

MathWorks
**AUTOMOTIVE
CONFERENCE 2023**
Korea

Formal Requirements and Generating Requirements-Based Test Cases

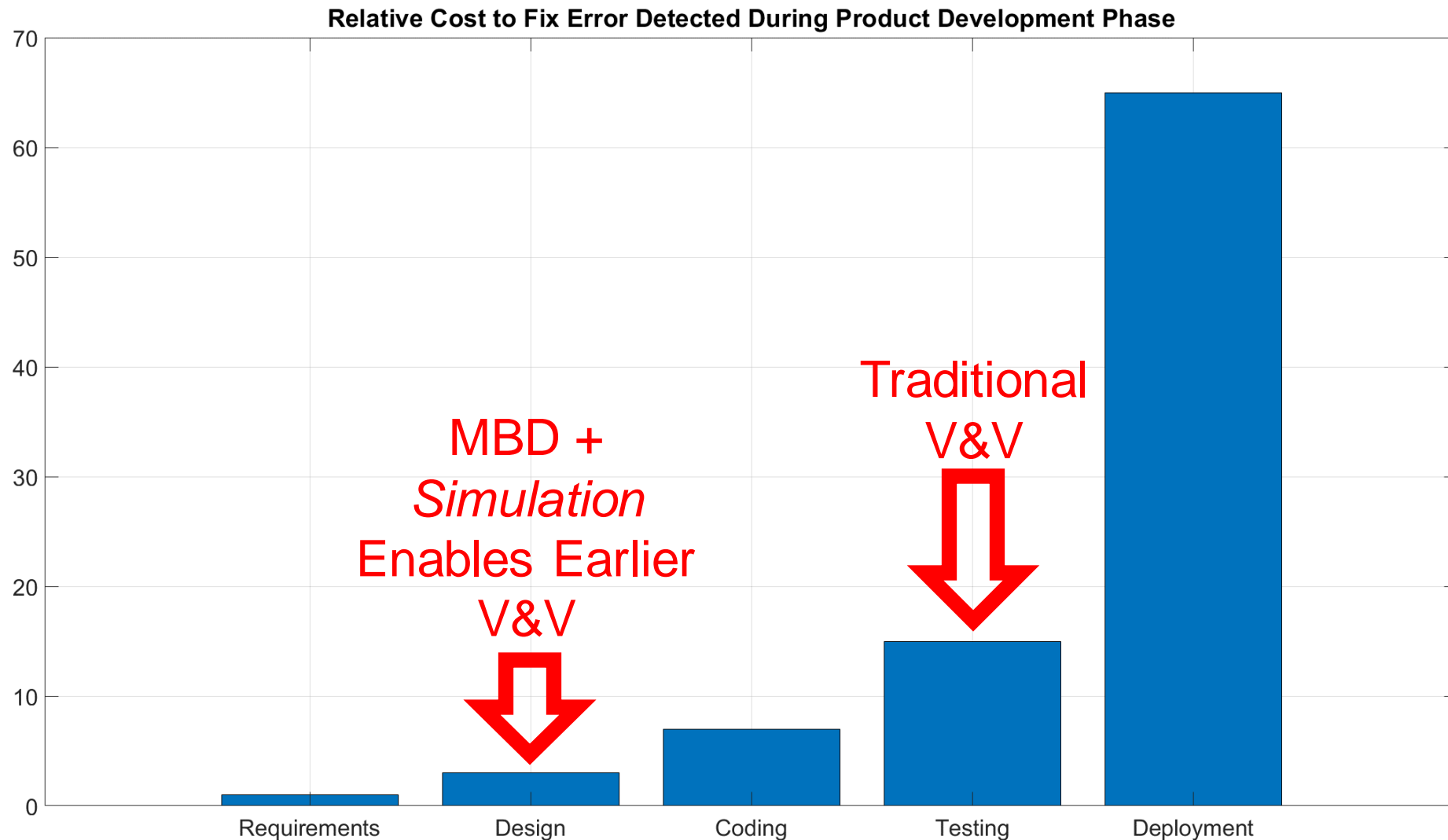
Tom Yoo, MathWorks



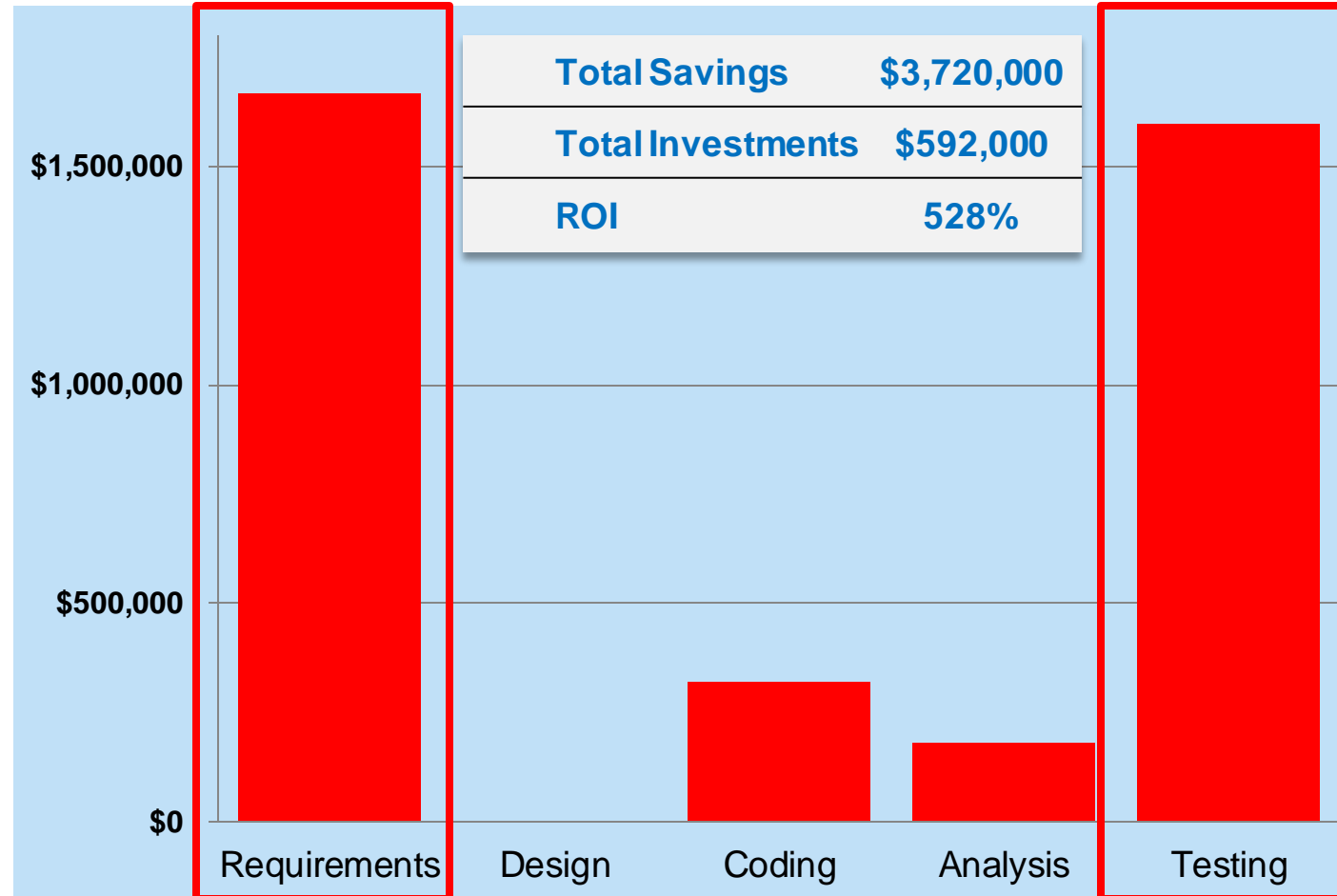
Agenda

- Why MBD in the Perspective of Creating Requirements
- Create Formal Requirements with Requirements Table
- Execute Requirements Based Test from Requirements Table
- Formal Verification with Requirements Table

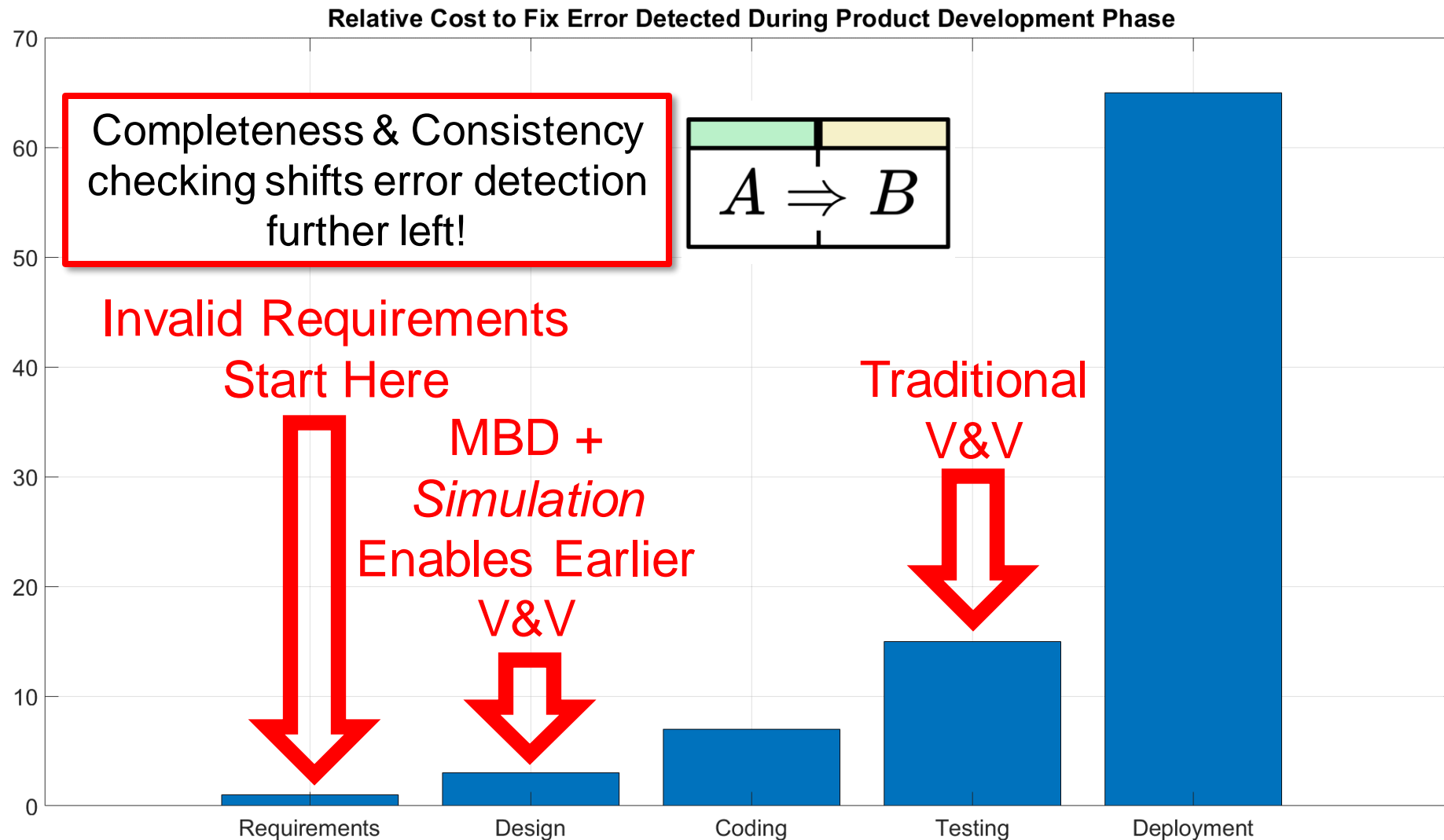
Model-Based Design: Detect Error Earlier to Minimize Costs



Model-Based Design ROI Calculation for Aerospace Applications



Can we Detect Errors Even Earlier?



