

# SESE TOUR 2022

## Systems Engineering for a Sustainable World

*How to Model Complete and  
Consistent Requirements*

Vincenzo Petrella

10 May 2022

# Biography



Vincenzo Petrella

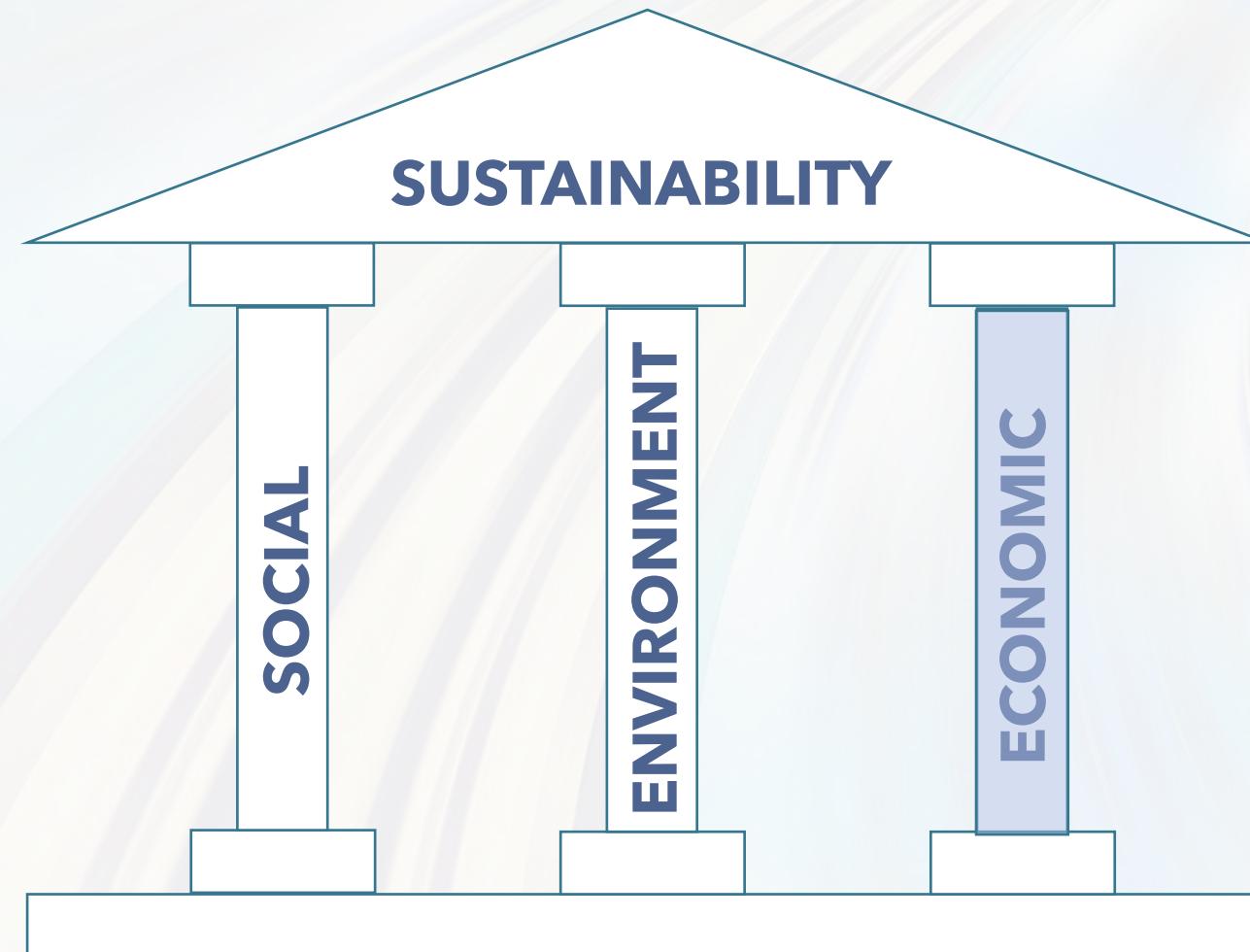
- Vincenzo holds a M.Sc. in Aeronautical Engineering from La Sapienza University of Rome with a major in flight-dynamics and helicopter flight-mechanics.
- Before joining MathWorks, he worked as a consultant in the aeronautical modelling department of Leonardo - Electronics Division located in Ronchi dei Legionari (GO). During his time at Leonardo, he was involved in the development of flight training simulator systems.
- In 2018 he took on the role of Application Engineer for MathWorks with a focus on Model-Based Systems Engineering and Model-Based Design.

