

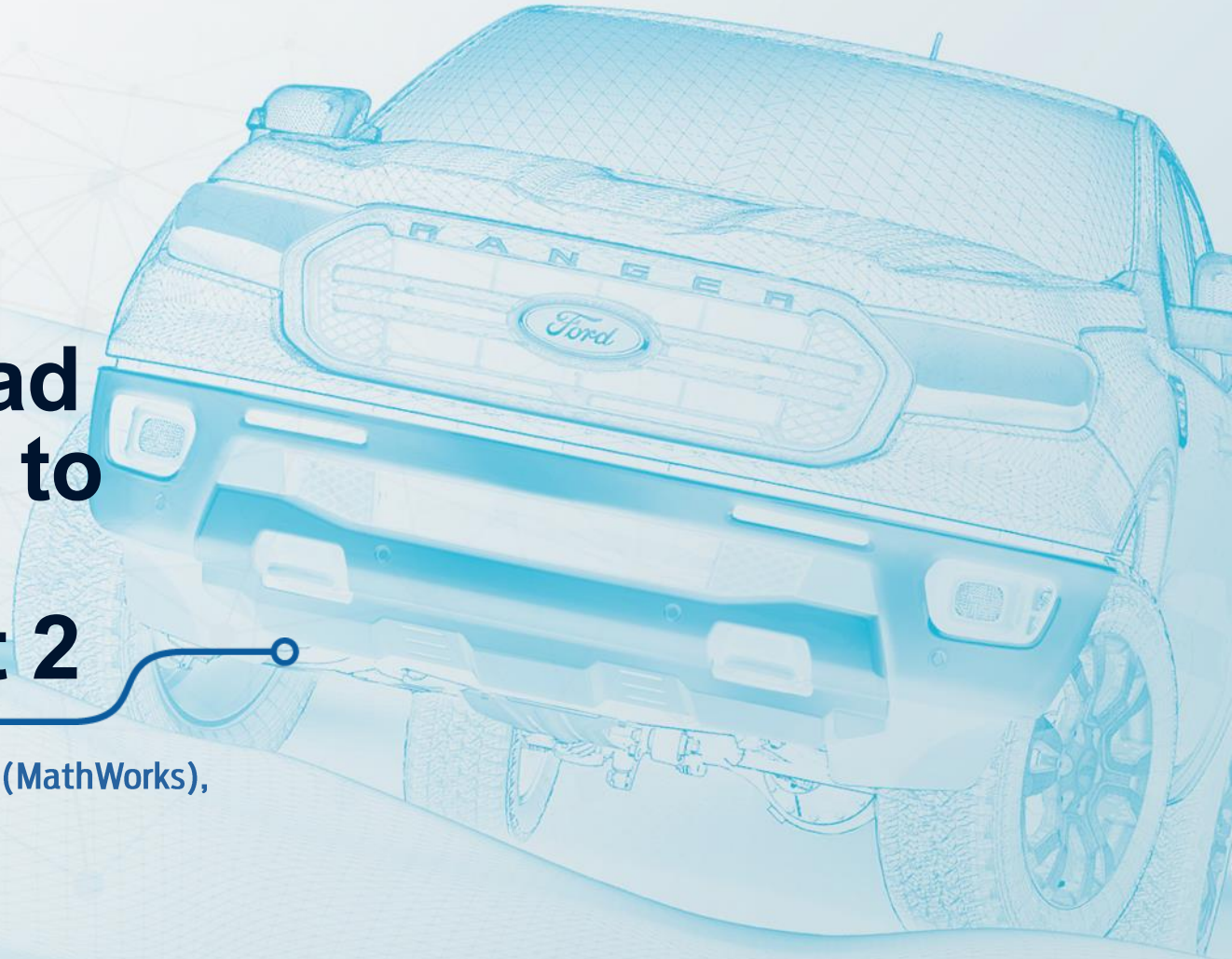


Vehicle Software &
Electronic Solutions

Building the Digital Thread between MBSE and MBD to Meet ISO26262 for Embedded Software Part 2

Authors: Joshua McCready (Ford), Hans Gangwar (Ford), Josh Kahn (MathWorks),
Eric Browning (Ford)

[Part 1 Link](#)



