MathWorks
AUTOMOTIVE CONFERENCE 2023
Korea

Formal Requirements and Generating Requirements-Based Test Cases

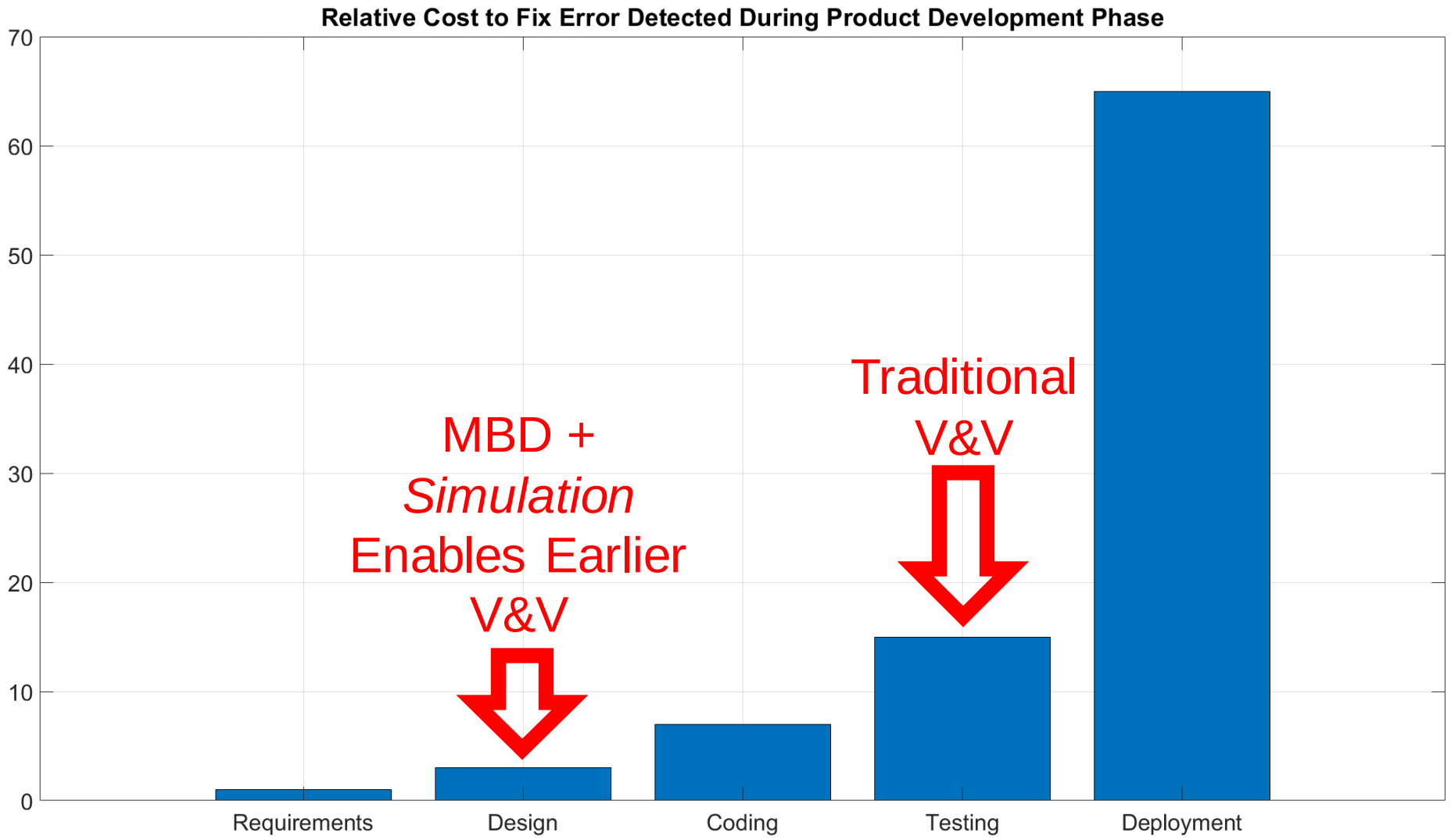
Tom Yoo, MathWorks

MathWorks

Agenda

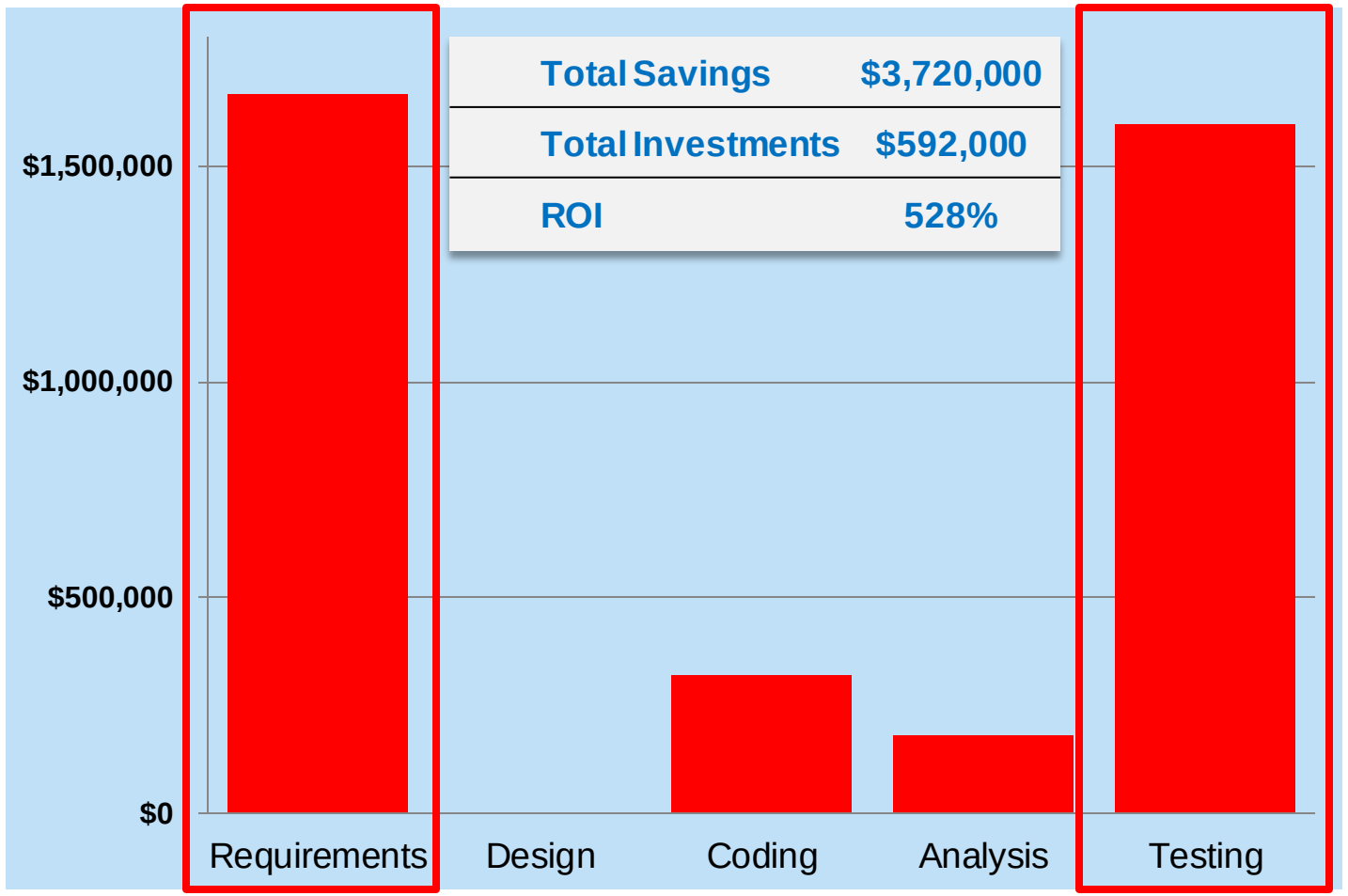
- Why MBD in the Perspective of Creating Requirements
- Create Formal Requirements with Requirements Table
- Execute Requirements Based Test from Requirements Table
- Formal Verification with Requirements Table

Model-Based Design: Detect Error Earlier to Minimize Costs

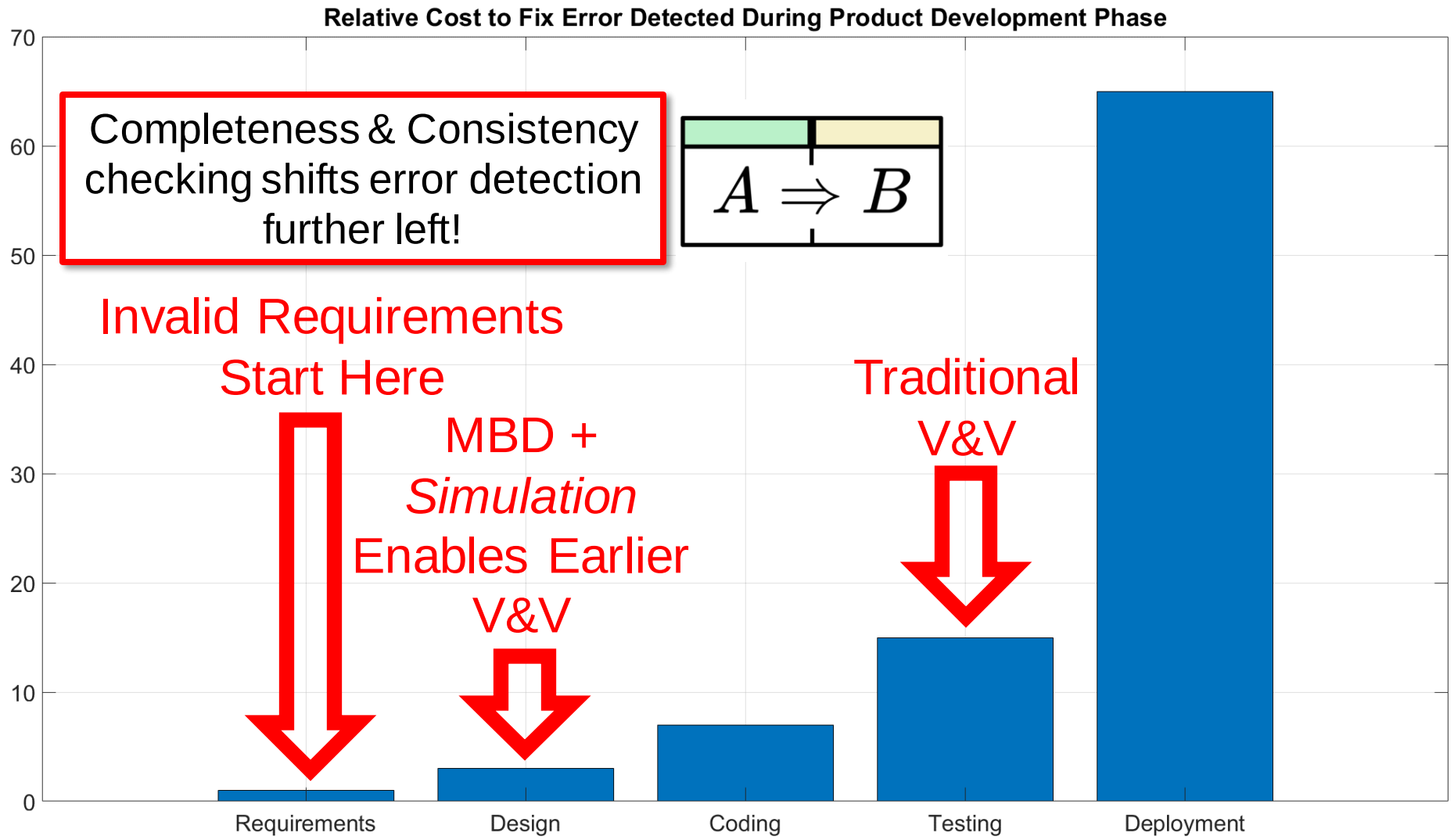


Data gathered by Hewlett Packard referred by XB in 2017
https://xbsoftware.com/blog/why-should-testing-start-early-software-project-development/

Model-Based Design ROI Calculation for Aerospace Applications



Can we Detect Errors Even Earlier?



