

# From GUIs to Generative AI: Explainability in the tooling for engineered system design



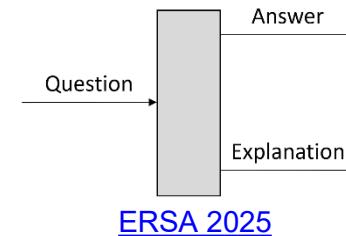
**Akshay Rajhans**

*MathWorks Advanced Research & Technology Office*

Keynote Talk, 4th International Workshop on Explainability of Real-time Systems and their Analysis, RTSS 2025

Boston, MA, USA

December 2, 2025



# Complex engineered systems (cyber-physical systems)



Autonomous and Connected Vehicles



Electric Cars and Aircraft



Marine Vehicles



Manufacturing Machinery



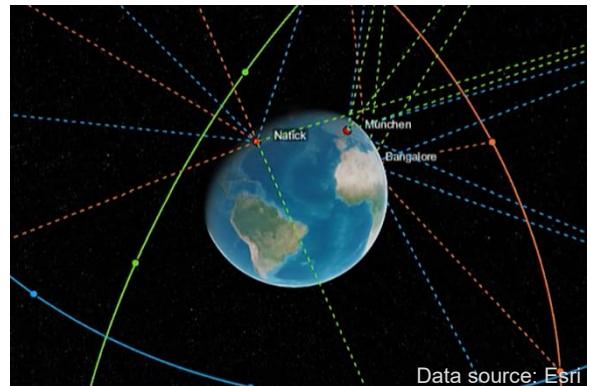
Renewable Power Generation Machines



Smart Medical Devices



Spacecraft



Wireless Networks

Image credit: from MathWorks.com

[TU Munich](#)

[Lawrence Livermore](#)

[Vanderhall](#)

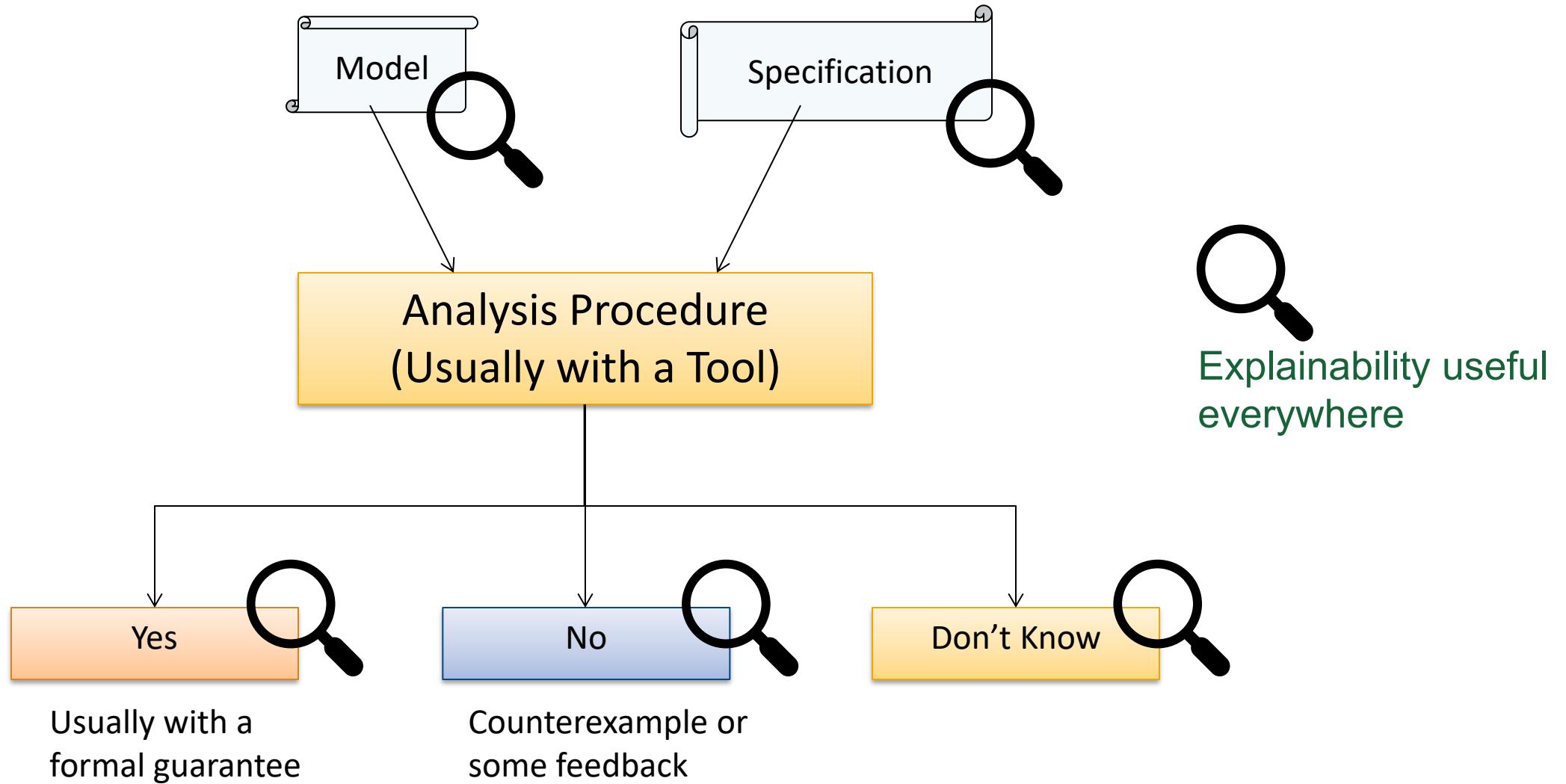
[Fighting Childhood Pneumonia](#)

[Datamaran](#)

[Lockheed Martin](#)

[Universal Robots](#)

[6G](#)



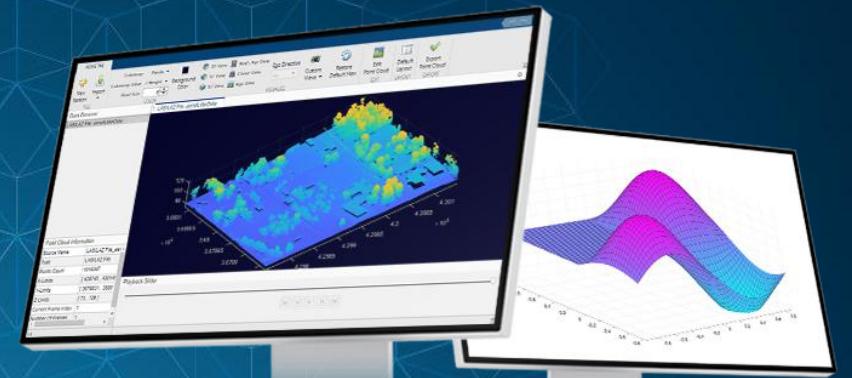
# MATLAB® & SIMULINK®

## Our Products

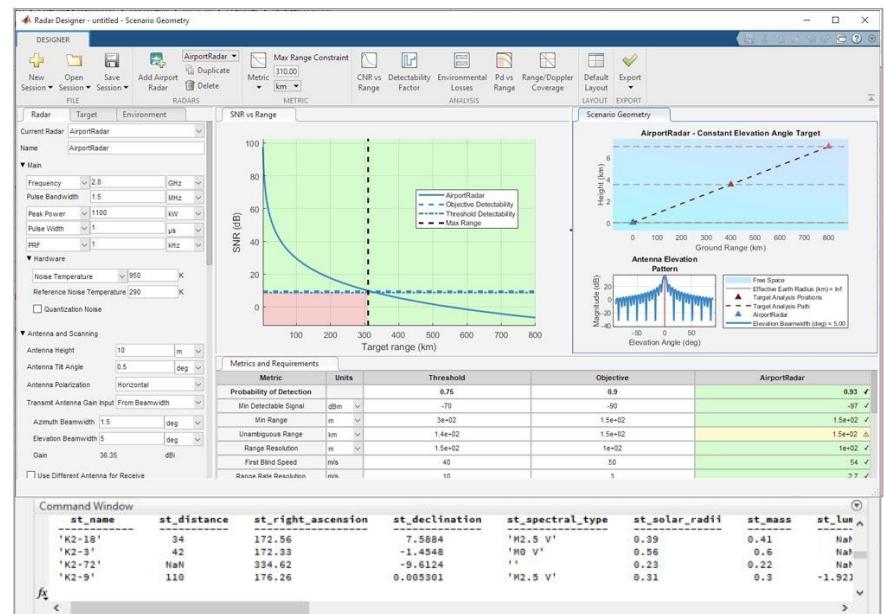
**MATLAB** – Programming environment for algorithm development, data analysis, visualization, and numeric computation.

**Simulink** – Block diagram environment for simulation and Model-Based Design of multidomain and embedded engineering systems.

130+ add-on products for specialized tasks.



Radar Toolbox



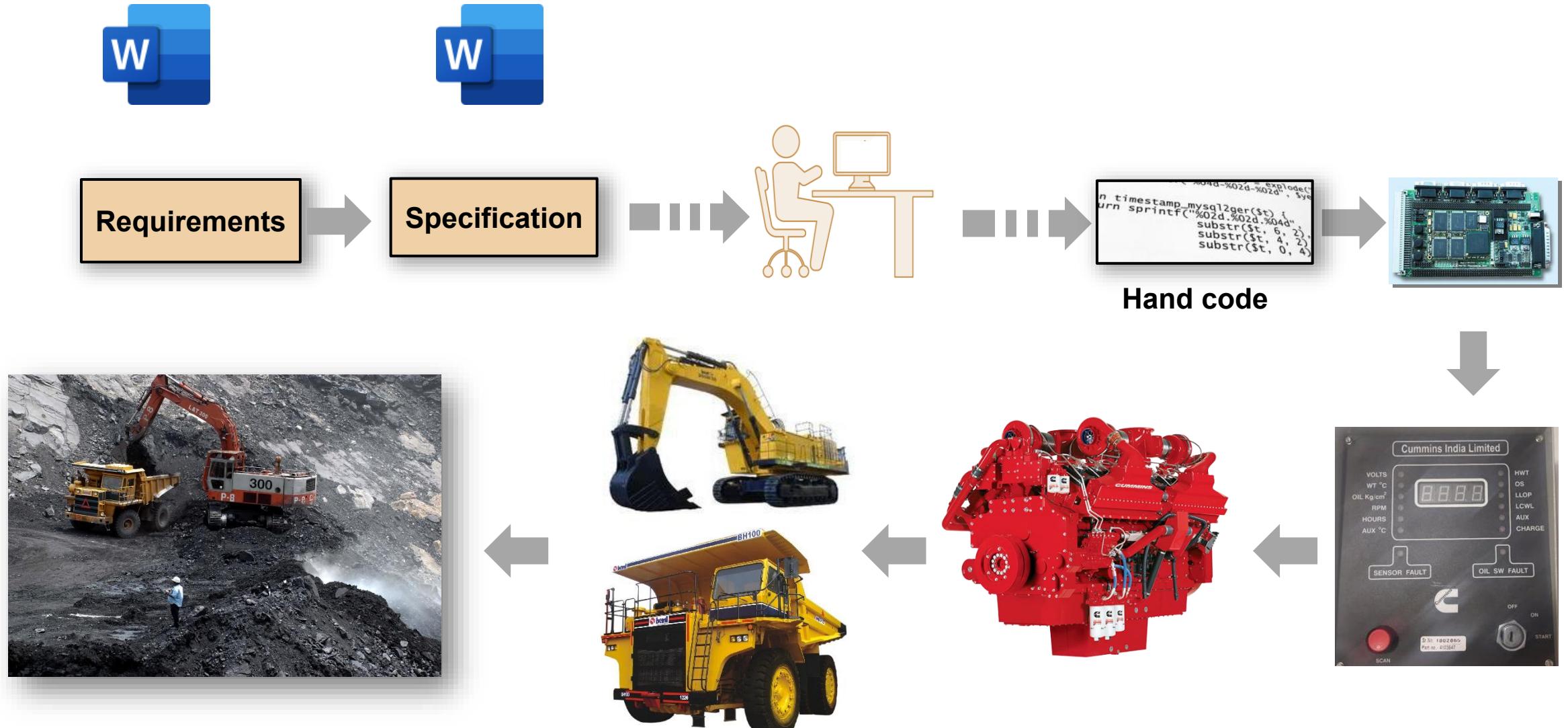
# Today's talk: Explainability in this tooling for designing complex engineered systems



# Outline

- Problem formulation
- Model-Based Design preliminaries
- Temporal Assessments as a case study for explainable feedback in tooling
  - Authoring time
  - Assessment time
- Advanced topic: Generative AI for explainability
- Wrap up

# Engineered System Design in my first job

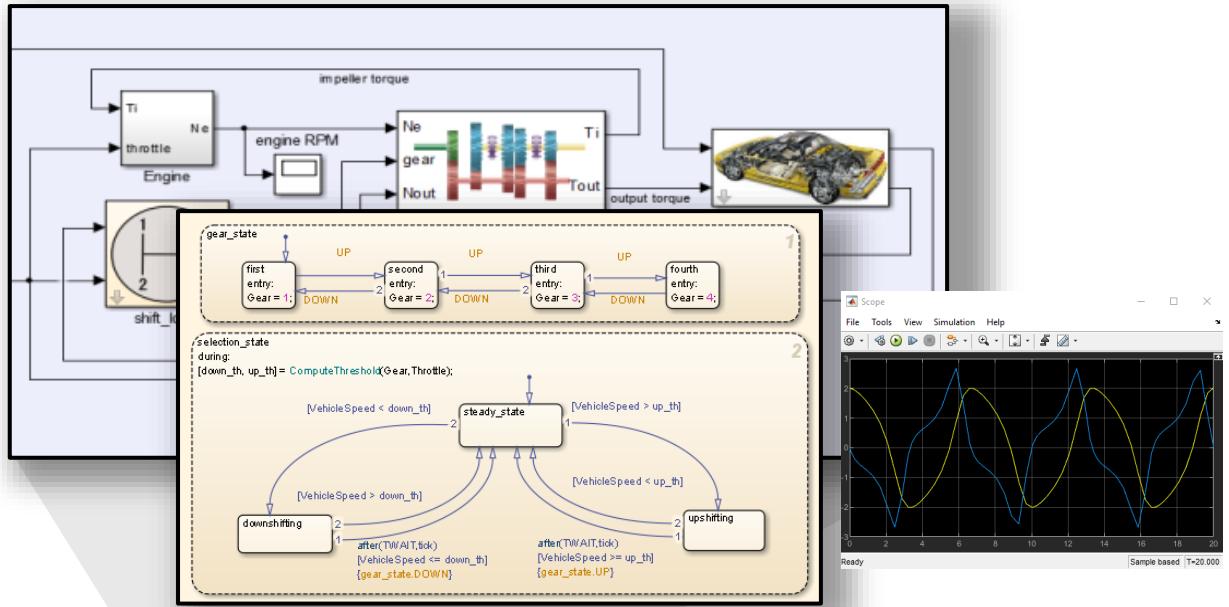


# Model-Based Design: 1) Model



Requirements

**Executable Specification**

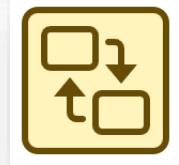
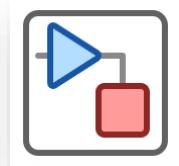


```
% timestamp_mysql2ger($t) = explode(" ", $t);
n_timestamp_mysql2ger($t) = sprintf("%02d.%02d-%04d-%02d-%02d", substr($t, 6, 2),
substr($t, 4, 2), substr($t, 0, 4))
```

Hand code



Tooling

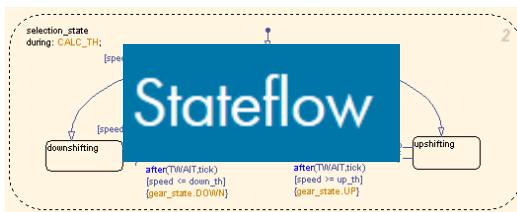
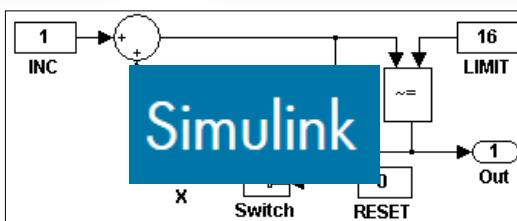


Can simulate, analyze, optimize on a computer!