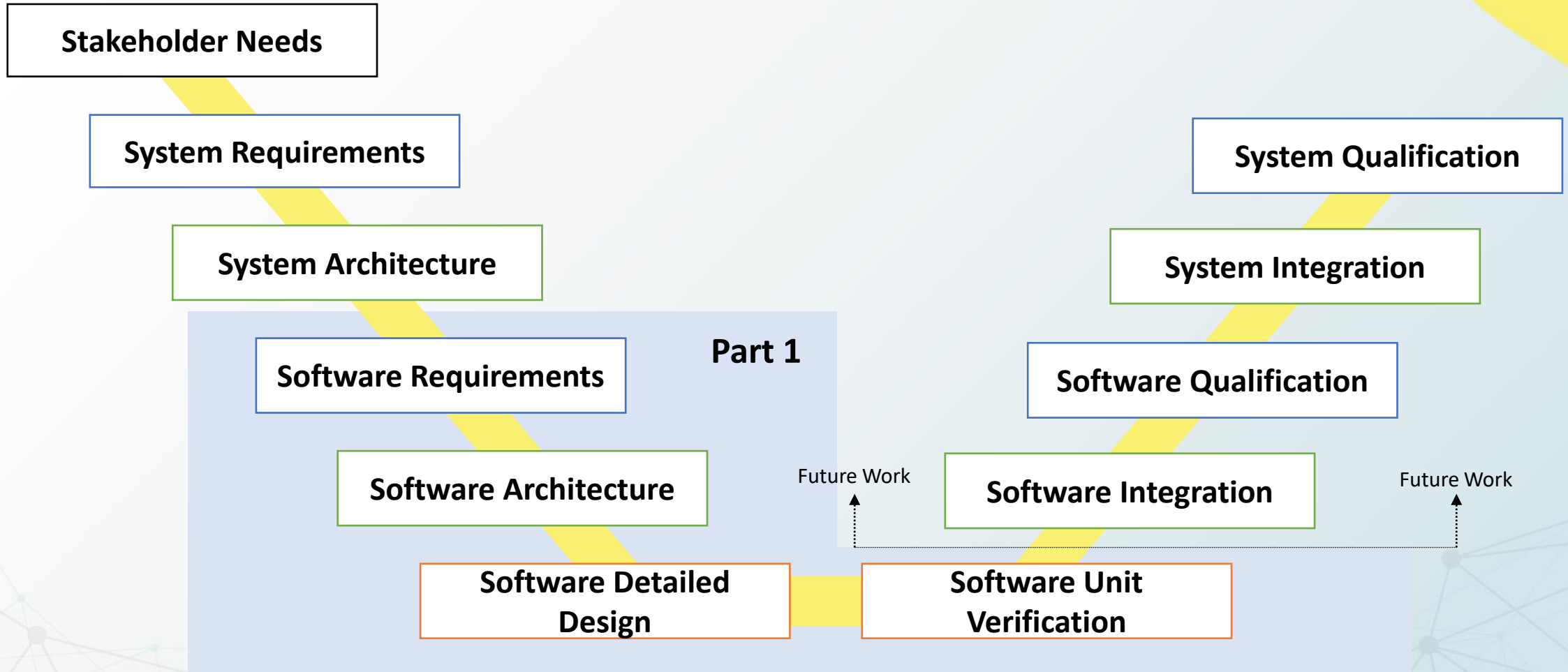
Assessing ISO26262 Part 6 compliance for new and existing Ford In House software developed with Model Based Design software has demonstrated the need for additional best practices

These best practices are needed to achieve connectivity to the System Engineering process and to allow for traceability and thread pulling of SW development artifacts*

