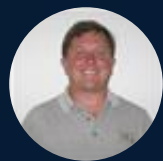


MATLAB EXPO

2021

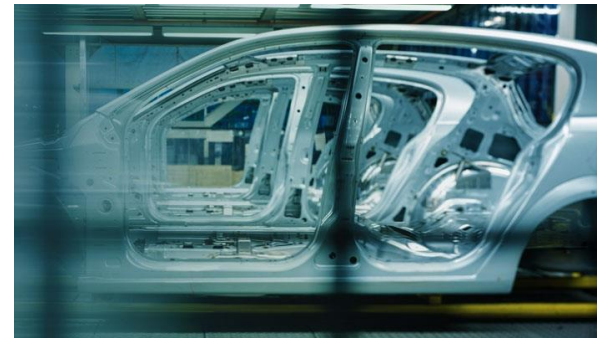
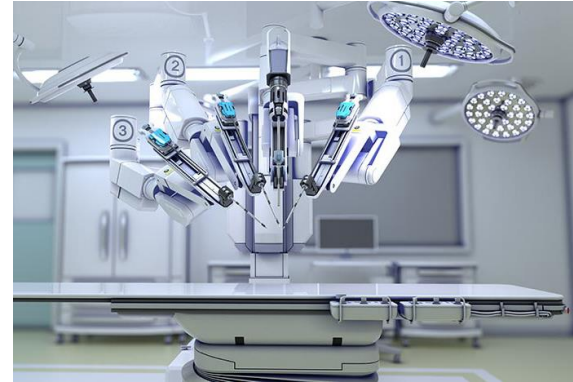
Developing Embedded Software with Model-Based Design to Meet Certification Standards

Paul Urban *Jeff Harper*



Software safety is important across industries

- Software errors are common
 - Automotive: **19%** of all recalls
 - Medical devices: **26%** of all recalls
- Recalls are extremely expensive
 - MCAS failure: **\$18B+**
 - Broken hip replacement: **\$3B+**
 - Faulty pedals: **\$3B+**
 - Airbag issues: **\$25B+**



Why functional safety?



Poll: Safety standards across different industries



IEC 61508 – All industries



ISO 26262, SOTIF – Automotive



DO 178C – Avionics



IEC 62304 – Medical



EN 50128 – Rail



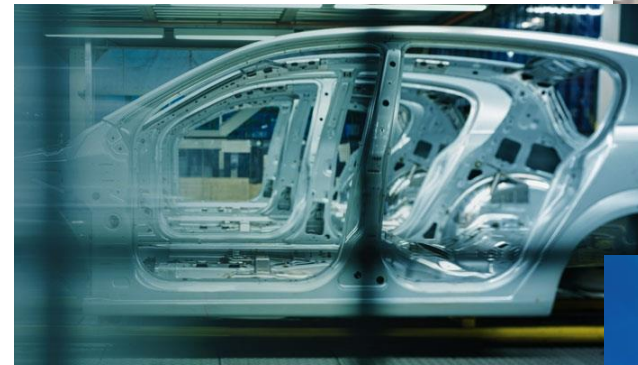
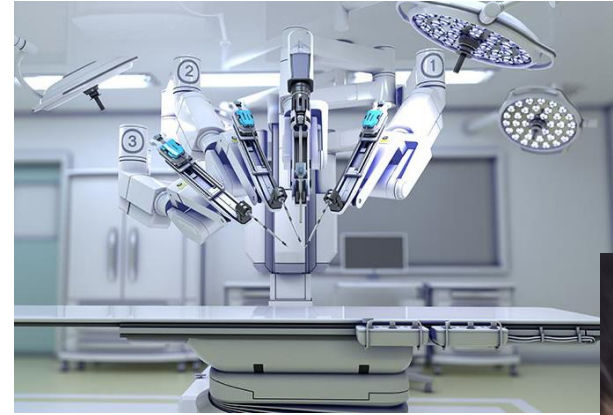
ISO 25119 – Agriculture and Forestry



IEC 61511 – Process Control



And many others.....

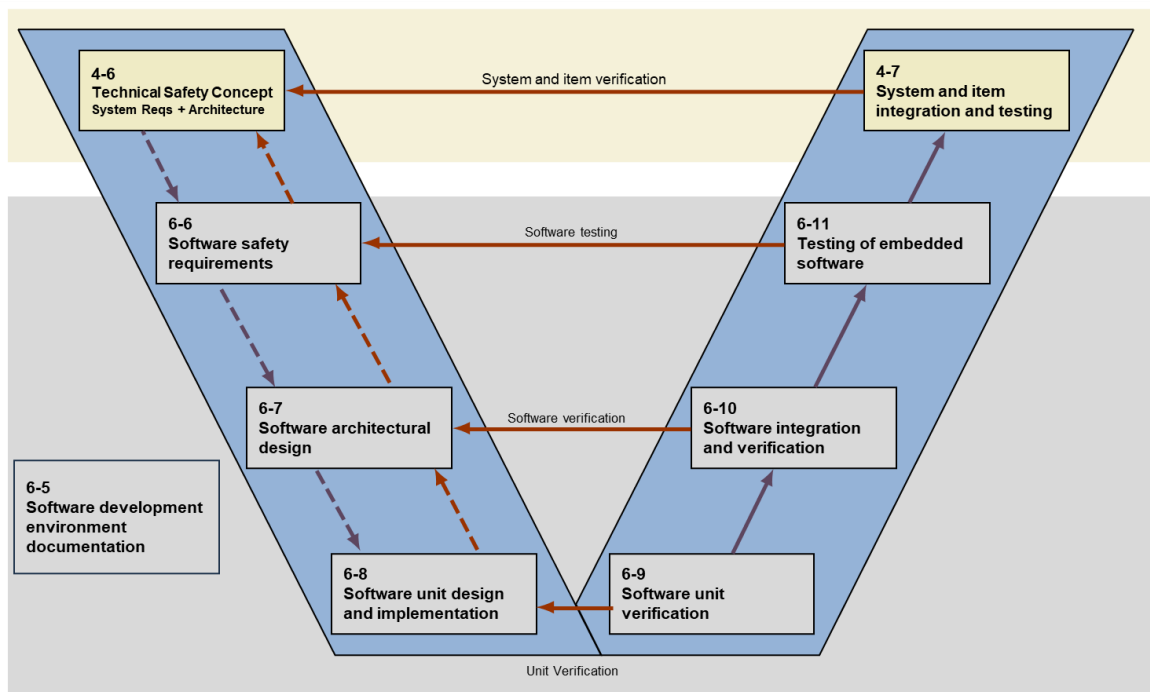


Key Takeaways

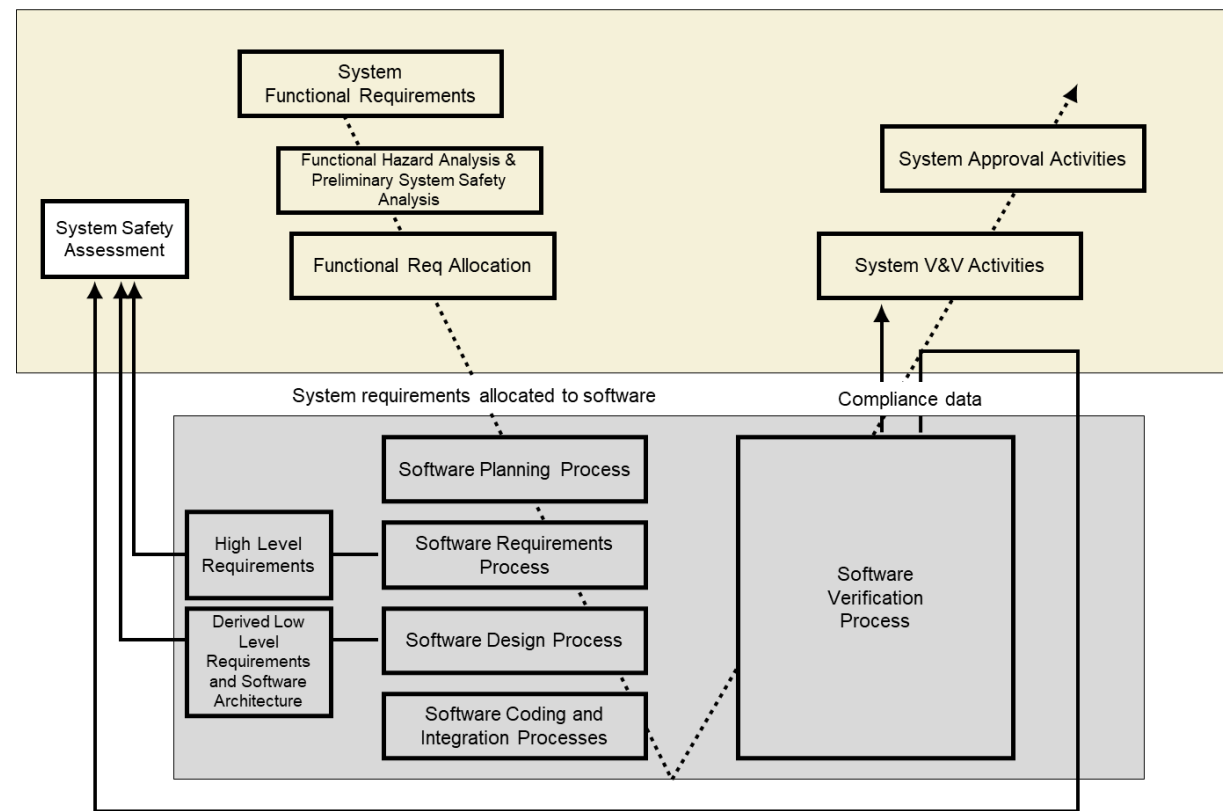
Use Reference Workflow with Model-Based Design to Meet Certification Standards

- Create **traceability** across requirements, architecture, design, test and code
- Detect errors **earlier** by continuous testing
- **Reduce coding errors** with automatic code generation
- **Automate** generation of documents and reports for reviews and audits

V-Model for Certifiable Product Development



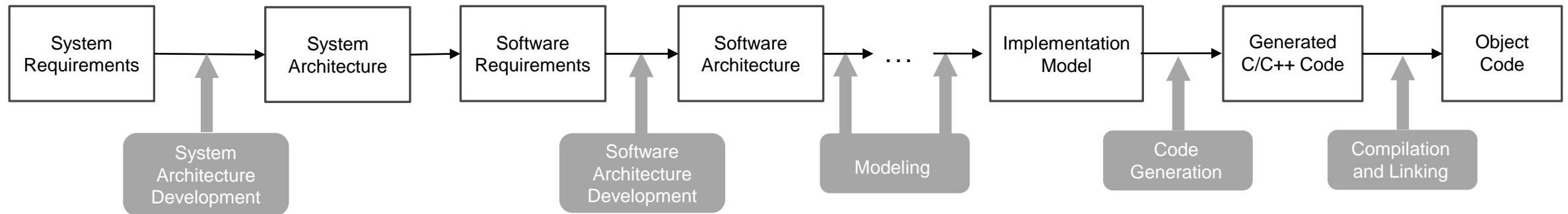
- - - - -> Decomposition, allocation and definition
 - - - - -> Design phase verification
 - - - - -> Integration and recomposition
 - - - - -> Verification and Validation
 □ Scope of part 4
 □ Scope of part 6



DO 178C, ARP 4754A, ARP 4761

ISO 26262: 2018

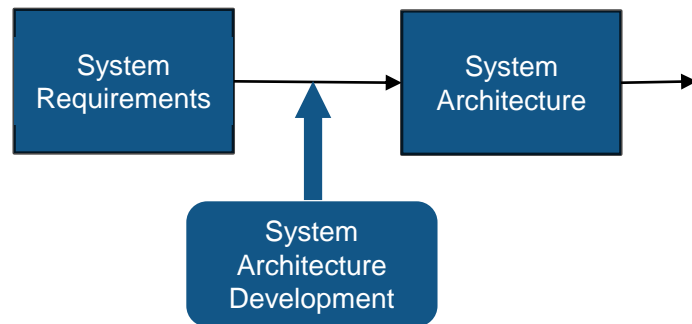
Reference Workflow for Certification



System Level

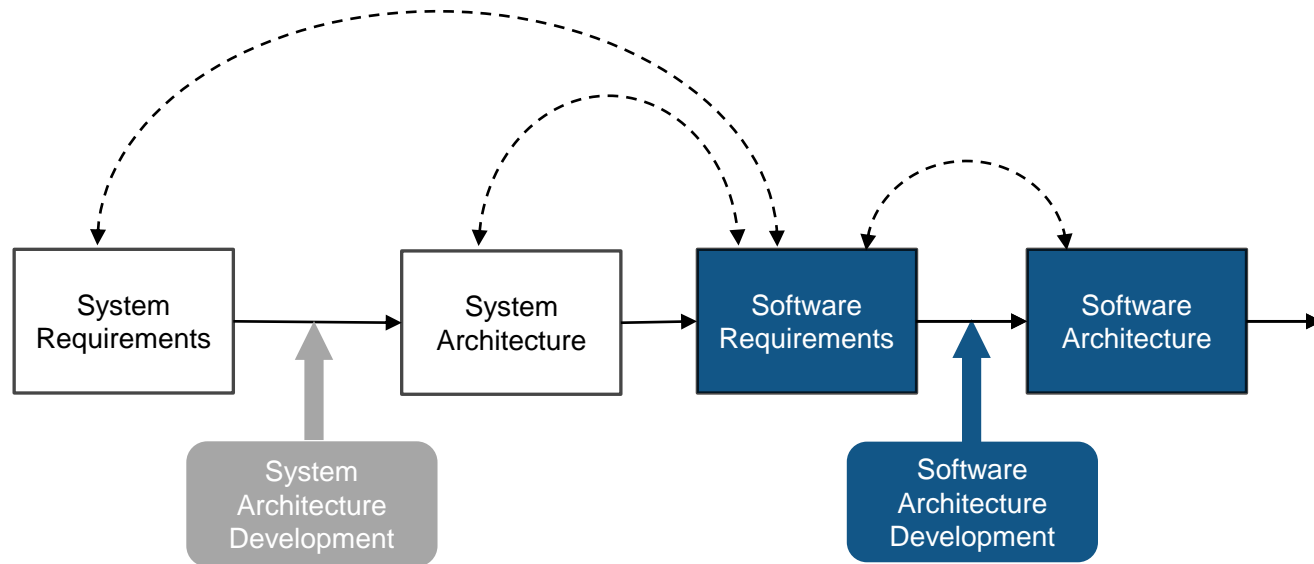
Software Level

Reference Workflow for Certification – System Requirements and Architecture



System Level

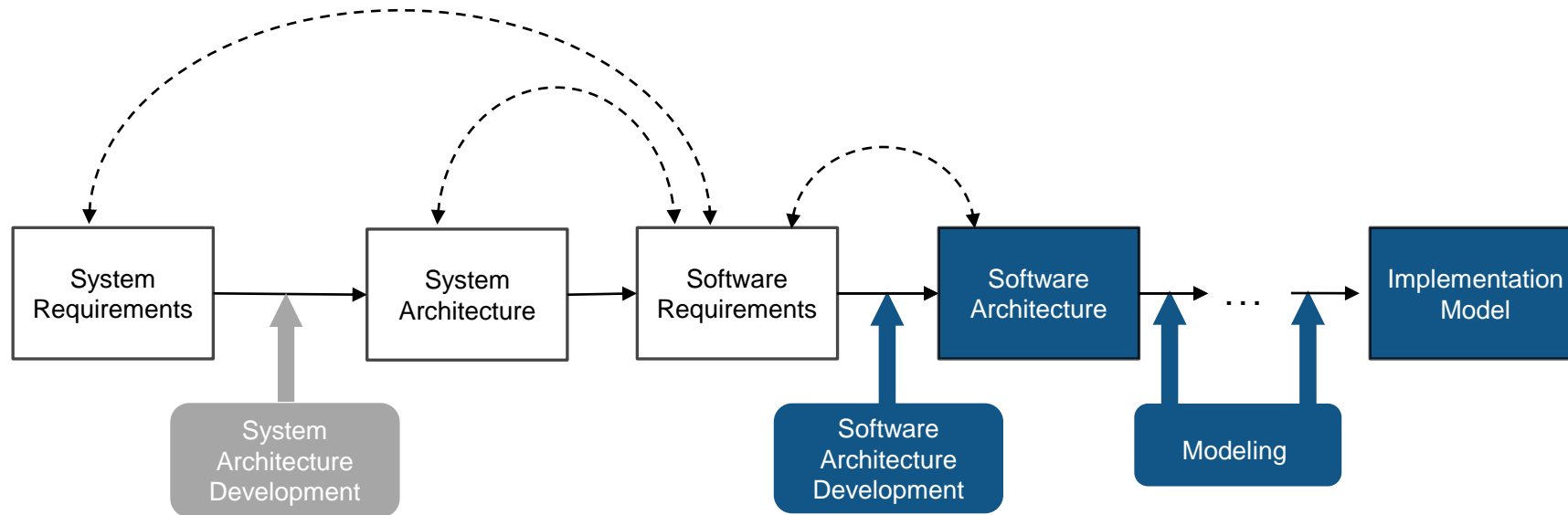
Reference Workflow for Certification – Software Requirements and Architecture