# Model-Based Design: 2) “Based Design”

**Requirements**

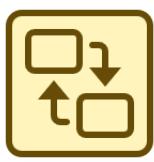
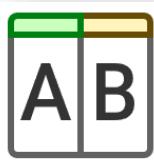
**Executable Specification**



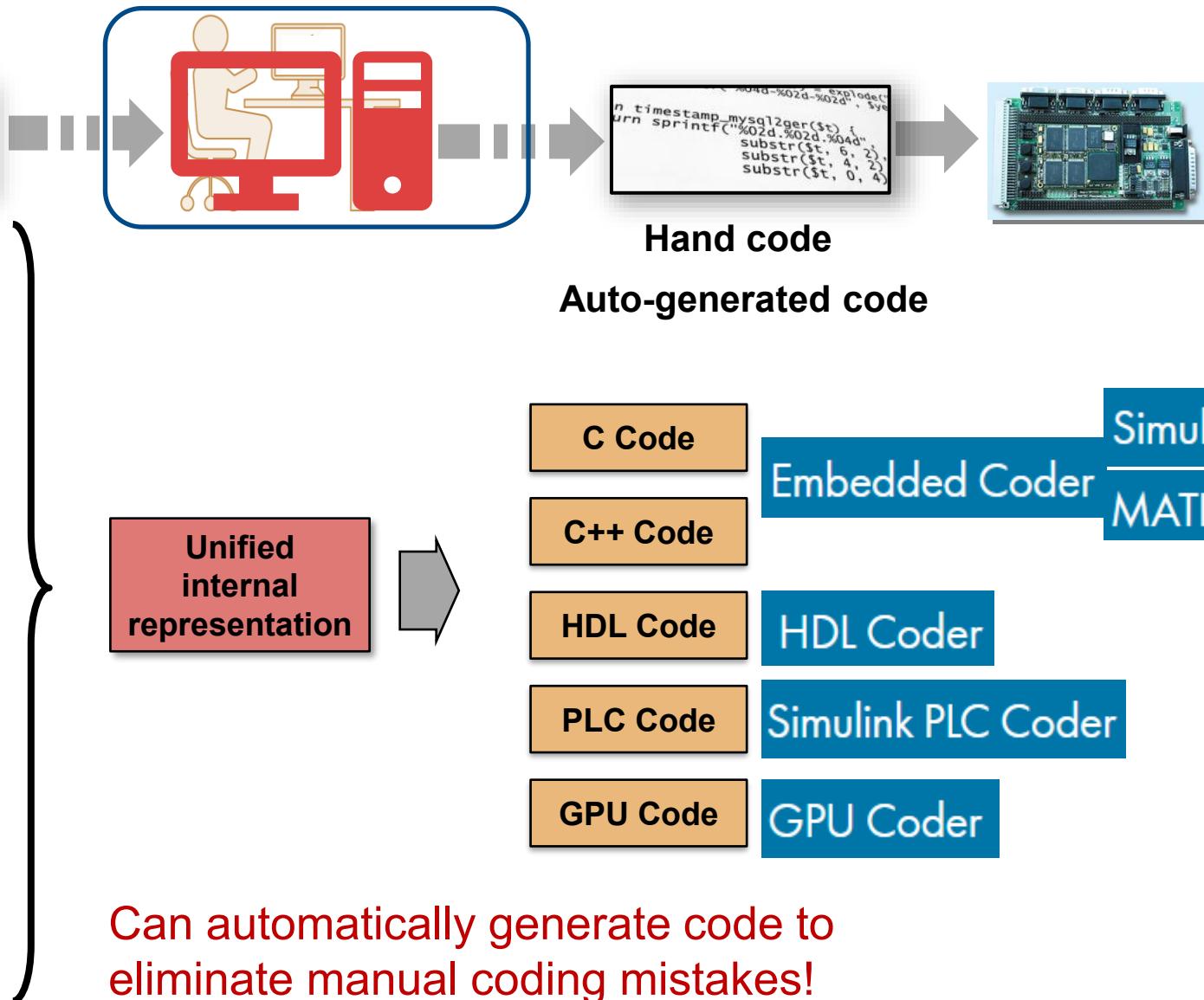
```
for n = 1:length(rxsig)
    u = rxsig(n); % received sample
    y = con;
    if n<=1
        d =;
    else
        d = imag(y));
    end
    % Single-tap RLS
    Delta = 1/(lambda/Delta + u*conj(u));

```

MATLAB code snippet showing a single-tap RLS filter implementation.

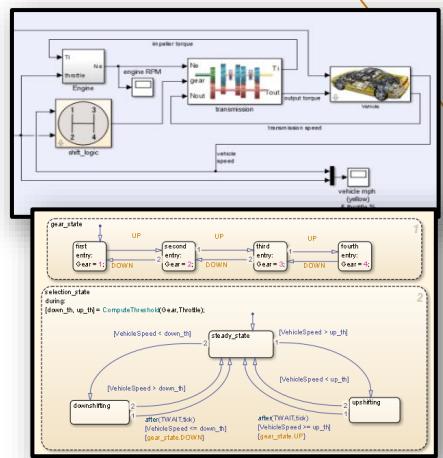


Tooling



Can automatically generate code to eliminate manual coding mistakes!

# Models are useful in both design and operation



Specification and design

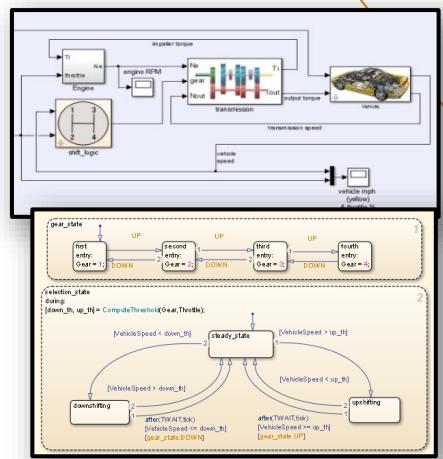


Integration and testing



*Challenges in the Operation and Design of Intelligent Cyber-Physical Systems*, S. Castro, P.J. Mosterman, A.H. Rajhans, and R.G. Valenti, book chapter, *Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy*, S. Mittal and A. Tolk, eds., Wiley, 2019.

# Models are useful in both design and operation



Specification and design



Integration and testing



Operation

*Challenges in the Operation and Design of Intelligent Cyber-Physical Systems*, S. Castro, P.J. Mosterman, A.H. Rajhans, and R.G. Valenti, book chapter, *Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy*, S. Mittal and A. Tolk, eds., Wiley, 2019.

# Models are useful in both design and operation

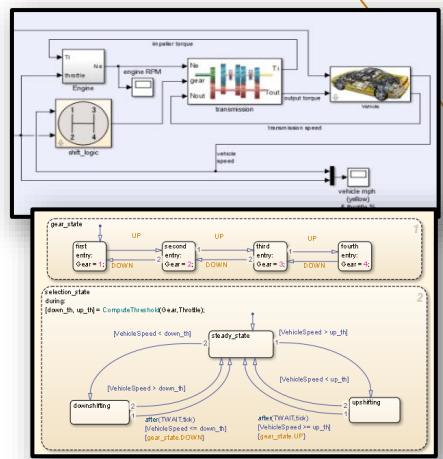


Online reconfiguration

Operation

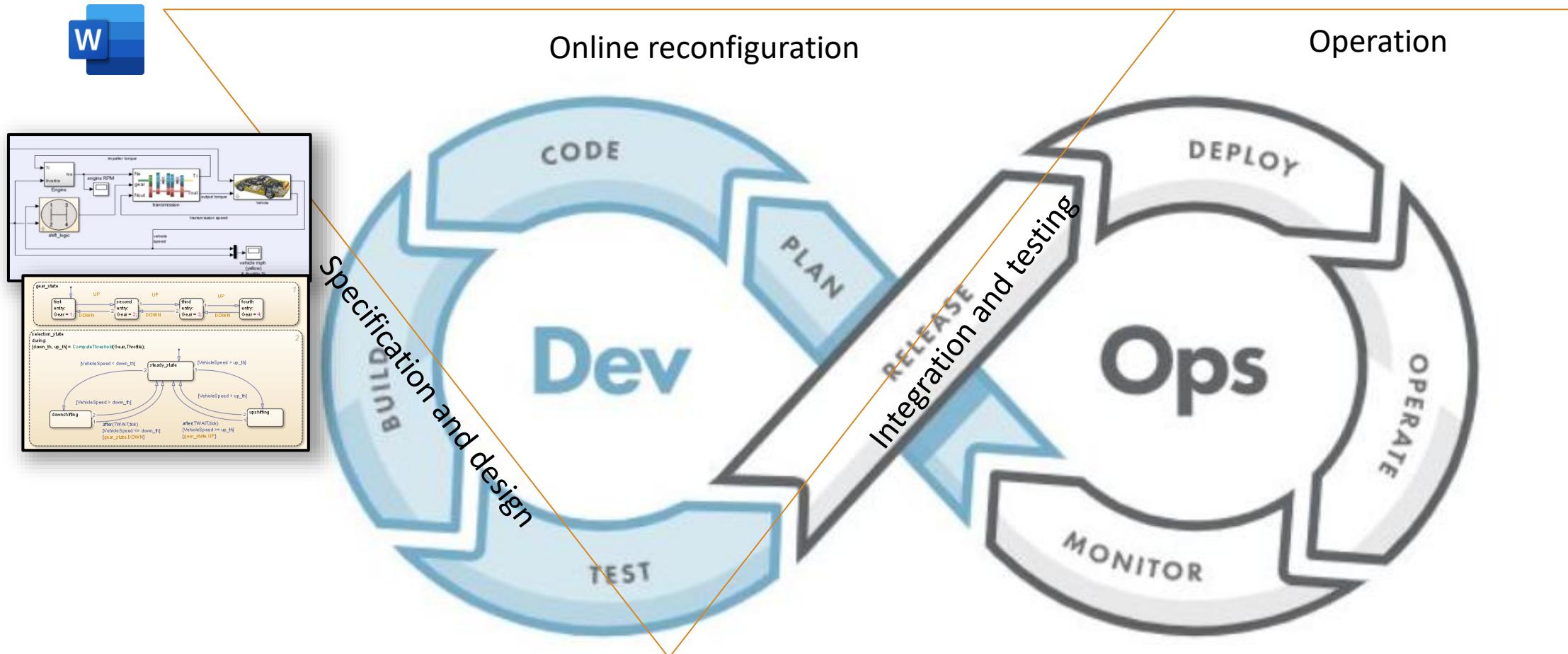
Specification and design

Integration and testing



*Challenges in the Operation and Design of Intelligent Cyber-Physical Systems*, S. Castro, P.J. Mosterman, A.H. Rajhans, and R.G. Valenti, book chapter, *Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy*, S. Mittal and A. Tolk, eds., Wiley, 2019.

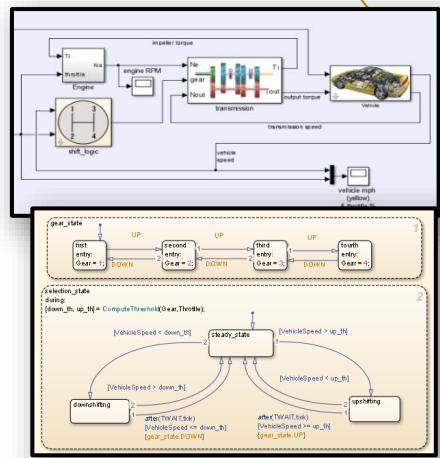
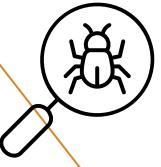
# Models are useful in both design and operation



*Challenges in the Operation and Design of Intelligent Cyber-Physical Systems*, S. Castro, P.J. Mosterman, A.H. Rajhans, and R.G. Valenti, book chapter, *Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy*, S. Mittal and A. Tolk, eds., Wiley, 2019.