Agenda

- Why MBD in the perspective of requirements
- **Create Formal Requirements with Requirements Table**
- Execute Requirement Based Test from the requirements
- Formal Verification with the Requirements

Formal Requirements

Activate Heat Pump

If the temperature difference exceeds 2 degrees for more than 2 seconds, then the pump shall activate for at least 2 seconds

When <condition 1> is true,
Then <condition 2> must be true for some time

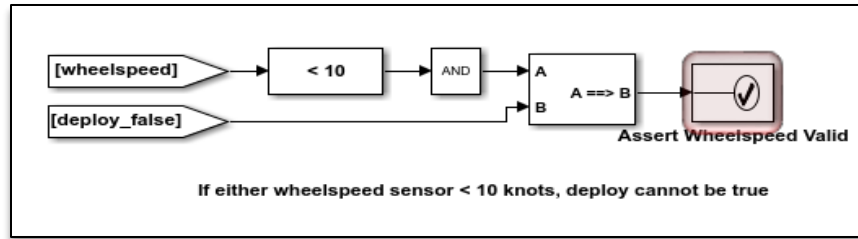
Simple concept

$$(|x_1 - x_2| \geq x_3)^{\varepsilon} \wedge \square_{[0, t_1]}(|x_1 - x_2| \geq x_3) \rightarrow \square_{[0, t_2]} x_4$$

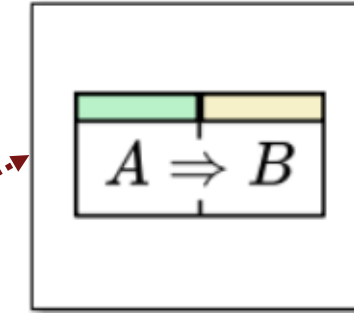
Hard to formalize

MTL logic

Formal Requirements Modeling “Styles”

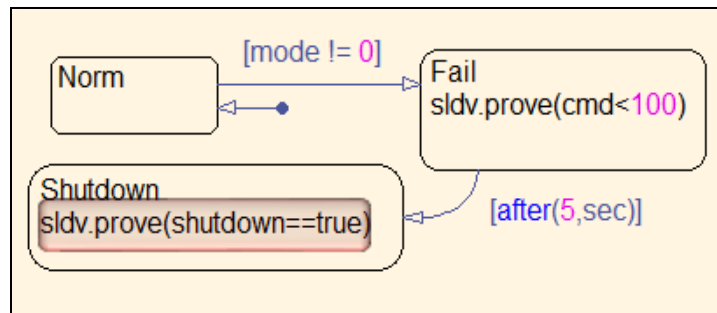


Simulink Blocks



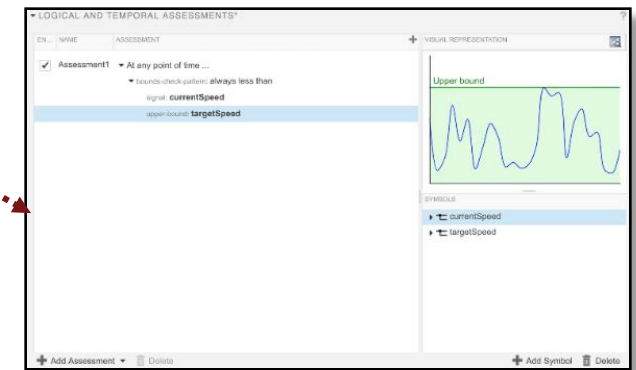
Requirements Table

R2022a



Stateflow

Requirements



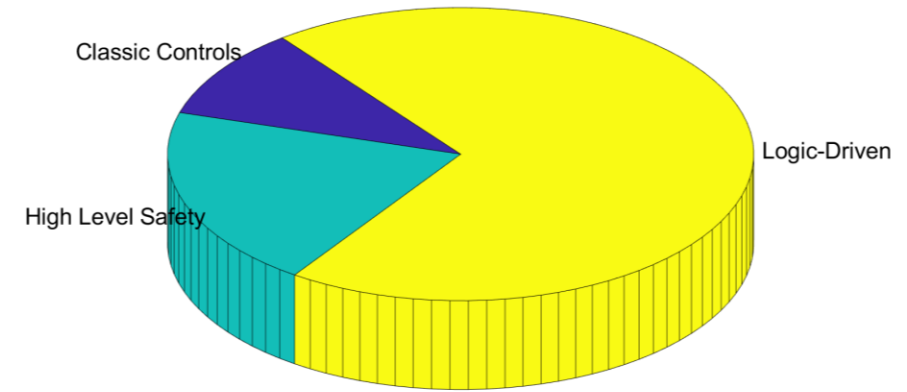
Temporal Assessments

```
function check_reminder(SeatBeltIcon,Inputs)
    % The seat belt light should be active whenever
    % the key is turned on and speed is less than 15
    % and the seatbelt is not fastened
    activeCond = ((Inputs.KEY ~= 0) && ...
                 (Inputs.SeatBeltFasten == 0) && ...
                 (Inputs.Speed < 15));
    sldv.prove(implies(activeCond, SeatBeltIcon));
```

MATLAB Function Block

A Typical Population of Requirements

- Many requirements are simple and “logic-driven”
- **Fault detection:** “The system shall fail the speed sensor when it exceeds X kph”
- **Signal selection:** “The system shall select the median value between the three temperature sensors.”
- **Mode logic:** “The system shall enter a degraded mode when no valid altitude sensors are available.”



Common Challenges with Requirements Validation

- An **individual** requirement is easy to manually validate
- A **set** of many requirements **is not easy to manually validate**
- **Completeness** and **Consistency** are the top challenges
 - **Completeness:** all required functionality is defined
 - **Consistency:** requirements do not conflict

How do Engineers ensure Consistency and Completeness?

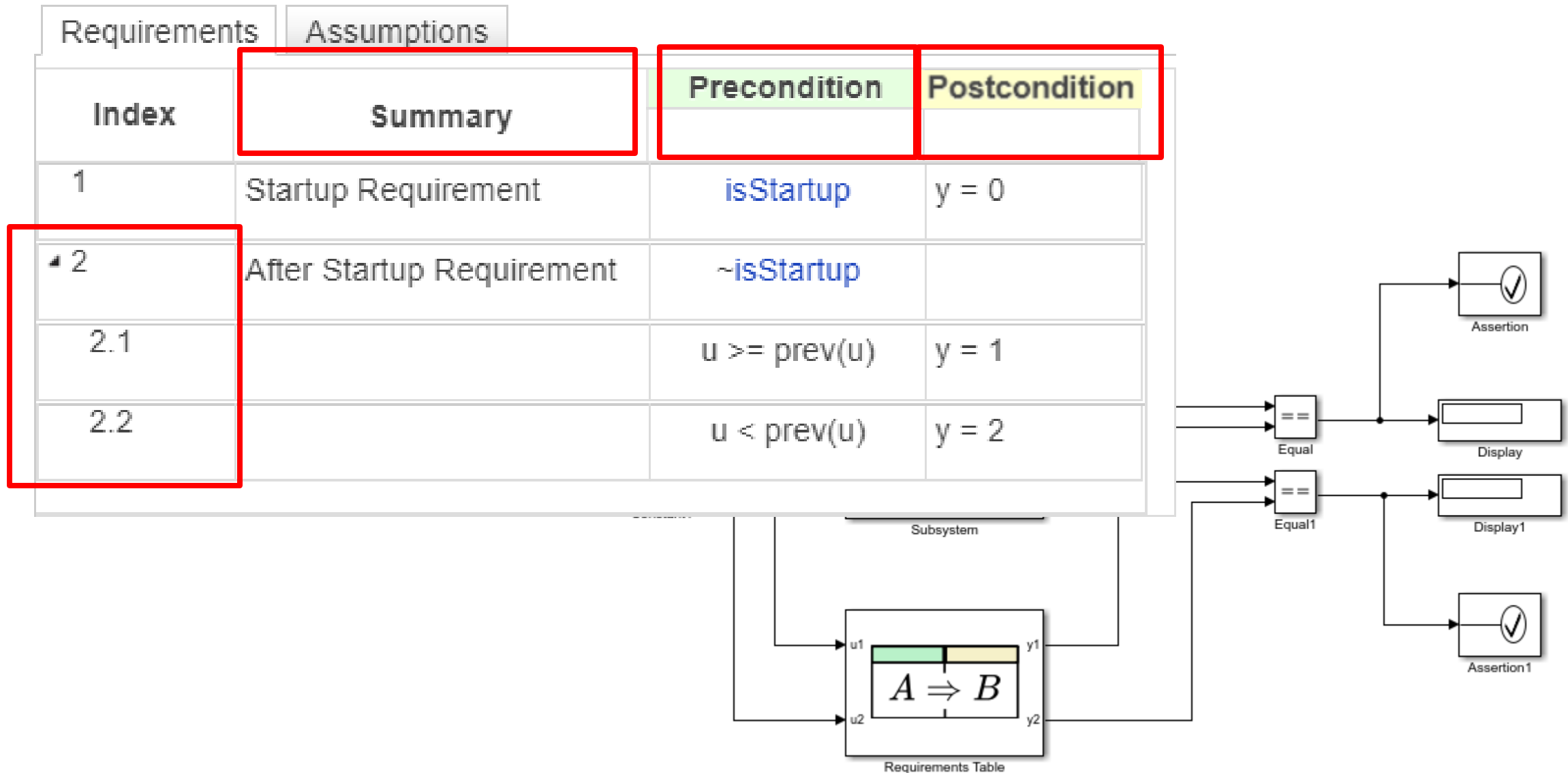
- Many organizations create “intermediate” text-based requirements
- These requirements often look like “pseudocode”:

*“The system shall enable **DRIVE_MODE_1** when **SWITCH_1** is pressed.”*

- Teams develop tools to parse requirements to check for issues
- This process is **expensive to create and maintain**

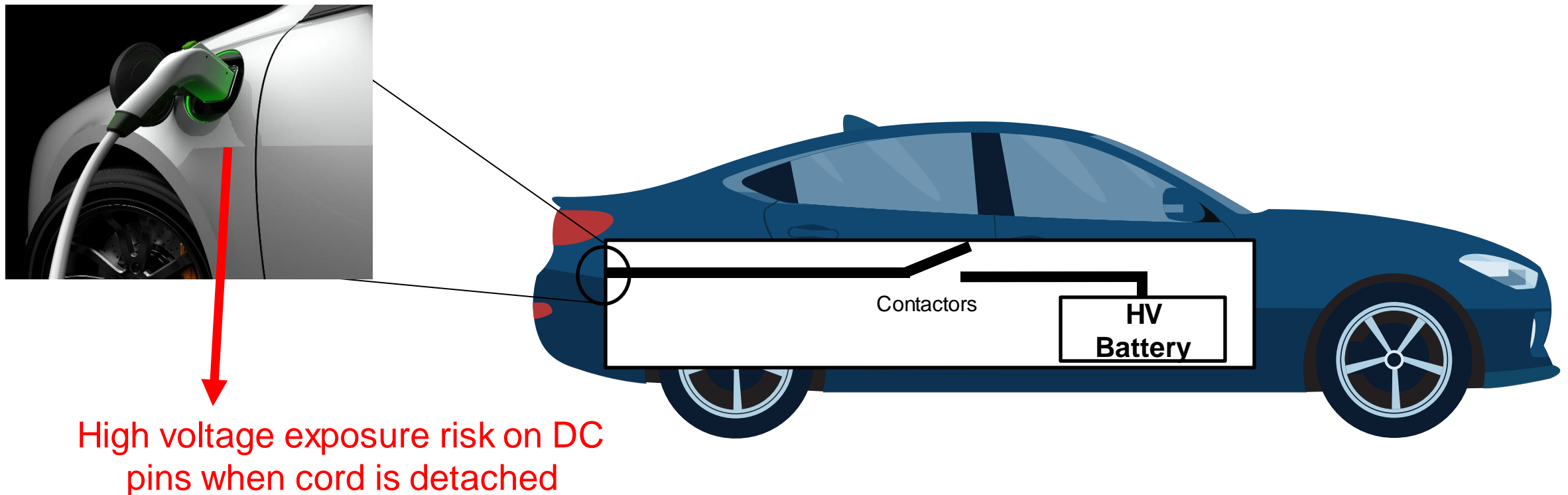
Text is not always the answer!

