

MATLAB EXPO

Master Class: Developing Safe and Secure Embedded Software from Desktop to Cloud Using Model-Based Design

Gaurav Ahuja, MathWorks



Rajat Arora, MathWorks



Tooling and approaches must address today's challenges and trends



Aerospace and Defense

Complex multi-domain systems, software-defined and autonomous, model-based and data-driven



Automotive



Communications

Comms infrastructure, plus all types of connected systems across industries



Software and Internet

Big Data, Agile, DevOps, integration with IT systems



Railway Systems

Modernization, often on legacy platforms, becoming data-centric for optimization and maintenance



Energy Production



Electronics

Wide range of compute platforms, many kinds of HW/SW integration



Financial Services



Process Industries



Industrial Machinery



Semiconductors

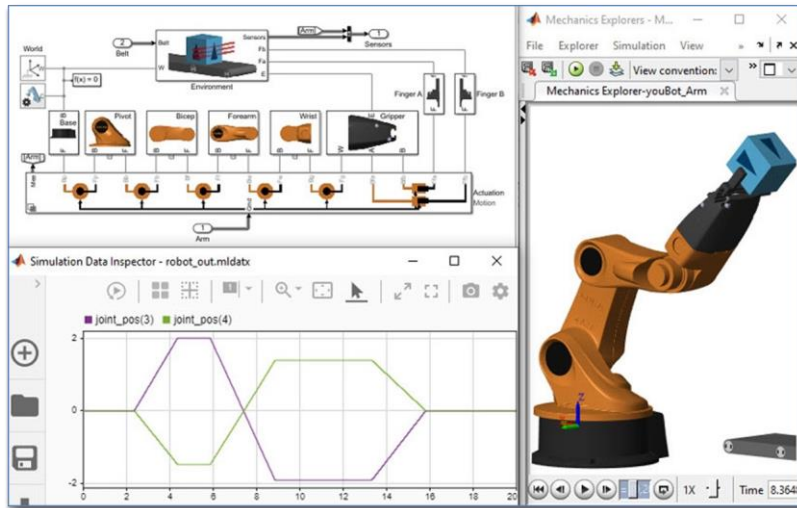
There are three key pieces to

Modeling & Simulation

Model-Based Design

Test & Verification

Code Generation



Conditions analyzed

Description	True	False
Condition 1, "alt>10000"	4 U1.1	185 U1.1
Condition 2, "anomaly"	0 U1.1	4 U1.1

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

Decision/Condition	True Out	False Out
Transition trigger expression		
Condition 1, "alt>10000"	TF U1.1	Fx U1.1

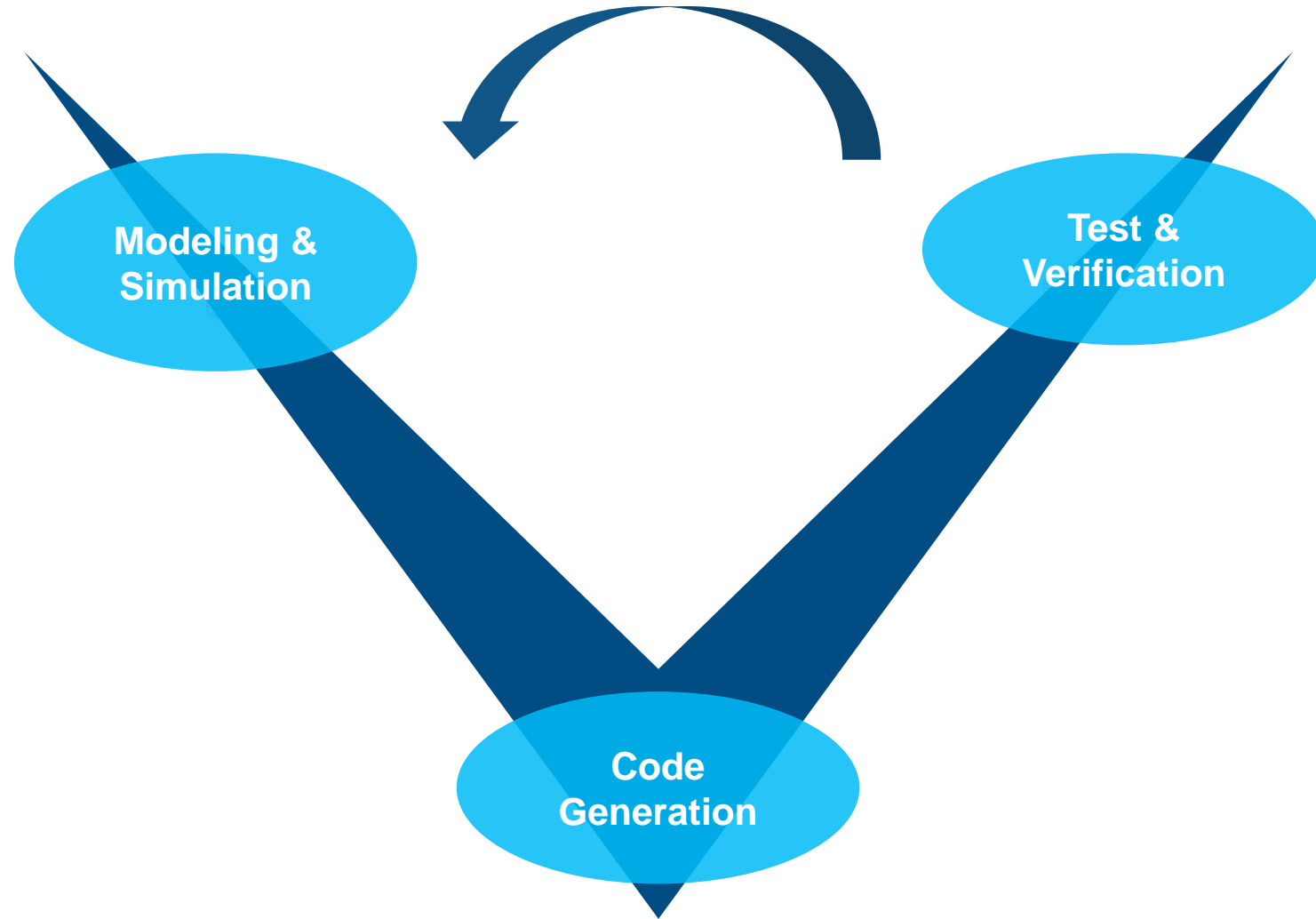
```

604      /* End of Saturate: '<S210>/Saturation' */
605
606      /* RelationalOperator: '<S196>/NotEqual' */
607      NotEqual_n = (0.0F != Switch_f);
608
609      /* Signum: '<S196>/SignPreSat' */
610      if (Switch_f <= 0.0F) {
611          Switch_f = -1.0F;
612      } else {
613          if (Switch_f >= 0.0F) {
614              Switch_f = 1.0F;
615          }
616      }
    
```

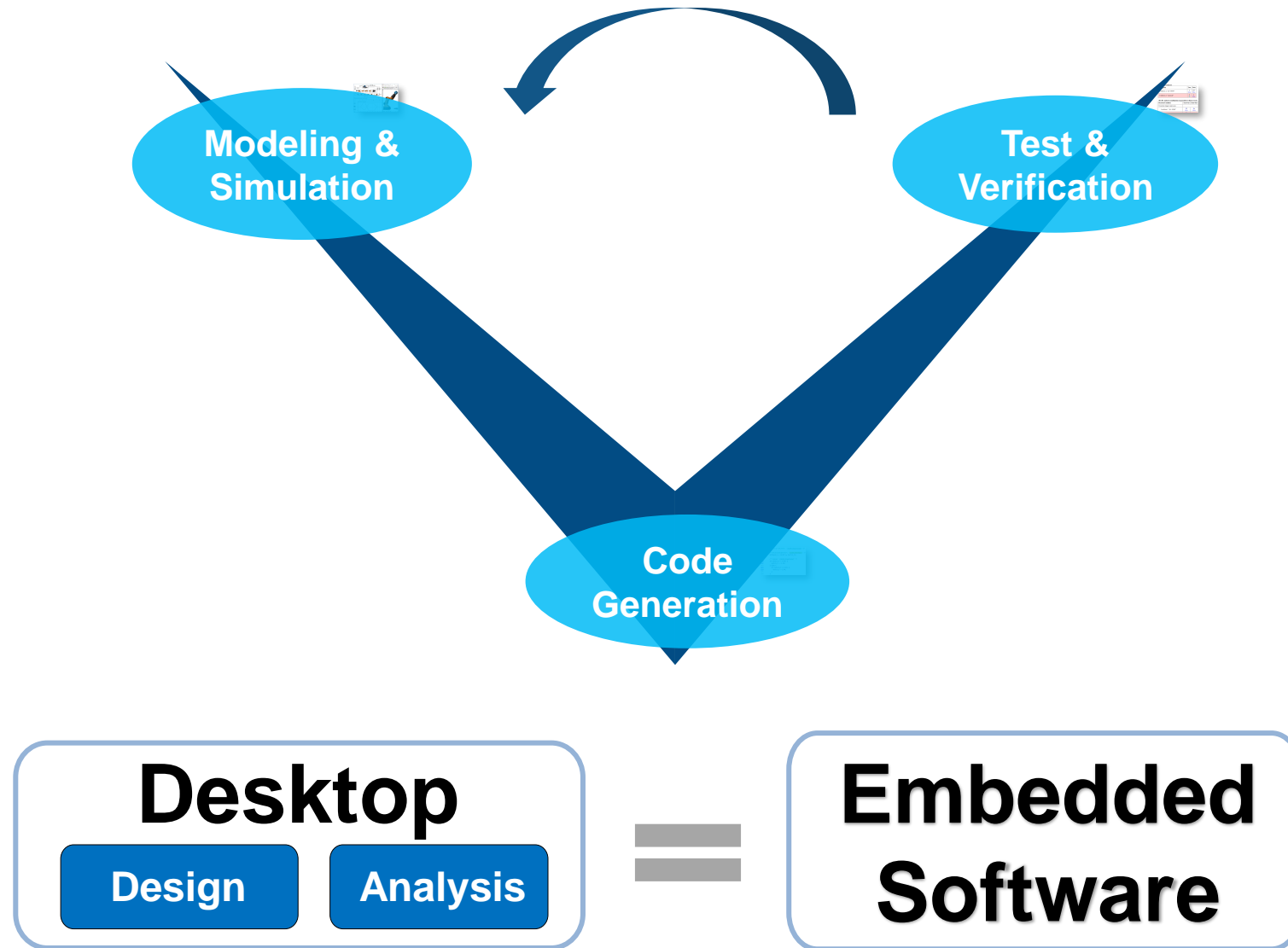
SIMULINK®

Simulation and Model-Based Design

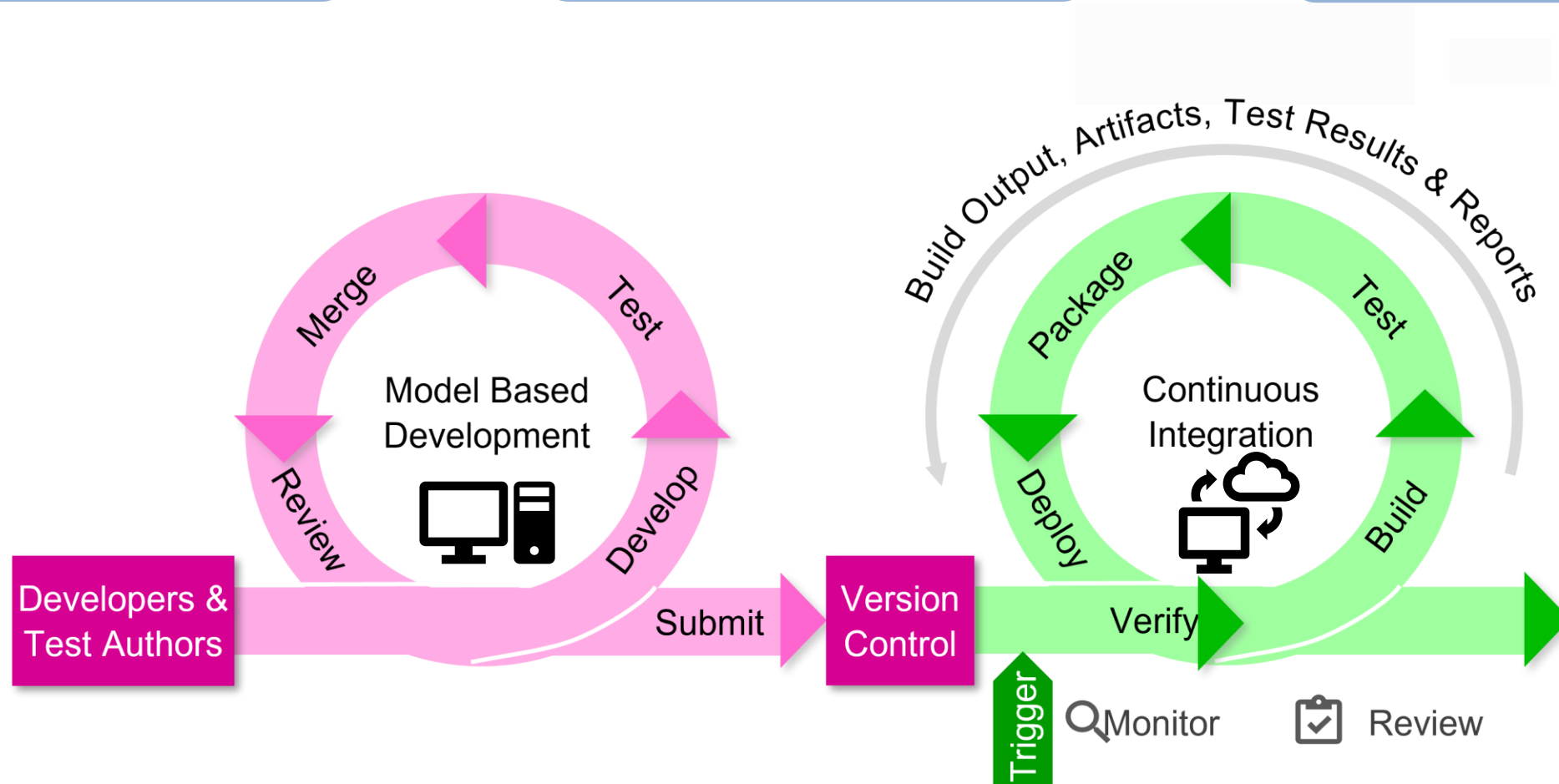
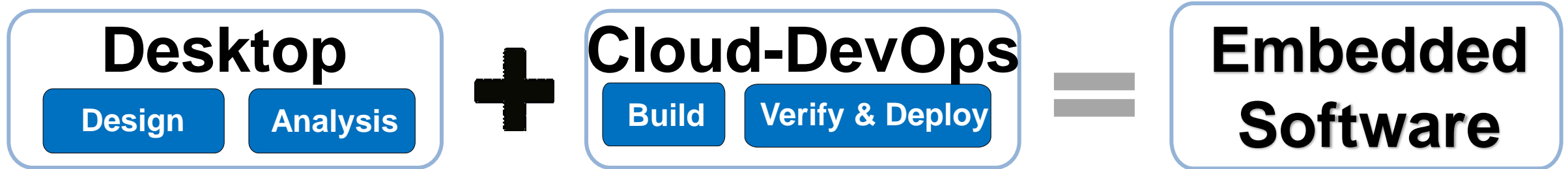
In Model-Based Design, a **system model** is at the **center** of the **workflow**



Traditional Model Based Design Development

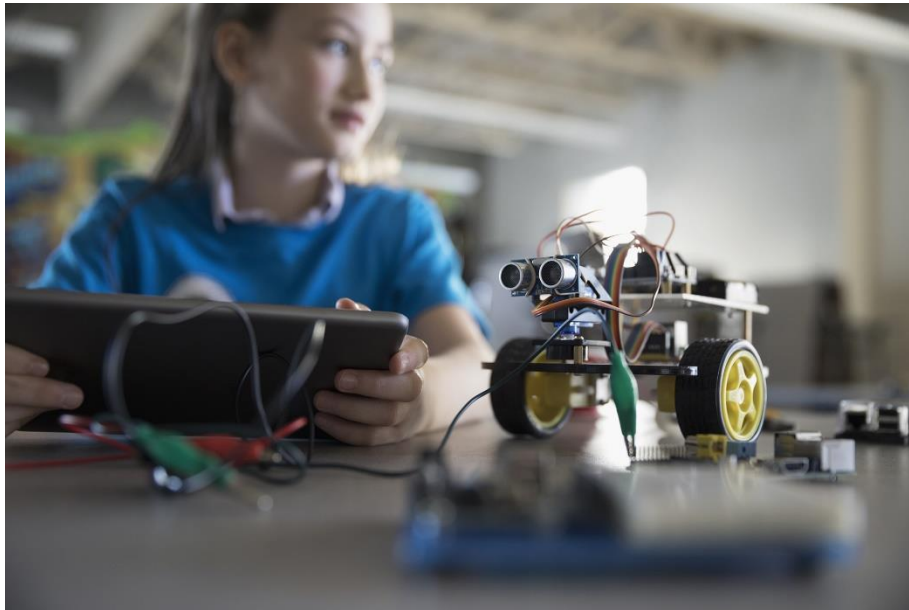
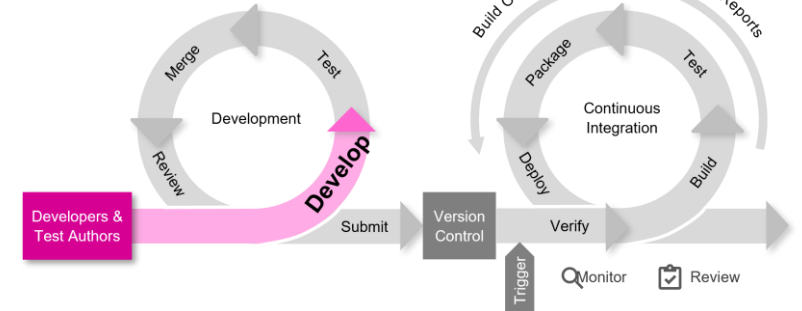


Scaling Development To Address Complexity



Make Modeling and Simulation Easier

1 Development

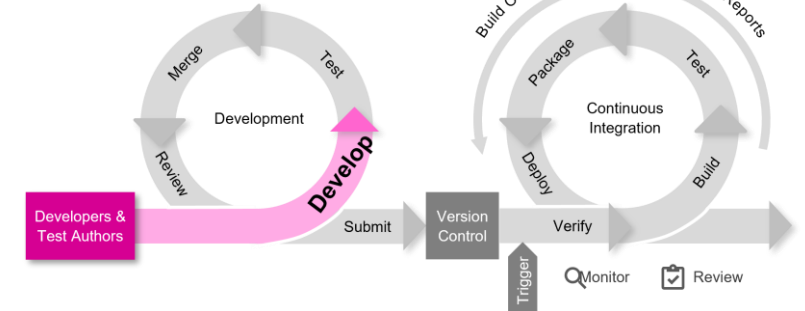
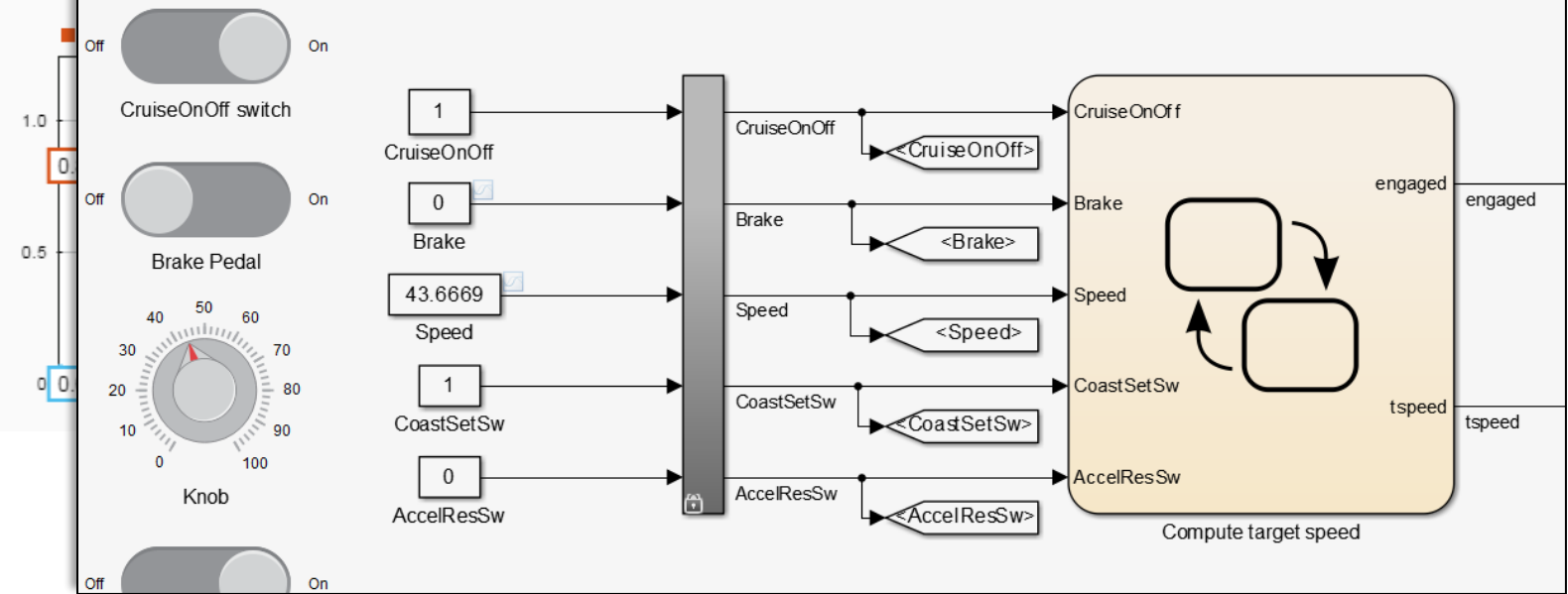
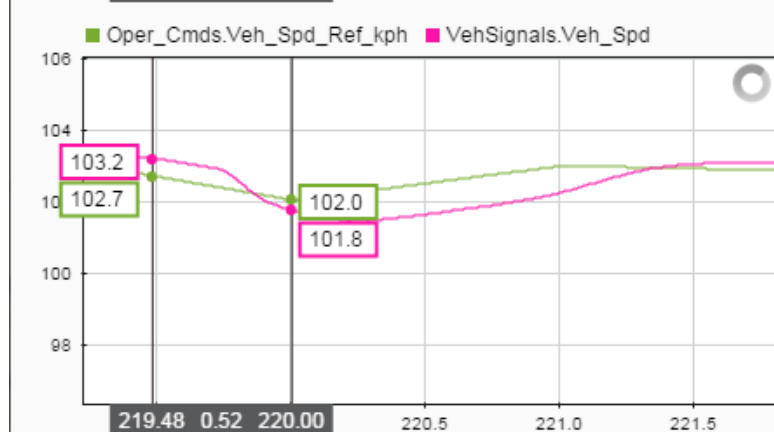
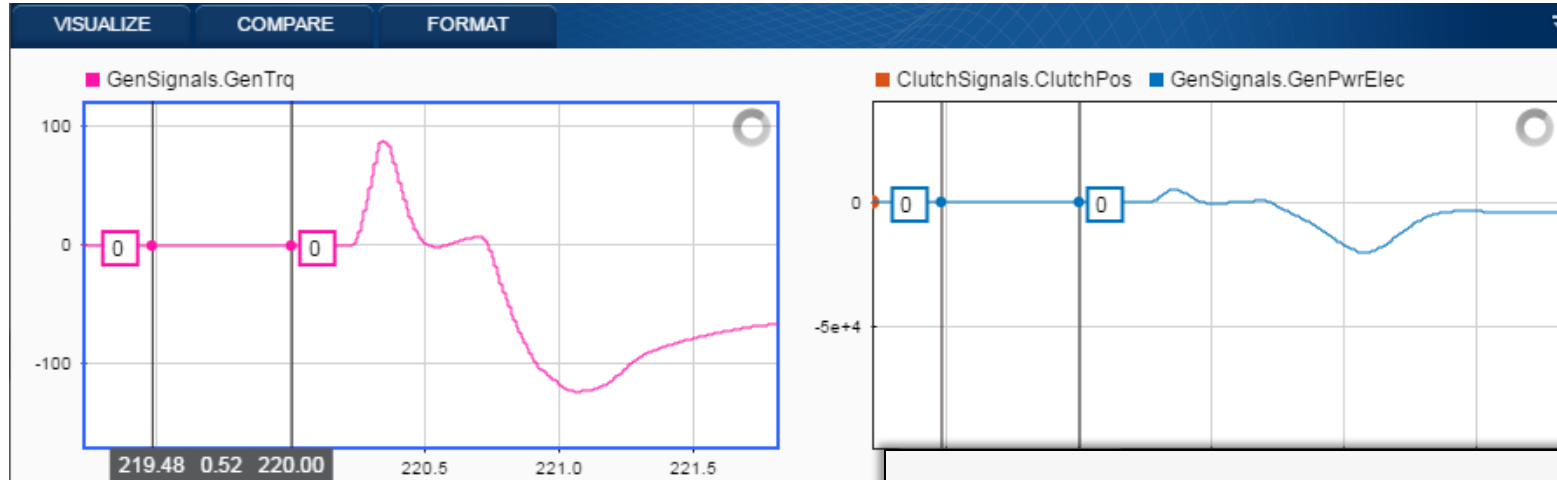


Enable Engineers
at Any Level to
Model and Simulate
Any System

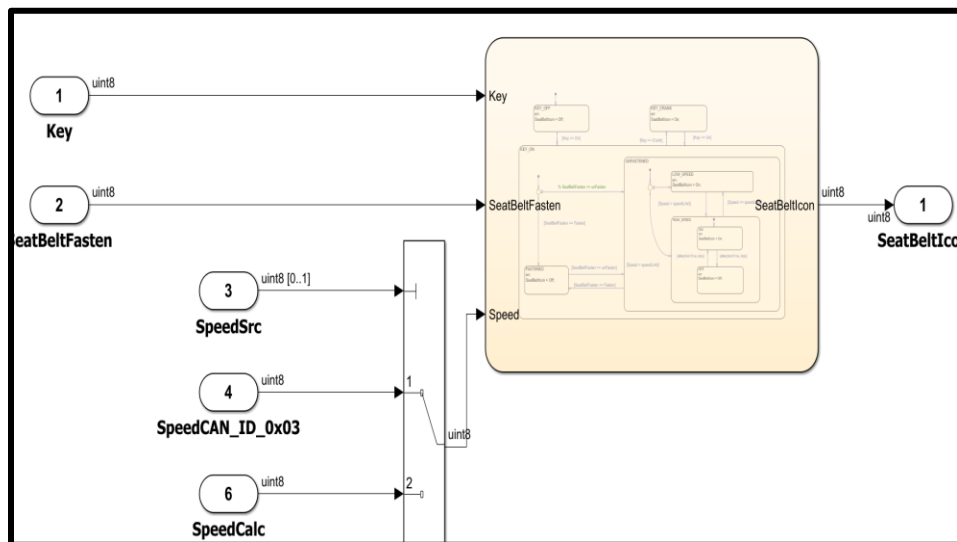
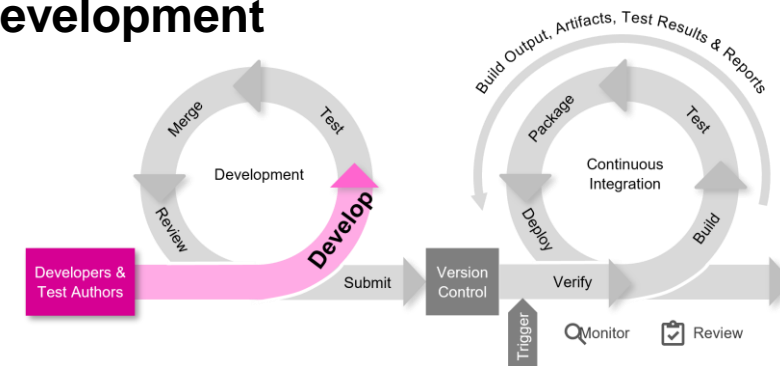


1 Development

Powerful Interfaces to Explore Behavior



1 Development



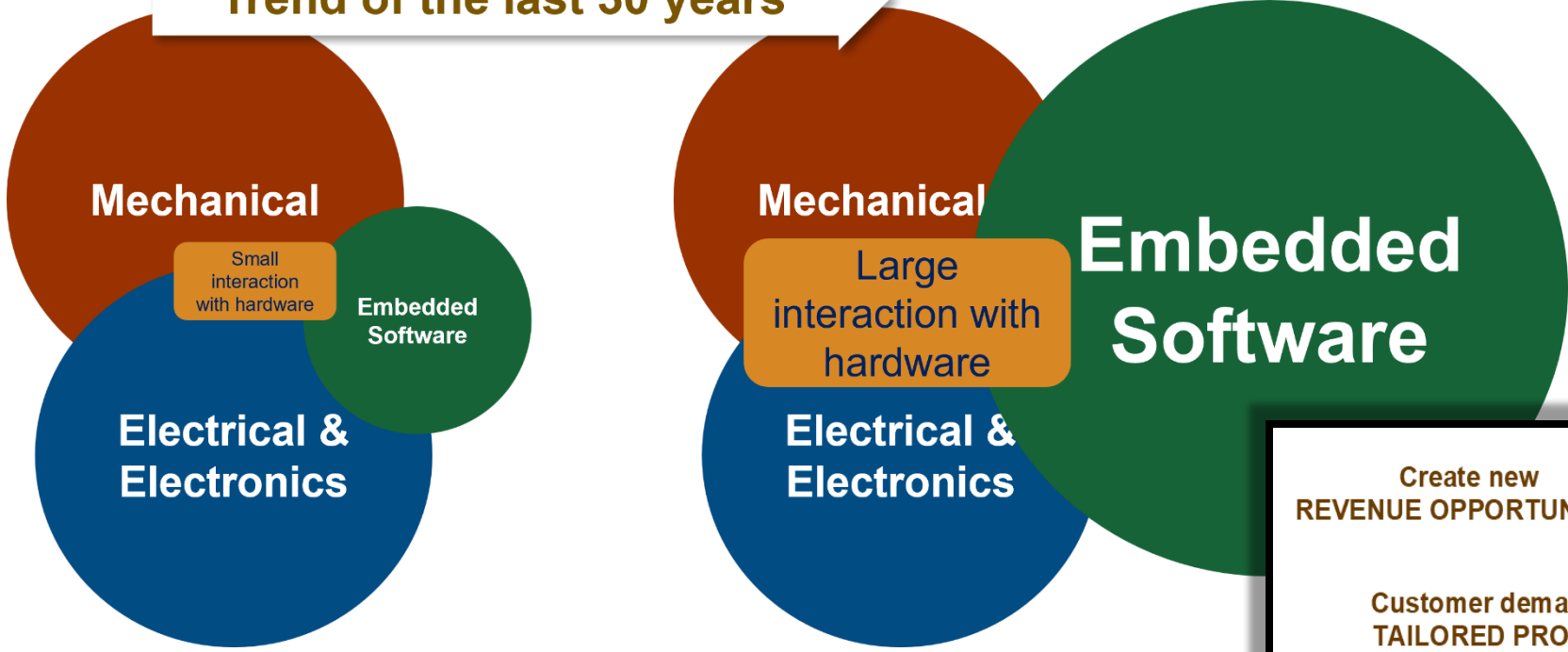
Simulink Model
Application Logic



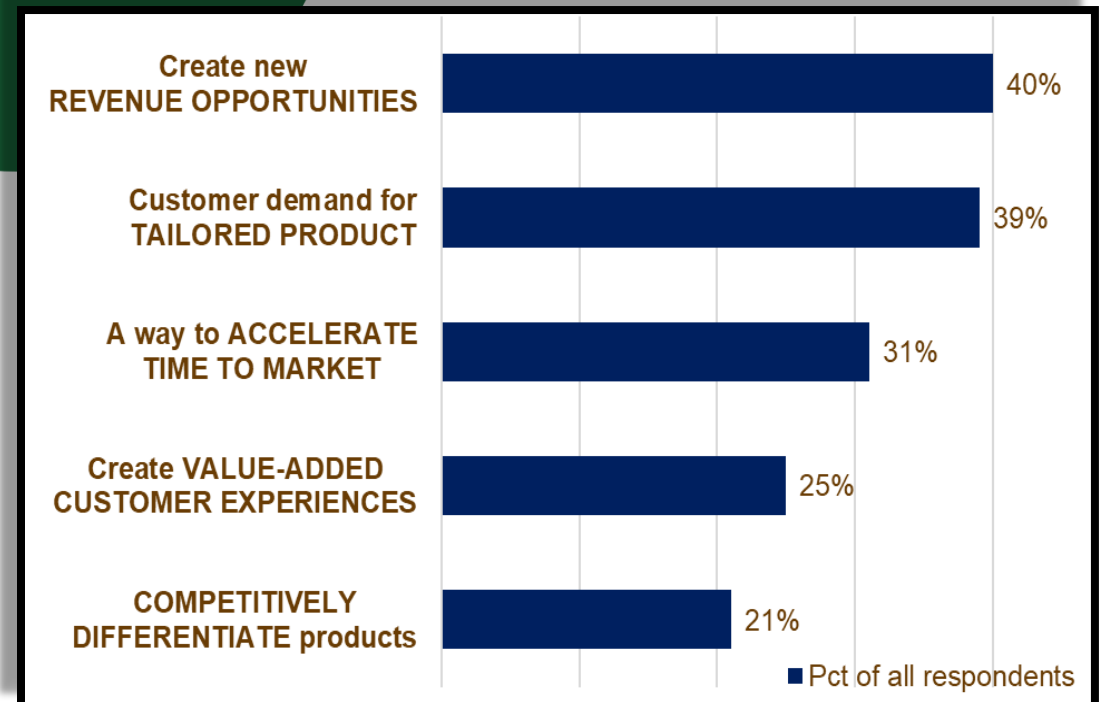
Efficient
C/C++

Why Use Model-Based Design for Embedded System Development?