# Agenda

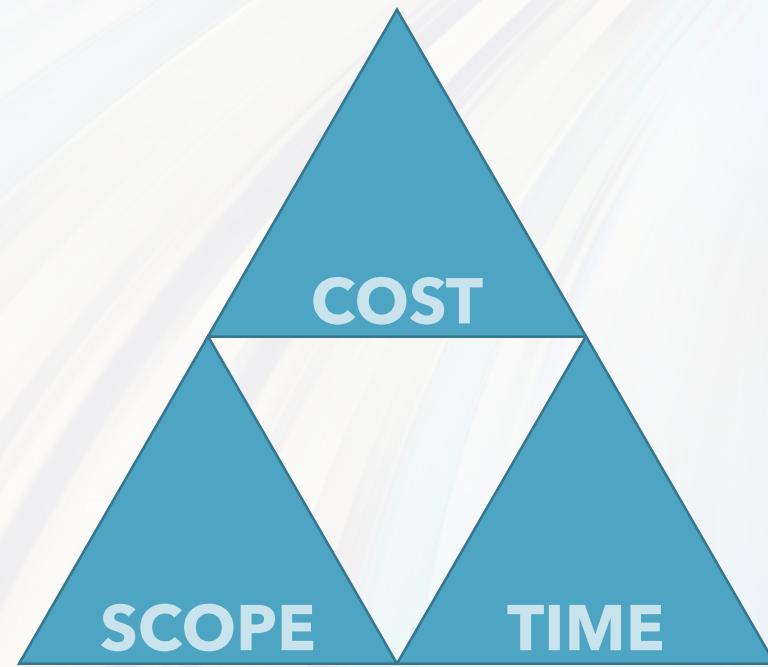
- Introduction
- Common Challenges with Requirements Validation
- What are engineers doing today to address these challenges?
- Why should you model requirements?
- How to model and validate requirements: a practical example