System Level

Software Level

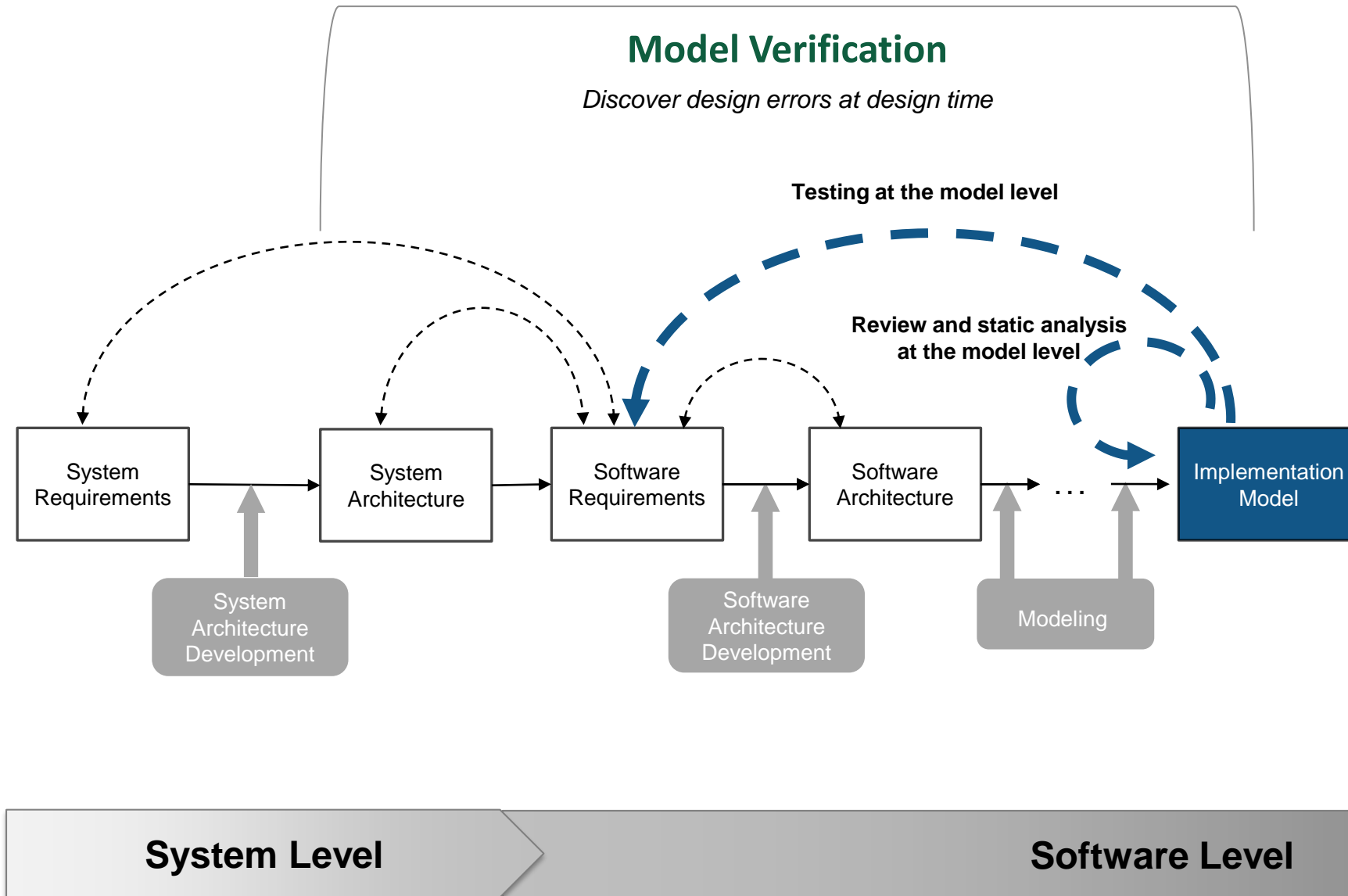
Reference Workflow for Certification – Detailed Design



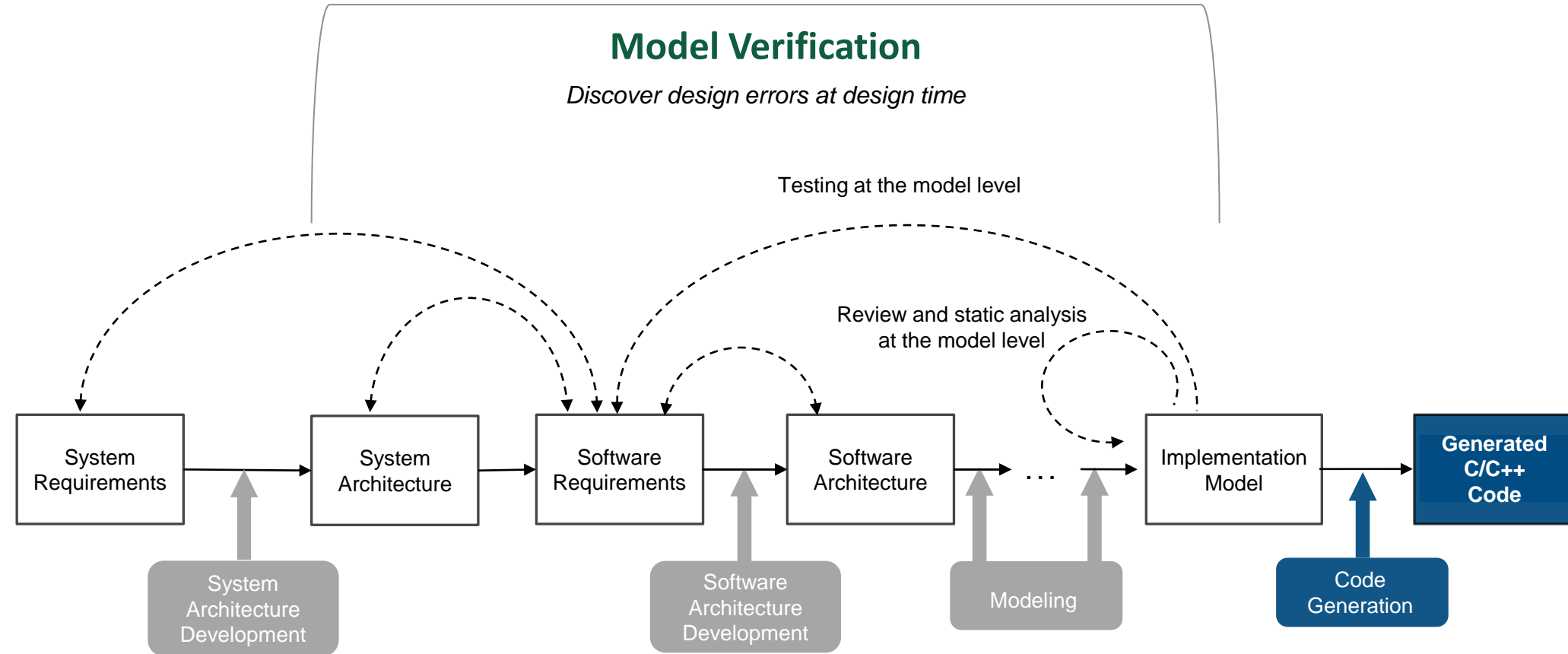
System Level

Software Level

Reference Workflow for Certification – Model Verification



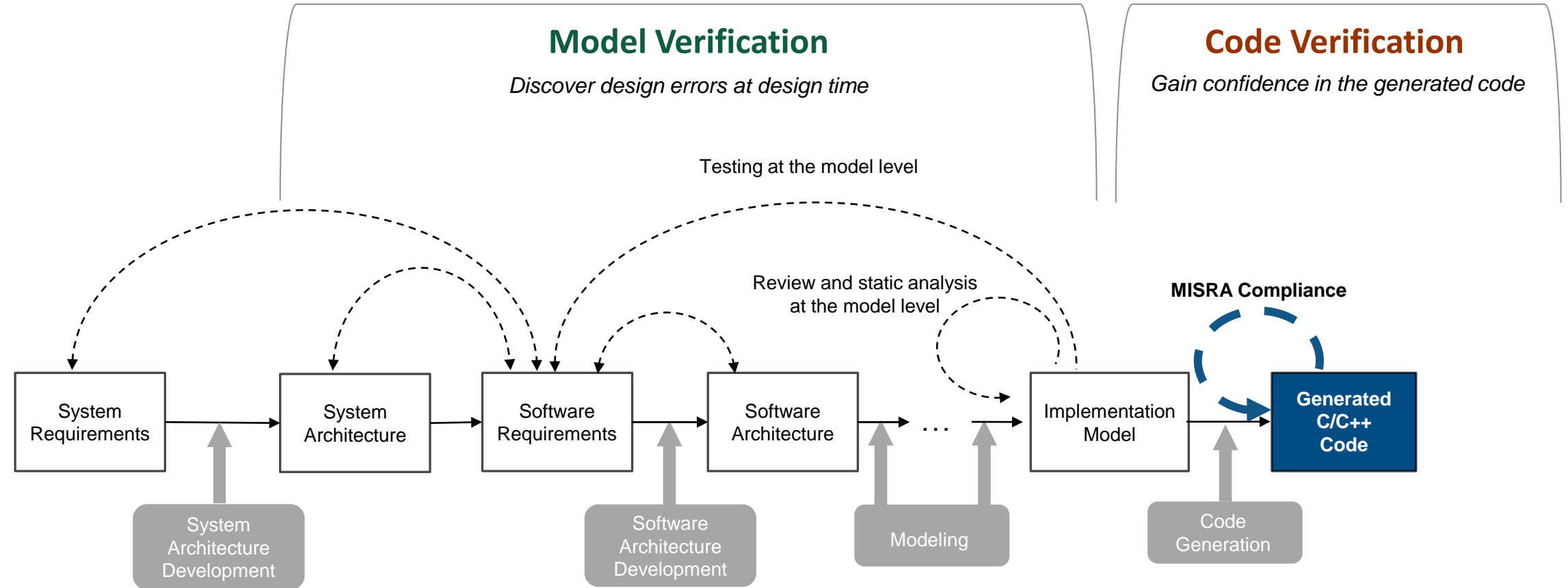
Reference Workflow for Certification – Code Generation



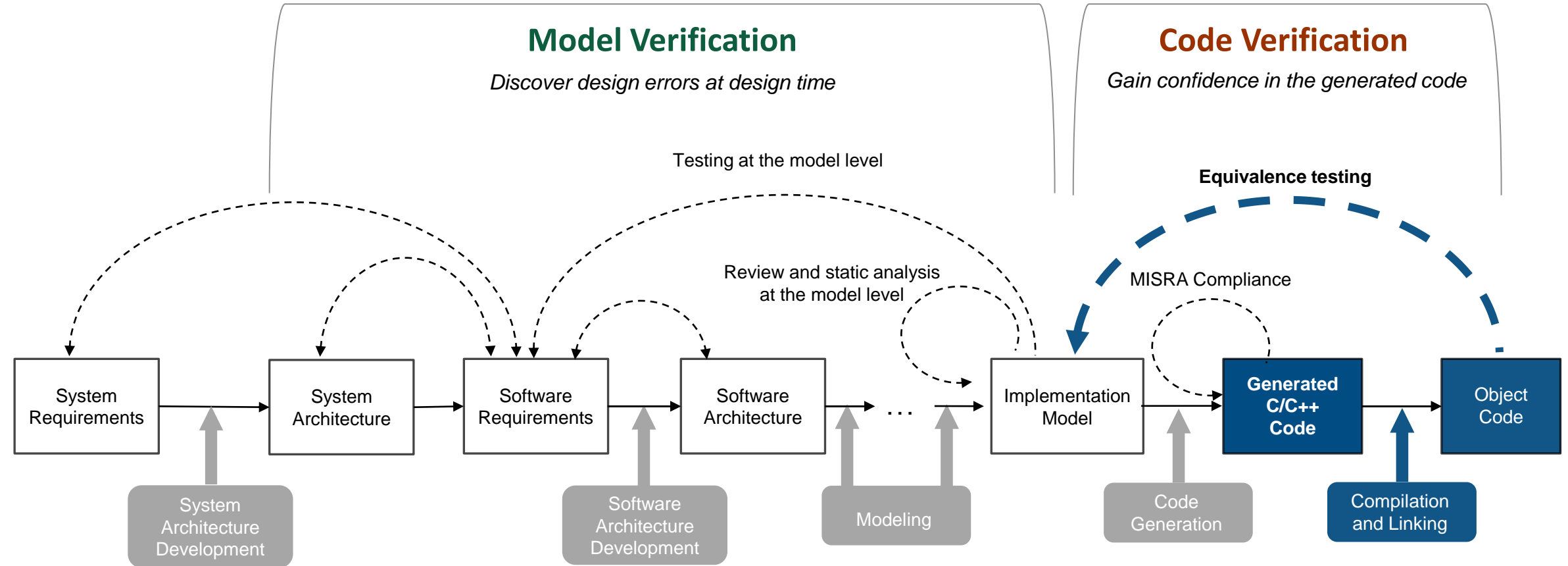
System Level

Software Level

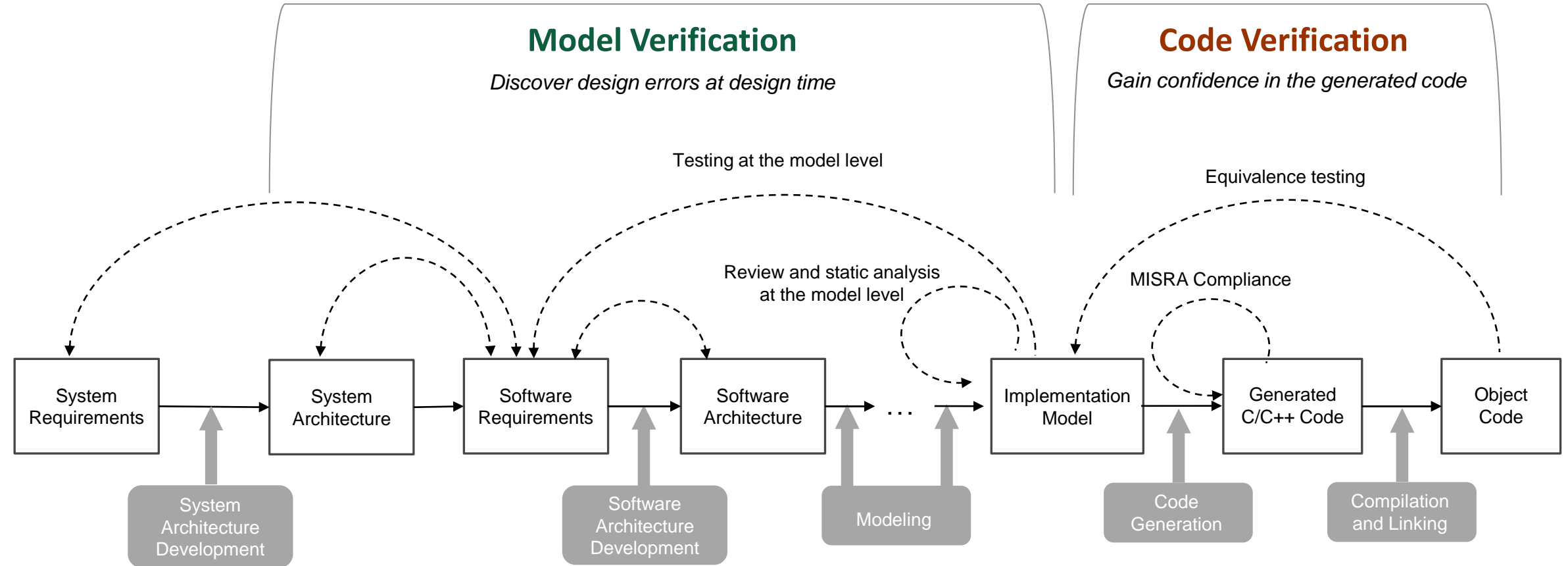
Reference Workflow for Certification – Static Code Verification