Part 1 – SW requirements and design

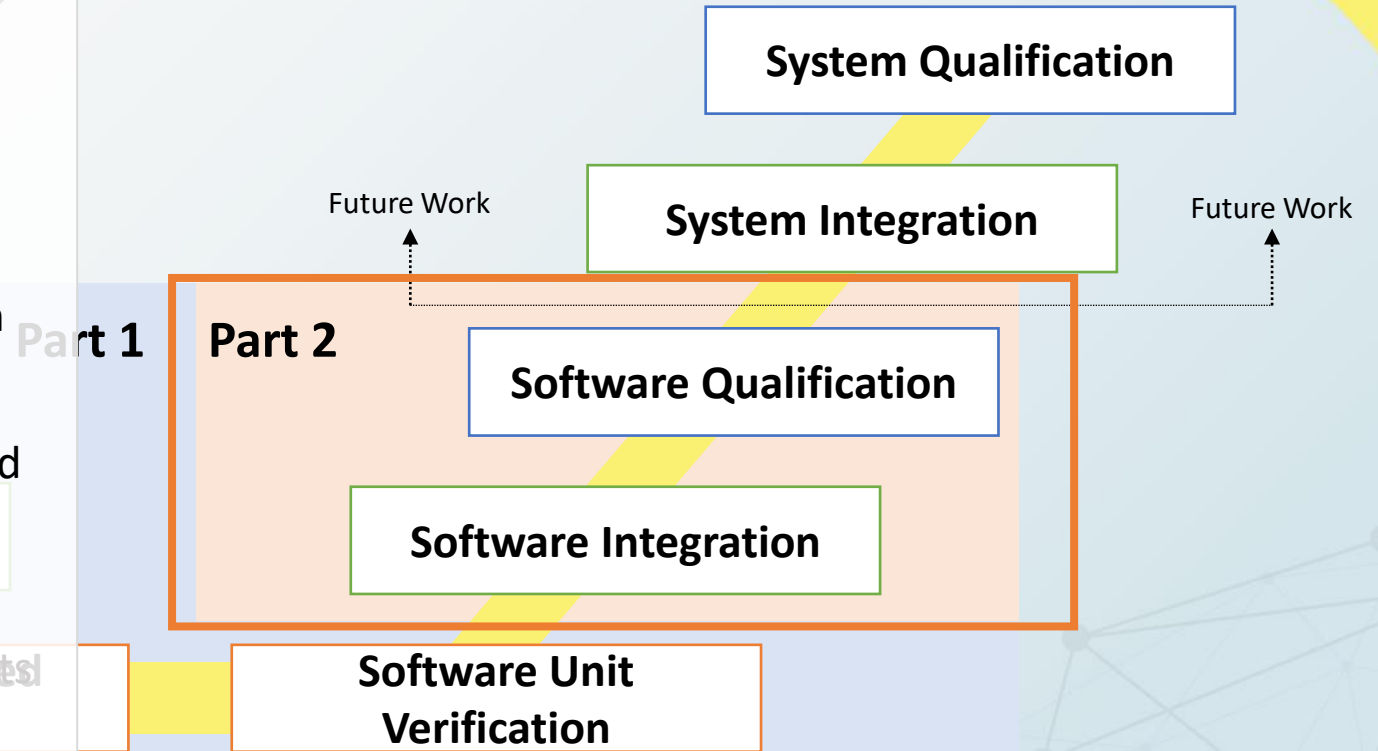
Part 2 - SW Integration and Qualification traceability & Implementation

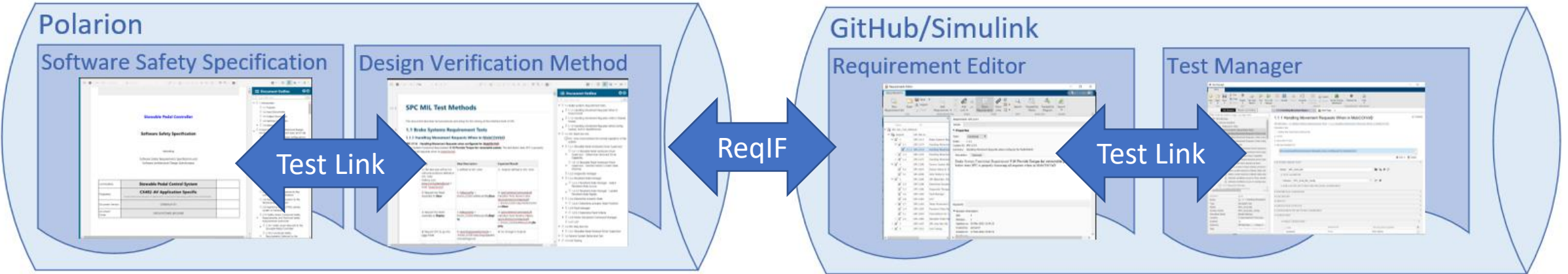
*System and software requirements, model and data dictionary, implementation, test cases