# Requirements: The key to sustainability



# Requirements: The key to sustainability



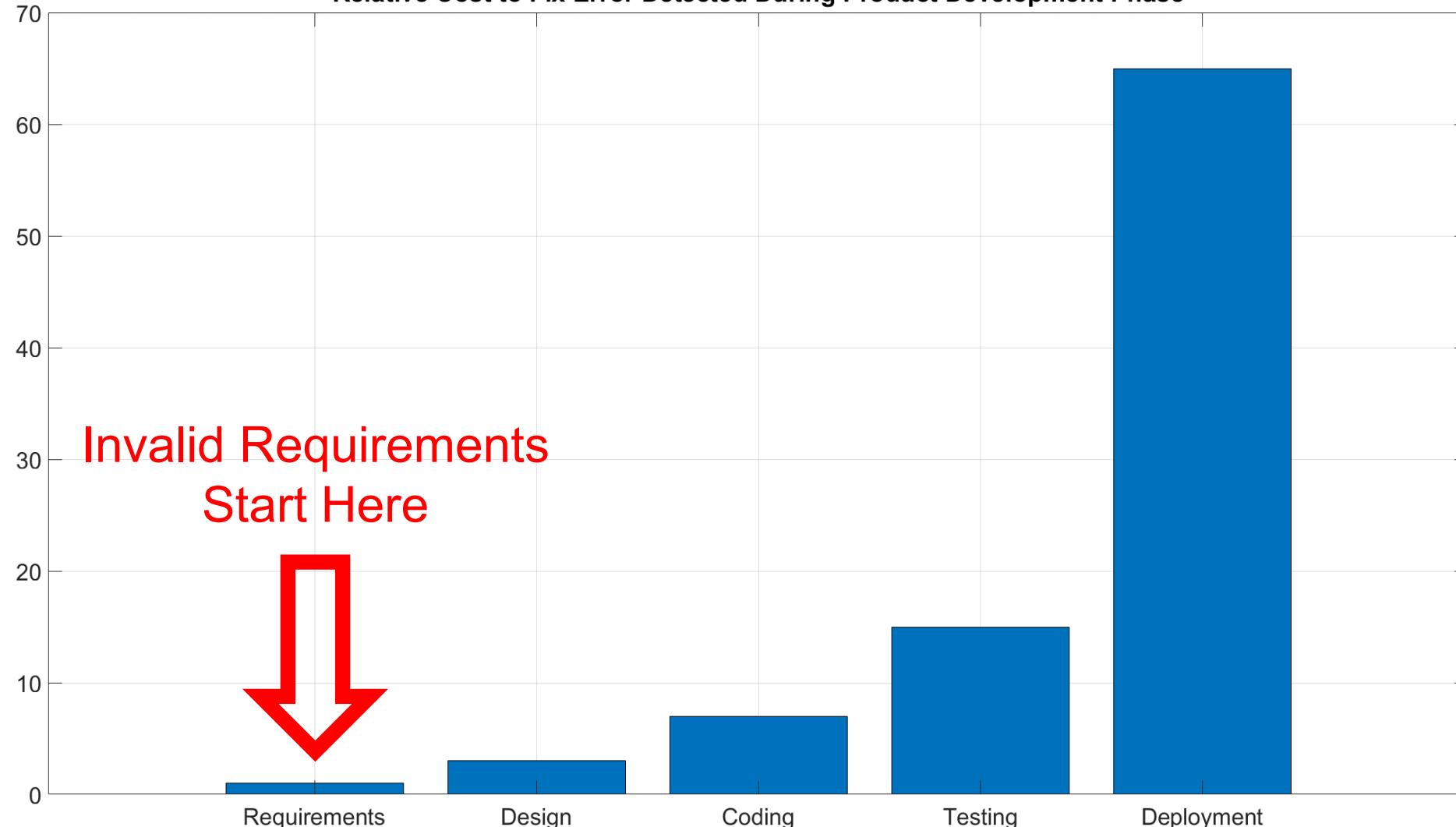


# Why do 71% of Embedded Projects Fail?

## Poor Requirements Management

Sources: Christopher Lindquist, *Fixing the Requirements Mess*, CIO Magazine, Nov 2005

Relative Cost to Fix Error Detected During Product Development Phase



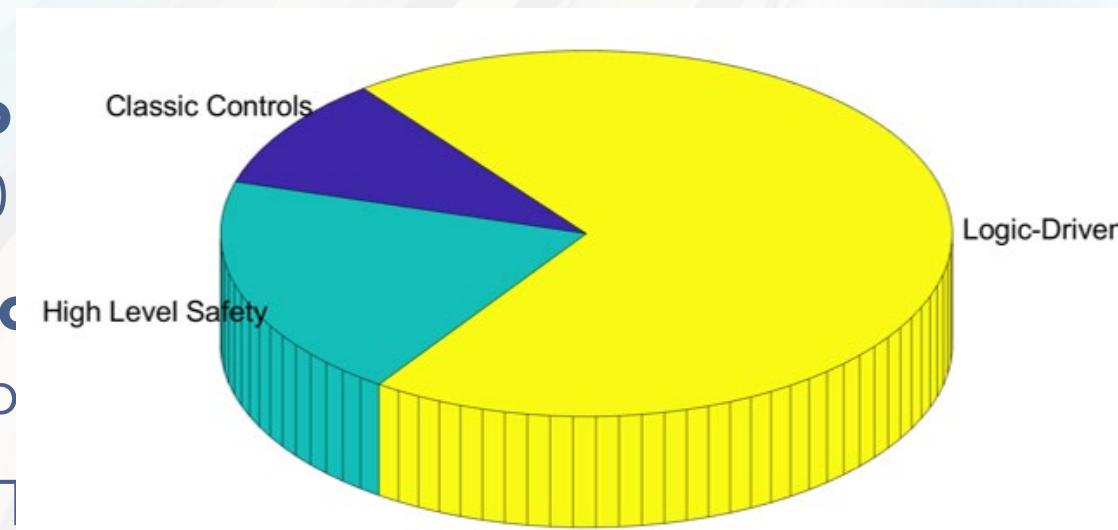
Data gathered by Hewlett Packard referred by XB in 2017