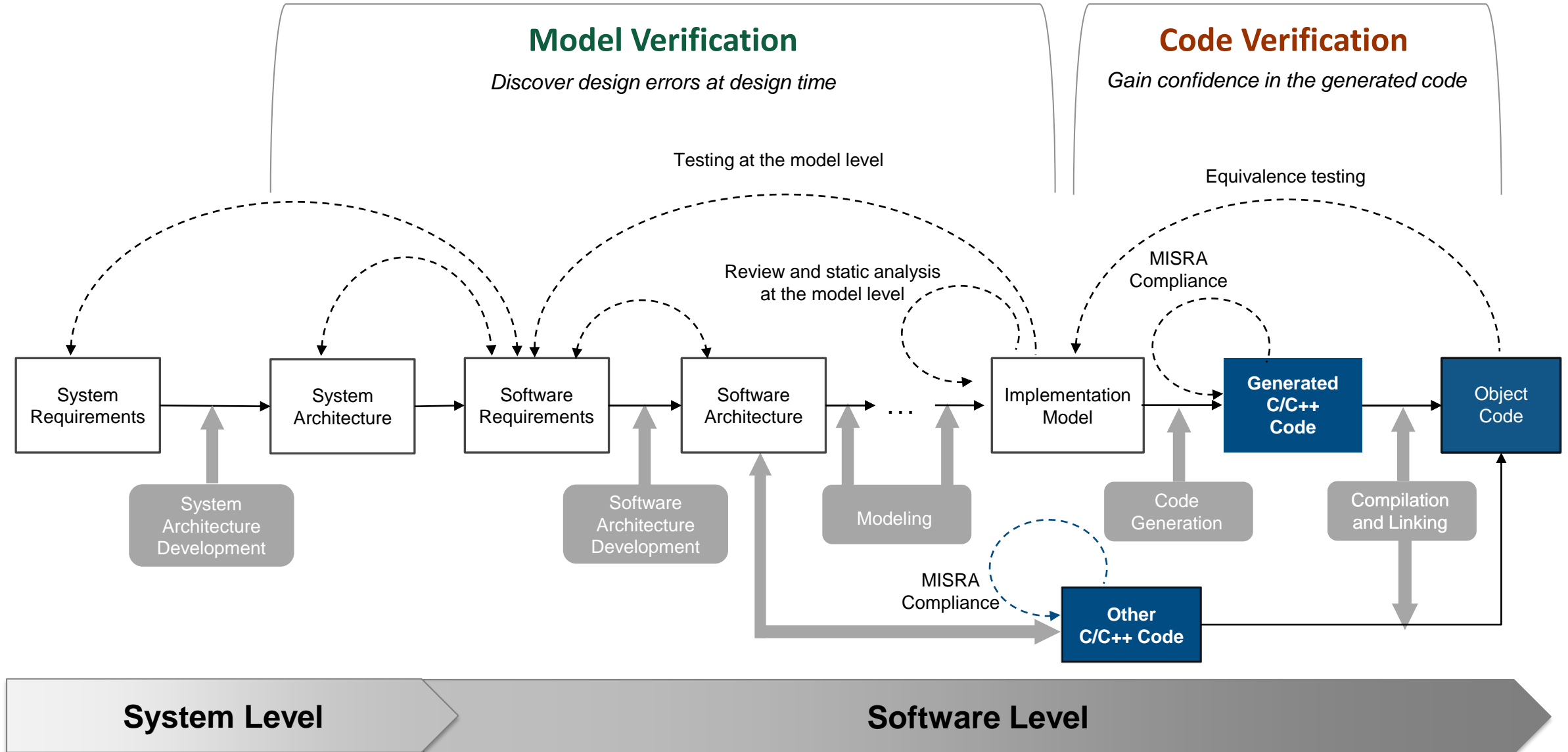
Reference Workflow for Certification – Dynamic Code Verification



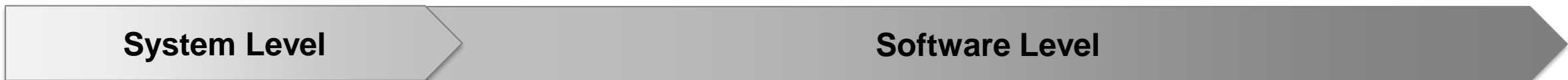
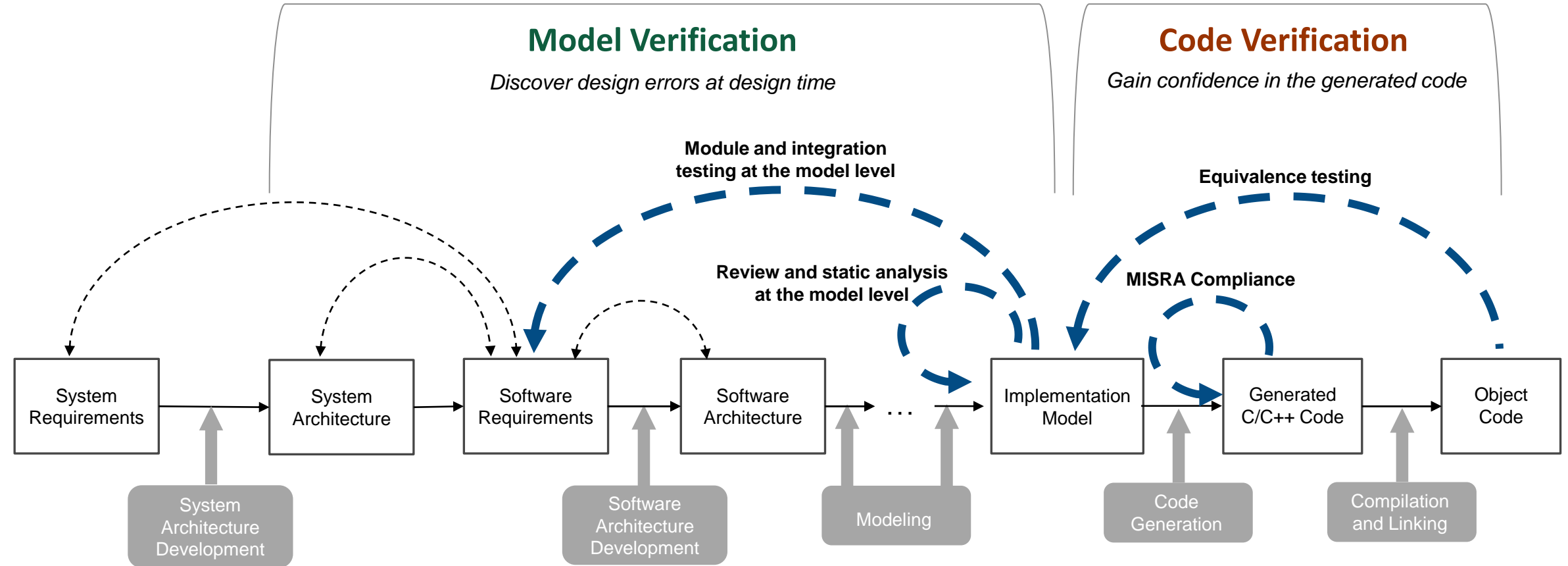
Reference Workflow for Certification – Atomic Components



Reference Workflow for Certification – Static Code Verification



Reference Workflow for Certification – Integration and Verification



ROI Results



Continental

Verification time cut by up to 50 percent



Leonardo

Recertification cycle times reduced by more than 90%



Tessella

Models reused on follow-on projects, cutting design effort by up to 80%



Corindus

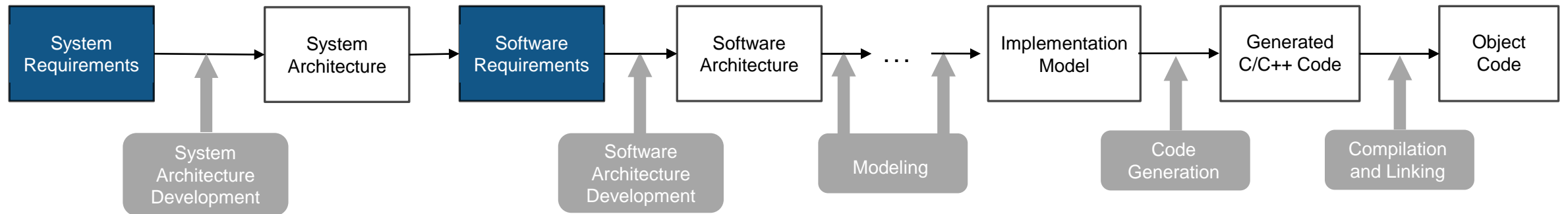
Development time halved and engineering effort reduced by 80%

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

Poll: What is your biggest challenge in verifying or testing your design? (choose the most important one)

- Tracing requirements to design and test
- Finding requirements errors late in development
- Meeting test coverage goals
- Reusing tests across design phases
- Performing system level validation
- Creating documentation and work products
- Performing manual reviews
- Other (Enter in Chat)

Requirements

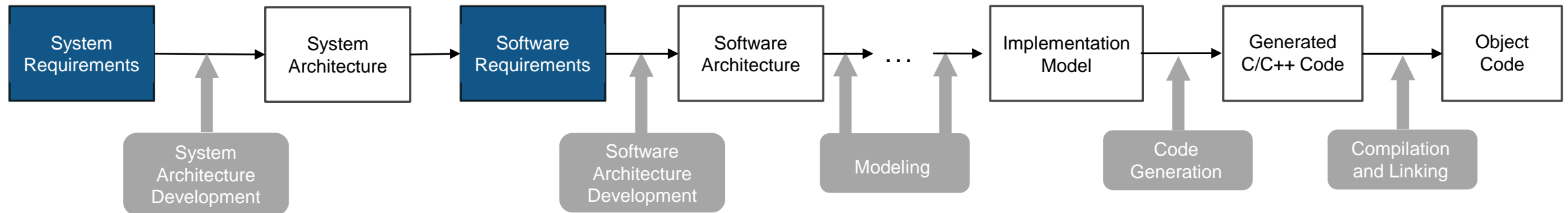


System Level

Software Level

Requirement span multiple levels across workflow

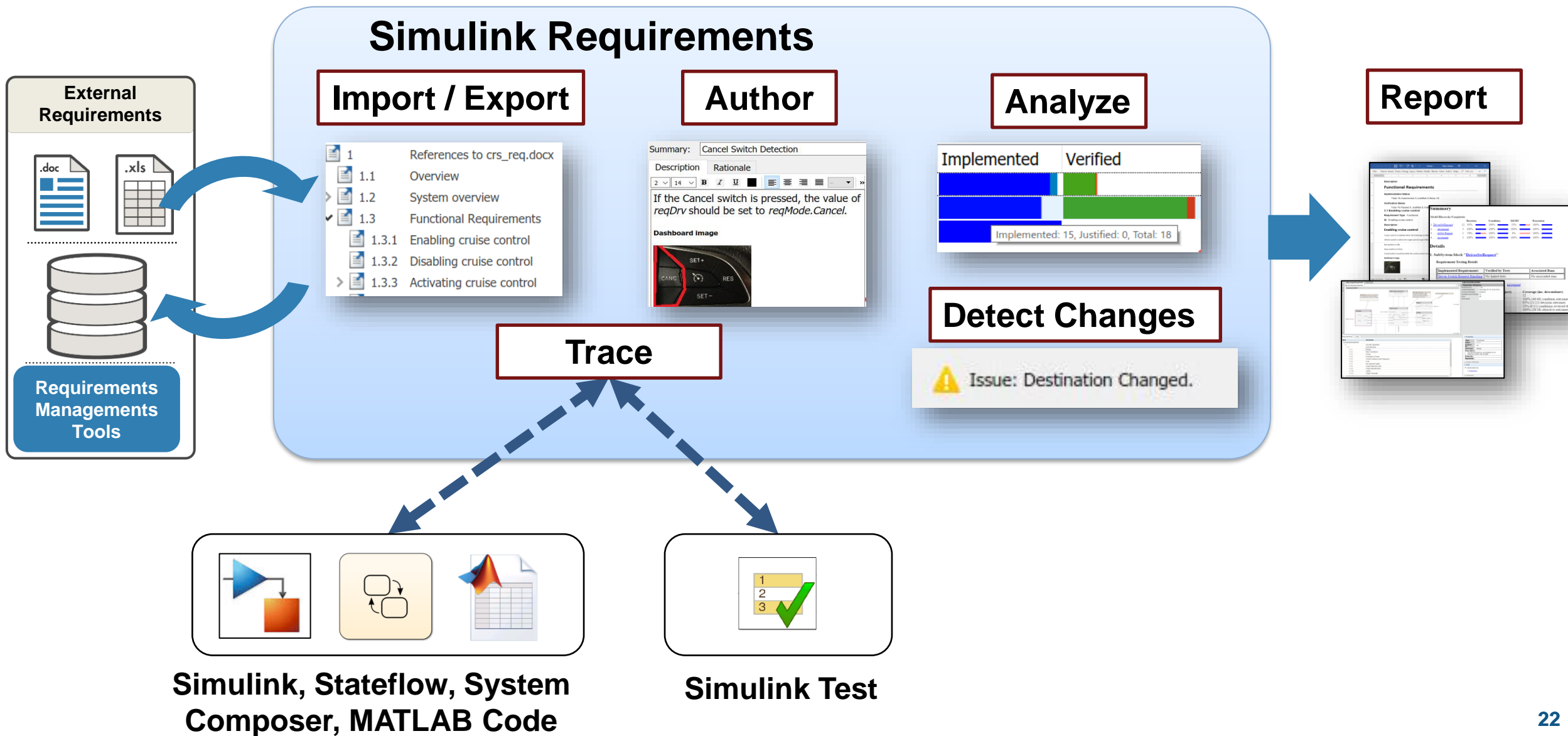
- Are all requirements implemented?
- Is each requirement tested?
- What artifacts are impacted by a change?



System Level

Software Level

Work with Requirements, Architecture and Design Together



Review and Analyze Traceability with Traceability Matrix

The screenshot displays the Traceability Matrix tool interface. At the top, there is a ribbon with tabs for HOME, ARTIFACTS, LINKS, VIEW, and SHARE. The HOME tab is active, showing various actions like Add, Configure Matrix, Highlight Missing Links, Create Link, Remove Links, Clear Change Issue, Update, Scope, Expand All, Collapse All, and Export. Below the ribbon is a Filter Panel with a tab for 'Simulink Requirements vs Simulink Test'. The main area shows a traceability matrix with requirements on the left and test cases on the top. The matrix cells contain arrows indicating the direction of traceability (e.g., left-pointing arrows for requirements to tests, right-pointing arrows for tests to requirements) and blue circles indicating the status of the links.

Requirement	db_DriverSwRequest_Tests	Unit test for DriverSwRequest	Enable button	Cancel button	Set button	Resume button	Increment button short	Increment button hold	Decrement button short	Decrement button hold
CC-REQ-1 Driver Switch Request Handling										
CC-REQ-2 Switch precedence	←									
CC-REQ-3 Avoid repeating commands	←									
CC-REQ-4 Long Switch recognition							○		○	
CC-REQ-7 Cancel Switch Detection				←						
CC-REQ-8 Set Switch Detection										
CC-REQ-9 Enable Switch Detection			←							
CC-REQ-10 Resume Switch Detection						←				
CC-REQ-11 Increment Switch Detection							←	○		
CC-REQ-15 Decrement Switch Detection									←	○

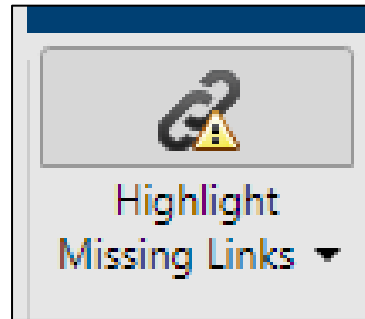
Review and Analyze Traceability with Traceability Matrix

- **Filters to focus view**

The screenshot displays the Traceability Matrix application window. The interface includes a ribbon with tabs for HOME, ARTIFACTS, LINKS, VIEW, and SHARE. The HOME tab is active, showing options like Add, Configure Matrix, Highlight Missing Links, Create Link, Remove Links, Update, Scope, Expand All, Collapse All, and Export. A Filter Panel is open on the left, highlighting the 'Type' filter section. The main area shows a traceability matrix for 'Simulink Requirements vs Simulink Test'. The matrix has columns for requirements (e.g., CC-REQ-1 Driver Switch Request Handling) and test cases (e.g., db_DriverSwRequest_Tests, Unit test for DriverSwRequest, Enable button, Cancel button, Set button, Resume button, Increment button short, Increment button hold, Decrement button short, Decrement button hold). The matrix cells contain arrows and circles indicating traceability relationships.

Requirement	db_DriverSwRequest_Tests	Unit test for DriverSwRequest	Enable button	Cancel button	Set button	Resume button	Increment button short	Increment button hold	Decrement button short	Decrement button hold
CC-REQ-1 Driver Switch Request Handling										
CC-REQ-2 Switch precedence		←								
CC-REQ-3 Avoid repeating commands		←								
CC-REQ-4 Long Switch recognition							○		○	
CC-REQ-7 Cancel Switch Detection				←						
CC-REQ-8 Set Switch Detection										
CC-REQ-9 Enable Switch Detection			←							
CC-REQ-10 Resume Switch Detection						←				
CC-REQ-11 Increment Switch Detection							←	○		
CC-REQ-15 Decrement Switch Detection									←	○

Review and Analyze Traceability with Traceability Matrix



The screenshot shows the Traceability Matrix application window. The toolbar at the top includes buttons for 'Add', 'Configure Matrix', 'Highlight Missing Links' (circled in red), 'Create Link', 'Remove Links', 'Update', 'Scope', 'Expand All', 'Collapse All', and 'Export'. The left sidebar contains filters for 'Type' (Test Assessment, Link, Tags), 'Change Tracking', and 'Cell'. The main area displays a traceability matrix for 'Simulink Requirements vs Simulink Test'.