Next Steps

- Continually feedback Software Detailed Design to Software Architecture
- Create Design Verification Methods
- Link test cases to Design Verification Methods
- Create the Software Integration and Qualification Test Suites
- Identify dependencies of software integration and qualification testing and how to establish traceability across the project artifacts
- Develop System Integration and Qualification Test Suites
- Integrate Software Architecture with System Architecture





Requirement changes impact on test methods ("Suspect Links")

Failed Tests impact on requirements (JIRA)

Polarion

Software Safety Specification

Design Verification Method

Test Link

Requirement changes im

Failed Tests

Polarion - Design Verification Method

SPC-2189 - When Motor demand Faults while the pedal assembly is moving then the Modify Motor Demand shall set the Motor Command to stop (8 iterations)

Test Steps

Step	Step Description	Expected Result
1) This test case will be run with preconditions defined in SPC-2093.	1) Defined in SPC-2093	1) Outputs defined in SPC-2093
2) Request the Pedal Assembly to: a) Stow b) Deploy	2) <u>PIdAsyRq</u> a) ENUM_CODE1:ePdAsyActRq Stow b) ENUM_CODE3:ePdAsyActRq Deploy	2) The Motor Command Left transition as described below: a) <u>SpcOutMotorCommandLeft</u> transition from None to Stow <u>SpcOutMotorCommandLeft</u> -> ENUM_CODE1:eSpcMotorCommand Stow b) <u>SpcOutMotorCommandLeft</u> transition from None to deploy <u>SpcOutMotorCommandLeft</u> -> ENUM_CODE3:ePdAsyActRq Deploy
3a&b) Input current profile to trigger the faults described in step 4	3a&b) TBD	3) a) <u>SpcOutMotorCommandLeft</u> remains ENUM_CODE1:eSpcMotorCommand Stow b) <u>SpcOutMotorCommandLeft</u> remains ENUM_CODE3:ePdAsyActRq Deploy

Work Item Properties

SPC-2189 - When Motor demand Faults while the pedal assembly is moving then the Modify Motor Demand shall set the Motor Command to stop (8 iterations)

Fields

*Severity: Basic

Links

Links

Edit Links

relates to

SPC-2214 - Modify Motor Demand Faulted Stop (8 iterations) verifies

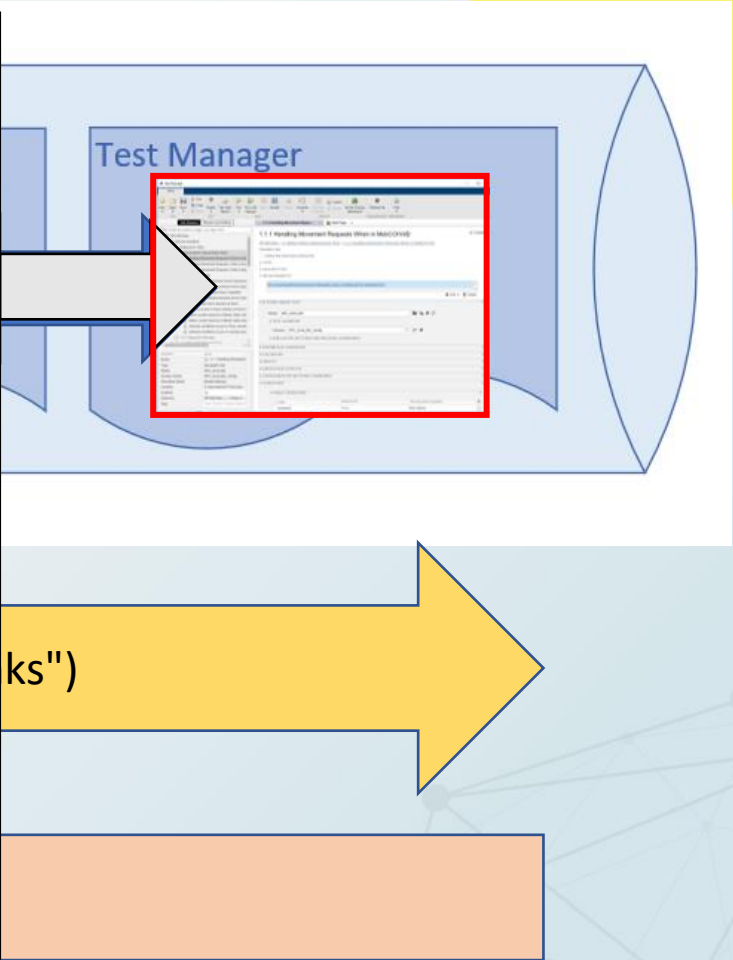
SPC-2205 - Modify Motor Demand Faulted Stop