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

# A Typical Population of Requirements

- Many requirements are simple and “logic-driven”

- **Fault detection**: “Fault detection logic exceeds 1,000 lines of code.”
- **Signal selection**: “Signal selection logic uses the three temperature sensors to calculate a value between 0 and 1000.”
- **Mode logic**: “Mode logic handles mode changes when no valid altitude sensors are available.”



or when it  
value between  
de when no valid

# 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



# What are engineers doing today?

- 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!**



# Why Should You Model Requirements?

- Mathematically **rigorous**
- **Executable** through simulation
- **Easier to author and maintain** than “pseudocode” text requirements
- **Easier to analyze** as requirement set grows
- **Reusable** for verification activities (e.g., automatic test generation)





## PRECONDITION

Expression that define the condition when the requirement is valid

## POSTCONDITION

Expression defining the condition that must be achieved when the precondition is met

Requirements		Assumptions			
Index	Summary	Precondition	Duration	Postcondition	Action
1	Corrected ground speed positive when corrected air speed positive	>0		>0	
2	Corrected ground speed within limits when corrected air speed within limits	(-5, 1000)		(0, 500)	

**Each row of the table define a requirements**  
Text based description of the requirement

## SUMMARY

## DURATION (optional)

The time in seconds during which the precondition must be satisfied

# Precondition and Postcondition Synthax

Header

Requirements		Assumptions		
Index	Summary	Precondition	Postcondition	
1	Requirement 1	u 1,2,3	y1 [0.3, 0.5]	y2 >= 0.3 && y2 <= 0.5

Syntax	Description
value	Checks if the header data equals value.
value1,value2,value3,...,valueN	Checks if the header data equals either of the comma-separated values.
[value1,value2]	Checks if the header data is greater than or equal to value1 and is less than or equal to value2.
(value1,value2)	Checks if the header data is greater than value1 and is less than value2.
[value1,value2)	Checks if the header data is greater than or equal to value1 and is less than value2.
(value1,value2]	Checks if the header data is greater than value1 and is less than or equal to value2.



# Actions

Requirements		Assumptions	
Index	Summary	Precondition	Action
1	Requirement 1	$u_1 > 0$	$y_1 = 2*u_1$
2	Requirement 2	$u_2 > 0$	$y_2 = 0.5*u_2$



## ACTION (optional)

Defining what actions are performed when the precondition is valid.

Examples of actions:

- Assigning a value
- Call a function

Requirements		Assumptions		Action		
Index	Summary	Precondition	Postcondition			
1	Requirement 1	$u > 0$	$y_1 \geq 0$	$y_2 \geq 0$	$y_3$	$[1,2,3]$



The action sets  $y_3$  equal to a vector of values.



# Assumptions

You may have a physical or mathematical limitations prevent data from being certain values.

Requirements		Assumptions	
Index	Summary	Precondition	Postcondition
1	Assumption 1		

If the precondition of an assumption is met

If the precondition of an assumption is **not** met

the postcondition enforces a constraint on data used in the requirements

ignores the constraint in the postcondition.



