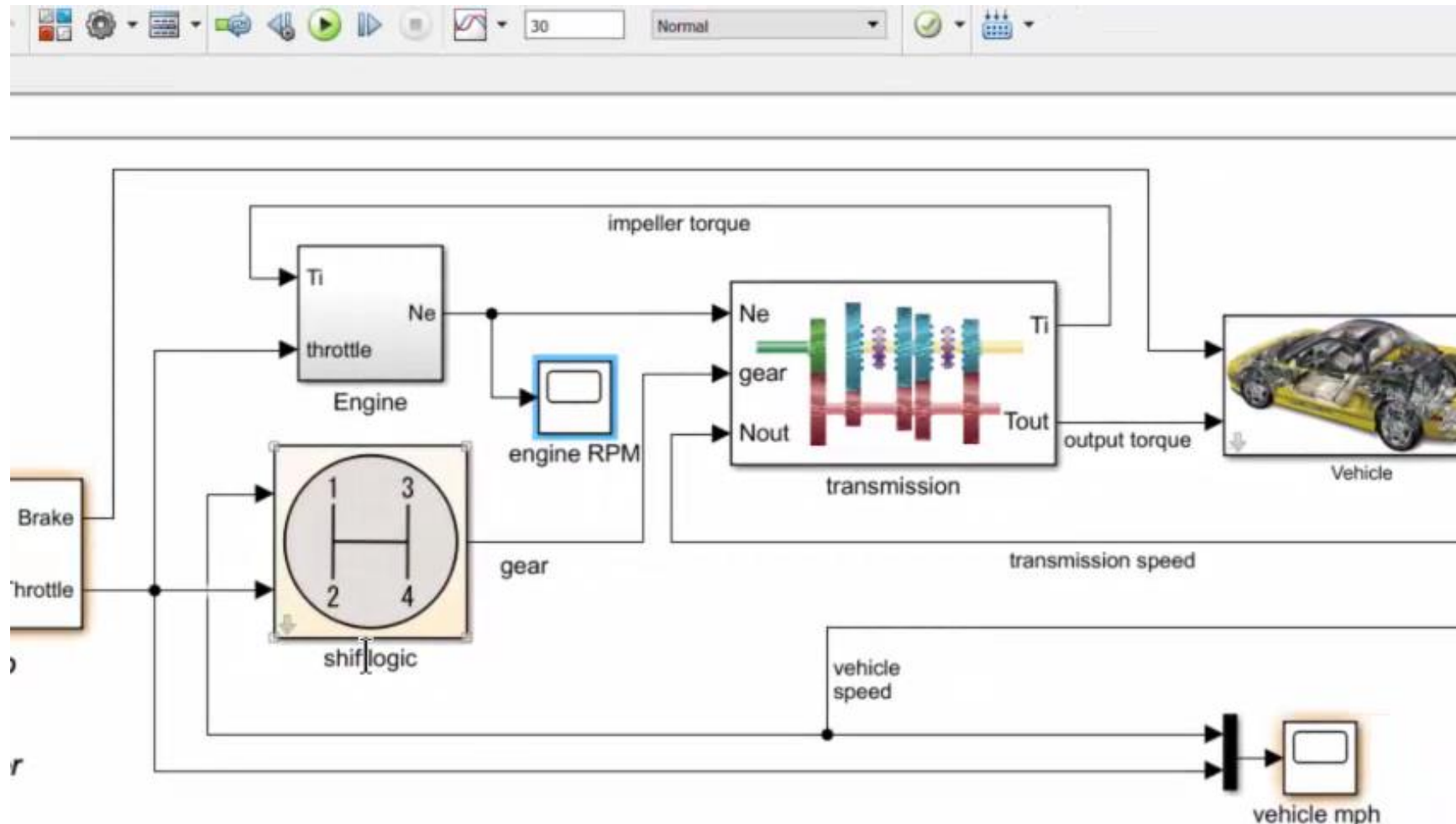


Shift Verification Earlier With Edit-Time Checking

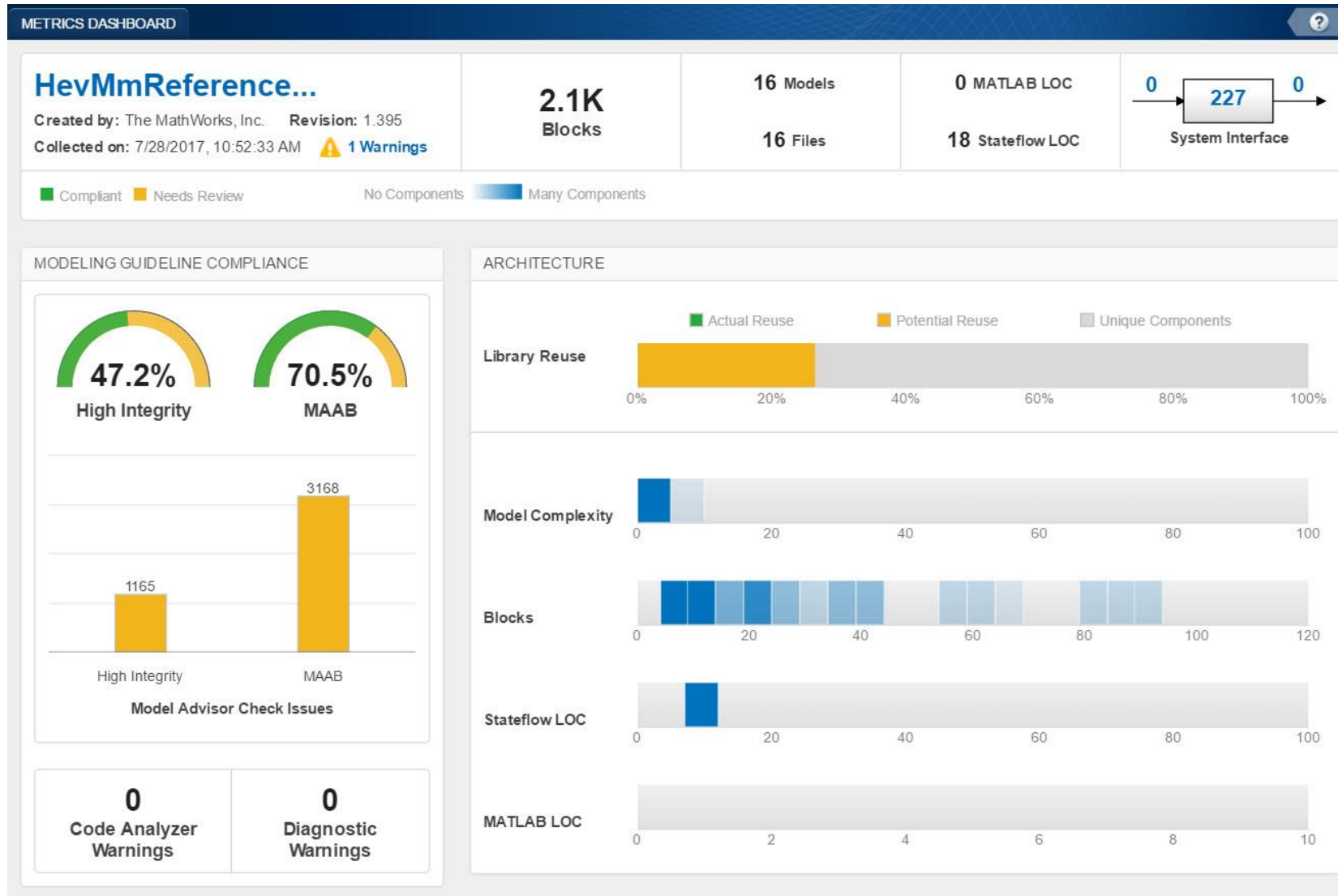
- Highlight violations as you edit
- Fix issues earlier
- Avoid rework