Requirements Table



SAE J1772 : DC Charging Cord Lock Requirements

- In DC charging, the electric vehicle battery is directly connected to the charging station.
- Due to safety concerns with voltage exposure at the port, DC charge cords are required to be locked during charging.



SAE J1772 Requirements for DC Charge Cord Locking

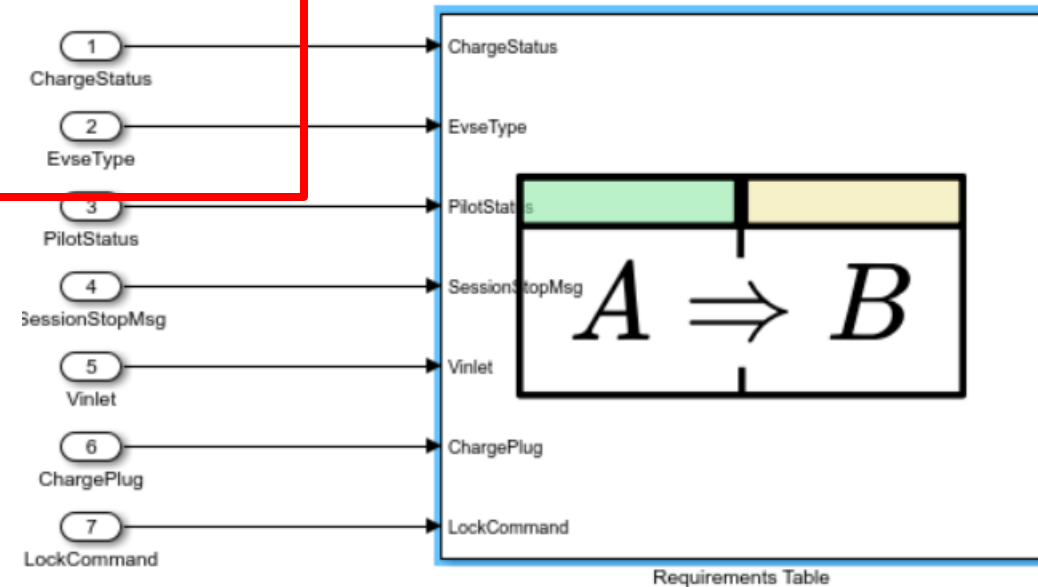
Requirement 1: The electric vehicle shall **lock** the charger in place if the **electric vehicle supply equipment** is **compatible**.

Requirement 2: The electric vehicle shall **unlock** the cord if the acknowledge **terminate charging session signal is received** and the vehicle measures an input voltage that is **less than 60 V** during **normal shutdown** procedures.

Requirement 3: The electric vehicle shall **unlock** the cord if the **pilot status** is **not ready** and the vehicle measures an input voltage that is **less than 60 V** during an **emergency shutdown**.

Outline of Requirements Table

- ChargeState : The charging status of the vehicle
- EvseType : Vehicle is compatible with the charging state
- PilotState : Pilot state of the charging state
- SessionStopResMsg : The termination of the charging session
- Vinlet : The voltage of the charging port
- ChargePlug : The charging state of the charger
- LockCommand : The status of the lock



Create Requirement Table

Requirements Table Assumptions Table

Index	Summary	Precondition	Postcondition LockCommand
1	Requirement 1: Lock when evse compatible	(EvseType == Compatible)	Locked
2	Requirement 2: Unlock during normal shutdown	(SessionStopMsg == Received) && (ChargeStatus == NrmlShutDown) && (Vinlet < 60)	Unlocked
3	Requirement 3: Unlock during emergency shutdown static pilot	(PilotStatus ~= C2) && (PilotStatus ~= D2) && (ChargeStatus == EmrgShutDown) && (Vinlet < 60)	Unlocked

Analyze Requirements Table

Index	Summary	Precondition	Action
1	Requirement 1: Lock when evse compatible	(EvseType == Compatible)	Locked
2	Requirement 2: Unlock during normal shutdown	(SessionStopMsg == Received) && (ChargeStatus == NmIshutDown) && (Vinlet < 60)	Unlocked
3	Requirement 3: Unlock during emergency shutdown static pilot	(PilotStatus ~= C2) && (PilotStatus ~= D2) && (ChargeStatus == EmrgShutDown) && (Vinlet < 60) && (EvseType == Compatible)	Unlocked

Block: CordLockReqTable_v1/Requirements Table

Inconsistency Issues

Inconsistency 1: 'LockCommand' is inconsistent at time 0 in requirements 1 and 3 for the following inputs:

Time	0
Step	1
ChargeStatus	ChrgStat.EmrgShutDown
EvseType	EvseStat.Compatible
PilotStatus	PilotStat.D1
SessionStopMsg	MsgStat.NotReceived
Vinlet	0

Incompleteness Issues

Incompleteness 1: 'LockCommand' is not specified at time 0.2 for the following inputs:

Time	0	0.2
Step	1	2
ChargeStatus	ChrgStat.ChargeStart	ChrgStat.NotC
EvseType	EvseStat.Compatible	EvseStat.NotC
PilotStatus	PilotStat.E_F	PilotStat.A
SessionStopMsg	MsgStat.NotReceived	MsgStat.NotR
Vinlet	87.6303	0

Complete Requirement Table

Requirements		Assumptions						
Index	Summary	Precondition						Action
		EvseType	ChargeStatus	SessionStopMsg	PilotStatus	ChargePlug	Vinlet	LockCommand
1	Requirement 1: Lock when evse compatible	Compatible	ChargeStart					Locked
2	Requirement 2: Unlock during non							
3	Requirement 3: Unlock during eme static pilot							
4	Requirement 4: Lock for unsafe vc							
5	Requirement 5: Unlock when unpl							
6	Requirement 6: Unlock during eme oscillating pilot							
7	Requirement 7: Unlock SessionSt							
8	Requirement 8: Unlock when not c							
9	Requirement 9: Unlock when com							

Assumptions		Postcondition
2	Assumption 2	EvseType == Incompatible ChargeStatus == NotCharging
3	Assumption 3	ChargePlug == NotPlugged ChargeStatus == NotCharging

Analysis Results

Date: 2022-04-21 11:25:46

Block: CordLockReqTable_v2/Requirements Table

Inconsistency Issues

No inconsistency issues found.

Incompleteness Issues

No incompleteness issues found.

Analyze Table with incomplete Requirements Table

The image displays the MATLAB R2023a software interface. The top menu bar includes options like HOME, PLOTS, and APPS. Below the menu is a toolbar with icons for file operations (New, Open, Save, etc.), variable management, code execution, and simulation. The main workspace is divided into three panes:

- Current Folder:** Shows a directory tree for the project located at `C:\Users\TomYoo\OneDrive - MathWorks\Documents\MATLAB\Examples\R2023a\slrequirements\AnalyzeFormalRequirementsEVSELockExample`. The files listed include:
 - test.mdatx
 - stateflowmodel_test.sbxc
 - stateflowmodel_test.sbx
 - stateflowmodel.slx
 - PlugStat.m
 - PilotStat.m
 - OnOff.m
 - mytest_property_out.sbxc
 - mytest_property_out.sbx
 - mytest_property.sbxc
 - mytest_property.sbx
 - mytest_property.slms
 - mytest.sbx
 - MsgStat.m
 - LockCmd.m
 - EvseStat.m
 - evse_example_requirements_3.png
 - evse_example_requirements_2.png
 - evse_example_requirements_1.png
 - evse_example_model.png
 - evse_example_assumptions.png
 - CordLockReqTable_v2.sbxc
 - CordLockReqTable_v2.slx
 - CordLockReqTable_v2_out.sbx
 - CordLockReqTable_v2_modi.sbxc
 - CordLockReqTable_v2_modi.sbx
 - CordLockReqTable_v2_Back.sbxc
 - CordLockReqTable_v2_Back.slx
 - CordLockReqTable_v1.slx** (highlighted)
 - CordLockReqTable_test.sbxc
 - CordLockReqTable_test.sbx
 - ChrgStat.m
 - AnalyzeFormalRequirementsEVSELockExample.mlx
 - analysis_results_1.png
 - slprj
 - slsv_output
 - modelslicer
- Command Window:** Shows the MATLAB prompt `>>`.
- Workspace:** An empty table with columns for Value and Name.

The status bar at the bottom indicates the active file is `CordLockReqTable_v1.slx (Simulink Model)`.

Analyze Table with complete Requirements Table

The image displays the MATLAB R2023a software interface. The top menu bar includes options like HOME, PLOTS, and APPS. Below the menu is a toolbar with various icons for file operations, workspace management, and simulation. The main workspace is divided into three panes:

- Current Folder:** Shows a directory tree with files such as `test.mdatx`, `stateflowmodel_test.slxc`, `PlugStat.m`, `mytest_property_out.slxc`, `mytest_property_out.slx`, `mytest_property.slxc`, `mytest_property.slx`, `mytest_property.slms`, `mytest.slx`, `MsgStat.m`, `LockCmd.m`, `EvseStat.m`, `evse_example_requirements_3.png`, `evse_example_requirements_2.png`, `evse_example_requirements_1.png`, `evse_example_model.png`, `evse_example_assumptions.png`, `CordLockReqTable_v2.slxc`, `CordLockReqTable_v2.slx`, `CordLockReqTable_v2_out.slx`, `CordLockReqTable_v2_modi.slxc`, `CordLockReqTable_v2_modi.slx`, `CordLockReqTable_v2_Back.slxc`, `CordLockReqTable_v2_Back.slx`, `CordLockReqTable_v1.slxc`, `CordLockReqTable_v1.slx`, `CordLockReqTable_test.slxc`, `CordLockReqTable_test.slx`, `ChrgStat.m`, `AnalyzeFormalRequirementsEVSELockExample.mlx`, `analysis_results_1.png`, `slprj`, `sldv_output`, and `modelslicer`.
- Command Window:** Shows the MATLAB prompt `>>`.
- Workspace:** Shows a table with columns for Value and Name.

The status bar at the bottom indicates the active file is `CordLockReqTable_v1.slx (Simulink Model)`.