Data gathered by Hewlett Packard referred by XB in 2017
https://xbsoftware.com/blog/why-should-testing-start-early-software-project-development/

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**

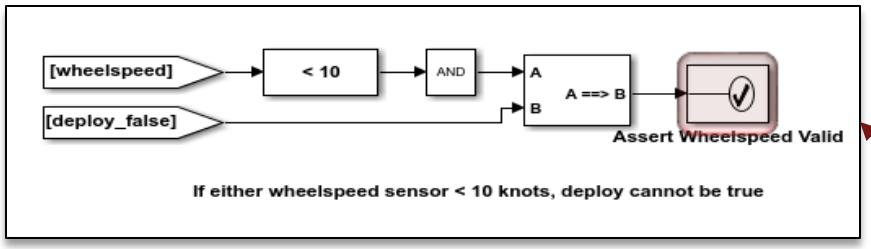
Simple concept

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

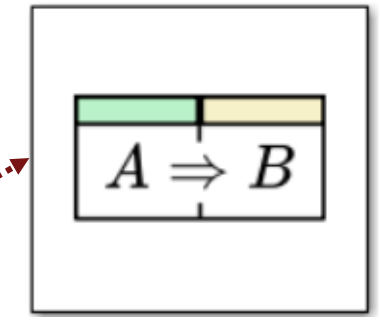
Hard to formalize

MTL logic

Formal Requirements Modeling "Styles"



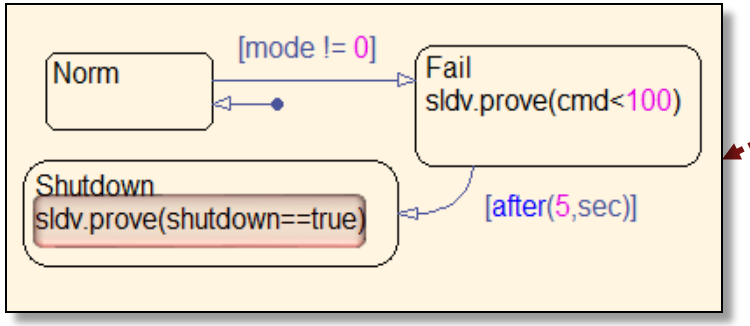
Simulink Blocks



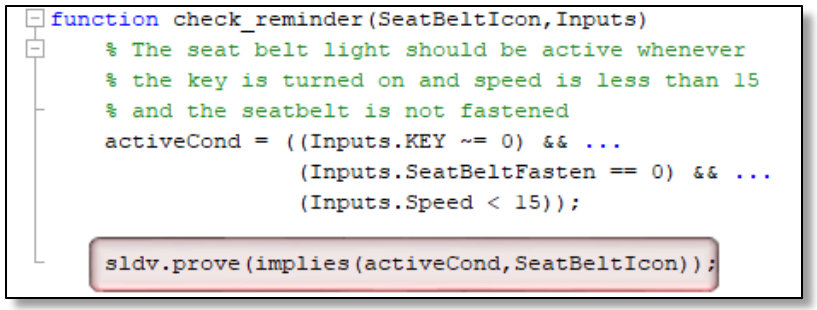
Requirements Table

R2022a

Requirements



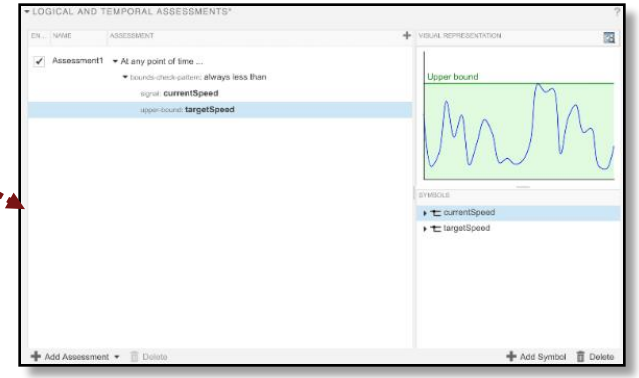
Stateflow



```
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));
```

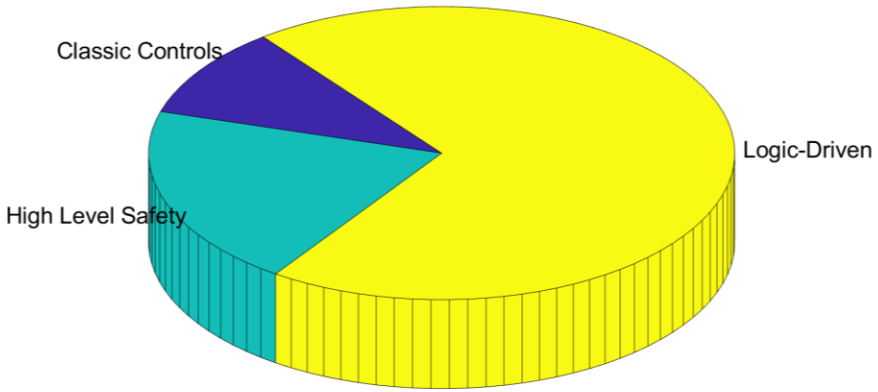
MATLAB Function Block



Temporal Assessments

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**

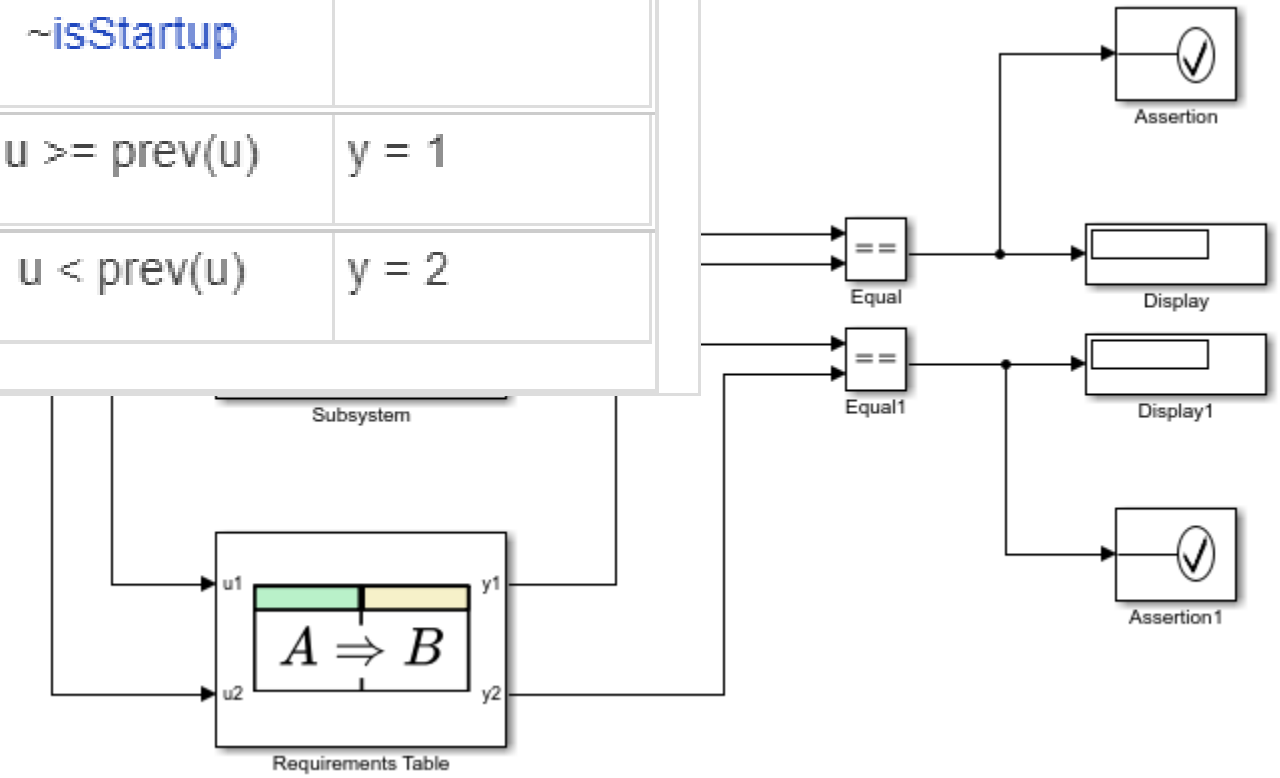
Text is not always the answer!

R2022a

Requirements Table

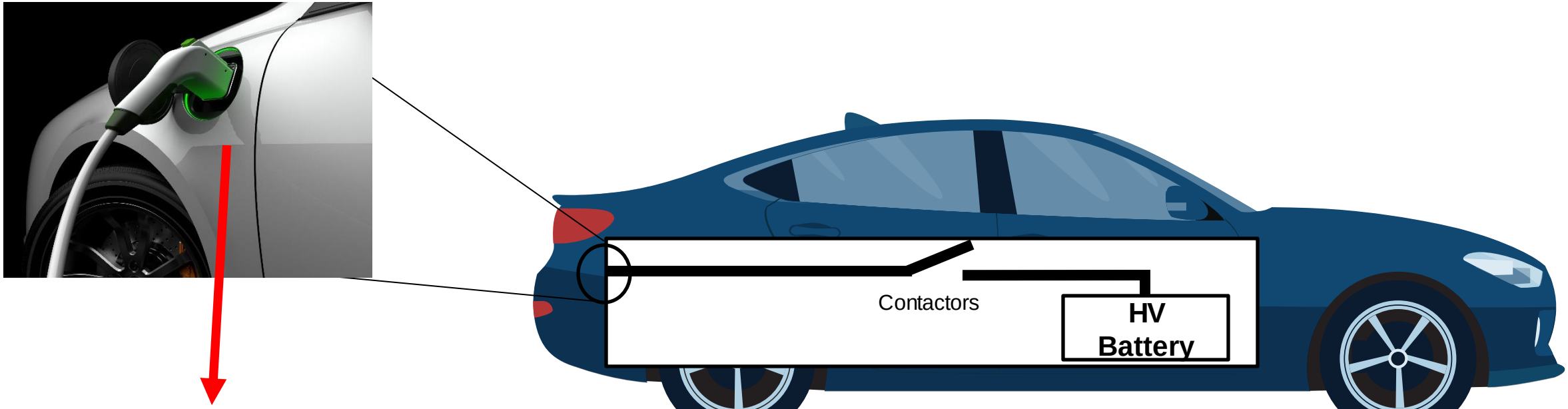
Requirements | Assumptions

Index	Summary	Precondition	Postcondition
1	Startup Requirement	isStartup	y = 0
2	After Startup Requirement	~isStartup	
2.1		u >= prev(u)	y = 1
2.2		u < prev(u)	y = 2



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

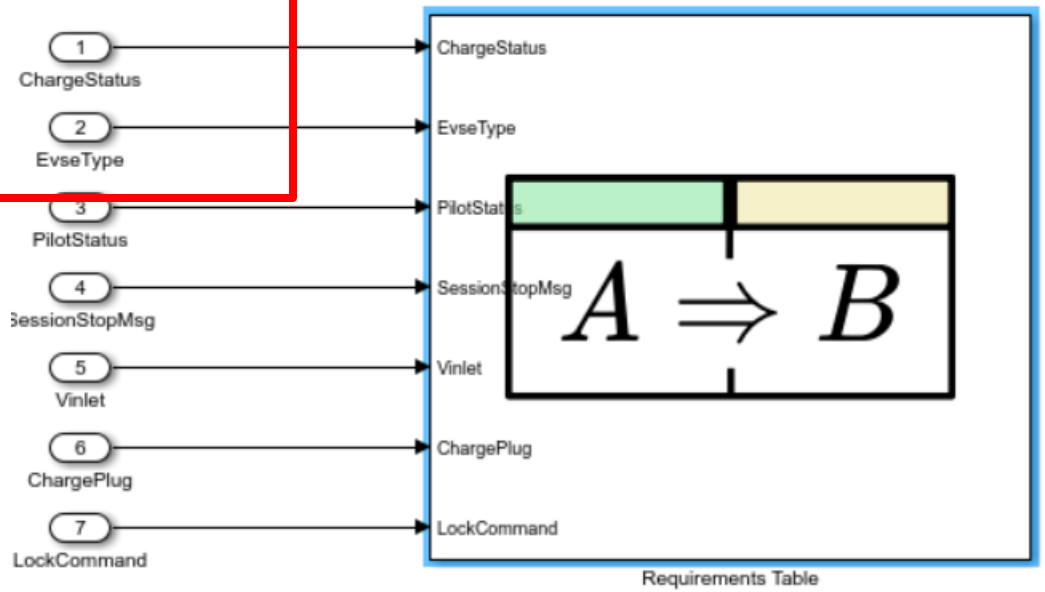
Requirement 1: The electric vehicle shall lock the charger in place if the electric vehicle supply equipment is compatible.

