

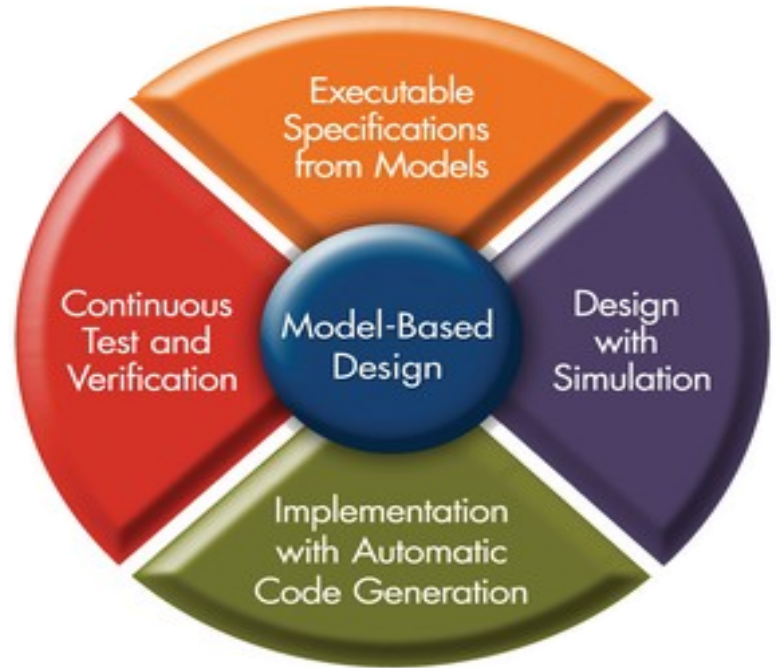
Certifiable Production Code Development

David Owens
Rolls-Royce Control Systems

© 2017 Rolls-Royce plc and/or its subsidiaries

The information in this document is the property of Rolls-Royce plc and/or its subsidiaries and may not be copied or communicated to a third party, or used for any purpose other than that for which it is supplied without the express written consent of Rolls-Royce plc and/or its subsidiaries.

This information is given in good faith based upon the latest information available to Rolls-Royce plc and/or its subsidiaries, no warranty or representation is given concerning such information, which must not be taken as establishing any contractual or other commitment binding upon Rolls-Royce plc and/or its subsidiaries.



Trusted to deliver excellence



Rolls-Royce

Overview

The Trent 7000 EMU is the first time that Rolls-Royce has gone through the DO-178C (ED-12C) certification process for certifying software in an airborne system using the Model Based Supplement DO-331 (ED-218)

This presentation discusses the Model Based Design method used during the application software development and the MathWorks tools which enabled the workflow.



Contents

- **Background**
 - What is the EMU
 - What steps are required for DO-178C certification
- **Development Process**
 - Model development from requirements
 - Model verification activities
 - Code production and verification
 - Tool Qualification
- **Future Enhancements**
 - Further tool adoption



Background



Rolls-Royce

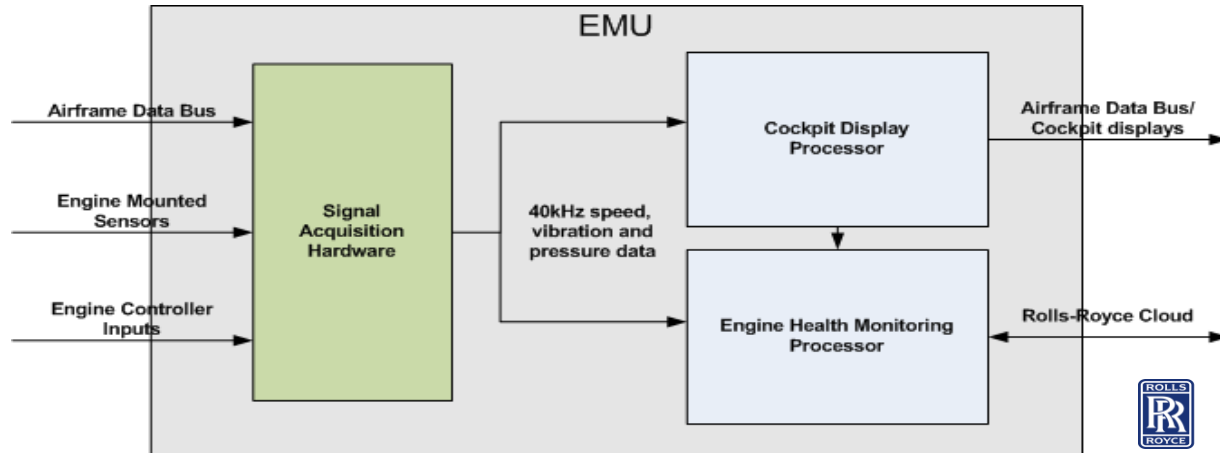
What is the EMU

The **EMU** is the **Engine Monitoring Unit**

Also known as the **EVHMU Engine Vibration and Health Monitoring Unit**.

The unit has two primary functions:-

1. A high integrity partition which interfaces with the cockpit displays.
2. A engine health monitoring partition which sends data in the form of reports back to Rolls-Royce to feed into the TotalCare[®] aftermarket service.



What is the EMU

From 2006 to 2016:

Hardware: Supplied by sub-tier supplier

Cockpit Displays Software: Specified by Rolls-Royce, supplied by sub-tier supplier

EHM Software : Specified by Rolls-Royce, supplied by sub-tier supplier

Engines fitted with EMUs: Trent900, 1000, XWB

From 2016 onwards:

Hardware: Rolls-Royce Control Systems

Cockpit Displays Software: Rolls-Royce Control Systems

EHM Software: Rolls-Royce Control Systems

Engines (to be) fitted with EMUs: All future Civil Large, and Medium Corporate engines

A software development method which could deliver to the tight engine timescales was required:

The solution:- **Model Based Design**



Rolls-Royce 6

What is the EMU

Aircraft level certification for large aeroplanes requires that:-

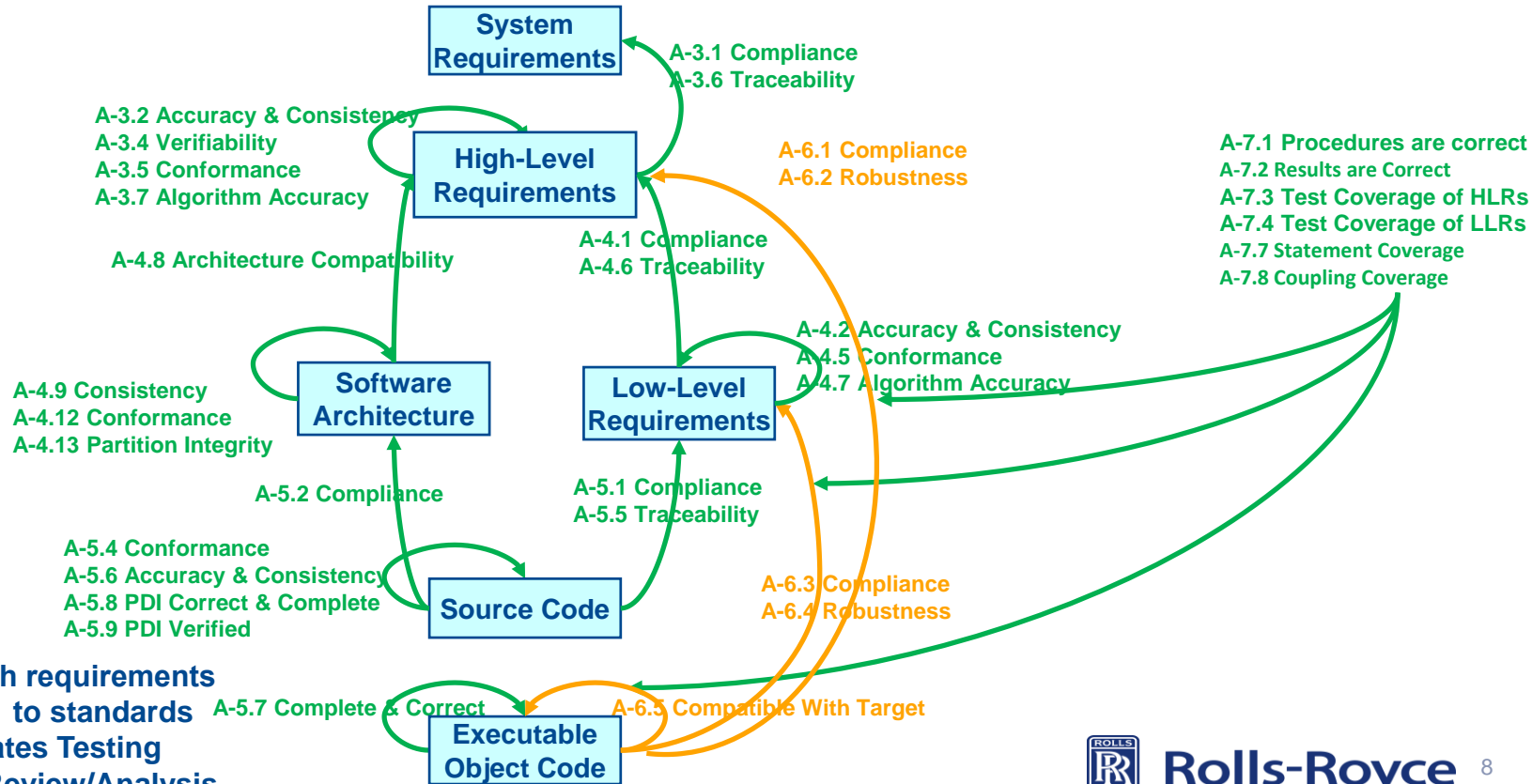
“An indicator to indicate rotor system unbalance” is displayed to the pilot for turbo-jet engine-powered aeroplanes. [CS-25 directive 25.1305 d3]