Requirement	db_DriverSwRequest_Tests	Unit test for DriverSwRequest	Enable button	Cancel button	Set button	Resume button	Increment button short	Increment button hold	Decrement button short	Decrement button hold
CC-REQ-1 Driver Switch Request Handling										
CC-REQ-2 Switch precedence		←								
CC-REQ-3 Avoid repeating commands		←								
CC-REQ-4 Long Switch recognition							○	○		
CC-REQ-7 Cancel Switch Detection			←							
CC-REQ-8 Set Switch Detection					■					
CC-REQ-9 Enable Switch Detection			←							
CC-REQ-10 Resume Switch Detection						←				
CC-REQ-11 Increment Switch Detection							←	○		
CC-REQ-15 Decrement Switch Detection									←	○

Review and Analyze Traceability with Traceability Matrix

The screenshot displays the Traceability Matrix interface. The main window shows a traceability matrix for 'Simulink Requirements vs Simulink Test'. A 'Create Link' dialog box is open, showing the source 'Set button (db_DriverSwRequest_Tests.mldatx)' and destination 'CC-REQ-8 Set Switch Detection (DOORS_crs_req_func_spec.slrqx)' with a 'Verifies' type. A red arrow points from the 'Create' button in the dialog to a cell in the matrix. A tooltip for the selected cell shows the source and destination names.

Source	Destination	Link
CC-REQ-8 Set Switch Detection	Set button	None (Create)

- **Create links to address gaps**

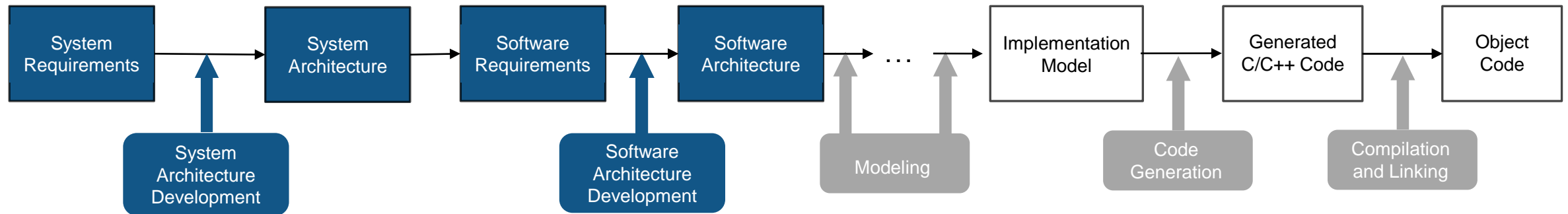
Review and Analyze Traceability with Traceability Matrix

The screenshot shows the Traceability Matrix tool interface. The main window displays a traceability matrix for "Simulink Requirements vs Simulink Test". The matrix lists requirements on the left and test cases on the top. A red circle highlights a blue arrow icon in the matrix, indicating a missing link.

Requirement	db_DriverSwRequest_Tests	Unit test for DriverSwRequest	Enable button	Cancel button	Set button	Resume button	Increment button short	Increment button hold	Decrement button short	Decrement button hold
CC-REQ-1 Driver Switch Request Handling										
CC-REQ-2 Switch precedence		←								
CC-REQ-3 Avoid repeating commands		←								
CC-REQ-4 Long Switch recognition								○	○	
CC-REQ-7 Cancel Switch Detection		←								
CC-REQ-8 Set Switch Detection					←					
CC-REQ-9 Enable Switch Detection		←								
CC-REQ-10 Resume Switch Detection						←				
CC-REQ-11 Increment Switch Detection							←	○		
CC-REQ-15 Decrement Switch Detection									←	○

- **Create links to address gaps**

System and Software Architecture



System Level

Software Level

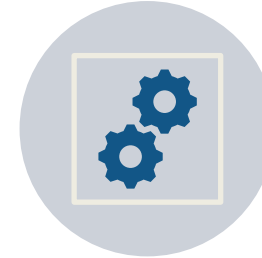
Why is architecture important?



AUTHOR AND VALIDATE
REQUIREMENTS EARLIER



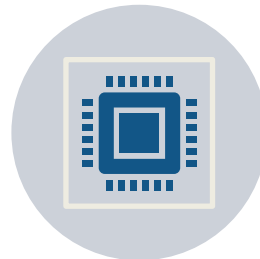
APPORTION THE SAFETY
OBJECTIVES



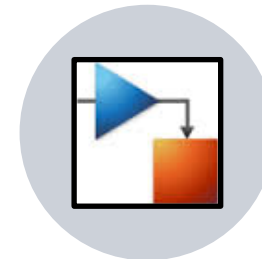
COORDINATE SOFTWARE
AND HARDWARE
INTERFACES



SHARE REQUIREMENTS
WITHOUT AMBIGUITY

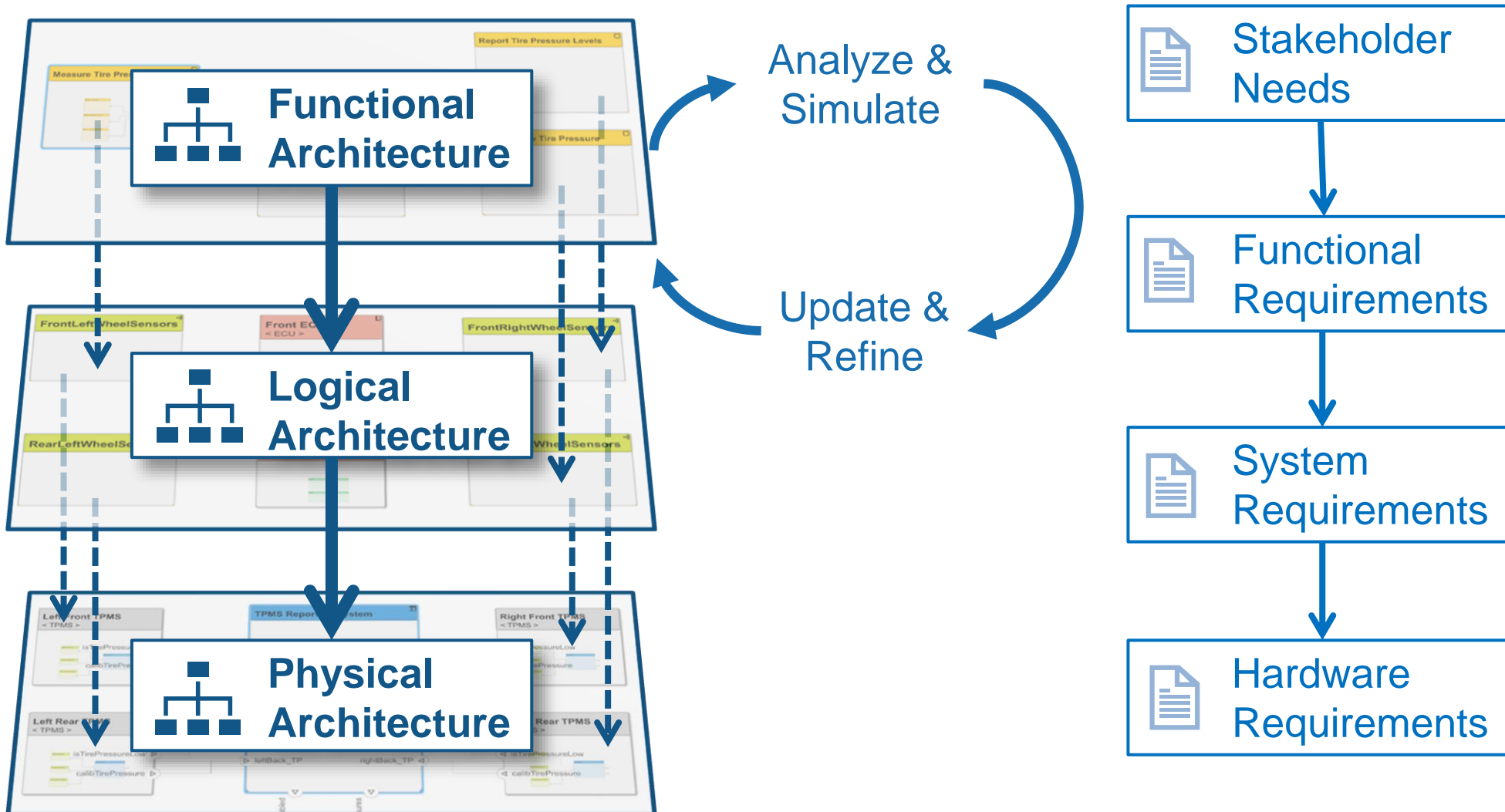


DETERMINE ALLOCATION
TO SOFTWARE OR
HARDWARE

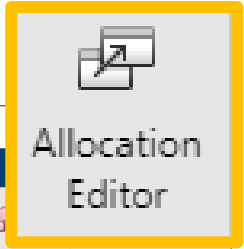


LINK TO BEHAVIORAL
MODELS

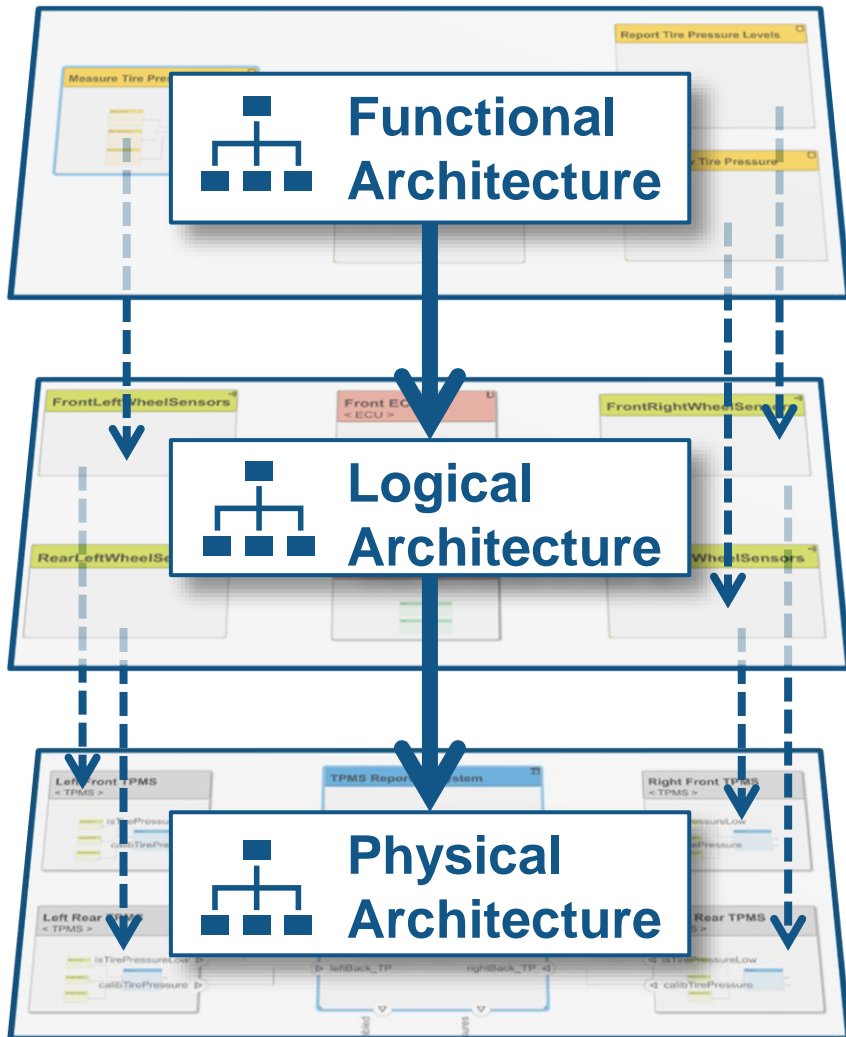
Describe systems using multiple architecture models to organize and check the completeness of the requirements



Allocate elements from one model to another to create a traceable & analyzable link



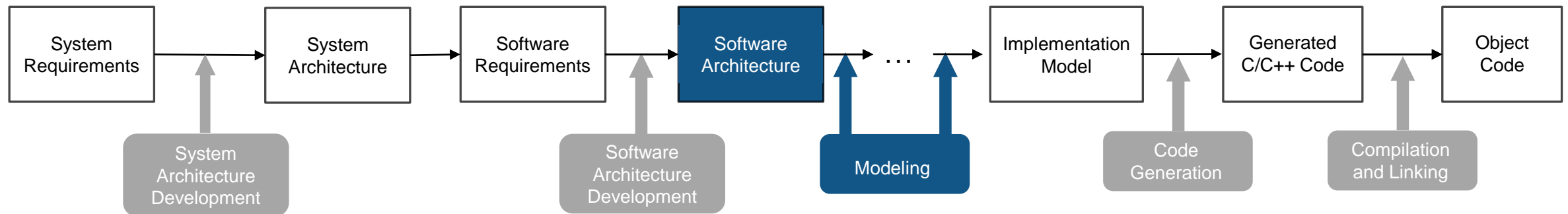
Allocation Editor



The screenshot shows the Allocation Editor interface. On the left is the Allocation Set Browser showing a tree structure for 'Scenario 1' under 'Functional2Logical', 'FunctionalAllocation', and 'SoftwareDeployment'. The main area is a grid for 'Scenario 1' with columns for various components and rows for functions. The 'Calculate Tire Pressure' row is highlighted. On the right is the Allocation Properties panel, which shows details for the selected allocation, including its name, source, function, ASIL, complexity, target, and SWComponent.

Name	Value
Allocated	<input checked="" type="checkbox"/>
Source	Component
Main	
Name	Calculate Tire Pressure
Function	
ASIL	ASIL_B
Complexity	
Target	Component
Main	
Name	Controller
SWComponent	
BinarySize	10 MB
Supplier	'Supplier B'

Reference Workflow for Certification – Detailed Design



System Level

Software Level

Reference Workflow for Certification – Detailed Design

Requirement: SW_HLF#6

Details

Properties

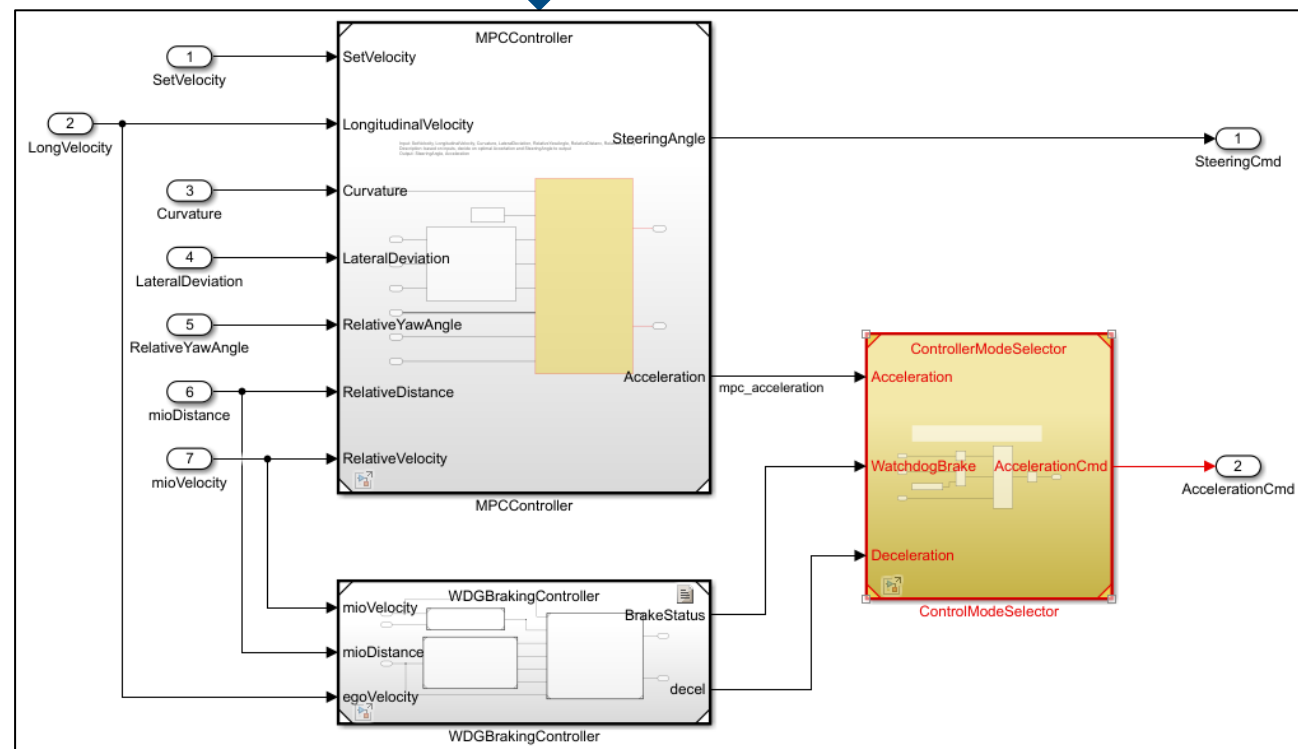
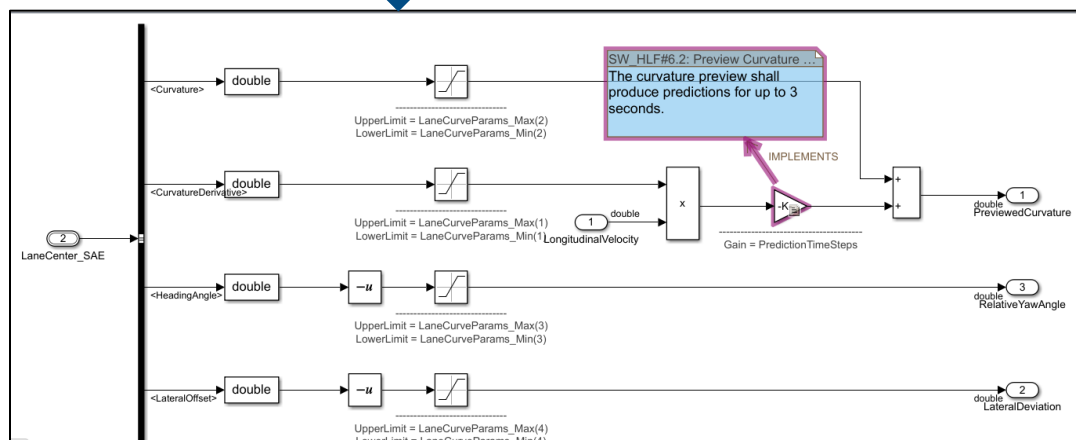
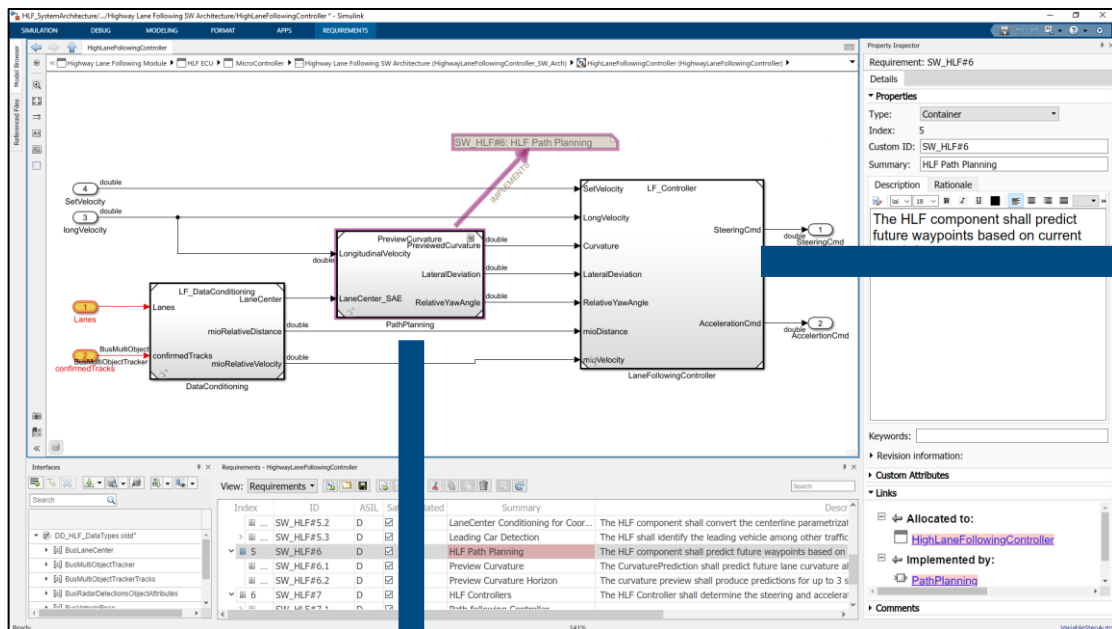
Type: Container
 Index: 5
 Custom ID: SW_HLF#6
 Summary: HLF Path Planning