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
- EvasType : 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

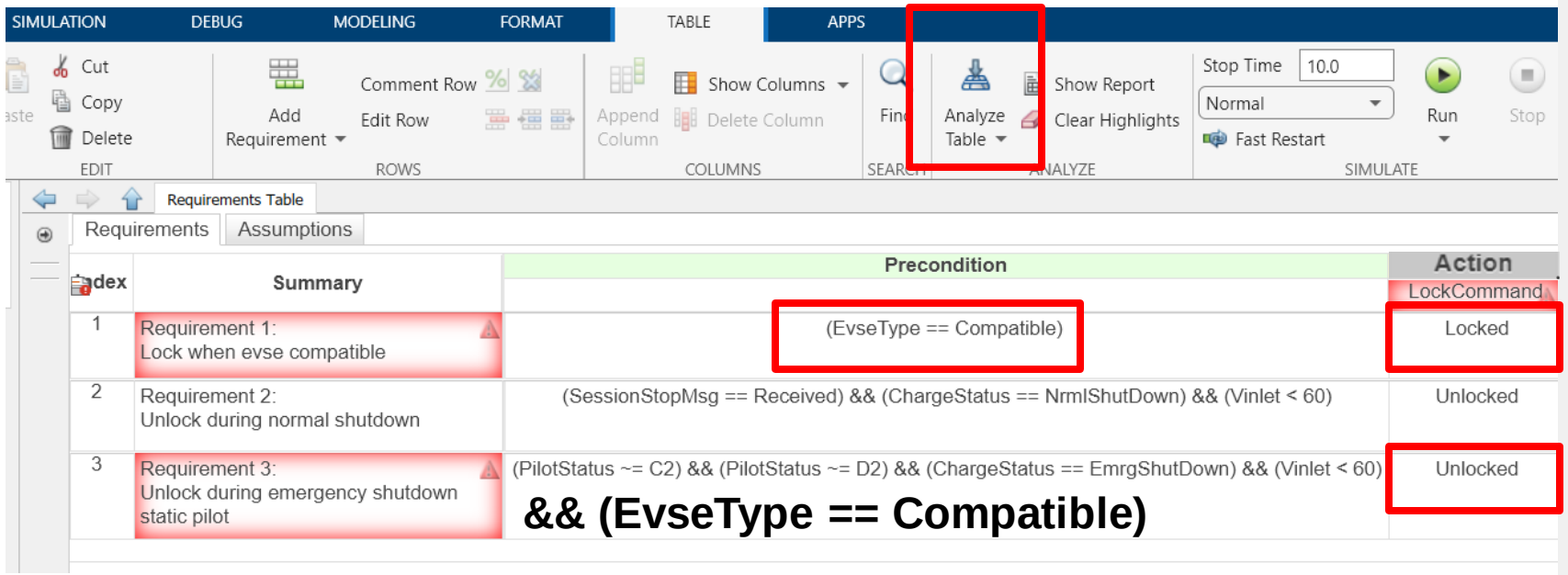


Create Requirement Table

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



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.Not(
EvseType	EvseStat.Compatible	EvseStat.NotI
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 nor							
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							

Analysis Results

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

Block: CordLockReqTable_v2/Requirements Table

Inconsistency Issues

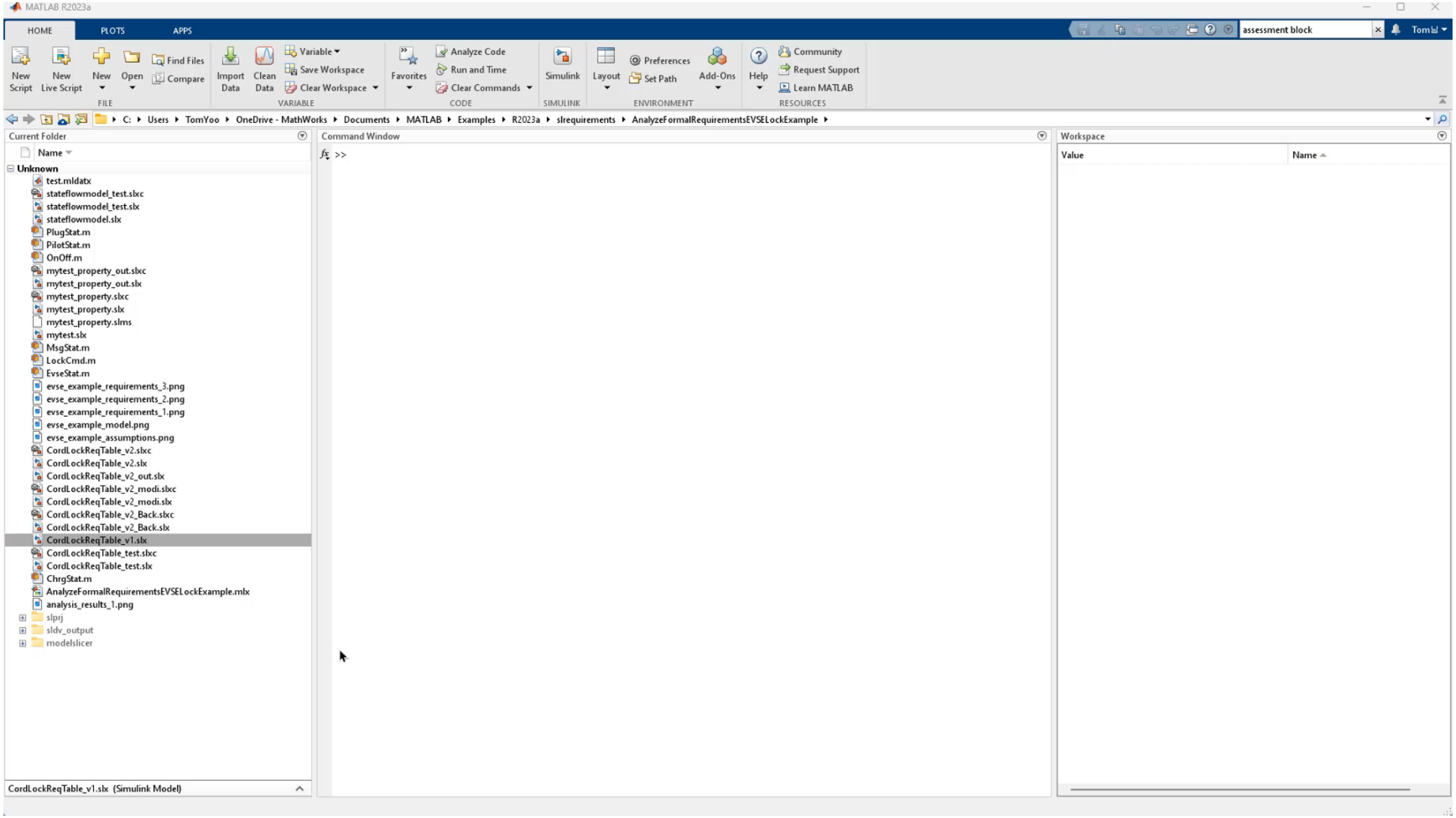
No inconsistency issues found.

Incompleteness Issues

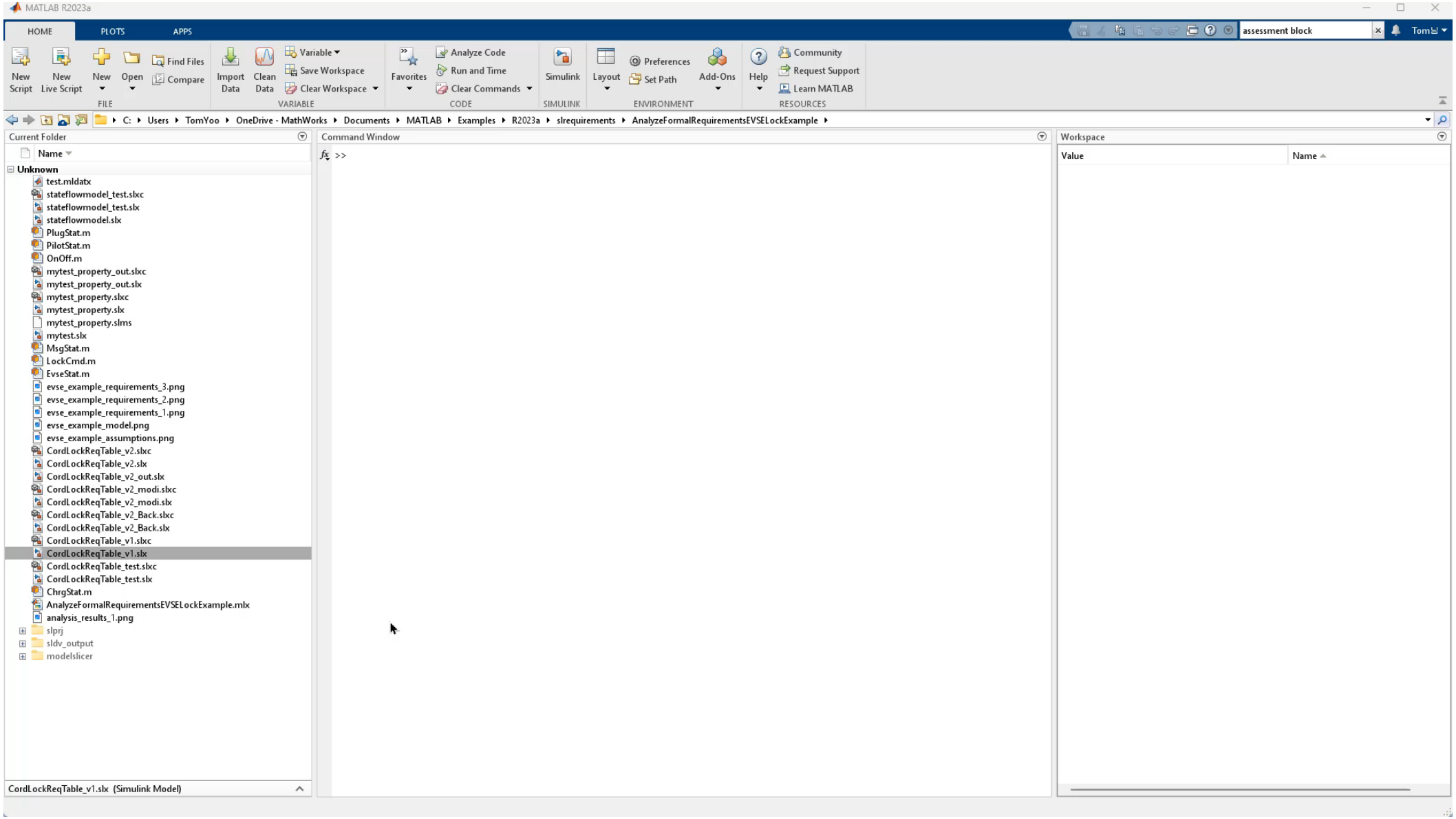
No incompleteness issues found.

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

Analyze Table with incomplete Requirements Table



Analyze Table with complete Requirements Table



Agenda

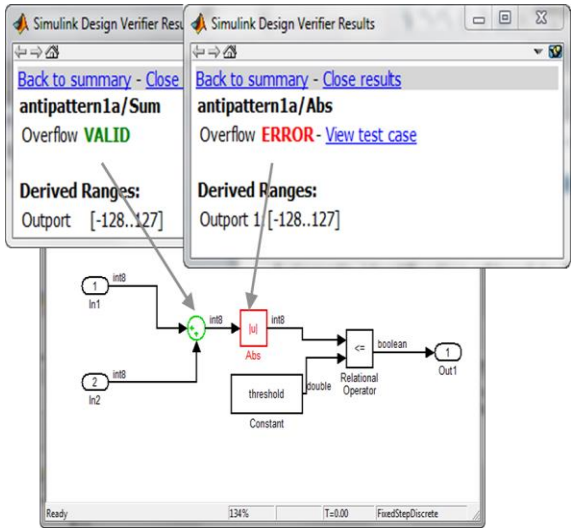
- Why MBD in the perspective of requirements
- Create Formal Requirements with Requirements Table
- Execute Requirement Based Test from Requirements Table
- Formal Verification with the Requirements

Simulink Design Verifier

Use formal methods to identify design errors

Design Error Detection

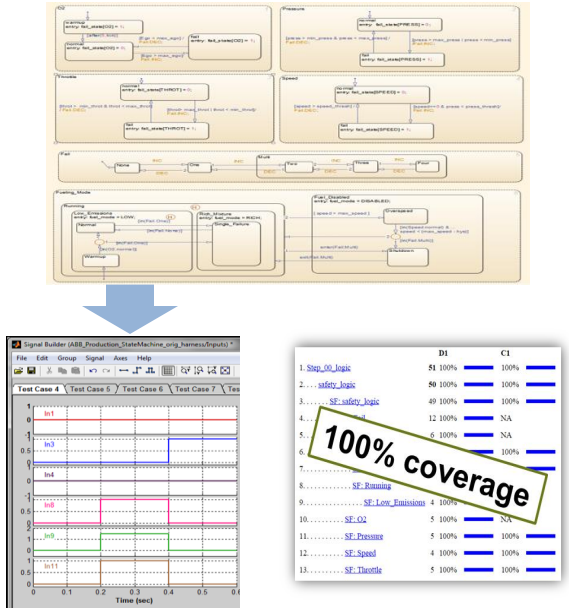
- **Uncover** hard to find dead logic and design flaws



[Examples]