is related to

ABCS_Jira-371 - Spc v6 Bug - Remove Fault Status 5 fr

Simulink - Test Manager

Links



Test Manager Test Types

Baseline (Current)

Compares the outputs from a simulation against the expected outputs capture in Excel file or a MAT-file.

Pros

- Quickly identify change/regression in model outputs

Cons

- Not adaptive to model changes (PARAM/Configs/Cal)
- Requirement/model changes require re-running baseline and visually inspecting

Simulation (New)

Assess model timing and event ordering by authoring and including temporal assessments with test cases

Pros

- Hybrid systems with discrete and continuous time behavior can require complex timing-dependent signal logic
- Removes the need to visually inspect/update baselines
- Adaptive to most requirement/model changes
- Iterations within one test

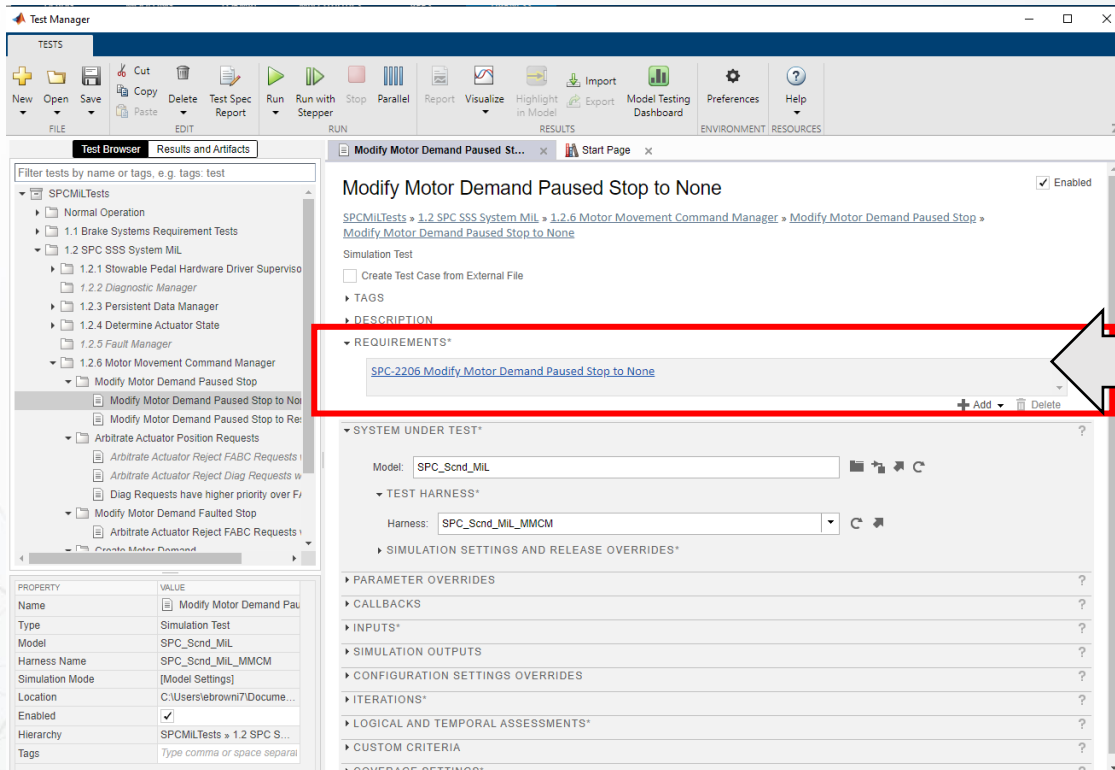
Cons

- Front loads validation
- Based on implementation, potential for missing bug identification (no visual inspection)