Description

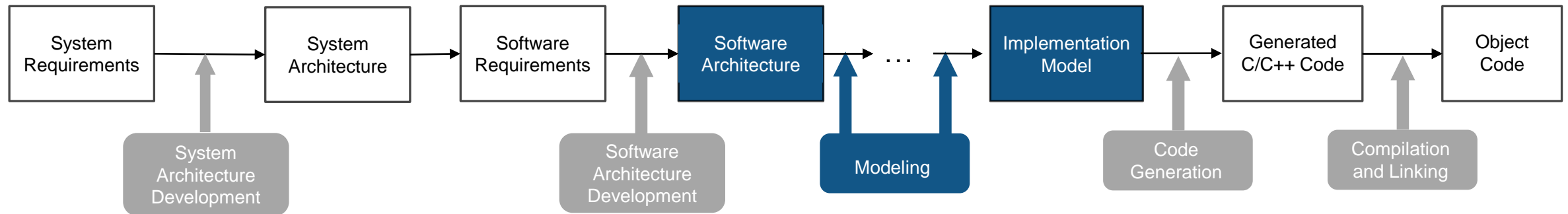
The HLF component shall predict future waypoints based on current lane information.

Index	ID	ASIL	Safety-related	Summary	Descr
...	SW_HLF#5.2	D	☑	LaneCenter Conditioning for Coord...	The HLF component shall convert the centerline parametrizat
...	SW_HLF#5.3	D	☑	Leading Car Detection	The HLF shall identify the leading vehicle among other traffic
5	SW_HLF#6	D	☑	HLF Path Planning	The HLF component shall predict future waypoints based on
...	SW_HLF#6.1	D	☑	Preview Curvature	The CurvaturePrediction shall predict future lane curvature al
...	SW_HLF#6.2	D	☑	Preview Curvature Horizon	The curvature preview shall produce predictions for up to 3 s
6	SW_HLF#7	D	☑	HLF Controllers	The HLF Controller shall determine the steering and accelera
...	SW_HLF#7.1	D	☑	Path following Controller	

Reference Workflow for Certification – Detailed Design



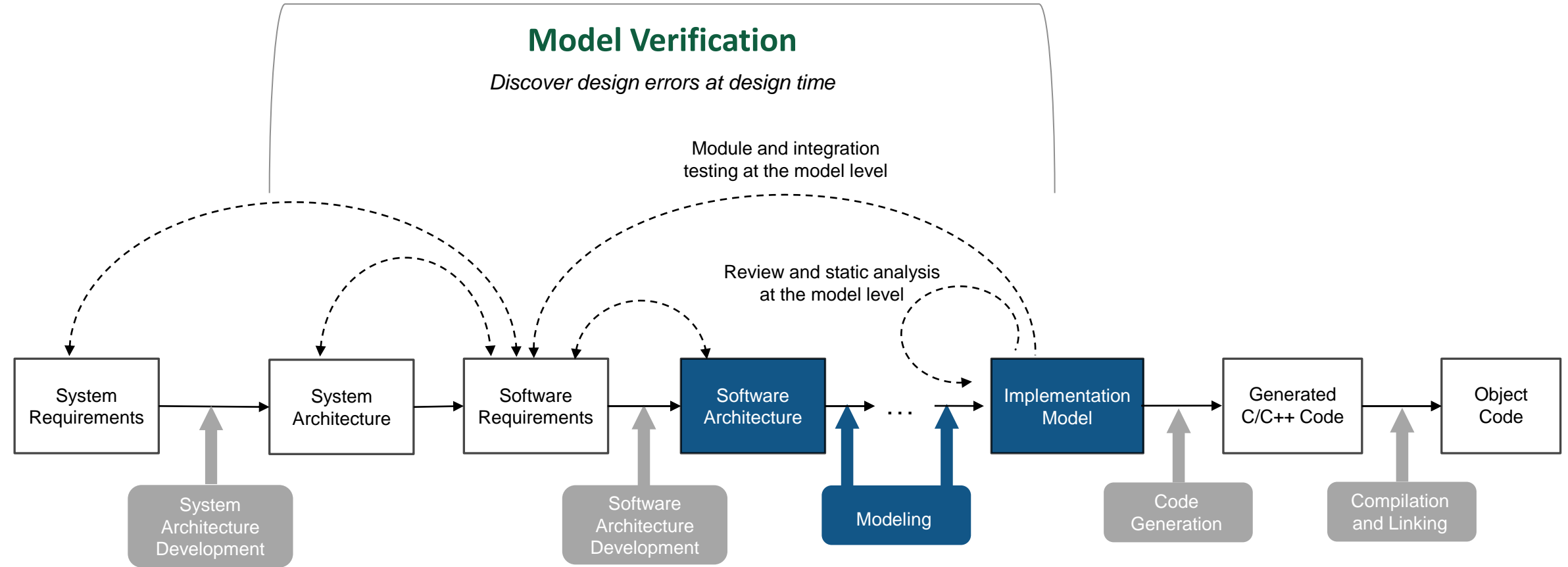
Reference Workflow for Certification – Detailed Design



System Level

Software Level

Reference Workflow for Certification – Model Verification



Verify Model Complies with Guidelines and Standards

Model Advisor Analysis

Static analysis checks for:

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

The screenshot shows the Model Advisor tool interface. The left pane displays a tree view of checks, with 'Check for blocks not recommended for C/C++ production code deployment' selected. The right pane shows the analysis results for this check, including a warning and a table of affected blocks.

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

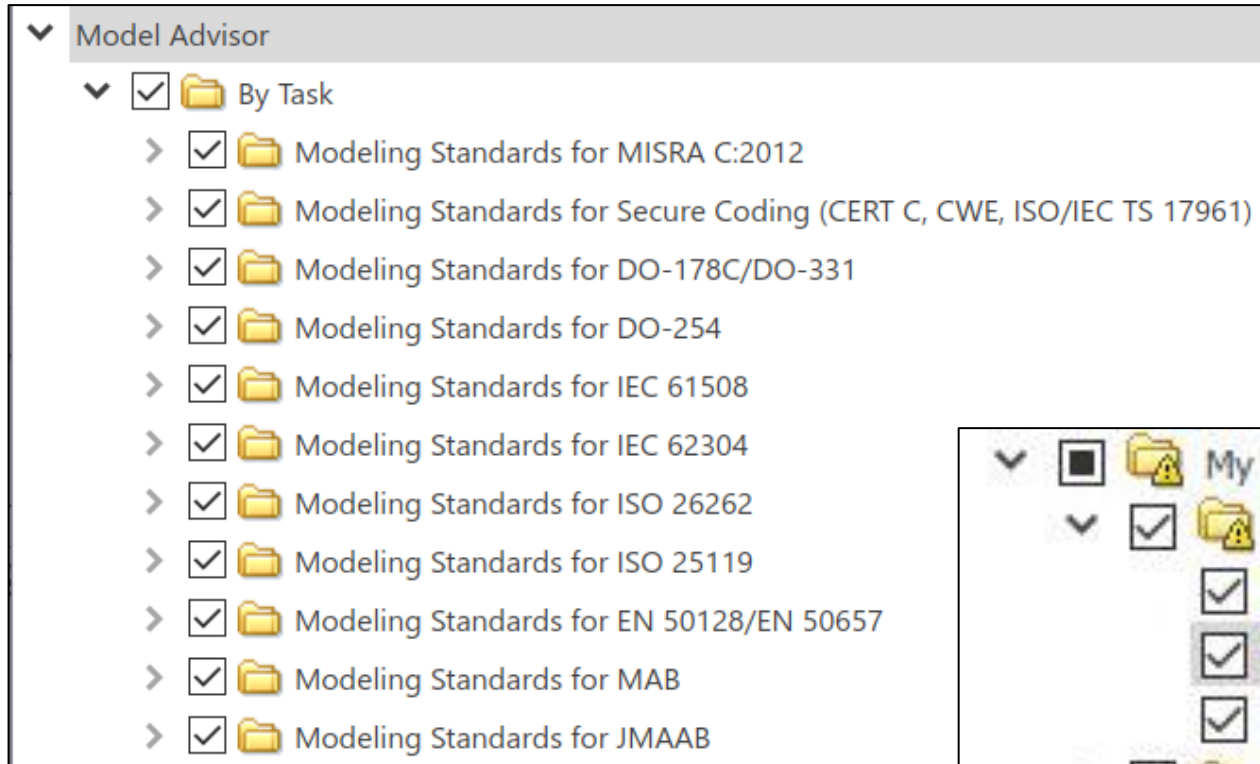
The following blocks are not supported or not recommended for C/C++ production code deployment:

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

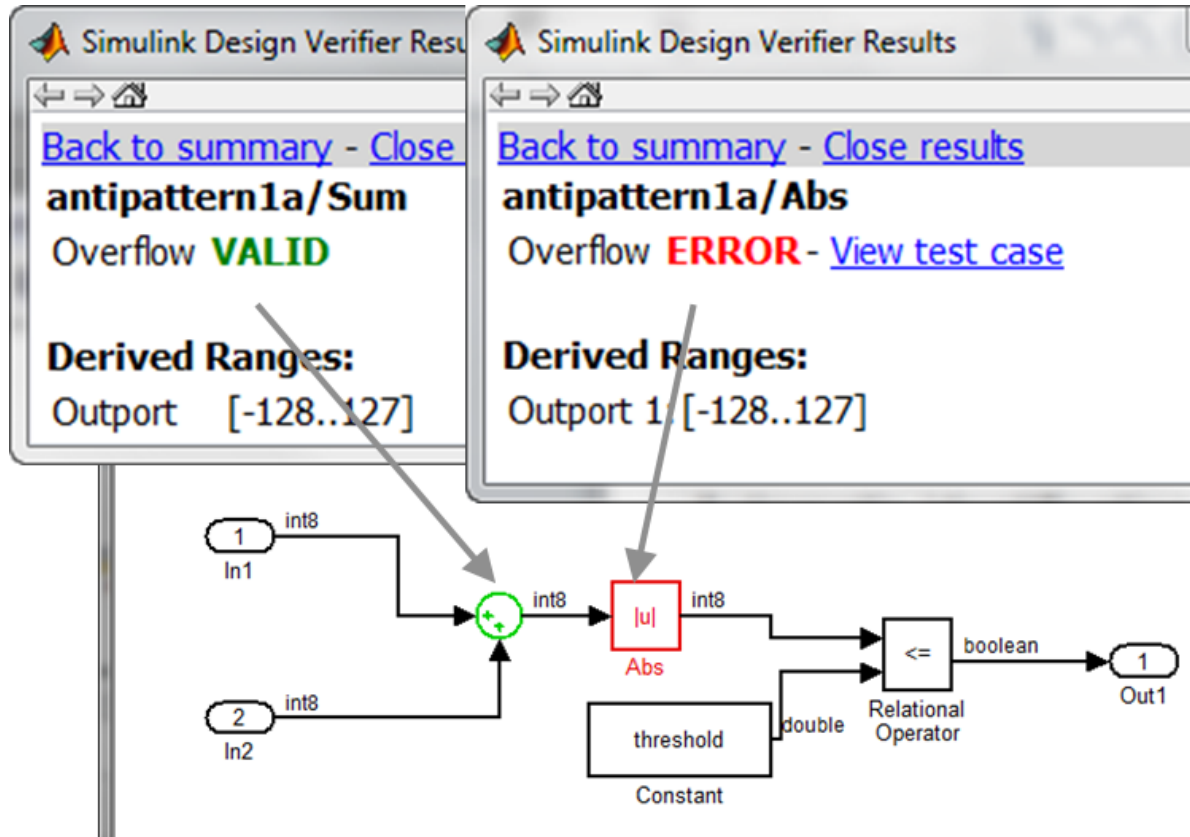
Recommended Action

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.