Find Compliance Issues as you Edit with Edit-Time Checking



Assess Quality with Metrics Dashboard

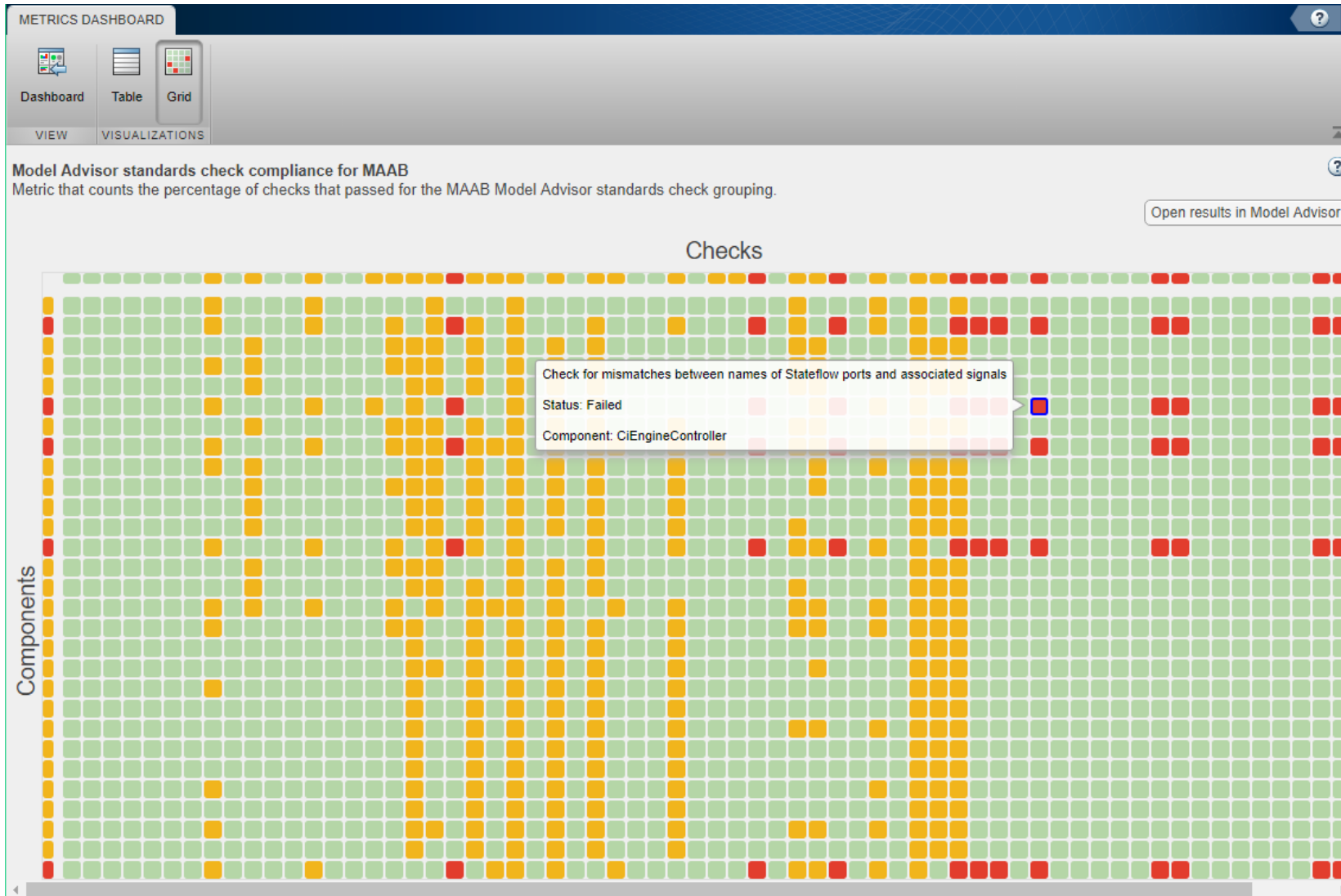


- Consolidated view of metrics
 - Size
 - Compliance
 - Complexity

- Identify where problem areas may be

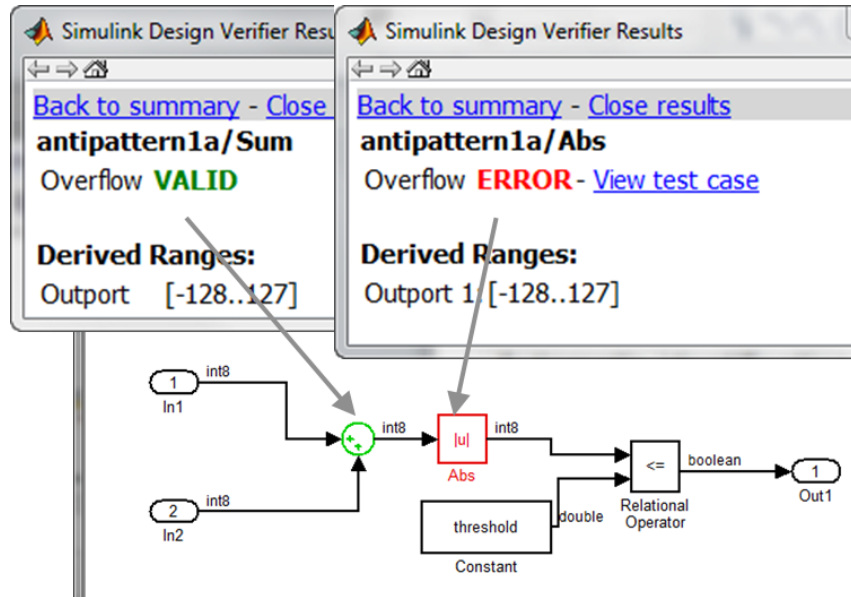
Grid Visualization for Metrics

R2018a



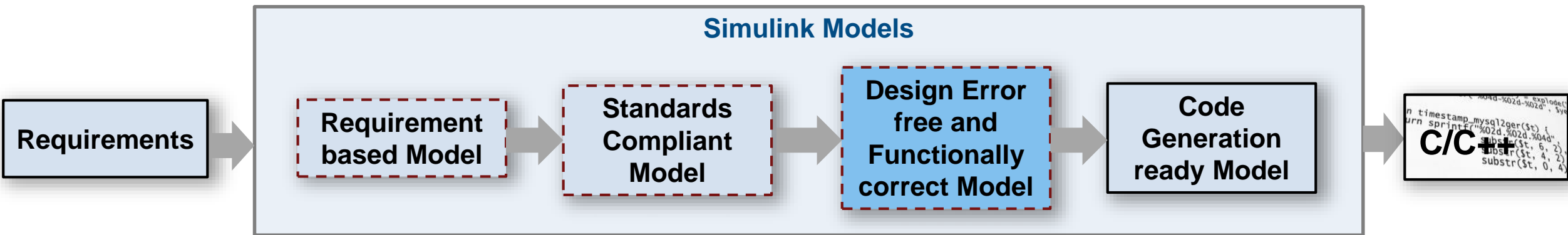
- Visualize Standards Check Compliance
 - Find Issues
 - Identify patterns
 - See hot spots

Detect Design Errors with Formal Methods

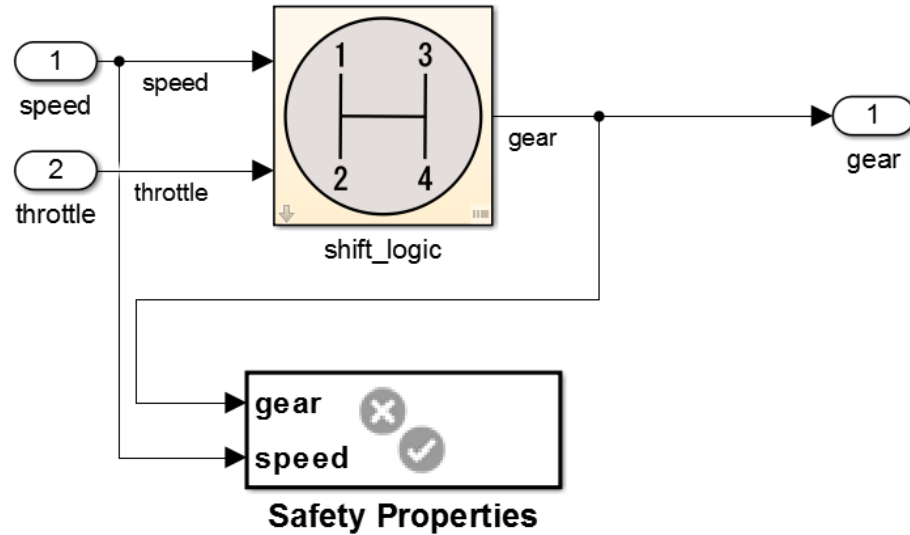


- Find run-time design errors:
 - Integer overflow
 - Dead Logic
 - Division by zero
 - Array out-of-bounds
 - Range violations

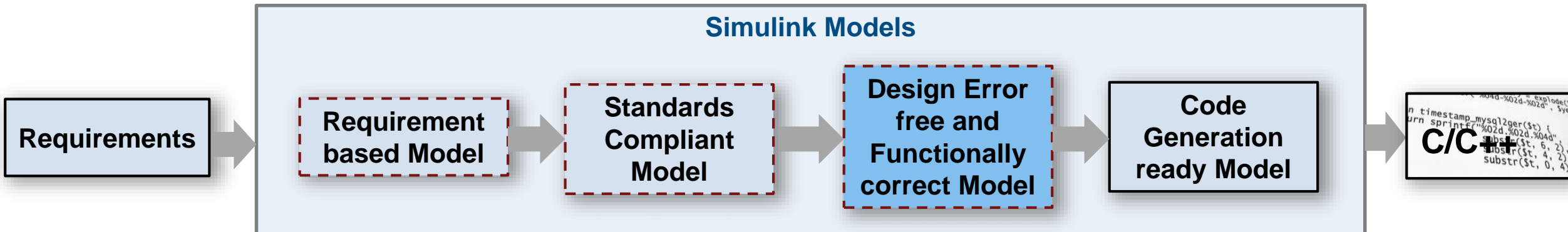
- Generate counter example to reproduce error