Test Generation

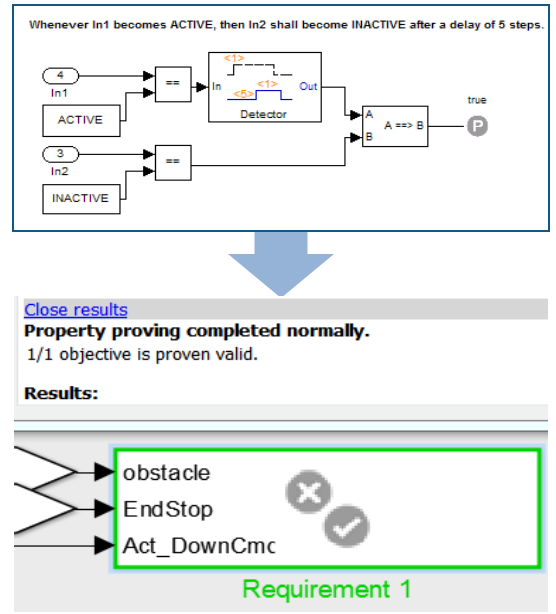
- **Automate** test vector generation to analyze missing coverage



[Examples]

Property Proving

- **Prove** formally design meets requirements

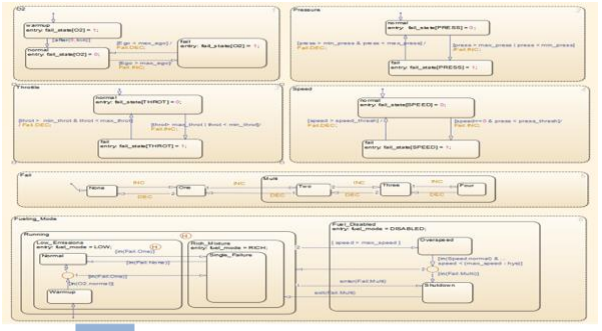


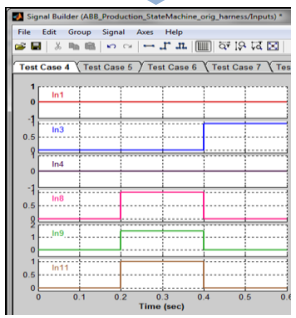
[Examples]

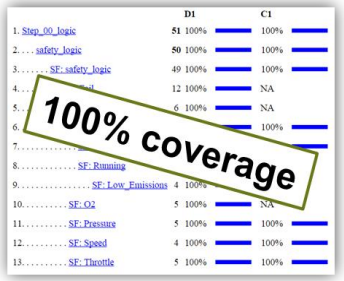
What is the Difference of Generating Testcase from Model vs Table?

Test Generation

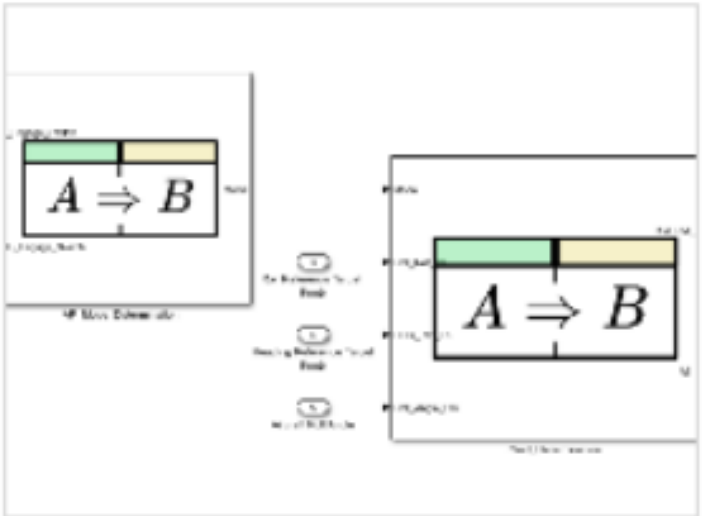
- **Automate** test vector generation to analyze missing coverage







VS



Use Specification Models for Requirements-Based Testing

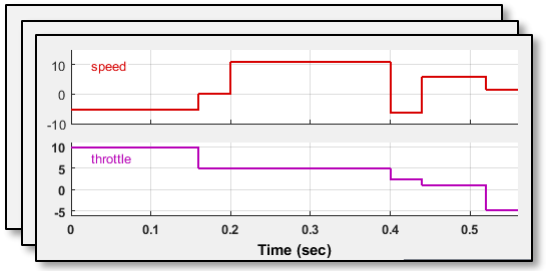
Follow a systematic approach to verify your design model against requirements.

Open Live Script

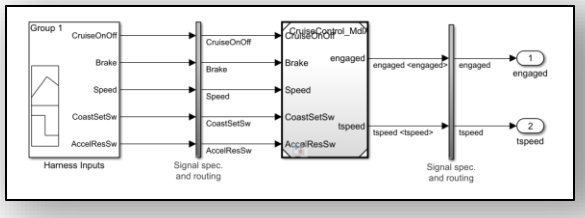
Requirements Based Test Workflow

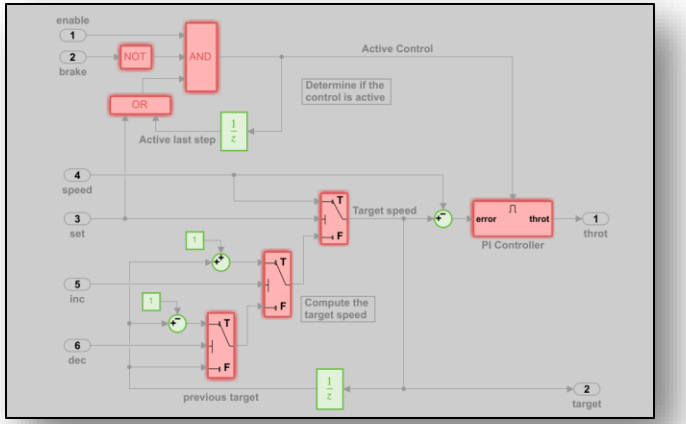
ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl_MdlAdv_ReqLink		31	47%	31%	6%

Partial Coverage



Test Cases



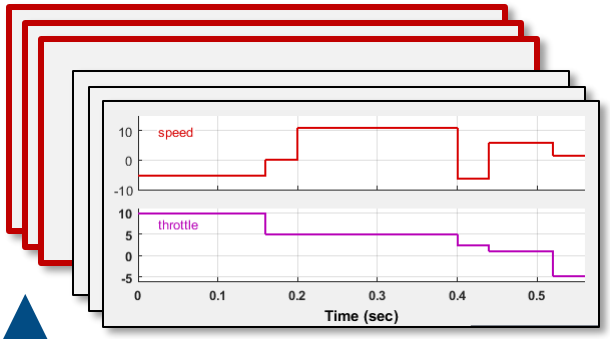


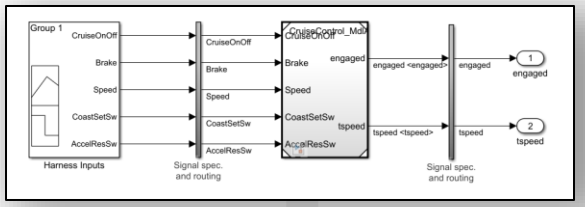
Requirements Based Test Workflow

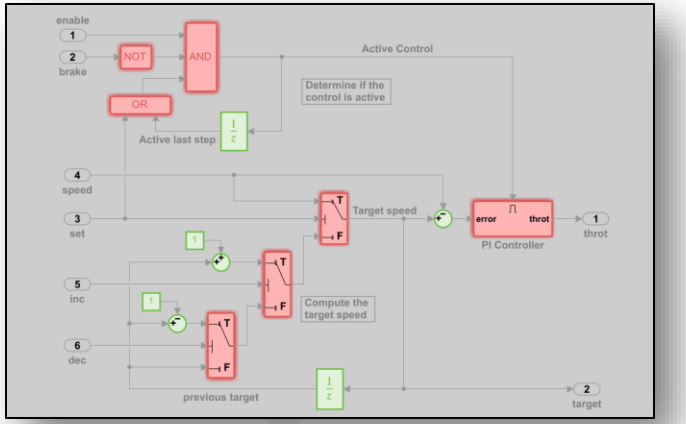
ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl_MdlAdv_ReqLink		31	47%	31%	6%

Partial Coverage

New Test Cases







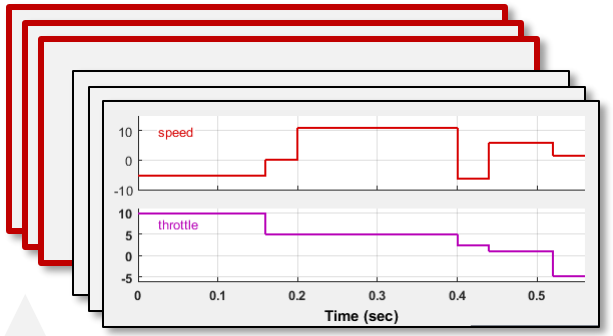
Test Cases

Test Generator

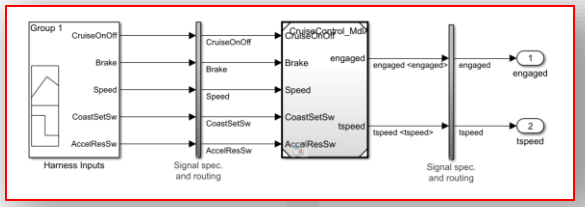
Requirements Based Test Workflow

ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	MCDC
CruiseControl_MdlAdv_ReqLink		31	100%	100%	100%