Using models and specifications to build confidence in  
engineered system design

# Building confidence by finding mistakes early



Specification and design

Online reconfiguration

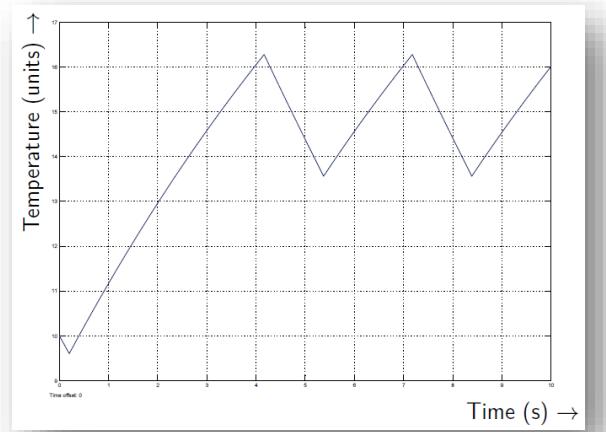
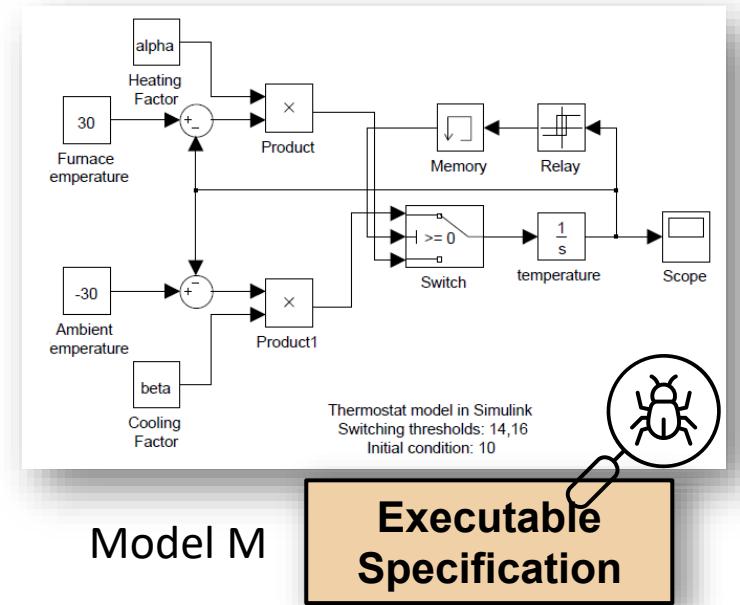


Integration and testing

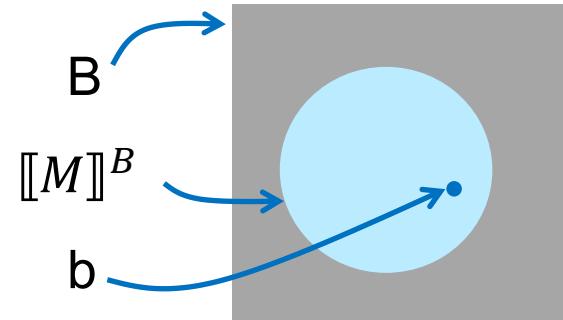


Operation

# Building confidence with models only



A behavior  $b$  that  $M$  exhibits



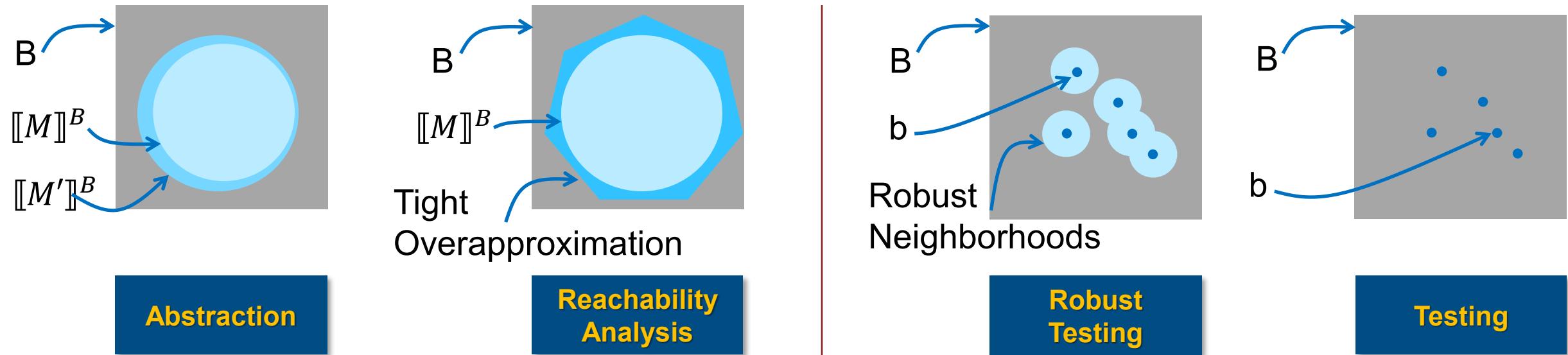
$\llbracket M \rrbracket^B$ : semantic interpretation of  $M$  in the behavior domain  $B$

Conceptually, set of all behaviors  $b$  that the model  $M$  exhibits in domain  $B$

In practice often infeasible if not impossible to figure out exactly

We don't know what properties hold for  $\llbracket M \rrbracket^B$  and therefore for  $M$

# Analysis problems with models only



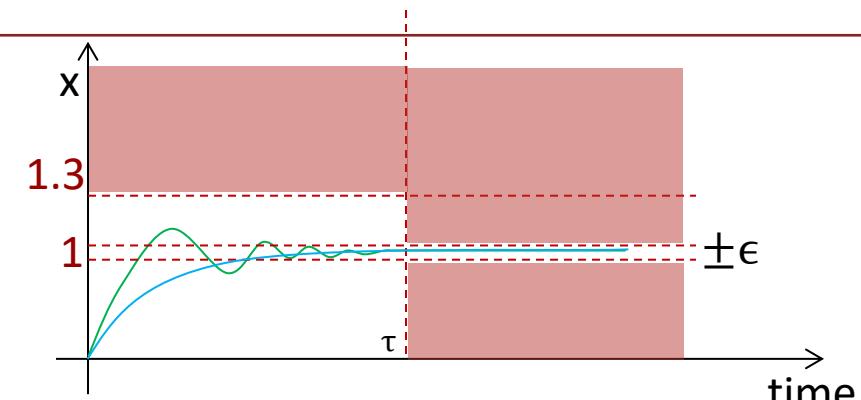
# Building confidence with specifications only

“overshoot is no more than 1.3  
and settling time is less than  $\tau$ ”

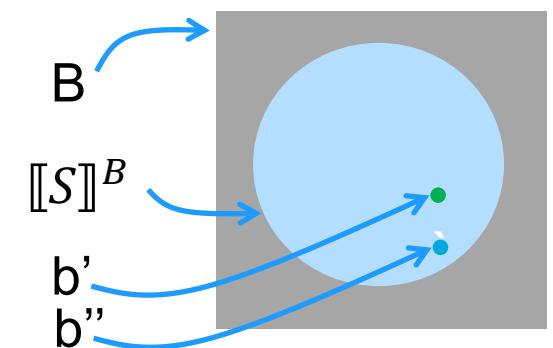


Specification S

Requirements

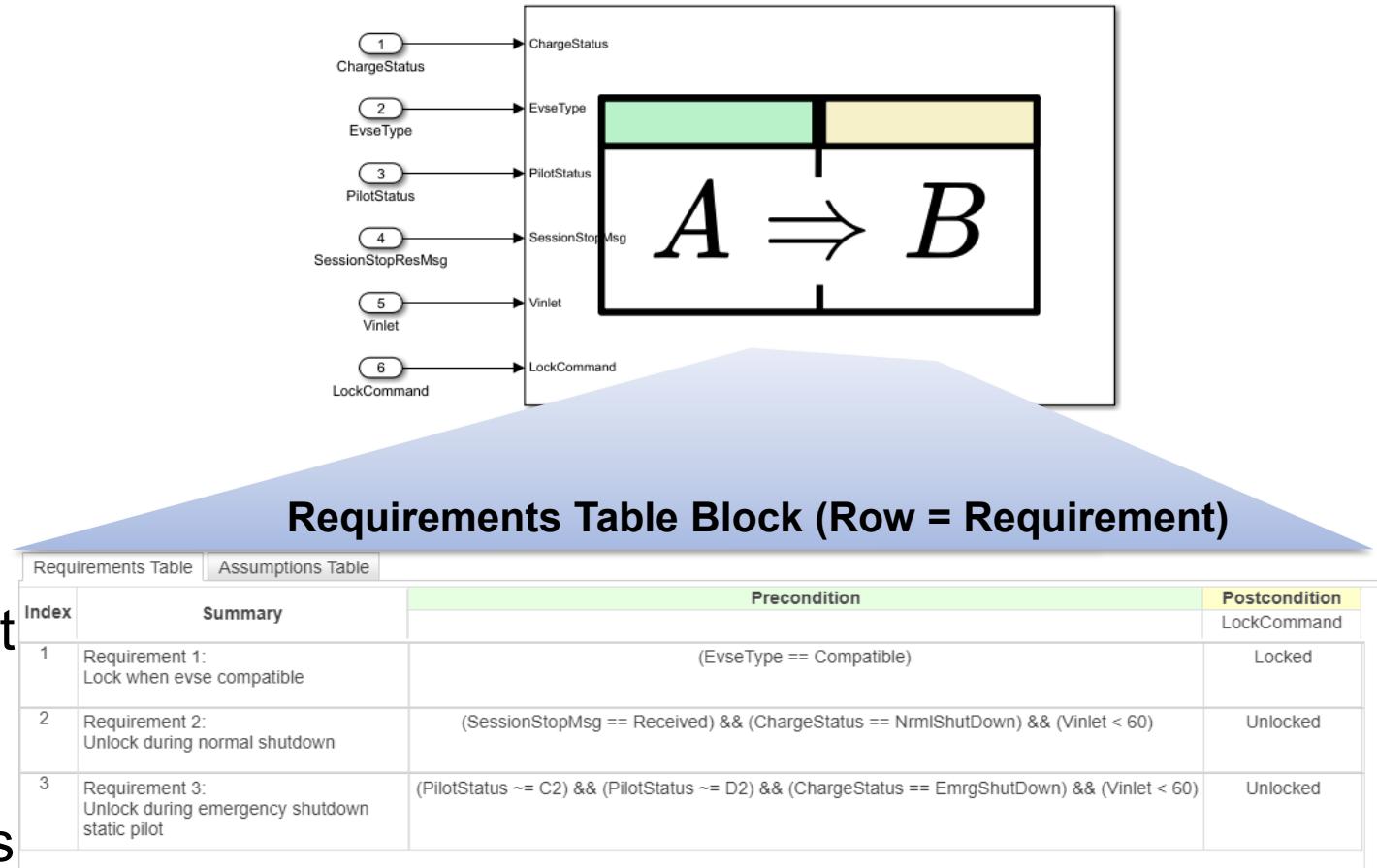


We know what properties hold for S  
S permits but doesn't prescribe  $b'$ ,  $b''$



# Requirements Modeling with Requirements Table Block

- **Mathematically rigorous**
- **Executable** through simulation
- **Easier to author and maintain** than text requirements
- **Easier to analyze** as requirement set grows
- **Reusable** for verification activities



# Analyze Requirements for Completeness and Consistency

- Automatically check for completeness and consistency issues using Simulink Design Verifier
- Debug within the table



Requirements		Assumptions	
		<b>Requirement 1:</b> Lock when evse compatible	
		Inconsistent with requirement 3 for inputs: Time   0 Step   1 ChargeStatus   ChrgStat.EmrgShutDown EvseType   EvseStat.Compatible PilotStatus   PilotStat.D1 SessionStopMsg   MsgStat.NotReceived Vinlet   0	
		<b>Requirement 2:</b> Unlock during normal shutdown	
		(SessionStopMsg == Received) && (ChargeStatus == NrmlShutDov	
		<b>Requirement 3:</b> Unlock during emergency shutdown static pilot	
		(PilotStatus ~= C2) && (PilotStatus ~= D2) && (ChargeStatus == En	

**Analysis Results**

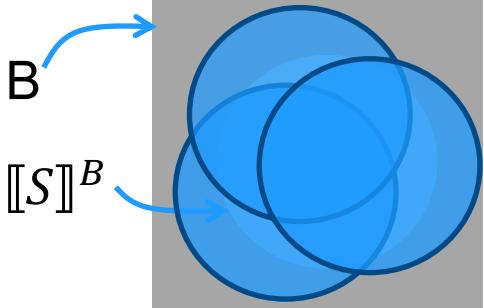
**Inconsistency Issues**  
 Inconsistency 1: 'LockCommand' is inconsistent at time 0 in requirements 1 and 3 for the following inputs:  

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:  

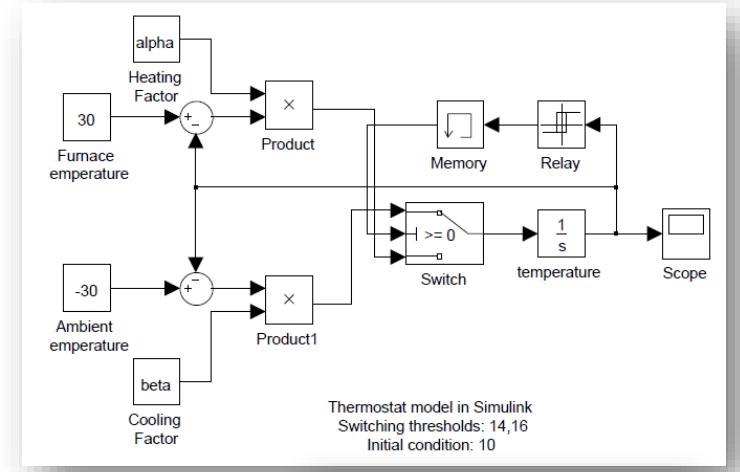
Time	0	0.2
Step	1	2
ChargeStatus	ChrgStat.ChargeStart	ChrgStat.Not
EvseType	EvseStat.Compatible	EvseStat.Not

# Analysis problems with specifications only



Consistency,  
completeness

# Building confidence with models and specifications



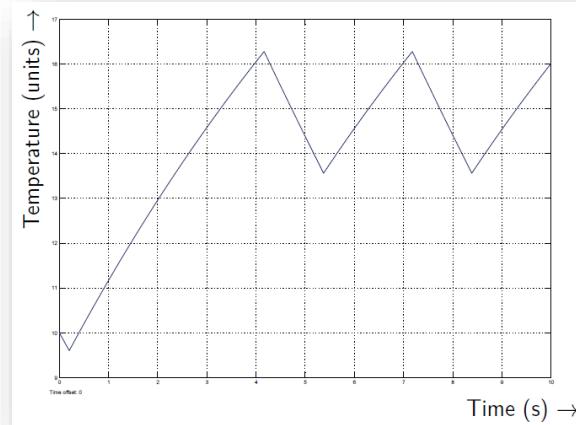
Model M

**Executable Specification**

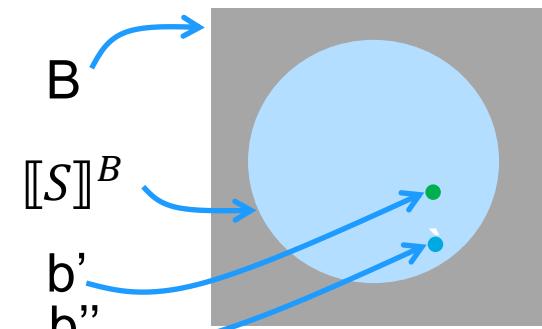
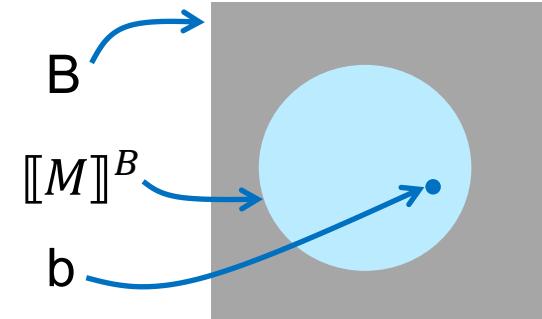
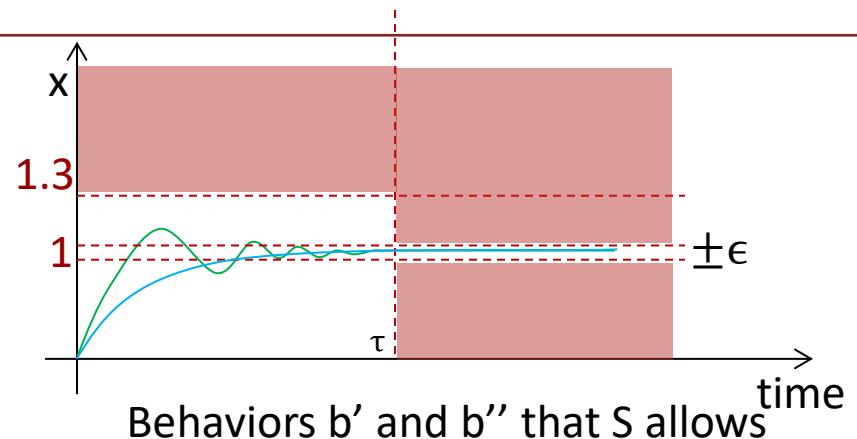
“overshoot is no more than 1.3 and settling time is less than  $\tau$ ”