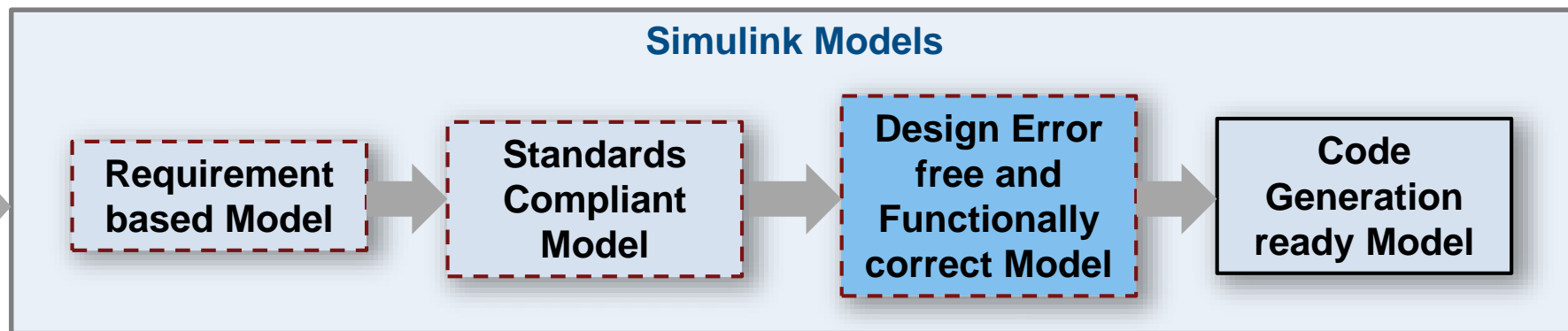
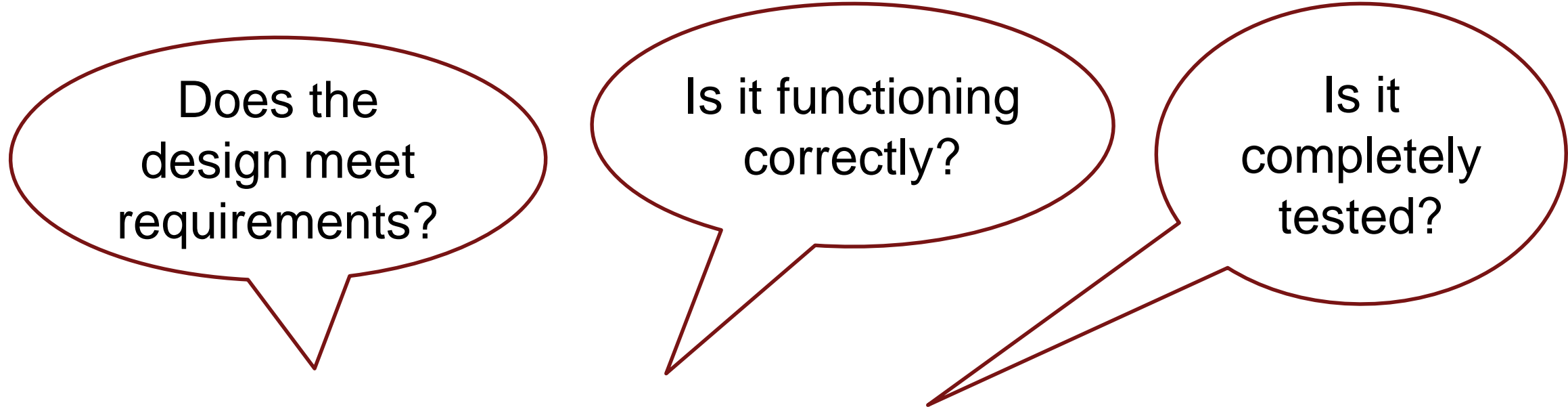
Prove That Design Meets Requirements



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



Functional Testing



```

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

C/C++

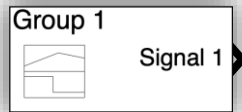
Systematic Functional Testing

Test Case

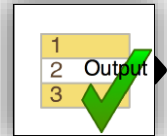
Inputs



MAT file (input)



Signal Builder



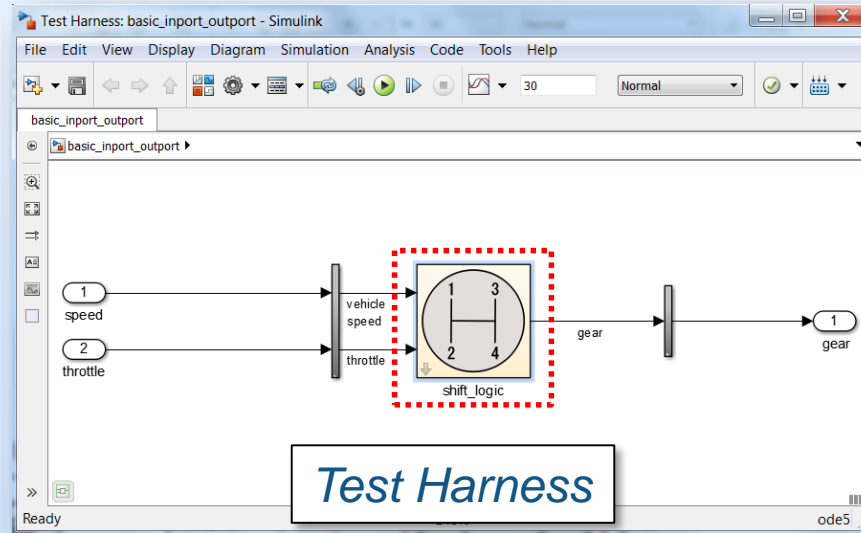
Test Sequence

and more!

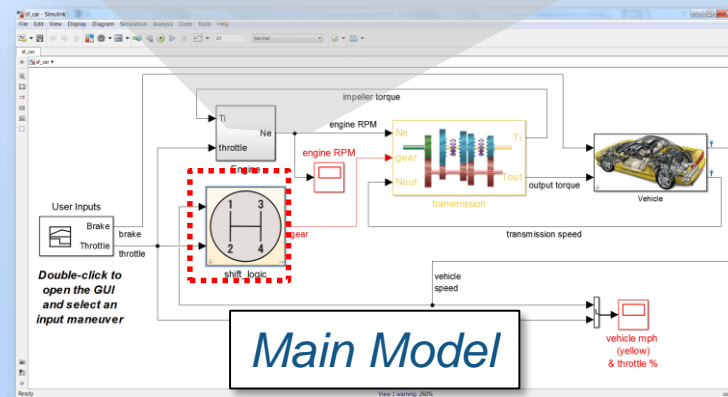


Excel file (input)

R2017b



Test Harness



Main Model

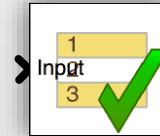
Assessments



MAT file (baseline)