As the pilot makes decisions based on information provided by the EMU the HW and SW need to be developed to the appropriate **Design Assurance Level (DAL)**.

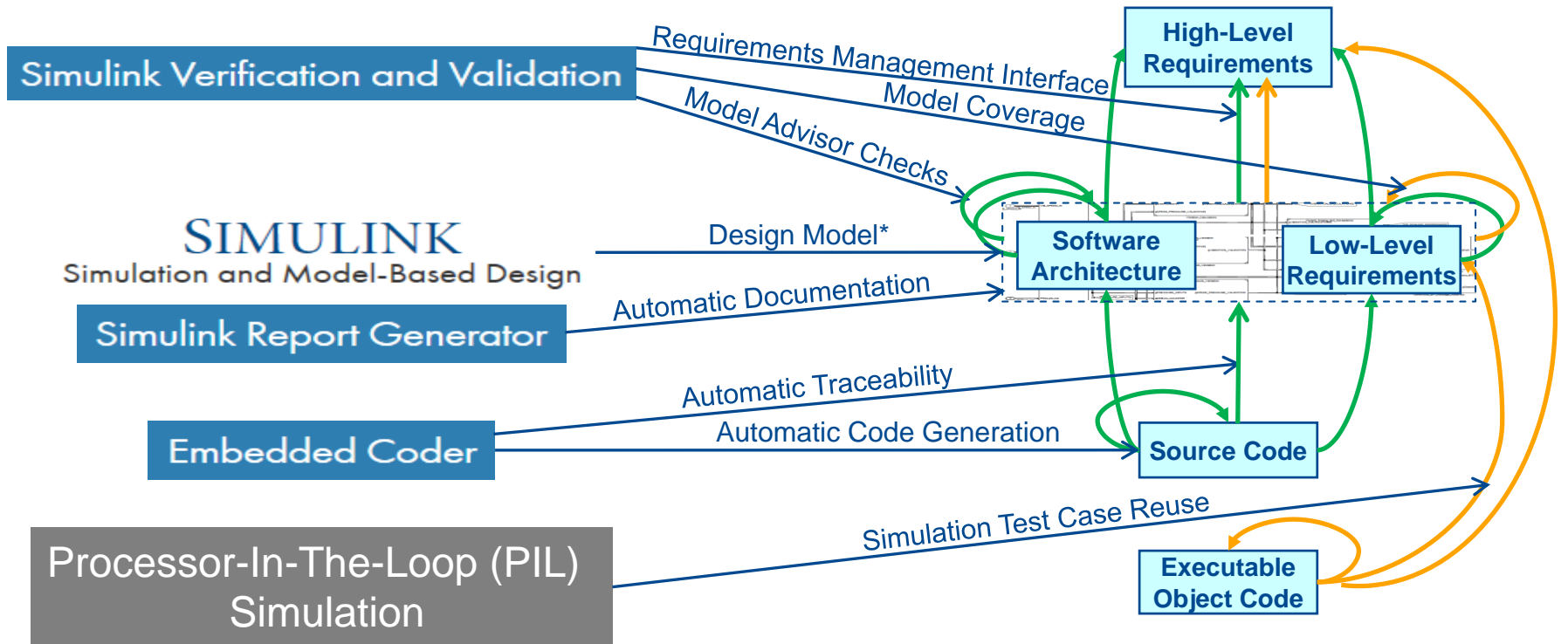
The EMU is designated as a DAL C unit as misleading indication could lead to increased pilot workload and as such it must be certified by EASA against the standard “Software Considerations In Airborne Systems and Equipment Certification” also known as DO-178 (ED-12)



What steps are required for DO-178C certification



What steps are required for DO-178C certification



“Orange indicates Testing
Green indicates Review/Analysis”



Rolls-Royce

* A Design Model includes low-level requirements and/or architecture [DO-178C MB.1.6.2]

Development Process

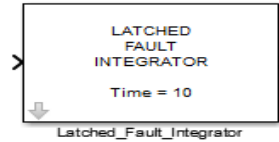
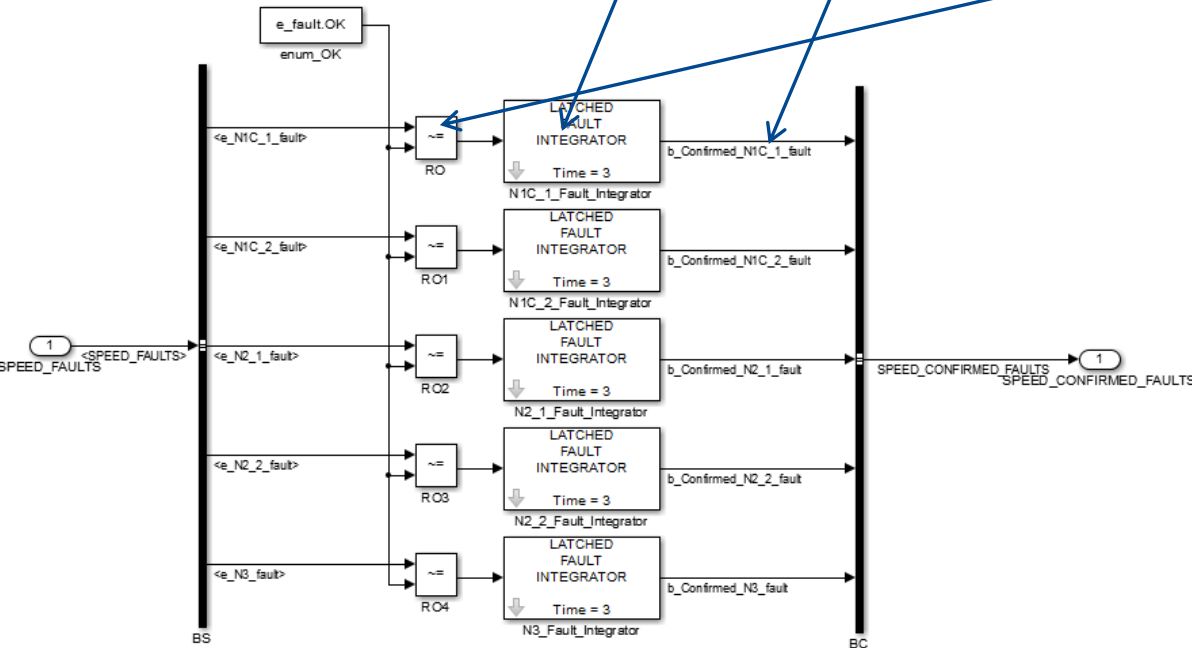


Rolls-Royce

Model development from requirements

Starting with an example High Level Requirement:

Rqmt : The Boolean fault status associated with the N1C-1 speed signal shall be confirmed and latched as being True if the N1C-1 fault status is continuously set to any other state than OK for more than 3 seconds.