**Requirements**

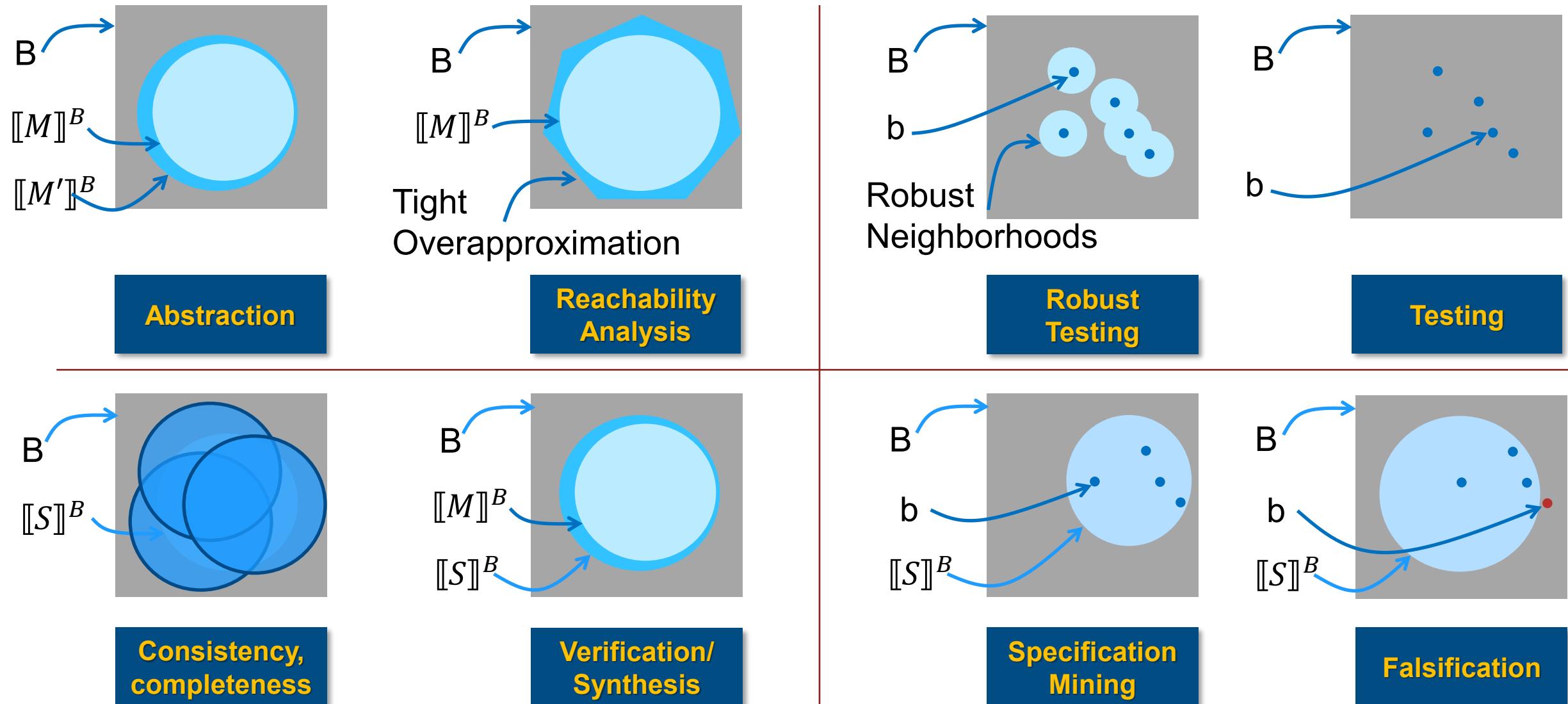
Specification S



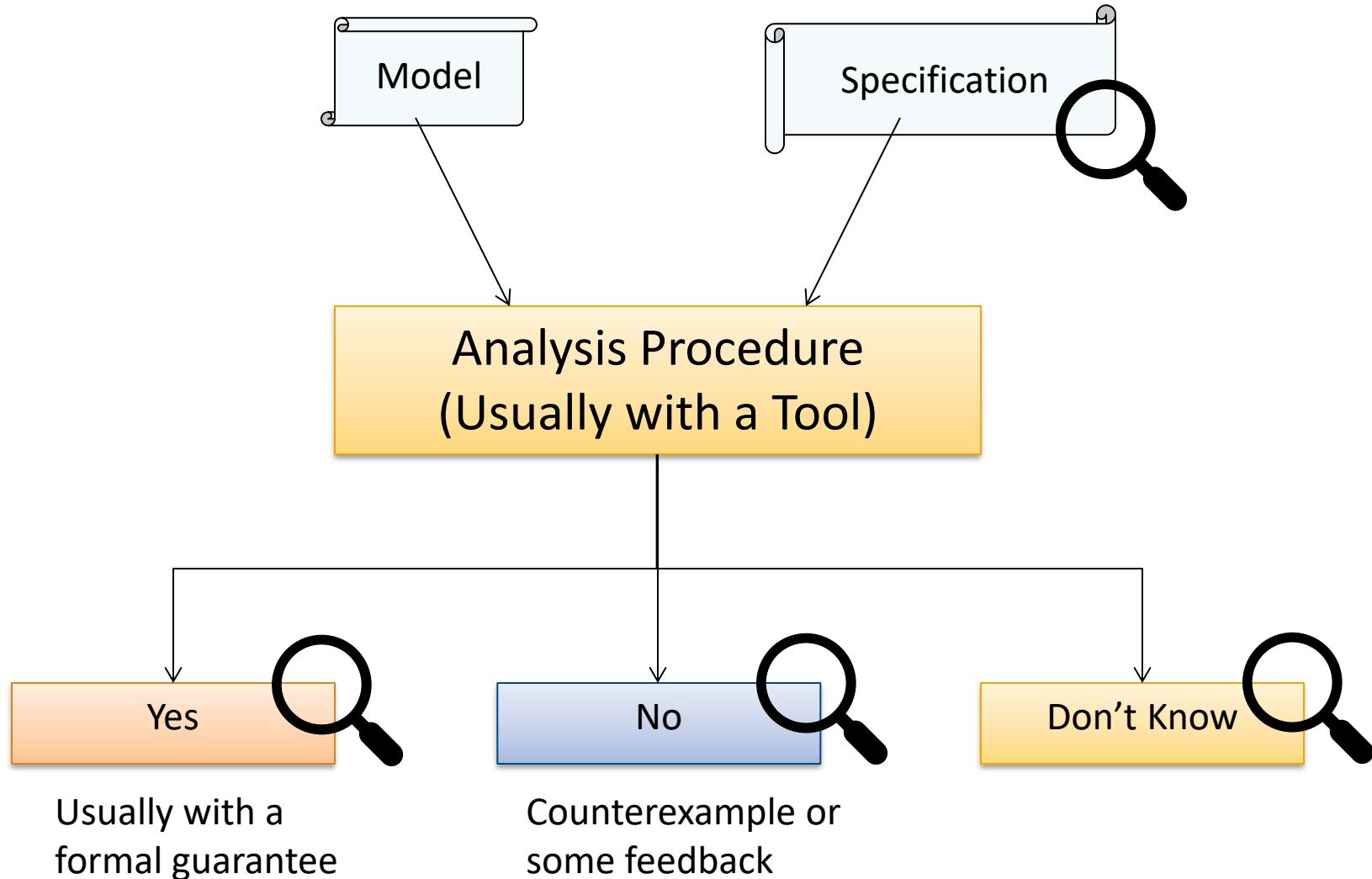
A behavior b that M exhibits



# Analysis problems with models and specifications



# The promise of formal methods



# Temporal logics as formal specifications have uncovered mistakes!

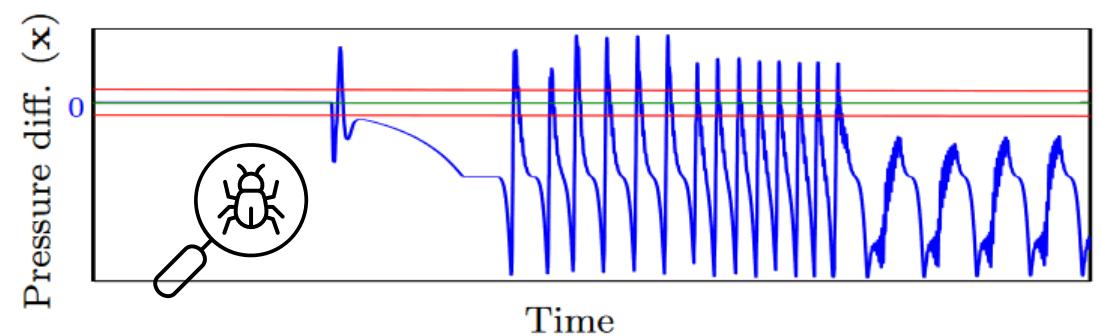
## Formal Specifications

Linear Temporal Logic, e.g.,  
 $\square(a \Rightarrow \diamond b)$

Metric-time Temporal Logic, e.g.,  
 $\square(a \Rightarrow \diamond_{[0,0.5s]} b)$

Signal Temporal Logic, e.g.,  
 $\square(x(t) > 1 \Rightarrow \diamond_{[0,0.5s]} y(t) < 0)$

- Success in the research community



Mining Requirements From Closed-Loop Control Models  
Jin, Donze', Deshmukh, Seshia, IEEE Trans. of CAD of Integrated Circuits and Systems 2015

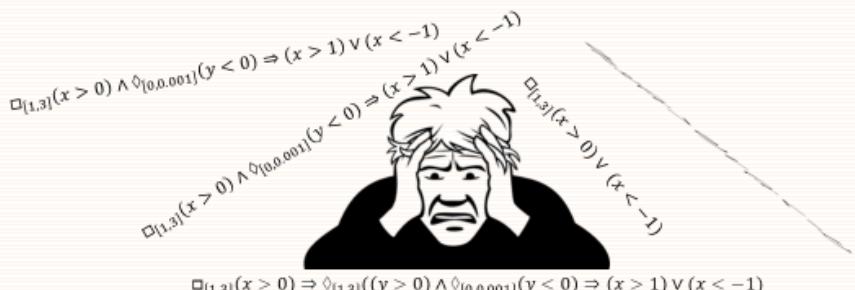
- Research tool support
  - Breach, S-TaLiRo, AMT 2.0, Vispec, ...

# But they are also hard to understand for practitioners

► Automotive Industry Trends ► MBD Verification ► Techniques ► Challenges

## Grand Challenge I: Requirement Engineering

- ▶ Key challenge for Toyota, Bosch, and others
  - How do you present requirements to control designers?
  - How do they convey their intention without using formalisms?
  - Is Temporal Logic the right requirement language?



HSCC 2015, Seattle

55/63



**Jyo Deshmukh (Toyota)**  
HSCC 2015 Keynote

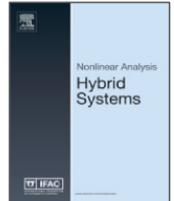
Nonlinear Analysis: Hybrid Systems 34 (2019) 147–165



Contents lists available at ScienceDirect

## Nonlinear Analysis: Hybrid Systems

journal homepage: [www.elsevier.com/locate/nahs](http://www.elsevier.com/locate/nahs)



Position paper on the challenges posed by modern applications to cyber-physical systems theory<sup>☆</sup>

Frank Allgöwer <sup>a</sup>, João Borges de Sousa <sup>b</sup>, James Kapinski <sup>c</sup>, Pieter Mosterman <sup>d</sup>, Jens Oehlerking <sup>e</sup>, Patrick Panciatici <sup>f</sup>, Maria Prandini <sup>g,\*</sup>, Akshay Rajhans <sup>d</sup>, Paulo Tabuada <sup>h</sup>, Philipp Wenzelburger <sup>a</sup>

<sup>a</sup> University of Stuttgart, Germany

<sup>b</sup> Universidade do Porto, Portugal

<sup>c</sup> Toyota Research Institute of North America, United States

<sup>d</sup> MathWorks, United States

<sup>e</sup> Robert Bosch GmbH, Germany

<sup>f</sup> RTE, France

<sup>g</sup> Politecnico di Milano, Italy

<sup>h</sup> University of California at Los Angeles, United States



**James Kapinski (Toyota),  
Jens Oehlerking (Bosch)**  
HSCC-ICCPs 2018 Panel



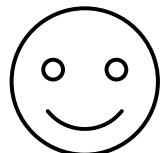
# Natural language

vs.

# formal specifications

English sentences

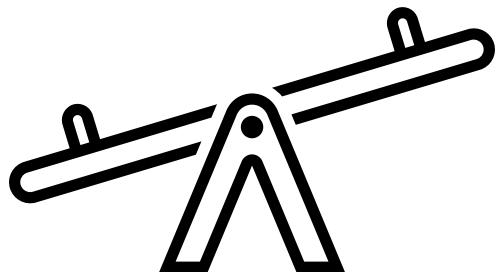
“whenever oil pressure becomes too low or coolant temperature becomes too high, engine shall shut down within 0.5s”



Intuitive



Informal and ambiguous



Temporal logic formulas

$$\begin{aligned} & \square((p(t - \tau) \geq lb) \wedge (p(t) < lb)) \\ & \vee ((T(t - \tau) \leq ub) \wedge (T(t) > ub)) \\ & \Rightarrow \diamond_{[0,0.5s]} v \leq 0 \end{aligned}$$


Formal and precise



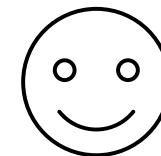
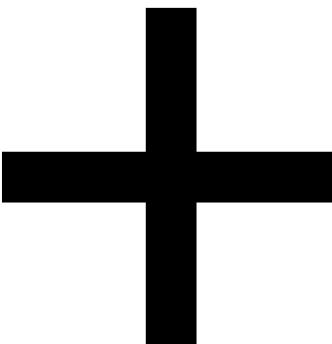
Not intuitive to practitioners

Is there a way to get the best of both?