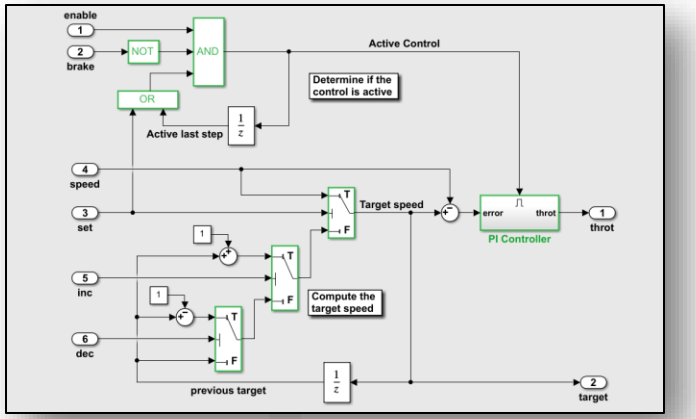
New Test Cases



Test Cases

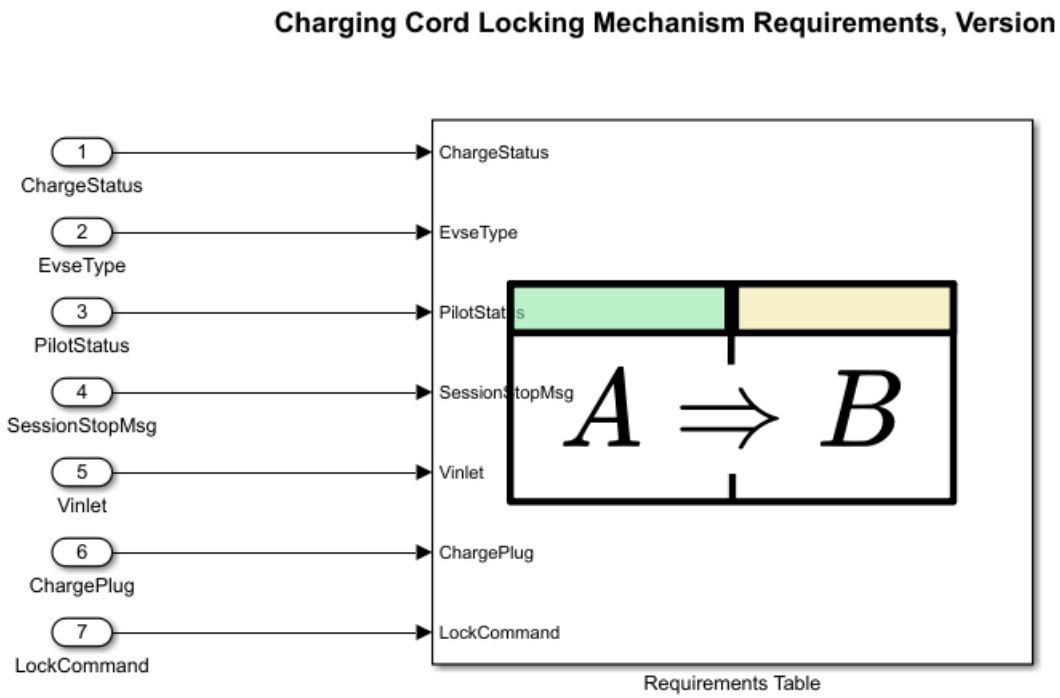


Higher Coverage

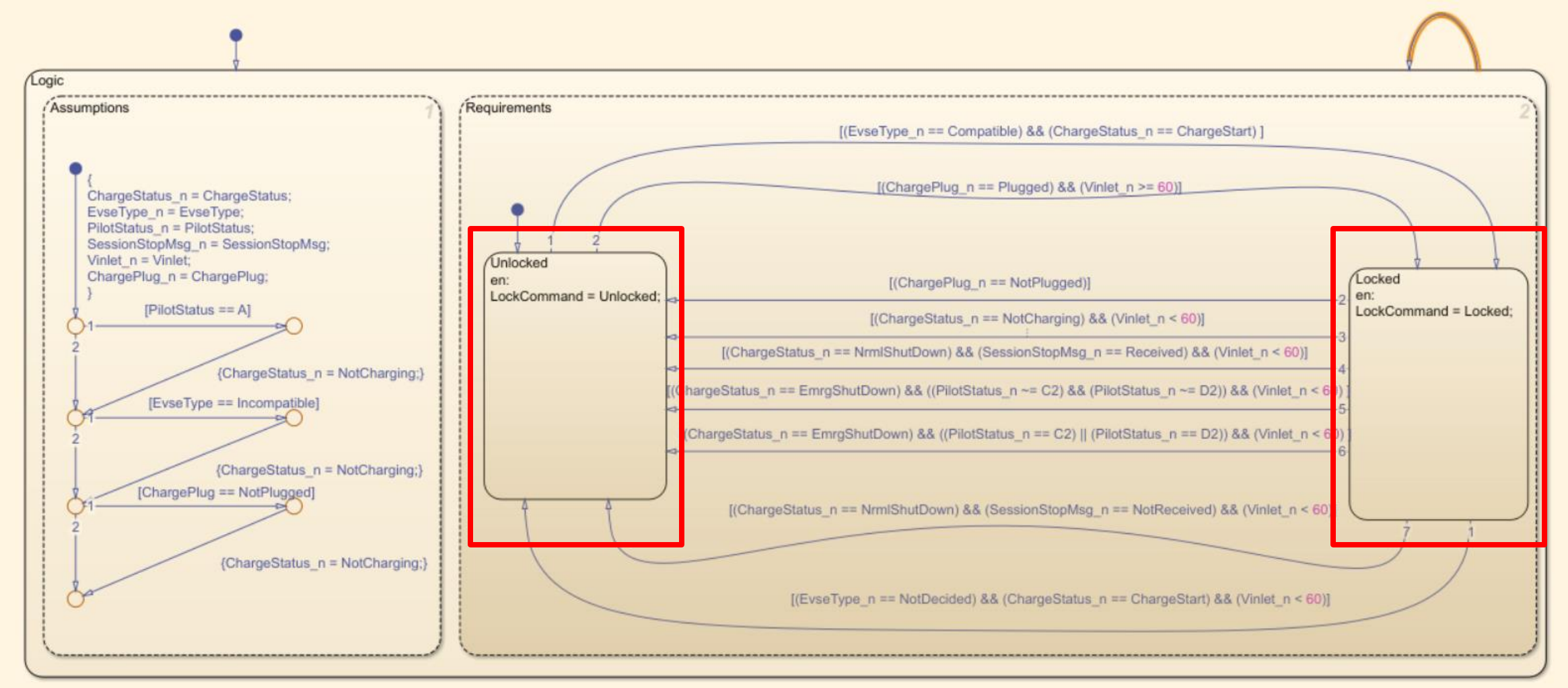


Test Generator

The Merits of generating Testcase from Requirements Table

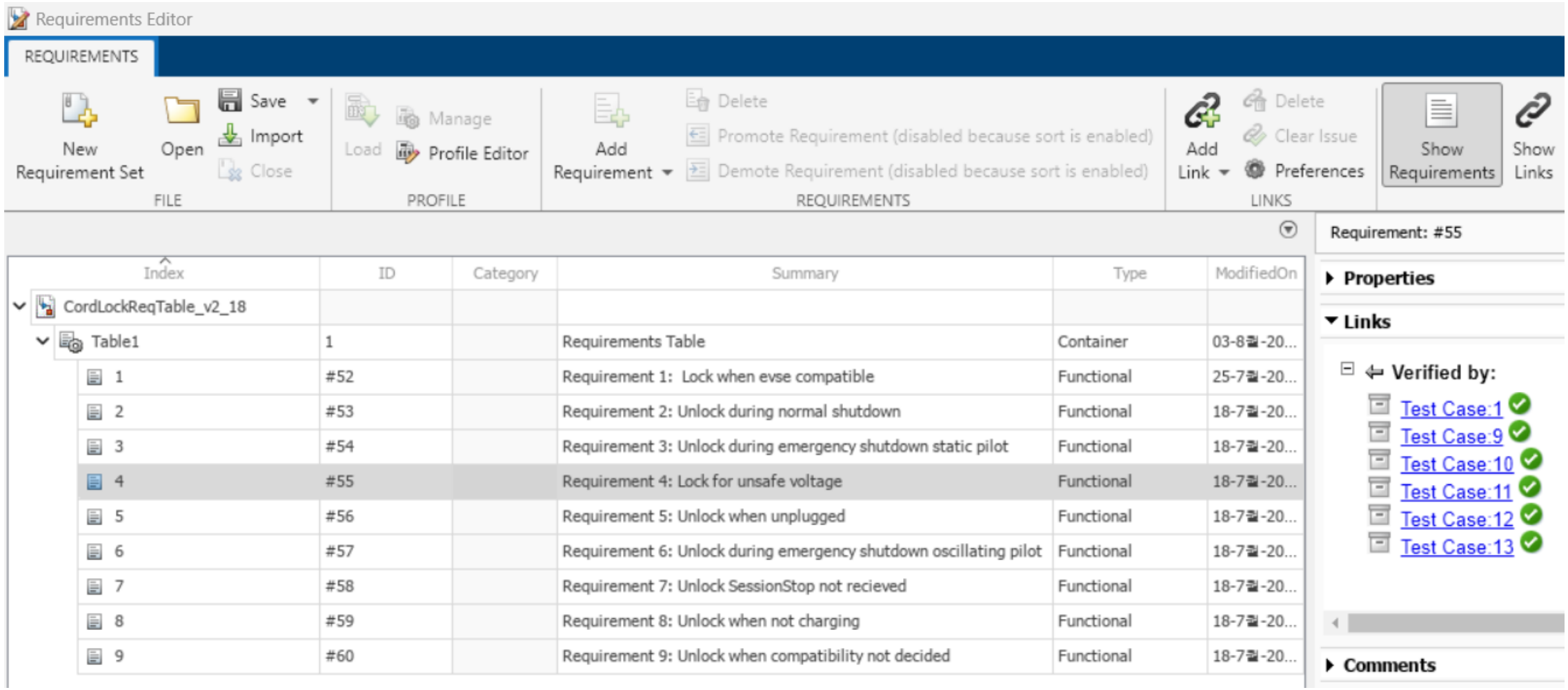


Design Model based on Requirements Table

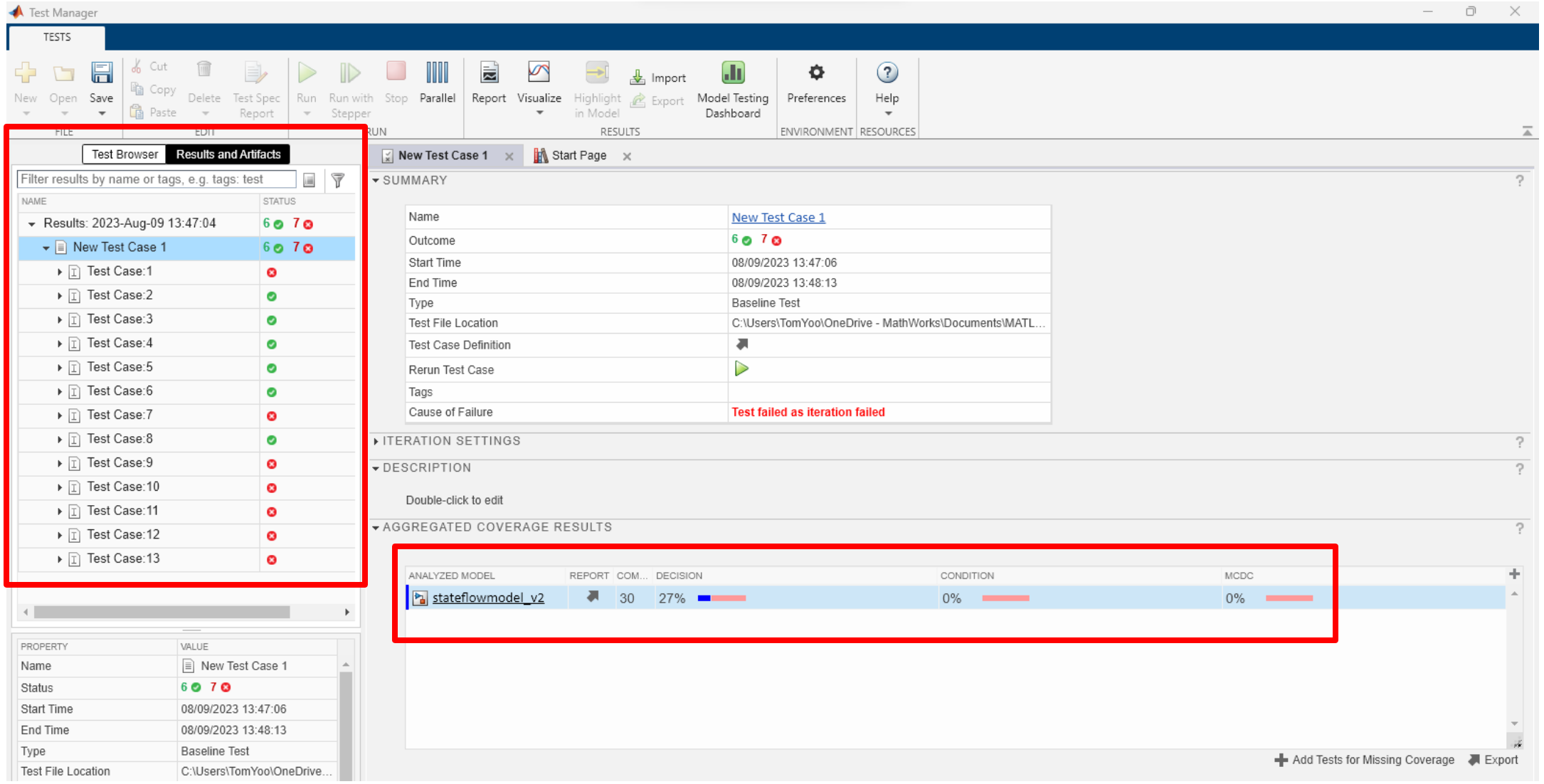


Automatic Linking Requirements Table Rows to Test Cases

R2023a



Requirements Based Test with Simulink Test

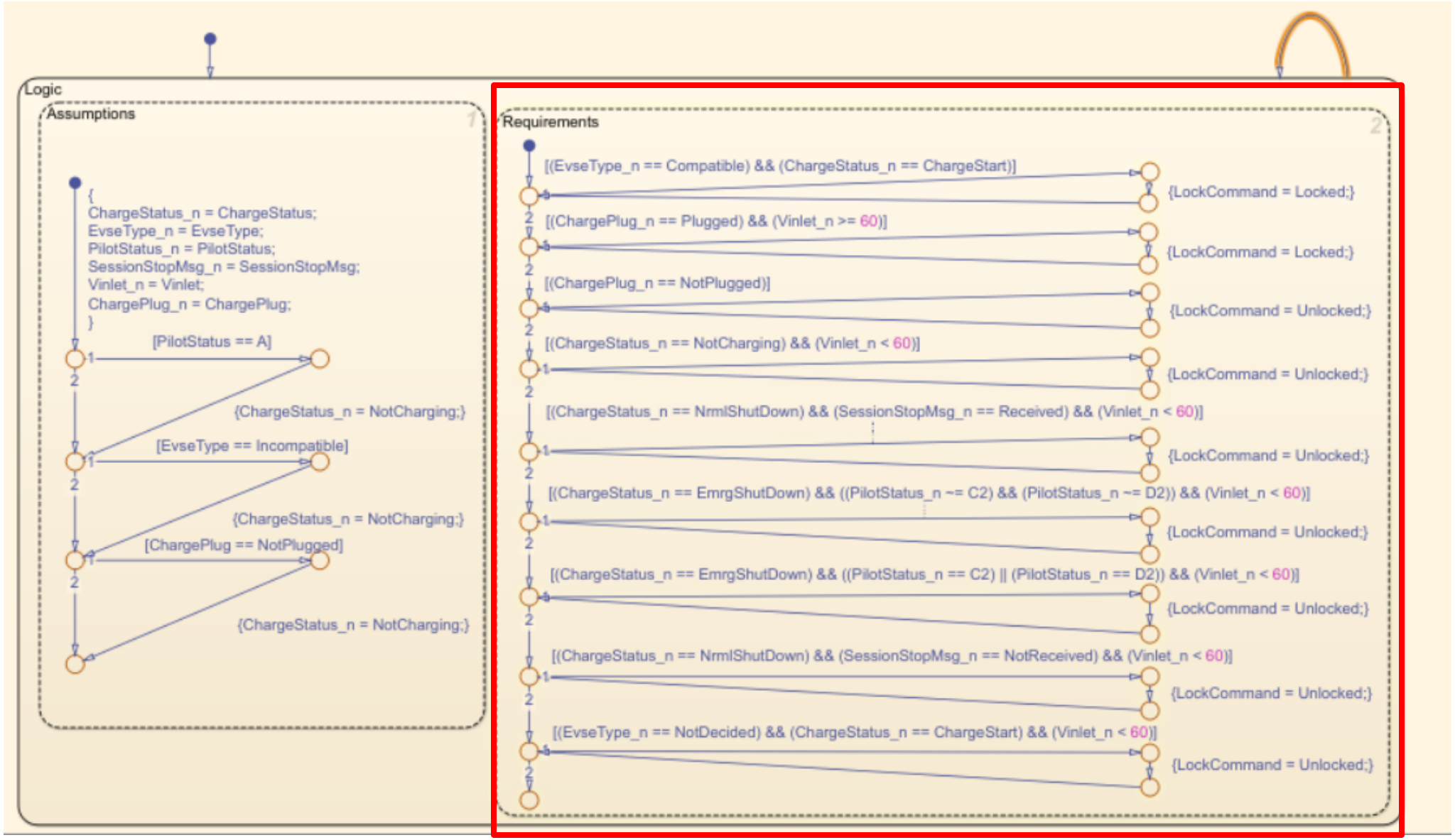


Analyze Requirements Table

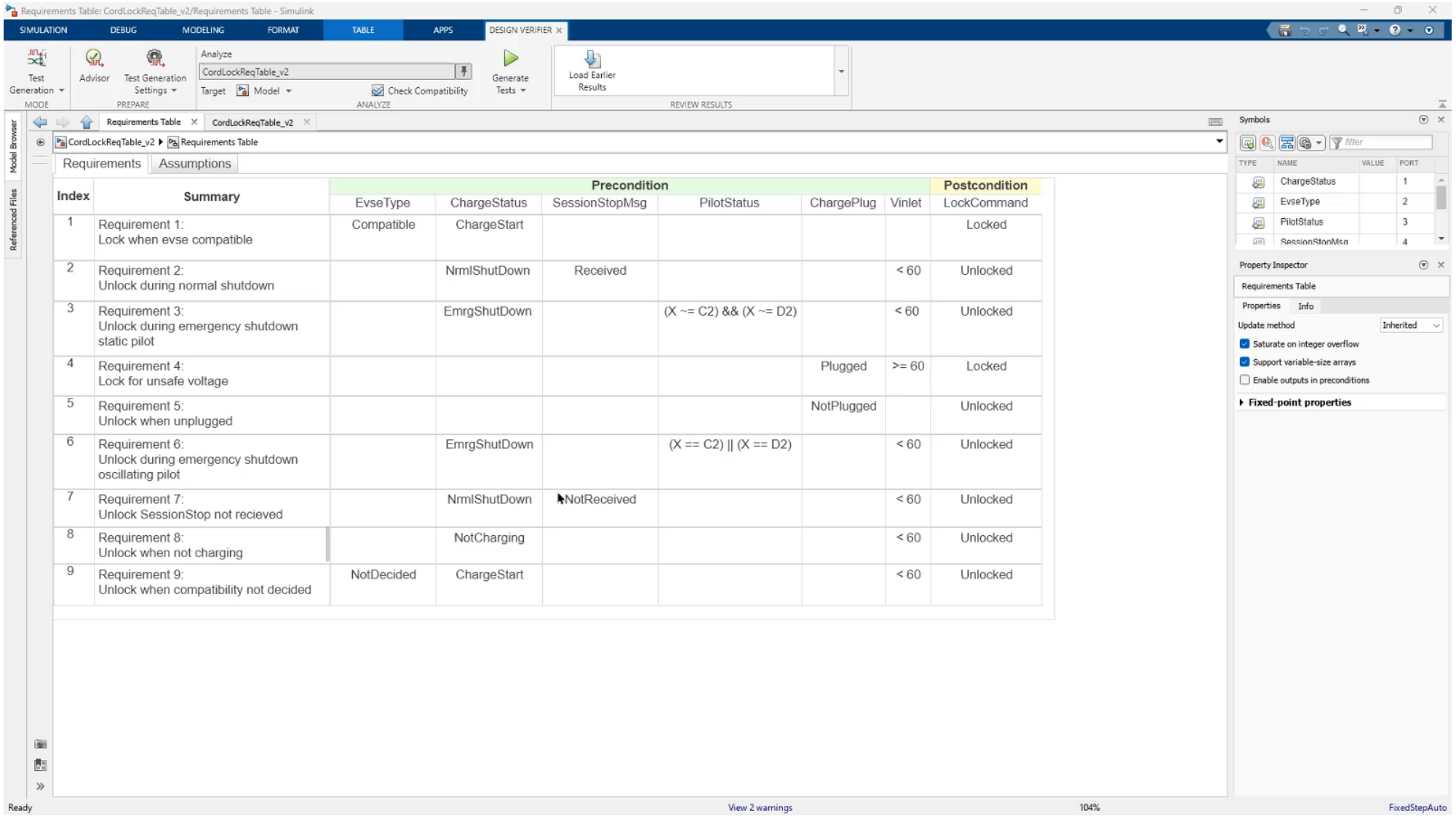
Requirements | Assumptions

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

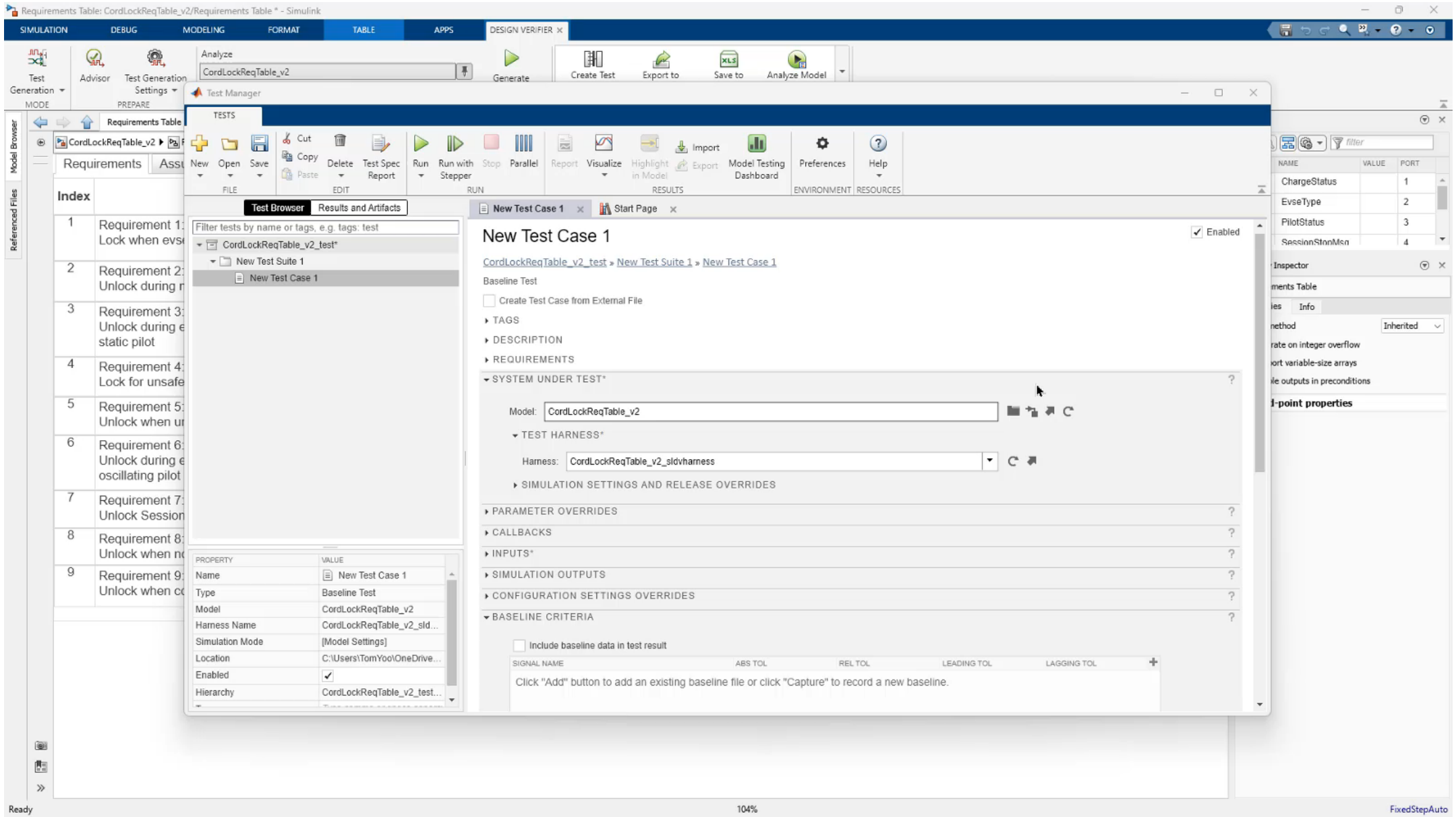


Generate Testcase from Requirements Table



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