Simulation - Requirements

- Link test method Stored in Polarion
- Trace Failed tests to JIRA Bug/Issue

Simulink - Test Manager



Polarion - DVM

SPC-2206 - Modify Motor Demand Paused Stop to None

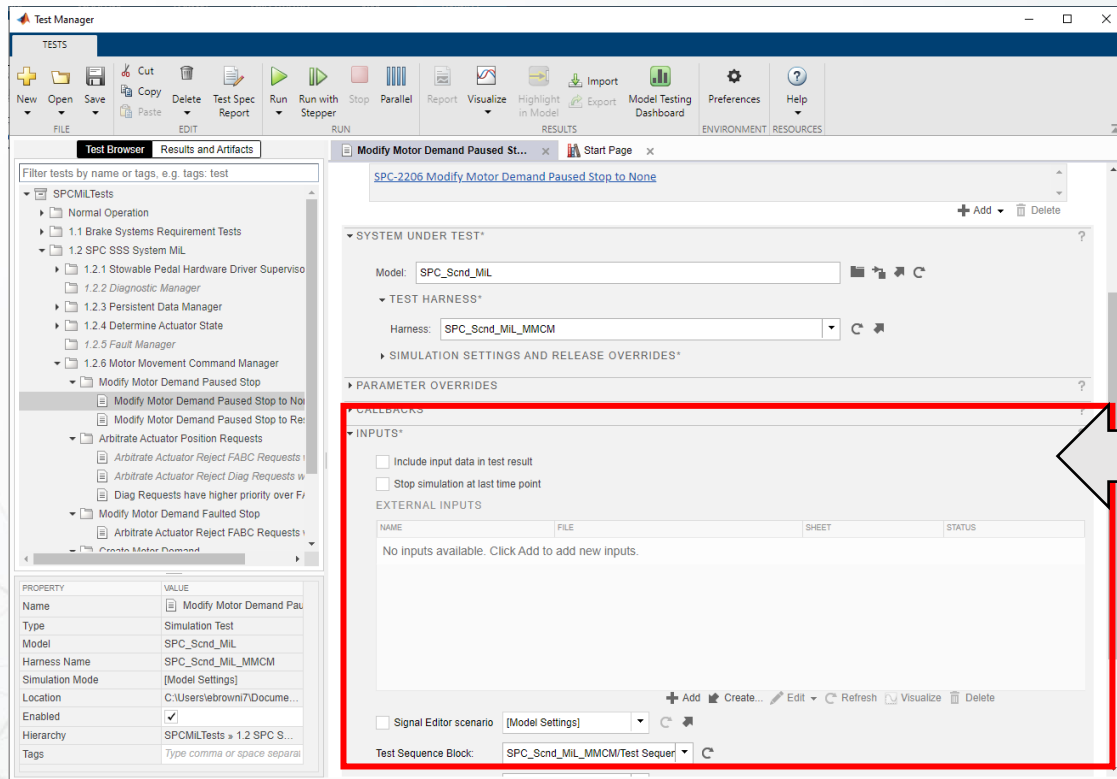
When Motor demand Pause Stop Triggers for longer than Pause time out, while the pedal assembly is moving then the Modify Motor Demand shall set the Motor Command to stop (8 iterations)

Step	Step Description	Expected Result
1) This test case will be run with preconditions defined in SPC-2093.	1) Defined in SPC-2093	1) Outputs defined in SPC-2093
2) Request the Pedal Assembly to: a) Stow b) Deploy	2) <u>PldAsyRq</u> a) ENUM_CODE1:ePdAsyActRq Stow b) ENUM_CODE3:ePdAsyActRq Deploy	2) The Motor Command Left transition as described below: a) <u>SpcOutMotorCommandLeft</u> transition from None to Stow - <u>SpcOutMotorCommandLeft</u> - > ENUM_CODE1:eSpcMotorCommand Stow b) <u>SpcOutMotorCommandLeft</u> transition from None to deploy - <u>SpcOutMotorCommandLeft</u> - > 040:ENUM_CODE2:eSpcMotorCommand Deploy
3a&b) Input current profile	3a&b) TBD	3) a) <u>SpcOutMotorCommandLeft</u> remains ENUM_CODE1:eSpcMotorComma