# Natural language and formal specifications



Intuitive



Formal and precise

## ASSESSMENT

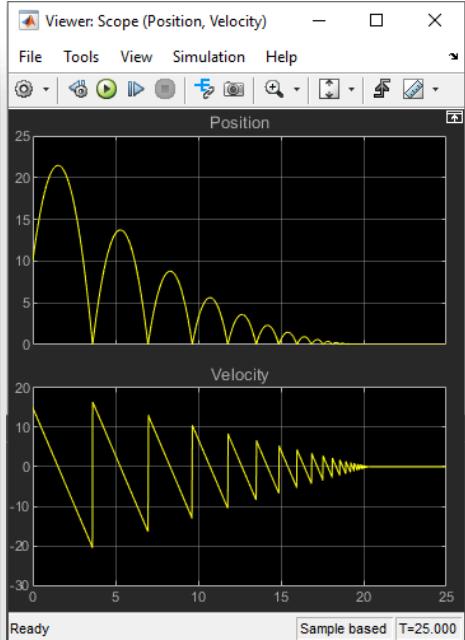
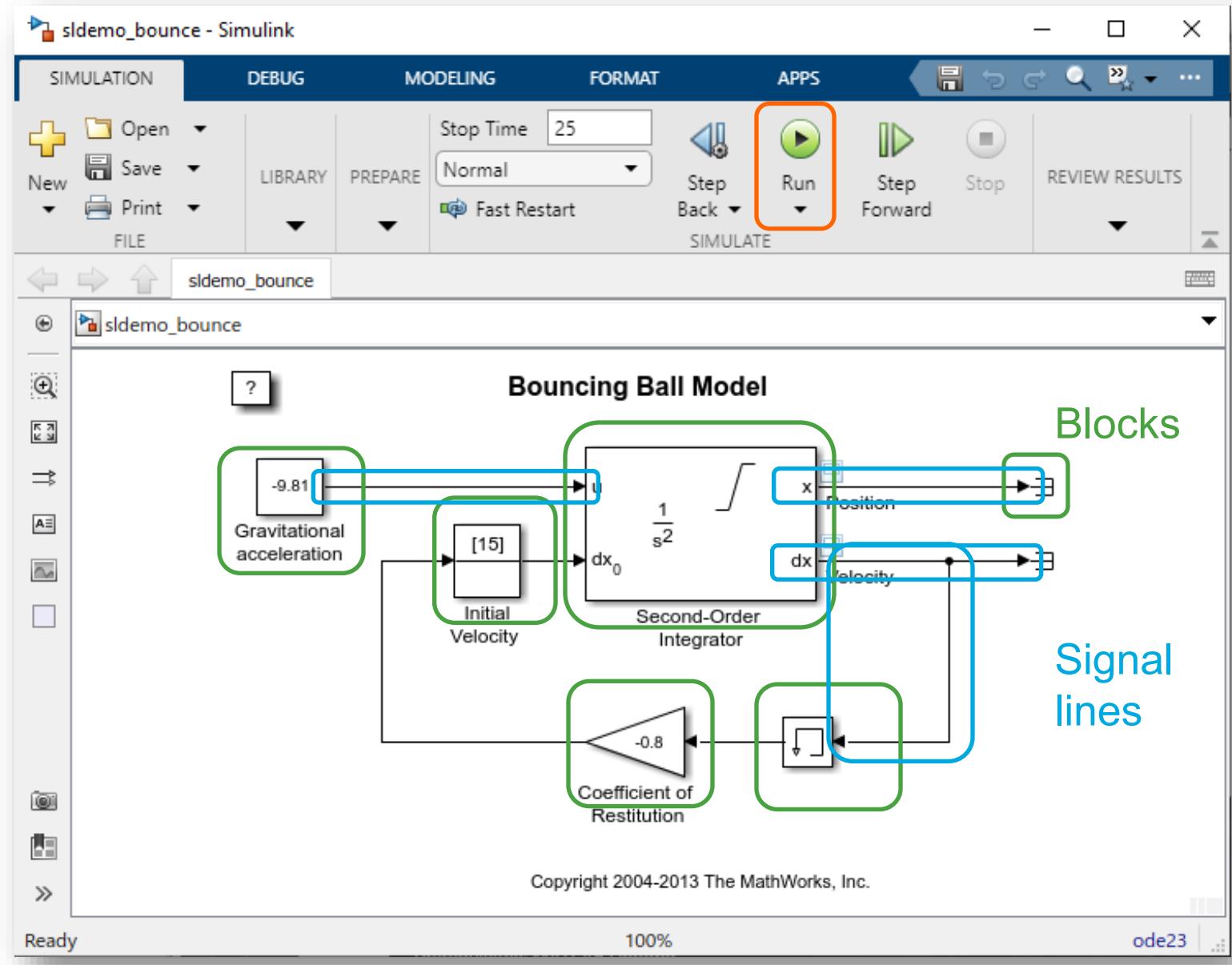
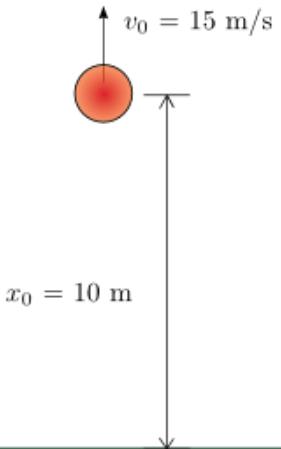
- ▶ At any point of time, position must be greater than or equal to 0
- ▶ At any point of time, if `position == 0` becomes true then, with a delay of at most `eps1` seconds, `velocity > 0` must stay true for at least `tau` seconds

## Logical and Temporal Assessments in Simulink® Test™

<https://www.mathworks.com/help/sltest/ug/temporal-assessments.html>

# Preliminaries

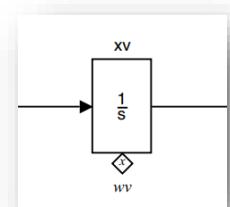
# Simulink preliminaries



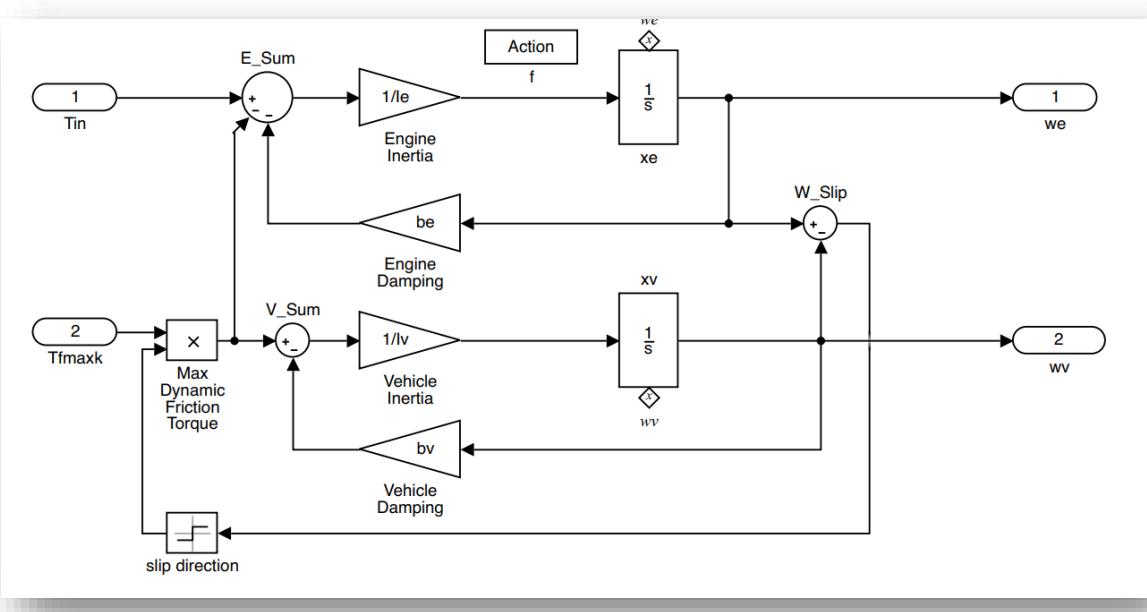
# Formal definition of a Simulink Block diagram

**Definition 2.1. Simulink Block.** A Simulink *block* is a tuple  $(x_{CT}, x_{DT}, f_{CT}, f_{DT}, g, u, y, x_0)$ , where:

- $x_{CT}$  and  $x_{DT}$  are (possibly-empty) continuous-time (CT) and discrete-time (DT) state vectors, together addressed as the continuous-valued state vector  $x$ ;
- $u$  and  $y$  are the input and output vectors;
- $f_{CT}$  is the *derivative function*, i.e.,  $\dot{x}_{CT}(t) = f_{CT}(u(t), x(t))$ , numerically integrated by an ODE solver at appropriate time points  $t$  based on the solver characteristics (e.g., number of internal states and error tolerances);
- $f_{DT}$  is the *update function*, i.e.,  $x_{DT}(t+dt) = f_{DT}(u(t), x(t))$ , with stepsize  $dt$  determined by the (specified or inferred) sample rate of the block;
- $g$  is the *output function*, i.e.,  $y(t) = g(x(t), u(t))$ ; and
- $x_0$  is the initial value of the state vector  $x$ , i.e.,  $x(0) = x_0$ , which can be either defined as a block parameter or read as an external input.



**Definition 2.2. Block Diagram.** A Simulink *block diagram* is a (hyper-)graph  $(N_{SL}, E_{SL})$ , with Simulink blocks as nodes  $N_{SL}$ , and signal or communication lines representing the connectivity constraints as edges  $E_{SL}$ .



# Basics of simulation semantics

- Simulink has its origins in CT/DT control systems theory
- It was designed to solve the following system equations

## Discrete-Time System

$$y[k] = g(x[k], u[k])$$

$$x[k+1] = f(x[k], u[k])$$

$$x[0] = x_0$$

*Output Equation*

*State Equation*

*Initial Condition*

## Continuous-Time System

$$y(t) = g(x(t), u(t))$$

$$\dot{x}(t) = f(x(t), u(t))$$

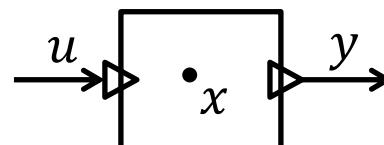
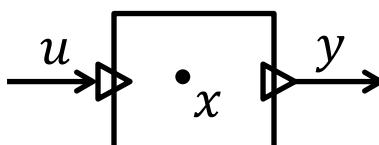
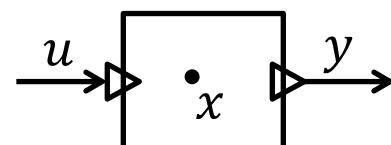
$$x(0) = x_0$$

where

$u$  – input

$y$  – output

$x$  – state



- Every block is a system, model is a composition

# Simulating a model using numerical Integration



'Solve' the difference/differential equation using numerical methods

## Discrete-Time System

$$y[k] = g(x[k], u[k])$$

$$x[k+1] = f(x[k], u[k])$$

$$x[0] = x_0$$

*Output Equation*

*State Equation*

*Initial Condition*

## Continuous-Time System

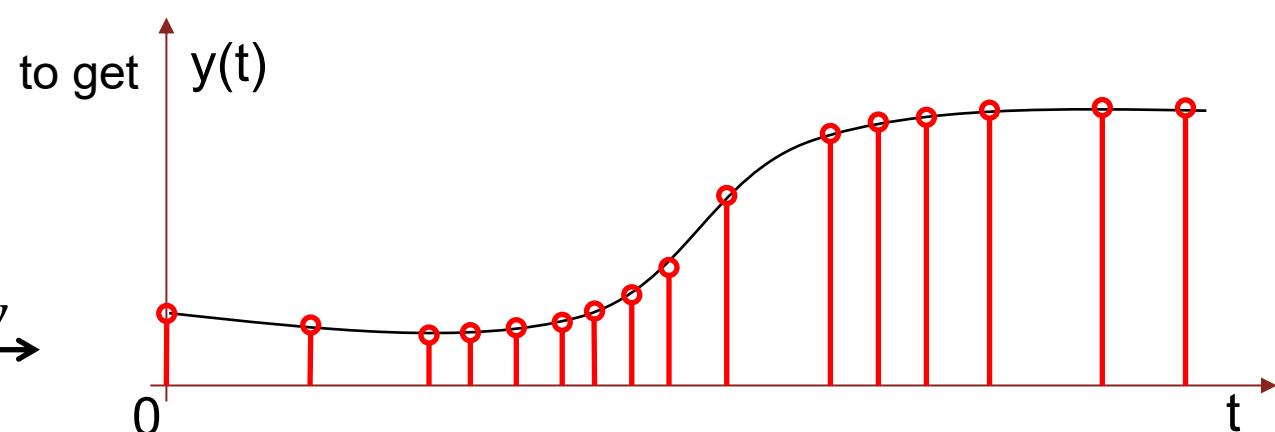
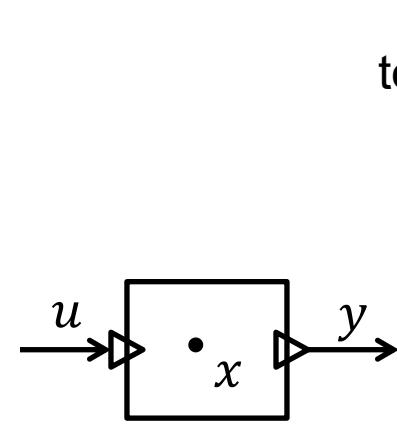
$$y(t) = g(x(t), u(t))$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$x(0) = x_0$$