Agenda

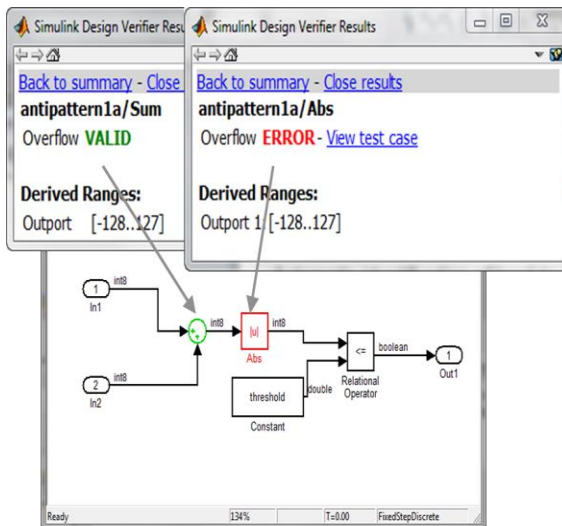
- Why MBD in the perspective of requirements
- Create Formal Requirements with Requirements Table
- **Execute Requirement Based Test from Requirements Table**
- Formal Verification with the Requirements

Simulink Design Verifier

Use formal methods to identify design errors

Design Error Detection

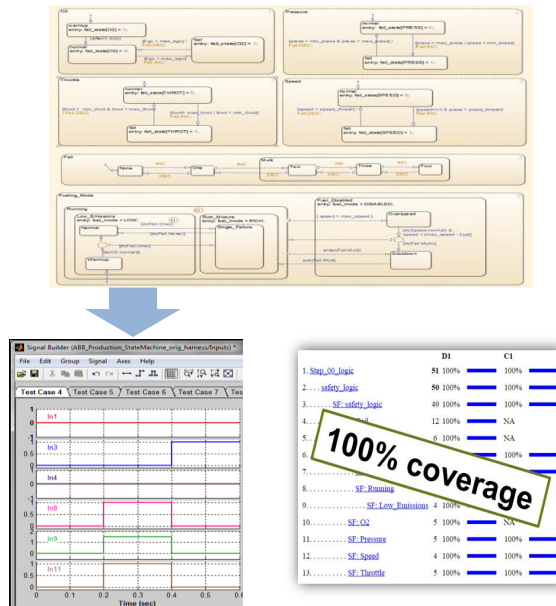
- **Uncover** hard to find dead logic and design flaws



Examples

Test Generation

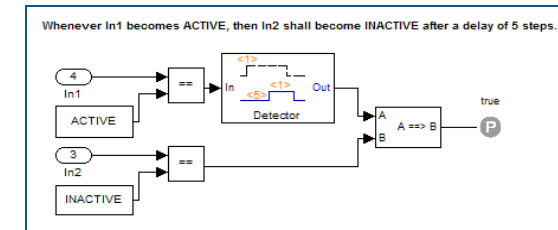
- **Automate** test vector generation to analyze missing coverage



Examples

Property Proving

- **Prove** formally design meets requirements



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

Results:

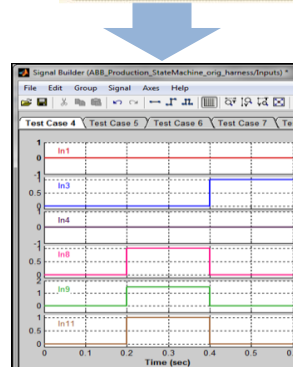
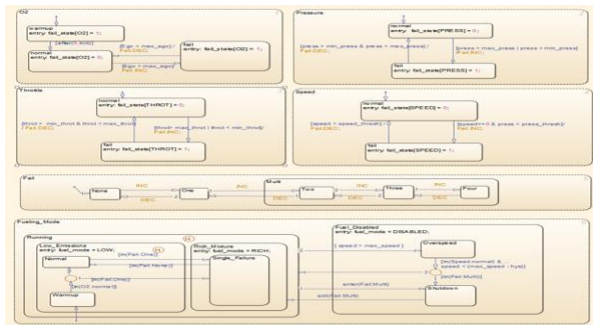


Examples

What is the Difference of Generating Testcase from Model vs Table?

Test Generation

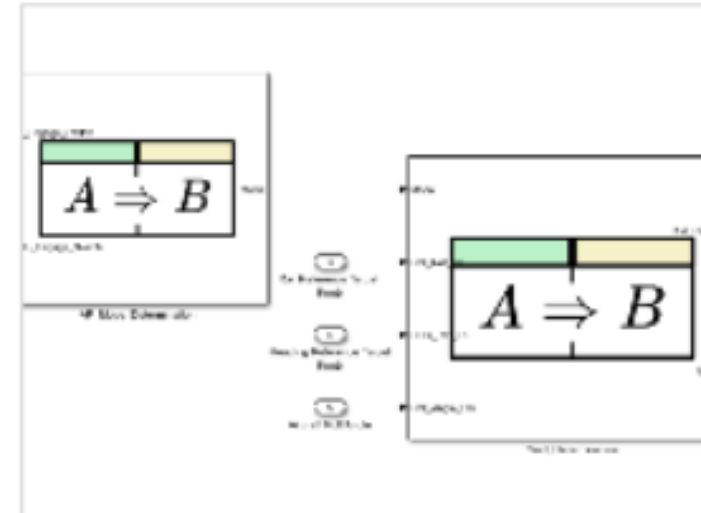
- Automate test vector generation to analyze missing coverage



	DI	CI
1. Step_00_logic	51 100%	100%
2. safety_logic	50 100%	100%
3. SF_safety_logic	49 100%	100%
4. ...	12 100%	NA
5. ...	6 100%	NA
6. ...	6 100%	100%
7. ...		
8. ... SF_Running		
9. ... SF_Low_Emissions	4 100%	
10. ... SF_O2	5 100%	NA
11. ... SF_Pressure	5 100%	100%
12. ... SF_Speed	4 100%	100%
13. ... SF_Throttle	5 100%	100%

100% coverage

VS



Use Specification Models for Requirements-Based Testing

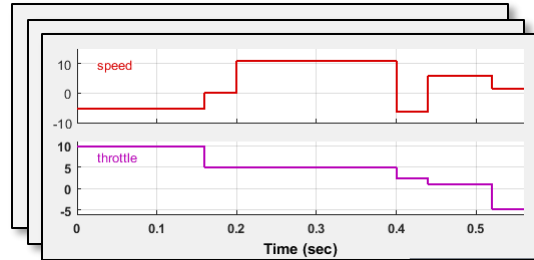
Follow a systematic approach to verify your design model against requirements.

[Open Live Script](#)

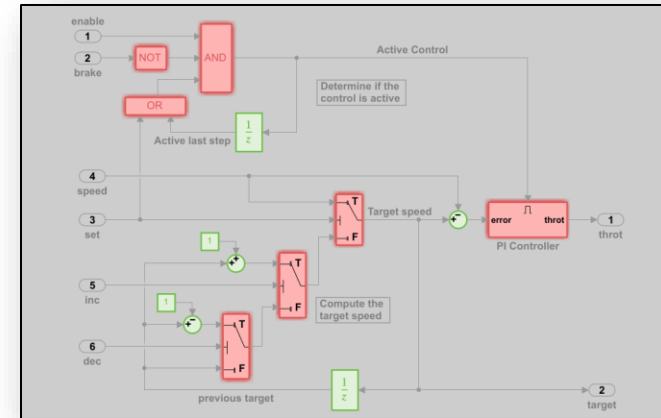
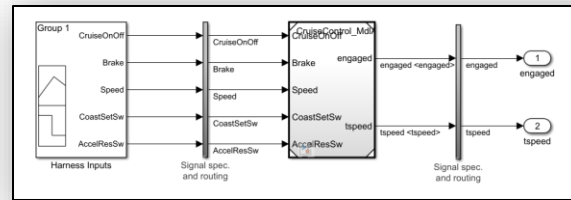
Requirements Based Test Workflow

ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl MdlAdv ReqLink		31	47%	31%	6%

Partial Coverage



Test Cases

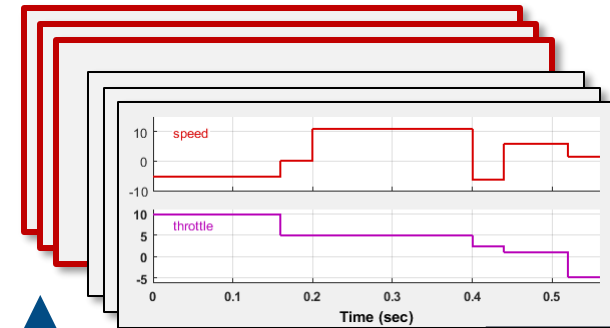


Requirements Based Test Workflow

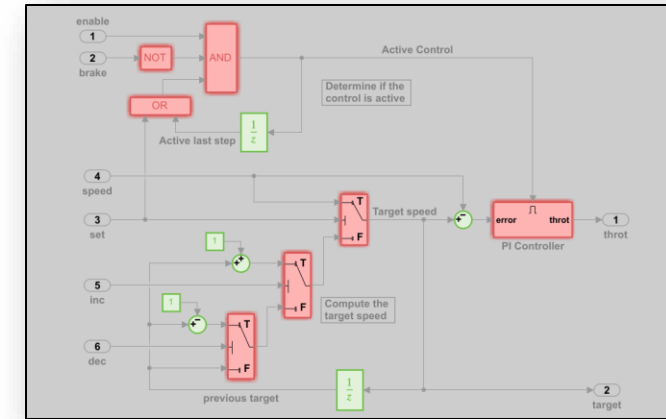
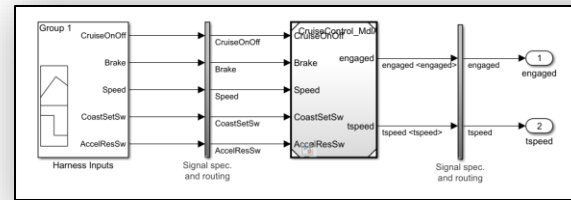
ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl MdlAdv ReqLink		31	47%	31%	6%

Partial Coverage

New Test Cases



Test Cases



Test Generator

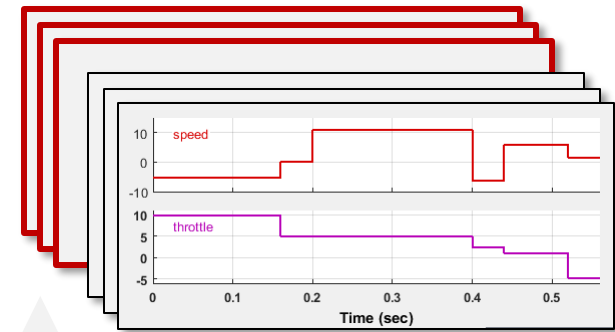


Requirements Based Test Workflow

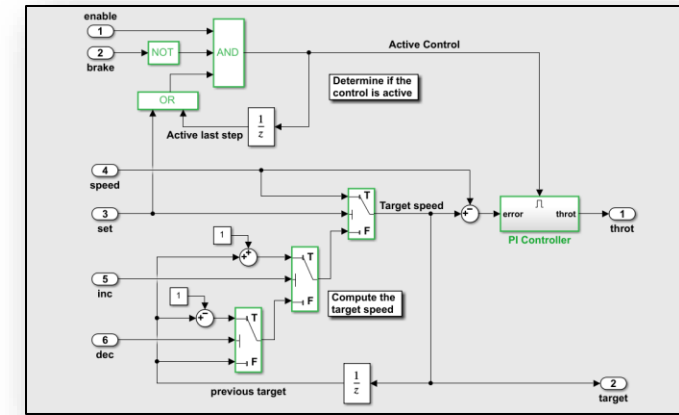
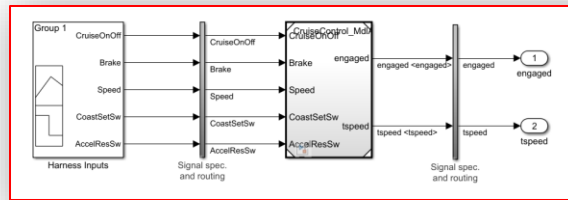
ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl MdlAdv ReqLink		31	100%	100%	100%