Built in checks for industry standards and guidelines

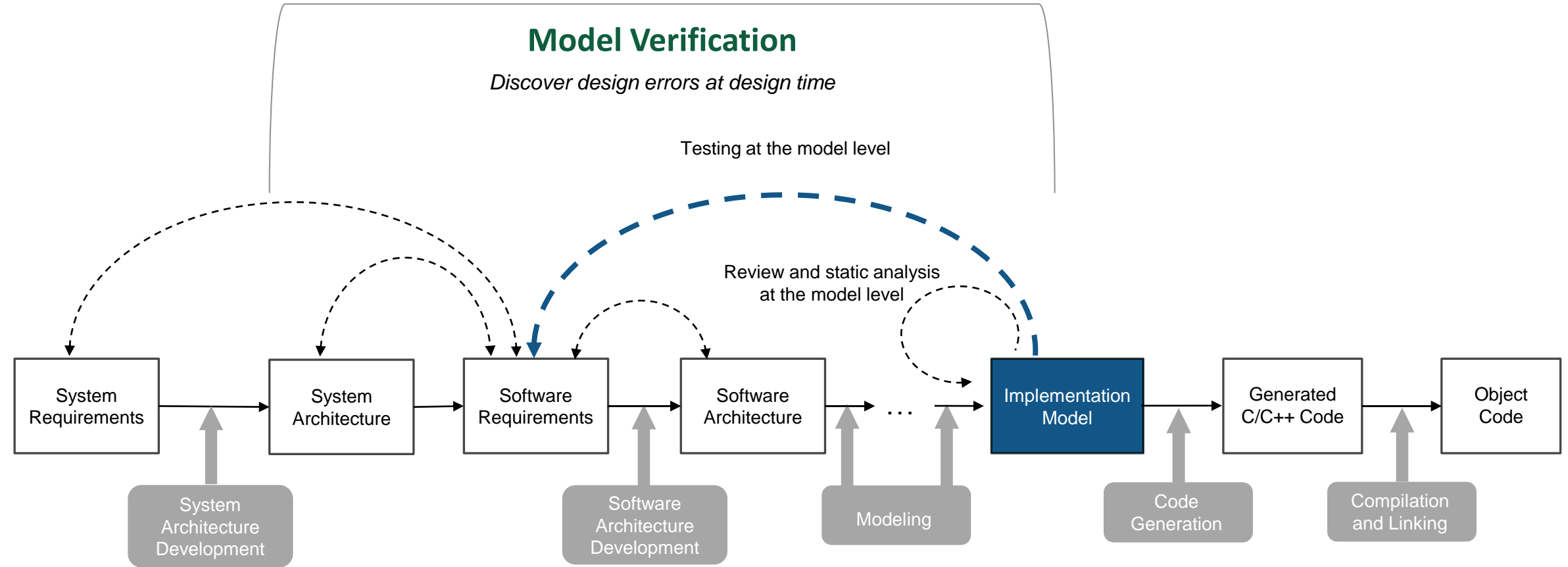


Detect Design Errors Using Formal Methods



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

Model-Based Design Reference Workflow



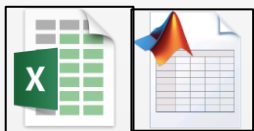
Systematic Functional Testing with Simulink Test

Test Case

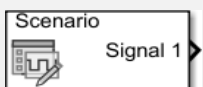


Model Sim through SIL, PIL and HIL
Scale with Parallel Computing Toolbox and Continuous Integration

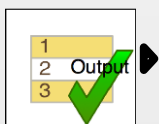
Inputs



Data file (input)



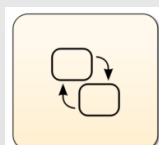
Signal Editor



Test Sequence

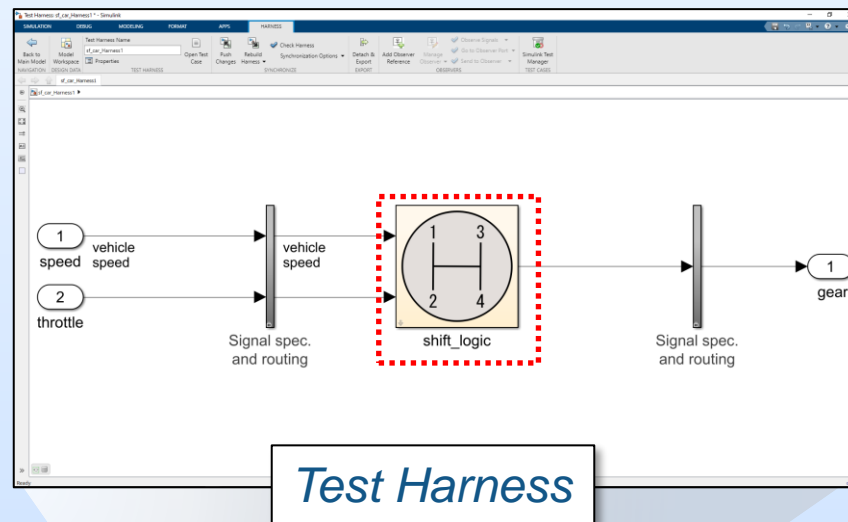
```
classdef BaselineTest < sltest.TestCase
    methods (Test)
        function testOne(testCase)
            simOut = testCase.simulate('TestExample');
            testCase.verifySignalsMatch(simOut, 'baseline');
        end
        function testTwo(testCase)
        end
    end
end
```

MATLAB Code

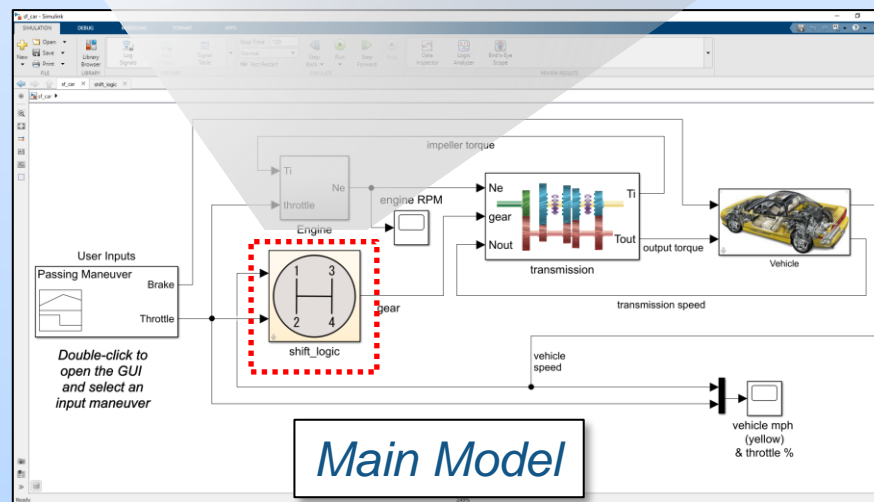


Stateflow

and more!

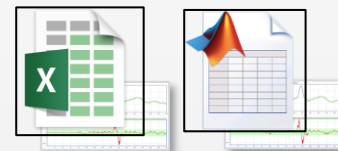


Test Harness

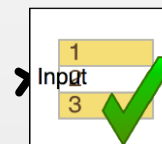


Main Model

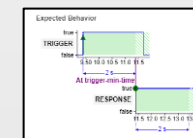
Assessments



Data file baseline



Test Assessment



Temporal Assessment

```
function customCriteria
Perform custom criteria
test.verifyThat(testCase, ...
```

MATLAB Code

and more!

Manage Testing and Test Results

Test Manager

TESTS

Filter tests by name or tags, e.g. tags: test

- Scenario
 - ACC_ISO_TargetDiscriminationTest
 - ACC_ISO_AutoRetargetTest
 - ACC_ISO_CurveTest
 - ACC_StopGo
 - LFACC_DoubleCurve_DecelTarget
 - LFACC_DoubleCurve_AutoRetarget
 - LFACC_DoubleCurve_StopGo
 - LFACC_Curve_CutInCutOut
 - LFACC_Curve_CutInCutOut_TooClose

ACC_ISO_CurveTest

Baseline Test

DESCRIPTION*

REQUIREMENTS*

SYSTEM UNDER TEST*

PARAMETER OVERRIDES*

CALLBACKS*

INPUTS*

SIMULATION OUTPUTS*

BASELINE CRITERIA*

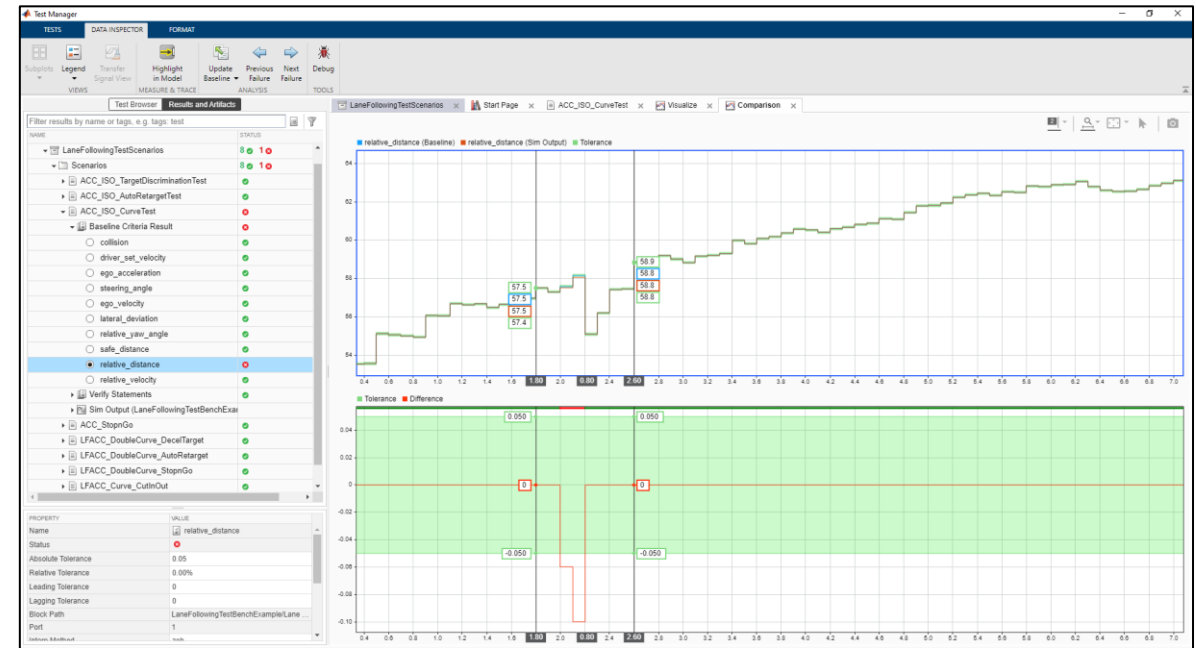
SIGNAL NAME	SHEETS	ABS TOL	REL TOL	LEADING TOL	LAGGING TOL
baseline1	baseline1	0	0.00%	0	0
collision		0	0.00%	0	0
driver_set_velocity		0	0.00%	0	0
ego_acceleration		0.2	0.00%	0	0
steering_angle		0	0.00%	0	0
ego_velocity		0.25	0.00%	0	0
lateral_deviation		0	0.00%	0	0

LOGICAL AND TEMPORAL ASSESSMENTS*

ASSESSMENT CALLBACK

EN	NAME	ASSESSMENT	REQUIREMENTS	VISUAL REPRESENTATION
✓	CheckLateralDeviat.	At any point of time	None	

Test Browser

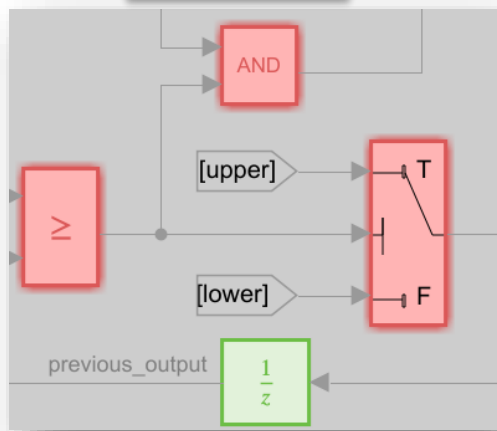


Test Results



Coverage Analysis to Measure Testing

Simulink



Stateflow



Code

Coverage annotation

Links to model element

```

Code
rtwdemo_sil_topmodel.c
99 /* Output and update for enable system: '<Root>/CounterTypeB' */
100 static void CounterTypeB(void)
101 {
102     /* Outputs for Enabled SubSystem: '<Root>/CounterTypeB' incorporates:
103      * EnablePort: '<S2>/Enable'
104      */
105     if (enable) {
106         /* Set ticks to count when incrementing:
107          * Decision covered false, but not true
108          * Inport: '<Root>/reset'
109          * Inport: '<Root>/ticks_to_count'
110          * Output: '<Root>/count_b'
111          * Sum: '<S2>/Add'
112          */
113         if (rtu.reset) {
114             rtv.count_b = 0;

```

Tooltip with code coverage results

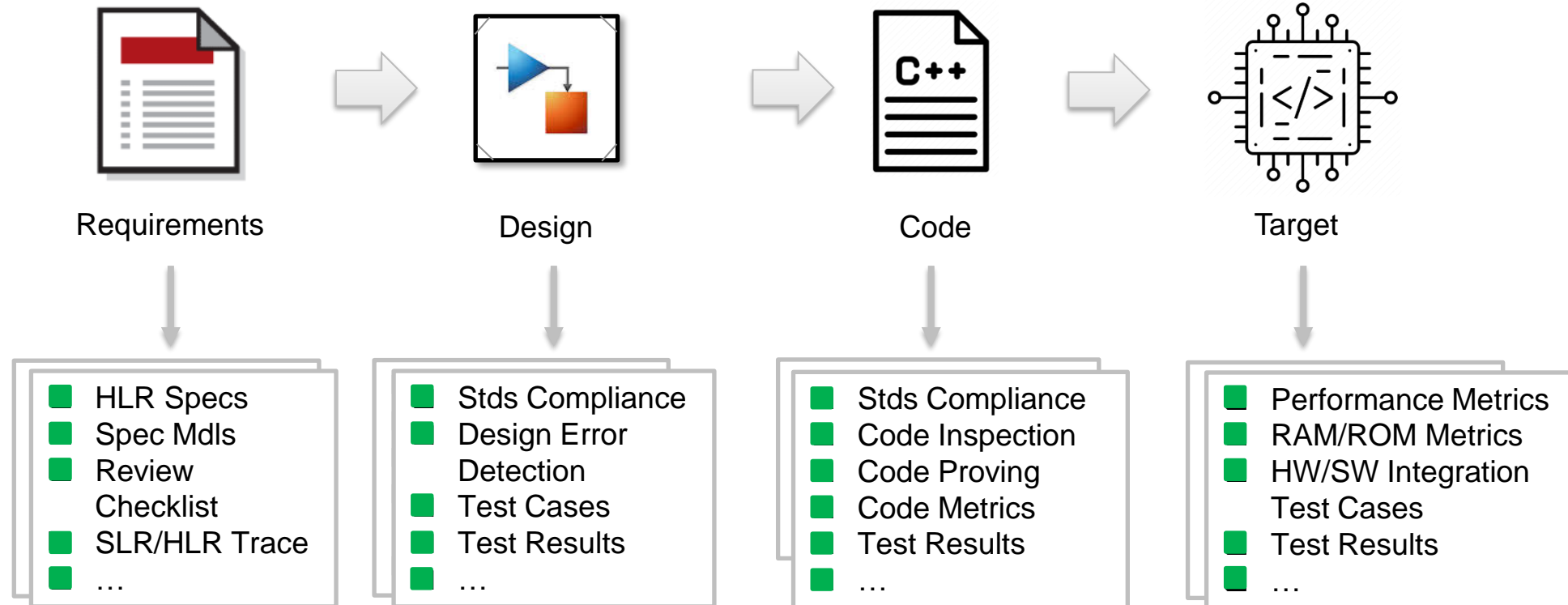
- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors

Coverage Reports

Summary

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

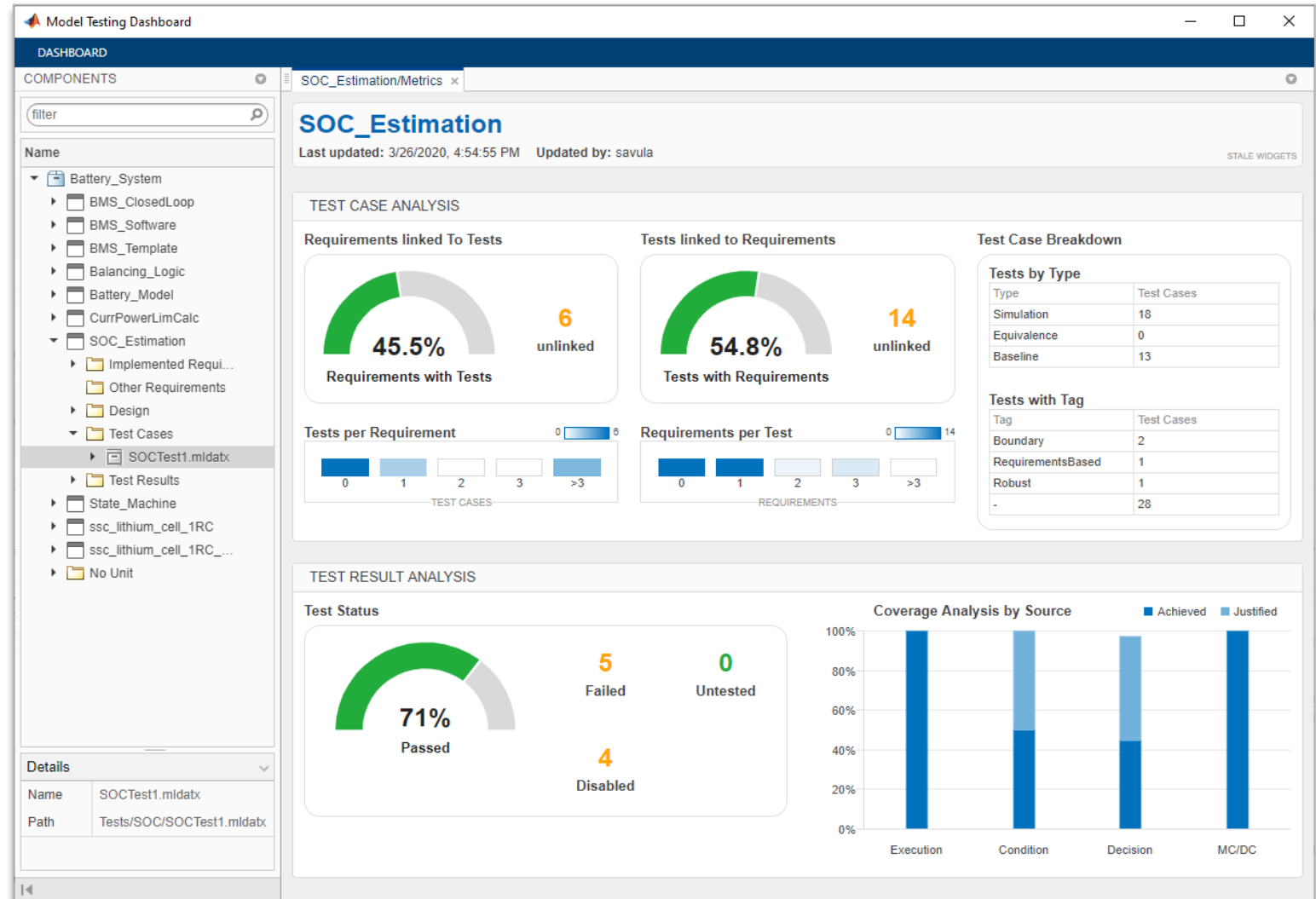
Challenge: Managing the many activities and artifacts needed to meet certification standards