```
function customCriteria
Perform custom criteria
1 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\monelli\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
- Dead Logic

Simulink

Stateflow

Generated Code

Coverage Reports

Summary

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

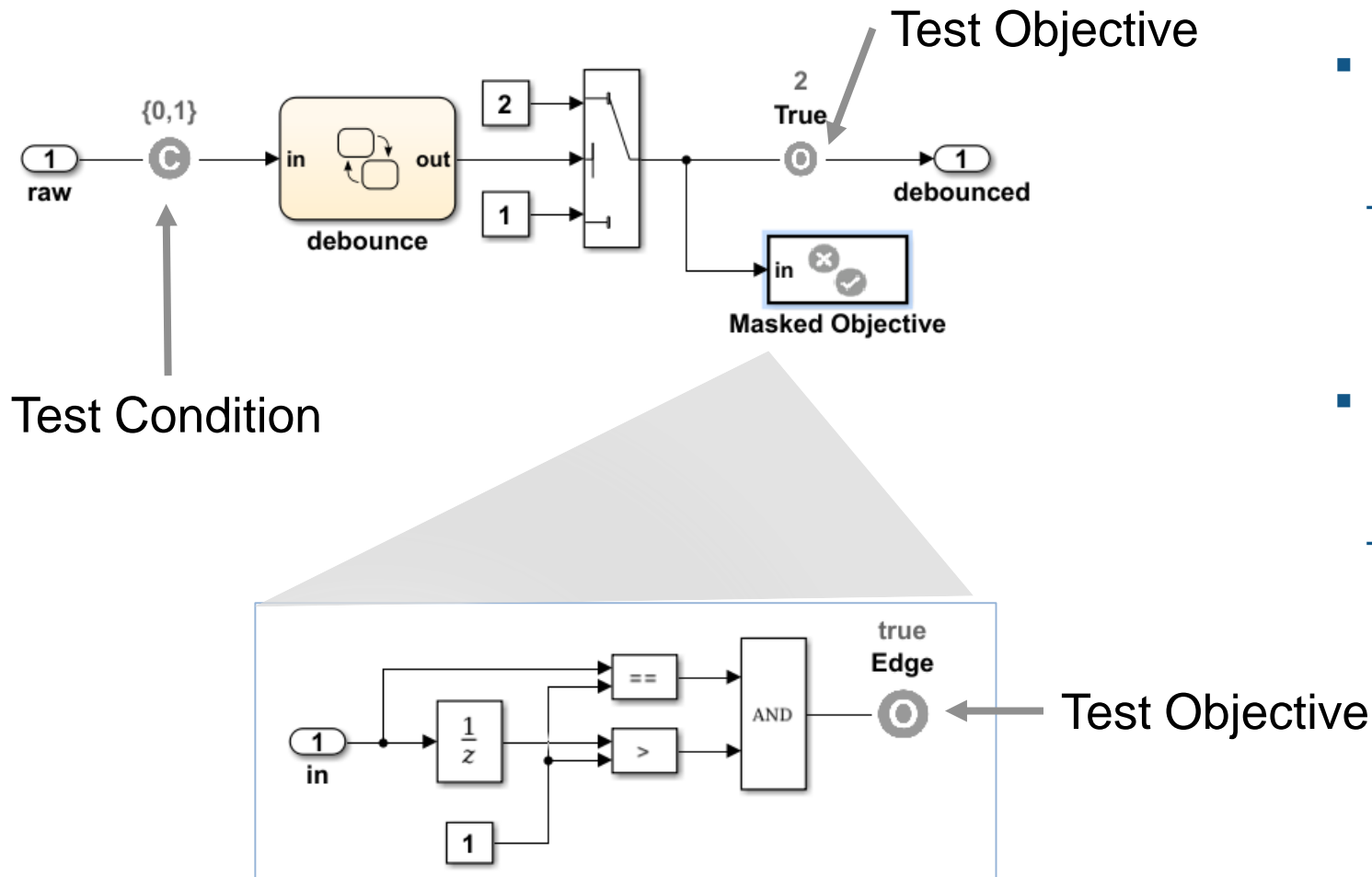
MC/DC analysis (combinations in parentheses did not occur)

decision outcomes:	True	False
	Out	Out

Coverage Reports Table:

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

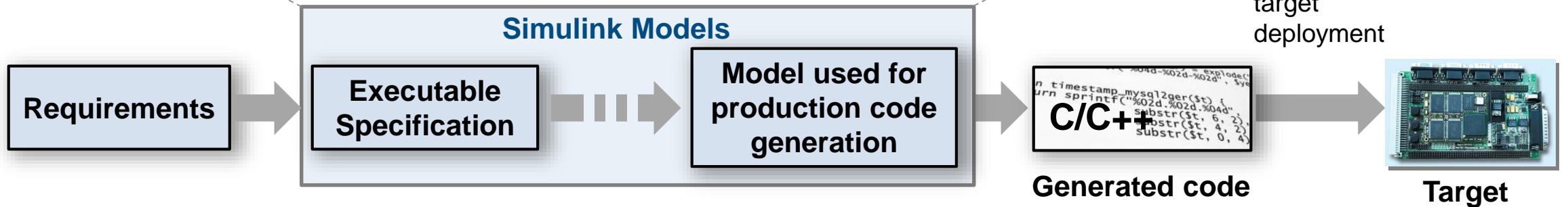
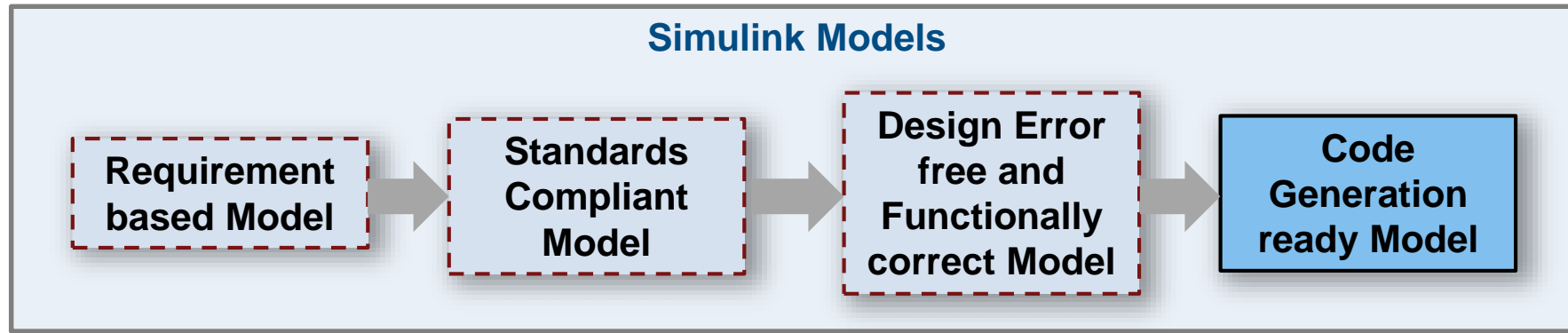
Test Case Generation for Functional Testing



- 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

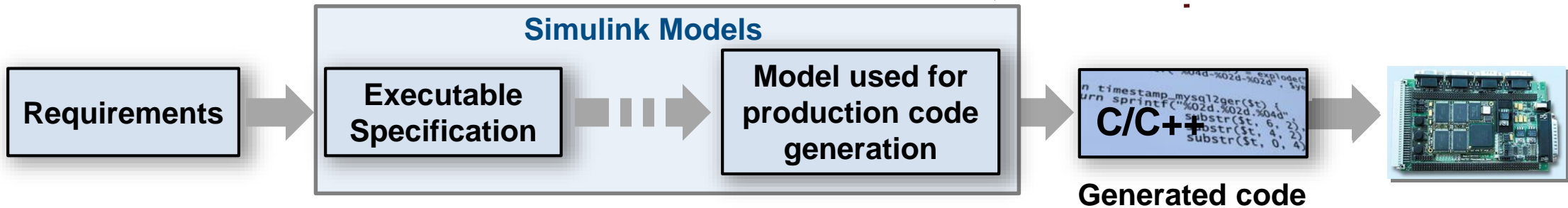
Model-Based Design, Verification and Validation



Equivalence Testing

Is the code functionally equivalent to model?

Is all the code tested?

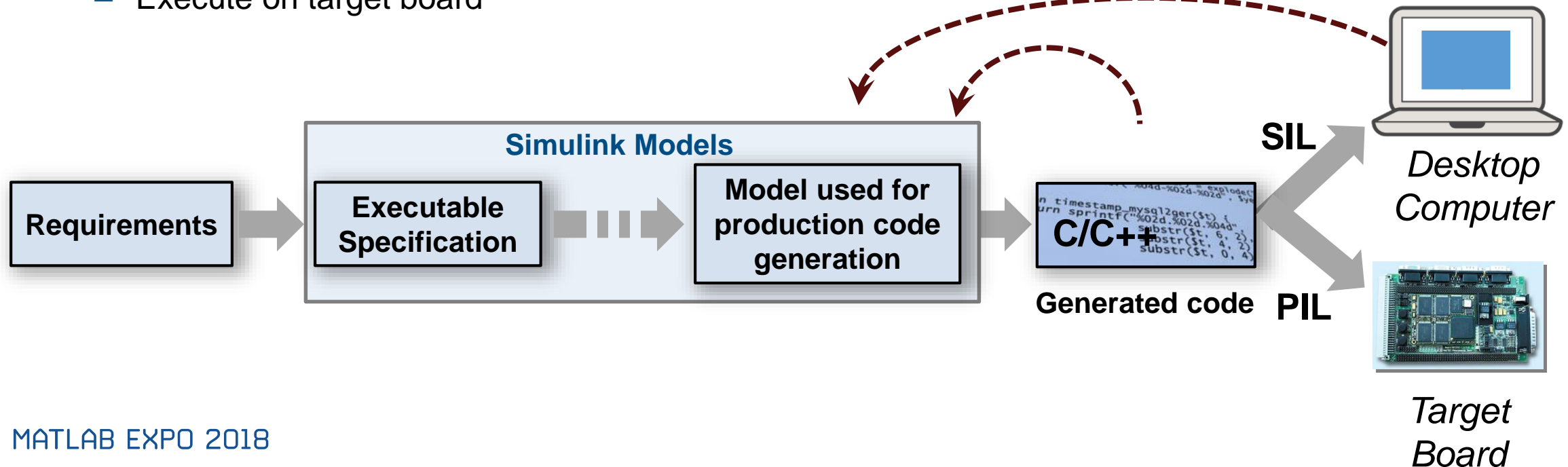


Equivalence Testing

- 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

Benefits

- Re-use tests developed for model to test code
- Collect code coverage
- Generate artefacts for IEC 61508, ISO 26262, EN 50128, and DO-178 certification
- Early verification and defect detection