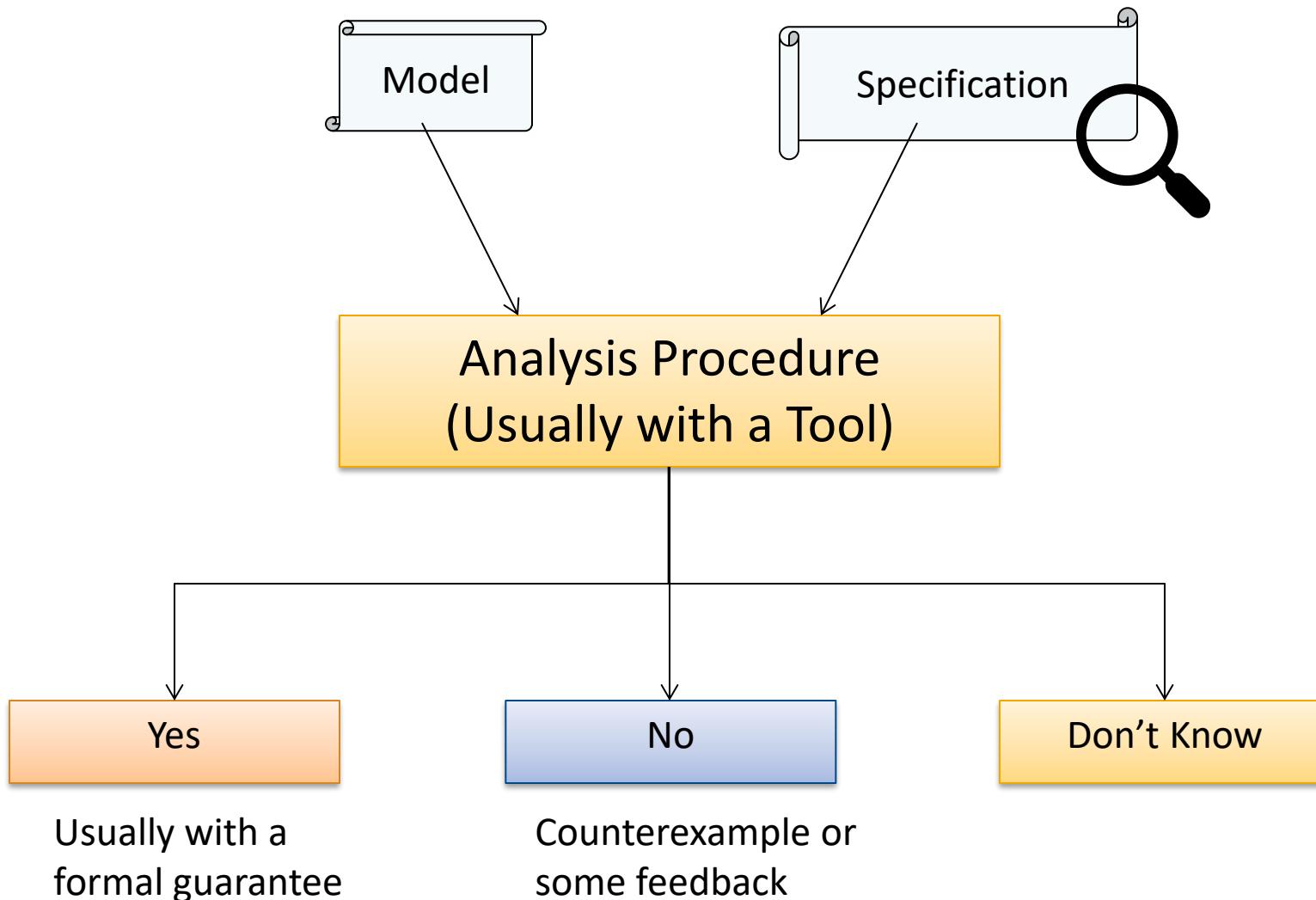
Call block methods in a loop

```
//initialize state
Initialize
//compute output
Output
//compute state at next time instance
Update/ Derivatives
```



# Temporal Assessments

# Authoring



# Authoring

>> sltestmgr

The screenshot shows the MATLAB Test Manager interface. At the top, there are two tabs: "Start Page" and "New Test Case 1". The "New Test Case 1" tab is active, indicated by a blue border. Below the tabs is a toolbar with buttons for "EN...", "NAME", "ASSESSMENT", and "REQUIREM...". A plus sign button (+) is also present.

**Logical Assessments**

- Bounds check**  
Check min/max bounds for signals and expressions
- Custom**  
Check if a custom expression holds true for all time steps

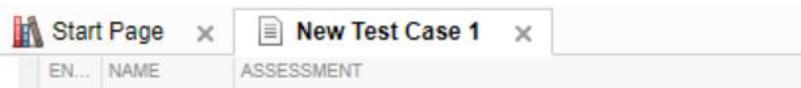
**Temporal Assessments**

- Trigger-response**  
Check for a signal response once a trigger has been detected

At the bottom of the sidebar, there are two buttons: "Add Assessment" with a plus sign and "Delete" with a trash can icon.

# Authoring

>> sltestmgr



**Logical Assessments**

- Bounds check**  
Check min/max bounds for signals and expressions
- Custom**  
Check if a custom expression holds true for all time steps

**Temporal Assessments**

- Trigger-response**  
Check for a signal response once a trigger has been detected

**Add Assessment** **Delete**

Table 1: Template Pattern Classes

Template Pattern Class	Equivalent Logical Formula
Bounds Check	<input type="checkbox"/> (signal satisfies bound constraint)
Custom Formula	<input type="checkbox"/> $\varphi$
Trigger Response	<input type="checkbox"/> $(\varphi_1 \rightarrow \Diamond \varphi_2)$

# Authoring

>> c1testman

LOGICAL AND TEMPORAL ASSESSMENTS\*

EN...	NAME	ASSESSMENT
<input checked="" type="checkbox"/>	Assessment1	▼ At any point of time ... bounds-check-pattern: always inside bounds signal: sig lower-bound: lb lower-bound-type: greater than upper-bound: ub upper-bound-type: less than

+ Add Assessment    Delete

Trigger-response  
Check for a signal response once a trigger occurs.

+ Add Assessment    Delete

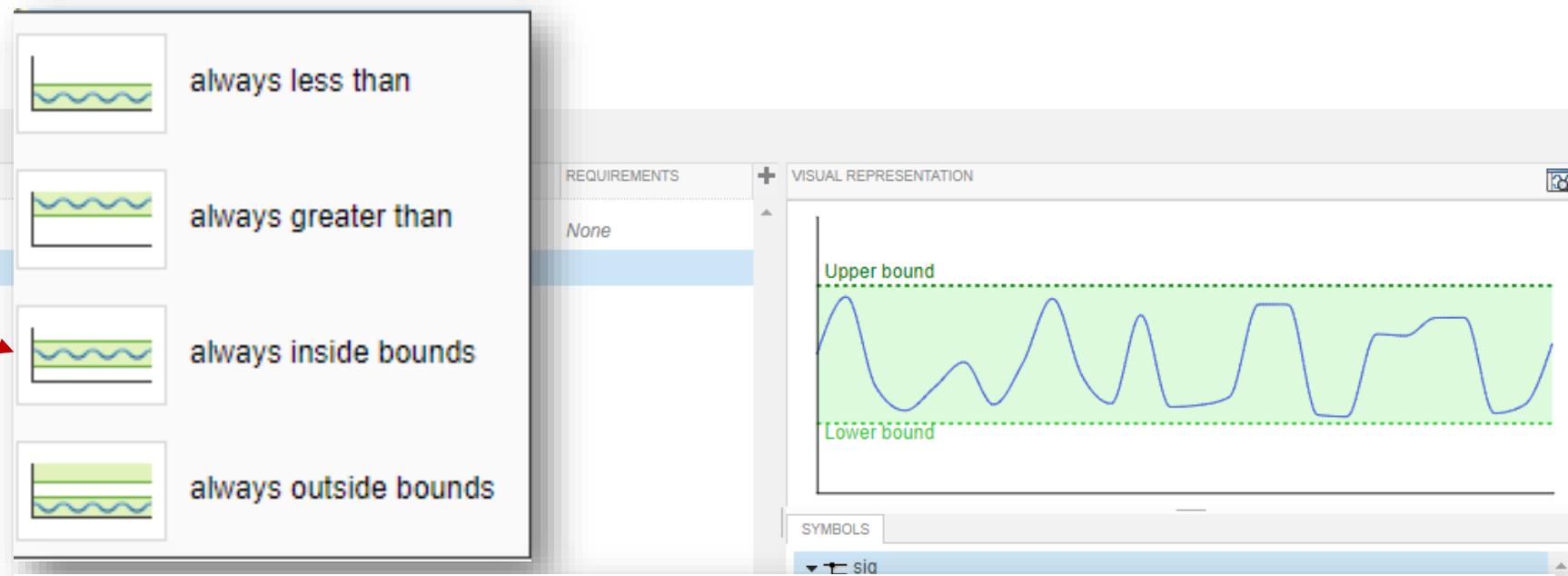


Table 2: Bounds check patterns and variations.

Pattern	Equivalent Logical Formula	Strict Variation(s)
Always less than	<input type="checkbox"/> $(x < \text{ub})$	<input type="checkbox"/> $(x \leq \text{ub})$
Always greater than	<input type="checkbox"/> $(x > \text{lb})$	<input type="checkbox"/> $(x \geq \text{lb})$
Always inside bounds	<input type="checkbox"/> $((x < \text{ub}) \wedge (x > \text{lb}))$	<input type="checkbox"/> $((x \leq \text{ub}) \wedge (x > \text{lb}))$ <input type="checkbox"/> $((x < \text{ub}) \wedge (x \geq \text{lb}))$ <input type="checkbox"/> $((x \leq \text{ub}) \wedge (x \geq \text{lb}))$
Always outside bounds	<input type="checkbox"/> $((x > \text{ub}) \vee (x < \text{lb}))$	<input type="checkbox"/> $((x \geq \text{ub}) \vee (x < \text{lb}))$ <input type="checkbox"/> $((x > \text{ub}) \vee (x \leq \text{lb}))$ <input type="checkbox"/> $((x \geq \text{ub}) \vee (x \leq \text{lb}))$

# Authoring

**>> c1testman**

LOGICAL AND TEMPORAL ASSESSMENTS\*

EN...	NAME	ASSESSMENT	REQUIREMENTS	+	VISUAL REPRESENTATION
<input checked="" type="checkbox"/>	Assessment1	<ul style="list-style-type: none"> <li>▼ At any point of time ...</li> <li>▶ bounds-check-pattern: always inside bounds</li> </ul> <p>signal: <b>sig</b> lower-bound: <b>lb</b> lower-bound-type: greater than upper-bound: <b>ub</b> upper-bound-type: less than</p>	None		
		<ul style="list-style-type: none"> <li>▼ At any point of time ...</li> <li>▶ bounds-check-pattern: <b>sig must be greater than lb and less than ub</b></li> </ul>			
		<ul style="list-style-type: none"> <li>► At any point of time, <b>sig must be greater than lb and less than ub</b></li> </ul>			
<b>+ Add Assessment</b>	<b>Delete</b>	<b>Trigger-response</b> Check for a signal response once a trigger has been detected			
<b>+ Add Assessment</b>	<b>Delete</b>	<b>Pattern explorer</b> <input type="checkbox"/> New Example <input type="checkbox"/> Passing Example <input checked="" type="checkbox"/> Dynamic bounds			
<b>+ Add Assessment</b>	<b>Delete</b>	<b>Pattern explorer</b> <input type="checkbox"/> New Example <input type="checkbox"/> Failing Example <input checked="" type="checkbox"/> Dynamic bounds			
<b>+ Add Assessment</b>	<b>Delete</b>	<b>Pattern explorer</b> <input type="checkbox"/> New Example <input type="checkbox"/> Passing Example <input checked="" type="checkbox"/> Dynamic bounds			
<b>+ Add Assessment</b>	<b>Delete</b>	<b>Pattern explorer</b> <input type="checkbox"/> New Example <input type="checkbox"/> Failing Example <input checked="" type="checkbox"/> Dynamic bounds			

# Authoring

>> sltestmgr

Start Page New Test Case 1

EN...	NAME	ASSESSMENT	REQUIREM...
			<b>+</b>

**Logical Assessments**

- Bounds check**  
Check min/max bounds for signals and expressions
- Custom**  
Check if a custom expression holds true for all time steps

**Temporal Assessments**

- Trigger-response**  
Check for a signal response once a trigger has been detected

**Add Assessment** Delete

$$\square_{[t_0, t_f]} \phi$$

# Authoring

>> sltestmgr

The screenshot shows the MATLAB Test Manager interface. At the top, there are two tabs: "Start Page" and "New Test Case 1". The "New Test Case 1" tab is active. Below the tabs is a table with columns: EN..., NAME, ASSESSMENT, and REQUIREMENT. A row in the table has a checked checkbox under "ASSESSMENT" and the text "At any point of time ...". Under "ASSESSMENT", there are three sections: "trigger: <empty>", "delay: with no delay ...", and "response: <empty>".

**Logical Assessments**

- Bounds check**  
Check min/max bounds for signals and expressions
- Custom**  
Check if a custom expression holds true for all time steps

**Temporal Assessments**

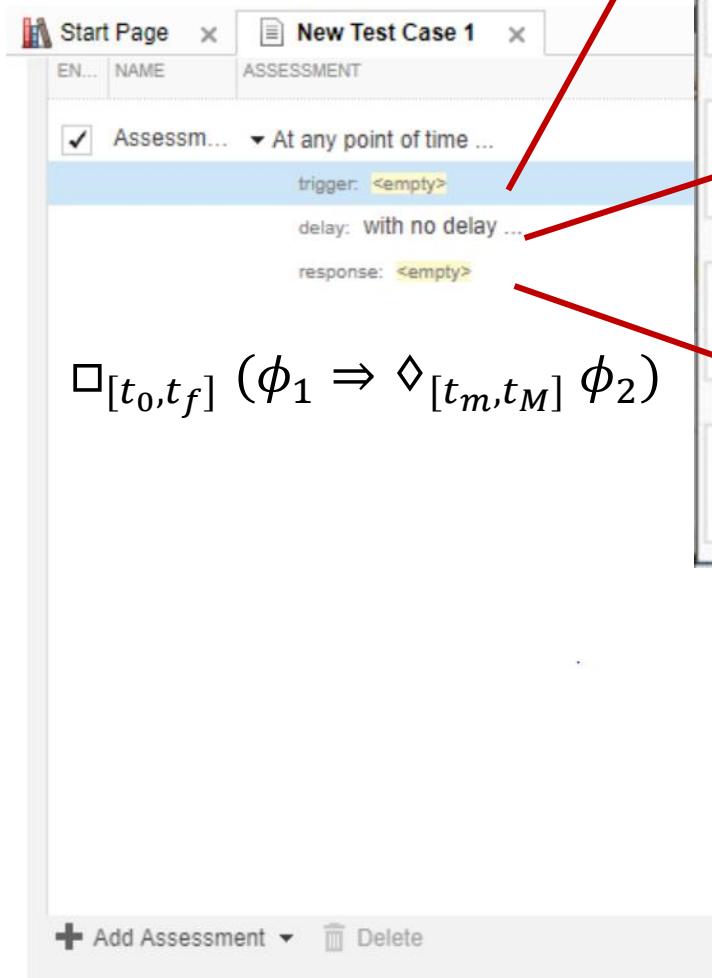
- Trigger-response**  
Check for a signal response once a trigger has been detected

At the bottom of the interface are buttons for "Add Assessment" (with a plus sign) and "Delete" (with a trash can icon).