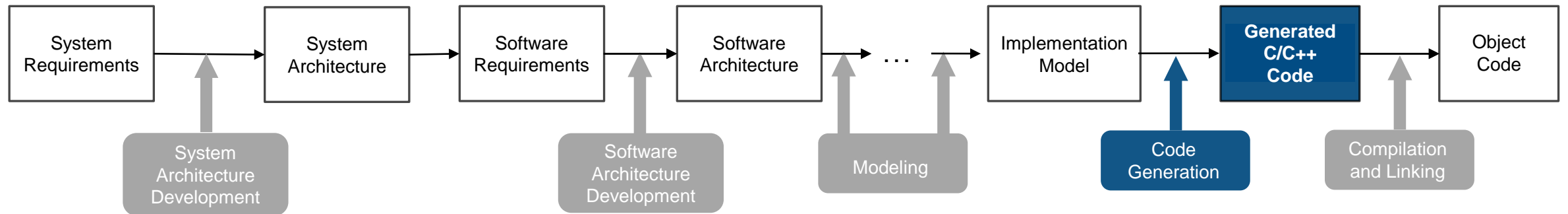
Model Testing Dashboard: Manage progress, completeness and quality of requirements-based testing

R2020b

- Central view that summarizes testing data and status
- Identify gaps and respond faster to requirements changes
- View results from a continuous integration system



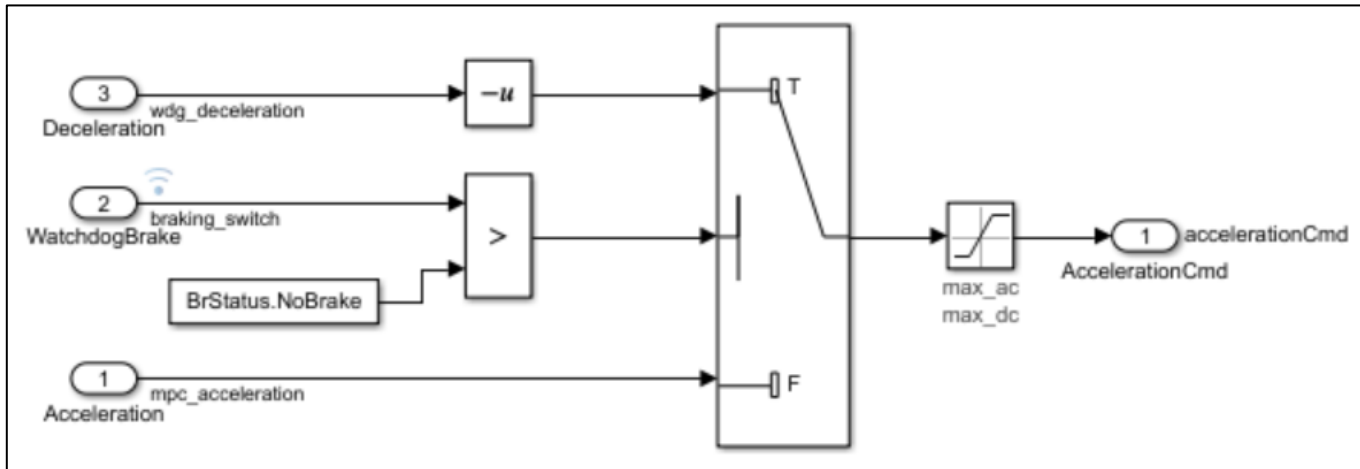
Code Generation



System Level

Software Level

Automatically generate production-quality code that behaves the same way as the model you created in Simulink

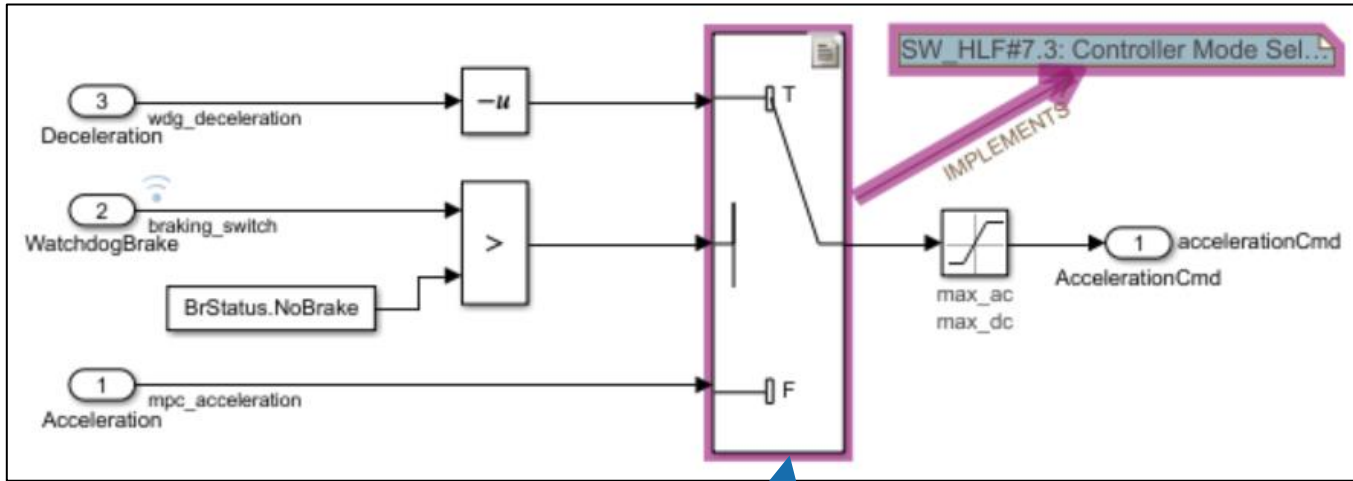


```

27  /* Switch: '<Root>/Switch' incorporates:
28     * Constant: '<Root>/Constant'
29     * RelationalOperator: '<Root>/GreaterThan'
30     *
31     * Block requirements for '<Root>/Switch':
32     * 1. SW_HLF#7.3 Controller Mode Selection
33     */
34  if ((*rtu_WatchdogBrake) > NoBrake) {
35     /* UnaryMinus: '<Root>/UnaryMinus' */
36     rtb_UnaryMinus = -(*rtu_Deceleration);
37  } else {
38     rtb_UnaryMinus = *rtu_Acceleration;
39  }
40
41  /* End of Switch: '<Root>/Switch' */

```

Requirement Traceability included in Generated Code



```

27  /* Switch: '<Root>/Switch' incorporates:
28     * Constant: '<Root>/Constant'
29     * RelationalOperator: '<Root>/GreaterThan'
30     *
31     Block requirements for '<Root>/Switch':
32     1. SW_HLF#7.3 Controller Mode Selection
33     */
34  if ((*rtu_WatndogBrake) > NoBrake) {
35     /* UnaryMinus: '<Root>/UnaryMinus' */
36     rtb_UnaryMinus = -(*rtu_Deceleration);
37  } else {
38     rtb_UnaryMinus = *rtu_Acceleration;
39  }
40
41  /* End of Switch: '<Root>/Switch' */

```

Requirements - ControllerModeSelector

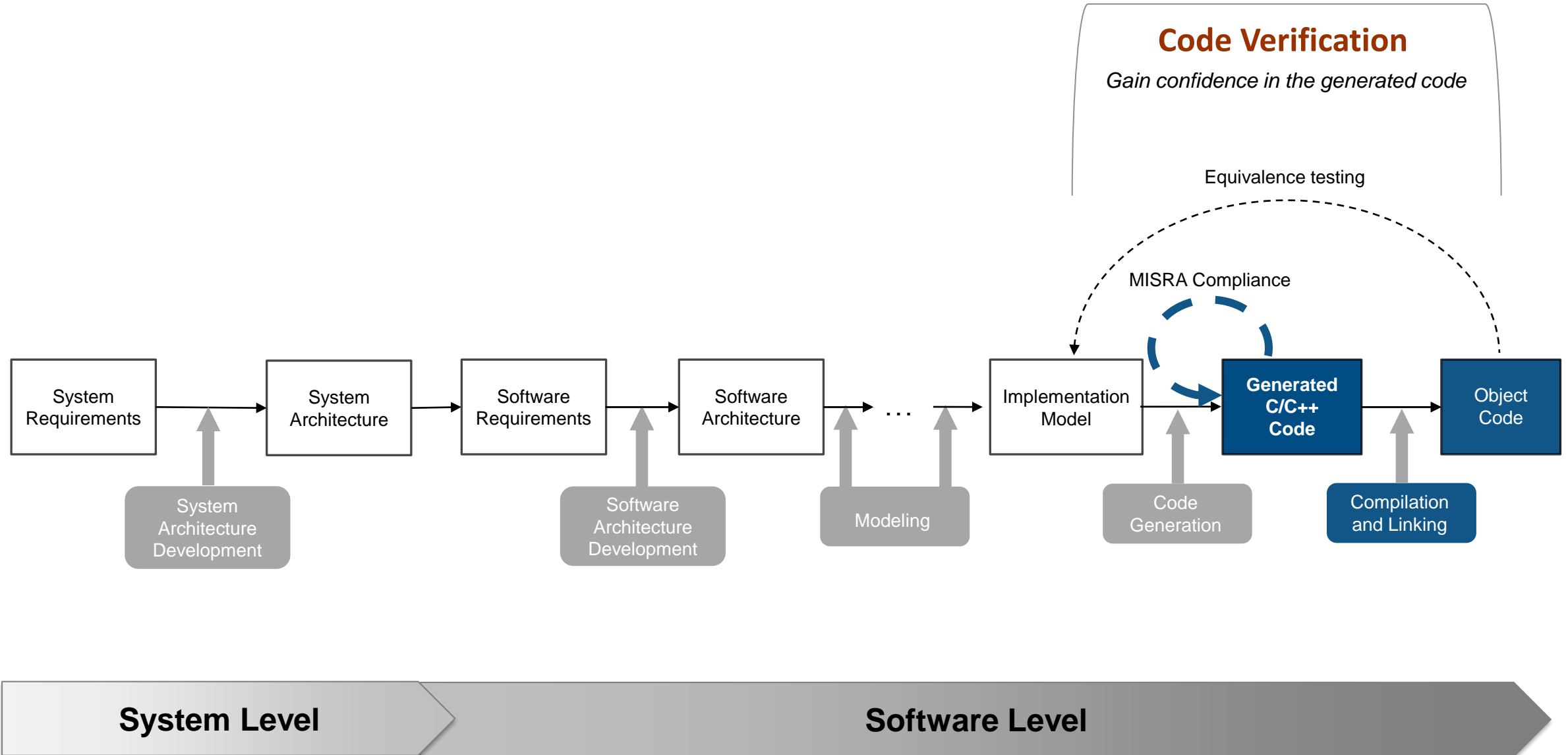
View: Requirements

Index	ID	Summary	ASIL	Safety-related
> 6.1	SW_HLF#7.1	Path following Controller	D	<input checked="" type="checkbox"/>
> 6.2	SW_HLF#7.2	Watchdog Controller	D	<input checked="" type="checkbox"/>
6.3	SW_HLF#7.3	Controller Mode Selection	D	<input checked="" type="checkbox"/>

▼ Links

- Implemented by:
 - Switch
- Coded at:
 - ControllerModeSelector.c:30,38,50,51,53,56
- TC1

Reference Workflow for Certification – Code Verification



Reference Workflow for Certification – Code Verification

Static Code Verification for MISRA Compliance

```

Web Browser - ControllerModeSelector Code Generation Report
ControllerModeSelector Code Generation Report
Location: file:///C:/Users/mabualqu/MATLAB/Projects/examples/iso26262CaseStudy/ISO_06_08_Software_Unit_Design_and_Implementation/WPs/ISO_06_08_5_2_Software_Unit_Implementation/

Contents
Summary
Subsystem Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Coder Assumptions

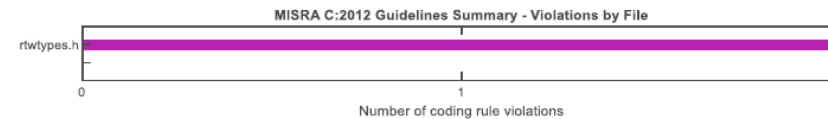
Generated Code
[-] Model files
ControllerModeSelector.c
ControllerModeSelector.h
ControllerModeSelector_private.h
ControllerModeSelector_types.h
[-] Shared files
BrStatus.h
rtwtypes.h

1 /*
2  * Prerelease License - for engineering feedback and testing purposes
3  * only. Not for sale.
4  *
5  * File: ControllerModeSelector.c
6  *
7  * Code generated for Simulink model 'ControllerModeSelector'.
8  *
9  * Model version           : 2.1
10 * Simulink Coder version  : 9.4 (R2021a) 14-Oct-2020
11 * C/C++ source code generated on : Wed Oct 28 04:51:06 2020
12 *
13 * Target selection: ert.tlc
14 * Embedded hardware selection: Intel->x86-64 (Windows64)
15 * Code generation objectives: Unspecified
16 * Validation result: Not run
17 */
18
19 #include "ControllerModeSelector.h"
20 #include "ControllerModeSelector_private.h"
21
22 /* Output and update for referenced model: 'ControllerModeSelector' */
23 void ControllerModeSelector(const real_T *rtu_Acceleration, const BrStatus
24 *rtu_WatchdogBrake, const real_T *rtu_Deceleration, real_T
25 *rtu_AccelerationCmd)
26 {
27     real_T rtb_UnaryMinus;
28     real_T y;
29
30     /* Switch: '<Root>/Switch' incorporates:
31      * Constant: '<Root>/Constant'
32      * RelationalOperator: '<Root>/GreaterThan'
33      */
34     if ((*rtu_WatchdogBrake) > NoBrake) {
35         /* UnaryMinus: '<Root>/UnaryMinus' */
36         rtb_UnaryMinus = -(*rtu_Deceleration);
37     } else {
38         rtb_UnaryMinus = *rtu_Acceleration;
39     }
40
41     /* End of Switch: '<Root>/Switch' */
42
43     /* Saturate: '<Root>/Saturation' */
44     if (rtb_UnaryMinus > 2.0) {
45         y = 2.0;
46     } else if (rtb_UnaryMinus < (-10.0)) {
47         y = (-10.0);
48     } else {
49         y = rtb_UnaryMinus;

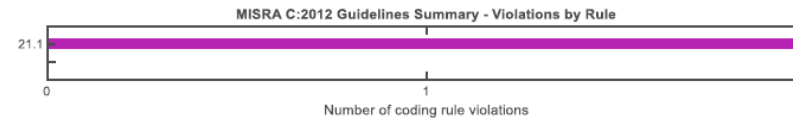
```

Chapter 1. MISRA C:2012 Guidelines

MISRA C:2012 Guidelines Summary - Violations by File



MISRA C:2012 Guidelines Summary - Violations by Rule



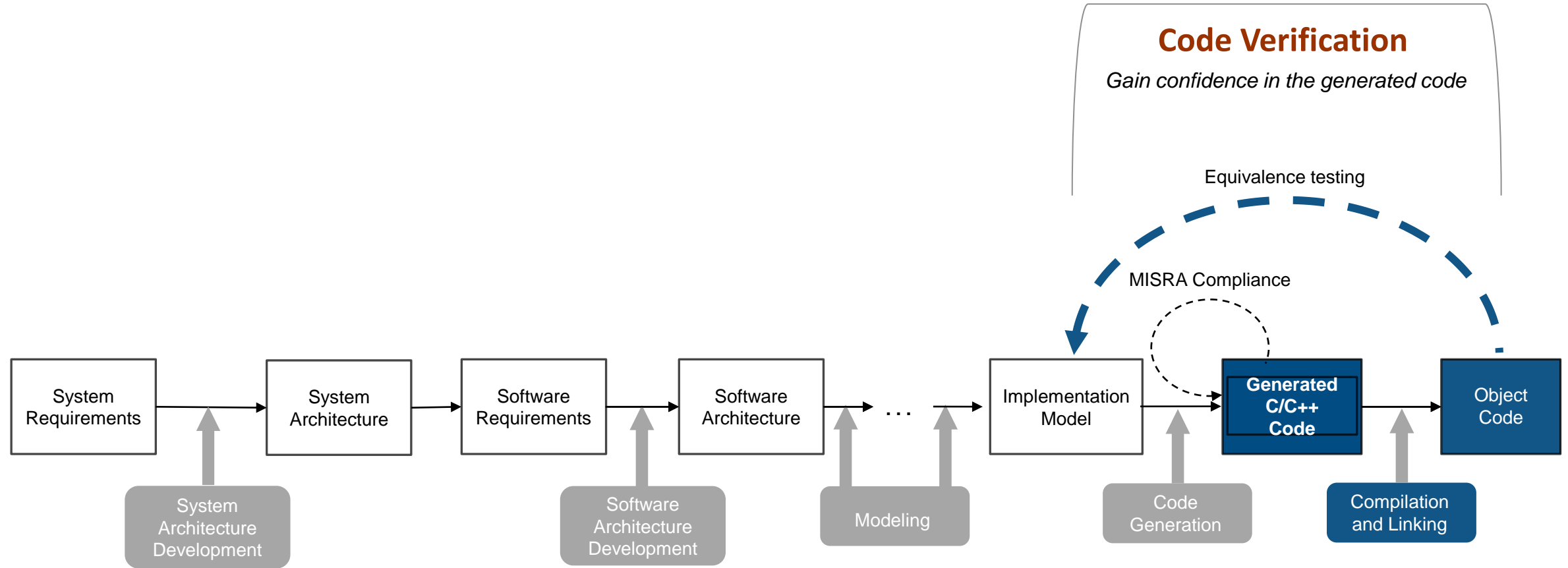
MISRA C:2012 Guidelines Summary for all Files