Requirements Based Test with Design Model

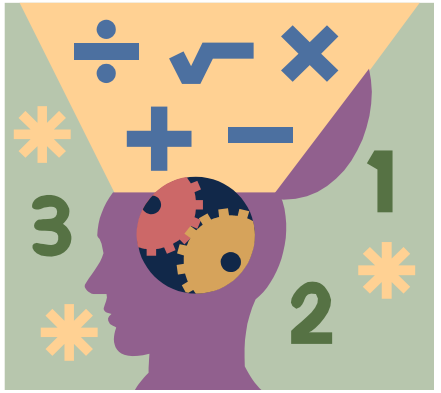


Agenda

- 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

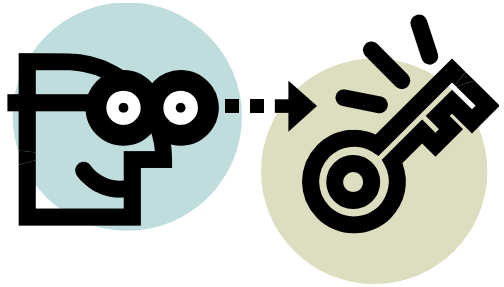
Formal Verification

- Set of math-based techniques for specification, development and verification of algorithm design

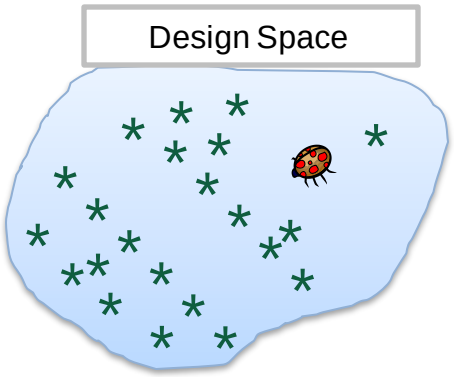


- Works with models of system behavior instead of concrete data values

- Provides deeper understanding of design

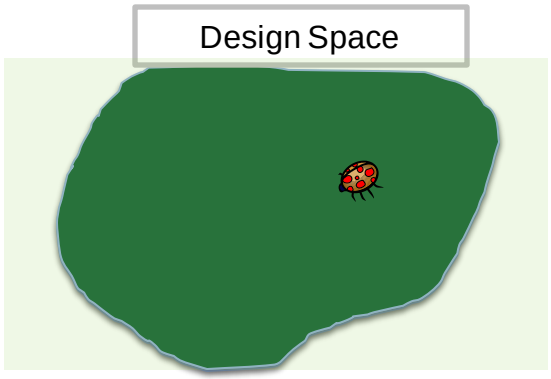


Simulation-Based Testing vs. Formal Verification



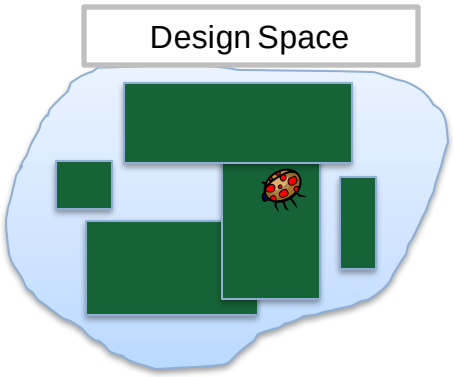
TESTING

Point coverage through simulation



FORMAL VERIFICATION
(Ideal case)

Complete coverage of design space (formal proof)



FORMAL VERIFICATION*
(Real-world)

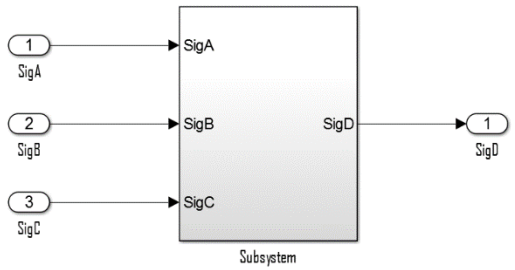
Real-world application of formal verification