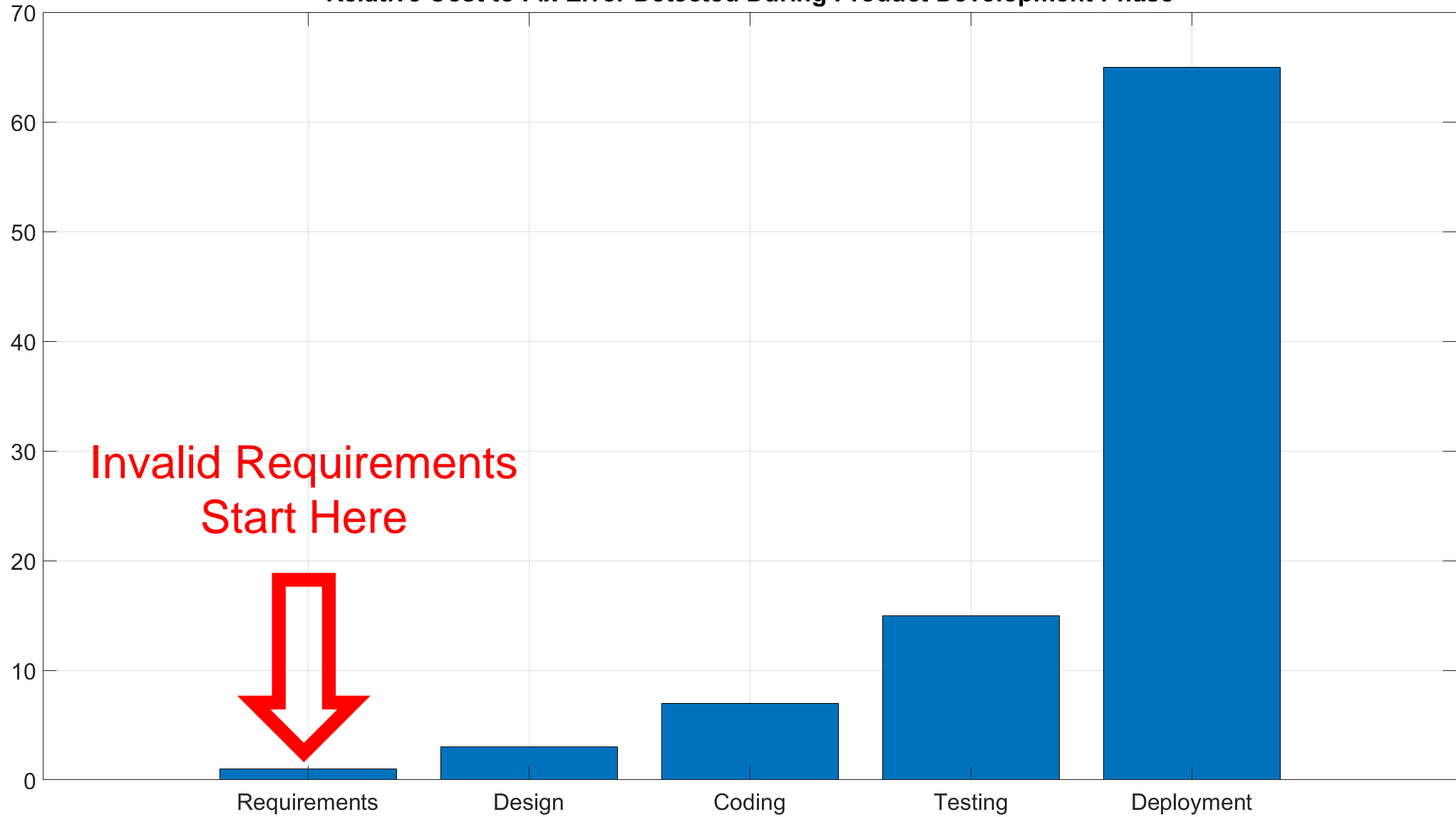
Trend of the last 30 years



MathWorks customers tell us that developing embedded software has become **a major portion of the product design effort**



Relative Cost to Fix Error Detected During Product Development Phase



Data gathered by Hewlett Packard referred by XB in 2017

<https://xbsoftware.com/blog/why-should-testing-start-early-software-project-development/>

Quantifiable benefits of Model-Based Design



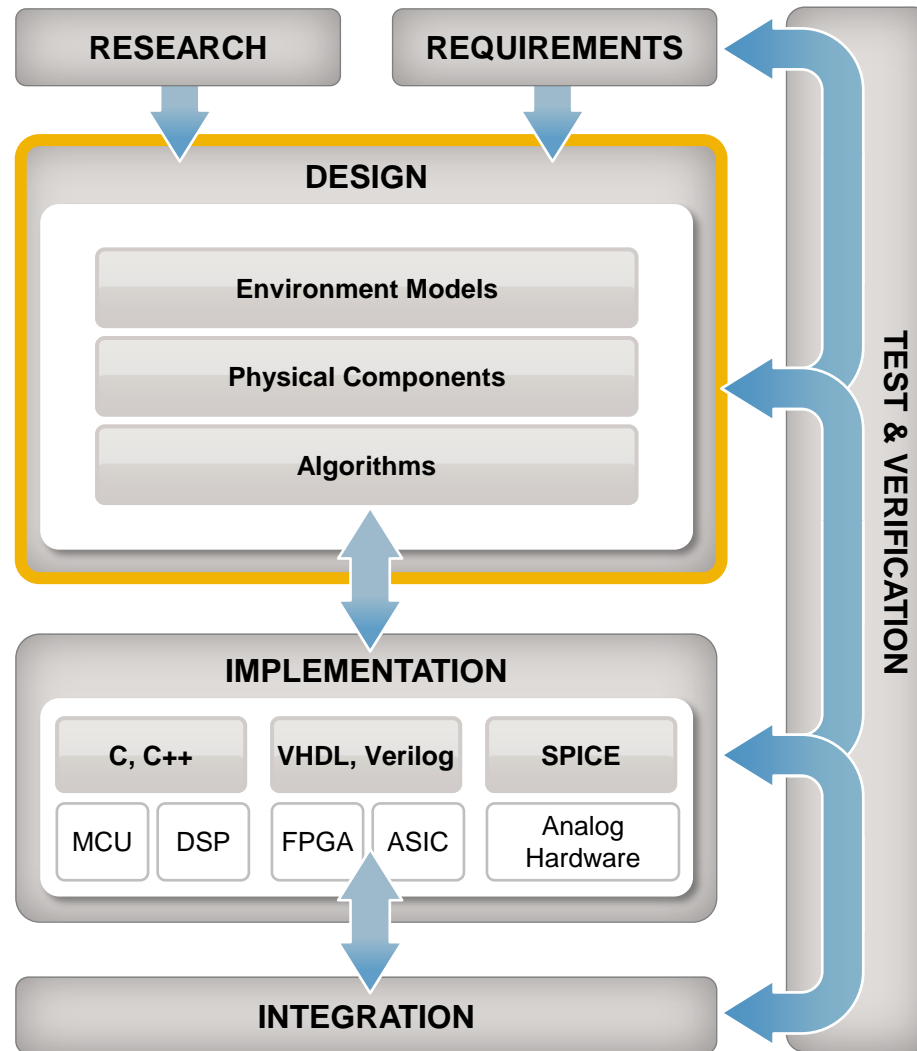
Model-Based Design enabled Continental to verify our design in-vehicle earlier, **eliminating six months of hardware development** and one prototype build. **Verification time was cut by up to 50 percent. 90 percent of application automatically coded.**

Thomas Ehl, Continental



“Front-loaded development with Model-Based Design enables us to **shorten development cycles and minimize rework**, which allows us to **deliver products earlier than our competitors.**”

Dr. Hisahiro Ito, Asst. GM.



System models reused across 54 products worldwide. “Once we had moved to Model-Based Design, we were able to use the same core system in many different vehicles by simply calibrating parameters such as the vehicle dimensions **and then re-generating production code.**”

Johan Hägnander, GM Engineering Europe



“We use our system design model in Simulink for ARP4754 to establish stable, objective requirements. **We save time by using the model as the basis for our software design model for DO-178—**from which we generate flight code— and reusing validation tests for software verification.”

Ronald Blanrue, Airbus Helicopters

Development Processes for High-Integrity Applications

- High integrity applications development follows **standards and guidelines**
- Standards and Guidelines have objectives for development process activities
 - Impose additional constraints on development
 - Require creation of additional artifacts
 - Require more thorough verification, validation and testing activities
- Standards and Guidelines require evidence that the objectives were met to certify: **compliance demonstration**

DO 178C



DO 254



ISO 26262



IEC 61508



EN 50128



IEC 62061



IEC 62304

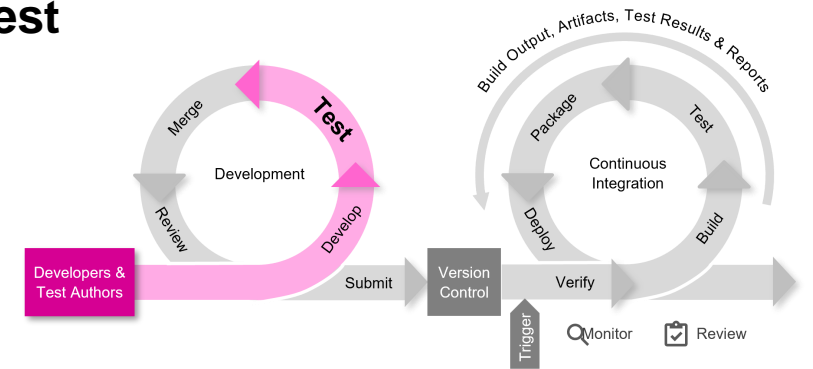


ISO 25119



Systems Analysis

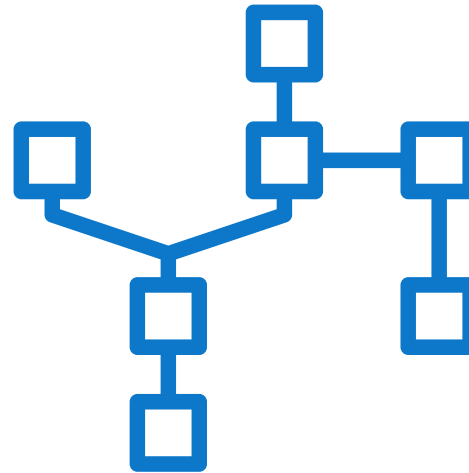
2 Test



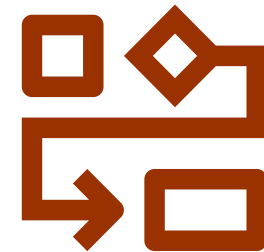
Requirements



Systems

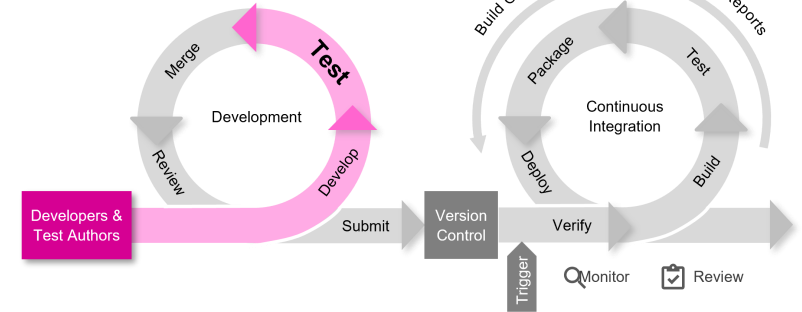


Components

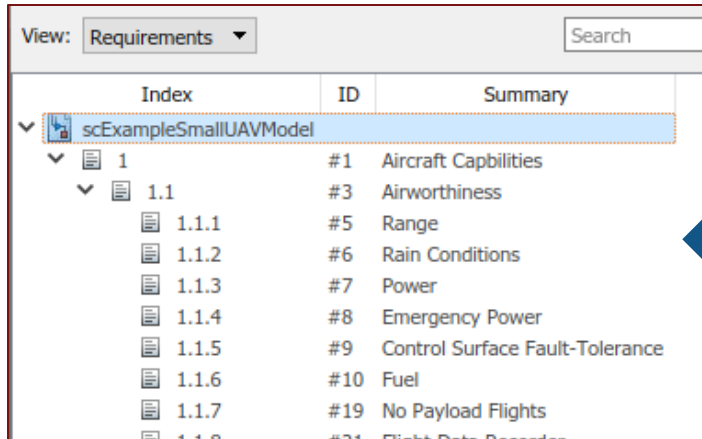


Systems Analysis

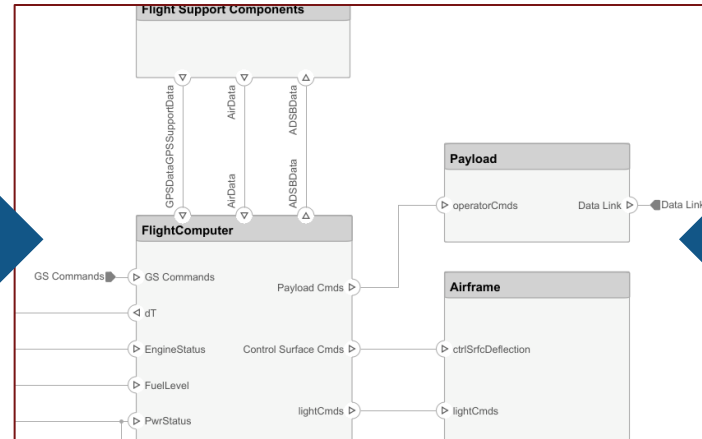
2 Test



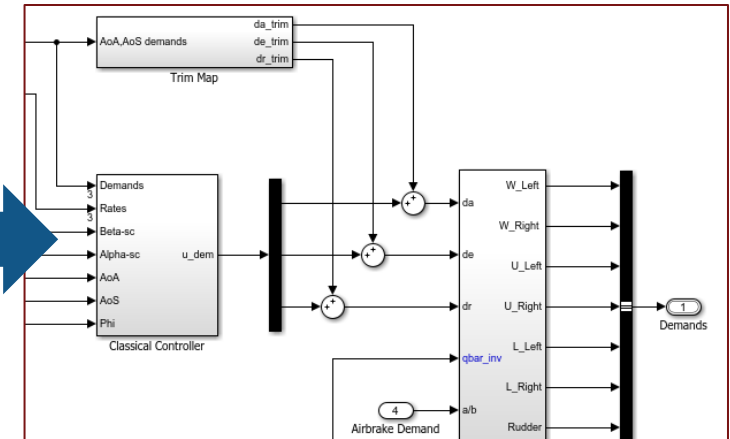
Requirements Toolbox



System Composer

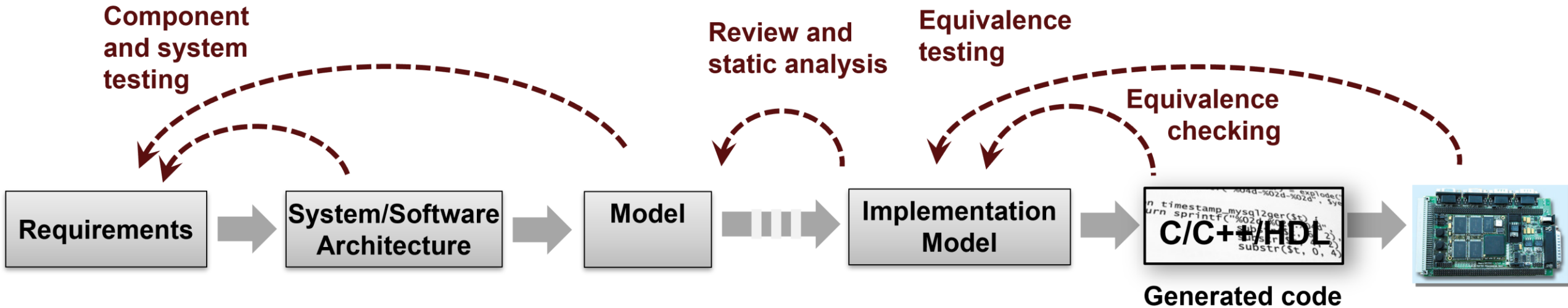
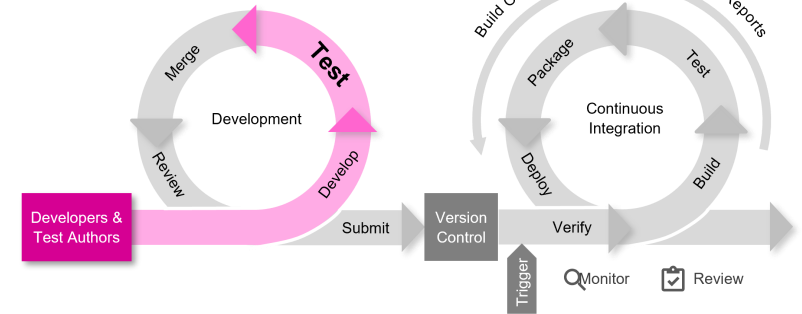


Simulink

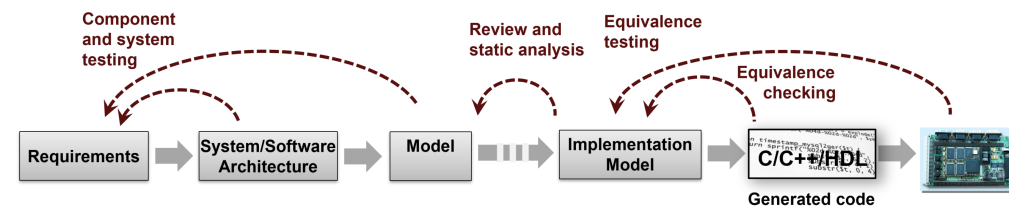


2 Test

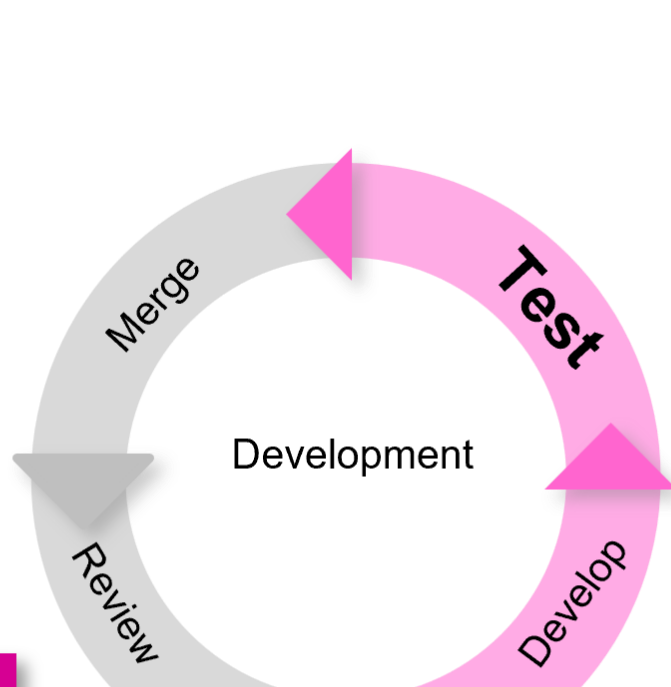
High Integrity Verification Workflow



High Integrity Verification Workflow

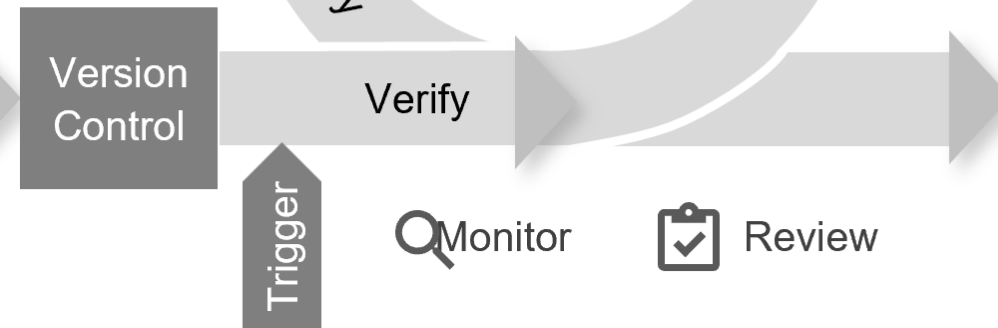


2 Test

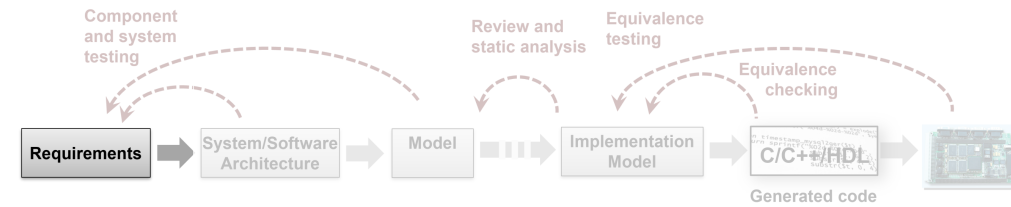


Developers & Test Authors

Submit



Requirements Traceability and Analysis



#31: Increment mode

↑ IMPLEMENTS

opMode.Increment

Implemented	Verified
<div style="width: 75%; height: 15px; background-color: blue;"></div>	<div style="width: 10%; height: 15px; background-color: green;"></div>
<div style="width: 90%; height: 15px; background-color: blue;"></div>	<div style="width: 60%; height: 15px; background-color: green;"></div>
Implemented: 16, Justified: 0, None: 2, Total: 18	

Implemented by:

- [counter](#)

Verified by:

- [Decrement button hold](#)

Issue: Destination Changed.

- Where are requirements implemented?
- Is design and requirements consistent?
- How are they tested?

Why traceability matters...



Ensure application is complete, fully tested, and meets customer requirements



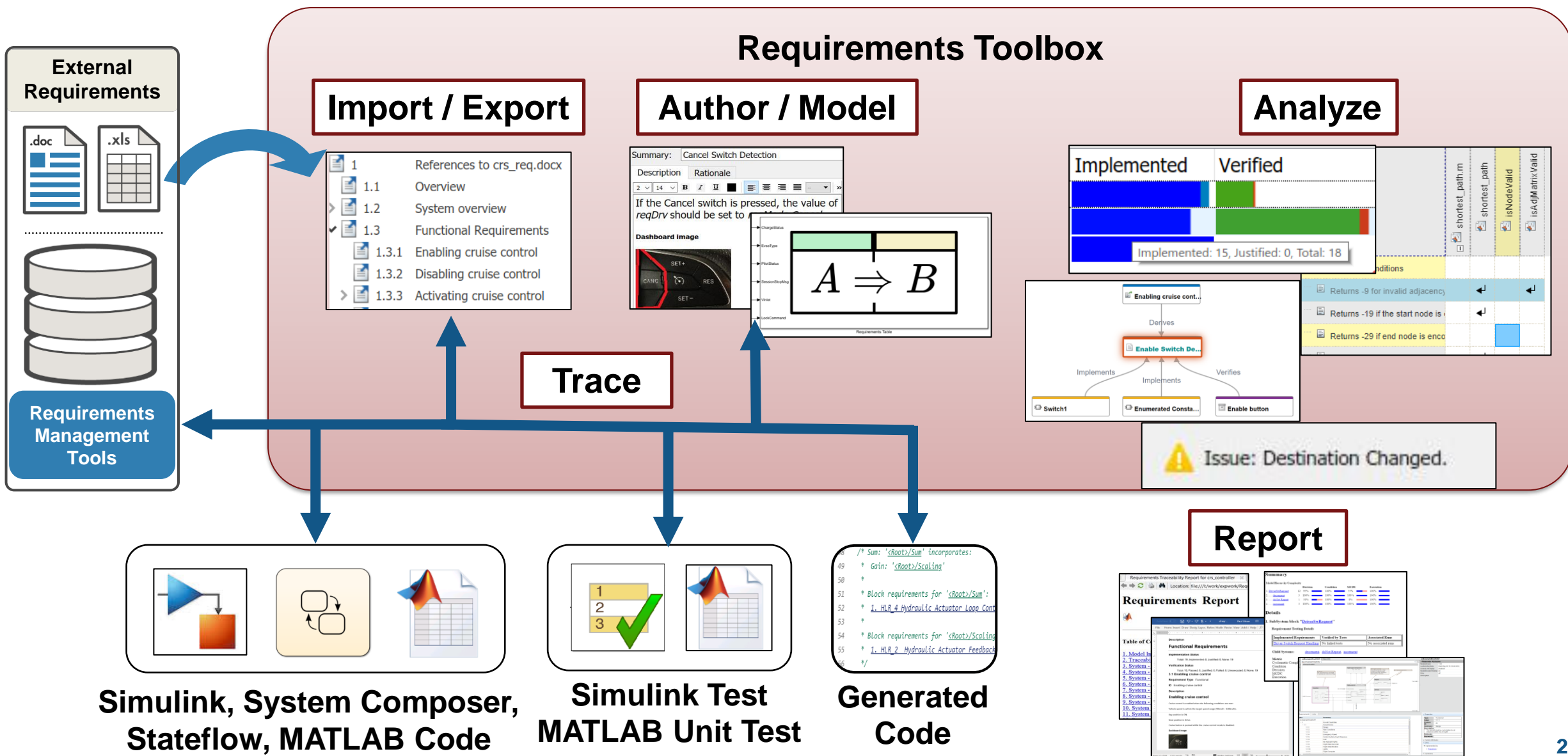
Understand the impact of requirement changes to implementation and test (i.e. “Digital Thread”)

Required to meet certification standards such as:



- ISO 26262, ASPICE for Auto
- DO-178C for Aerospace
- IEC 62304 in Medical
- Many others....

Author, link, and validate requirements for designs and tests

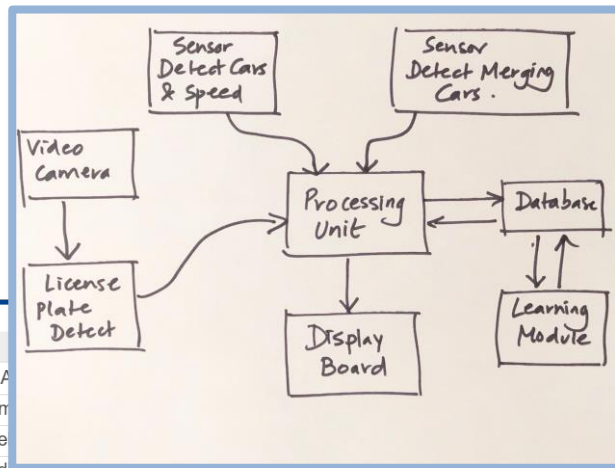
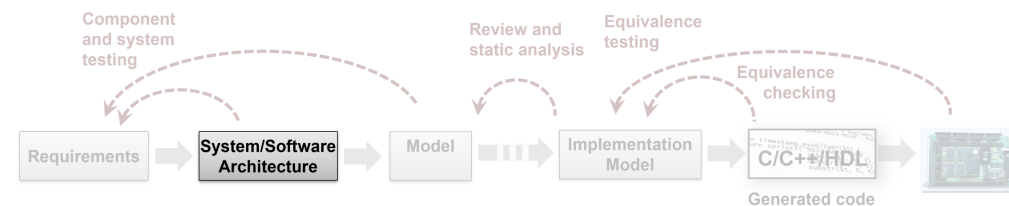


Work with DOORS requirements within System Composer, Simulink or Stateflow with Requirements Perspective

The screenshot displays the MATLAB System Composer interface with the Requirements Perspective. The main workspace shows a Simulink model for a HelicopterSystem. A requirement SYS-5 is highlighted with a tooltip that reads: "SYS-5: Pilot Input Signals. The flight control system shall process three LVDT inputs from the pilot cockpit controls, including fore/aft cyclic position, left/right cyclic position and pedal left/right position." The model includes blocks for Pilot Inputs, Flight Control Computer (FCC), three Actuators, and an AHRS Sensor. The bottom panel shows a requirements table with the following data:

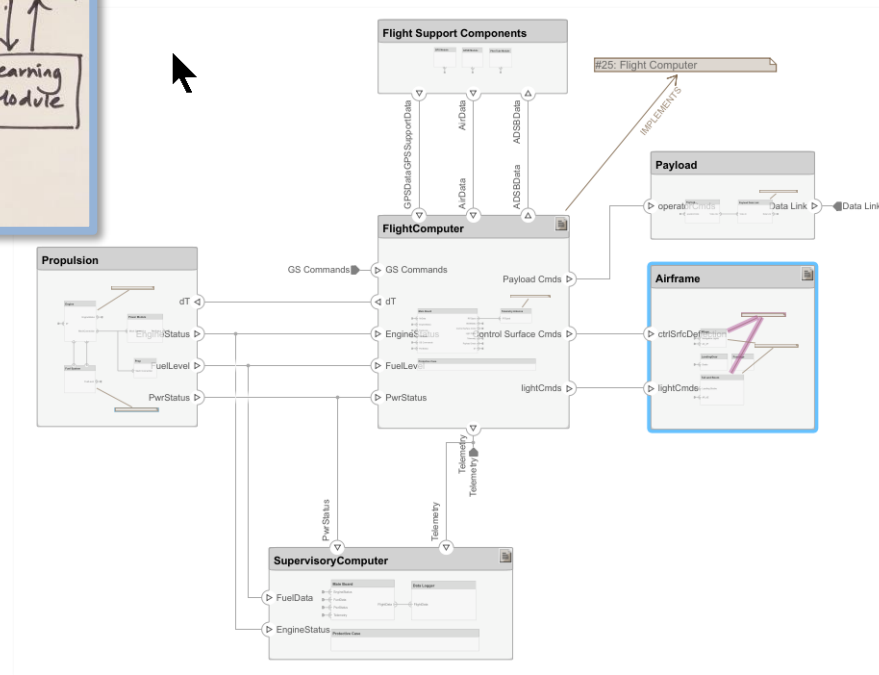
Index	ID	Summary	Status	Implemented	Verified
Helicopter_System_Requirements					
Import1	00000780	References to Helicopter_System_Requirements			
1	SYS-1	Helicopter Flight Control System Requirements	Proposed		
1.1	SYS-2	Introduction	Accepted		
1.2	SYS-3	System Description	Accepted		
1.3	SYS-4	System Requirements	Accepted		
1.3.1	SYS-5	Pilot Input Signals	Proposed		

Model-Based Systems Engineering



Instances		
SmallUAV		
SmallUA		
Airfram		
Fuse		
Landing Gear		1.00
Tail and Boom		2.7
Wings		3.2
Flight Support Components		0.629
ADSB Module		0.156
ABDSB Antenna		0.058
ADSB Board		0.098
GPS Module		0.398
GPS Antenna		0.128
GPS Board		0.27
Pitot Tube Module		0.075
FlightComputer		0.388
Main Board		0.145
Protective Case		0.195

System Composer



How to Model and Analyse System and Software Architecture

Model-Based Systems Engineering

System requirements (Simulink Requirements)

The screenshot shows the Simulink Requirements Editor. On the left, a table lists various requirements. The right pane shows the details for requirement SYS_HLF#7.1.