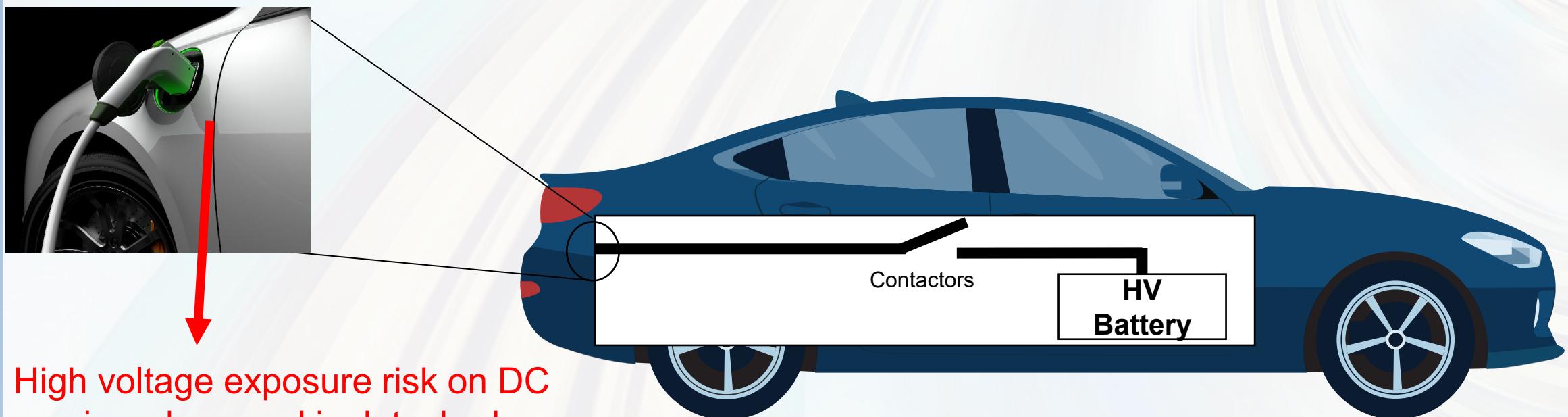
# Assumptions

Requirements		Assumptions	
Index	Summary	Precondition	Postcondition
1	Switch is off then throttle is zero	Ignition == <b>false</b>	Throttle == 0
2	Throttle is engaged		Throttle >= 0

- If the car ignition is off, then the throttle must also be off.
- The throttle cannot be negative.

# Example: 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

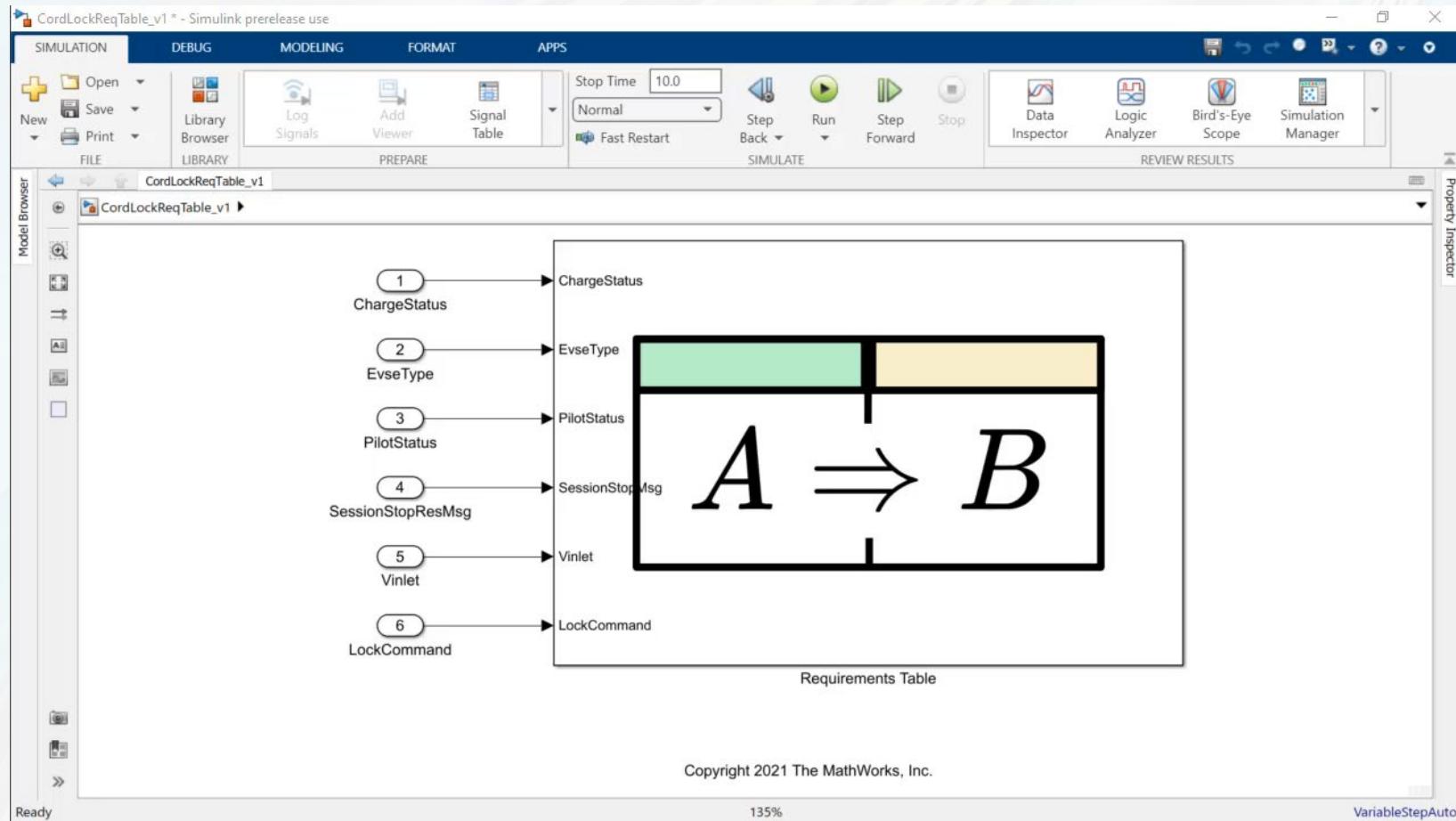
**Requirement 1:** The electric vehicle shall lock the charger in place if the electric vehicle supply equipment is compatible.