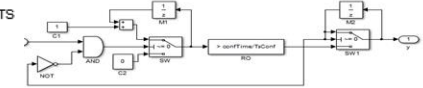
Latched Fault Integrator

Library - Boolean Operators



Input of the block is set Boolean True when the input signal has been continuously Boolean True for the time defined in the mask 'Confirmation Time' parameter the output is set True the state is latched until the end of the simulation

Block Functionality



The block takes the Boolean input to determine whether to start a counter. If the input is True the counter starts counting in increments of 1 every sample period. When the counter value is greater than the confirmation time divided by the sample time the output is set True and a latch is set to keep the state at True. The latched state is combined with the input signal to disable the counter once the output is set True. This ensures that the counter never saturates (this presumes that the confirmation time will always be set less than 4294967295 sample times).

Type Support

The Latched Fault Integrator block is designed to work with Boolean inputs.

Parameter Dialogue Box



Model development from requirements

Adding traceability between the model and the requirement.

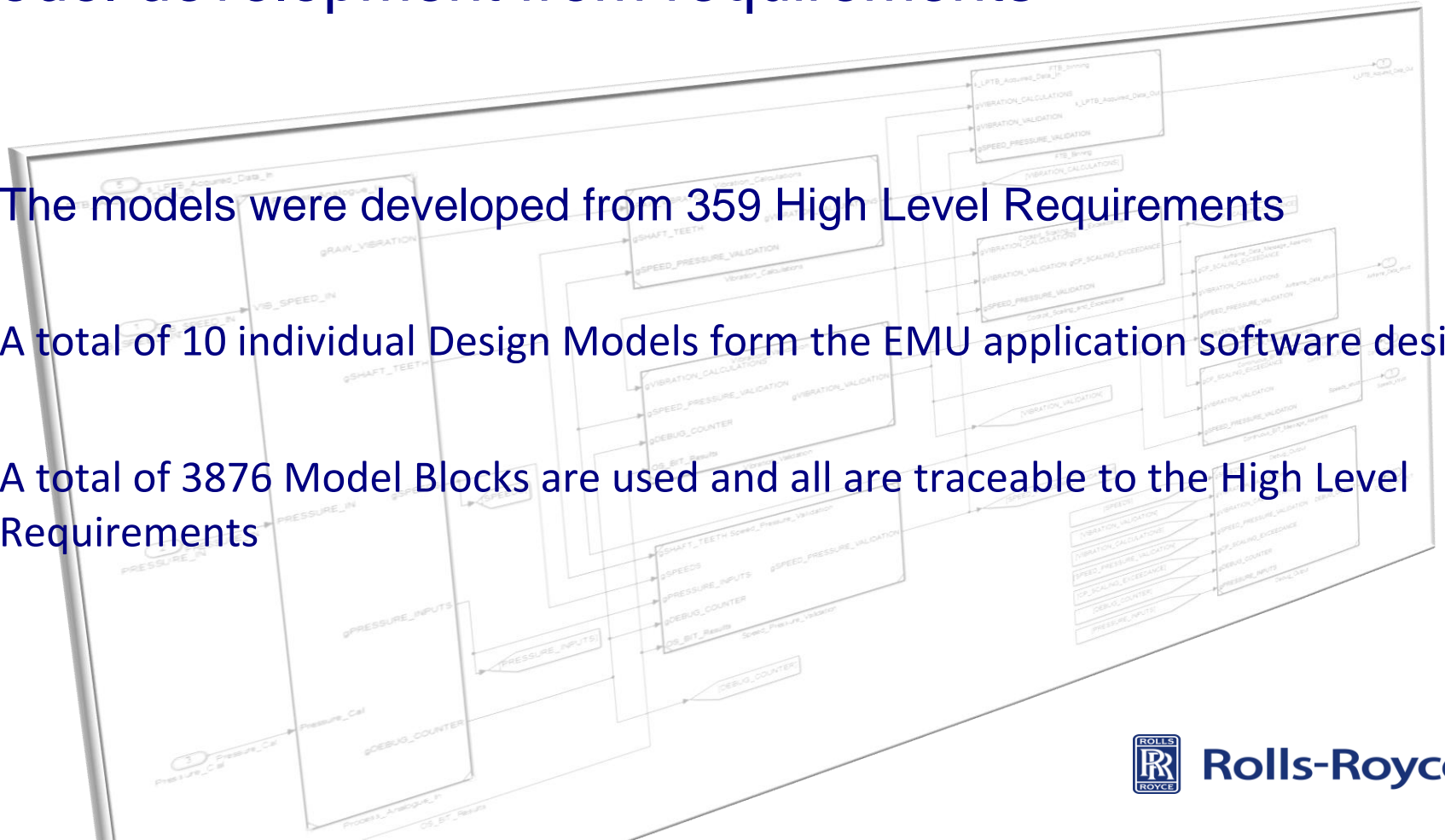
The **Requirements Management Interface** from the **Verification and Validation Toolbox** is used to add bidirectional traceability.

The screenshot displays the DOORS synchronization interface. On the left is a Simulink model with several Latched Fault Integrator blocks. The central window shows a list of requirements with a context menu open over the second item: "4.2.3.0-77 The Boolean fault status associated with the N1C-1 speed sign". The menu options include "Add link to current DOORS object", "Select for linking with Simulink", and "Delete All Links...". At the bottom, a dialog box offers to "Save DOORS surrogate module" and "Save Simulink model (recommended)", with "Synchronize", "Cancel", and "Save settings" buttons.

Object Version	Object Status	Reason for Change
1	Unchanged	
2	Unchanged	
3	Unchanged	
4	Unchanged	
5	Unchanged	
6	Unchanged	
7	Unchanged	
8	Unchanged	
9	Unchanged	
10	Unchanged	
11	Unchanged	
12	Unchanged	
13	Unchanged	
14	Unchanged	
15	Unchanged	
16	Unchanged	
17	Unchanged	
18	Unchanged	
19	Unchanged	
20	Unchanged	
21	Unchanged	
22	Unchanged	
23	Unchanged	
24	Unchanged	
25	Unchanged	
26	Unchanged	
27	Unchanged	
28	Unchanged	
29	Unchanged	
30	Unchanged	
31	Unchanged	
32	Unchanged	
33	Unchanged	
34	Unchanged	
35	Unchanged	
36	Unchanged	
37	Unchanged	
38	Unchanged	
39	Unchanged	
40	Unchanged	
41	Unchanged	
42	Unchanged	
43	Unchanged	
44	Unchanged	
45	Unchanged	
46	Unchanged	
47	Unchanged	
48	Unchanged	
49	Unchanged	
50	Unchanged	

Model development from requirements

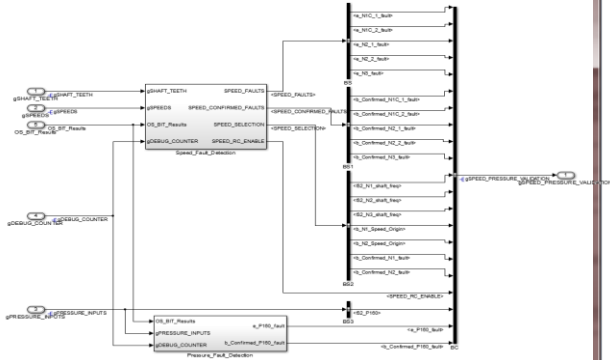
- The models were developed from 359 High Level Requirements
- A total of 10 individual Design Models form the EMU application software design
- A total of 3876 Model Blocks are used and all are traceable to the High Level Requirements



Model verification activities

Check that the model conforms to standards.

The **Model Advisor Checks** from the **Verification and Validation Toolbox** are used.