Index	ID	Summary	Implemented	Verified
1	SYS_IntendedFunction	Intended Function		
2	SYS_Definitions#1	Terms and Definitions		
2.1	SYS_Definitions#1.1	Time gap and clearance		
2.2	SYS_Definitions#1.2	Lead Car or forward Vehicle		
2.3	SYS_Definitions#1.3	Set speed		
2.4	SYS_Definitions#1.4	Target Vehicle		
3	SYS_CarModel#1	Car Model Specs		
4	SYS_HLF#1	HLF Modes of Operation		
4.1	SYS_HLF#2	Off Mode		
4.2	SYS_HLF#3	Standby Mode		
4.3	SYS_HLF#4	Active Mode		
5	SYS_HLF#5	HLF Driver Interface		
5.1	SYS_HLF#5.2	Target Speed Selection		
5.2	SYS_HLF#5.3	Driver Displays		
6	SYS_HLF#6	HLF Performance Requirements		
6.1	SYS_HLF#6.1	Operational Limits		
6.2	SYS_HLF#6.2	Speed Control Performance		
6.3	SYS_HLF#6.3	Following Control Performance		
6.4	SYS_HLF#6.4	Lateral Control Performance		
7	SYS_HLF#7	HLF Detection Capabilities		
7.1	SYS_HLF#7.1	Detection of lead car		
7.2	SYS_HLF#7.2	Leading Vehicle Discrimination		
7.3	SYS_HLF#7.3	Attributes of the leading vehicle		
7.4	SYS_HLF#7.4	Lane Boundary detection		

Requirement: SYS_HLF#7.1

Details

Type: Functional
 Index: 7.1
 Custom ID: SYS_HLF#7.1
 Summary: Detection of lead car
 Description: Rationale
 The HLF system shall detect the leading vehicle.

Links

- Allocated to: Traffic Participants Detection

The screenshot shows the Simulink System Architecture Modeler interface. The main workspace displays a hierarchical block diagram of the system architecture. The diagram includes blocks for 'Plant Vehicle', 'Other', 'External Environment', 'Vehicle Functional Systems', 'Other Interface', 'HLF Modes', 'Traffic Participants Detection', 'Lane Detection', 'Speed Limit Recognition', and 'Highways Lane Following System'. The 'HLF Modes' block is highlighted in green. The right pane shows the 'Property Inspector' for the selected block, and the bottom pane shows an 'Interfaces' table.

Interface	Type	Dimensions	Units	Complexity	Minimum	Maximum	Description
DD_HLF_DataTypes.sldd							

System Architecture Models (System Composer)

Model-Based Systems Engineering

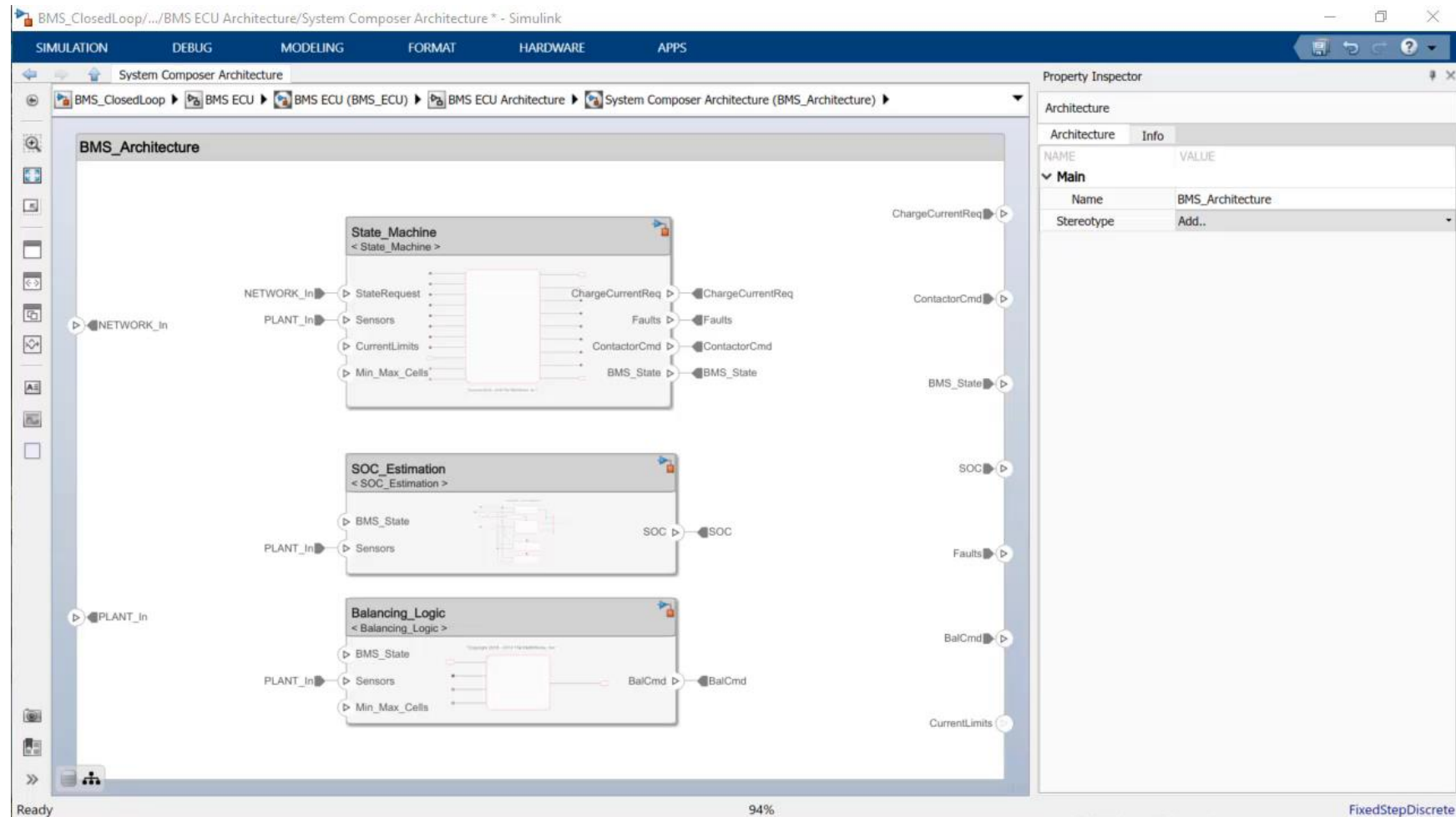
The screenshot shows the Requirement Editor interface. On the left is a tree view of requirements. The main area displays a table of requirements with columns for Index, ID, Summary, Implemented, and Verified. The detailed view on the right shows the properties for requirement SW_HLF5.3.1, including its type (Functional), index (4.3.1), and description: "The HLF system shall identify the leading vehicle among other traffic participants within the same lane boundaries based on the relative distance to ego vehicle."

Index	ID	Summary	Implemented	Verified
1	SW_HLF#1	Intended Functionality of the HLF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	SW_HLF#2	HLF timing	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	SW_HLF#4	HLF Assumptions for Inputs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	SW_HLF#5	HLF Input Conditioning	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4.1	SW_HLF5.1	Lane Center Estimation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4.2	SW_HLF5.2	LaneCenter Conditioning for Coordinate systems	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4.3	SW_HLF5.3	Leading Car Detection	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4.3.1	SW_HLF5.3.1	Identification of leading car	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4.3.2	SW_HLF5.3.2	Information about Leading Car	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	SW_HLF#6	HLF Path Planning	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5.1	SW_HLF6.1	Preview Curvature	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5.2	SW_HLF6.2	Preview Curvature Horizon	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	SW_HLF#7	HLF Controllers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.1	SW_HLF7.1	Path following Controller	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.1.1	SW_HLF7.1.1	Leading Vehicle Limiter	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.1.2	SW_HLF7.1.2	Longitudinal Velocity Limiter	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.1.3	SW_HLF7.1.3	Time Gap	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.1.4	SW_HLF7.1.4	MPC Path Following Controller	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.2	SW_HLF7.2	Watchdog Controller	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6.3	SW_HLF7.3	Controller Mode Selection	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
TSRS			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Software Requirements (Simulink Requirements)

The screenshot displays the Simulink environment. The main workspace shows a software architecture diagram for a Highway Lane Following system. It includes several components: a 'System and Battery Services' block, an 'ECU' block, and a 'HighwayLaneFollowingController_SW_Arch' block. The controller block is further decomposed into 'HLF States' and 'HLF Controller' sub-diagrams. The interface includes a top toolbar with tabs for Simulation, Debug, Modeling, Format, and Apps. A Model Browser on the left shows the project hierarchy, and a Property Inspector on the right shows details for the selected component.

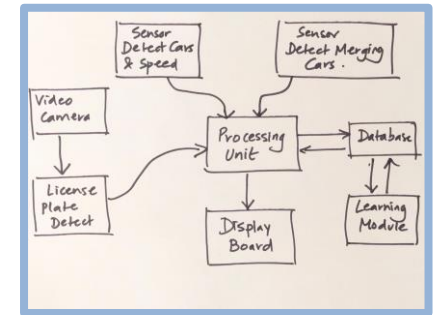
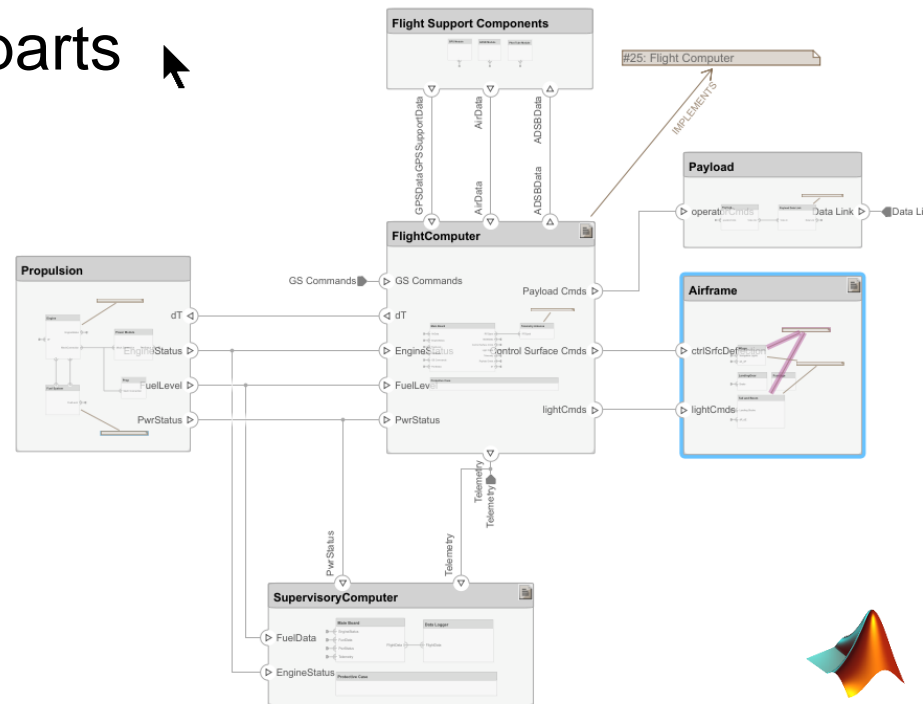
Model-Based Systems Engineering



Model-Based Systems Engineering

- Architecture Models
- Profiles, stereotypes, properties
- Allocate requirements
- Views to focus on relevant parts
- Perform Analysis

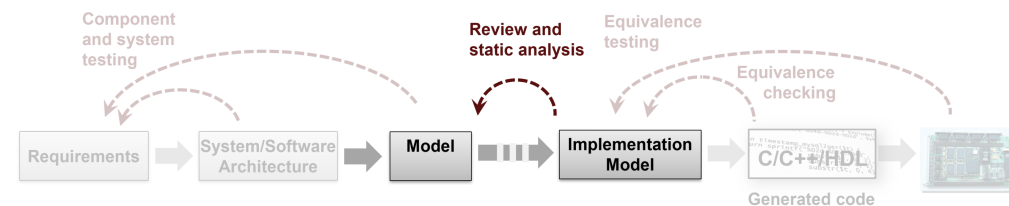
System Composer



SmallUAV	
Instances	Mass(kg)
SmallUAV	15.932
Airframe	9.25
Fuselage	1.7
LandingGear	1.65
Tail and Boom	2.7
Wings	3.2
Flight Support Components	0.629
ADSB Module	0.156
ABDSB Antenna	0.058
ADSB Board	0.098
GPS Module	0.398
GPS Antenna	0.128
GPS Board	0.27
Pitot Tube Module	0.075
FlightComputer	0.388
Main Board	0.145
Protective Case	0.195



Compliance to Standards and Guidelines



- High-Integrity Systems
 - Simulink
 - Check usage of While Iterator blocks
 - Check usage of For and While Iterator subsystems
 - Check for blocks not recommended for C/C++ production
 - Check for inconsistent vector indexing methods
 - Check usage of variant blocks
 - Check usage of looku

Usage of disapproved block

Simulink Design Verifier Results

[Back to summary - Close](#)

antipattern1a/Sum

Overflow **VALID**

Derived Ranges:

Output 1 [-128..127]

Simulink Design Verifier Results

[Back to summary - Close results](#)

antipattern1a/Abs

Overflow **ERROR - View test case**

Derived Ranges:

Output 1 [-128..127]

Is the design built right?
 Is it too complex?
 Is it ready for code generation?

Automate verification with static analysis

The screenshot displays the Model Advisor tool interface. The top toolbar includes buttons for Open, Run Checks, Fix, Justify, and Report. Below the toolbar is the Check Selector panel, which shows a tree view of standards. The 'Modeling Standards' folder is expanded, showing sub-folders for DO-178C/DO-331 Checks, IEC 61508, IEC 62304, ISO 26262, ISO 25119, EN 50128 and EN 50657 Checks, MAB Checks, and JMAAB Checks. The 'By Task' section is also visible, listing various modeling standards for MISRA C:2012, Secure Coding (CERT C, CWE, ISO/IEC TS 17961), and other standards.

The 'Verify compliance with modeling guidelines' panel is also visible, providing tips and check types. The 'Result Statuses' panel is highlighted, showing the following status icons and labels:

- Not Run
- Passed
- Justified
- Warning
- Failed
- Incomplete

The 'Check for:' panel lists the following items:

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

Generate reports for audits

Model Advisor

Guidance Provided to Address Issues or Automatically Correct

Check safety-related diagnostic settings for solvers

Check ID: mathworks.hism.hisl_0043
hisl_0043: Solver

Check diagnostic settings in the model configuration that apply to solvers and might impact safety.

Summary
Status: ⚠ Warning

Warning (4)

Report Result Details

Check diagnostic settings in the model configuration that apply to solvers and might impact safety.

Warning
The model configuration parameters are not set to the recommended values specified in the data file.

Status	Parameter	Current Value	Recommended Values
Warning	Algebraic loop (AlgebraicLoopMsg)	warning	error
Warning	Minimize algebraic loop (ArtificialAlgebraicLoopMsg)	warning	error
Warning	Block priority violation (BlockPriorityViolationMsg)	warning	error
Warning	Automatic solver parameter selection (SolverPrmCheckMsg)	none	error

Recommended Action
Follow the links in the result table to modify the model configuration parameters.

Action Report

The following model configuration parameters have been modified as specified in the data file:

Parameter	Previous Value	Current Value
Algebraic loop (AlgebraicLoopMsg)	warning	error
Minimize algebraic loop (ArtificialAlgebraicLoopMsg)	warning	error
Block priority violation (BlockPriorityViolationMsg)	warning	error
Automatic solver parameter selection (SolverPrmCheckMsg)	none	error

OK

Fix

Check safety-related diagnostic settings for solvers

Check ID: mathworks.hism.hisl_0043
hisl_0043: Solver

Check diagnostic settings in the model configuration that apply to solvers and might impact safety.

Summary
Status: ✔ Passed

Report Result Details

Check diagnostic settings in the model configuration that apply to solvers and might impact safety.

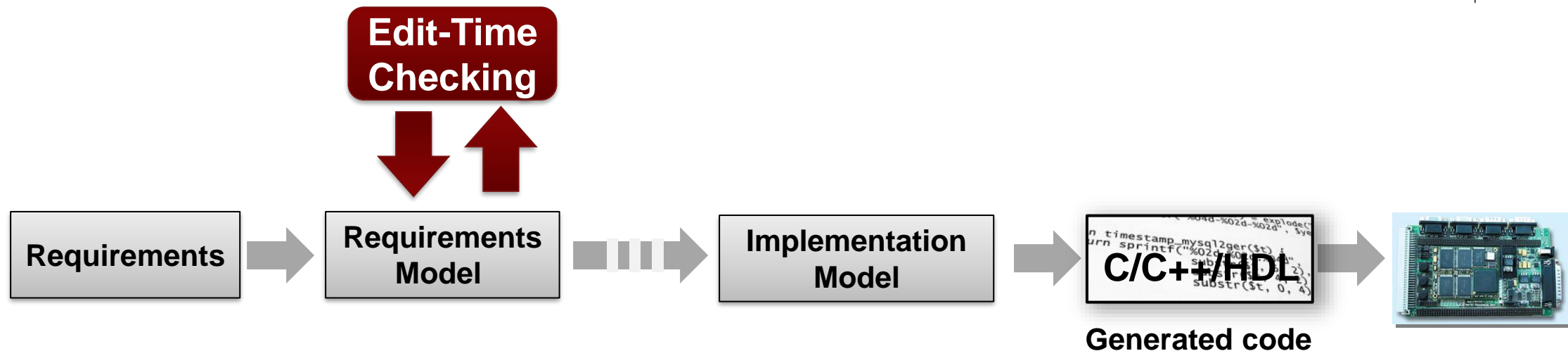
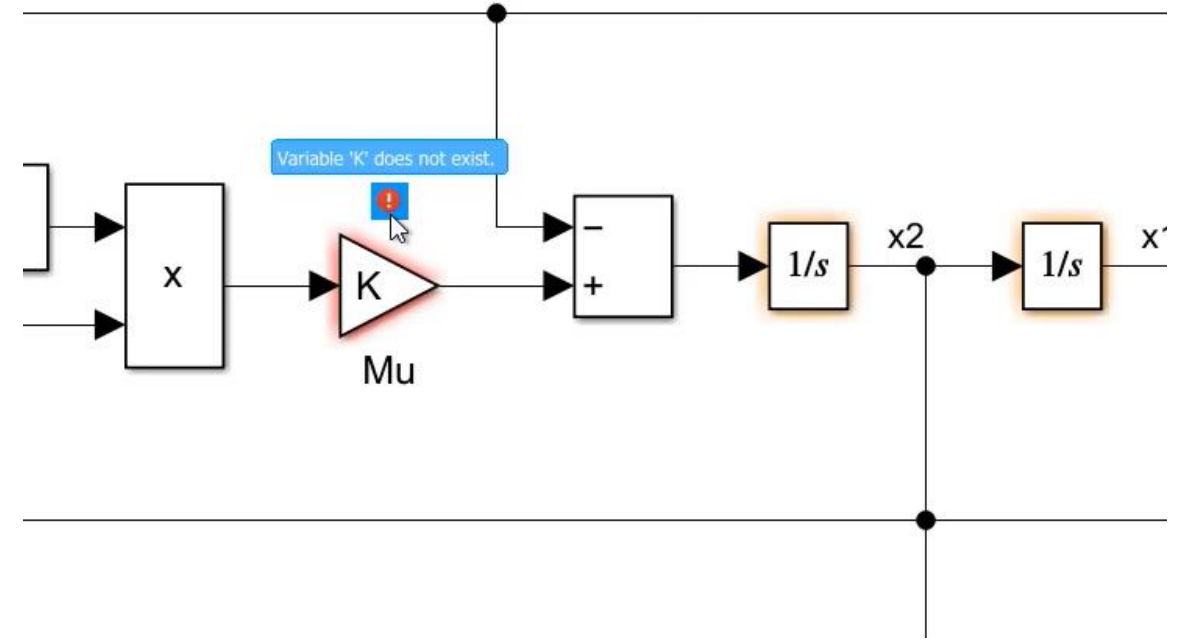
Passed
All constraints on model configuration parameters have been met.

Status	Parameter	Current Value	Recommended Values
Pass	Algebraic loop (AlgebraicLoopMsg)	error	error
Pass	Minimize algebraic loop (ArtificialAlgebraicLoopMsg)	error	error
Pass	Block priority violation (BlockPriorityViolationMsg)	error	error
Pass	Automatic solver parameter selection (SolverPrmCheckMsg)	error	error
Pass	State name clash (StateNameClashWarn)	warning	warning

✔

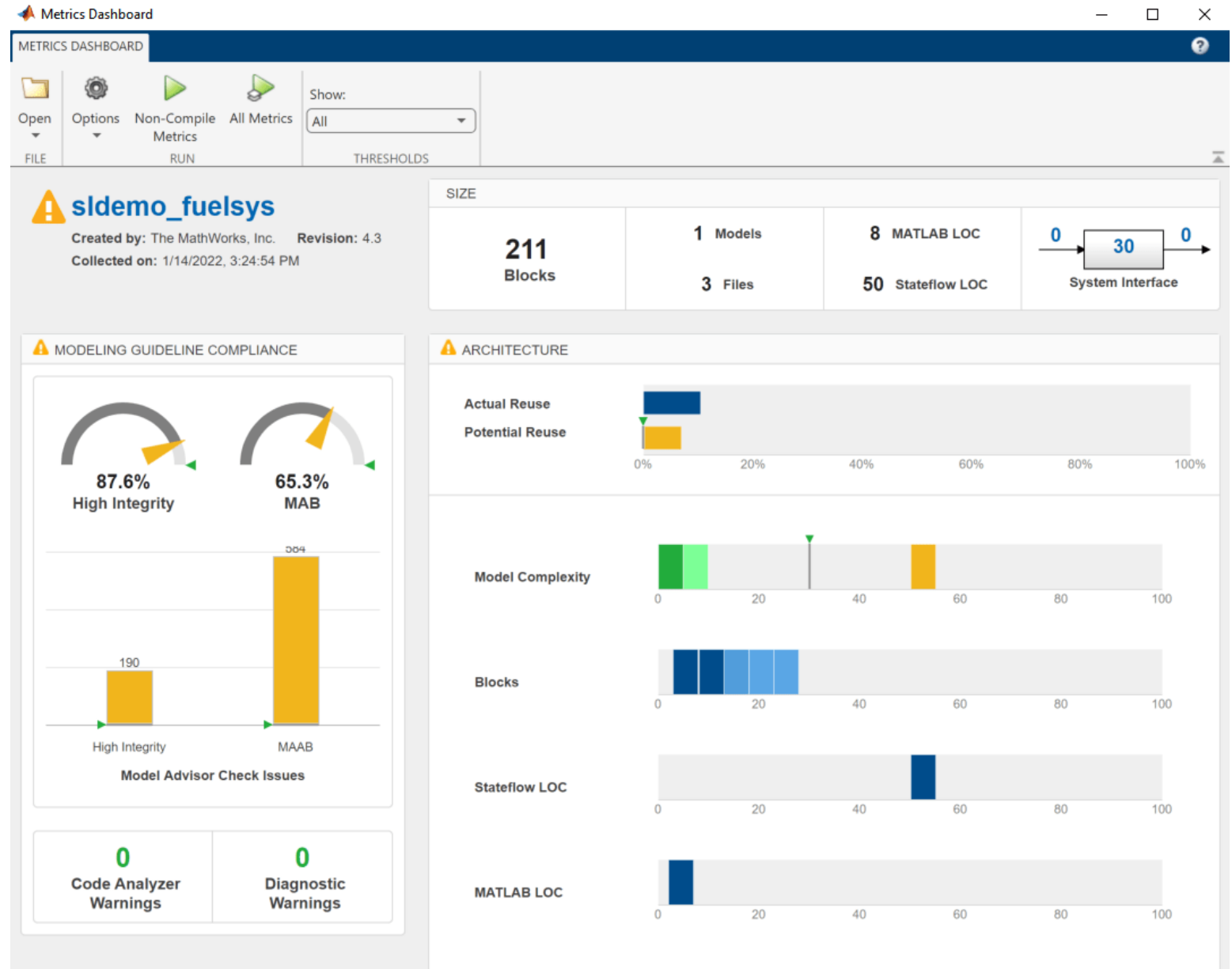
Shift Verification Earlier With Edit-Time Checking

- Highlight violations as you edit
- Fix issues earlier
- Avoid rework
- Author and customize edit-time checks

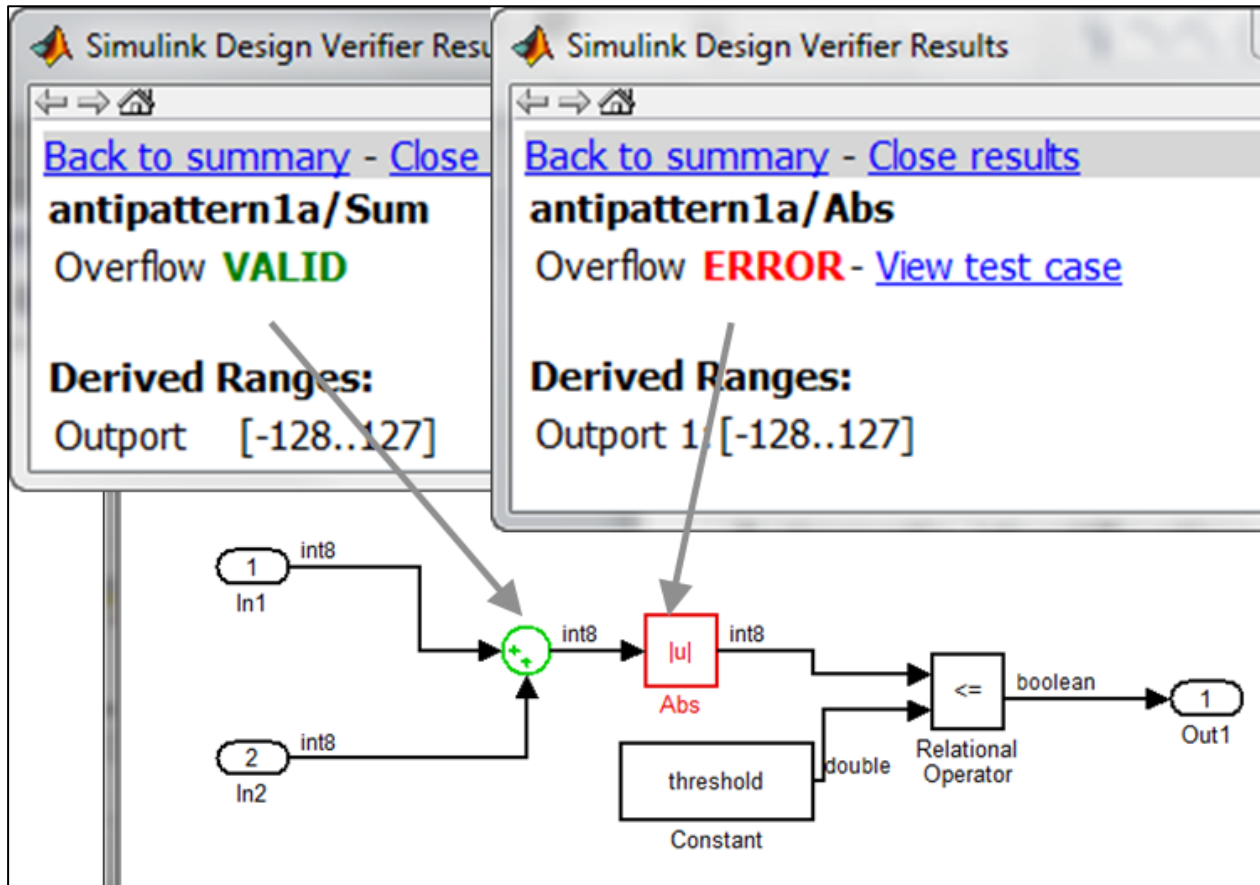


Assess Quality with Metrics Dashboard

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



Detect Design Errors with Formal Methods

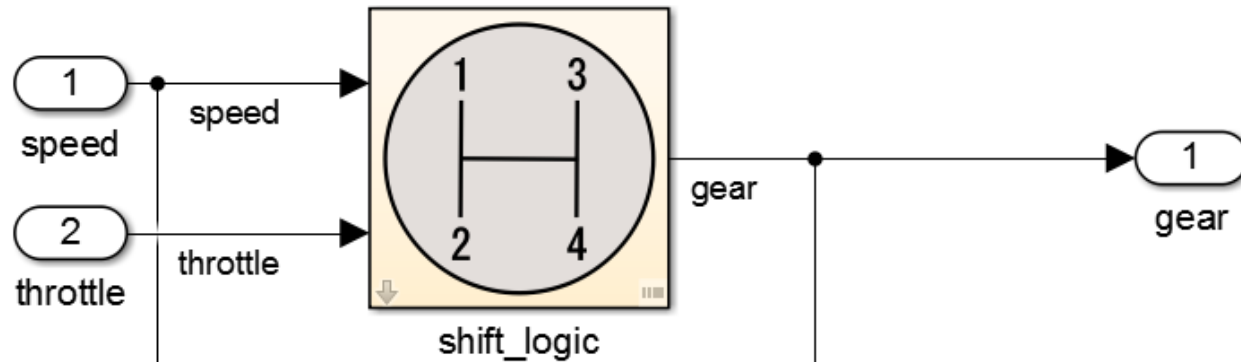


- Find run-time design errors:
 - Integer overflow
 - Dead Logic
 - Division by zero
 - Array out-of-bounds
 - Range violations

- Generate counter example to reproduce error

Prove That Design Meets Requirements

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



Checks that design meets requirements

- Gear 2 *always* engages when $5 \leq \text{speed} \leq 25$
- Gear 2 *never* engages when $\text{speed} < 5$ or $\text{speed} > 25$

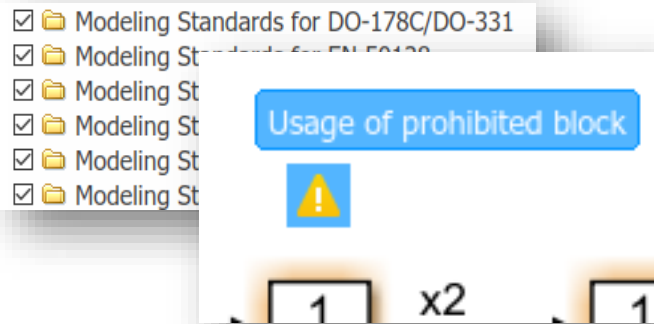
Expected behavior of design

Behavior that design should not exhibit

Reviews, Static Analysis and Formal Verification at the Model Level

Standards & Guidelines Checks

- **Automate** compliance to standards
- **Customize** checks
- **Find and fix** compliance issues while you design with Edit Time Checking