New Test Cases

Higher Coverage



Test Cases

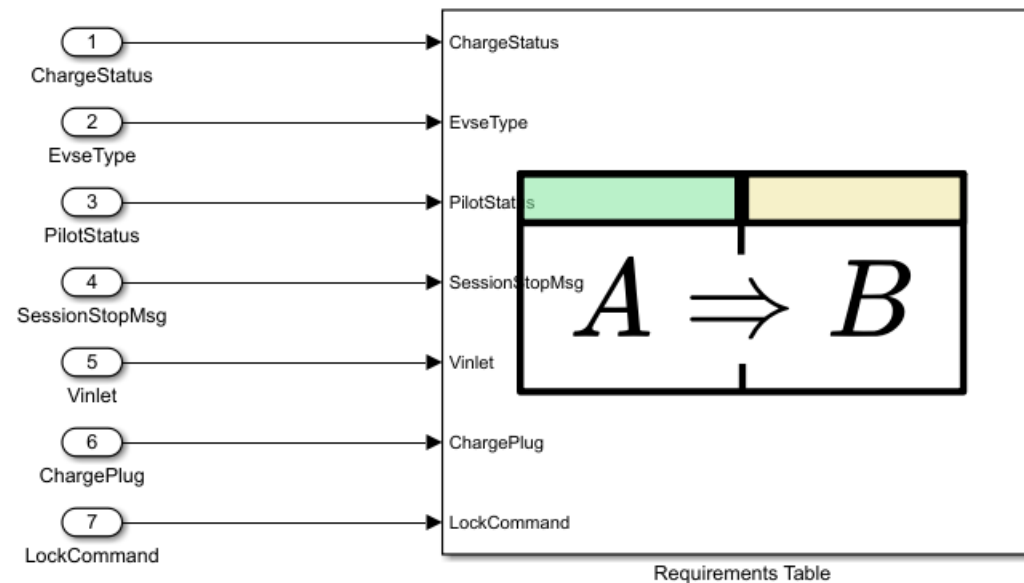


Test Generator

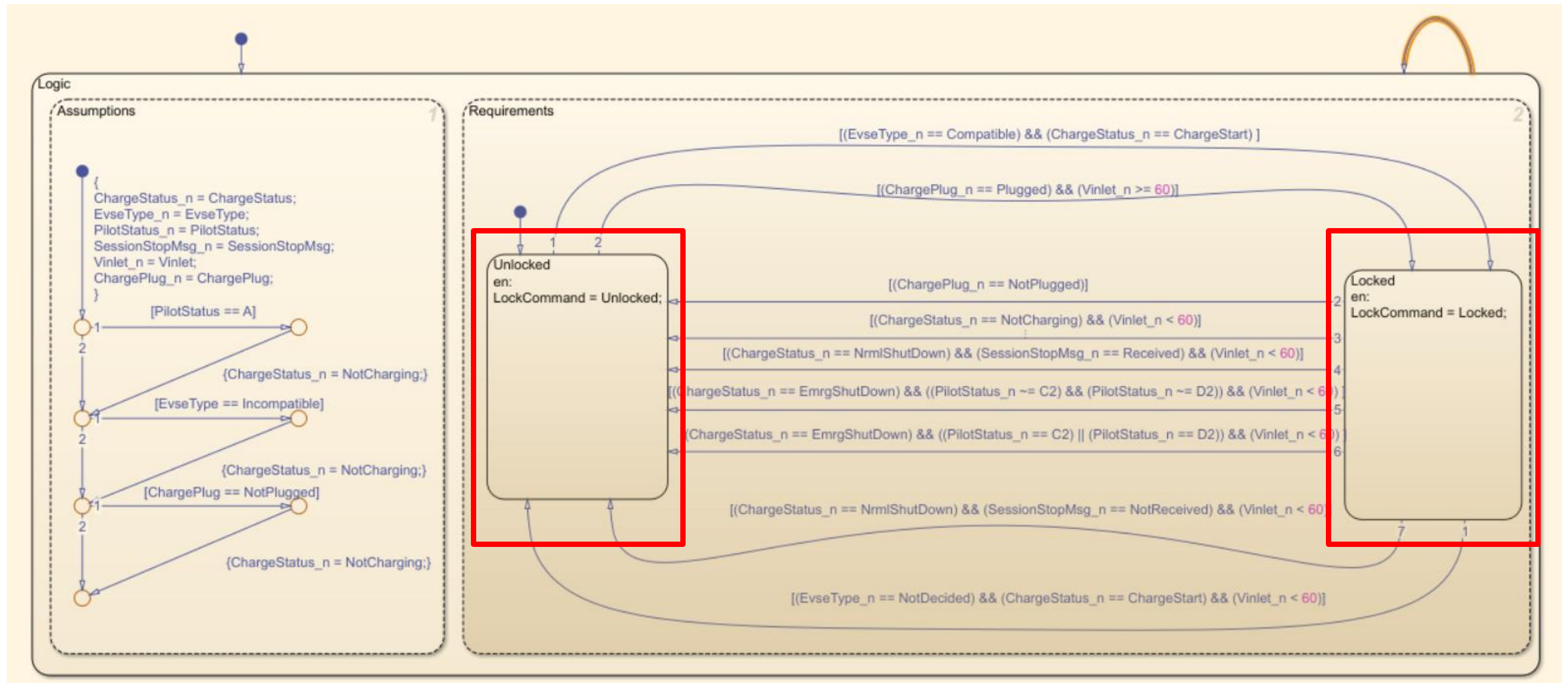


The Merits of generating Testcase from Requirements Table

Charging Cord Locking Mechanism Requirements, Version 2



Design Model based on Requirements Table



Automatic Linking Requirements Table Rows to Test Cases

R2023a

Requirements Editor

REQUIREMENTS

New Requirement Set | Open | Save | Import | Close | FILE

Load | Manage | Profile Editor | PROFILE

Add Requirement | Delete | Promote Requirement (disabled because sort is enabled) | Demote Requirement (disabled because sort is enabled) | REQUIREMENTS

Add Link | Delete | Clear Issue | Preferences | LINKS

Show Requirements | Show Links

Index	ID	Category	Summary	Type	ModifiedOn
▼ CordLockReqTable_v2_18					
▼ Table1	1		Requirements Table	Container	03-8월-20...
1	#52		Requirement 1: Lock when evse compatible	Functional	25-7월-20...
2	#53		Requirement 2: Unlock during normal shutdown	Functional	18-7월-20...
3	#54		Requirement 3: Unlock during emergency shutdown static pilot	Functional	18-7월-20...
4	#55		Requirement 4: Lock for unsafe voltage	Functional	18-7월-20...
5	#56		Requirement 5: Unlock when unplugged	Functional	18-7월-20...
6	#57		Requirement 6: Unlock during emergency shutdown oscillating pilot	Functional	18-7월-20...
7	#58		Requirement 7: Unlock SessionStop not recieved	Functional	18-7월-20...
8	#59		Requirement 8: Unlock when not charging	Functional	18-7월-20...
9	#60		Requirement 9: Unlock when compatibility not decided	Functional	18-7월-20...

Requirement: #55

► Properties

▼ Links

← Verified by:

- Test Case:1 ✓
- Test Case:9 ✓
- Test Case:10 ✓
- Test Case:11 ✓
- Test Case:12 ✓
- Test Case:13 ✓

► Comments

Requirements Based Test with Simulink Test

The screenshot displays the Test Manager application window. The interface is divided into several sections:

- TESTS** (Top): A toolbar with icons for New, Open, Save, Cut, Copy, Paste, Delete, Test Spec Report, Run, Run with Stepper, Stop, Parallel, Report, Visualize, Highlight in Model, Export, Import, Model Testing Dashboard, Preferences, and Help.
- Test Browser** (Left): A tree view showing test results. A red box highlights this section. It includes a filter field and a table of results.

NAME	STATUS
Results: 2023-Aug-09 13:47:04	6 7
New Test Case 1	6 7
Test Case:1	
Test Case:2	
Test Case:3	
Test Case:4	
Test Case:5	
Test Case:6	
Test Case:7	
Test Case:8	
Test Case:9	
Test Case:10	
Test Case:11	
Test Case:12	
Test Case:13	
- Summary** (Right): A table showing details for 'New Test Case 1'. A red box highlights the 'Cause of Failure' field.

Name	New Test Case 1
Outcome	6 7
Start Time	08/09/2023 13:47:06
End Time	08/09/2023 13:48:13
Type	Baseline Test
Test File Location	C:\Users\TomYoo\OneDrive - MathWorks\Documents\MATL...
Test Case Definition	
Rerun Test Case	
Tags	
Cause of Failure	Test failed as iteration failed
- AGGREGATED COVERAGE RESULTS** (Bottom Right): A table showing coverage for the analyzed model. A red box highlights this section.

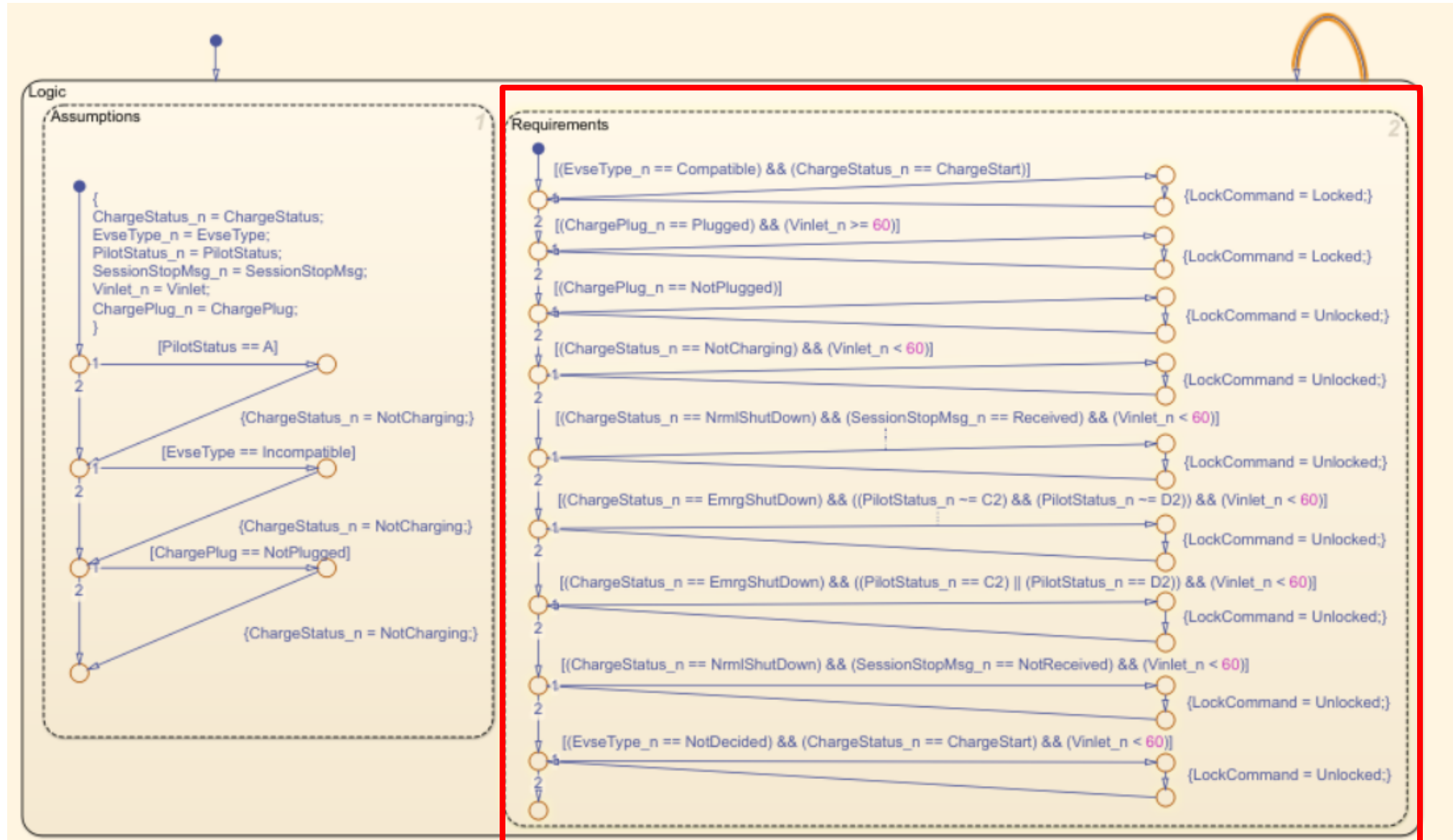
ANALYZED MODEL	REPORT	COM...	DECISION	CONDITION	MCDC
stateflowmodel_v2		30	27%	0%	0%
- PROPERTY** (Bottom Left): A table showing properties for 'New Test Case 1'.

PROPERTY	VALUE
Name	New Test Case 1
Status	6 7
Start Time	08/09/2023 13:47:06
End Time	08/09/2023 13:48:13
Type	Baseline Test
Test File Location	C:\Users\TomYoo\OneDrive...