Model Advisor Report - Speed_Pressure_Validation.slx

Simulink version: 8.1
System: Speed_Pressure_Validation
Model version: 1d
Current run: 07-Nov-2016 15:28:13

Run Summary

<input checked="" type="checkbox"/> Pass	<input checked="" type="checkbox"/> Fail	<input checked="" type="checkbox"/> Warning	<input checked="" type="checkbox"/> Not Run	Total
51	0	2	10	63

Modeling Standards for DO-178C/DO-331

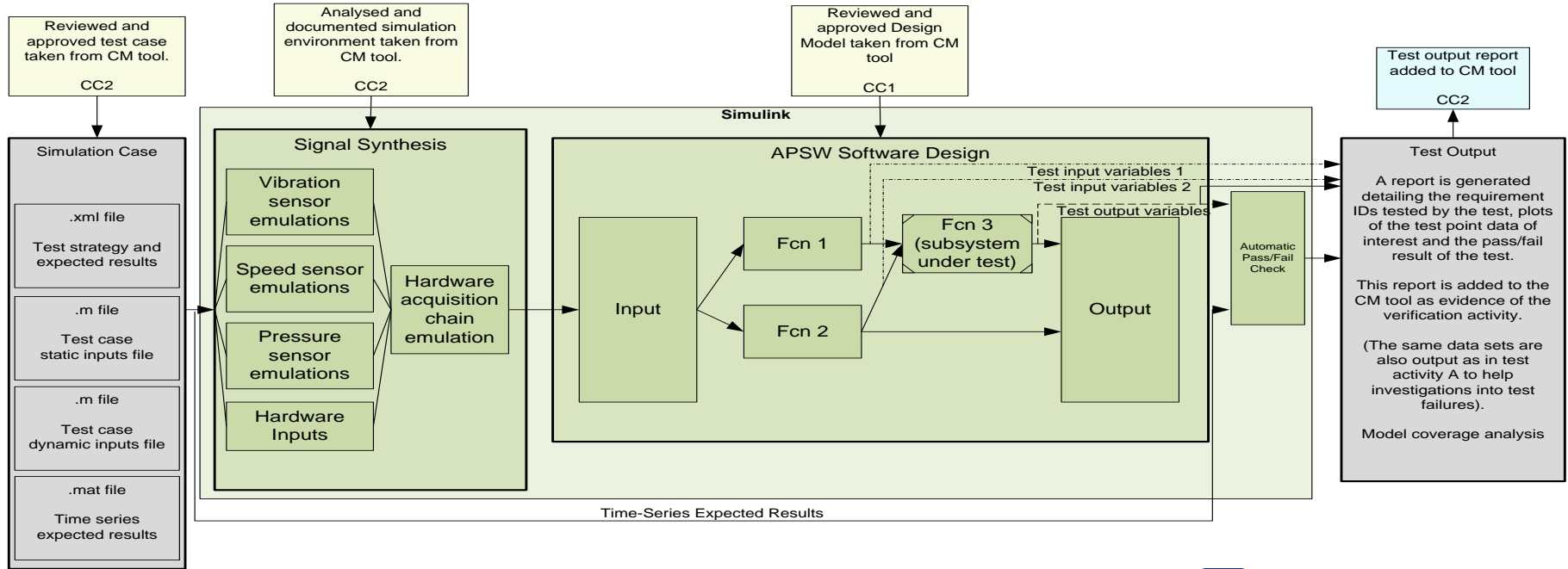
- Display model version information**
Display model configuration and checksum information.

Attribute	Value
Model Version	1d
Author	EMU_DEV_PC1
Date	Wed Nov 02 11:50:32 2016
Model Checksum	1684242041 1349583278 410703400 385426520
- Check safety-related optimization settings**
Check optimization settings in the model configuration that might impact safety.
Verify 'Block reduction' setting
Check whether the Block reduction check box is cleared.
Passed
Optimization > Block reduction is cleared.
- Verify 'Implement logic signals as boolean data (vs. double)' setting**
Check whether the Implement logic signals as boolean data (vs. double) check box is selected.
Passed
Optimization > Implement logic signals as Boolean data (vs. double) is selected.
- Verify 'Application lifespan' setting**

Model verification activities

Simulate the model

Simulation cases written against the high level requirements were used to fully exercise the model using a test tool developed in house.



Model verification activities

Simulation activity outputs - Results Report

The Test Results are Simulation Results for the Model. The results returned

SPV_TEST26

This is a Matlab auto-generated report. It provides a results summary of requirement testing performed on a set of Design Model blocks targeting EMU APSW functionality. Refer to the test process documentation.

TEST SUMMARY

Tester Identification: RRLocal [EMU_DEV_PC1]
EMU-DEV-PC-ONE

Date and Time: 10 March 2017 17:03:24

Unit Under Test: Design Model

Module Name	Compiled Checksum	Unit Version	Configured Version
Speed Pressure Validation	1684242041-1349583278-410703400-385426520	1D	MATCHED

TEST RESULT PASS

A point-to-point comparison of output signals is carried out against a manually reviewed set of expected signals. A PASS result indicates that all output signals yield their expected values at all times during the simulation.

REQUIREMENTS UNDER TEST

Requirement number	Requirement Description
203	The Boolean fault status associated with the NIC-1 speed signal shall be confirmed and latched as being True until the Application Layer is re-initialised, if the NIC-1 fault status is continuously set any other state than OK for more than 3 seconds.
541	The Boolean fault status associated with the NIC-1 speed signal shall be initialised to False.

Linked Requirements: 203 541

Expected Results: The hardware fail flag is set to fail for the first 2.9952s which is 1 software cycle under the three second timer therefore the fault should not be confirmed.

The Confirmation output is initialised to 0.

At 3.072s a combination of continuous faults are introduced; these are out of range low then high then hardware acquisition fail. The combined failures persist for longer than 3 seconds so the fault is confirmed at 6.0928s.

All faults are then cleared and the confirmed fault stays latched.

Floating-Point Tolerance: Pass/Fail tolerance value set to 0.

Signal Name	Description	Data Type	Dims
e_NIC_1_fault	Enumeration indicating operational state of the NIC_1 sensor	Enum: e_fault	1
b_Confirmed_NIC_1_fault	Confirmed fault with NIC_1 sensor, True: NIC-1 Fault (latched)	boolean	1

Figure 2: Matlab Figure: SPV_Test26_OutputPlot1.fig

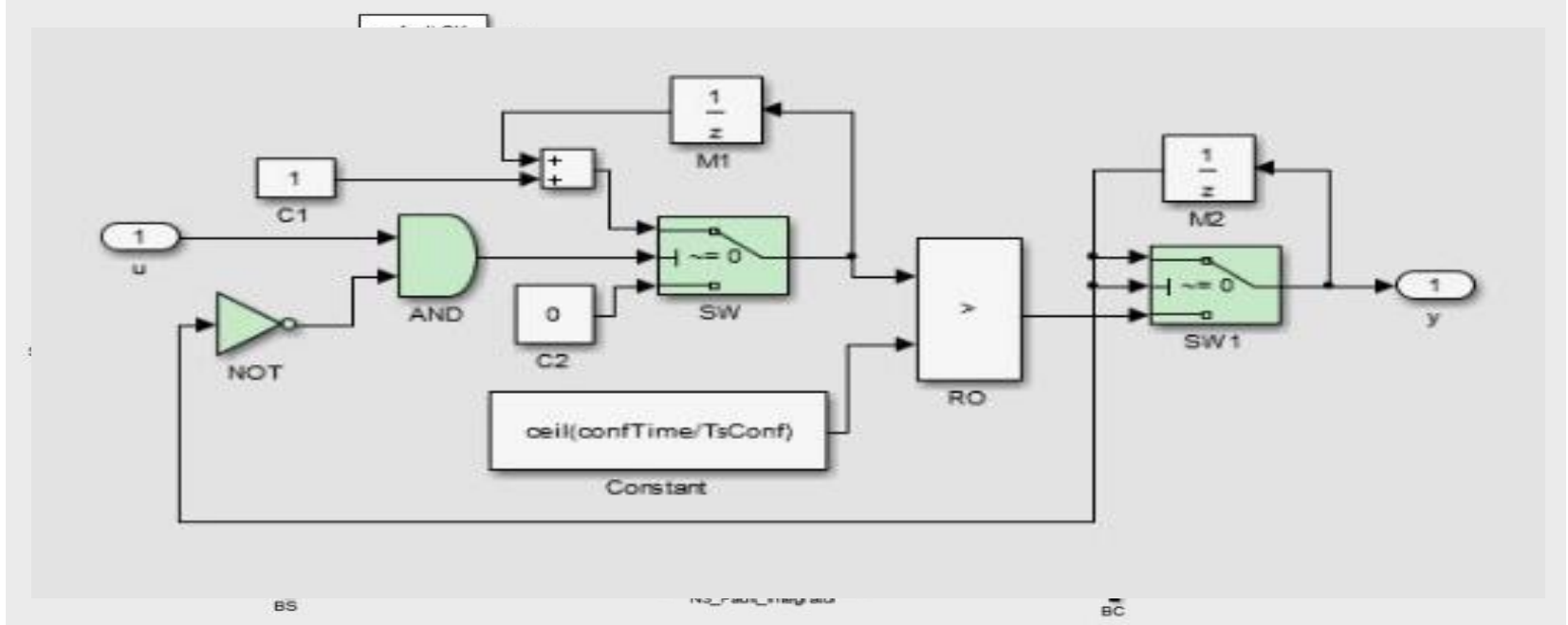
Model-in-the-Loop: Simulink Test Case Auto-Generated Report