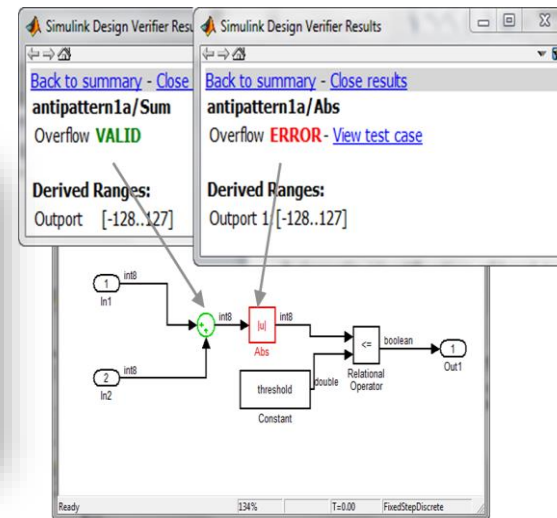
Model Metrics

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



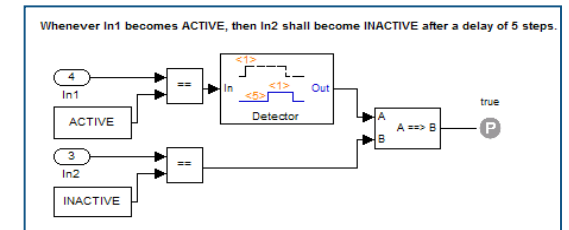
Design Error Detection

Uncover hard to find dead logic and design flaws

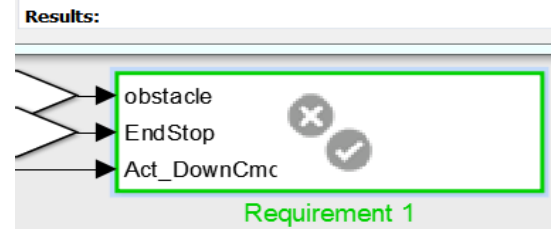


Property Proving

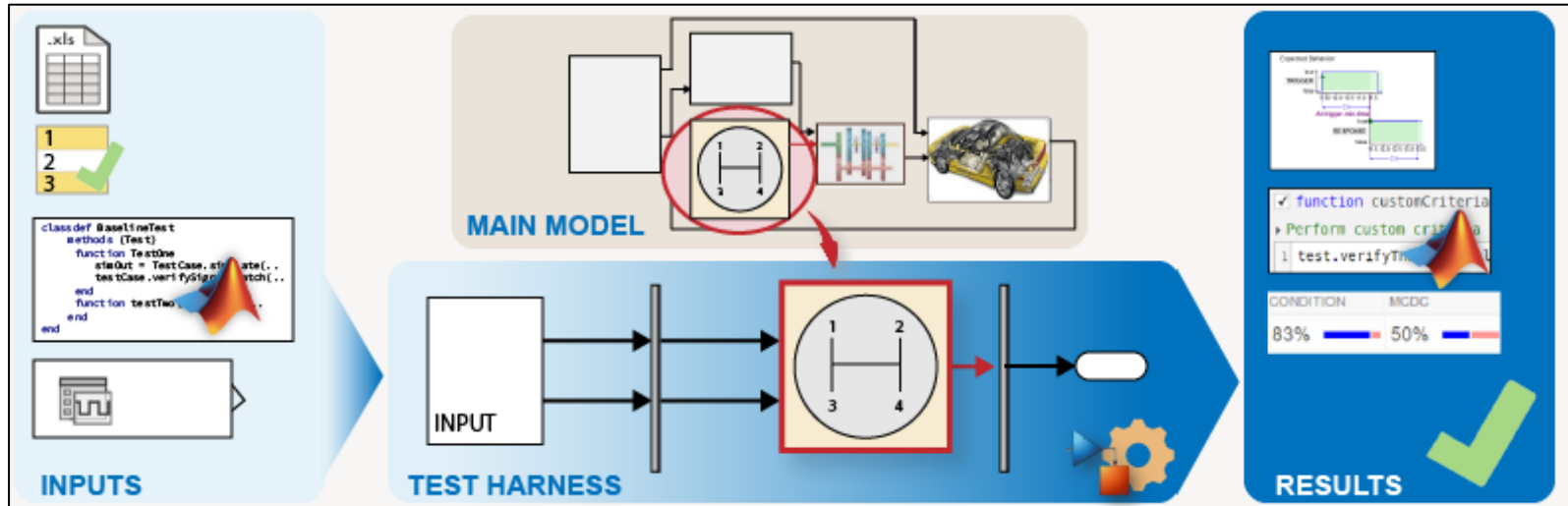
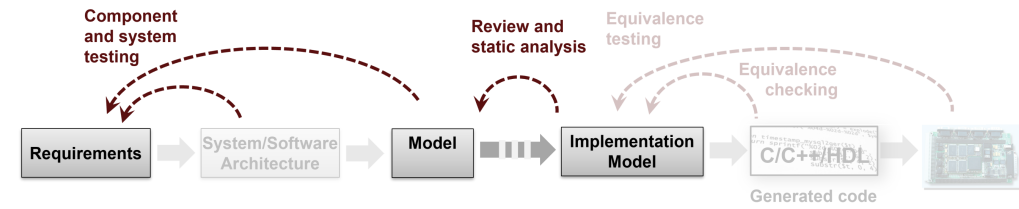
Prove design meets requirements



Close results
Property proving completed normally.
1/1 objective is proven valid.

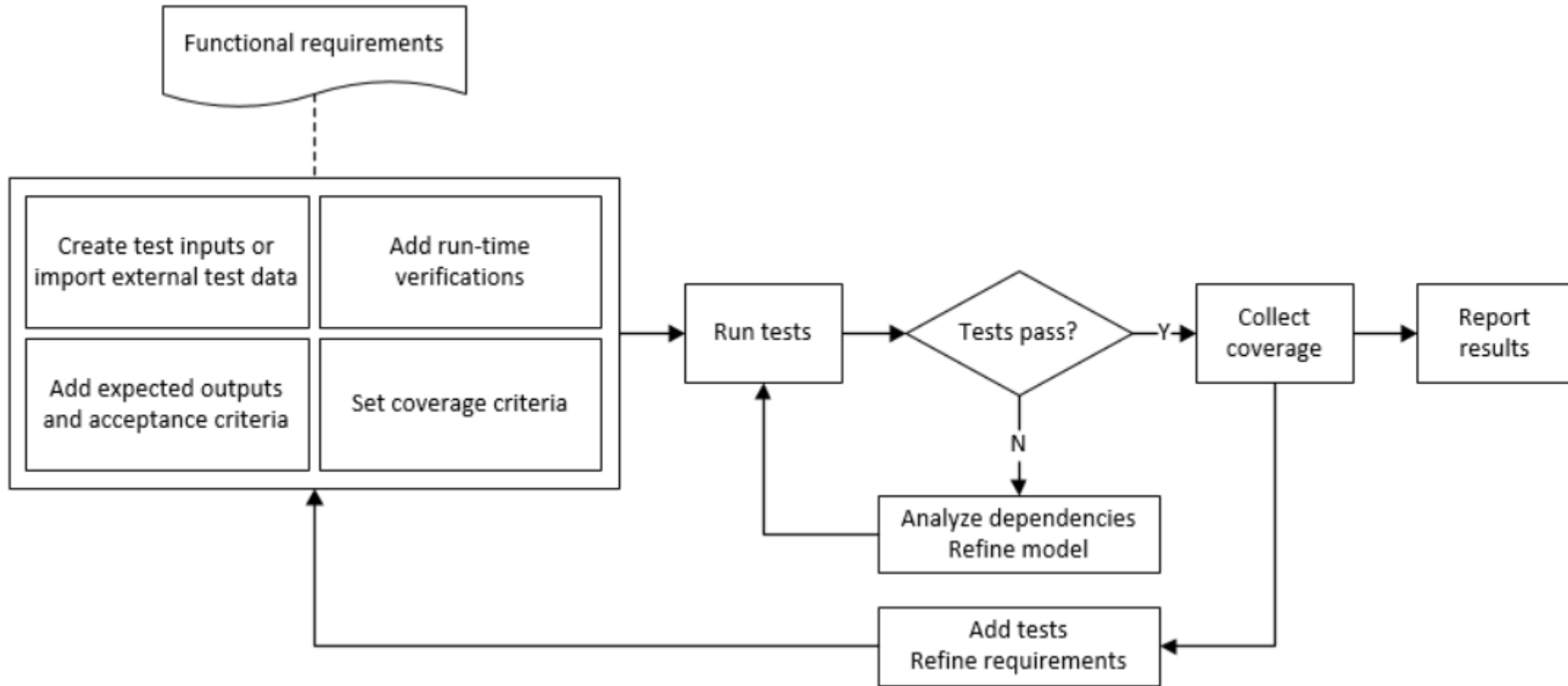


Systematic Functional Testing



Does the design meet requirements?
 Is it functioning correctly?
 Is it completely tested?

Typical Functional Testing Workflow in Model Based Design



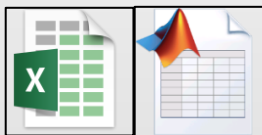
Systematic Functional Testing with Simulink Test

Test Case

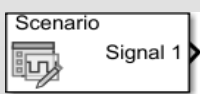


Model Sim through SIL, PIL and HIL
Scale with Parallel Computing Toolbox and Continuous Integration

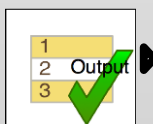
Inputs



Data file (input)



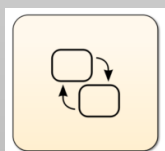
Signal Editor



Test Sequence

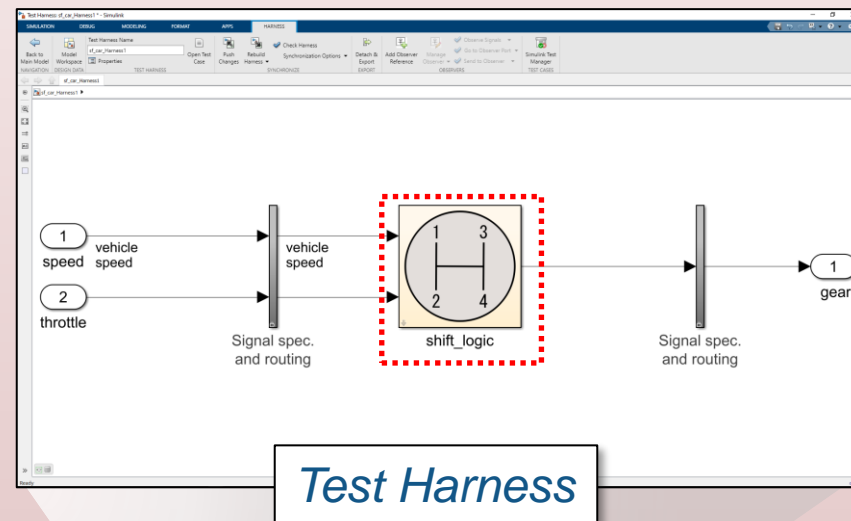
```
classdef BaselineTest < sltest.TestCase
    methods (Test)
        function testOne(testCase)
            simOut = testCase.simulate('TestExample');
            testCase.verifySignalsMatch(simOut, 'passell');
        end
        function testTwo(testCase)
        end
    end
end
```

MATLAB Code

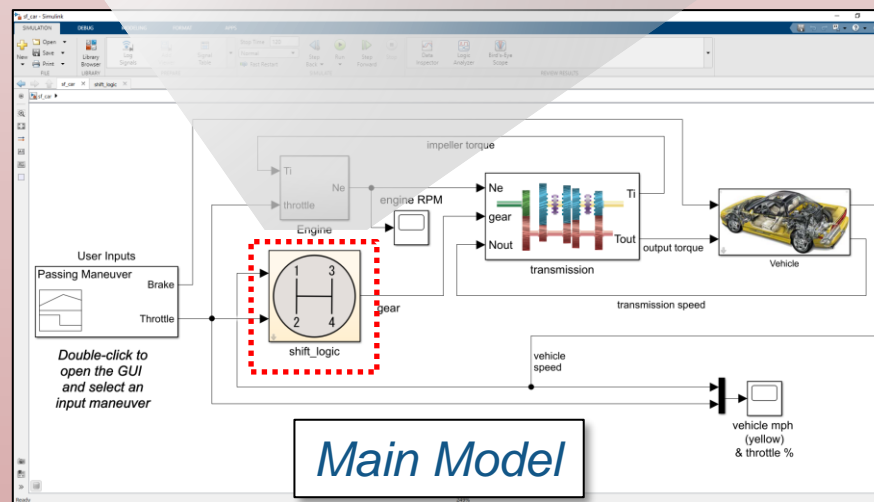


Stateflow

and more!

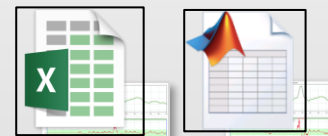


Test Harness

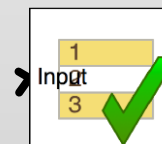


Main Model

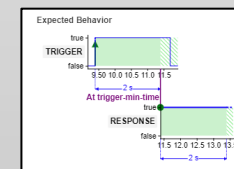
Assessments



Data file baseline)



Test Assessment



Temporal Assessment

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

MATLAB Code

and more!

Simulink Test

Develop, manage, and execute simulation-based tests

Test Authoring

- Specify test inputs, expected outputs, and tolerances
- Construct complex test sequences and assessments

Test Sequence

Step	Transition	Next Step
init_step speed = ramp (t); throttle = ramp (t);	1 after (2, sec)	step_2
step_2 speed = 2* ramp (t); throttle = 2* ramp (t);	1 gear == 3	step_3
step_3 peak_speed = speed; peak_throttle = throttle;		

Signal Editor

Scenario Signal 1

Temporal Assessments

Expected Behavior

TRIGGER: true (9.50 to 10.5), false (10.5 to 11.5)

RESPONSE: true (11.5 to 13.0), false (13.0 to 13.5)

At trigger-min-time: 2 s

Test Harnesses

- Isolate Component Under Test
- Synchronized, simulation test environment

Component under test

Main Model

Test Harness

Test Manager

- Author, manage, organize tests
- Execute simulation, equivalence and baseline tests
- Review, export, report

Test Manager

Test Browser

Test Results

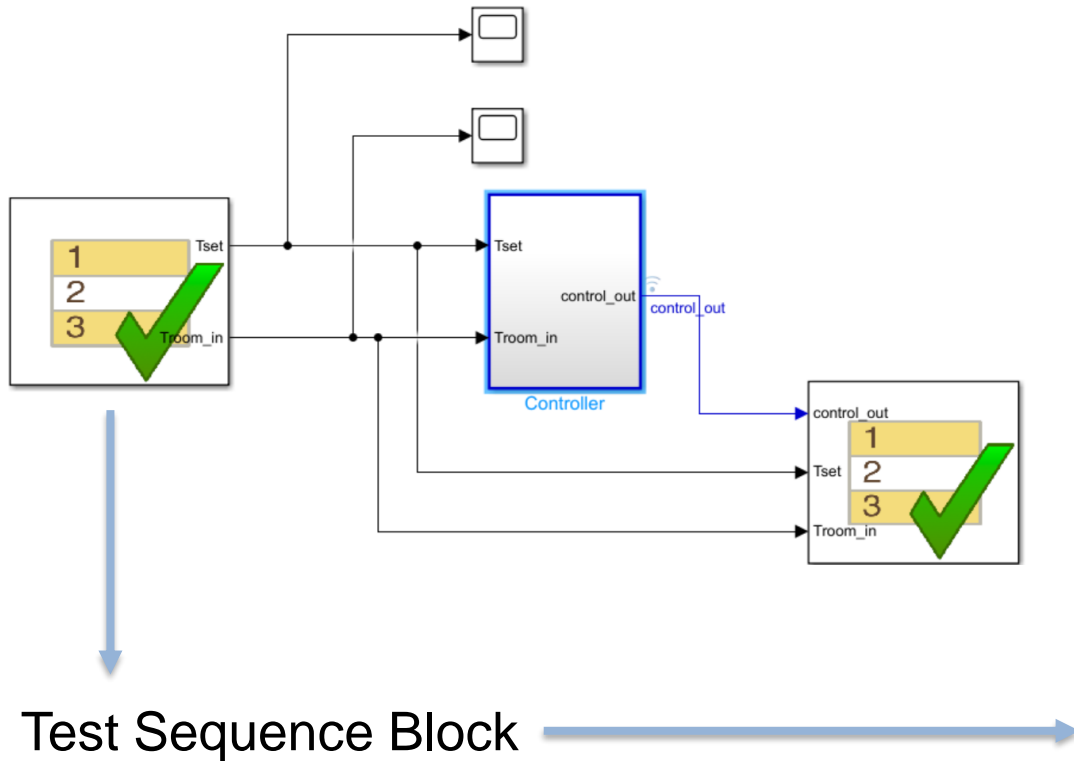
Reports

Report Generated by Test Manager

Title: LandingGearControl-Regression Tests
Author: Jessica Johnson
Date: 20-Feb-2015 18:28:22

Test Environment
Platform: PCWIN64
MATLAB: (R2015a)

Test Sequence Block: Step-based and temporal test sequences



sssc_house_heating_system_1_Harness1/Test Sequence - Test Sequence Editor

Step	Transition	Next Step
Initialize %% Initialize data inputs. Tset = 23; Troom_in = 23;	1. true	Cold_Outside
Cold_Outside %% Check heating mode Troom_in = 23 - ramp(et*0.2);	1. Troom_in <= 15	Hot_Outside
Hot_Outside %% Check cooling mode Troom_in = 23 + ramp(et*0.2);	1. Troom_in >= 27	Return_Idle
Return_Idle %% Return to idle mode Troom_in = Troom_in - ramp(et*0.2);	1. Troom_in <= 22	End
End Troom_in = 22		

Input
 1. control_out
Output
 1. Tset
 2. Troom_in
Local
Constant
Parameter
Data Store Memory

Step Hierarchy
 Initialize
 Cold_Outside
 Hot_Outside

Requirements Editor

File Edit Display Analysis Report Help

View: Requirements Search

Index	ID	Summary	Implemented	Verified
▼ BMS_Requirements				
> Import1	B...	References to BMS_Requireme...		
▼ StateMachine_Requirem...				
1	#...	Overview		
2	#7	Inputs		
3	#...	Outputs		
4	#...	State Machine Architecture		
4.1	#...	BMS State and Charging Mode ...		
4.2	#...	BMS Fault Monitoring		
4.2.1	#...	Current Limitation		
4.2.2	#...	Temperature Fault		
4.2.3	#...	Voltage Fault		
4.2.3.1	#...	Init State		
4.2.3.2	#...	No Voltage Fault State		
4.2.3.3	#...	Over Voltage Fault		
4.2.3.4	#...	Under Voltage Fault		
4.2.3.5	#...	Sensor Fault		
4.3	#...	Charger Contactors Management		
4.4	#...	Inverter Contactors Management		
5	#...	Justifications		

Properties

Type: Functional

Index: 4.2.3.4

Custom ID: #109

Summary: Under Voltage Fault

Description: In this state set:
- FaultPresent = true
- Fault_out.UnderVolt = 1;

Keywords:

Revision information:

Links

- Implemented by: UnderVoltageFault
- Verified by: Iteration9, Iteration9

Test Manager

TESTS

Test Browser Results and Artifacts

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

NAME	STATUS
▶ Results: 2020-Apr-26 10:19:25	9 2
▼ Results: 2020-Apr-26 10:40:10	11
▼ State_Machine_Harness_SignalB	11
▶ Iteration1	
▶ Iteration10	
▶ Iteration11	
▶ Iteration2	
▶ Iteration3	
▶ Iteration4	
▶ Iteration5	
▶ Iteration6	
▶ Iteration7	
▶ Iteration8	
▶ Iteration9	

PROPERTY	VALUE
Name	State_Machine_Harn...
Status	11
Start Time	04/26/2020 10:40:18
End Time	04/26/2020 10:41:12
Type	Simulation Test
Test File Location	C:\ML_WORKSPACE\de...
Test Case Definition	
Tags	

Results: 2020-Apr-26 10:40:10 x Start Page x Assessment Result x

▼ SUMMARY

Name	Results: 2020-Apr-26 10:40:10
Outcome	11
Start Time	04/26/2020 10:40:18
End Time	04/26/2020 10:41:12
Type	Result Set

▼ AGGREGATED COVERAGE RESULTS

Create a coverage report from coverage results to justify or exclude missing coverage. The filters and updated coverage values will be displayed with this result.

ANALYZED MODEL	REPORT	COMPLEXI...	DECISION	CONDITION	MCDC
State_Machine		101	87%	90%	73%

Scope coverage results to linked requirements

+ Add Tests for Missing Coverage Export

▼ COVERAGE FILTERS

NAME

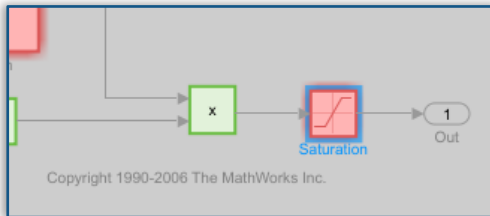
There are no coverage filter files applied.

Simulink Coverage

Measure test coverage in models and generated code

Model Coverage

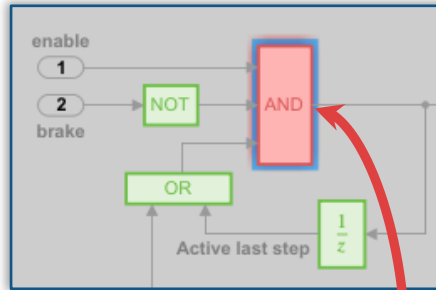
- Measure test completeness
- Identify missing tests or unintended functionality



Decisions analyzed	
input >= lower limit	50%
false	0/7
true	7/7
input > upper limit	100%
false	7/59
true	52/59

Generated Code Coverage

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

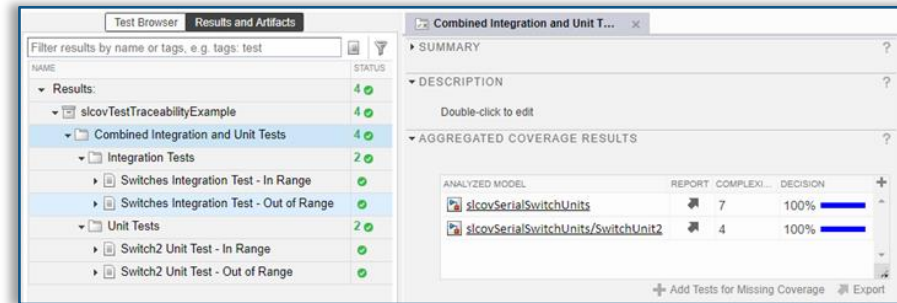


```

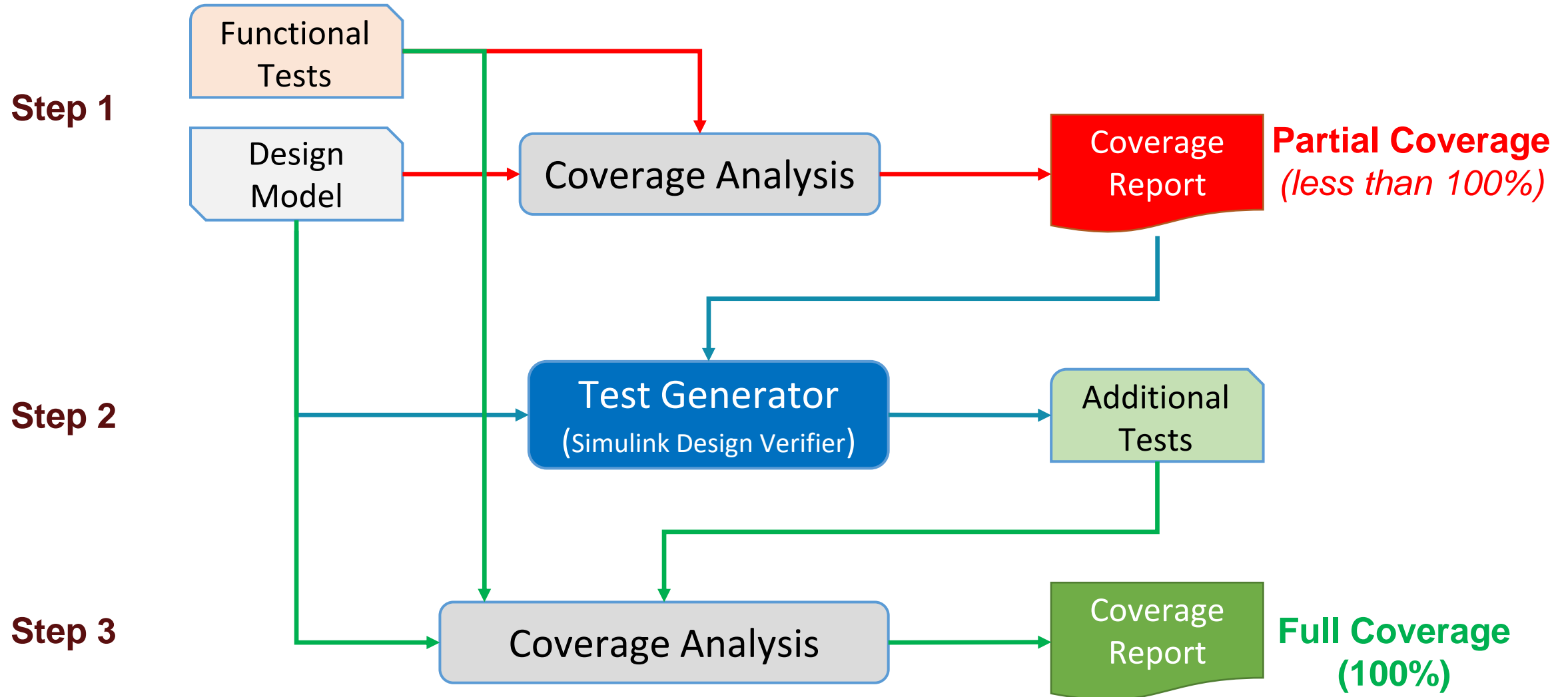
Coverage Details
49 localDW->UnitDelay1_DSTATE = (rtu_enable && (!rtu_brake)
50 && (rtu_set || localDW->UnitDelay1_DSTATE));
51
52 /* Sum: '<S1>/Sum' incorporates:
53 * Constant: '<S1>/Constant1'
54 * UnitDelay: '<S1>/Unit Delay'
55 */
56 rtb_Sum = localDW->UnitDelay_DSTATE + 1.0;
57
58 /* Switch: '<S1>/Switch3' incorporates:
  
```

Highlighting and Reporting

- View coverage results on diagrams
- Manage coverage results in **Simulink Test Manager**



Addressing Missing Coverage



Automatically Address Missing Coverage

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

- View coverage results in the Test Manager for existing tests
- Select coverage results and click Add Tests for Missing Coverage

▼ AGGREGATED COVERAGE RESULTS

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

Add Tests for Missing Coverage Report

Demo: Generate Tests for Coverage from Test Manager

The screenshot displays the MATLAB Test Manager application window. The interface includes a menu bar with options like New, Open, Save, Copy, Delete, Run, Stop, Debug, Parallel, Report, Visualize, Highlight in Model, Import, Export, Preferences, and Help. Below the menu bar, there are tabs for 'Test Browser' and 'Results and Artifacts'. The 'Test Browser' panel shows a tree view of test cases under 'RollRefTest', with 'Logged Data and Coverage' selected. A search filter is present at the top of the browser. The 'Properties' panel at the bottom left shows details for the selected test case.