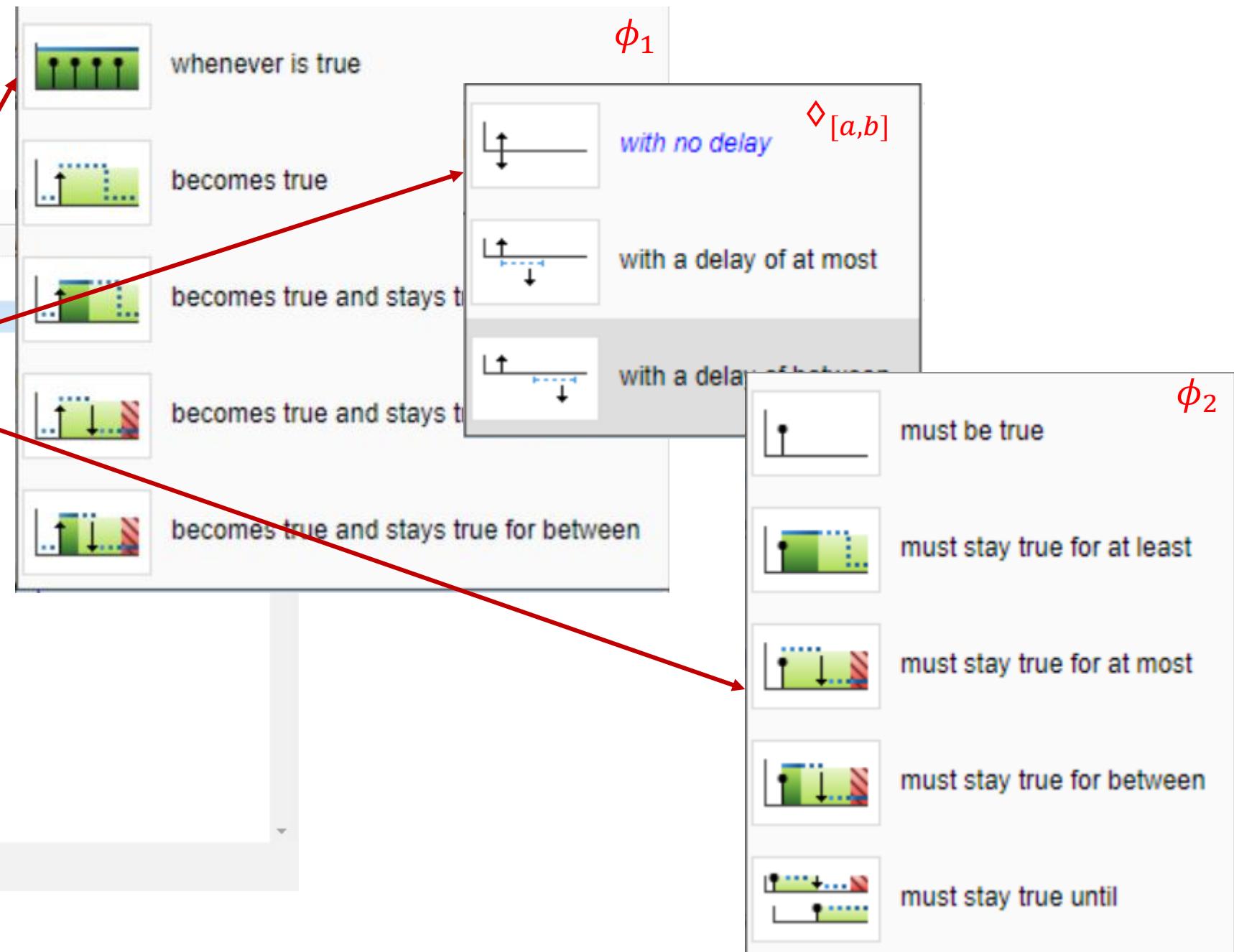
$$\square_{[t_0, t_f]} (\phi_1 \Rightarrow \diamond_{[t_m, t_M]} \phi_2)$$

# Authoring

>> sltestmgr



$$\square_{[t_0, t_f]} (\phi_1 \Rightarrow \diamond_{[t_m, t_M]} \phi_2)$$



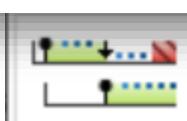
# Authoring

>> sltestmgr

The screenshot shows the MATLAB Test Manager interface. On the left, there's a sidebar with tabs for 'Start Page', 'New Test', 'EN...', 'NAME', 'ASSESSMENT', and a checked checkbox for 'Assessm... At any point'. Below these are fields for 'trigger: <empty>', 'delay: With', and 'response: <empty>'. At the bottom, there are buttons for '+ Add Assessment' and 'Delete'.

Table 3: Trigger response pattern ( $\varphi_1 \rightarrow \Diamond_{[a,b]} \varphi_2$ ) and its variations

Element	Variation
Trigger ( $\varphi_1$ )	<p>whenever &lt;condition1&gt; is true</p> <p>&lt;condition1&gt; becomes true</p> <p>&lt;condition1&gt; becomes true and stays true for at least</p> <p>&lt;condition1&gt; becomes true and stays true for at most</p> <p>&lt;condition1&gt; becomes true and stays true for between</p>
Response delay ( $\rightarrow \Diamond_{[a,b]}$ )	<p>with no delay</p> <p>with a delay of at most</p> <p>with a delay of in between</p>
Response ( $\varphi_2$ )	<p>&lt;condition2&gt; must be true</p> <p>&lt;condition2&gt; must stay true for at least</p> <p>&lt;condition2&gt; must stay true for at most</p> <p>&lt;condition2&gt; must stay true for between</p> <p>&lt;condition2&gt; must stay true until &lt;condition3&gt; becomes true</p>



must stay true until

# Symbol resolution and mapping

LOGICAL AND TEMPORAL ASSESSMENTS\*

EN...	NAME	ASSESSMENT	REQUIREMENTS
<input checked="" type="checkbox"/>	Assessment1	At any point of time ... bounds-check-pattern: always inside bounds signal: sig lower-bound: lb lower-bound-type: greater than upper-bound: ub upper-bound-type: less than	None

VISUAL REPRESENTATION

SYMBOLS

- sig: Name: Sine Wave:1, Path: sine\_wave/Sine Wave, Port Index: 1, Field/Element: <type an expression>
- lb: Expression: -0.4
- ub: Expression: 0.2

Map to signal

Map to workspace variables

Sine Wave

sine\_wave

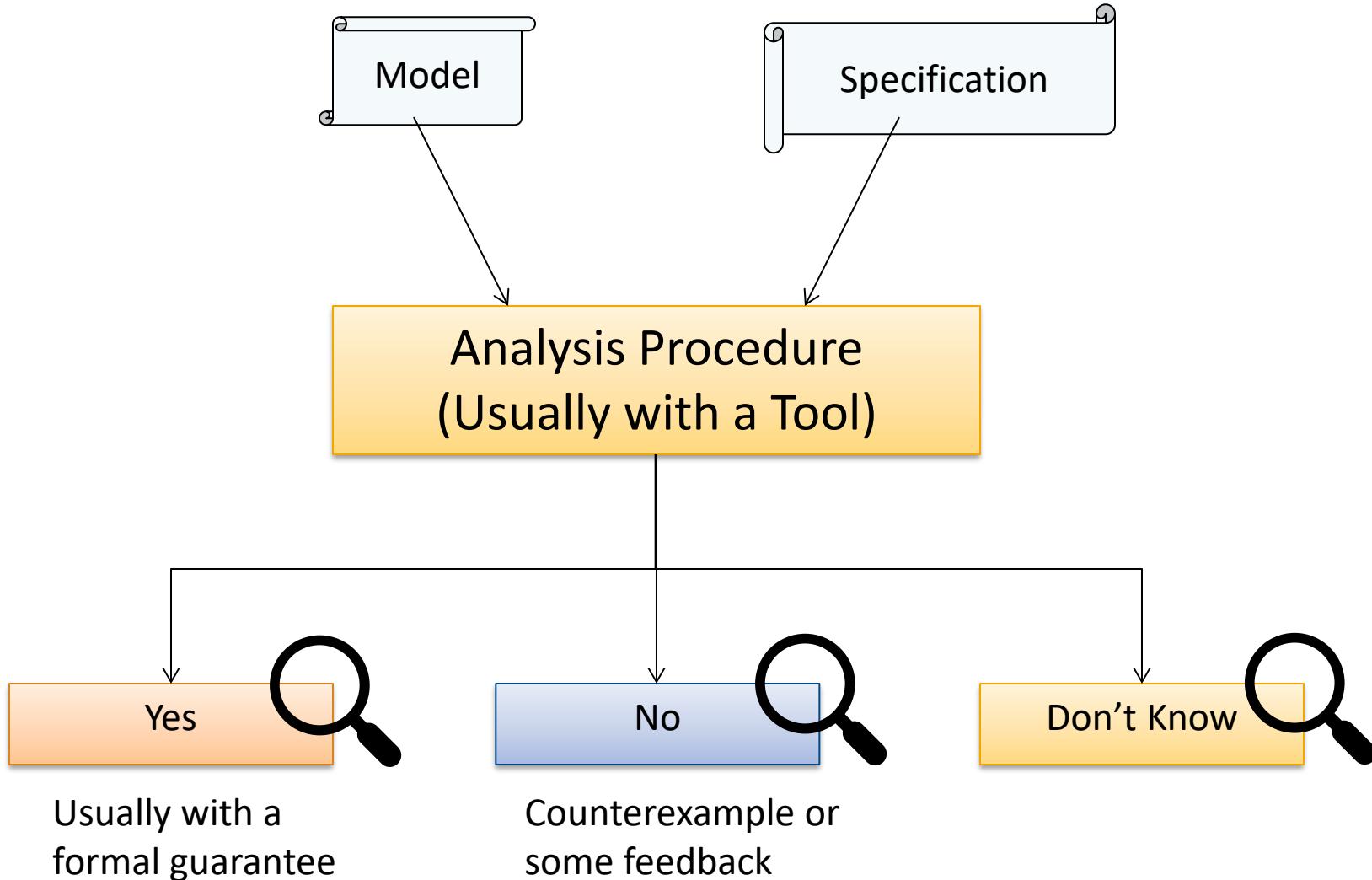
simout

1

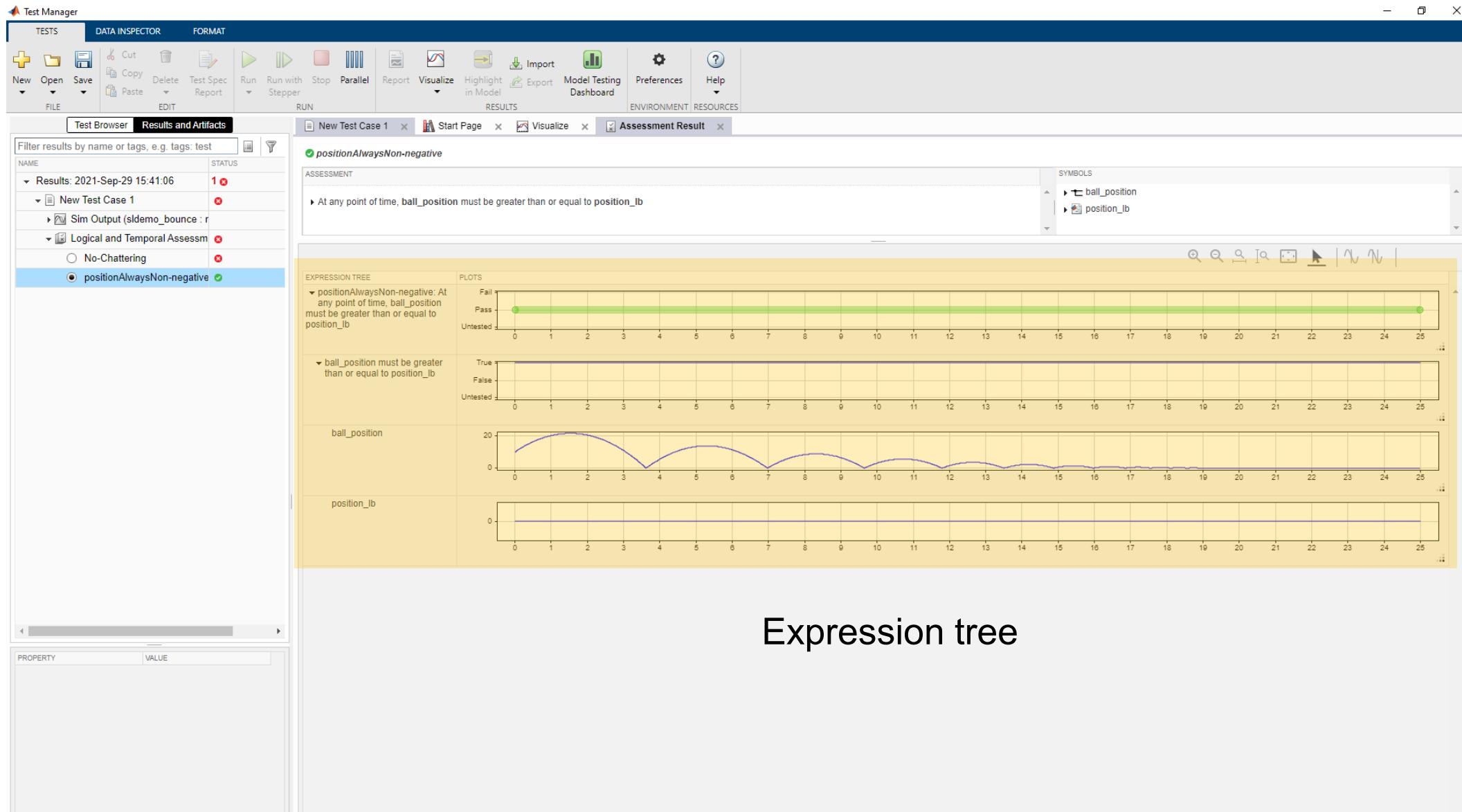
Add Assessment Delete Add Symbol Delete

44

# Assessment

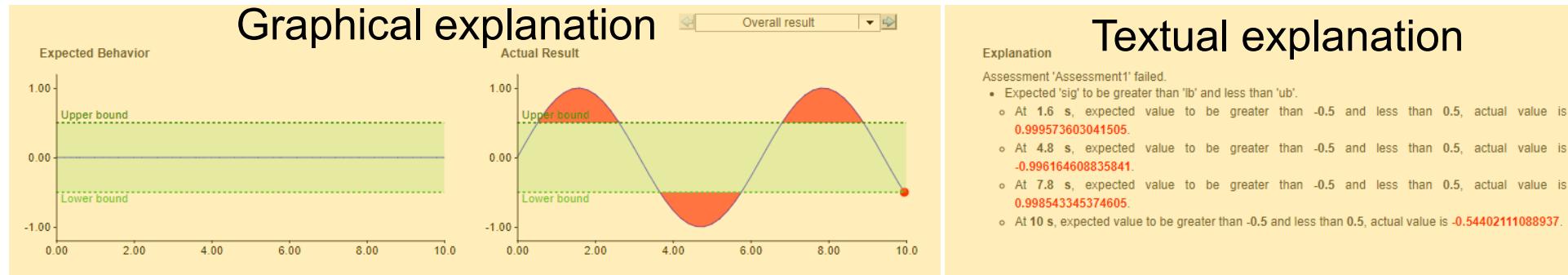
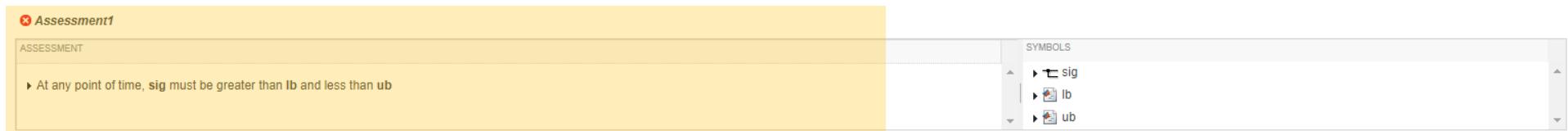


# Passing result feedback: visualize expression tree



Expression tree

# Fail result: additional graphical and textual explanations



# Failing result with explanations

► At any point of time, if  $\text{ball\_position} == 0$  becomes true then, with a delay of at most  $\text{eps1}$  seconds,  $\text{ball\_velocity} > 0$  must stay true for at least  $\tau$  seconds

$\square_{[t_0, t_f]} (\phi_1 \Rightarrow \Diamond_{[0, \epsilon_1]} (\Box_{[0, \tau]} \phi_2))$

$(\phi_1 \Rightarrow \Diamond_{[0, \epsilon_1]} (\Box_{[0, \tau]} \phi_2))$

$\phi_1$

$\Diamond_{[0, \epsilon_1]} (\Box_{[0, \tau]} \phi_2)$

$\Box_{[0, \tau]} \phi_2$

$\phi_2$

# Failing result with explanations

**Test Manager**

TESTS DATA INSPECTOR FORMAT

Test Browser Results and Artifacts

Filter results by name or tags, e.g. tags: test

NAME STATUS

- Results: 2021-Sep-29 15:41:06 1
- New Test Case 1
- Sim Output (sldemo\_bounce : r)
- Logical and Temporal Assessm
- No-Chattering
- positionAlwaysNon-negative

**No-Chattering**

ASSESSMENT

At any point of time, if ball\_position == 0 becomes true then, with a delay of at most eps1 seconds, ball\_velocity > 0 must stay true for at least tau seconds

SYMBOLS

- ball\_position
- eps1
- ball\_velocity

**Expected Behavior**

**Actual Result**

**Explanation**

Assessment 'No-Chattering' failed when triggered at 20.1646780511768 s.