File	Total
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector.c	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector_private.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert\ControllerModeSelector\ControllerModeSelector_types.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert_sharedutils\BrStatus.h	0
C:\Users\mabualqu\MA\ISO_06_08_Software_Unit_Design_and_Implementation\WPs\ISO_06_08_5_2_Software_Unit_Implementation\slprj\ert_sharedutils\rtwtypes.h	2
Total	2

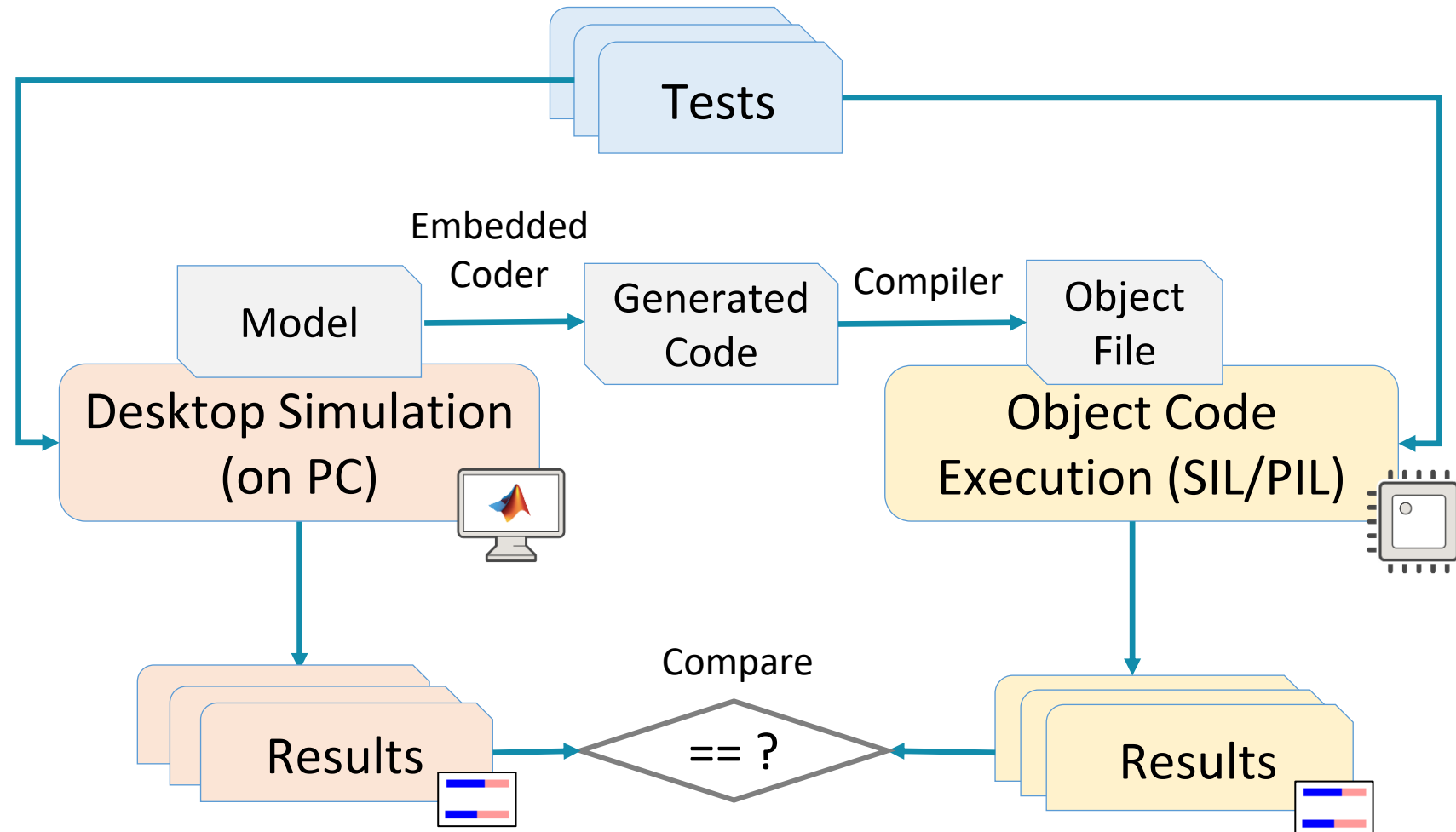
MISRA C:2012 Guidelines Summary for Enabled Guidelines

Guideline	Description	Mode	Total
-----------	-------------	------	-------

Model-Based Design Reference Workflow



Back-to-Back Testing



Automate Test Creation using Test Manager Wizard

AUTO CREATE

- Test File from Model**
Create a test file from model
- Test for Model Component**
Create a new baseline or back-to-back test for model component
- Test from Spreadsheet**
Create a new test with data specified in a spreadsheet
- Test for C/C++ code**
Import and test external C/C++ code

Create Test for Model Component

System > Test Inputs > Verification Strategy > Generated Test

How do you want to test the component?

- Use component under test output as baseline
Simulate the top model and record the outputs of the component to be used as baseline
- Perform back-to-back testing
Set up a test to compare the component under test output in different simulation modes
- Define the verification logic manually
No verification logic is generated for this test

Select simulation modes:

Simulation1: Normal

Simulation2: **Software-in-the-Loop (SIL)**

Back Next

Back to Back Test

sltest_slidemo_fuelsys_tests > New Test Suite 1 > Back to Back Test

Equivalence Test

DESCRIPTION*

SIMULATION 1

SYSTEM UNDER TEST*

Model: slidemo_fuelsys

TEST HARNESS*

Harness: slidemo_fuelsys_Harness1

SIMULATION SETTINGS AND RELEASE OVERRIDES*

INPUTS*

SIMULATION OUTPUTS*

SIMULATION 2 [Copy settings from Simulation 1](#)

SYSTEM UNDER TEST*

Model: slidemo_fuelsys

TEST HARNESS*

Harness: slidemo_fuelsys_SILHarness1

SIMULATION SETTINGS AND RELEASE OVERRIDES*

Simulation Mode: **Software-in-the-Loop (SIL)**

Select releases for simulation:

- Start Time: 0
- Stop Time: 10
- Initial State:

Normal
Accelerator
Rapid Accelerator
Software-in-the-Loop (SIL)
Processor-in-the-Loop (PIL)

Generate Reports for Reviews and Audits

The image displays four overlapping browser windows showcasing different types of reports generated by MathWorks tools:

- Dependency Analyzer Report:** Shows a sidebar with a file tree and a main content area with a 'Contents' section listing various report types like Summary, Subsystem Report, Traceability Report, etc.
- Code Generation Report for 'ControllerModeSelector':** Features a 'Model Information' section and a table for coverage metrics.
- SIL Coverage Report by Model:** Displays a table with columns for Complexity, Decision, Statement, and Function, showing 100% coverage for the 'ControllerModeSelector' model.
- Model Advisor Report for 'ControllerModeSelector':** Includes a 'Filter checks' section with a legend (Passed, Failed, Warning, Not Run), a 'Run Summary' table, and a 'Navigation' sidebar.

Model	Complexity	Decision	Statement	Function
TOTAL COVERAGE	100%	100%	100%	100%
ControllerModeSelector	4	100%	100%	100%

Pass	Fail	Warning	Not Run	Total
347	0	0	16	363

The screenshot shows a 'User Stories' page on the MathWorks website. The main story is titled 'Leonardo Accelerates Development and Compliance of Radar Navigation Software to DO-178C'. It includes a quote from Dr. Calum Brown, Leonardo, and a 'Results' section with a red box highlighting a key achievement.

Challenge
Develop radar navigation software for use on search and rescue helicopters and certify it to DO-178

Solution
Use Model-Based Design to trace requirements to design elements; generate certifiable code; run automated simulation-based, SIL, and PIL tests; and generate reports and documentation

Results

- Recertification cycle times reduced by more than 90%
- Rate of testing quadrupled
- 250,000 pages of interactively linked documentation generated**

250,000 pages of interactively linked documentation generated. "Altogether, we generated the equivalent of around 250,000 pages of interactively linked test reports and other documentation using Simulink Test and Simulink Report Generator," notes Brown. "As the certification authority has *carte blanche* to review whatever they want, we felt it was easier to be able to provide evidence for everything we had done. If the DER wanted to see any particular result, it was available for them and fully linked to our model, which really built their trust."

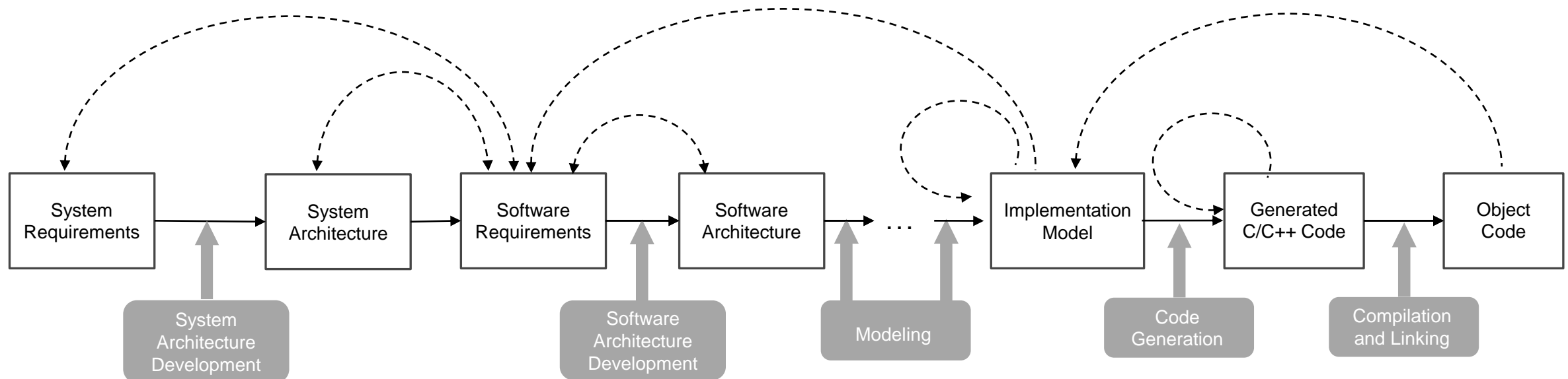
Leonardo User Story

Interactivity Poll: What capabilities presented today could help you improve your development? (choose all that apply)

- Requirements Traceability
- Architecture Analysis for safety
- Standards and Guideline Checking
- Functional Testing at Model level
- Measuring test coverage
- Dashboards and process management
- Automatic code generation
- Back-to-Back or Equivalence Testing
- Report generation
- Other (Enter in Chat)

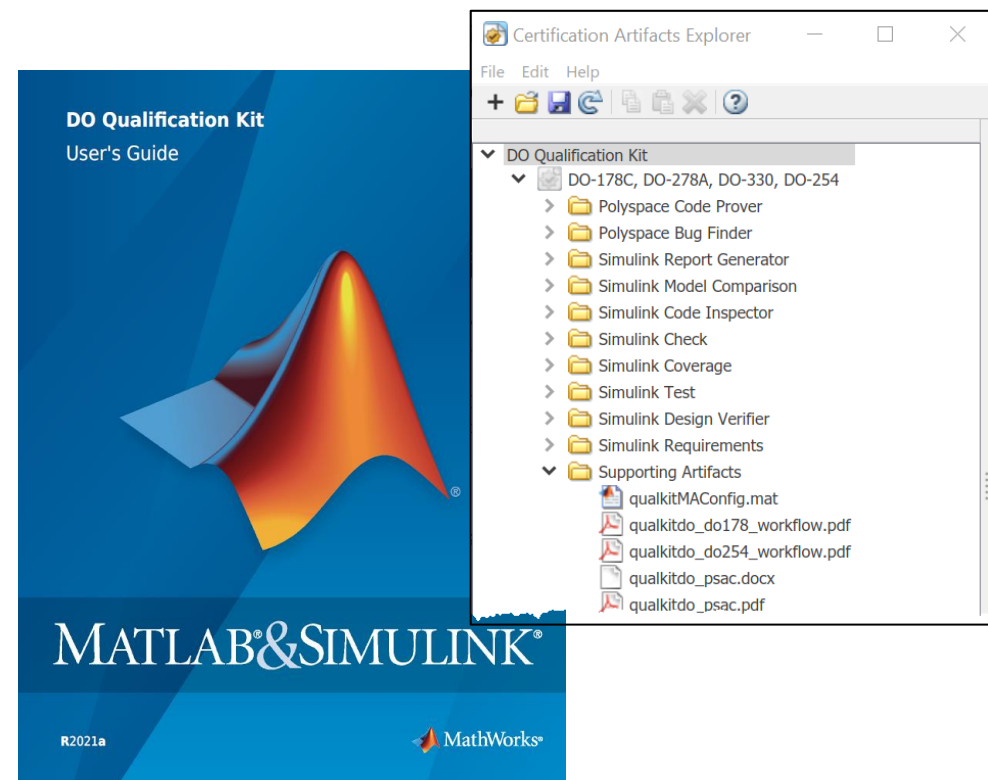
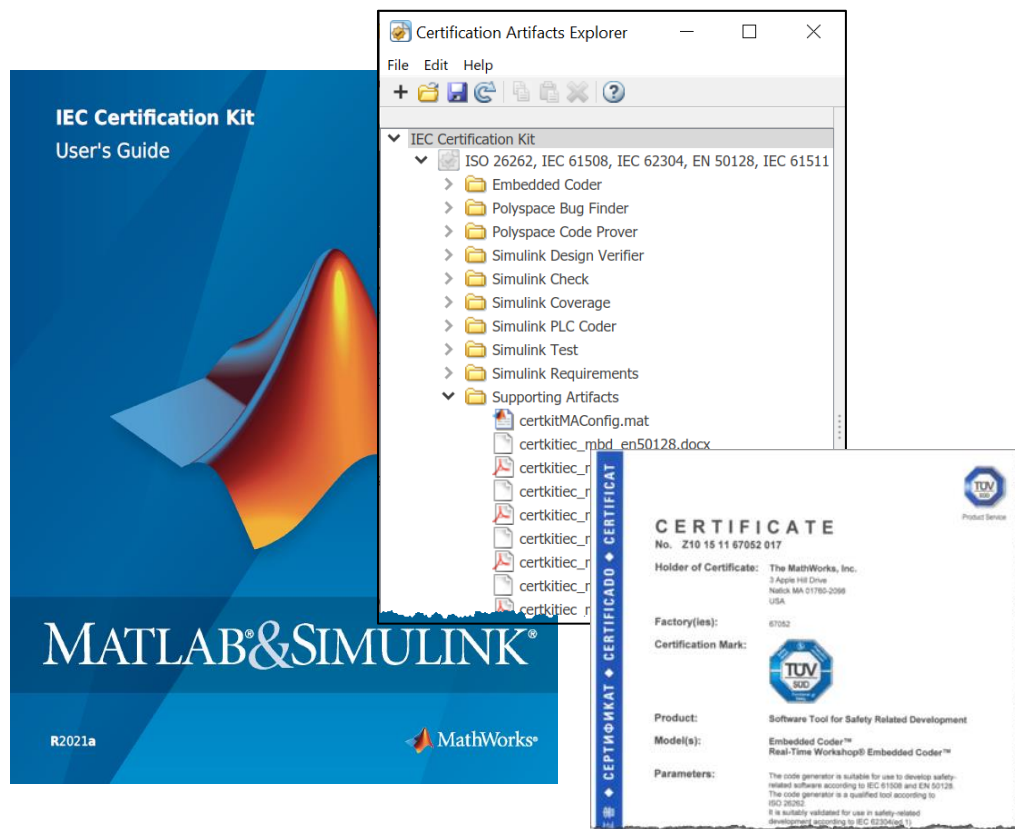
Summary: Reference Workflow with Model-Based Design

- Create traceability across requirements, architecture, design, test and code
- Detect design errors early by continuous testing
- Automatic code generation
- Generate documents and reports for reviews and audits



IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases, and procedures to help certify



Learn More

- [Embedded Code Generation](#)
- [Verification, Validation, and Test Solution Page](#)
- [Verifying Models and Code for High-Integrity Systems](#)
- [Using Model Based Design in the ISO 26262:2018 Case Study](#)
- [Helicopter Flight Control: A Model-Based Design Example for DO-178C and DO-331](#)

MATLAB EXPO 2021

Thank you