PROPERTY	VALUE
Name	Logged Data and Cove...
Location	C:\Program Files\MATLAB\...
Hierarchy	RollRefTest » Logged Data ...
Enabled	<input checked="" type="checkbox"/>
Tags	Type comma or space separai

Simulink Test Manager – Integrates MathWorks V&V tools

Simulink Requirements

Simulink Check

TEST CASE ANALYSIS

Requirements Linked To Tests: 42.9% Unlinked (12)

Tests Linked to Requirements: 87.5% Unlinked (1)

Tests per Requirement: 0 to 12

Requirements per Test: 0 to 5

TEST RESULT ANALYSIS

Model Test Status: 75% Passed, 1 Failed, 0 Untested, 1 Disabled

Model Coverage: Achieved, Justified, No data available

Type	Test Cases
Simulation	8
Equivalence	0
Baseline	0

Type	Test Cases
Simulation	8
Equivalence	0
Baseline	0

Tag	Test Cases
1_Draft	1
2_Under_review	2
3_Reviewed	2
4_Released	3

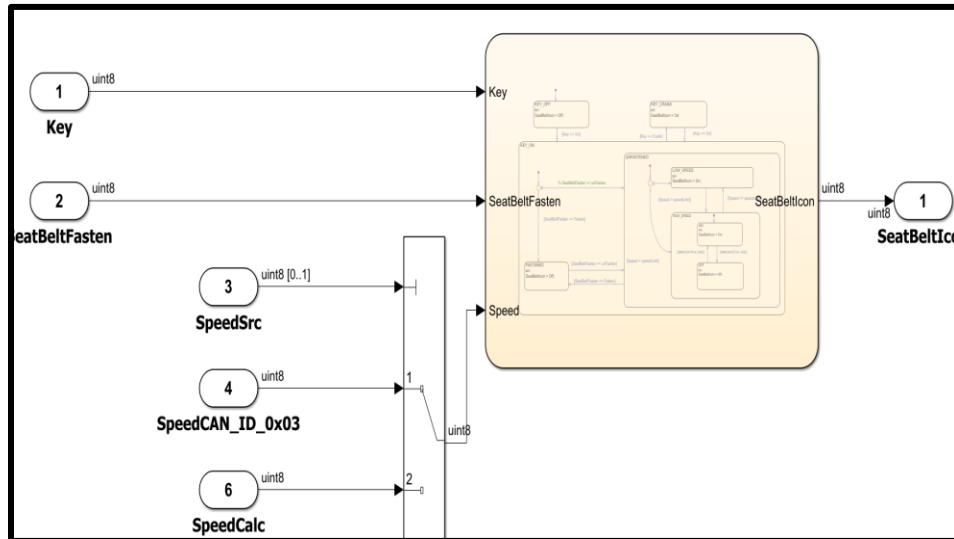
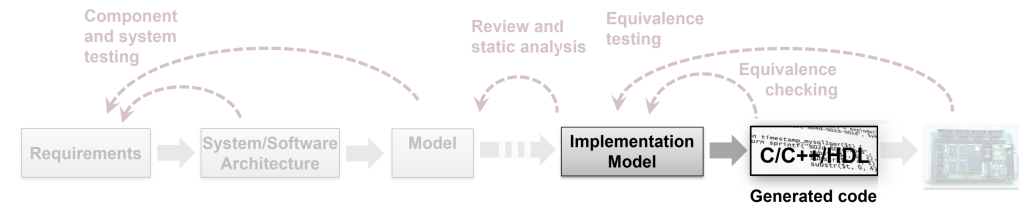
Quality

+

Export

Automatic Code Generation

Reliable and high performance, with flexible choice of targets



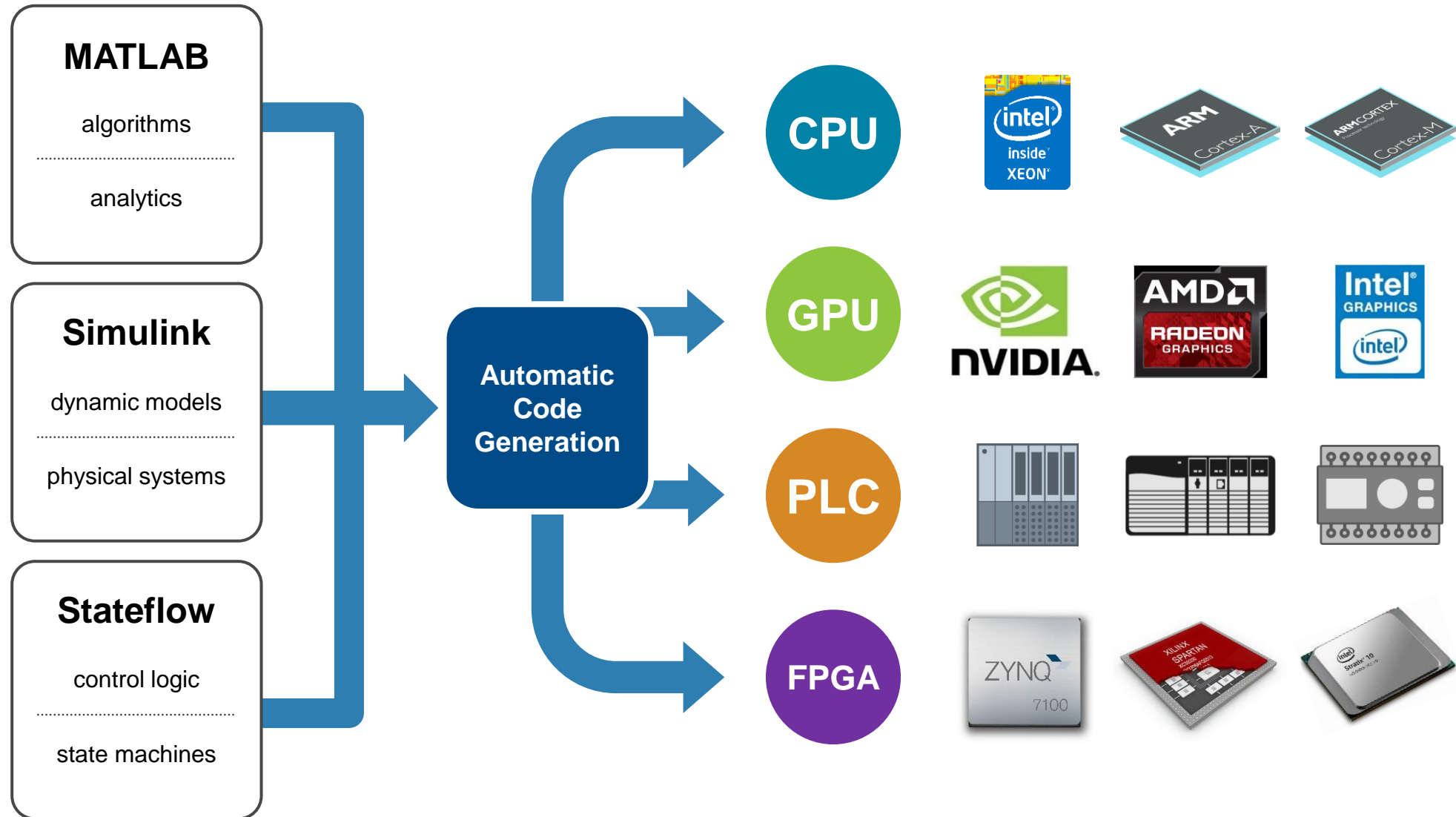
Simulink Model
Application Logic



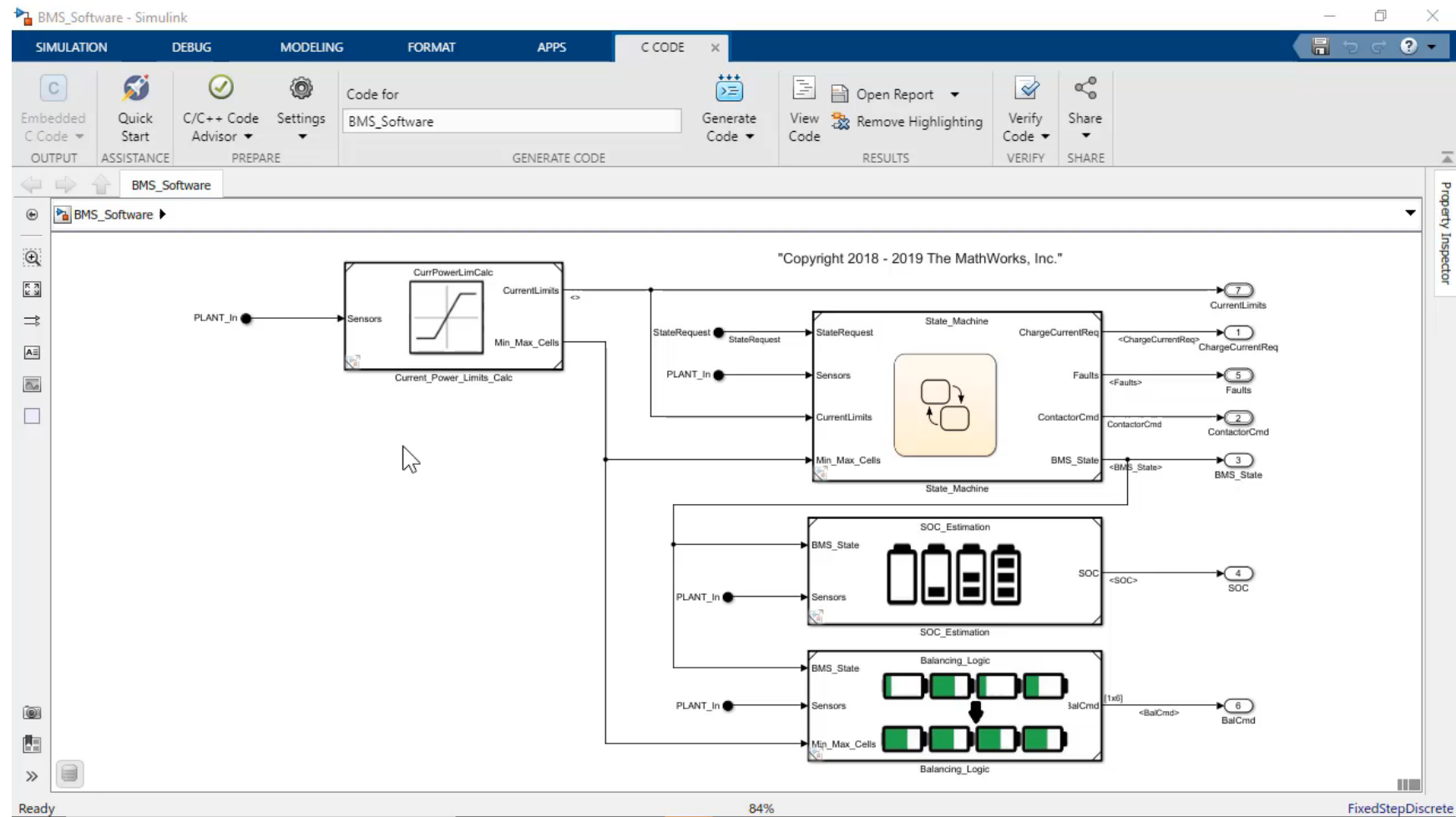
Efficient
C/C++

Automatic Code Generation

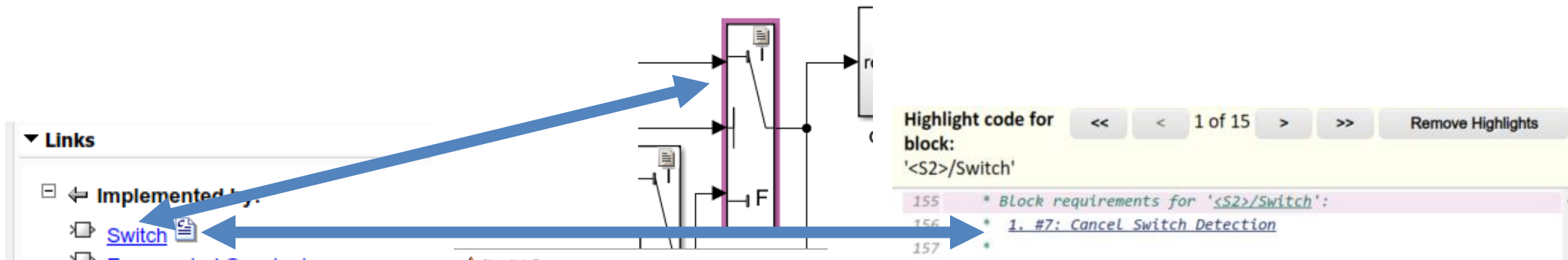
Reliable and high performance, with flexible choice of targets



Automatic Code Generation



Requirements Traceability to Model, Code and Test Cases



Links

- Implemented by:
 - Switch
 - Enumerated Constant
- Verified by:
 - Cancel button ✘
- Derived from:
 - Disabling cruise control

Requirements

Simulink Test

TESTS

Test Browser Results and Artifacts

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

Results: 2017-Aug-01 18:08:34

- Unit test for DriverSwRequest
 - Enable button ✔
 - Cancel button ✘
 - Set button ✔
 - Resume button ✔
 - Increment button short ✔

Tests & Test Results

Generated Code

Traceability Matrix

HOME

Expand All Collapse All Highlight Missing Links Create Link

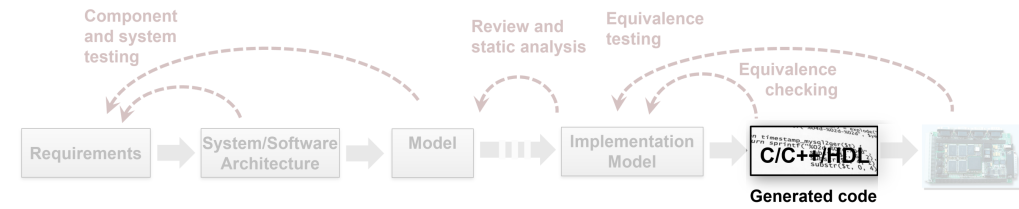
Simulink Requirements vs Simulink Model

Type	Component	Link
Component	scExampleSmallUAVModel	Functional
Port	scExampleSmallUAVModel	Functional
Link	Supervisory Computer	Functional
Link	Flight Support Components	Functional
Link	Flight Computer	Functional
Link	Main Board	Functional
Link	Protective Case	Functional
Link	Telemetry Antenna	Functional
Link	Airframe	Functional
Link	Payload	Functional
Link	Payload	Functional
Link	Payload	Functional
Link	Payload Retraction Sy	Functional
Link	Payload Data Link	Functional
Link	Propulsion	Functional
Link	Power Module	Functional
Link	Prop	Functional
Link	Engine	Functional
Link	Fuel System	Functional
Link	Fuel Level Sensors	Functional
Link	Fuel Pump	Functional
Link	Fuel Tank	Functional
Link	Pressure Regulator	Functional
Link	Fuel Filter	Functional

Left #12 Flight Control
Top Telemetry Antenna
Link None (Create)

	B	C	D	E	F	G	H	I	J	K	L	M	N
	Model Object	Model Object	Model Object	Model Object	Model Object	Model Optimization	Code File	Code Function	Code Line Number	Code File Location	Code Comment Checksum	Requirements Source	Requirements Location
1	Type	Path	Subsystem	SID	Optimized	Rationale	Name						
2	Inport	model1	model1	model1:4			model1.c	Global	22	C:\work\model1_ert_rtw	CS_533947273	C:\work\req1.doc	@ID_0123: Input Interface
3	Inport	model1	model1	model1:4			model1.c	model1_step	38	C:\work\model1_ert_rtw	CS_116158590	C:\work\req1.doc	@ID_0123: Input Interface
4	Inport	model1	model1	model1:4			model1.c	model1_step	43	C:\work\model1_ert_rtw	CS_116158590	C:\work\req1.doc	@ID_0123: Input Interface
5	Inport	model1	model1	model1:4			model1.h	Global	49	C:\work\model1_ert_rtw	CS_533947273	C:\work\req1.doc	@ID_0123: Input Interface
6	Gain	model1	model1	model1:2			model1.c	Global	23	C:\work\model1_ert_rtw	CS_405027874	C:\work\req1.doc	@ID_0005: Functionality
7	Gain	model1	model1	model1:2			model1.c	Global	27	C:\work\model1_ert_rtw	CS_104796431	C:\work\req1.doc	@ID_0005: Functionality

Traceability Matrix



Static Code Analysis with Polyspace

The screenshot shows the Polyspace interface with a Simulink model on the left and a code editor on the right. The code editor displays C code with annotations for requirements and transitions. A table at the bottom lists analysis results for various code elements.

File	Class	Function	Status	Severity	Comment
controller.c	Global Scope	controller_en...	Unreviewed	Unset	
controller.c	Global Scope	controller_en...	Unreviewed	Unset	
sr_s32.c	Global Scope	sr_s32()	Unreviewed	Unset	
sr_s32.c	Global Scope	sr_s32()	Unreviewed	Unset	
sr_s32.c	Global Scope	sr_s32()	Unreviewed	Unset	

Polyspace is independent of the origin of code

```
#include <assert.h>
int speed(int k)
{
    int i,j,v;
    i = 2;
    j = k+5;
    while (i < 10) {
        i++;
        j+=3;
    }
    return 1 / (i-j);
}
```

Hand Code

C, C++

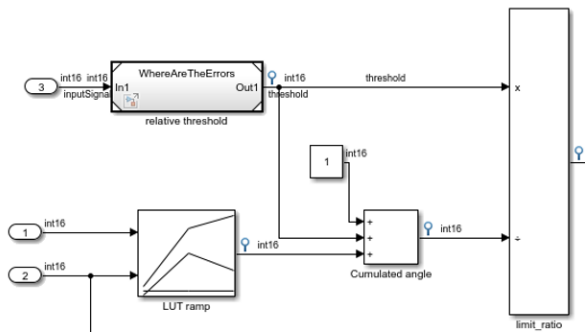
Polyspace

Violations
Defects
Runtime errors
Reports

C, C++

Model-Based Design
(MATLAB, Simulink, Stateflow)

Model-Based V&V tools
Code Generation tools



Can you find a bug?

```
1  int new_position(int sensor_pos1, int sensor_pos2)
2  {
3  int actuator_position;
4  int x, y, tmp, magnitude;
5
6  actuator_position = 2; /* default */
7  tmp = 0;               /* values */
8  magnitude = sensor_pos1 / 100;
9  y = magnitude + 5;
10
11 while (actuator_position < 10)
12 {
13     actuator_position++;
14     tmp += sensor_pos2 / 100;
15     y += 3;
16 }
17 if ((3*magnitude + 100) > 43)
18 {
19     magnitude++;
20     x = actuator_position;
21     actuator_position = x / (x - y);
22 }
23 return actuator_position*magnitude + tmp; /* new value */
24 }
```

Could there be a bug on this line?

Consider the operation: $x / (x - y)$

Potential run-time errors

- Variables x and y may not be initialized
- An overflow on subtraction
- If $x == y$, then a divide by zero will occur

How to prove that run-time errors do or do not exist?

Static Code Analysis with Polyspace

- Code metrics and standards
 - Comment density, cyclomatic complexity,...
 - MISRA and Cybersecurity standards
 - Support for DO-178, ISO 26262,
- Bug finding and code proving
 - Check data and control flow of software
 - Detect bugs and security vulnerabilities
 - Prove absence of runtime errors

Green: reliable
safe pointer access

Red: faulty
out of bounds error

Gray: dead
unreachable code

Orange: unproven
may be unsafe for some conditions

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

Range data
tool tip

```

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

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

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

    i = get_bus_status();

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

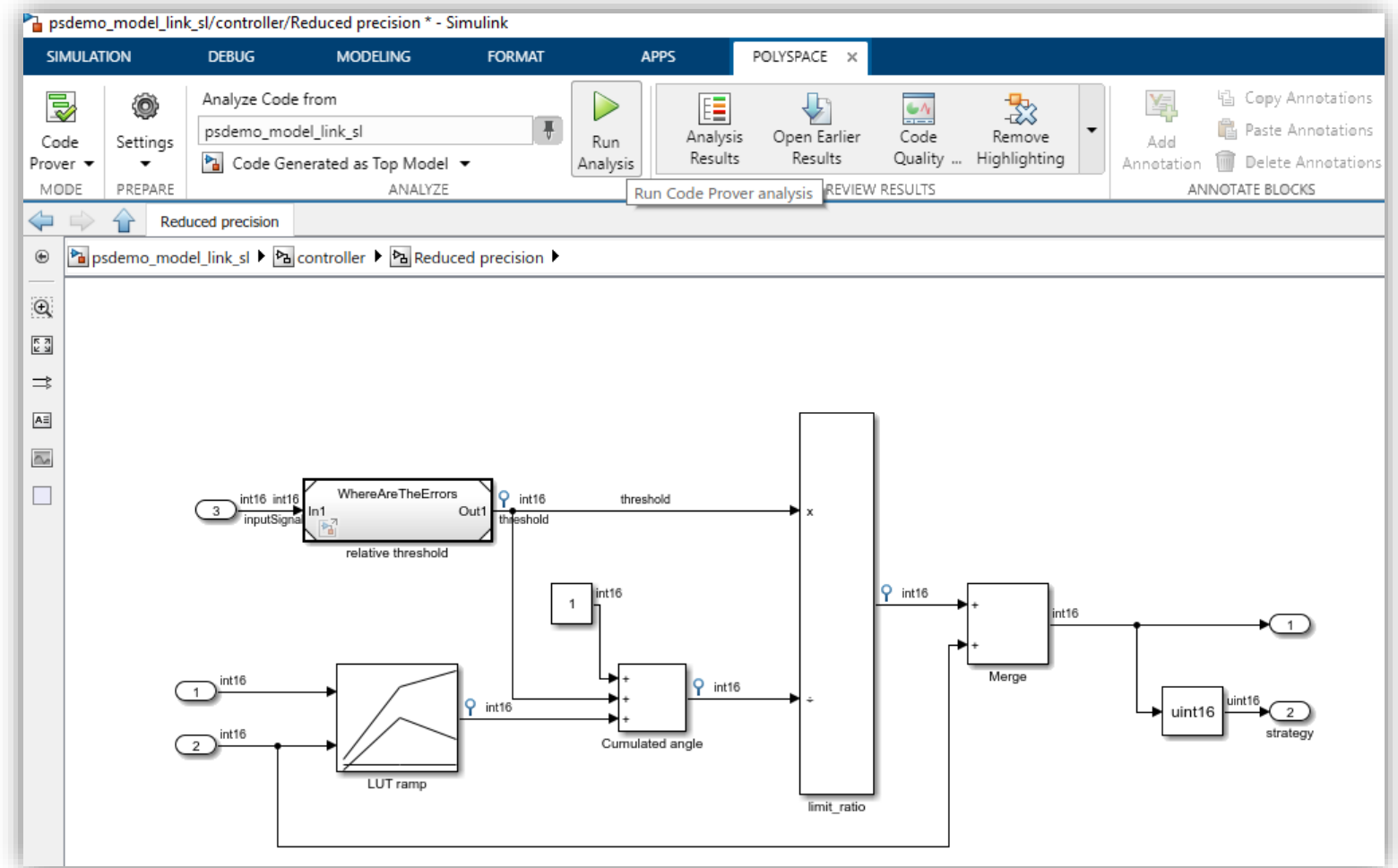
```

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

Results from Polyspace Code Prover

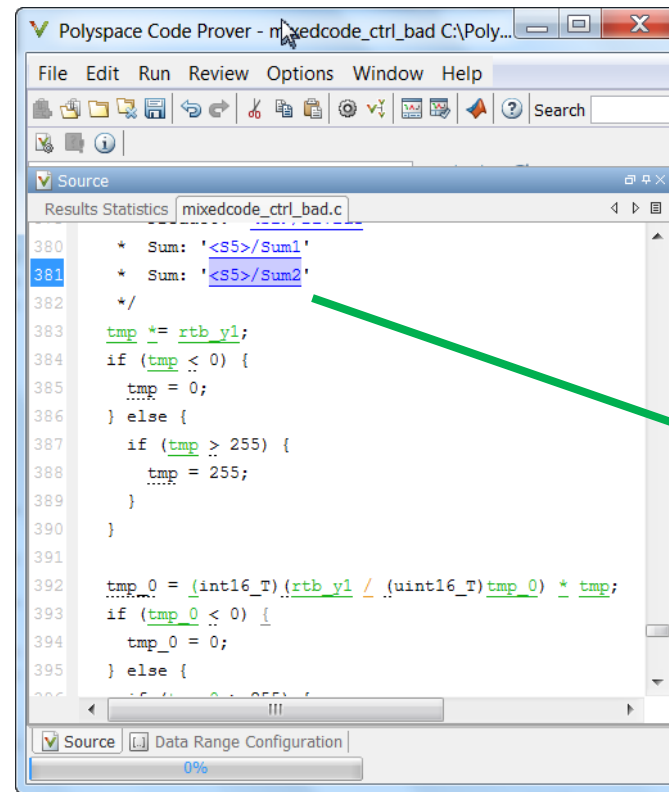
Polyspace is Integrated with Simulink

1. **Launch** Polyspace from Simulink

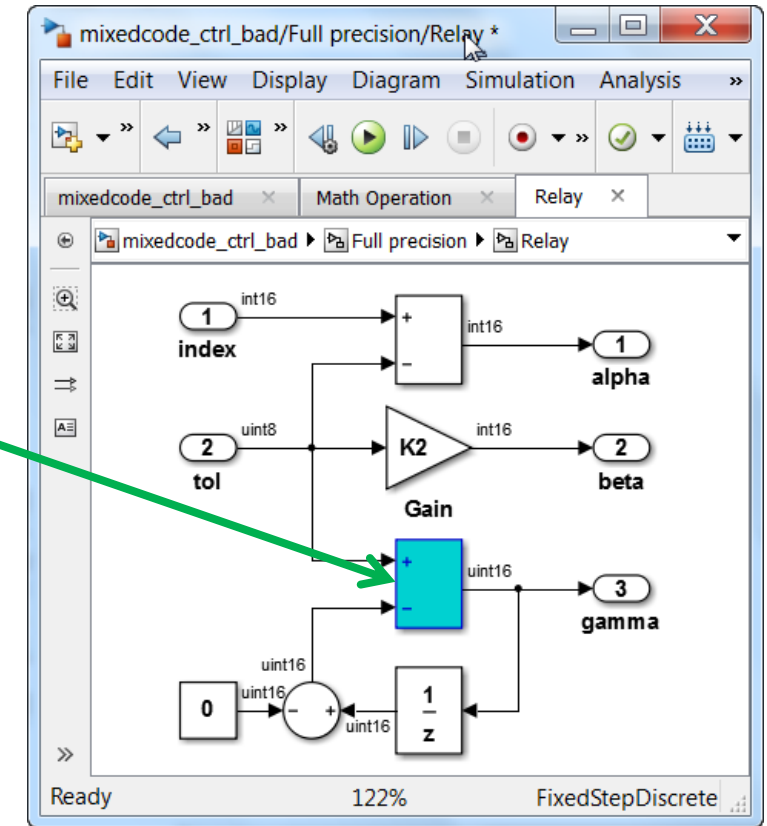


Polyspace is Integrated with Simulink

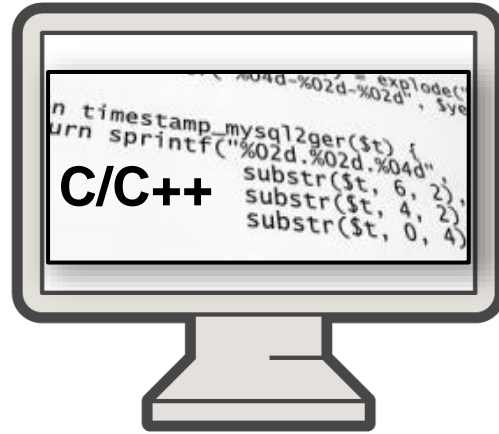
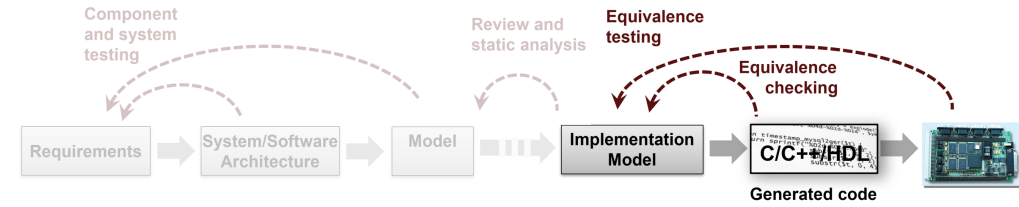
1. **Launch** Polyspace from Simulink
2. **Navigate** from Code to Model



```
380 * Sum: '<S5>/Sum1'  
381 * Sum: '<S5>/Sum2'  
382 */  
383 tmp *= rtb_y1;  
384 if (tmp <= 0) {  
385     tmp = 0;  
386 } else {  
387     if (tmp >= 255) {  
388         tmp = 255;  
389     }  
390 }  
391  
392 tmp_0 = (int16_T)(rtb_y1 / (uint16_T)tmp_0) * tmp;  
393 if (tmp_0 < 0) {  
394     tmp_0 = 0;  
395 } else {  
396     if (tmp_0 >= 255) {  
397         tmp_0 = 255;  
398     }  
399 }
```



Equivalence Testing



**Software in the Loop
(SIL)**

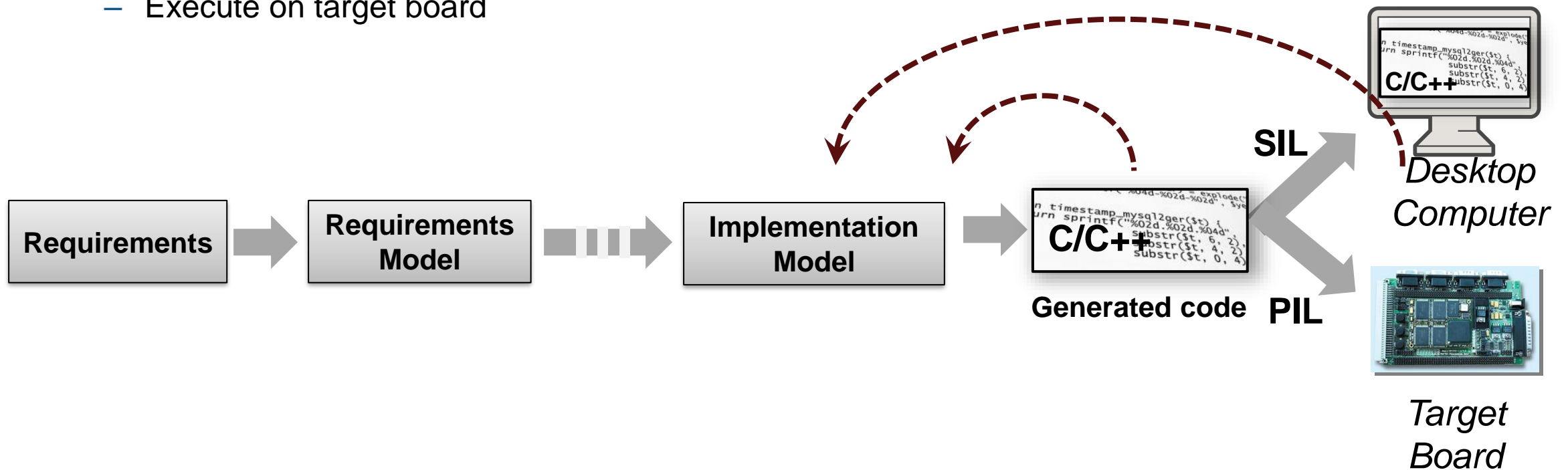


**Processor in the Loop
(PIL)**

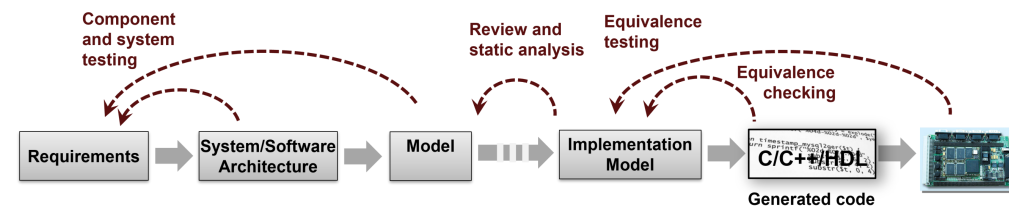
Is the code functionally equivalent to model?
Is all the code tested?