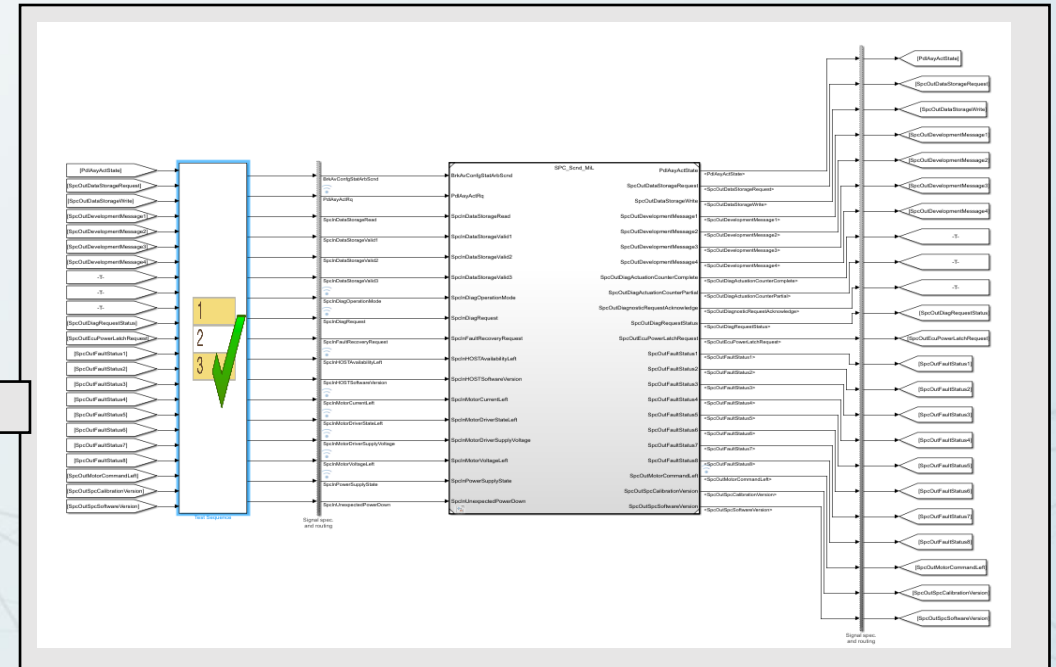
Simulation - Inputs

- Sequence editor allows for I/O closed loop feedback of function under test
- Use of reference model allows for ease of updating to multiple releases

Simulink - Test Manager



Model Test Harness



Simulation - Test Sequence/Iterations

- Run multiple iteration changing input parameters in a single test
- Link Model based Trigger & Temporal based Transitions (i.e. Model Cals, I/O)
- Apply specific assessment criteria to specific iterations

Simulink - Test Manager

The screenshot shows the Simulink Test Manager interface. The 'TESTS' tab is active, displaying a list of tests on the left and a detailed view of the 'Modify Motor Demand Paused Stop' test on the right. The 'TEST HARNESS' is set to 'SPC_Scnd_MIL_MMCM'. The 'SIMULATION SETTINGS AND RELEASE OVERRIDES' section is expanded, showing various parameter overrides. The 'ITERATIONS' section is highlighted with a red box, containing a table of iterations.

NAME	DESCRIPTION	ASSESSMENTS	TEST SEQUENCE SCENARIO
SSN_HostNoStow	None	Stop to None, Start to Stop	SSN_HostNoStow
SSN_HostNoAvail	None	Stop to None, Start to Stop	SSN_HostNoAvail
SSN_PowLim	None	Stop to None, Start to Stop	SSN_PowLim
SSN_VoltLow	None	Stop to None, Start to Stop	SSN_VoltLow
SSS_HostNoStow	None	[Default] All enabled	SSS_HostNoStow
DSN_HostNoStow	None	Stop to None, Start to Stop	DSN_HostNoStow
DSN_HostNoAvail	None	Stop to None, Start to Stop	DSN_HostNoAvail
DSN_PowLim	None	Stop to None, Start to Stop	DSN_PowLim
DSN_VoltLow	None	Stop to None, Start to Stop	DSN_VoltLow