Model verification activities

Simulation activity outputs – Model Coverage

Decision and Condition



Model verification activities

Simulation activity outputs – Model Coverage Report

27. SubSystem block "NIC 1 Fault"

Model Hierarchy/Complexity:

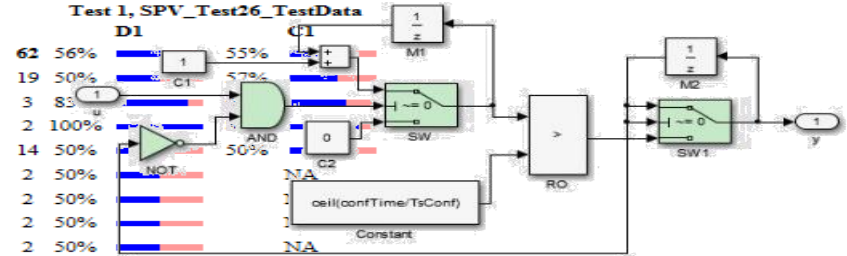
Parent:	<u>Speed Pressure</u>	1. <u>Speed Pressure Validation</u>
		2. <u>Pressure Fault Detection</u>
Metric	Coverag	3. <u>Fault Confirmation</u>
Cyclomatic Complexity	0	4. <u>NIC 1 Fault Integrator</u>
Condition (C1)	NA	5. <u>P160 Calibration Fail</u>
Decision (D1)	NA	6. <u>P160_pres_0V5Cal</u>
		7. <u>P160_pres_0VCal</u>
		8. <u>P160_temp_0VCal</u>
		9. <u>P160_temp_2V5Cal</u>
		10. <u>Pressure Excitation Wrap</u>
		11. <u>Pressure Excitation Wrap 0V Cal</u>
Metric	Coverag	12. <u>Pressure Excitation Wrap 5V Cal</u>
Cyclomatic Complexity	0	13. <u>P160 Range Check</u>
Condition (C1)	100% (4)	14. <u>P160 Fault</u>
		15. <u>Speed Fault Detection</u>
		16. <u>HW Speed Faults</u>
		17. <u>OPR HW Fail</u>
		18. <u>Resetable Fault Integrator</u>
		19. <u>Count Clear</u>
		20. <u>Count Set</u>
		21. <u>Reset Priority Latch</u>
		22. <u>Resetable Fault Integrator1</u>
		23. <u>Count Clear</u>
		24. <u>Count Set</u>
		25. <u>Reset Priority Latch</u>
		26. <u>Speed Fault Confirmation</u>
Metric	Coverag	27. <u>NIC 1 Fault Integrator</u>
Cyclomatic Complexity	0	28. <u>NIC 2 Fault Integrator</u>
Condition (C1)	100% (2)	29. <u>N2 1 Fault Integrator</u>
		30. <u>N2 2 Fault Integrator</u>
		31. <u>N3 Fault Integrator</u>
		32. <u>Speed Range Check Enable</u>
		33. <u>Speed Range Checks</u>

Logic block "AND"

Parent:	<u>Speed Pressure</u>
Metric	Coverag
Cyclomatic Complexity	0
Condition (C1)	100% (4)
Conditions analyzed:	
Description:	
input port 1	
input port 2	

Logic block "NOT"

Parent:	<u>Speed Pressure</u>
Metric	Coverag
Cyclomatic Complexity	0
Condition (C1)	100% (2)
Conditions analyzed:	
Description:	
input port 1	

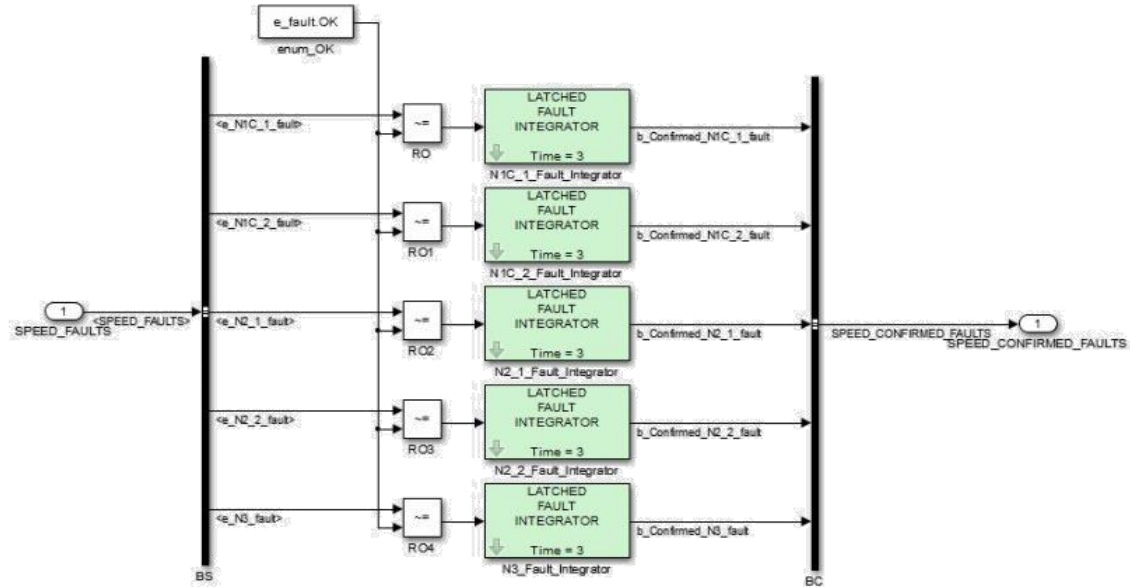


62	56%		
19	50%		
3	83%		
2	100%		
14	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
2	50%		
42	58%	55%	
5	60%	54%	
8	50%	55%	
4	50%	50%	
2	50%	50%	
2	50%	50%	
2	50%	50%	
0	NA	50%	
4	50%	50%	
2	50%	50%	
2	50%	50%	
0	NA	50%	
10	60%	60%	
2	100%	100%	
2	50%	50%	
2	50%	50%	
2	50%	50%	
2	50%	50%	
2	50%	50%	
0	NA	60%	
15	57%		NA