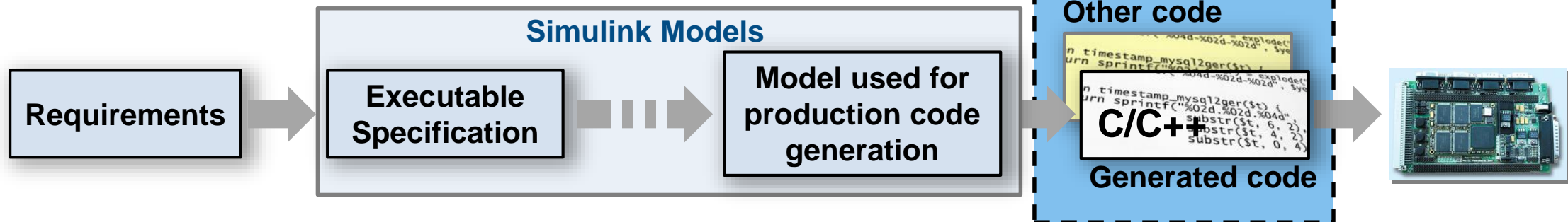


Static Code Analysis

Is interface between generated and other code fully tested?

Is integrated code free of run-time errors?

Is the code compliant to MISRA?



The Generated Code is integrated with Other Code (Handwritten)