Analyze Requirements Table

Requirements		Assumptions						
Index	Summary	Precondition					Postcondition	
		EvseType	ChargeStatus	SessionStopMsg	PilotStatus	ChargePlug	Vinlet	LockCommand
1	Requirement 1: Lock when evse compatible	Compatible	ChargeStart					Locked
2	Requirement 2: Unlock during normal shutdown		NrmlShutDown	Received			< 60	Unlocked
3	Requirement 3: Unlock during emergency shutdown static pilot		EmrgShutDown		(X ~= C2) && (X ~= D2)		< 60	Unlocked
4	Requirement 4: Lock for unsafe voltage					Plugged	>= 60	Locked
5	Requirement 5: Unlock when unplugged					NotPlugged		Unlocked
6	Requirement 6: Unlock during emergency shutdown oscillating pilot		EmrgShutDown		(X == C2) (X == D2)		< 60	Unlocked
7	Requirement 7: Unlock SessionStop not recieved		NrmlShutDown	NotReceived			< 60	Unlocked
8	Requirement 8: Unlock when not charging		NotCharging				< 60	Unlocked
9	Requirement 9: Unlock when compatibility not decided	NotDecided	ChargeStart				< 60	Unlocked

Modified Design Model



Generate Testcase from Requirements Table

Requirements Table: CordLockReqTable_v2/Requirements Table - Simulink

ANALYZE

CordLockReqTable_v2

Check Compatibility

Generate Tests

Load Earlier Results

REVIEW RESULTS

Requirements Table

CordLockReqTable_v2

Requirements Table

Requirements Assumptions

Index	Summary	Precondition					Postcondition	
		EvseType	ChargeStatus	SessionStopMsg	PilotStatus	ChargePlug	Vinlet	LockCommand
1	Requirement 1: Lock when evse compatible	Compatible	ChargeStart					Locked
2	Requirement 2: Unlock during normal shutdown		NrmlShutDown	Received			< 60	Unlocked
3	Requirement 3: Unlock during emergency shutdown static pilot		EmrgShutDown		(X ~= C2) && (X ~= D2)		< 60	Unlocked
4	Requirement 4: Lock for unsafe voltage					Plugged	>= 60	Locked
5	Requirement 5: Unlock when unplugged					NotPlugged		Unlocked
6	Requirement 6: Unlock during emergency shutdown oscillating pilot		EmrgShutDown		(X == C2) (X == D2)		< 60	Unlocked
7	Requirement 7: Unlock SessionStop not recieved		NrmlShutDown	NotReceived			< 60	Unlocked
8	Requirement 8: Unlock when not charging		NotCharging				< 60	Unlocked
9	Requirement 9: Unlock when compatibility not decided	NotDecided	ChargeStart				< 60	Unlocked

Model Browser

Referenced Files

Symbols

TYPE	NAME	VALUE	PORT
	ChargeStatus		1
	EvseType		2
	PilotStatus		3
	SessionStonMsn		4

Property Inspector

Requirements Table

Properties Info

Update method: Inherited

Saturate on integer overflow

Support variable-size arrays

Enable outputs in preconditions

Fixed-point properties

Ready View 2 warnings 104% FixedStepAuto

Requirements Based Test with Design Model

The screenshot displays the Simulink Test Manager interface. The main window shows a 'New Test Case 1' dialog box with the following configuration:

- Model:** CordLockReqTable_v2
- Harness:** CordLockReqTable_v2_sldvharness
- System Under Test:** CordLockReqTable_v2
- Test Harness:** CordLockReqTable_v2_sldvharness

The 'Test Browser' window is open, showing a list of requirements and test cases:

Index	Requirement
1	Requirement 1: Lock when evs
2	Requirement 2: Unlock during r
3	Requirement 3: Unlock during e static pilot
4	Requirement 4: Lock for unsafe
5	Requirement 5: Unlock when u
6	Requirement 6: Unlock during e oscillating pilot
7	Requirement 7: Unlock Session
8	Requirement 8: Unlock when n
9	Requirement 9: Unlock when c

The 'New Test Case 1' dialog also includes sections for 'PARAMETER OVERRIDES', 'CALLBACKS', 'INPUTS*', 'SIMULATION OUTPUTS', 'CONFIGURATION SETTINGS OVERRIDES', and 'BASELINE CRITERIA'. The 'BASELINE CRITERIA' section includes a table for defining baseline criteria:

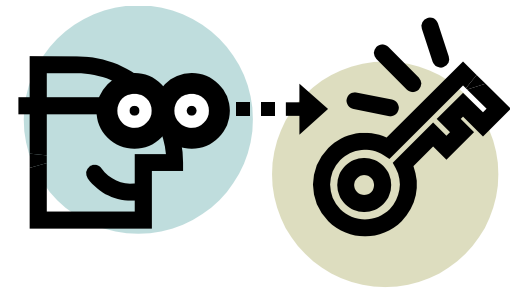
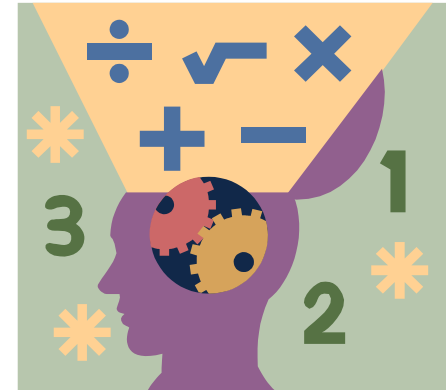
SIGNAL NAME	ABS TOL	REL TOL	LEADING TOL	LAGGING TOL
Click "Add" button to add an existing baseline file or click "Capture" to record a new baseline.				

Agenda

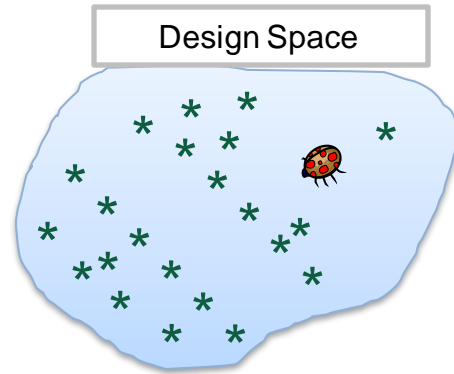
- Why MBD in the perspective of requirements
- Create Formal Requirements with Requirements Table
- Execute Requirement Based Test from the requirements
- **Formal Verification with Requirements Table**

Formal Verification

- Set of math-based techniques for specification, development and verification of algorithm design
- Works with models of system behavior instead of concrete data values
- Provides deeper understanding of design

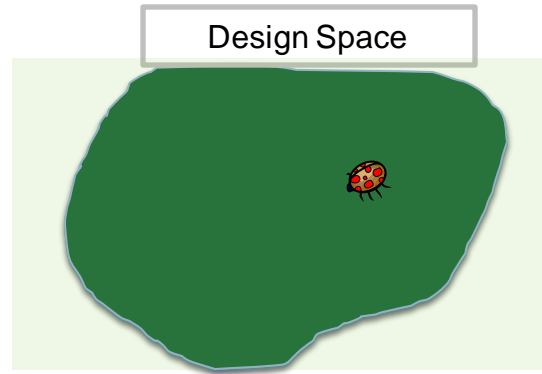


Simulation-Based Testing vs. Formal Verification



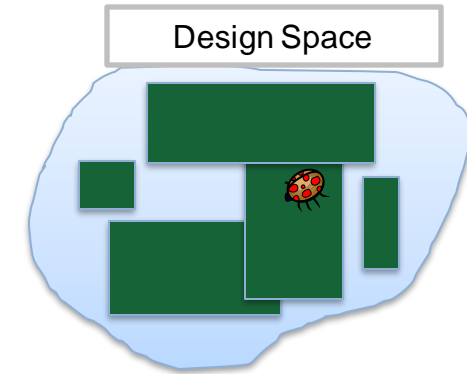
TESTING

Point coverage
through simulation



FORMAL VERIFICATION (Ideal case)

Complete coverage of
design space (formal proof)



FORMAL VERIFICATION* (Real-world)

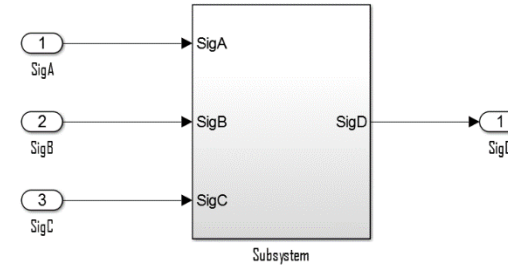
Real-world application
of formal verification

* Source: Erik Seligman, "Formal Verification – An Overview"

Property Proving

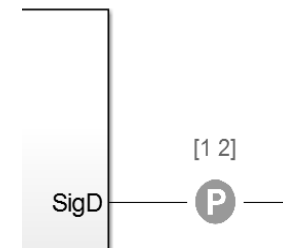
Prove design properties using formal requirement models

- Model functional and safety requirements
- Generates counter example for analysis and debugging

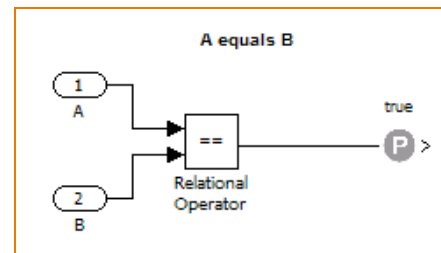


Design model

+



Specified properties

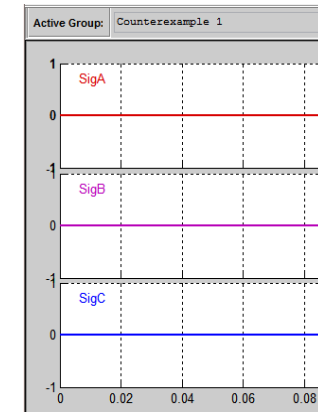


[1 2]