Model verification activities

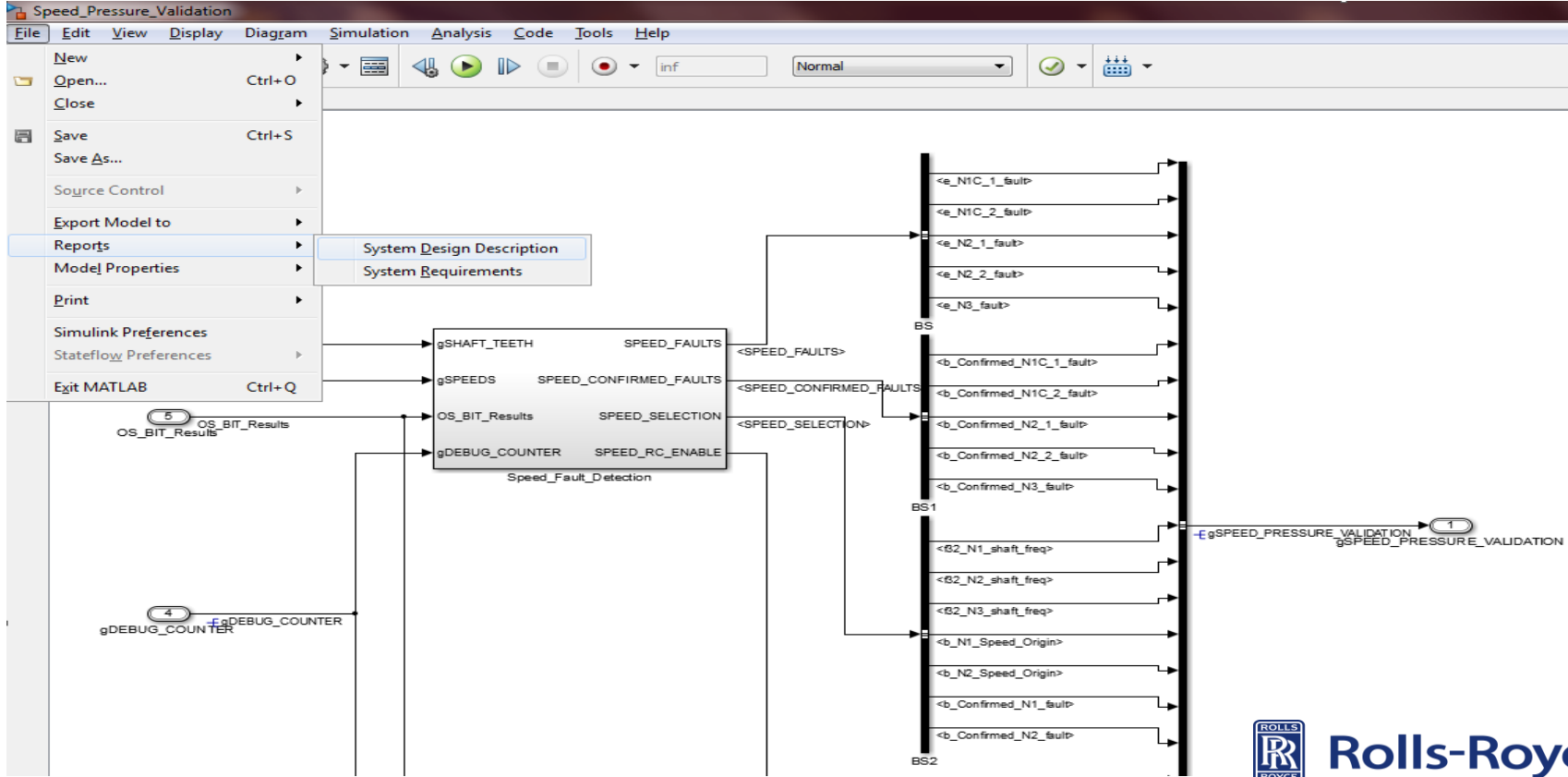
Simulation activity outputs – Model Coverage Report - Cumulative

19.....	Count_Clear	2	100%	100%	100%
20.....	Count_Set	2	100%	100%	100%
21.....	Reset_Priority_Latch	0	NA	100%	100%
22.....	Resetable_Fault_Integrator1	4	100%	100%	100%
23.....	Count_Clear	2	100%	100%	100%
24.....	Count_Set	2	100%	100%	100%
25.....	Reset_Priority_Latch	0	NA	100%	100%
26.....	Speed_Fault_Confirmation	10	100%	100%	100%
27.....	NIC_1_Fault_Integrator	2	100%	100%	100%
28.....	NIC_2_Fault_Integrator	2	100%	100%	100%
29.....	N2_1_Fault_Integrator	2	100%	100%	100%
30.....	N2_2_Fault_Integrator	2	100%	100%	100%
31.....	N3_Fault_Integrator	2	100%	100%	100%
32.....	Speed_Range_Check_Enable	0	NA	100%	100%
33.....	Speed_Range_Checks	15	100%	NA	NA
34.....	NIC_1_Fault	2	100%	NA	NA
35.....	NIC_2_Fault	2	100%	NA	NA
36.....	N2_1_Fault	2	100%	NA	NA
37.....	N2_2_Fault	2	100%	NA	NA
38.....	N3_Fault	2	100%	NA	NA
39.....	Speed_Selection	4	100%	100%	100%



Model verification activities

The model is automatically documented using the report generation tool



Model verification activities

The Design Description Report captures the complete In-Memory Representation of the Model

The screenshot displays a Design Description Report (DDR) interface. On the left is a Table of Contents with the following items:

- Table of Contents
- Chapter 1. Model Version
- Chapter 2. Root System
- Chapter 3. Subsystems
- Chapter 4. System Design Variables
- Chapter 5. Requirements Traceability
- Chapter 6. System Model Configuration** (highlighted)
- Chapter 7. Glossary
- Chapter 8. About this Report

The main area shows a list of error messages from the model verification process:

SolverPrimCheckMsg	error
InheritedTsInSrcMsg	error
DiscreteInheritContinuousMsg	error
MultiTaskDSMMsg	error
MultiTaskCondExecSysMsg	error
MultiTaskRateTransMsg	error
SingleTaskRateTransMsg	error
TasksWithSamePriorityMsg	error
SigSpecEnsureSampleTimeMsg	error
CheckMatrixSingularityMsg	error
IntegerOverflowMsg	error
Int32ToFloatConvMsg	warning
ParameterDowncastMsg	error
ParameterOverflowMsg	error
ParameterUnderflowMsg	error
ParameterPrecisionLossMsg	error
ParameterTunabilityLossMsg	error
FixptConstUnderflowMsg	none
FixptConstOverflowMsg	none
FixptConstPrecisionLossMsg	none
UnderSpecifiedDataTypeMsg	error

Below the report, a block diagram shows a component named "FAULT INTEGRATOR" (N3_Fault_Integrator). It has an input port "<e_N3_fault>" and an output port "b_Confirmed_N3_fault". The component is connected to a block labeled "RO4" and another block labeled "BS". The fault integrator block shows "Time = 3".

Model verification activities

- 129 Test Cases were authored against the high level requirements
- A total of 1.7 million test points compared in the automatic pass/fail reporting
- Testing against high level requirements achieved the necessary level of model coverage
- The functionality of the design is fully validated before a single line of code is produced

Code production and verification

10 10:10
16 11:57
16 11:59
16 16:30
16 10:22
16 16:41
16 15:57
16 15:42
16 16:24
16 14:55
16 15:57
16 15:57
16 11:31
16 15:57
16 15:57
16 15:57
16 15:34

Code Generation Report

Back Forward

Contents

- Summary
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

Generated Code

- [-] Model files**
 - [Speed Pressure Validation.c](#)
 - [Speed Pressure Validation.h](#)
 - [Speed Pressure Validation private.h](#)
 - [Speed Pressure Validation types.h](#)
- [+] Utility files (2)**
- [+] Interface files (1)**

```
2338
2339 /* End of Switch: '<S17>/SW4' */
2340
2341 /* UnitDelay: '<S36>/M2' */
2342 Speed_Pressure_Validation_B.M2_ktmo =
2343 Speed_Pressure_Validation_DW.M2_DSTATE_inel;
2344
2345 /* UnitDelay: '<S36>/M1' */
2346 Speed_Pressure_Validation_B.M1_hqpa =
2347 Speed_Pressure_Validation_DW.M1_DSTATE_hllb;
2348
2349 /* RelationalOperator: '<S19>/RO' incorporates:
2350 * Constant: '<S19>/enum_OK'
2351 */
2352 Speed_Pressure_Validation_B.RO_hixv =
2353 (Speed_Pressure_Validation_B.e_N1C_1_fault_kiyw != rtcP_pooled17);
2354
2355 /* Logic: '<S36>/NOT' */
2356 Speed_Pressure_Validation_B.NOT_11rk = !Speed_Pressure_Validation_B.M2_ktmo;
2357
2358 /* Logic: '<S36>/AND' */
2359 Speed_Pressure_Validation_B.AND_gxyf = ((Speed_Pressure_Validation_B.RO_hixv) &&
2360 (Speed_Pressure_Validation_B.NOT_11rk));
2361
2362 /* Switch: '<S36>/SW' incorporates:
2363 * Constant: '<S36>/C2'
2364 */
2365 if (Speed_Pressure_Validation_B.AND_gxyf) {
2366 /* Sum: '<S36>/Sum' incorporates:
2367 * Constant: '<S36>/C1'
2368 */
2369 Speed_Pressure_Validation_B.Sum_awv1 = Speed_Pressure_Validation_B.M1_hqpa +
2370 rtcP_pooled12;
2371 Speed_Pressure_Validation_B.SW_feqe = Speed_Pressure_Validation_B.Sum_awv1;
2372 } else {
2373 Speed_Pressure_Validation_B.SW_feqe = rtcP_pooled11;
2374 }
2375
```