Static Code Analysis with Polyspace

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;
    }
}
    
```

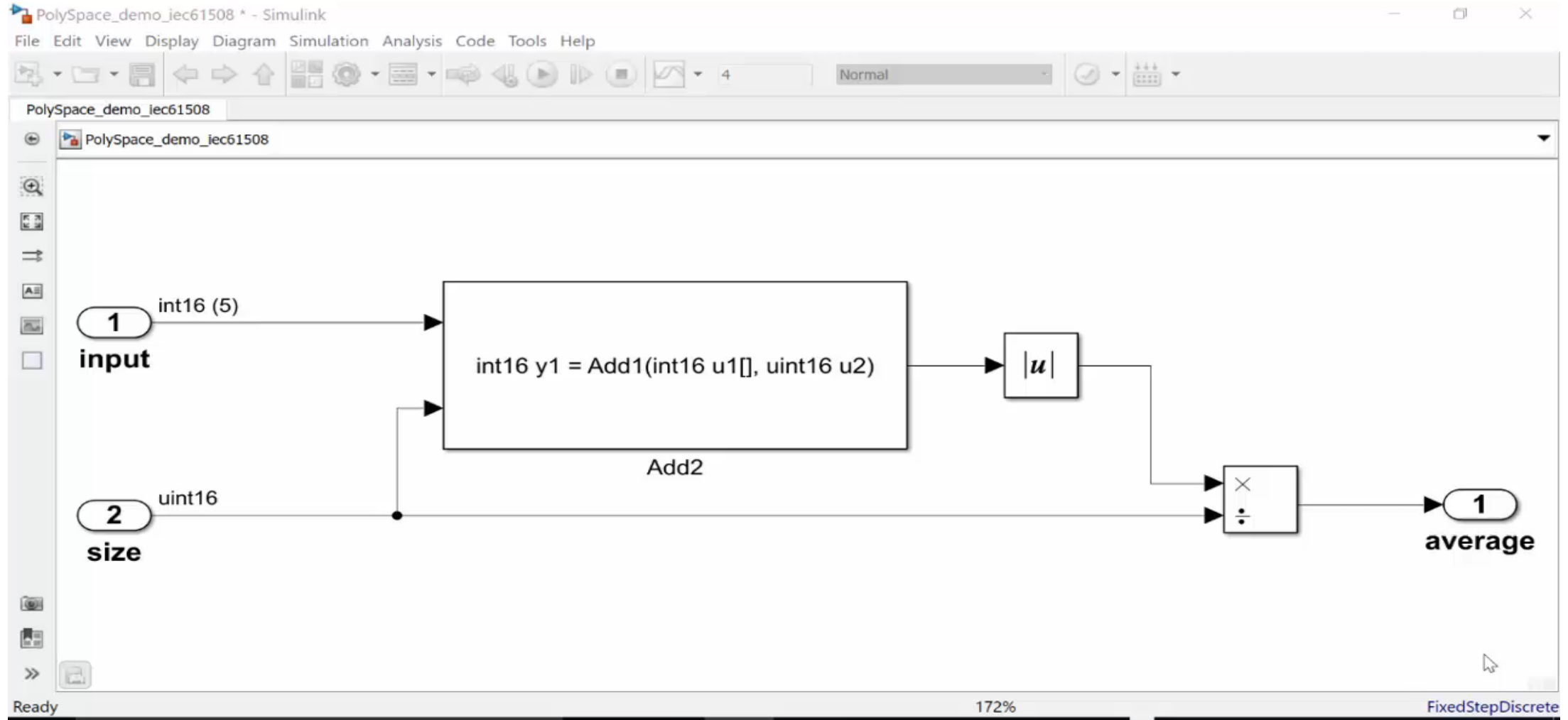
variable 'i' (int32): [0 .. 99]
assignment of 'i' (int32): [1 .. 100]

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

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

Results from Polyspace Code Prover

Code Proving with Polyspace



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.

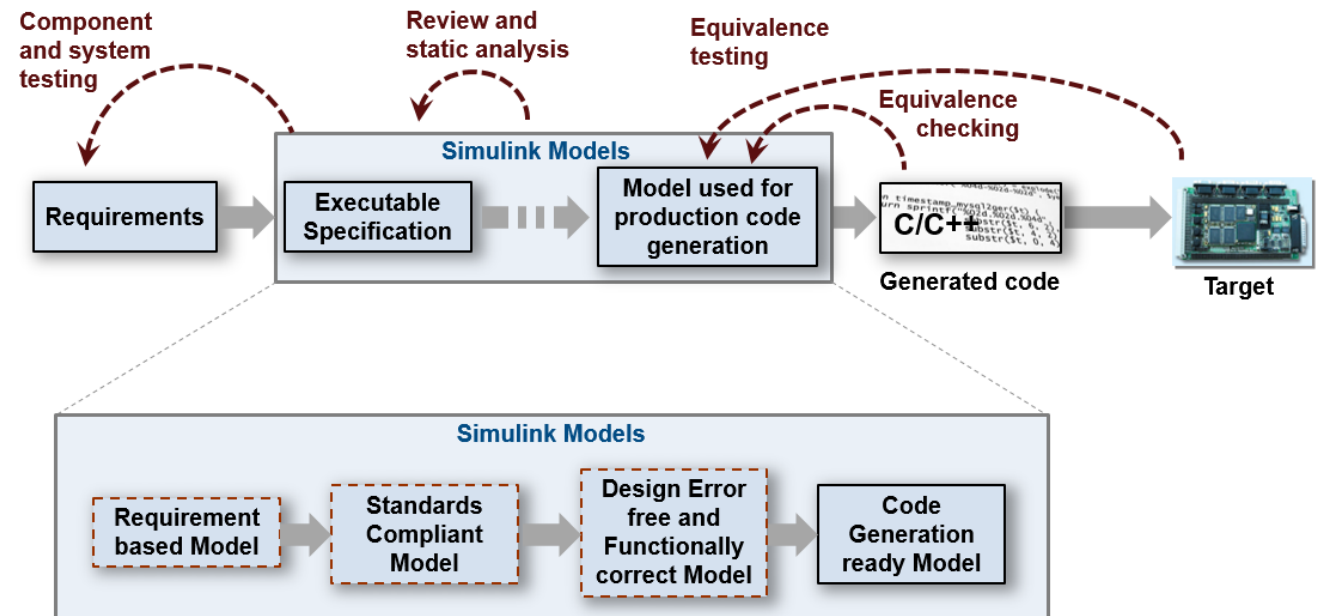
BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design



Primary flight control computers from BAE Systems.

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
5. Static Code verification using Polyspace



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

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

Challenge

Develop automotive electronic steering column lock software and certify it to the highest-level functional safety standard

Solution

Use Model-Based Design to design, implement, and verify the application software via back-to-back PIL testing required for ISO 26262 ASIL D certification

Results

- Development and certification time cut by 30%
- 80% of errors identified in modeling phase
- PIL test framework for ISO 26262 established



Kostal's electronic steering column lock module.

“Using Model-Based Design to design, implement, and verify our software for the highest functional safety standard enabled our team to save costs, increase efficiency, and ensure software quality. Without Model-Based Design, more engineers would be needed to complete the project in the same time frame.”

– Cheng Hui, KOSTAL

Miele Proves Absence of Run-Time Errors in Control Software Across Its Entire Product Line

Challenge

Maintain a reputation for producing quality appliances and other products by minimizing defects in the control software

Solution

Integrate Polyspace Code Prover and Polyspace Bug Finder into the development process to prove the absence of run-time errors in the software and enforce standard coding rules