- P -

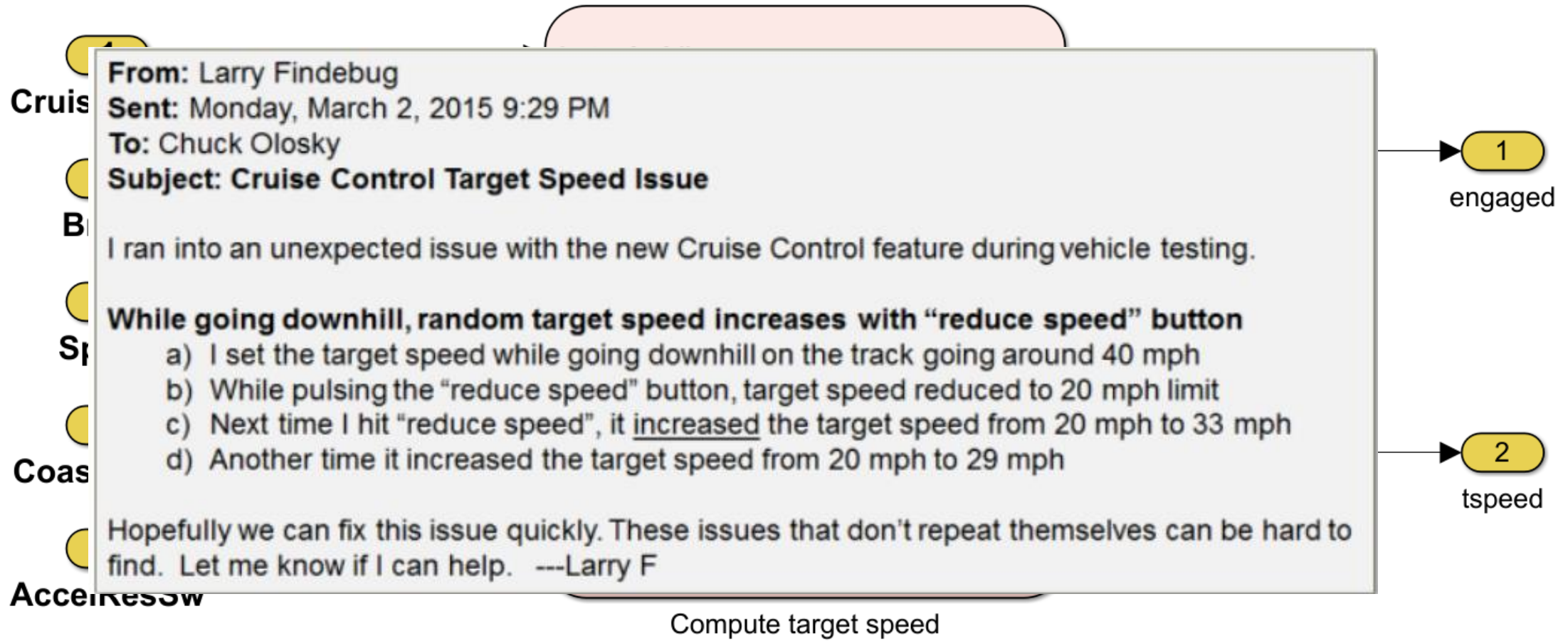
Proof

OR

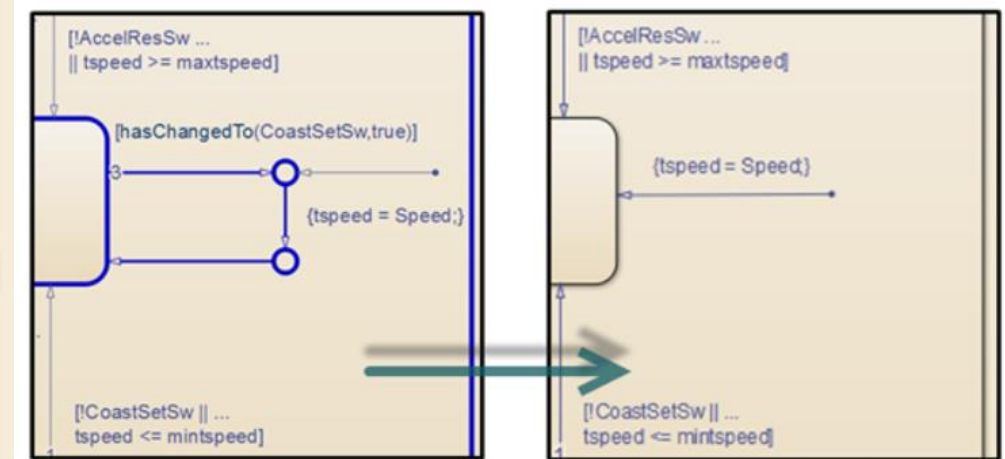
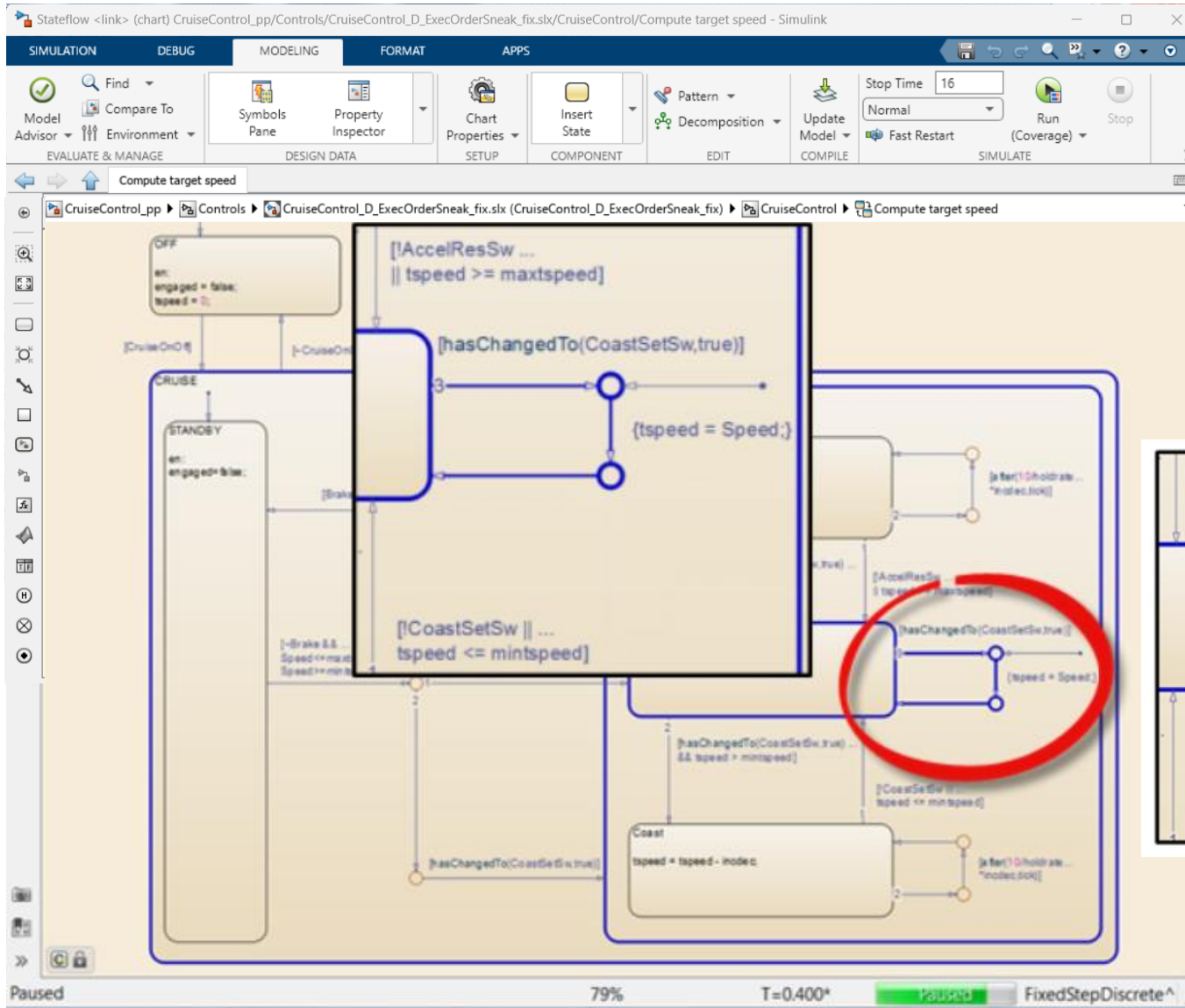


Counterexample

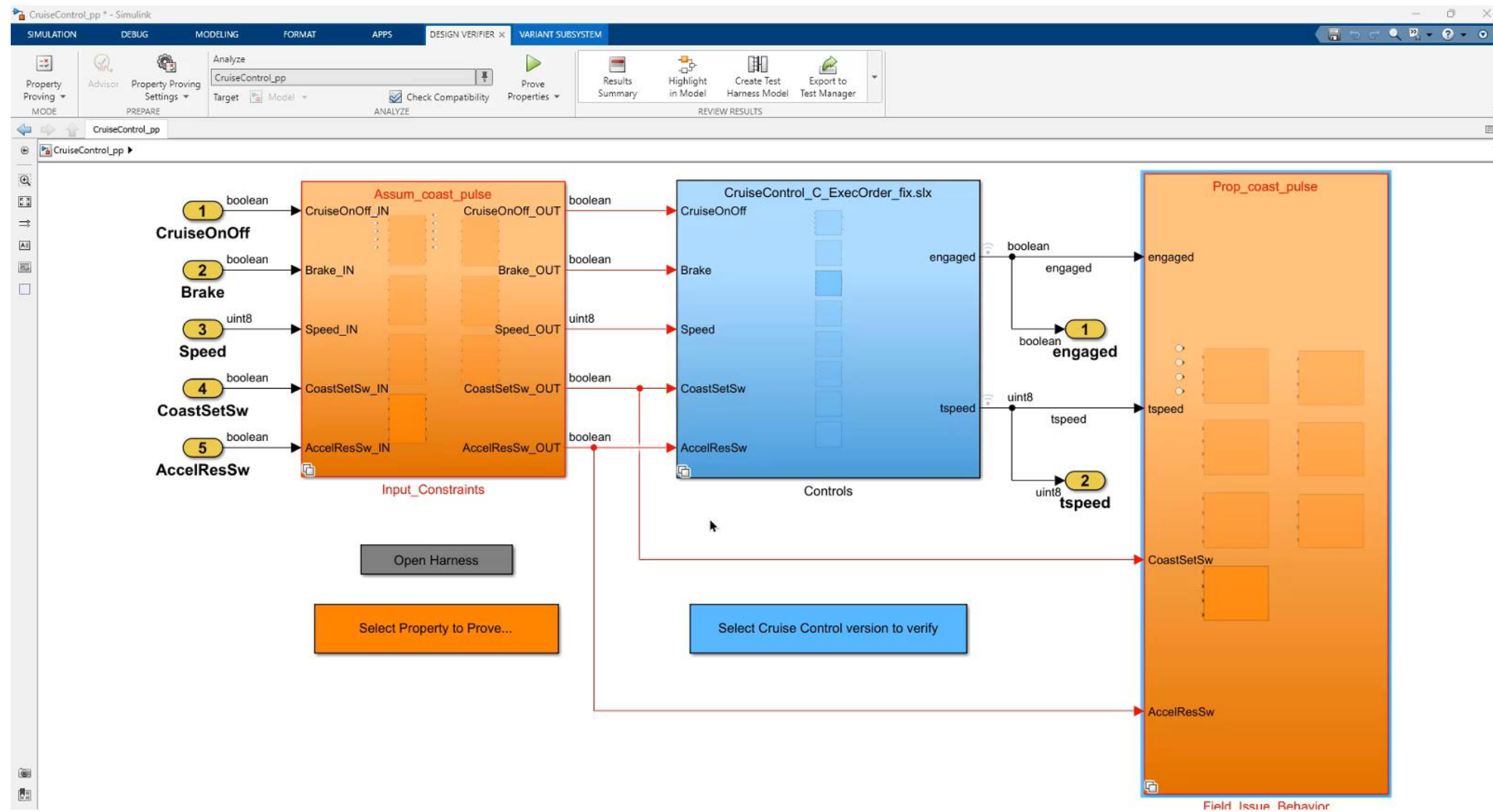
Why is Property proving needed?