Equivalence Testing

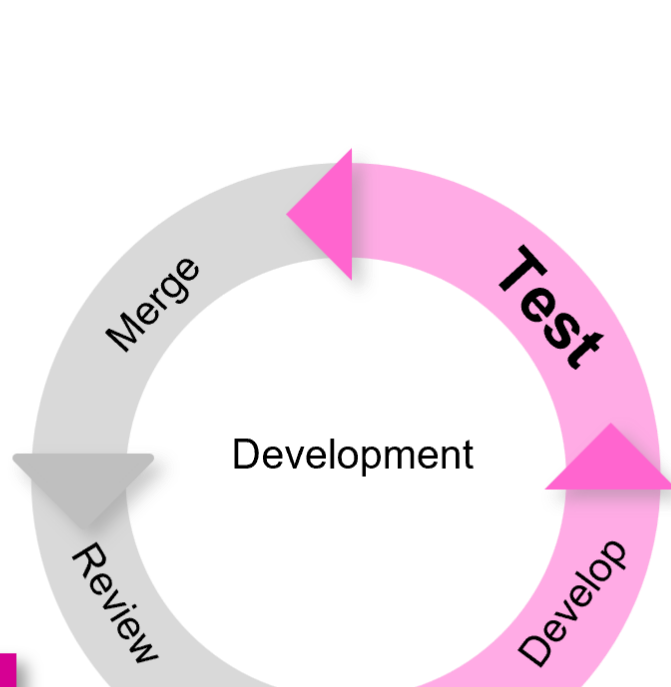
- Software in the Loop (SIL)
 - Show functional equivalence, model to code
 - Execute on desktop / laptop computer
 - Processor in the Loop (PIL)
 - Numerical equivalence, model to target code
 - Execute on target board
- Re-use tests developed for model to test code
 - Collect code coverage



High Integrity Verification Workflow



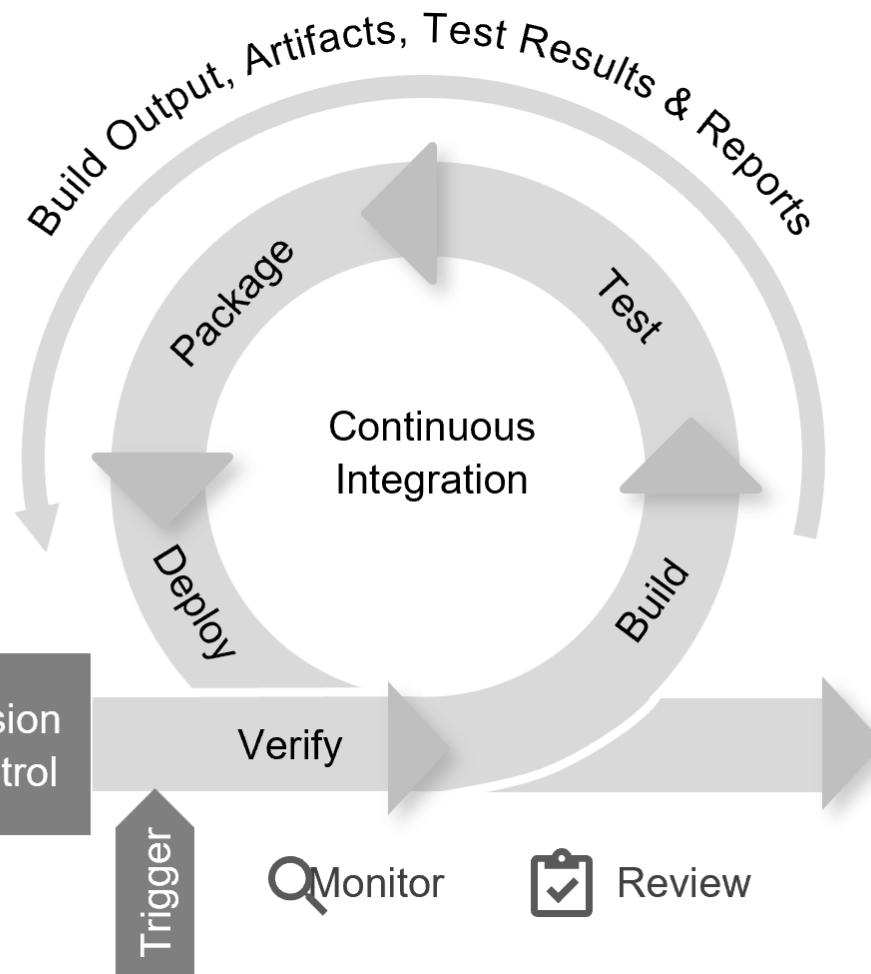
2 Test



Developers & Test Authors

Submit

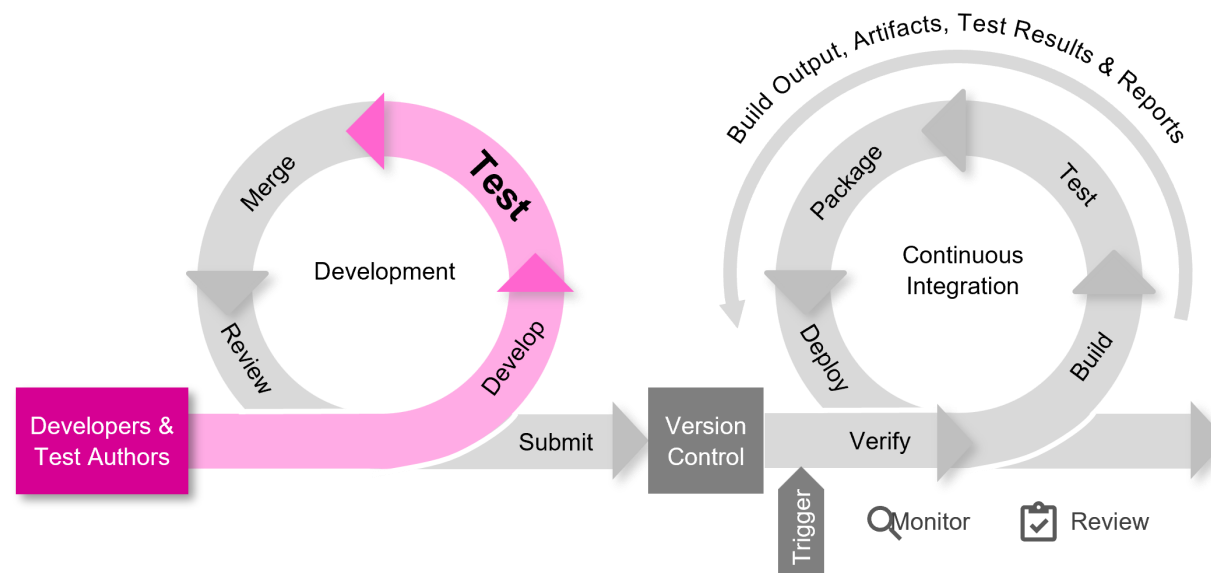
Version Control



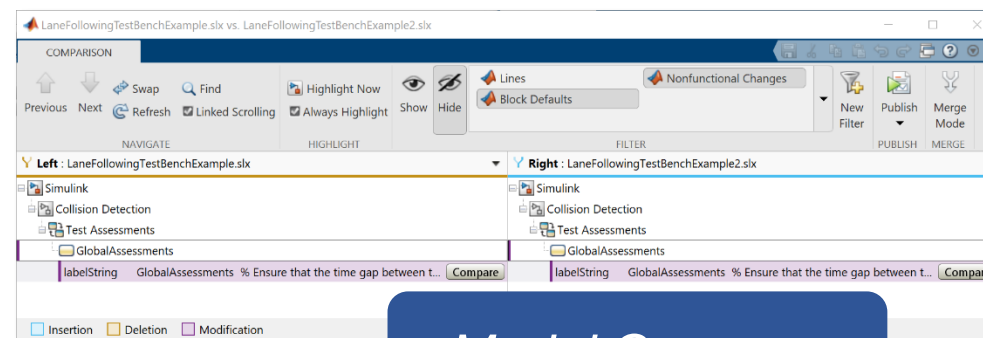
Monitor

Review

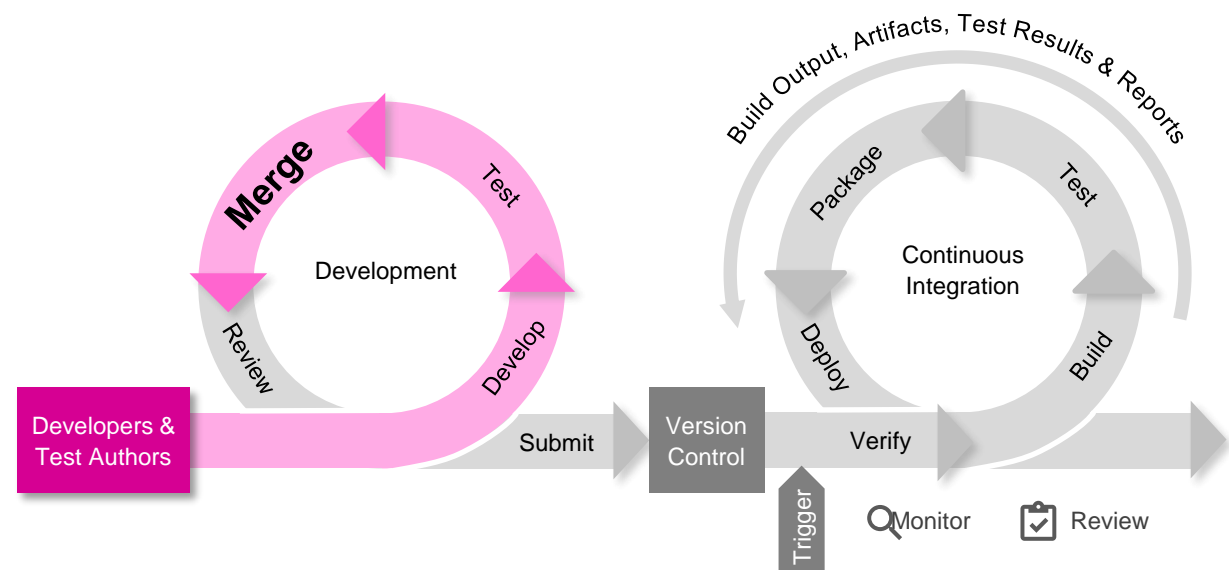
Continuous Integration Workflow with Model-Based Design



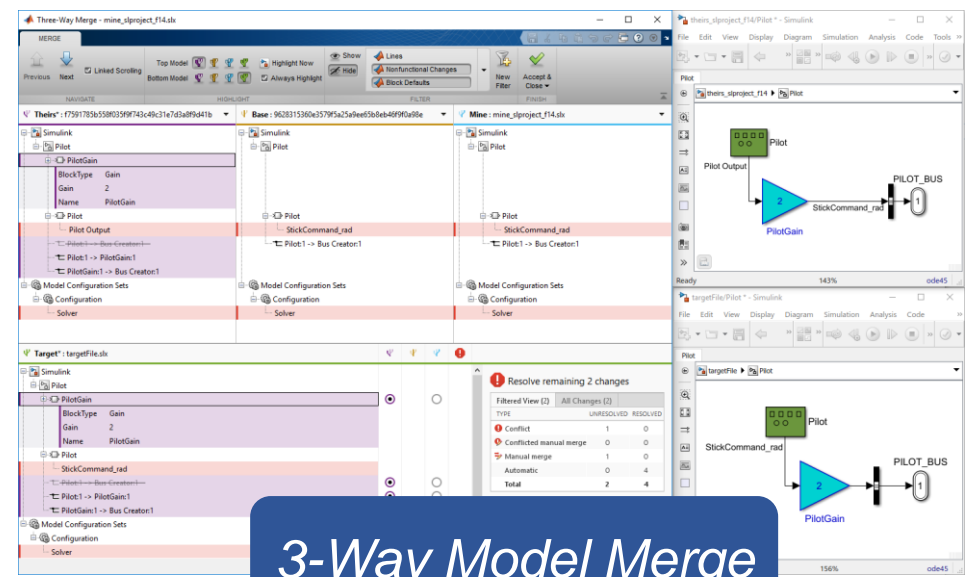
3 Merge



Continuous Integration Workflow with Model-Based Design

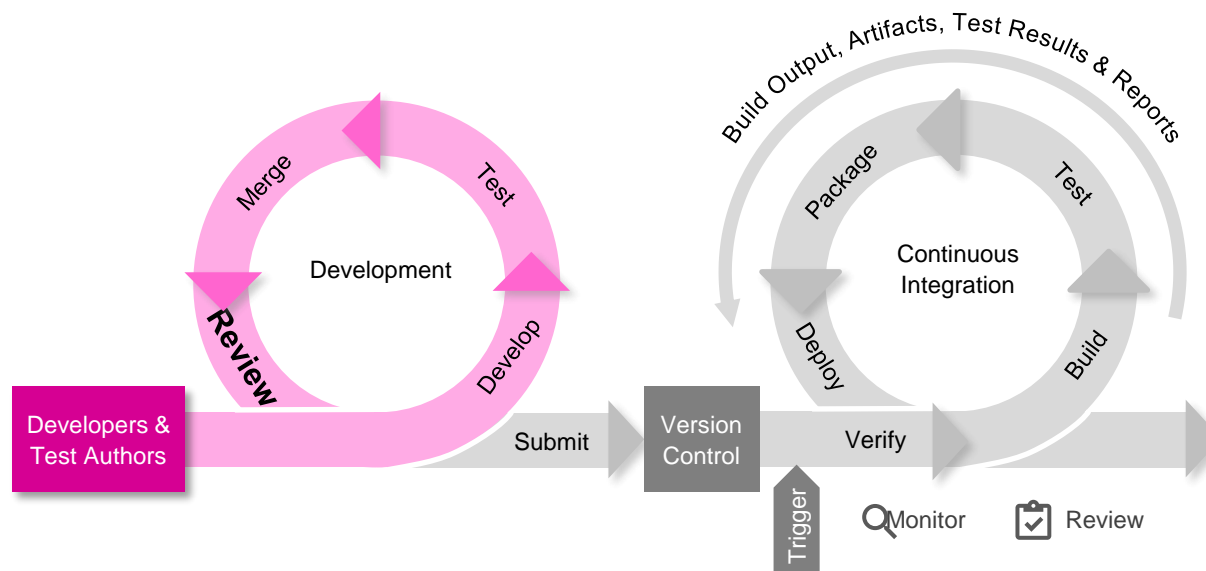


3 Merge



3-Way Model Merge

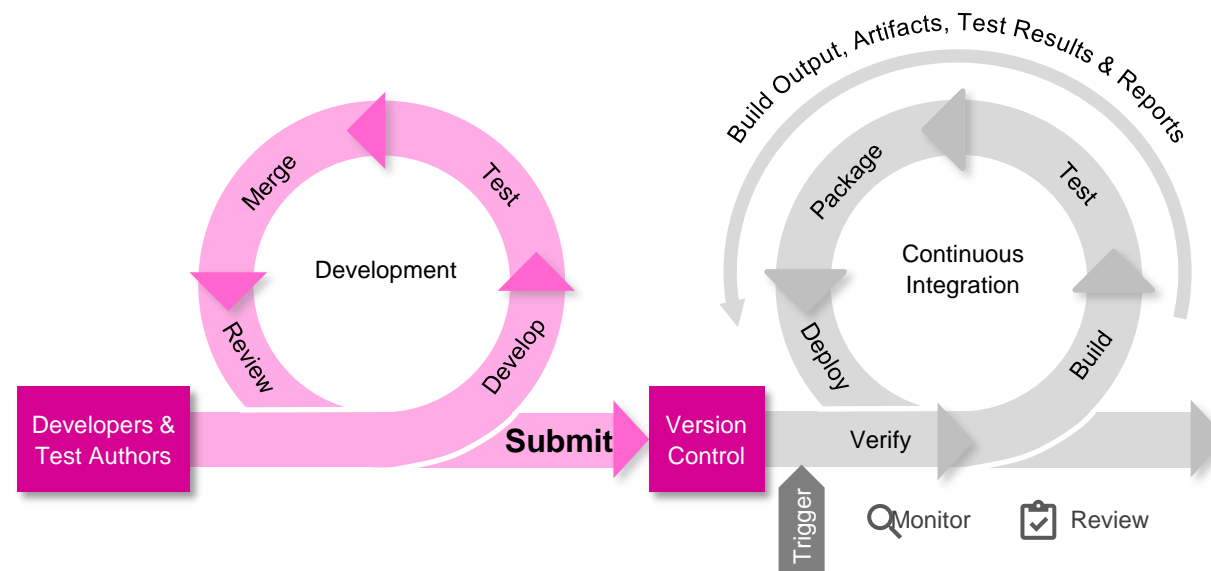
Continuous Integration Workflow with Model-Based Design



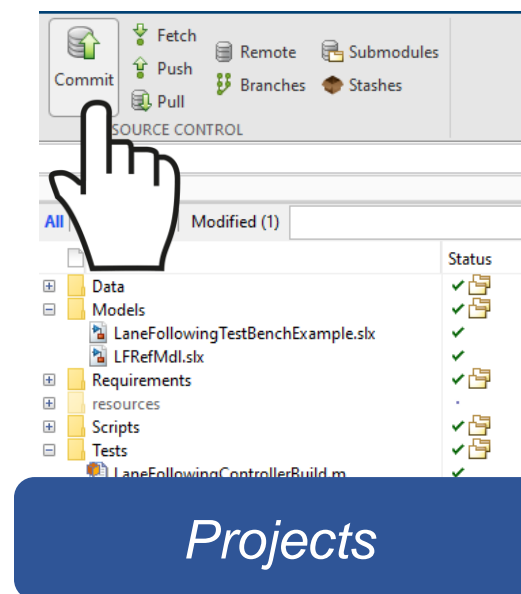
4 Review

Compare Report

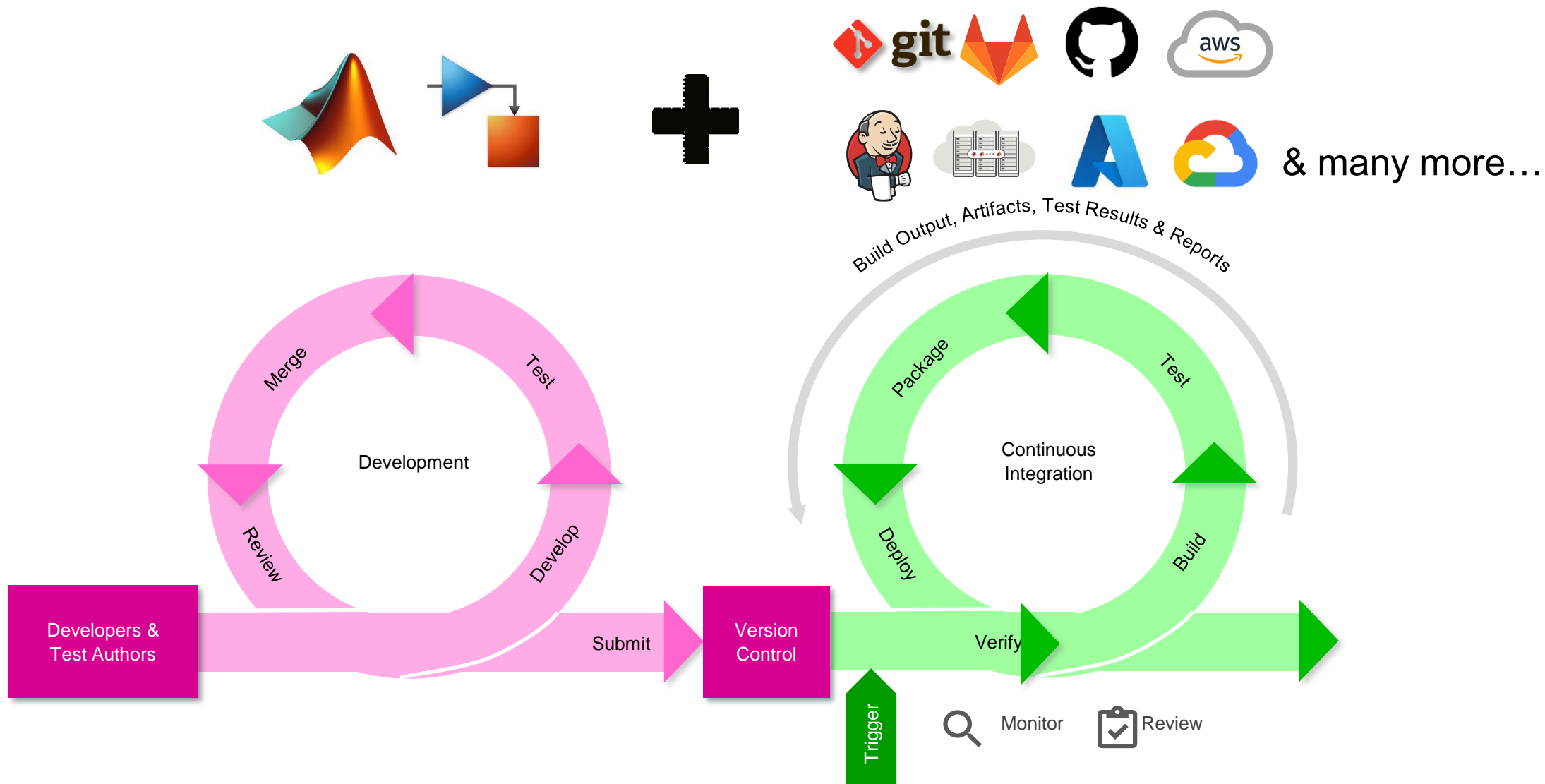
Continuous Integration Workflow with Model-Based Design



5 Submit



From Desktop to Cloud for Model Based Design using CI/CD



**What are the benefits
of CI?**

**What problems does
it attempt to solve?**

**How does Simulink fit
into the CI ecosystem?**

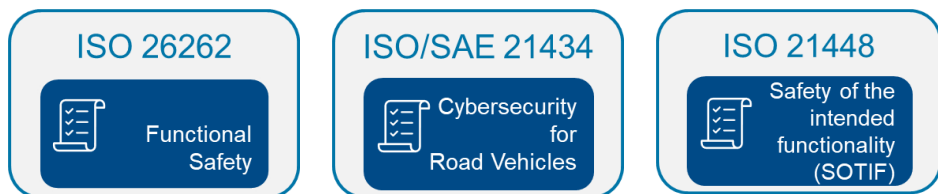
**How to best leverage
CI with MBD?**

Benefits of Continuous Integration



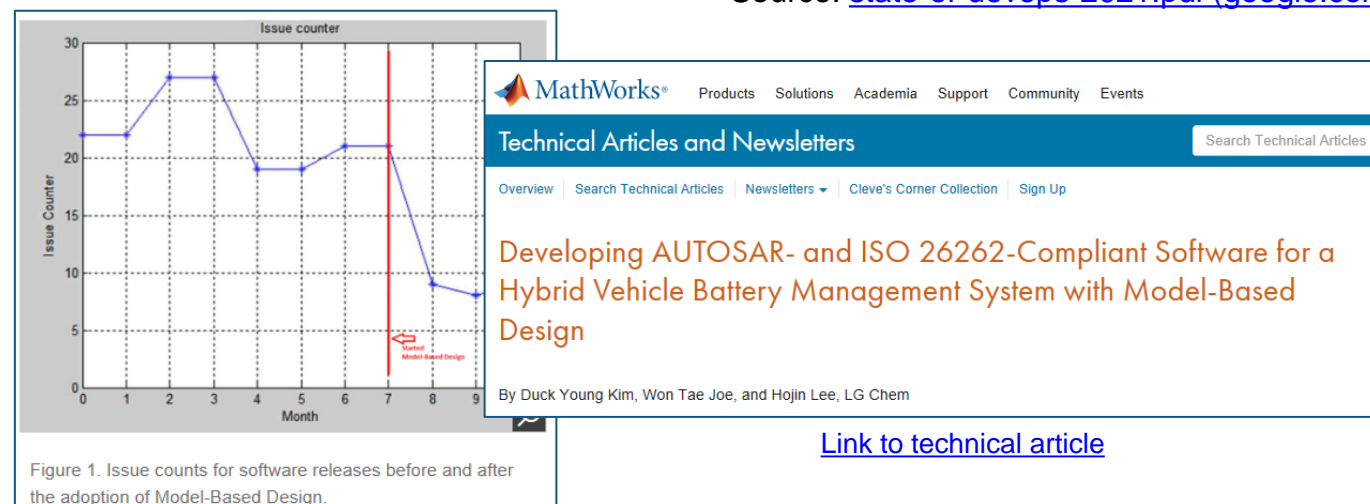
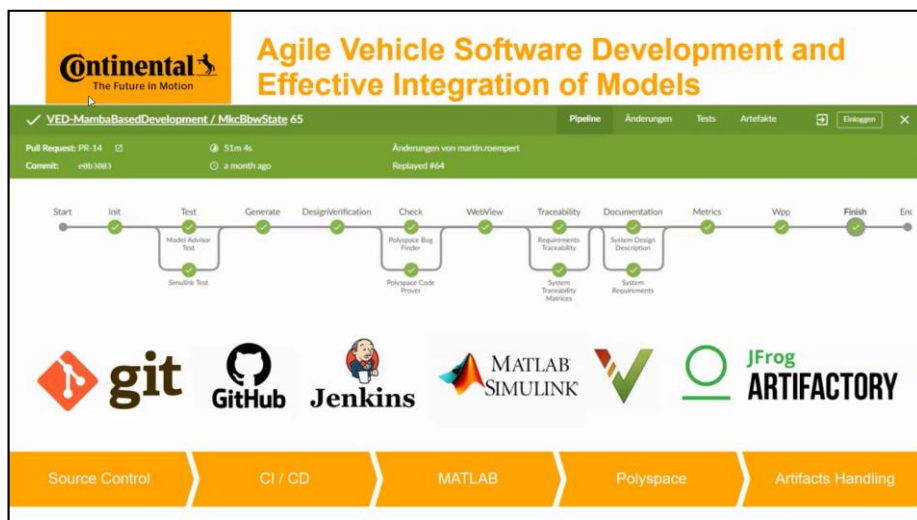
Model-Based Design enables high DevOps performance

- DevOps Goal: „**Reduce the time** between committing a change and placing it in production, **while ensuring high quality and compliance**„



	High Performers	Low Performers
Lead Time	<1hour	>6months
Change Failure Rate	0-5%	15-30%

Source: [state-of-devops-2021.pdf \(google.com\)](https://www.google.com/search?q=state-of-devops-2021.pdf)



MathWorks® Products Solutions Academia Support Community Events

Technical Articles and Newsletters

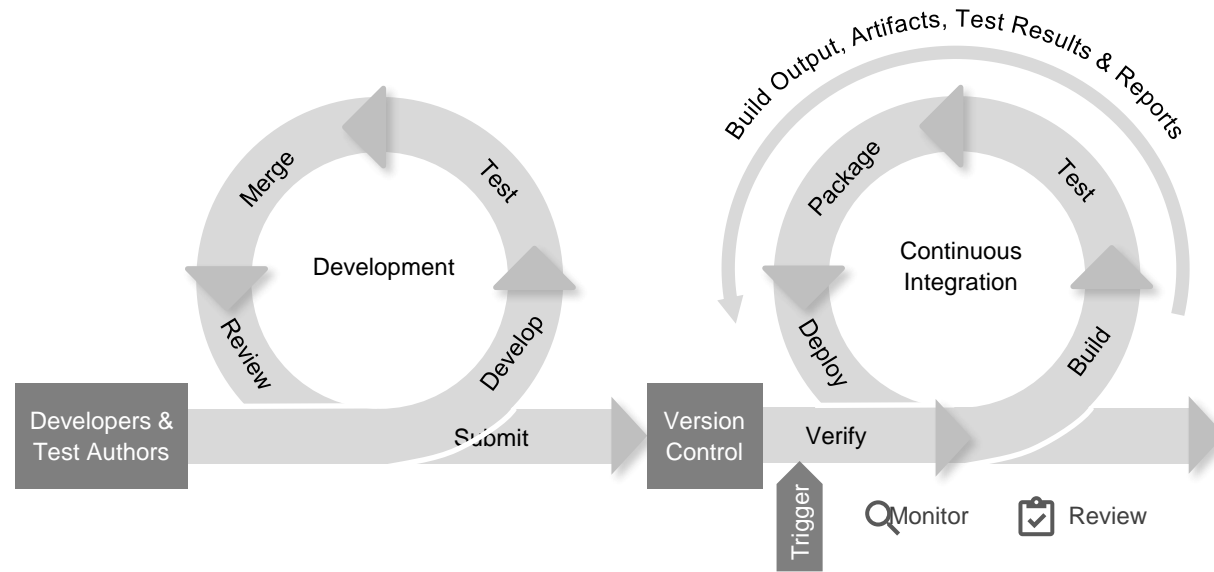
Overview | Search Technical Articles | Newsletters | Cleve's Corner Collection | Sign Up

Developing AUTOSAR- and ISO 26262-Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design

By Duck Young Kim, Won Tae Joe, and Hojin Lee, LG Chem

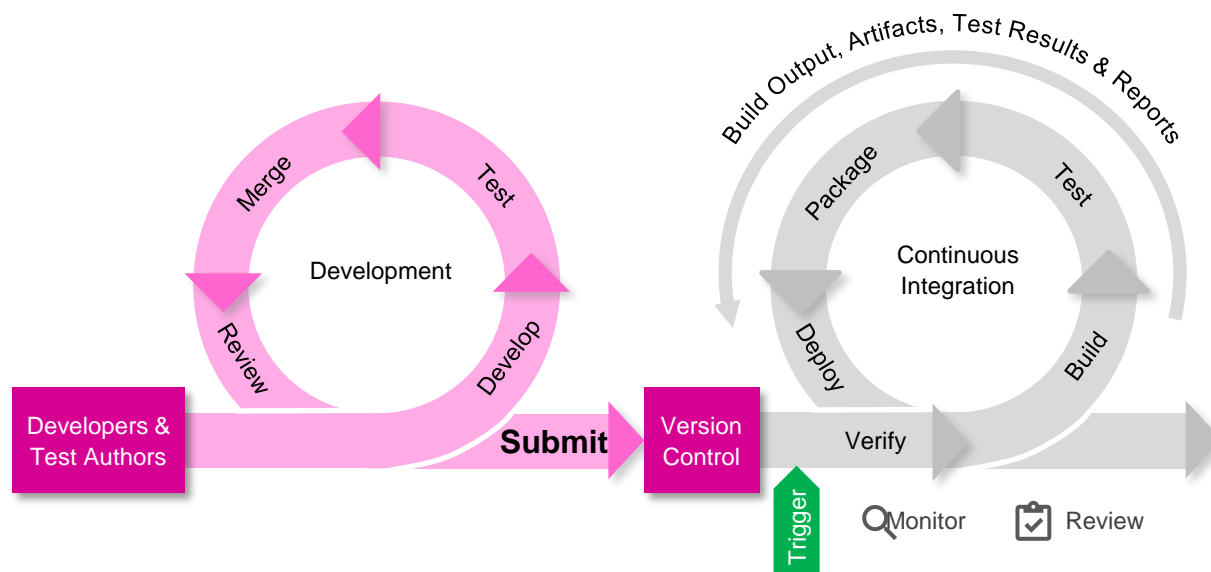
[Link to technical article](#)

CI workflow and tools are language- and domain-neutral



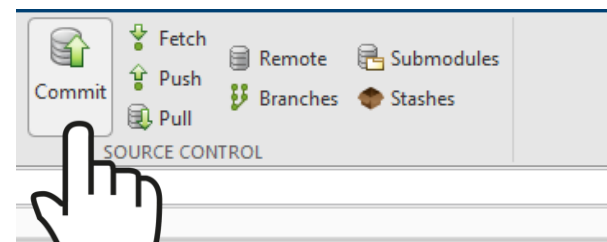
Each of these can “speak” MATLAB and Model-Based Design