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

Property Proving

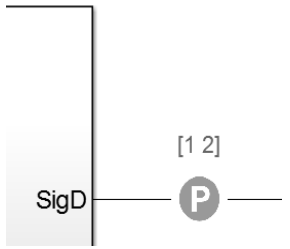
Prove design properties using formal requirement models

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

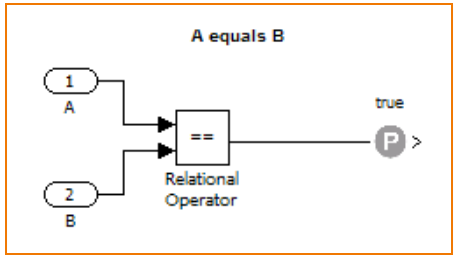


Design model

+



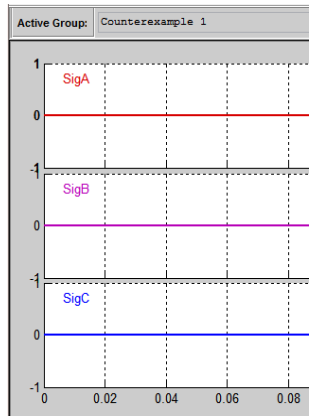
Specified properties



[1 2]

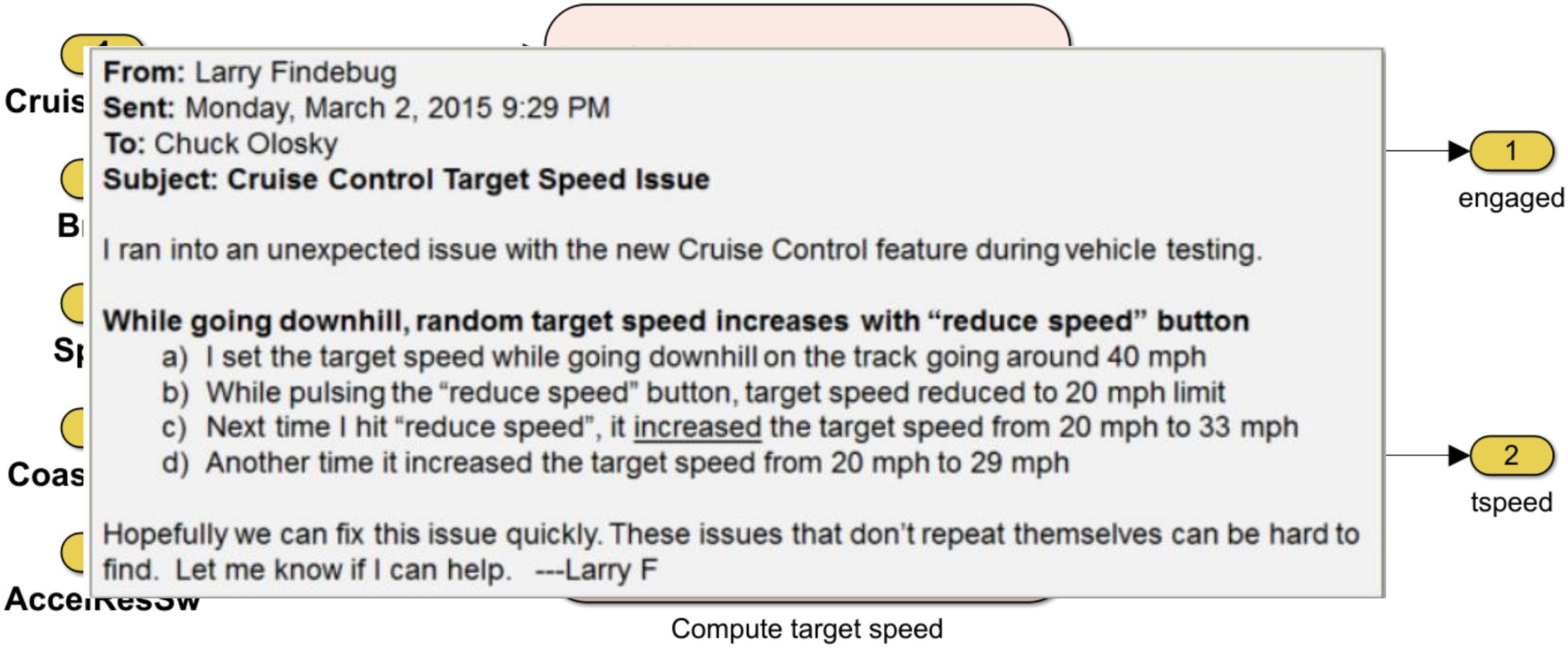
Proof

OR

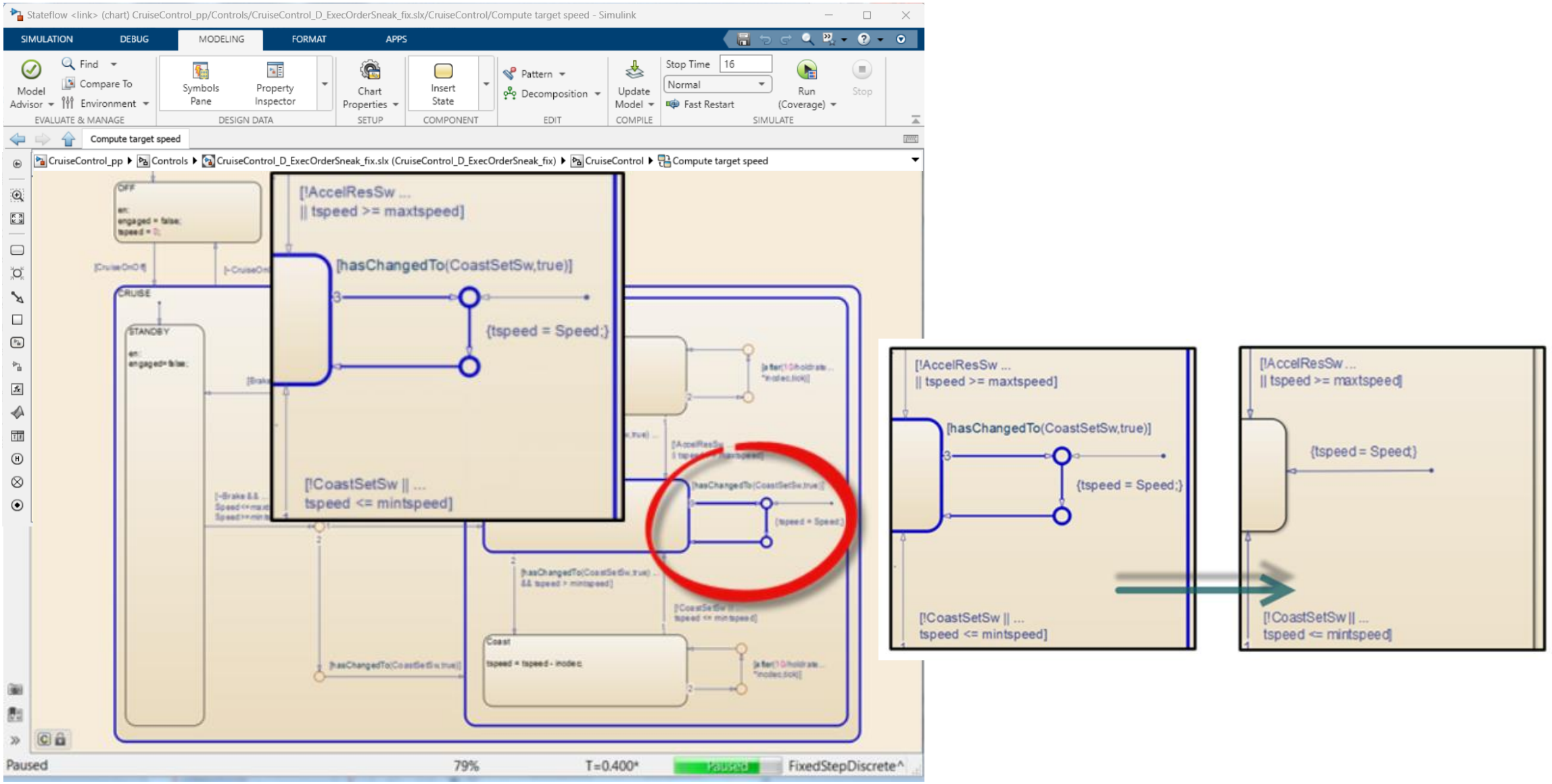


Counterexample

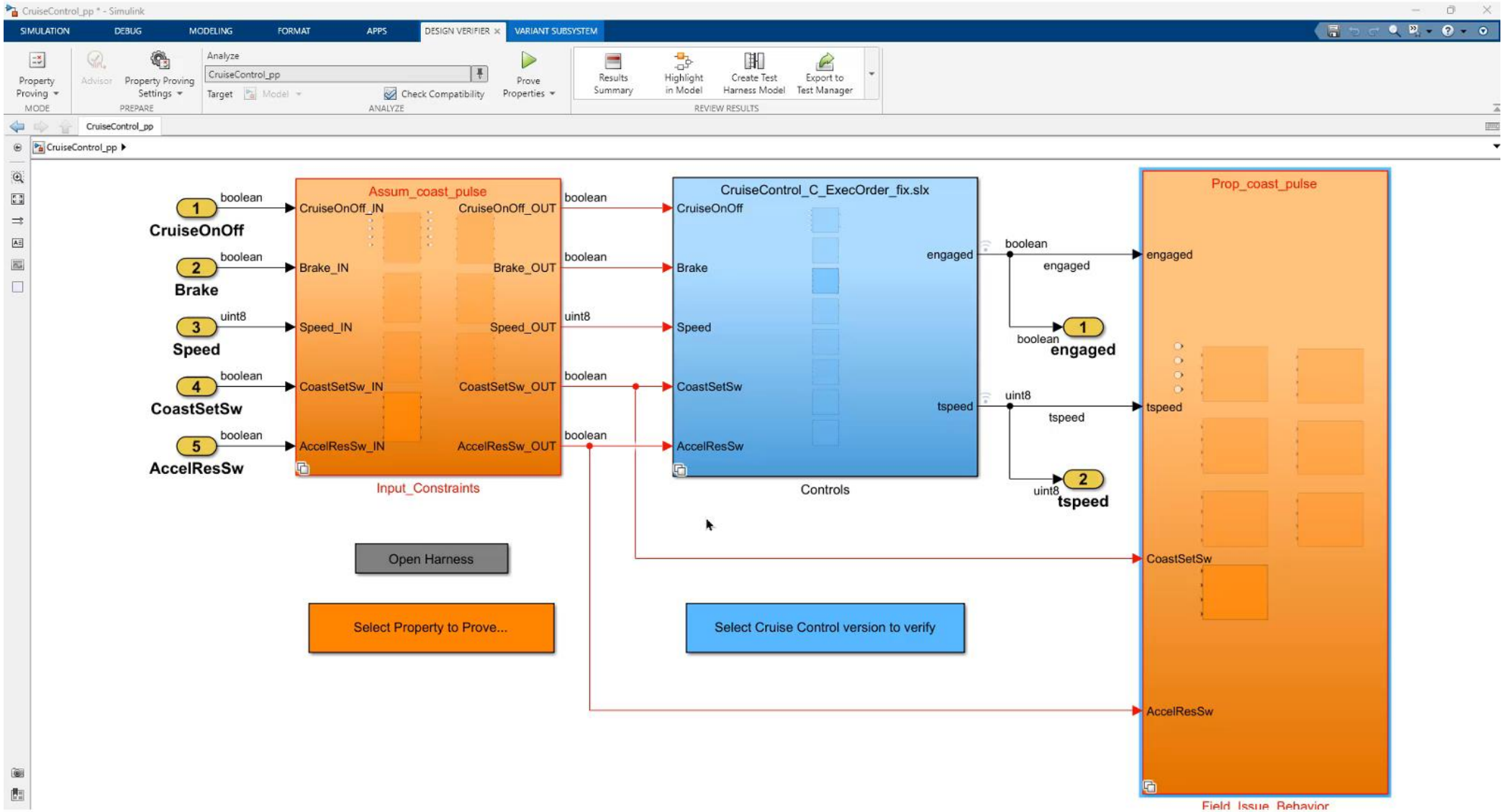
Why is Property proving needed?