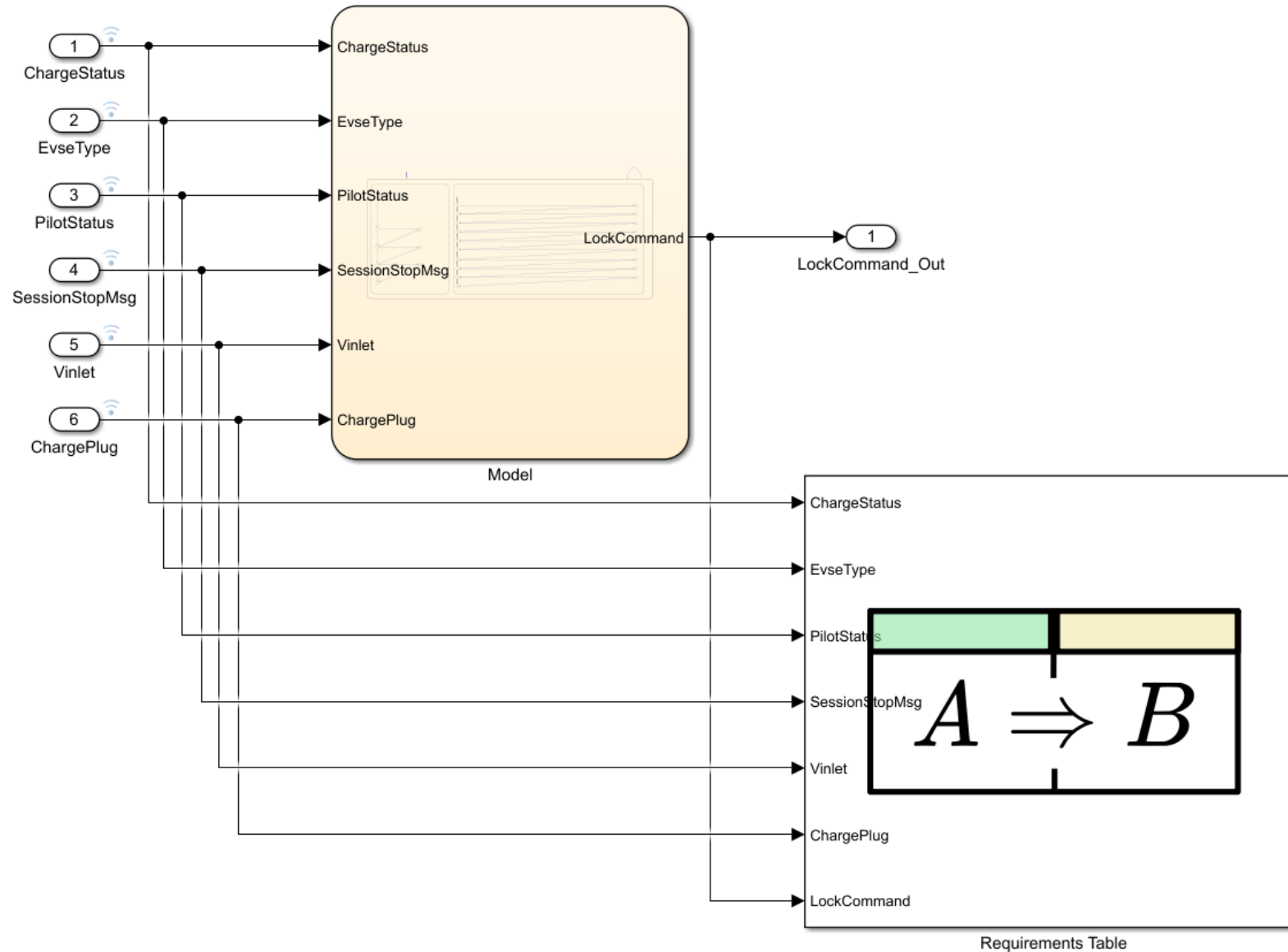
Drawback of Simulation Based Testing



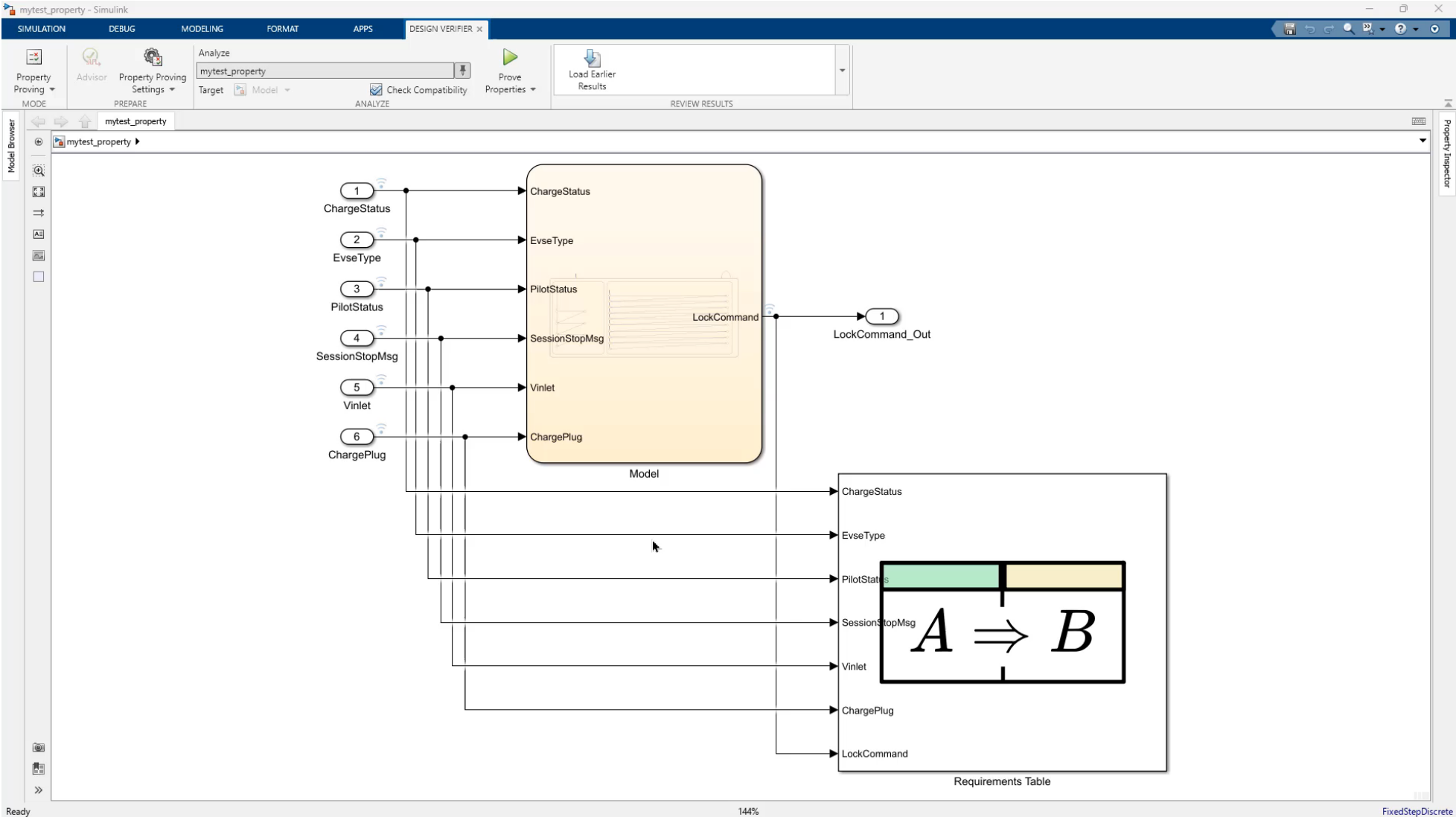
The Process of Traditional Property Proving



Property Proving with Requirements Table

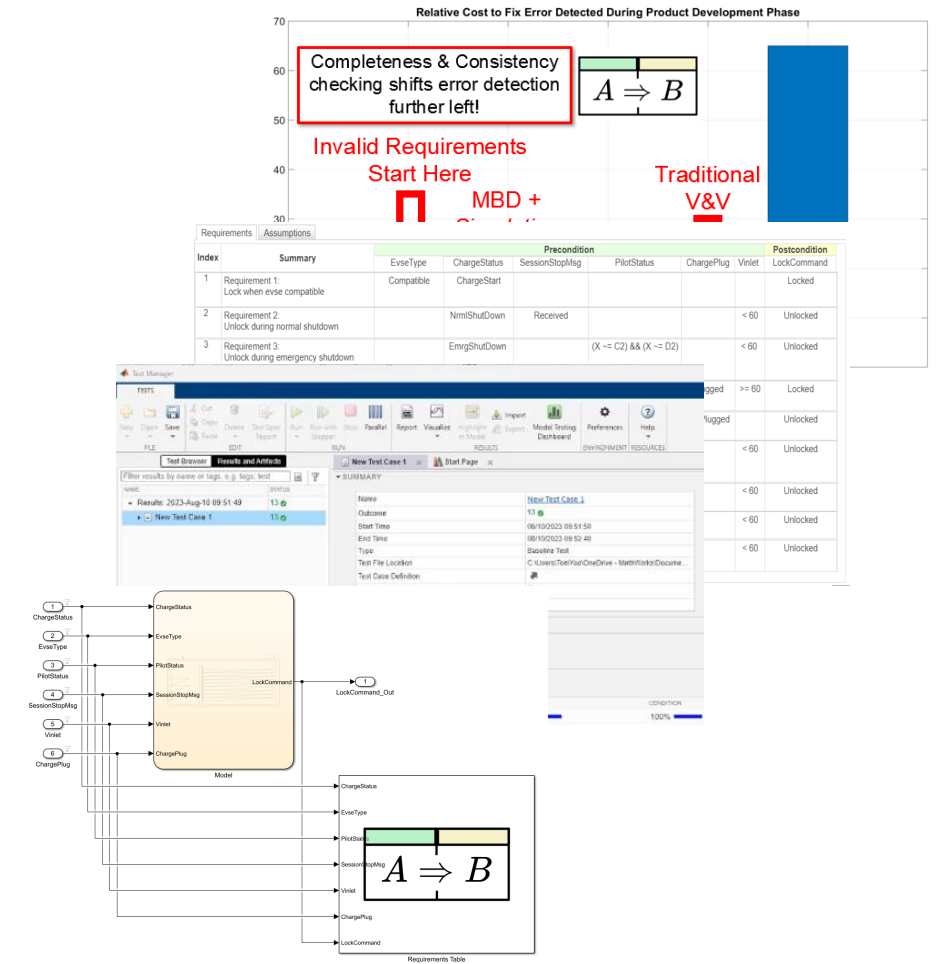


Recording of Property Proving



Key Takeaways

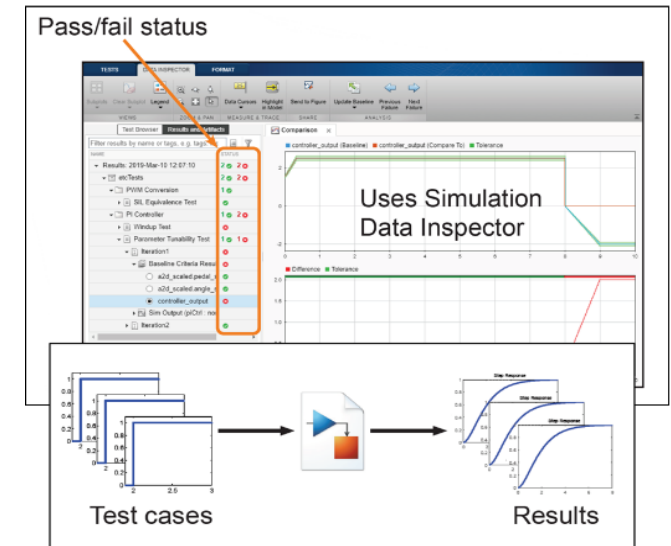
- MBD enables early verification
 - MBD can move starting point of verification from Testing to Design
 - Requirements can be verified by Simulation
- The Features of Requirements Table
 - Requirements Table can check completeness and coherence of Requirements
 - Requirements Table can generate testcases and expected values for Requirements Based Test
 - Requirements Table can execute Property proving easier



임베디드 소프트웨어 개발을 위한 시뮬링크 모델 검증

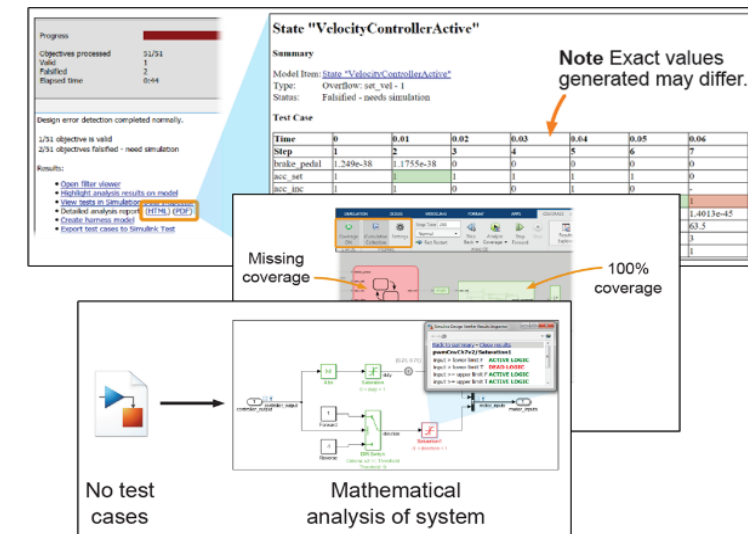
Simulation-Based Testing with Simulink (1 day)

- Learn techniques for testing Simulink model behavior against system requirements.
 - Identify the role of verification and validation in Model-Based Design
 - Create test cases
 - Analyze simulation results
 - Automate testing activities
 - Produce testing reports



Design Verification with Simulink (1 day)

- Learn how to use Simulink Design Verifier to ensure that a design is devoid of possible design errors, is fully tested, and satisfies necessary requirements.
 - Detecting and debugging common design errors
 - Collecting model coverage
 - completing missing coverage using automatic test generation
 - Proving model properties for requirement-based verification



MathWorks
**AUTOMOTIVE
CONFERENCE 2023**
Korea

Thank you