OK Help



Code production and verification

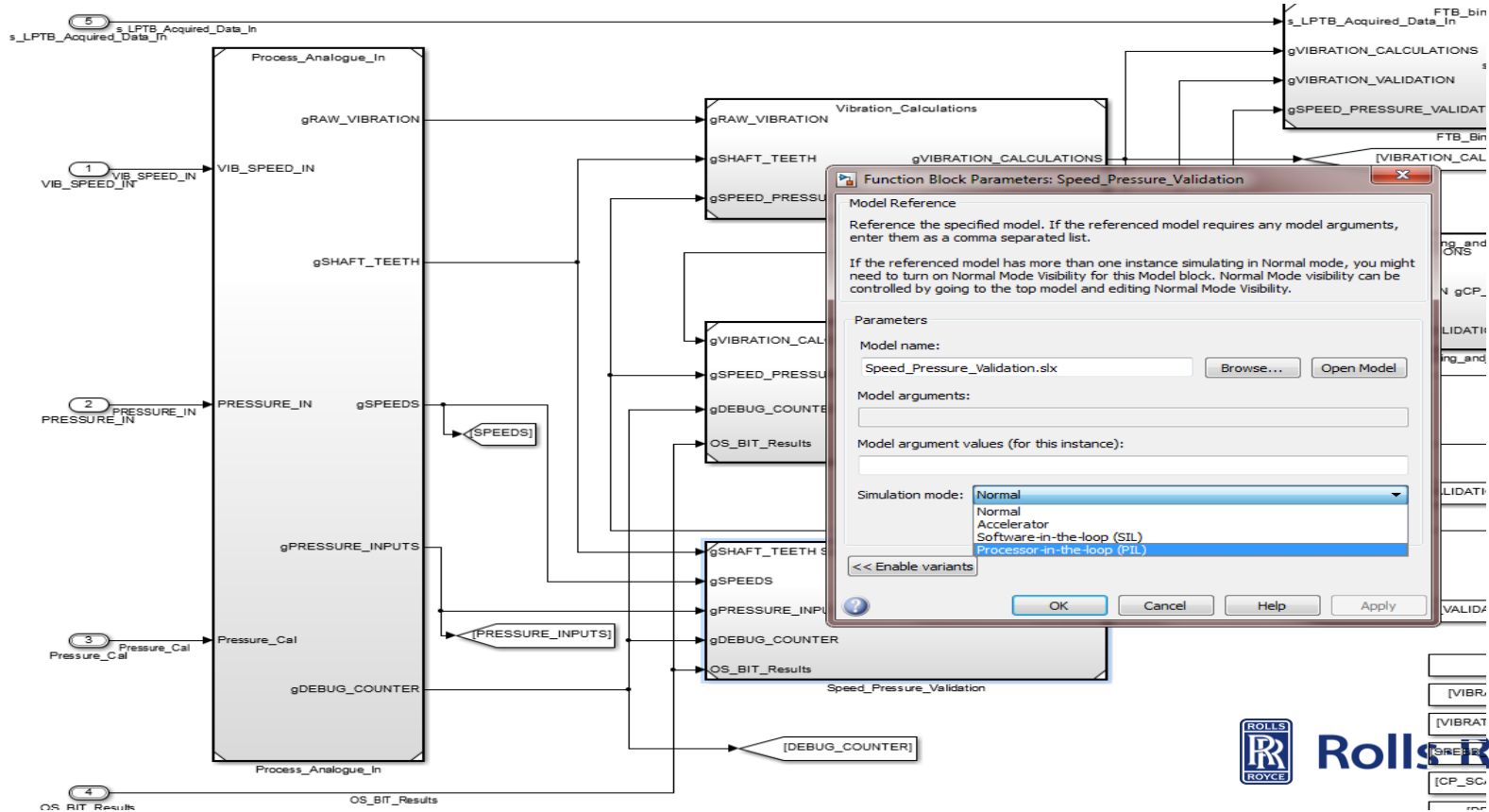
- The compiled code must be tested to show compliance with the high level requirements.
 - Tests written against the high level requirements already exist from the model simulation.
- The compiled code must be tested to show compliance against the Design Model.
 - This can be achieved by equivalence testing the object code behaviour against the Design Model behaviour.
- The compiled code must be shown to be compatible with the target architecture.
 - This can be achieved by running all paths of the compiled code on representative target architecture

Is there a way that all this can be achieved with the test cases which already exist?