Continuous Integration Workflow with Model-Based Design

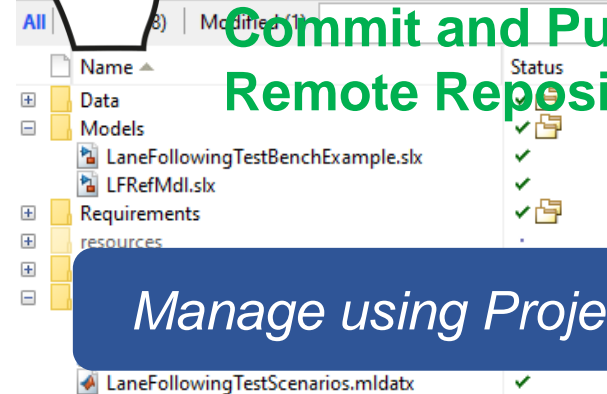


1 Development

2 Continuous Integration

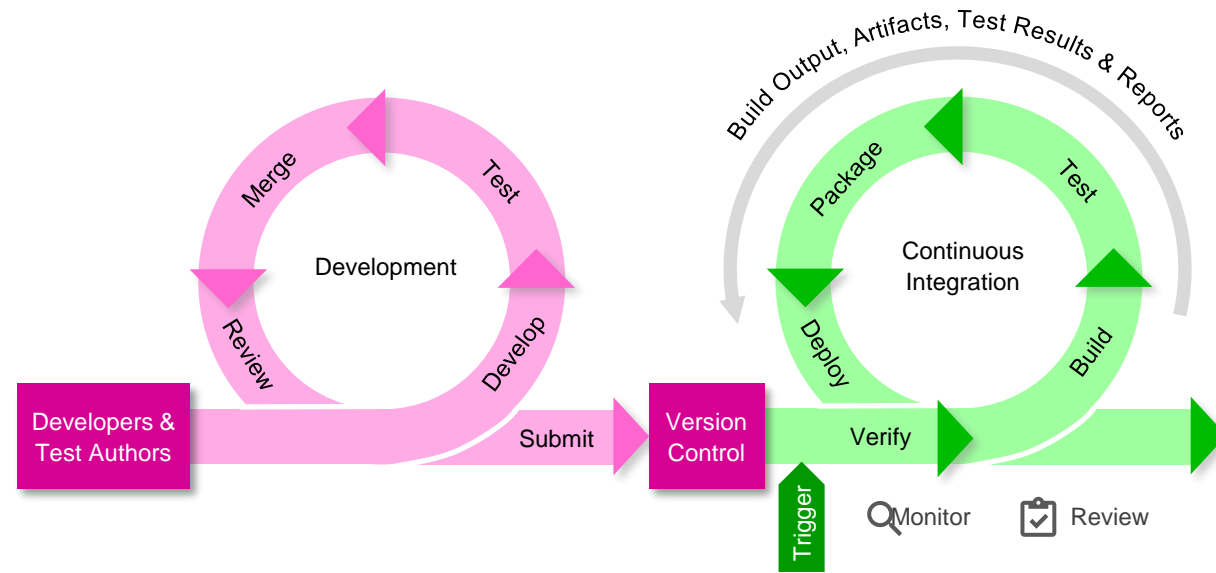


Commit and Push to Remote Repository

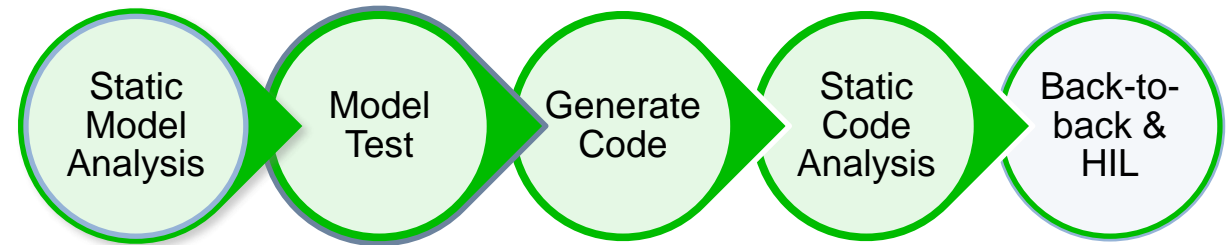


Manage using Projects

Continuous Integration Workflow with Model-Based Design



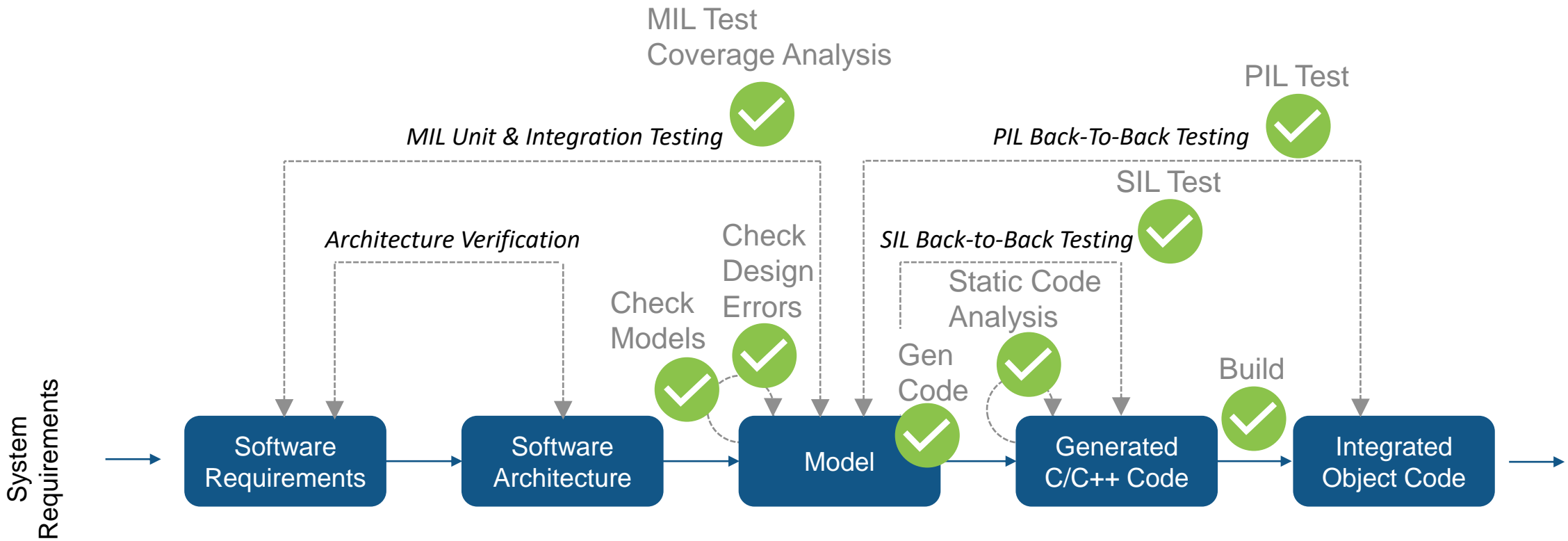
- 1** Development
- 2** Continuous Integration



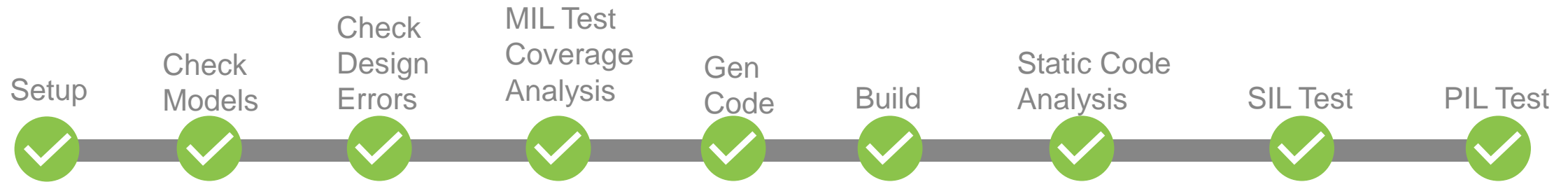
Reports, Build Logs, Test Results, Code Coverage

Verification, Build and Test

Model-Based Design Reference Workflow



Model-Based Design Reference Workflow



- Define Process and Automate

- Identify Tasks
- Define Sequence
- Define Outputs
- Script the Tools



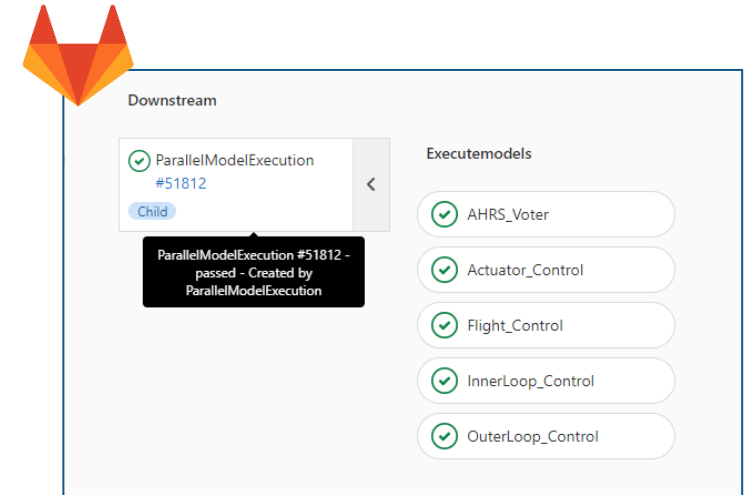
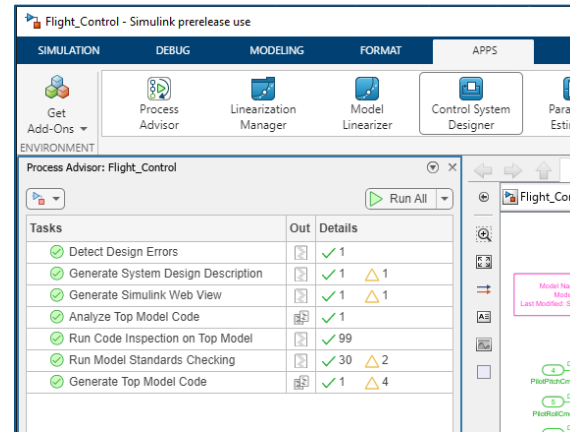
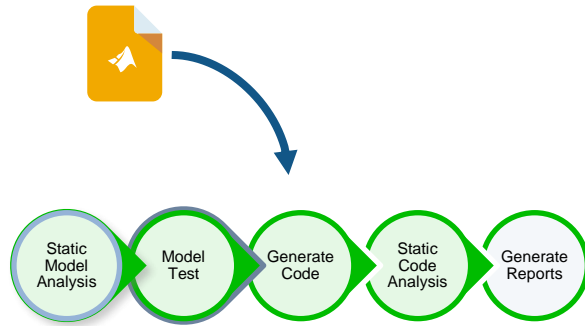
build.m



genCode.m



CI/CD Automation for Simulink Check Support Package



1) Simple Setup

- ✓ Prebuilt Model-Based Design pipeline
- ✓ Built-in Model-Based Design tool support
- ✓ Tailorable

2) Desktop Integration with Process Advisor app

- ✓ Local prequalification
- ✓ Local Debugging

3) 3rd Party CI Integration

- ✓ Jenkins/Gitlab YAML
- ✓ Optimized Model-Based Design Builds
- ✓ CI Results Integration

Prebuilt & Tailorable MBD Pipeline

Built-in Library of Tasks

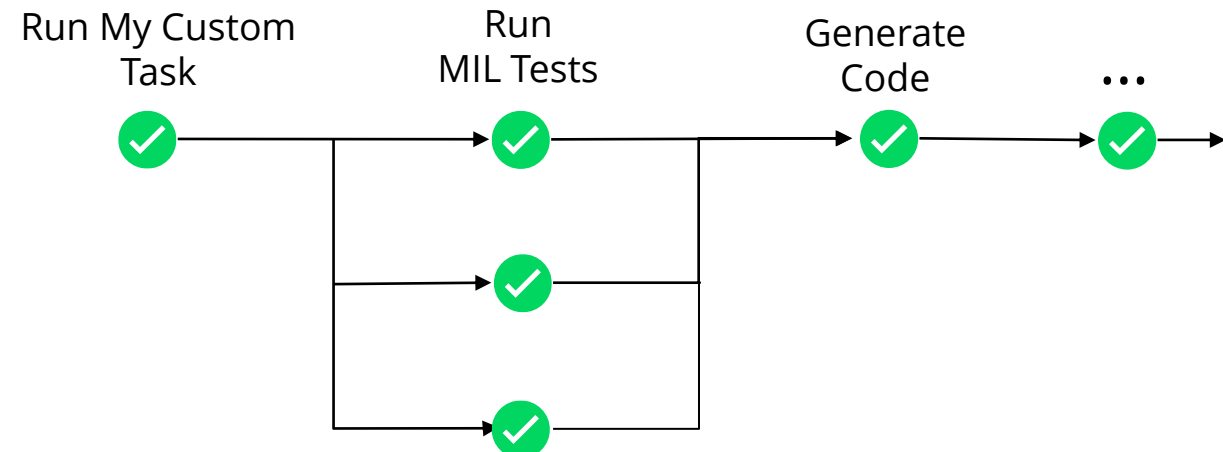
- Static Analysis
- Code Generation
- Testing

Zero Upfront Code

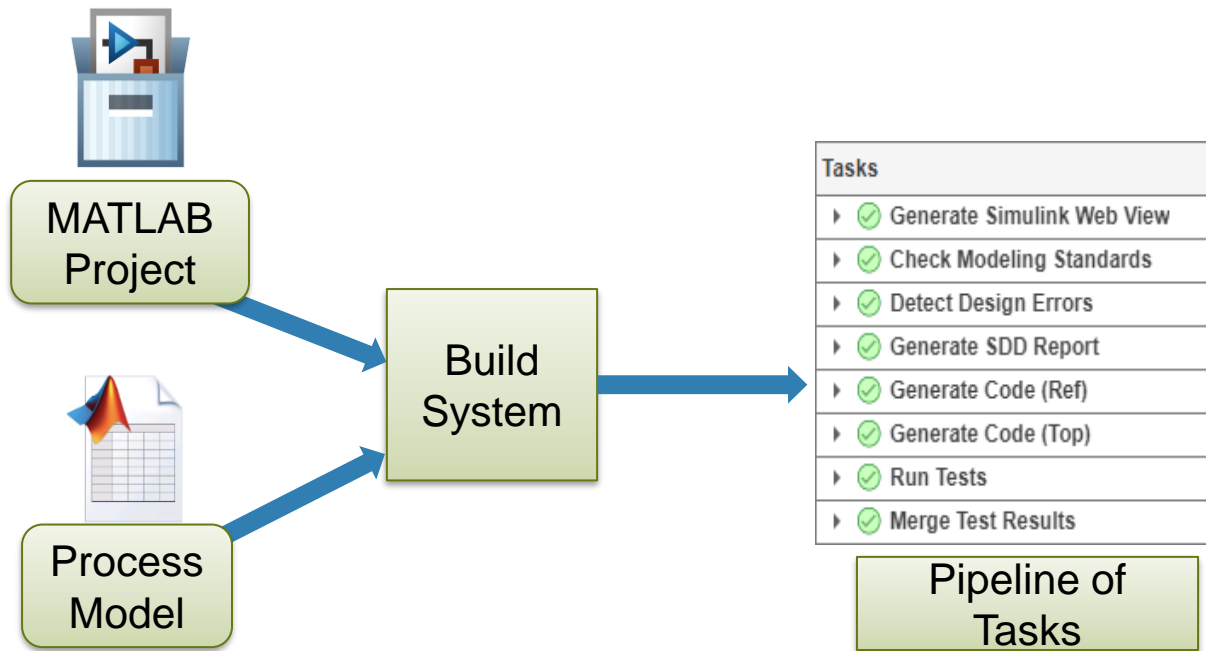
Fully Tailorable

- Modify existing steps
- Remove steps
- Add custom steps

TASKS	TOOLS
Check Model Standards Compliance	Simulink Check
Run Tests	Simulink Test
Generate Source Code	Embedded Coder
Check Code Standards Compliance	Polyspace Bug Finder
Generate Software Design Description	Simulink Report Generator
Design Error Detection	Simulink Design Verifier
Verify Model Update & Simulation	Simulink
Check Model Metrics	Model Advisor



MBD Pipeline Generation and Task Execution System

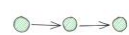








Capabilities

Execute in different workflows

- Interactive in Desktop (Process Advisor)
- Automated in CI
- Import results from CI into Desktop

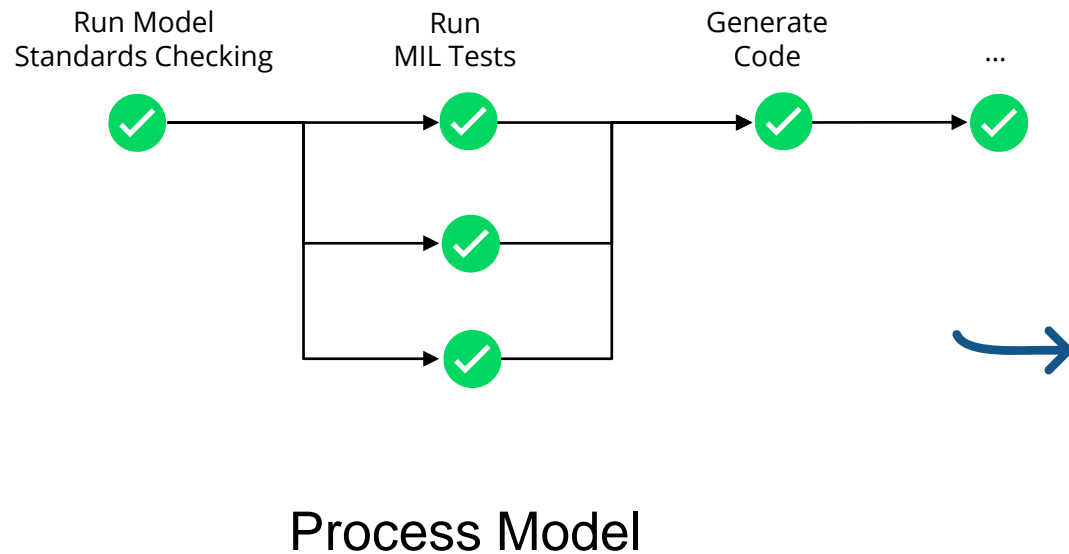
Generate CI Pipeline

- Multiple Architectures   
- Multiple Platforms
 - OS agnostic  
 - Current: GitLab , Jenkins 
 - Future: GitHub, Azure Pipelines

Smart Orchestration

- Incremental execution
- Repeatable results

Pre-qualification with Process Advisor



The screenshot shows the Process Advisor interface for a project named 'db_DriverSwRequest'. The top part of the window displays a block diagram of the system. Below the diagram, the 'Process Advisor - db_Controller' window is open, showing a table of tasks and their results.

MBD Build Tool

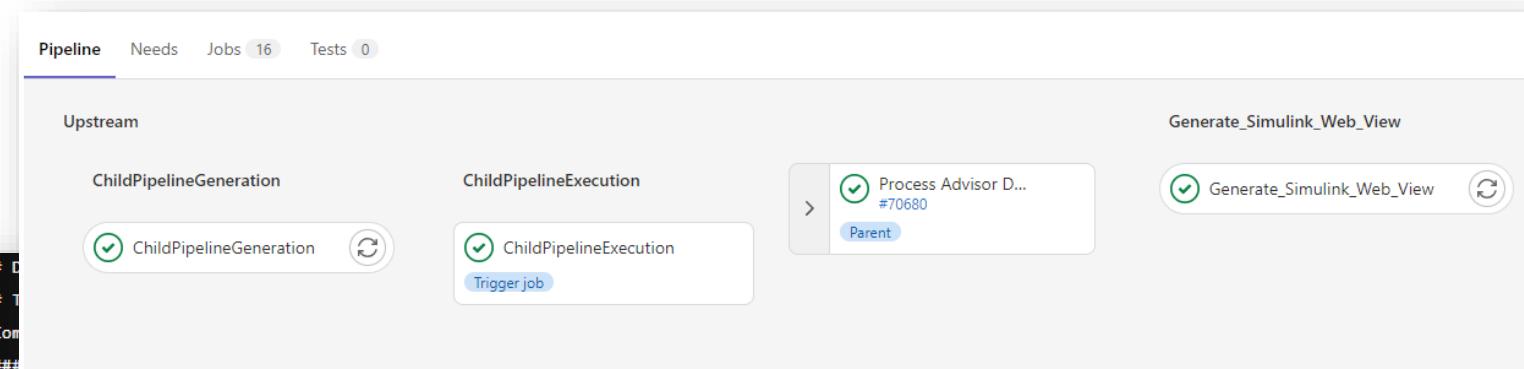
Tasks	Out	Results
✓ Run Code Generator	📄	✓ 20 ✗ 2 ⚠ 3
✓ db_ControlMode	📄	✓ 10 ✗ 1 ⚠ 1
✓ db_Controller	📄	✓ 2
✓ db_DriverSwRequest	📄	✓ 3 ⚠ 2
✓ db_TargetSpeedThrottle	📄	✓ 5 ✗ 1
✓ Run Model Standards Checking	📄	✓ 60 ✗ 1 ⚠ 5
✓ db_ControlMode	📄	✓ 15 ⚠ 4
✓ db_Controller	📄	✓ 15
✓ db_DriverSwRequest	📄	✓ 15 ✗ 1
✓ db_TargetSpeedThrottle	📄	✓ 15 ⚠ 1
▶ ✓ Run Design Error Detection	📄	✓ 93 ✗ 7 ⚠ 15

Process Advisor

 Local Desktop Workflow

Integration and Run with common CI Systems

- Automated Pipeline Generation
- Execute Pipeline in CI Systems like Jenkins
- Publish Results
- Debug on Desktop



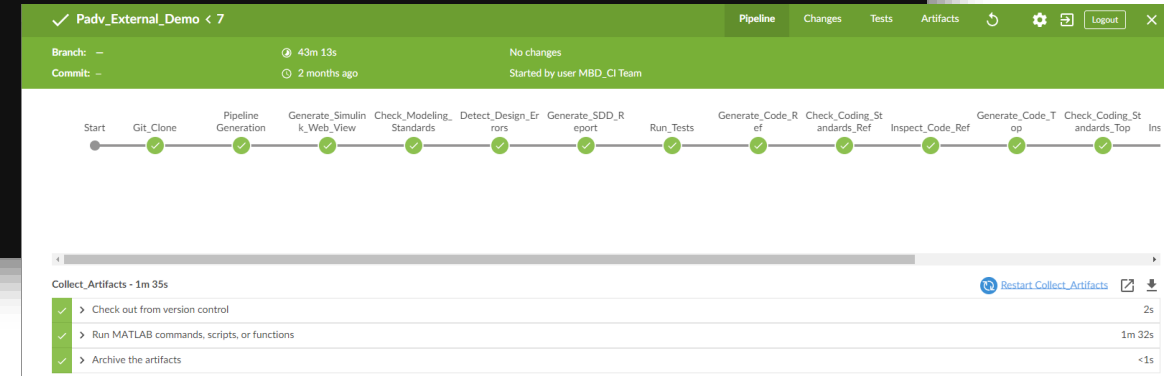
```

181 #### D
182 #### T
183 ## Co
184 #####
185 #####
186 ## Ending Process Advisor build at 29-Nov-2022 08:00:16
187 #### Duration: 00:05:45
188 #### Build Status: Pass
189 #### Number of tasks: 5
190 #### Number of tasks executed: 5
191 #### Number of tasks skipped: 0
192 #### Number of tasks in queue: 0
193 #### Number of tasks failed: 0
194 #####
195 Saving cache for successful job
    
```

Process Advisor - db_Controller

Run All

Tasks	Out	Results
<ul style="list-style-type: none"> Run Code Generator <ul style="list-style-type: none"> db_ControlMode db_Controller db_DriverSwRequest db_TargetSpeedThrottle Run Model Standards Checking <ul style="list-style-type: none"> db_ControlMode db_Controller db_DriverSwRequest db_TargetSpeedThrottle Run Design Error Detection 	<ul style="list-style-type: none"> db_ControlMode db_Controller db_DriverSwRequest db_TargetSpeedThrottle db_ControlMode db_Controller db_DriverSwRequest db_TargetSpeedThrottle 	<ul style="list-style-type: none"> 20 ✓ 2 ✗ 3 ⚠ 10 ✓ 1 ✗ 1 ⚠ 2 ✓ 3 ✓ 0 ✗ 0 ⚠ 5 ✓ 1 ✗ 0 ⚠ 60 ✓ 1 ✗ 5 ⚠ 15 ✓ 0 ✗ 4 ⚠ 15 ✓ 15 ✓ 1 ✗ 0 ⚠ 15 ✓ 0 ✗ 1 ⚠ 93 ✓ 7 ✗ 15 ⚠



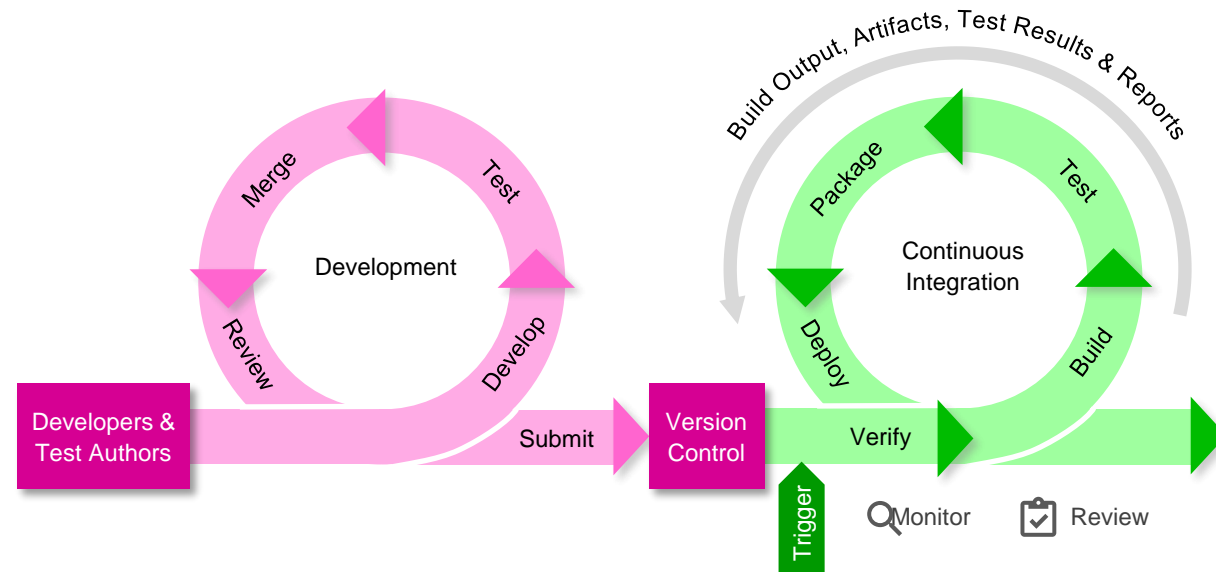
Continuous Integration Workflow with Model-Based Design – Invoke pipeline

The screenshot displays the MATLAB R2022a environment. The top menu bar includes HOME, PLOTS, APPS, PROJECT, and PROJECT SHORTCUTS. The ribbon contains various toolbars such as FILE, TOOLS, ENVIRONMENT, and SOURCE CONTROL. The main workspace is divided into several panes:

- Current Folder:** Shows the path `C:\sandbox\ProcessAdvisor\work`.
- Project - ProcessAdvisorExample:** A tree view of the project structure. The `02_Models` folder is selected. The table below shows the project's file structure and status.
- Workspace:** An empty table with columns for Name and Value.
- Command Window:** Shows the command `runprocess` being executed.

Name	Status	Git	Classification
Files			
01_Requirements	✓	.	
02_Models	✓	.	
03_Code	✓	.	
04_Results	✓	.	
tools	✓	.	
work	✓	.	
.gitattributes	✓	.	
.gitignore	✓	.	
.gitlab-ci.yml	✓	.	
derived	✓	.	
processmodel.m	✓	.	Design

Continuous Integration Workflow with Model-Based Design



1 Development

2 Continuous Integration



Build

Run MATLAB Tests

MATLAB root: D:\bsinst\matlab

Test mode: Automatic

Generate Test Artifacts

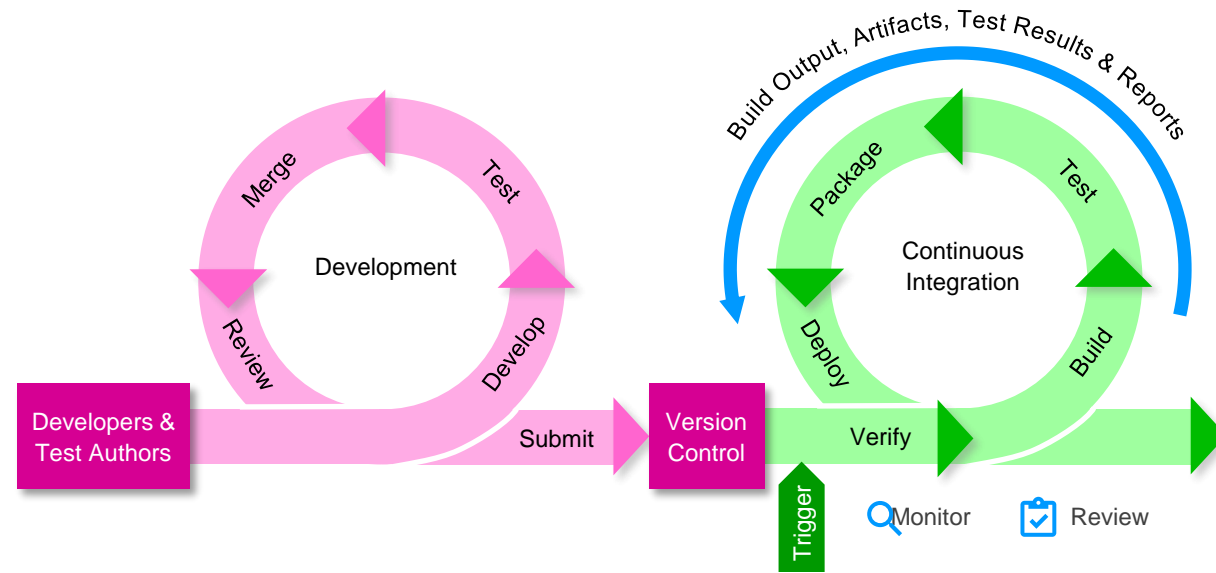
- PDF test report
- TAP test results
- JUnit-style test results
- Simulink Test Manager results

Generate Coverage Artifacts

MATLAB Jenkins Plugin

Add build step

Continuous Integration Workflow with Model-Based Design



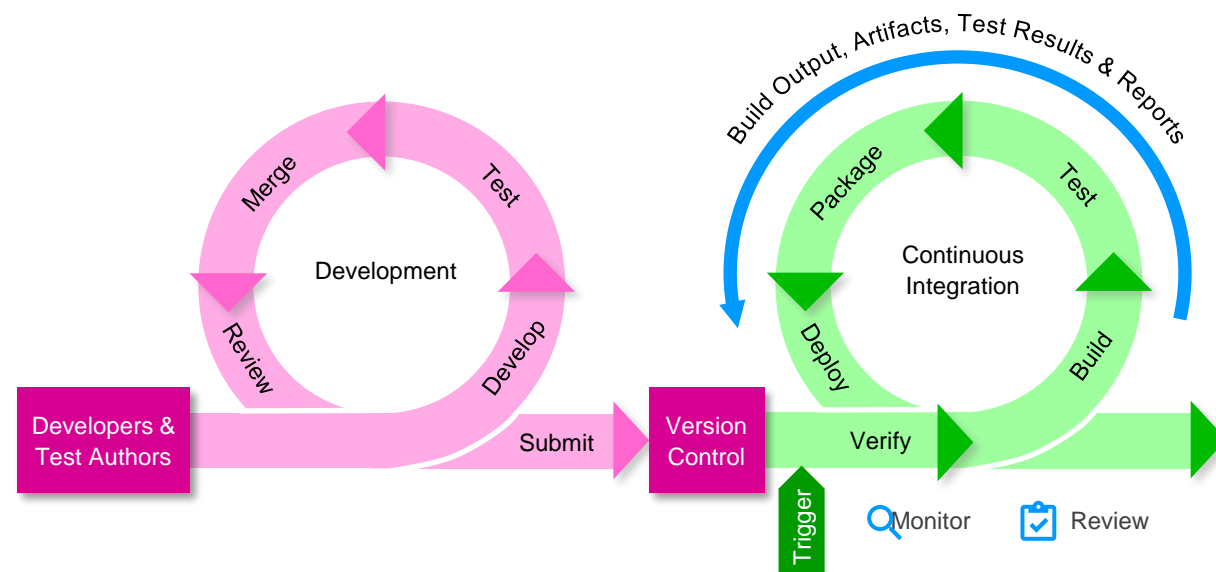
- 1** Development
- 2** Continuous Integration
- 3** Results Monitor and Review

```

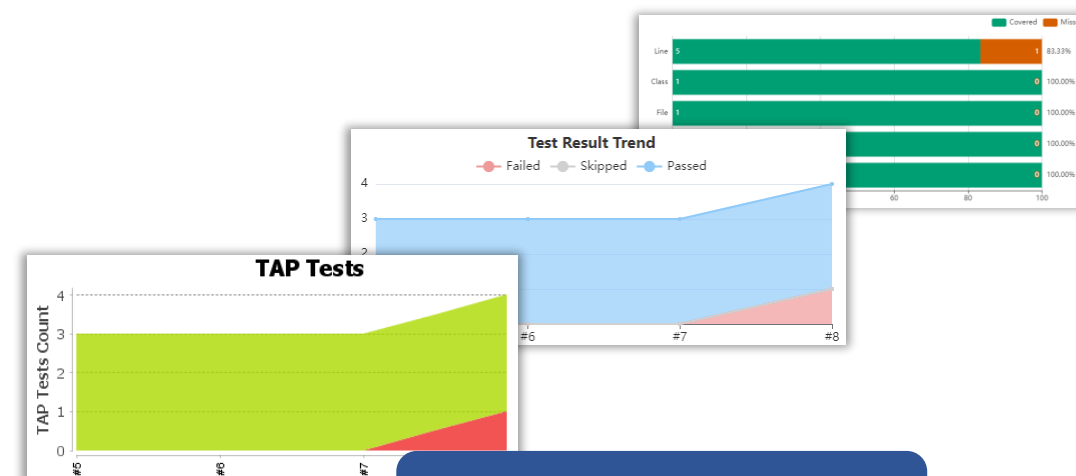
Failure Summary:
-----
Name                                                                    Failed
-----
LaneFollowingTestScenarios > Scenarios/LFACC_Curve_CutInOut_TooClose    X
ERROR: MATLAB error Exit Status: 0x00000001
Build step 'Run MATLAB Tests' changed build result to FAILURE
Finished: FAILURE
    
```