Test Sequence/Iterations

The screenshot shows the SPC_Scnd_MIL_MMCM Test Sequence Editor. The 'Scenario List' on the left shows a sequence of steps: 'Run', 'step_1', 'step_2', 'step_3', 'step_4', 'step_5', 'step_6', 'step_7', 'step_8', 'step_9', 'step_10', 'step_11', 'step_12', 'step_13', 'step_14', 'step_15', 'step_16', 'step_17', 'step_18', 'step_19', 'step_20', 'step_21', 'step_22', 'step_23', 'step_24', 'step_25', 'step_26', 'step_27', 'step_28', 'step_29', 'step_30', 'step_31', 'step_32', 'step_33', 'step_34', 'step_35', 'step_36', 'step_37', 'step_38', 'step_39', 'step_40', 'step_41', 'step_42', 'step_43', 'step_44', 'step_45', 'step_46', 'step_47', 'step_48', 'step_49', 'step_50', 'step_51', 'step_52', 'step_53', 'step_54', 'step_55', 'step_56', 'step_57', 'step_58', 'step_59', 'step_60', 'step_61', 'step_62', 'step_63', 'step_64', 'step_65', 'step_66', 'step_67', 'step_68', 'step_69', 'step_70', 'step_71', 'step_72', 'step_73', 'step_74', 'step_75', 'step_76', 'step_77', 'step_78', 'step_79', 'step_80', 'step_81', 'step_82', 'step_83', 'step_84', 'step_85', 'step_86', 'step_87', 'step_88', 'step_89', 'step_90', 'step_91', 'step_92', 'step_93', 'step_94', 'step_95', 'step_96', 'step_97', 'step_98', 'step_99', 'step_100'. The 'Transition' column shows the conditions for moving from one step to the next. The 'Next Step' column shows the next step in the sequence.

Simulation - Logical and Temporal Assessments

- Create triggered response assessments to be applied to multiple iterations
- Dynamic assessments criteria automates the "Assessment"
- Link Assessments to Model parameters (i.e. Model Cals, I/O)

Simulink - Test Manager

The screenshot shows the Simulink Test Manager interface. The 'TESTS' tab is active, displaying a list of tests on the left and a detailed view of the 'Modify Motor Demand Paused St...' test on the right. The 'LOGICAL AND TEMPORAL ASSESSMENTS' section is highlighted with a red box. It contains a table of assessments with columns for Name, Assessment, and Requirements. The 'Assessment' column shows logical expressions involving variables like `uint16(54)`, `uint16(27)`, and `uint16(40)`. The 'Requirements' column shows conditions like 'None' or 'true'. A 'Visual Representation' section on the right shows a timing diagram with 'TRIGGER' and 'RESPONSE' signals. A 'PROPERTY' table at the bottom left lists test properties like Name, Type, Model, and Location.

Logical and Temporal Assessments

The screenshot shows the 'Assessment Result' window. The 'Start to Stop' assessment is selected, showing its logical expression and a timing diagram. The 'Assessment' column shows a complex logical expression involving variables like `uint16(5)`, `uint16(27)`, and `uint16(40)`. The 'Requirements' column shows conditions like 'None' or 'true'. The 'Visual Representation' section shows a timing diagram with 'TRIGGER' and 'RESPONSE' signals. The 'PROPERTY' table at the bottom left lists test properties like Name, Type, Model, and Location.

Conclusion

- Identified a Traceability method from Software requirement to implemented DVMs/bugs
- Linking used to trace closure of bugs and impact analysis of updated requirements on DVMs
- Simulink Test Manager – Simulation Tests creates an adaptive environment to programmatically implement DVMs

Lessons Learned

- Full top-down decomposition of requirements creates a Higher lvl Software Implementation requirement that can be tested at a function lvl
- Resources/Program delays allowed for time allocation on a process focused, continuous improvement activity

Polarion

- Ongoing work with Siemens for full functionality of *Test Steps*
- Linking of HIL/Vehicle Ivl DVM and Test results
- Resolve Polarion -> JIRA link (currently uni-directional, not bi-directional)
- Compatibility/transition to JIRA Cloud

Simulink

- Investigate SIL implementation of *Logical and Temporal Assessments*
- Potentially re-run Simulink Test manager test suite on HIL using vector tools
- Collaborate with Ford VC Core/IT to automatically run tests when new model is pushed to GitHub

Thank You

Thank you for joining us today.

Please direct any follow-up questions to:

Eric Browning
Ebrowni7@ford.com

Joshua McCready
jmccrea8@ford.com