**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.



# Requirements writing and analysis



# Inconsistency Issues

The block detects a scenario where the data can equal more than one value during simulation.

## 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

The block detects a scenario where the data is not defined during simulation.

## 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.NotCharging
EvseType	EvseStat.Compatible	EvseStat.NotDecided
PilotStatus	PilotStat.E_F	PilotStat.A
SessionStopMsg	MsgStat.NotReceived	MsgStat.NotReceived
Vinlet	87.6303	0



# Incompleteness Issues

Requirements		Assumptions		Precondition						Postcondition	
Index	Summary	EvseType	ChargeStatus	SessionStopMsg	PilotStatus	ChargePlug	Vinlet	LockCommand			
1	Requirement 1: Lock when evse compatible	Compatible	ChargeStart							Locked	
2	Requirement 2: Unlock during normal shutdown		NrmIShutDown	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 received		NrmIShutDown	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		



# Assumptions

These assumptions will be used when analyzing the table

- If the pilot status is on standby (A), then the car is not charging.
- If the vehicle is not compatible with the charger, then the vehicle is not charging.
- If the charger is not plugged in, then the vehicle is not charging

Requirements		Assumptions	
Index	Summary	Precondition	Postcondition
1	Assumption 1	PilotStatus == A	ChargeStatus == NotCharging
2	Assumption 2	EvseType == Incompatible	ChargeStatus == NotCharging
3	Assumption 3	ChargePlug == NotPlugged	ChargeStatus == NotCharging



# Observe model behavior

Requirements		Assumptions			
Index	Summary	Precondition	Duration	Postcondition	Action
		1		Requirement 1	$u_1 > 0$
2	Requirement 2	$u_2 > 0$		$y_2 > 0$	

The diagram illustrates a Requirements Table and its interaction with a Subsystem. The Requirements Table is a grid with columns: Index, Summary, Precondition, Duration, Postcondition, and Action. It contains two rows: Requirement 1 (Index 1) with precondition  $u_1 > 0$  and postcondition  $y_1 > 0$ , and Requirement 2 (Index 2) with precondition  $u_2 > 0$  and postcondition  $y_2 > 0$ . A large grey parallelogram labeled 'Subsystem' is positioned above the table. A red dashed arrow points from the 'Constant1' entry in the Requirements Table to the bottom-left corner of the 'Subsystem' parallelogram. A blue dashed arrow points from the 'A  $\Rightarrow$  B' block to the bottom-right corner of the 'Subsystem' parallelogram.