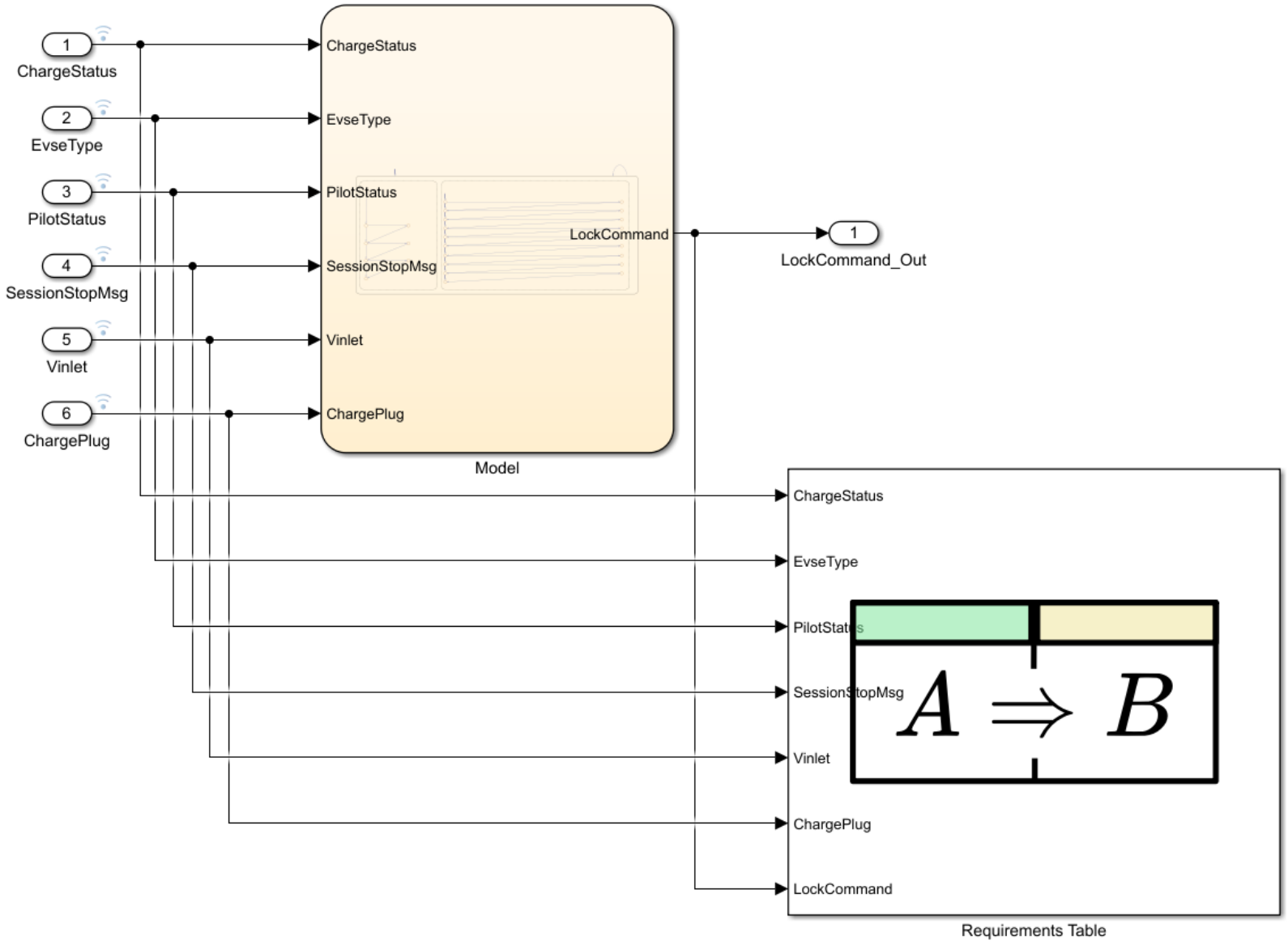
Drawback of Simulation Based Testing



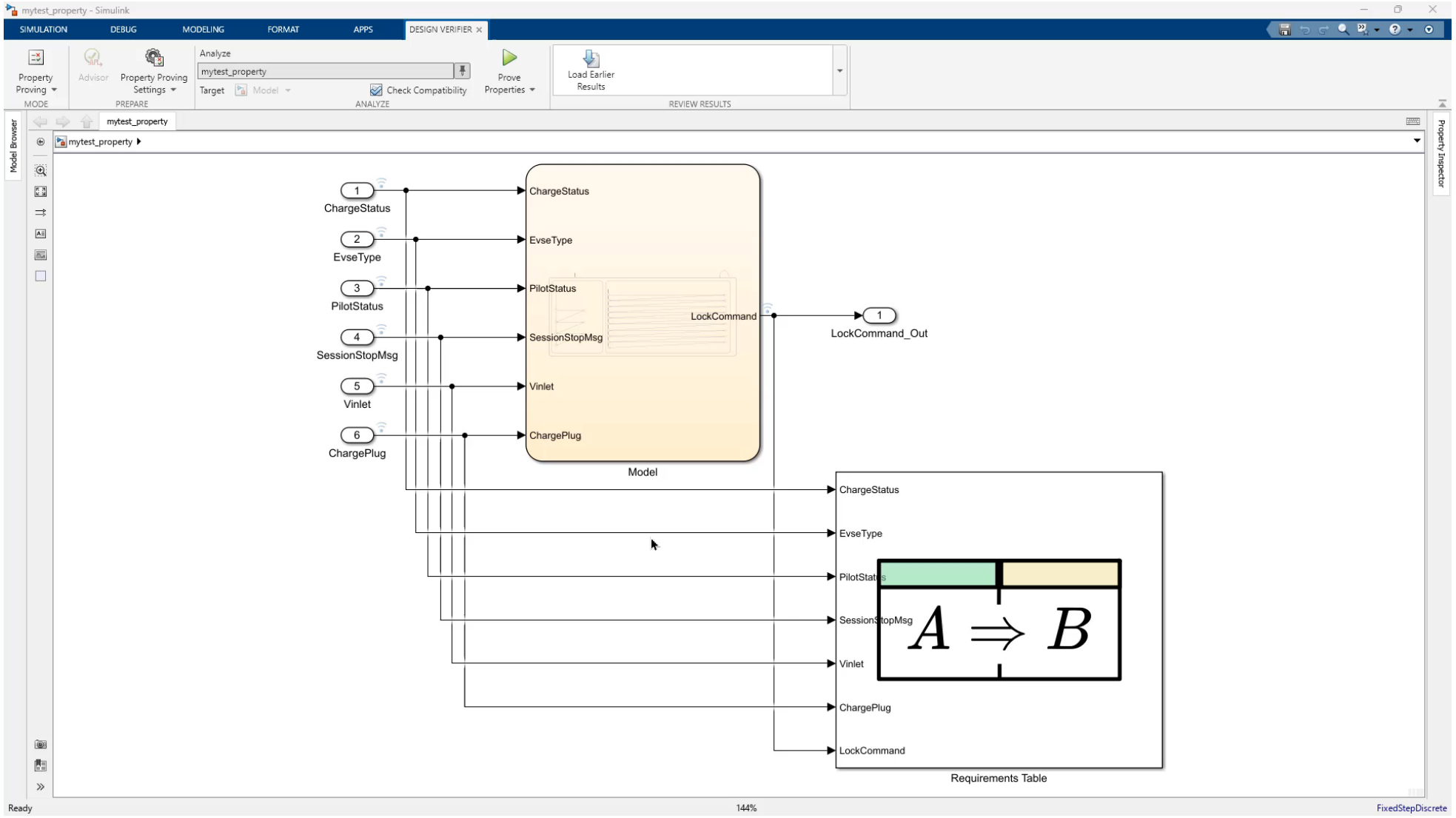
The Process of Traditional Property Proving



Property Proving with Requirements Table

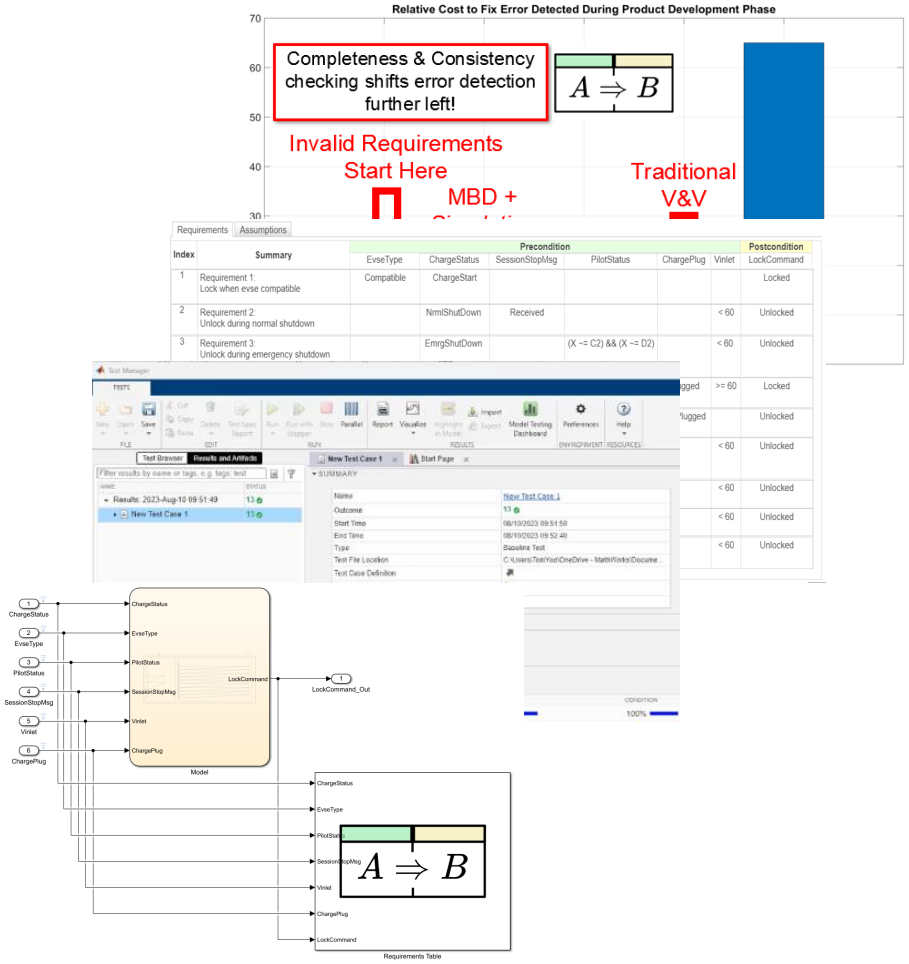


Recording of Property Proving



Key Takeaways

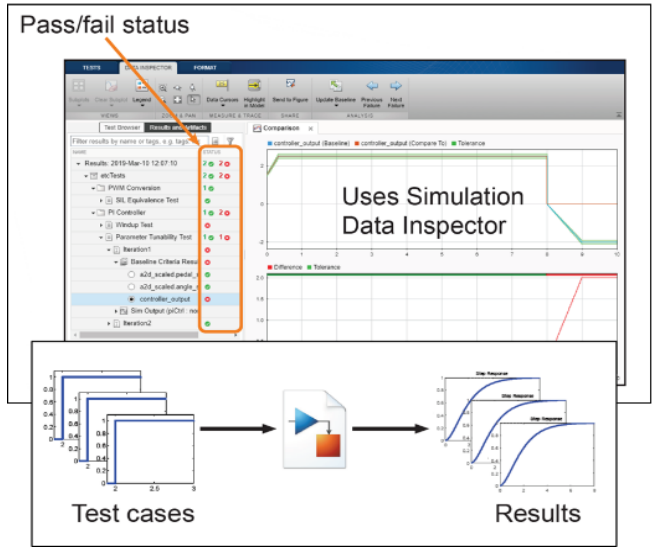
- MBD enables early verification
  - MBD can move starting point of verification from Testing to Design
  - Requirements can be verified by Simulation
- The Features of Requirements Table
  - Requirements Table can check completeness and coherence of Requirements
  - Requirements Table can generate testcases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



임베디드 소프트웨어 개발을 위한 시뮬링크 모델 검증

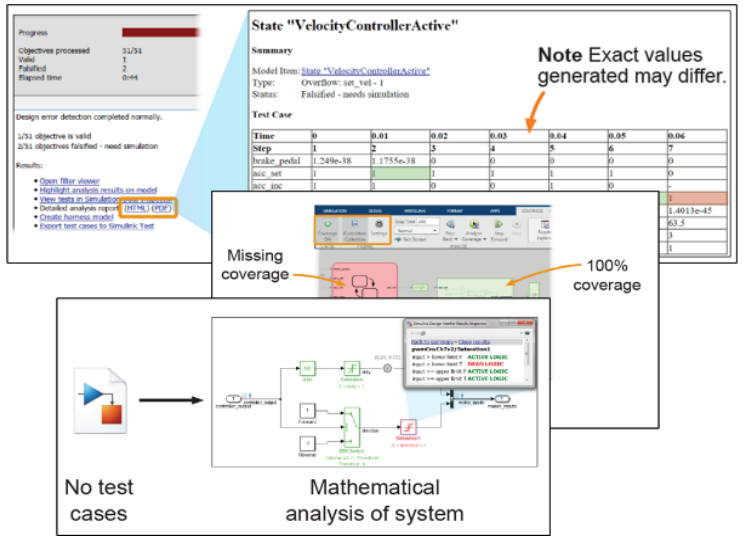
Simulation-Based Testing with Simulink (1 day)

- 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