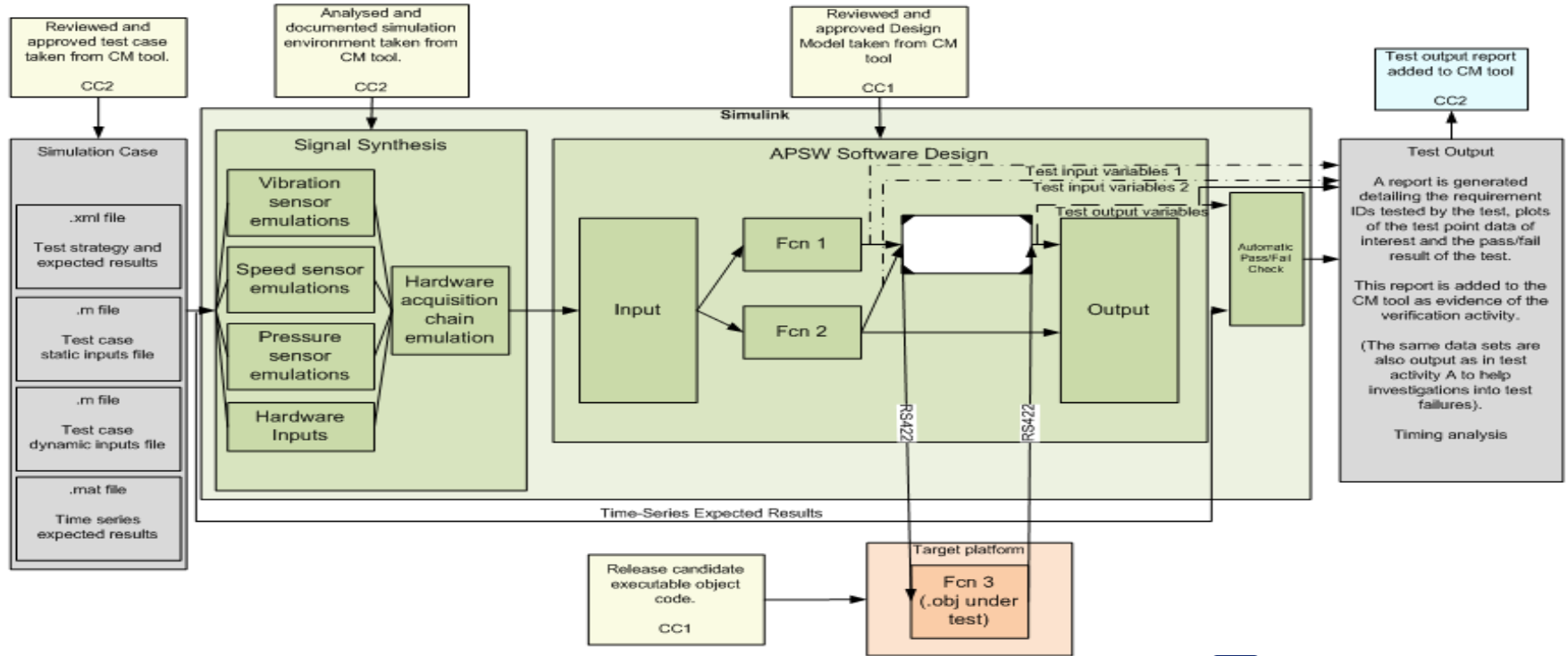
Code production and verification

Processor-in-the-Loop testing



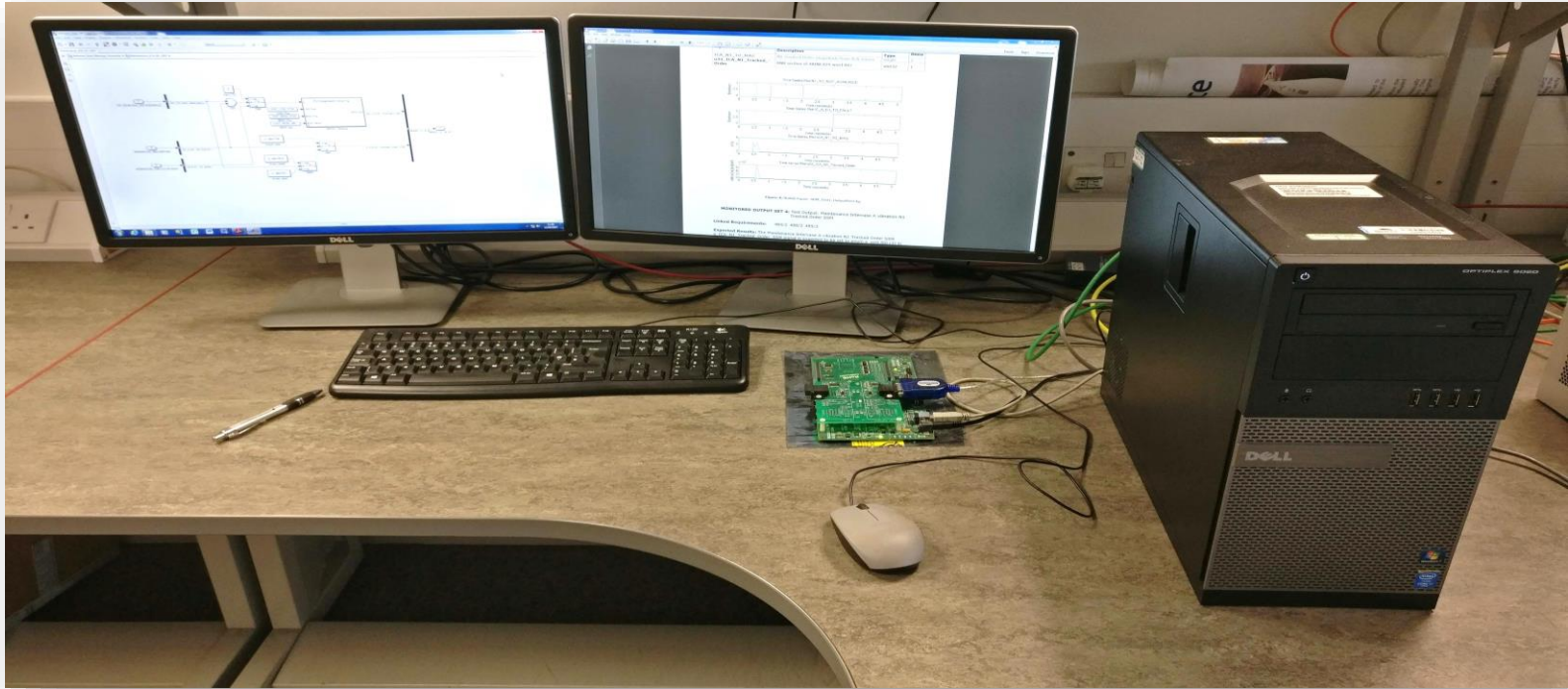
Code production and verification

Processor-in-the-Loop testing



Code production and verification

Processor-in-the-Loop testing



Code production and verification

Processor-in-the-Loop testing – Results Report

The number of test objects, timing and definitions reported

REQUIREMENT TESTING [PIL] SPV_TEST26

This is a Matlab auto-generated report. It provides a results summary of requirement testing performed on a set of Software Object. Code blocks targeting EMU APSW functionality. Refer to the test process documentation.

TEST SUMMARY

Tester Identification: RRLocal [EMU_DEV_PC1]
EMU_DEV_PC-ONE

Date and Time: 20 March 2017 17:41:16

Unit Under Test: Speed Pressure Validation

Module Name	Software Object Code Unit Version
Speed Pressure Validation	1.0

TEST RESULT PASS

A point-to-point comparison of output signals is carried out against a manually reviewed set of expected signals. A PASS result indicates that all output signals yield their expected values at all times during the simulation.

CODE EXECUTION PROFILE

Profiled Code Section	Maximum Execution Time [us]	Average Execution Time [us]	Calls
1 Speed_Pressure_Valid_initialize	2.3733	2.3733	1
2 Speed_Pressure_Validation [0.0256 0]	190.24	162.6198	261

REQUIREMENTS UNDER TEST

Requirement Number	Requirement Description
1 203	The Boolean fault status associated with the N1C-1 speed signal shall be confirmed and latched as being True until the Application Layer is re-initialised, if the N1C-1 fault status is continuously set any other state than OK for more than 3 seconds.
2 541	The Boolean fault status associated with the N1C-1 speed signal shall be initialised to False.

Linked Requirements: 203 541

Expected Results: The hardware fail flag is set to fail for the first 2.9952s which is 1 software cycle under the three second timer therefore the fault should not be confirmed.

The Confirmation output is initialised to 0.

At 3.072s a combination of continuous faults are introduced; these are out of range low then high then hardware acquisition fail. The combined failures persist for longer than 3 seconds so the fault is confirmed at 6.0928s.

All faults are then cleared and the confirmed fault stays latched.

Floating-Point Tolerance: Pass/Fail tolerance value set to 0.

Signal Name	Description	Data Type	Dims
e_N1C_1_fault	Enumeration indicating operational state of the N1C_1 sensor	Enum: e_fault	1
b_Confirmed_N1C_1_fault	Confirmed fault with N1C_1 sensor; True: N1C-1 Fault (latched)	boolean	1

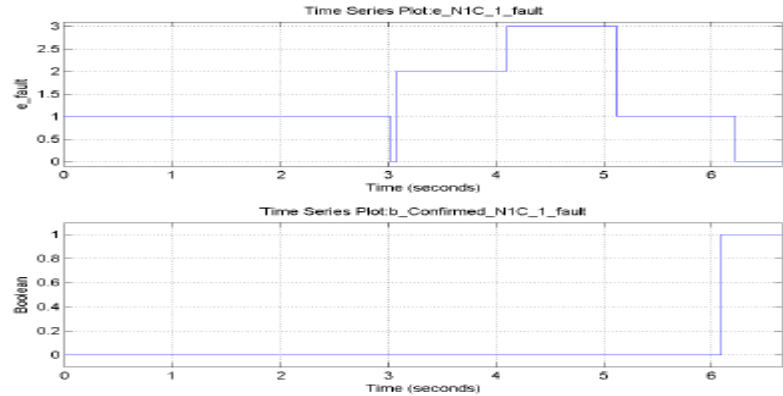


Figure 2: Matlab Figure: SPV_Test26_OutputPlot1.fig

Model-in-the-Loop: Simulink Test Case Auto-Generated Report



Tool Qualification

When is tool qualification required?

“Qualification of a tool is needed when processes of this document are **eliminated, reduced, or automated** by the use of a software tool without its output being verified [DO-178C 12.2.1]”

Which parts of this process required Tool Qualification?

- The Model Advisor checks automated part of the ‘compliance with standards’ process.
- The Report Generator is qualified to show equivalence with the in-memory representation of the model.
- The Model Coverage tool is qualified as this automates the metrics associated with the ‘model coverage’ process.
- The pass/fail reporting of test results automated parts of the ‘compliance’ processes
- The ‘PIL’ infrastructure automates code coverage.



Tool Qualification

DO Qualification Kit (for DO-178)

- DO Qualification Kit provides documentation, test cases, and procedures that let you qualify Simulink software verification tools
- The kit contains tool qualification plans, tool operational requirements, and other materials required for qualifying software verification tools



Future Enhancements



Rolls-Royce

Further tool adoption

Automated code review

- The source code required a manual review to show compliance against the Design Model.
- This activity could be automated by using the 'Simulink Code Inspector'

Simulink Test

- The in-house test frame work has a maintenance overhead, especially around version migration.
- The introduction of 'Simulink Test' in means we now have an alternative solution which is natively supported by the MathWorks.



Thank You