Results

- Hundreds of source files analyzed daily
- Developer focus on core functionality enabled
- Reusable, trusted components proven free of run-time errors



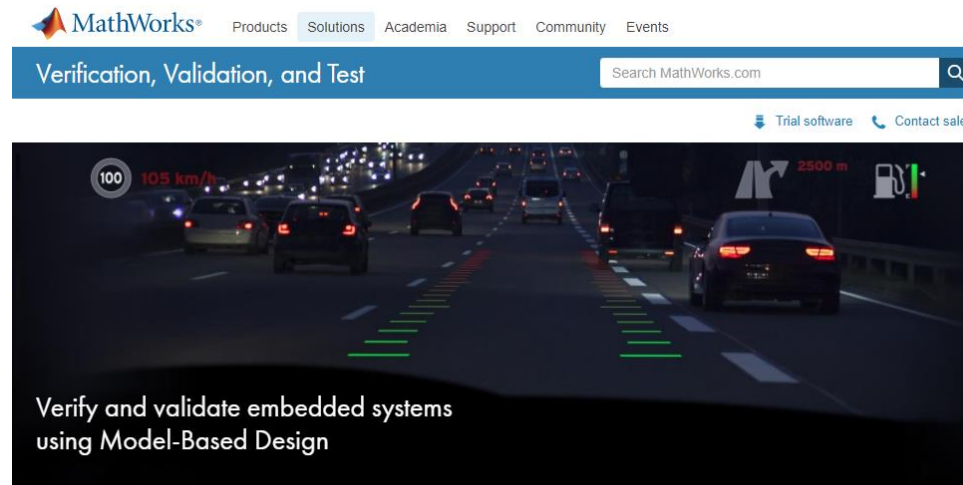
The Miele Center Gütersloh in Germany.

“We have embedded static code analysis with Polyspace products deeply into our quality assurance processes. It is much better to find run-time errors as development begins than to find them at the end of development—or worse, after the product is delivered.”

- Stefan Trampe, Miele

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

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).

Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses



More than 48 course offerings:

- Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
- Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis, and other areas

 **MathWorks®** | *Training Services*

Verification and Validation of Simulink Models

This one-day course describes techniques for testing Simulink model behavior against system requirements.

Topics include:

- Identifying the role of verification and validation in Model-Based Design
- Creating test cases for Simulink models
- Analyzing simulation results to verify model behavior
- Automating testing activities and managing results
- Formally verifying model behavior
- Automatically generating artifacts to communicate results

 **MathWorks®** | *Training Services*

Polyspace for C/C++ Code Verification

This two-day course discusses the use of Polyspace Bug Finder™ and Polyspace Code Prover™ to prove code correctness, improve software quality metrics, and ensure product integrity.

Topics include:

- Creating a verification project
- Reviewing and understanding verification results
- Emulating target execution environments
- Handling missing functions and data
- Managing unproven code (color-coded in orange by Polyspace® products)
- Applying MISRA C® rules
- Reporting

Thank You!