use the preconditions to specify model input behavior

use the postconditions to specify model output behavior

Requirements Table

Subsystem

Constant1

2

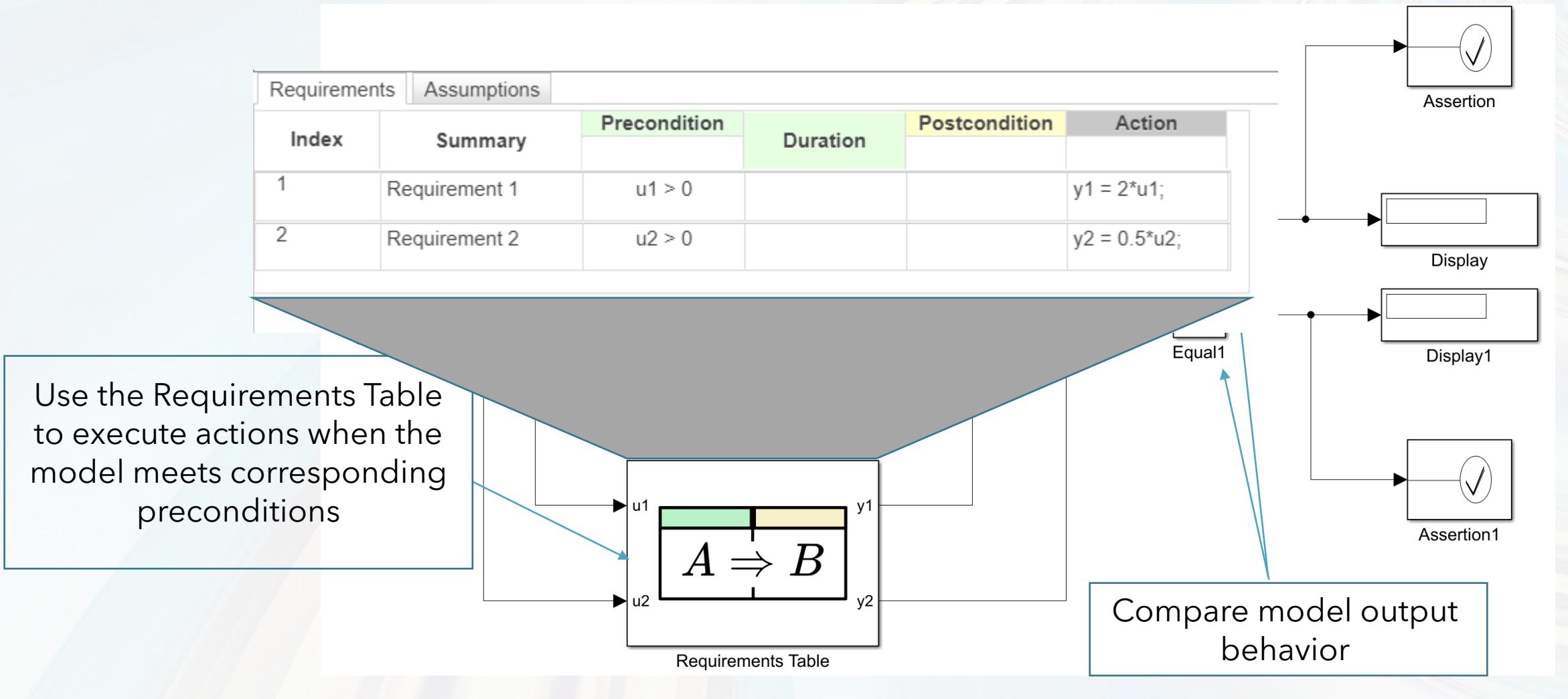
$A \Rightarrow B$

If the model behavior does **not** satisfy the postcondition for a requirement when the precondition is satisfied, **a warning is provided**

SESE TOUR 2022 \_ MAY 09 TO 11  
"SYSTEMS ENGINEERING FOR A SUSTAINABLE WORLD"

27

# Generate model behavior





SSSE



# Questions ?