Logs

Continuous Integration Workflow with Model-Based Design



- 1 Development
- 2 Continuous Integration
- 3 Results Monitor and Review



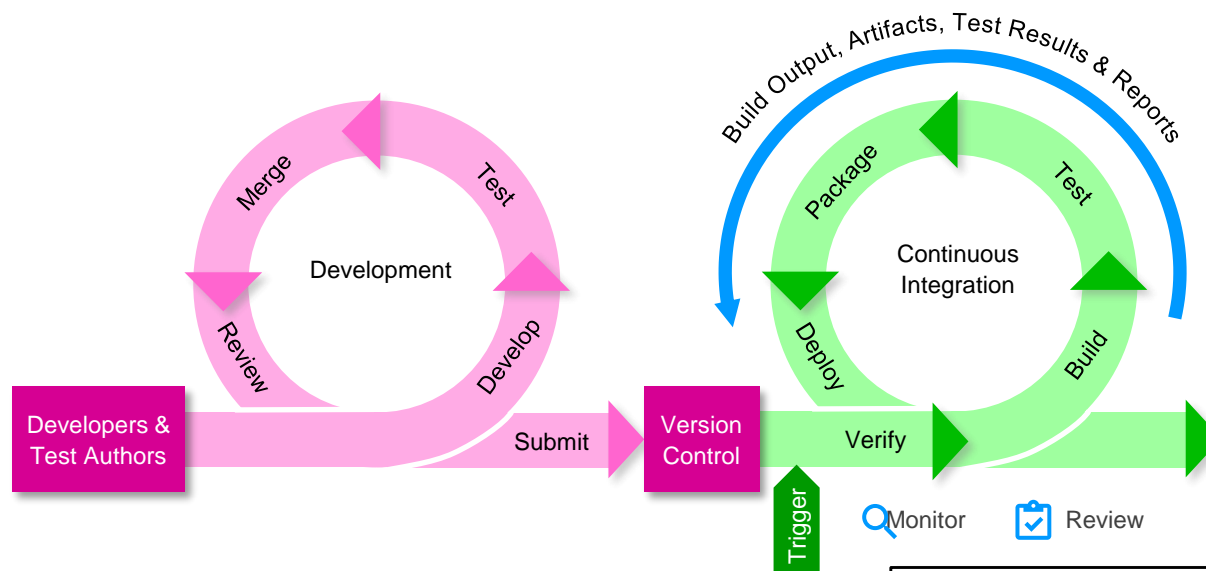
junit, TAP, Cobertura

Continuous Integration Workflow with Model-Based Design

ISO 26262
Functional Safety

ISO/SAE 21434
Cybersecurity for Road Vehicles

ISO 21448
Safety of the intended functionality (SOTIF)



- 1 Development
- 2 Continuous Integration
- 3 Results Monitor and Review

Results: 2017-Jan-19 13:34:39

Result Type: Result Set
Parent: None
Start Time: 2017-Jan-19 13:34
End Time: 2017-Jan-19 13:35
Outcome: Total: 2, Passed: 2

Aggregated Coverage Results

Analyzed Model	Sim Mode	Comp.	Decision	Condition
AHRS_voter1	ModelRefSIL32	24%	100%	

[Back to Report Summary](#)

AHRS_voter_SLDV_TestCases

Test Result Information

Result Type: Test File Result
Parent: Results: 2017-Jan-19 13:34
Start Time: 2017-Jan-19 13:34
End Time: 2017-Jan-19 13:35
Outcome: Total: 2, Passed: 2

Test Suite Information

Name: AHRS_voter_SLDV

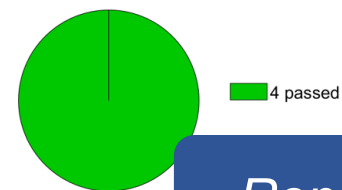
[Back to Report Summary](#)

MATLAB® Test Report

Timestamp: 04-Feb-2021 20:27:27
Host: SEBDEERAZER
Platform: win64
MATLAB Version: 9.9.0.1570001 (R2020b) Up

Number of Tests: 4
Testing Time: 170.7677 seconds

Overall Result: PASSED

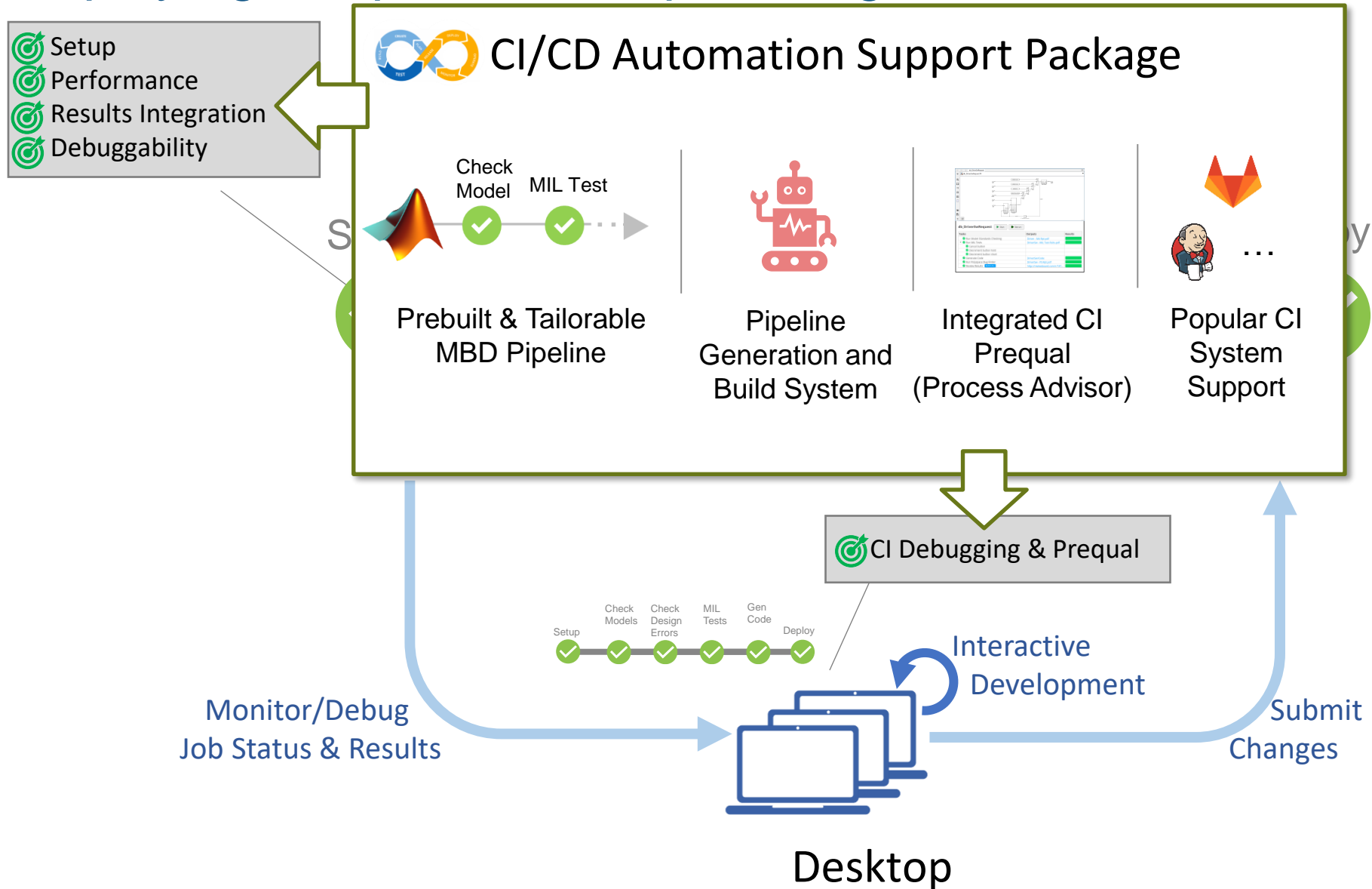


Reports

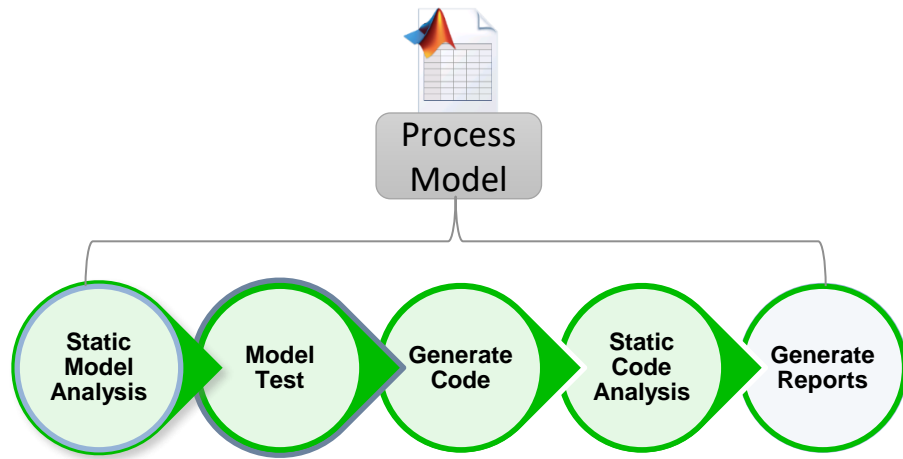
Trapped Mass Calculation

Cylinder 1 Trapped Mass Correction	
- Main	
RowIndex	ACDCharge
ColumnIn...	ACDEngSpe
Table	ACDTrapped
LookUpMeth	Interpolation-Use End Values
- Signal Attributes	
OutMin	[]
OutMax	[]
InputSam...	off
OutDataT...	Inherit: Same as first input
LockScale	off
RndMeth	Floor
SaturateO...	off
Other	

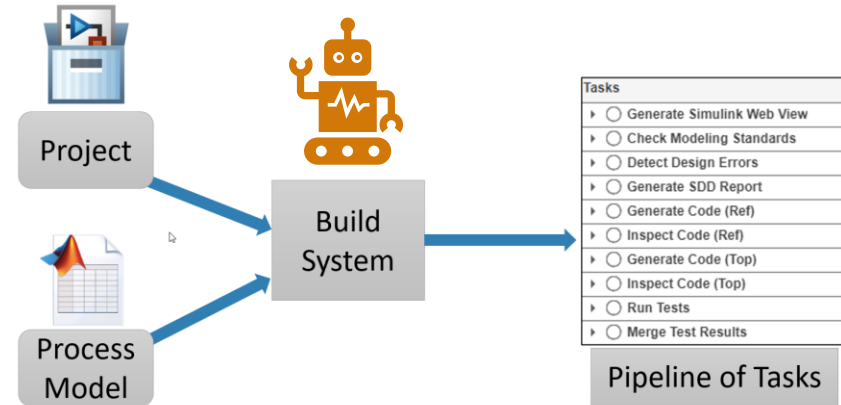
Simplifying Adoption and Optimizing CI/CD for Model-Based Design



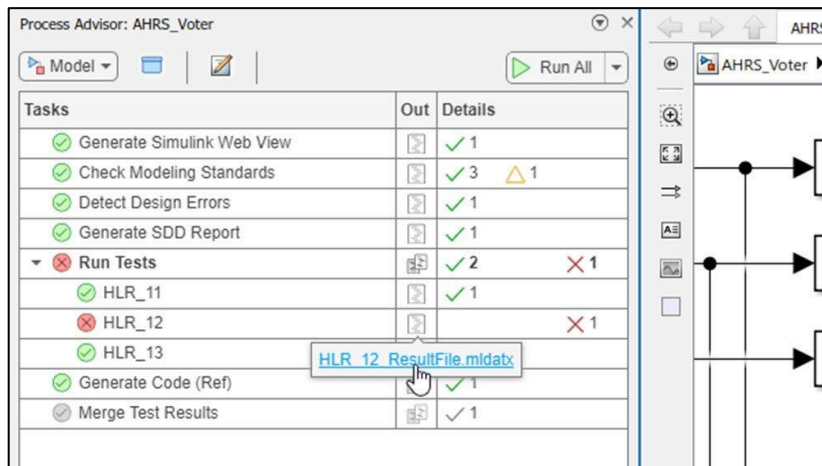
CI/CD Automation for Simulink Check Support Package



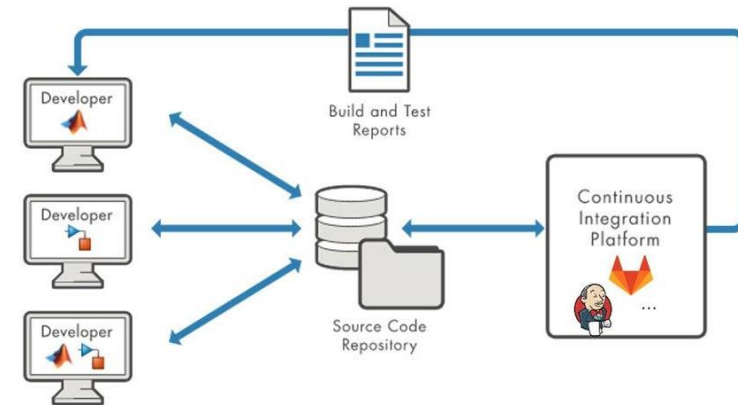
Prebuilt & Tailorable Model-Based Design Pipeline



Build system to generate and optimally execute the process in your CI system



Prequalification with Process Advisor

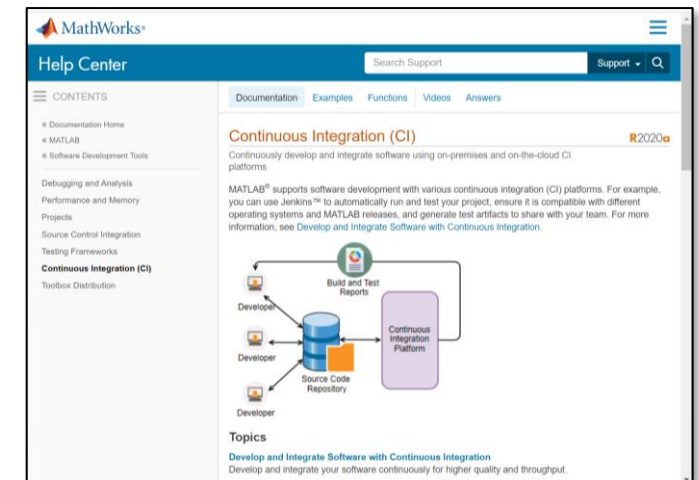
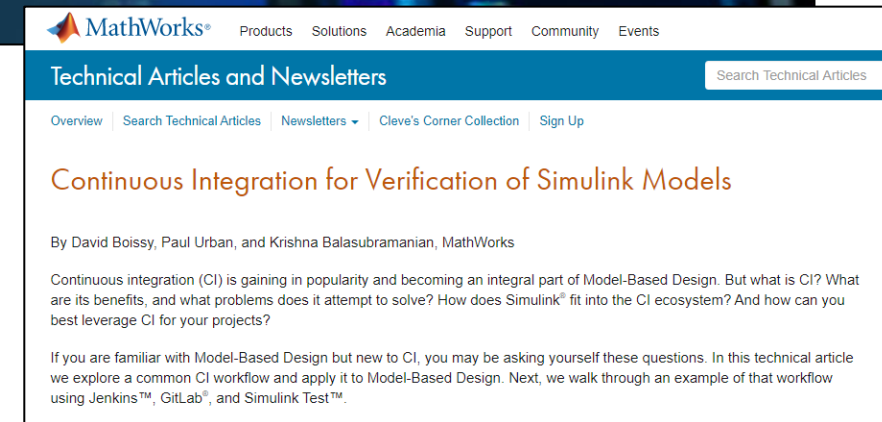


Examples to run process on common CI Systems

Learn more: [Continuous Integration for Model-Based Design](#)

Other CI resources:

- [Continuous Integration Solution Page](#)
- Videos:
 - [Continuous Integration with MATLAB and Simulink](#)
 - [Automotive DevOps for Model-Based Design with AWS](#)
- Technical Articles:
 - [Continuous Integration for Verification of Simulink Models](#)
 - [Continuous Integration for Verification of Simulink Models Using GitLab](#)
 - [Agile Model-Based Design: Accelerating Simulink Simulations in CI Workflows](#)
- Documentation:
 - [CI/CD Automation for Model-Based Design Support Package](#)
 - [Continuous Integration Documentation Hub](#)
 - [Tests for Continuous Integration](#)
- [Developer Zone: Continuous Integration](#)



Getting Started: CI plugins and code examples

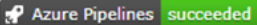
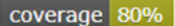


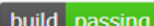
- Code examples
 - [CI configuration examples](#)
 - [CI with Simulink](#)
 - [Code coverage using Codecov](#)

- CI plugins
 - [Azure DevOps](#)
 - [CircleCI](#)
 - [GitHub Actions](#)
 - [Jenkins](#)
 - [Travis CI](#)

- Reference architectures (AWS, GCP, ...)
 - <https://github.com/mathworks-ref-arch>

MATLAB CI Configuration Examples

This repository shows how to run MATLAB tests with a variety of continuous integration systems.

CI Platform	Badges	Badge Help
Azure DevOps	 	Blog with helpful information for setting up Azure DevOps badges
CircleCI		CircleCI documentation for setting up badges
GitHub Actions		GitHub Actions documentation for setting up badges
Travis CI		

Orbs > mathworks/matlab@0.4.0

mathworks/matlab@0.4.0 PARTNER

Run MATLAB and Simulink as part of your build pipeline.

Created: October 25, 2019 | Version Published: February 4, 2021 | Releases: 12

Homepage: <https://www.mathworks.com/solutions/continuous-integration.html>

Source: <https://github.com/mathworks/matlab-circleci-orb>

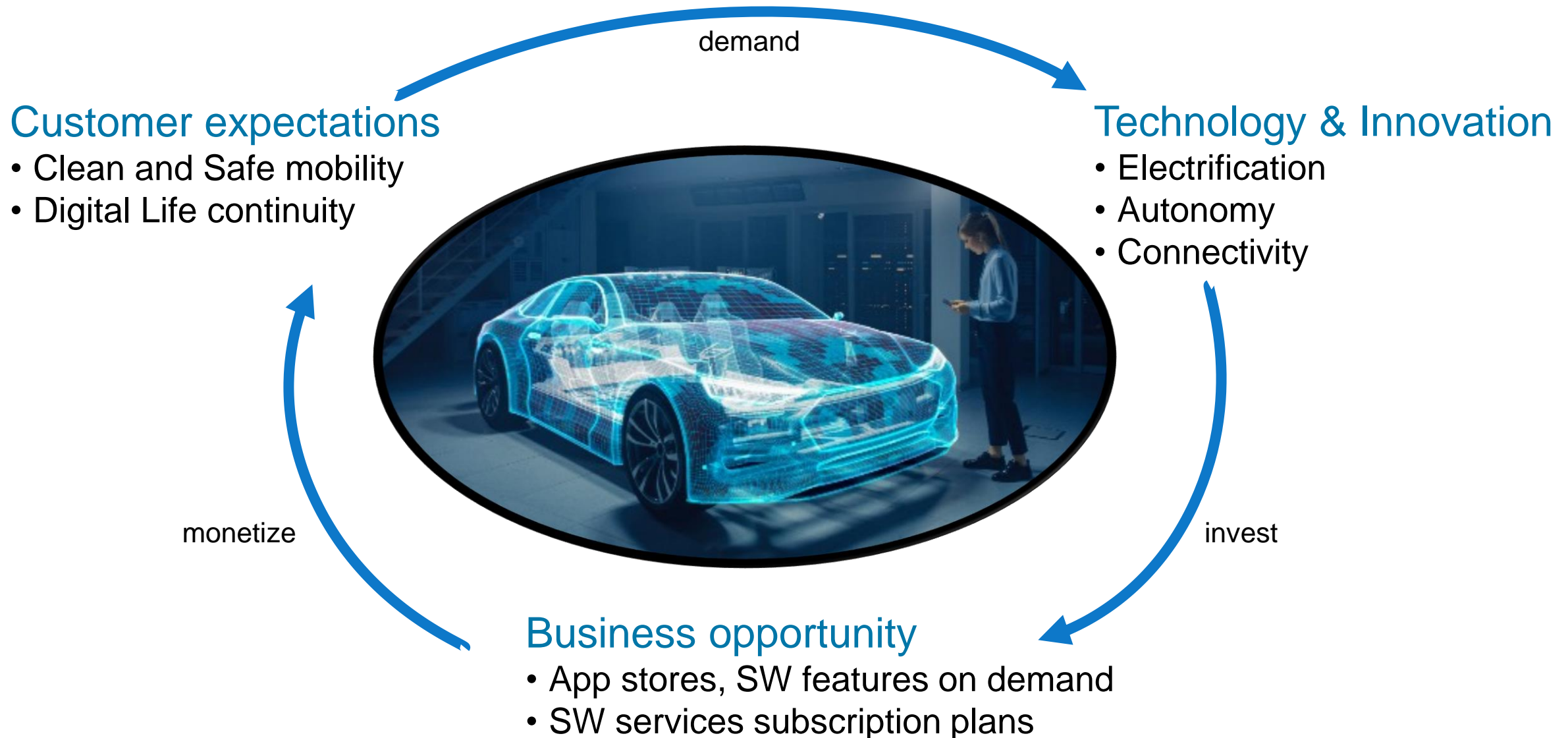
[See Orb Licensing](#)

Relevant Training Classes



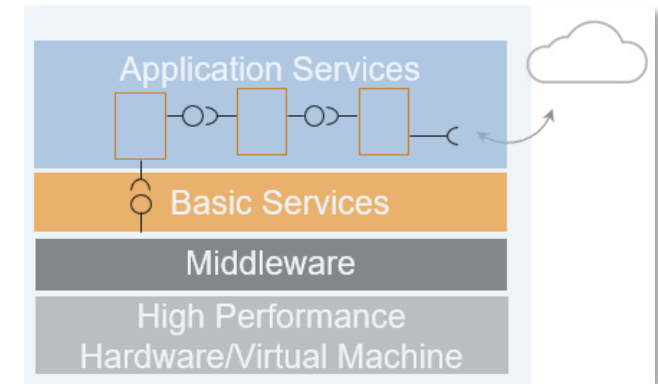
- [Simulink Fundamentals](#) – introduction to designing models using Simulink
- [Simulink Model Management and Architecture](#) – Requirements Toolbox, Simulink Projects, Architectural Choices, Data Management, Simulink Report Generator
- [Simulation-Based Testing with Simulink](#) – includes Simulink Test
- [Design Verification with Simulink](#) – Simulink Design Verifier
- [Embedded Coder for Production Code Generation](#) – generating and using code from Simulink models
- [Polyspace for C/C++ Code Verification](#) – static analysis of hand code and automatically-generated code
- **Applying Model-Based Design for ISO 26262** (available upon request)

Software-Defined Vehicle



SOA – What's it all about?

- With SOA, applications are standalone processes that provide and/or require services distributed across the vehicle computing platform and the cloud
- SOA provides flexibility to add, remove, or update applications without impacting the entire, typically large, software system
- SOA is used by multiple industrial standards:
 - AUTOSAR Adaptive Platform
 - DDS (Data Distribution Services)
 - ROS (Robot Operating System)



AUTOSAR Blockset

Design and simulate AUTOSAR software

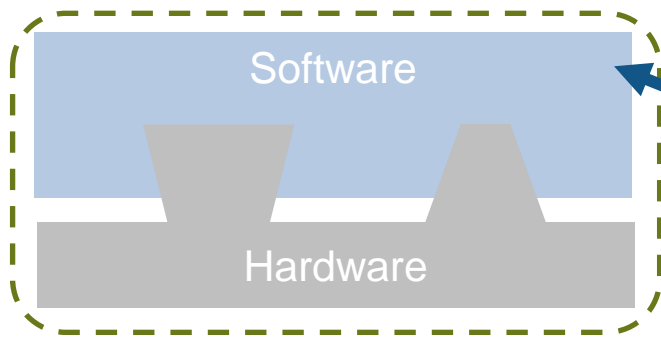
DDS Blockset

Design and simulate DDS applications

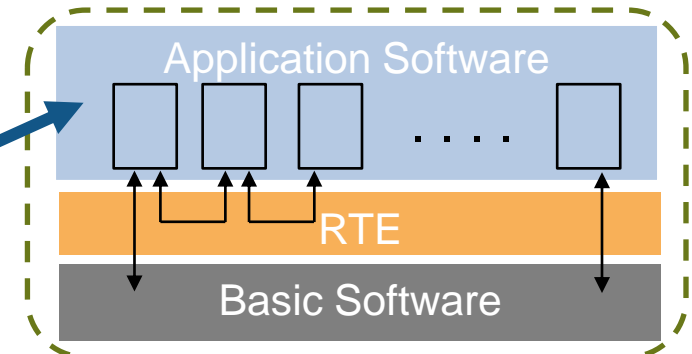
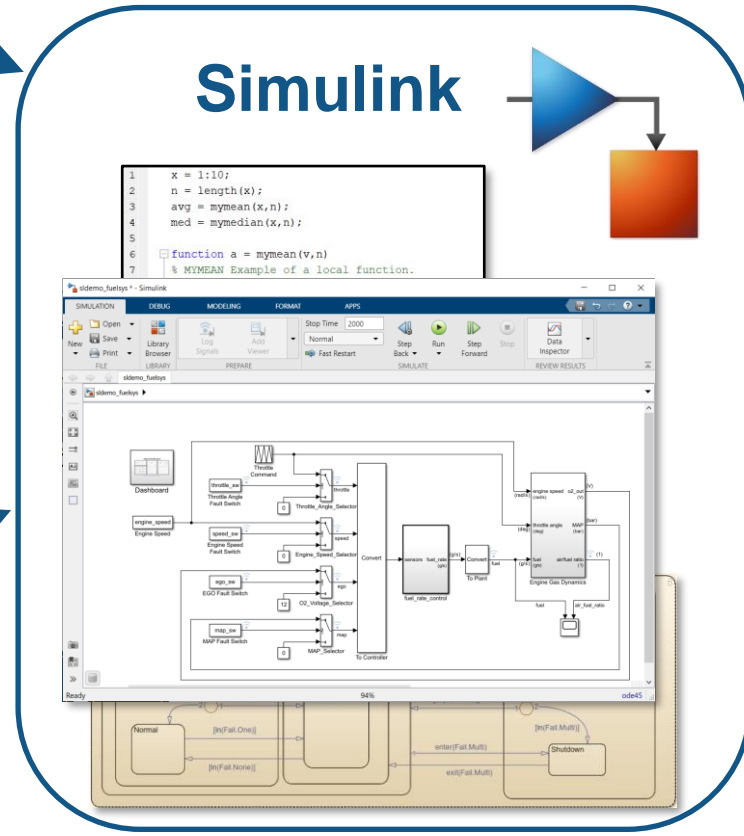
ROS Toolbox

Design, simulate, and deploy ROS-based applications

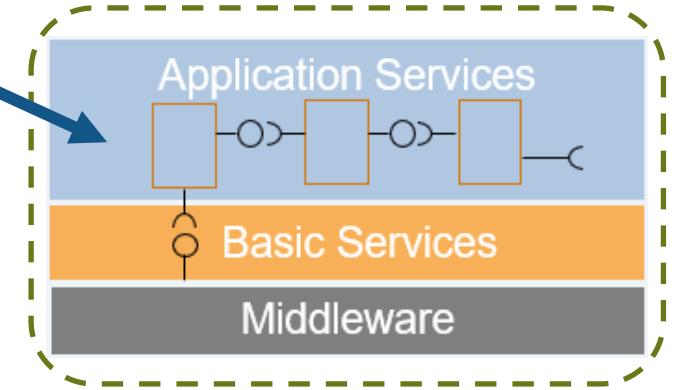
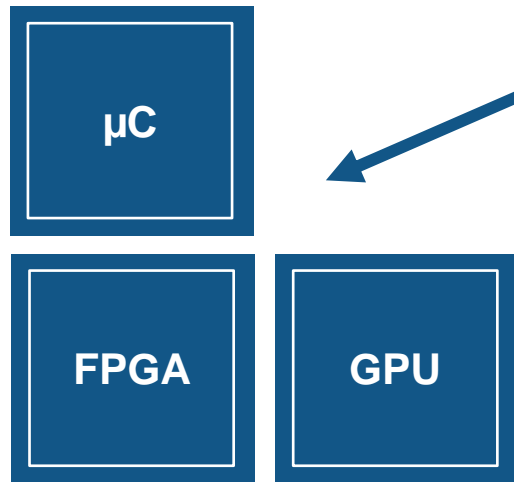
Simulink: Deploy software to different targets and standards



Legacy ECU



AUTOSAR Classic



AUTOSAR Adaptive / ROS / DDS

MathWorks Consulting

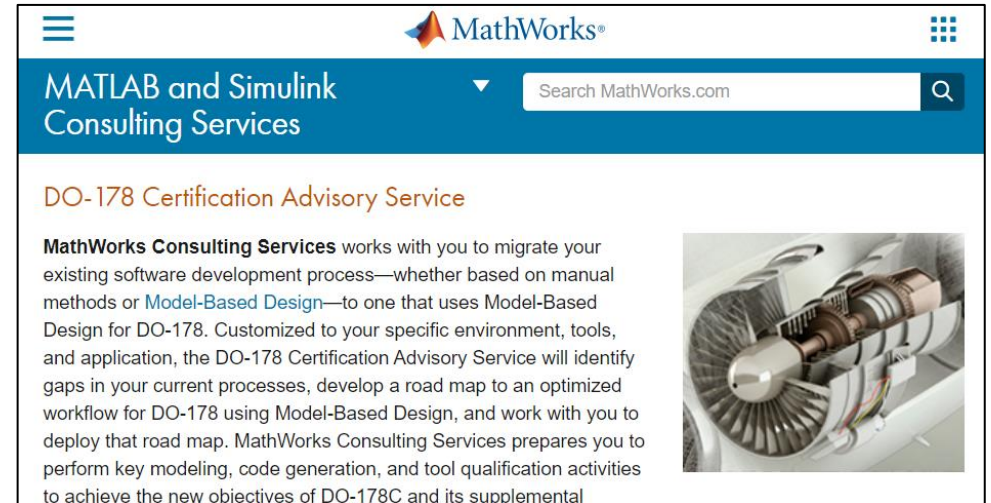
Certification Advisory Service Overview

- MathWorks Consulting will help you to:
 - Leverage Model-Based Design and supporting tools to their fullest extent to maximize ROI
 - Reduce duplicated and manual effort
 - Avoid common pitfalls by providing proven best practices for Model Based Design, system engineering and development of safety critical software with MBD



The screenshot shows the MathWorks website interface. At the top, there is a navigation bar with the MathWorks logo and a search bar. Below the navigation bar, the page title is "MATLAB and Simulink Consulting Services". The main content area features a section titled "ISO 26262 Process Deployment Advisory Service". The text describes how MathWorks Consulting Services helps migrate existing processes to a Model-Based Design framework for ISO 26262. To the right of the text is an image of a yellow car chassis.

[ISO 26262 Process Deployment Advisory Service](#)



The screenshot shows the MathWorks website interface. At the top, there is a navigation bar with the MathWorks logo and a search bar. Below the navigation bar, the page title is "MATLAB and Simulink Consulting Services". The main content area features a section titled "DO-178 Certification Advisory Service". The text describes how MathWorks Consulting Services helps migrate existing software development processes to a Model-Based Design framework for DO-178. To the right of the text is an image of a jet engine turbine.

[DO-178 Certification Advisory Service](#)



Thank you



For further details, Q&A and feedback kindly reach out to

Gaurav Ahuja

gahuja@mathworks.com

Application Engineering Group

[LinkedIn](#)



Rajat Arora

rarora@mathworks.com

Application Engineering Group

[LinkedIn](#)





MathWorks ✓

@MathWorks

Share the EXPO experience
#MATLABEXPO



[profile handle]



[profile handle]



[profile handle]



[profile handle]



[profile handle]