- Trigger condition '(ball\_position == 0)' is true at 20.1646780511768 s.
- Expected response condition to be true within 20.1646780511768 s and 20.1746780511768 s.
  - I.e. with a delay of at most 0.01 s after 'rising edge' of trigger.
- Expected '(ball\_velocity > 0)' to be true at 20.1746780511768 s for at least 0.02 s, actual value at 20.1839739449984 s is false.

**“pass” (trigger true and response true)**

**EXPRESSION TREE**

No-Chattering: At any point in time, if (ball\_position == 0) becomes true then, with a delay of at most 0.01 seconds, (ball\_velocity > 0) must stay true for at least 0.02 seconds

if (ball\_position == 0) becomes true then, with a delay of at most 0.01 seconds, (ball\_velocity > 0) must stay true for at least 0.02 seconds

(ball\_position == 0) becomes true

with a delay of at most 0.01 seconds, (ball\_velocity > 0) must stay true for at least 0.02 seconds

**PLOTS**

Pass Fail

True False

True False

**PROPERTY VALUE**

$\Diamond_{[0,\epsilon_1]} (\Box_{[0,\tau]} \phi_2)$

$\Box_{[0,\tau]} \phi_2$

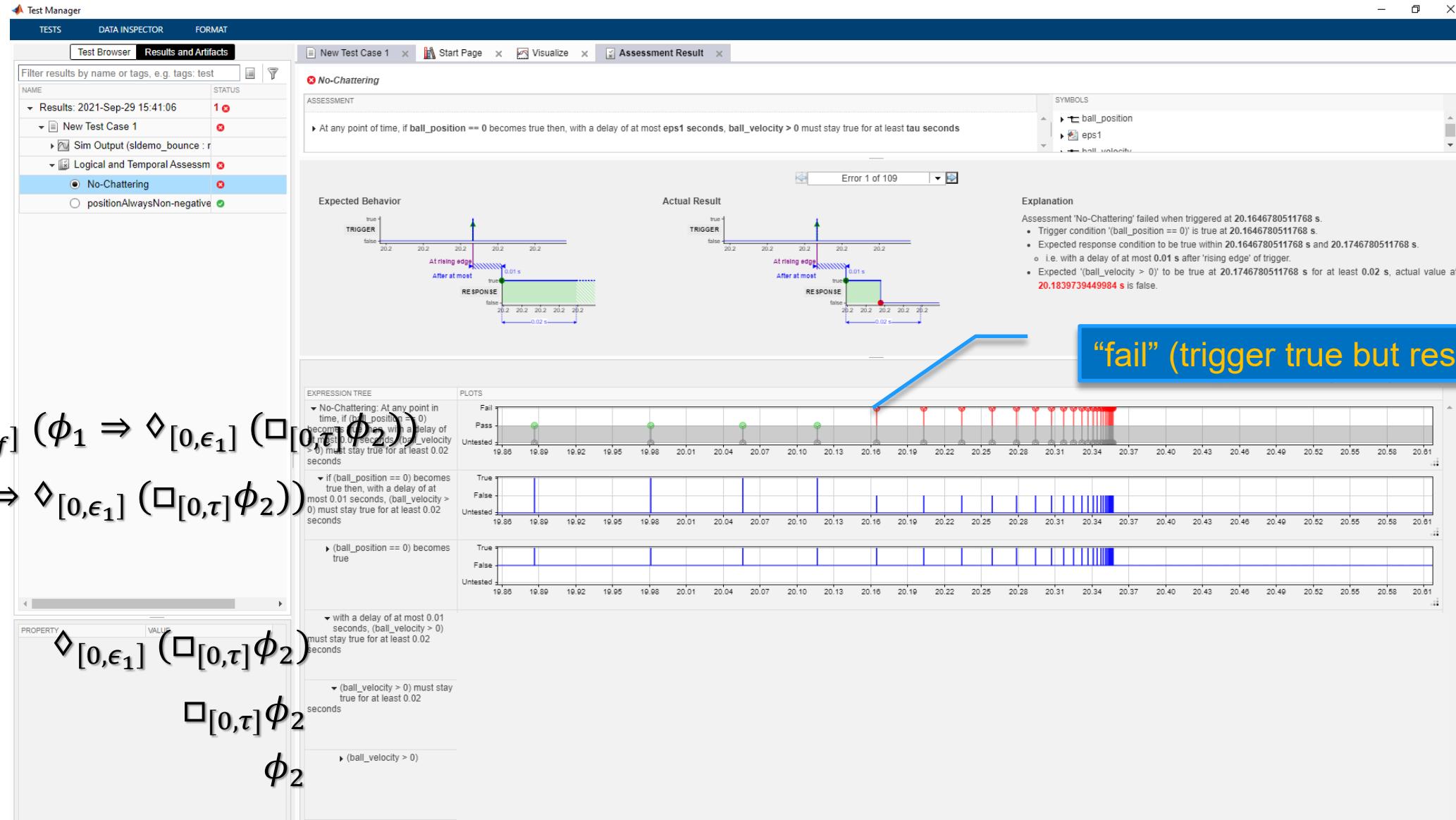
$\phi_2$

$\Diamond_{[0,\epsilon_1]} (\Box_{[0,\tau]} \phi_2)$

$\Box_{[0,\tau]} \phi_2$

$\phi_2$

# Failing result with explanations



# Failing result with explanations

**Test Manager**

TESTS DATA INSPECTOR FORMAT

Test Browser Results and Artifacts

Filter results by name or tags, e.g. tags: test

NAME STATUS

- Results: 2021-Sep-29 15:41:06 1
- New Test Case 1
- Sim Output (sldemo\_bounce : r)
- Logical and Temporal Assessment
- No-Chattering
- positionAlwaysNon-negative

**No-Chattering**

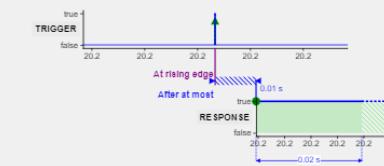
ASSESSMENT

At any point of time, if ball\_position == 0 becomes true then, with a delay of at most eps1 seconds, ball\_velocity > 0 must stay true for at least tau seconds

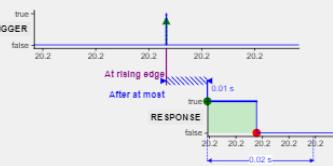
SYMBOLS

- ball\_position
- eps1
- ball\_velocity

**Expected Behavior**



**Actual Result**



**Explanation**

Assessment 'No-Chattering' failed when triggered at 20.1646780511768 s.

- Trigger condition '(ball\_position == 0)' is true at 20.1646780511768 s.
- Expected response condition to be true within 20.1646780511768 s and 20.1746780511768 s.
  - I.e. with a delay of at most 0.01 s after 'rising edge' of trigger.
- Expected '(ball\_velocity > 0)' to be true at 20.1746780511768 s for at least 0.02 s, actual value at 20.1839739449984 s is false.

**“untested” (vacuously true because trigger false)**

**EXPRESSION TREE**

Fail

Pass

Untested

19.88 19.89 19.90 19.91 19.92 19.93 19.94 19.95 19.96 19.97 19.98 19.99 20.00 20.01 20.02 20.03 20.04 20.05 20.06 20.07 20.08 20.09 20.10 20.11 20.12 20.13 20.14 20.15 20.16 20.17 20.18 20.19 20.20 20.21 20.22 20.23 20.24 20.25 20.26 20.27 20.28 20.29 20.30 20.31 20.32 20.33 20.34 20.35 20.36 20.37 20.38 20.39 20.40 20.41 20.42 20.43 20.44 20.45 20.46 20.47 20.48 20.49 20.50 20.51 20.52 20.53 20.54 20.55 20.56 20.57 20.58 20.59 20.60

**PLOTS**

True

False

Untested

19.88 19.89 19.90 19.91 19.92 19.93 19.94 19.95 19.96 19.97 19.98 19.99 20.00 20.01 20.02 20.03 20.04 20.05 20.06 20.07 20.08 20.09 20.10 20.11 20.12 20.13 20.14 20.15 20.16 20.17 20.18 20.19 20.20 20.21 20.22 20.23 20.24 20.25 20.26 20.27 20.28 20.29 20.30 20.31 20.32 20.33 20.34 20.35 20.36 20.37 20.38 20.39 20.40 20.41 20.42 20.43 20.44 20.45 20.46 20.47 20.48 20.49 20.50 20.51 20.52 20.53 20.54 20.55 20.56 20.57 20.58 20.59 20.60

**PROPERTY VALUE**

$\Diamond_{[0,\epsilon_1]} (\Box_{[0,\tau]} \phi_2)$

$\Box_{[0,\tau]} \phi_2$

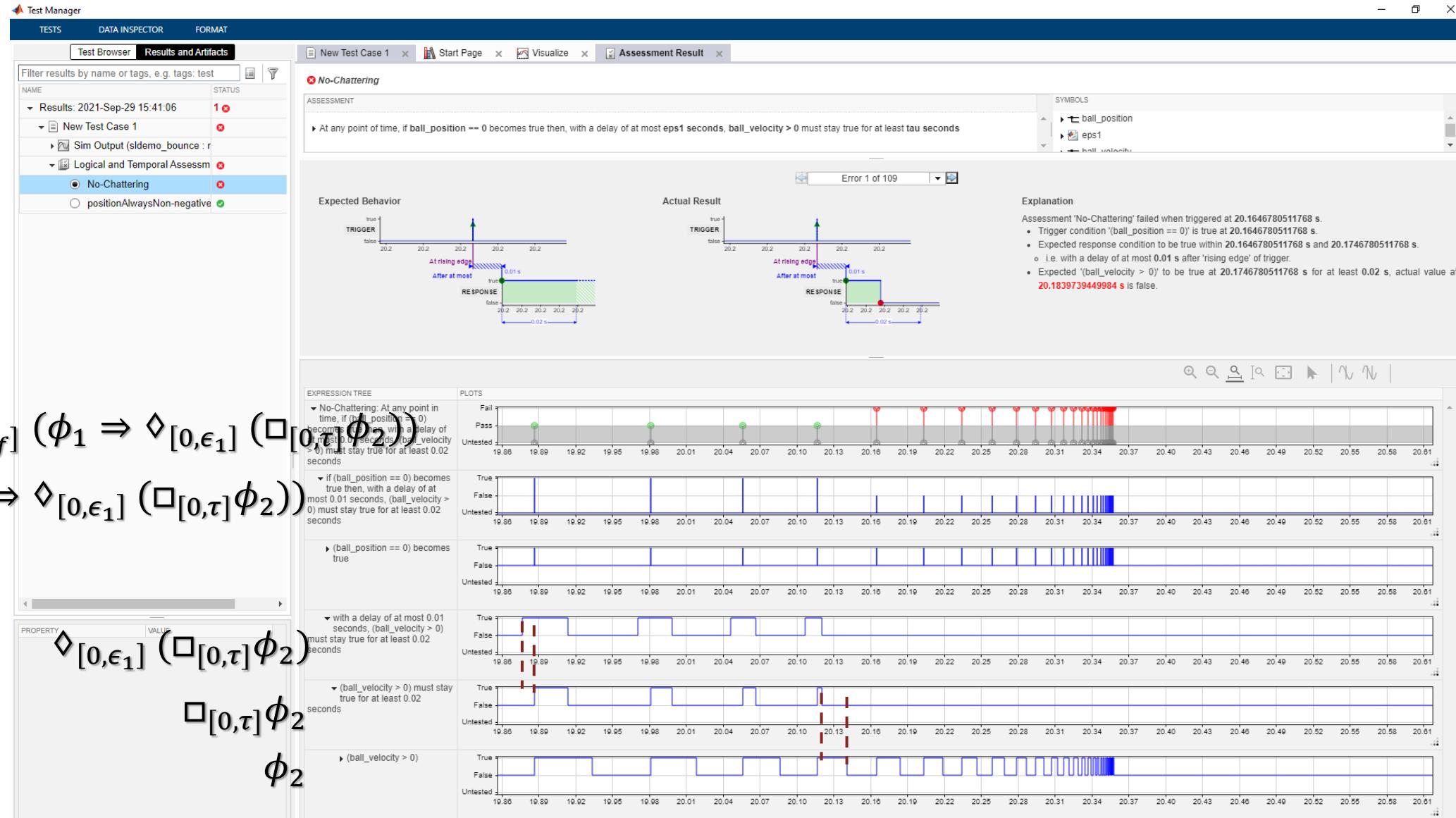
$\phi_2$

$\Diamond_{[0,\epsilon_1]} (\Box_{[0,\tau]} \phi_2)$

$\Box_{[0,\tau]} \phi_2$

$\phi_2$

# Failing result with explanations



$$\square_{[t_0, t_f]} (\phi_1 \Rightarrow \diamond_{[0, \epsilon_1]} (\square_{[0, \epsilon_1]} (\phi_1 \Rightarrow \diamond_{[0, \epsilon_1]} (\square_{[0, \tau]} \phi_2)))$$

$\phi_1$

$$\diamondsuit_{[0,\epsilon_1]} (\Box_{[0,\tau]} \phi_2)_s$$

$$\square_{[0,\tau]} \phi_2$$

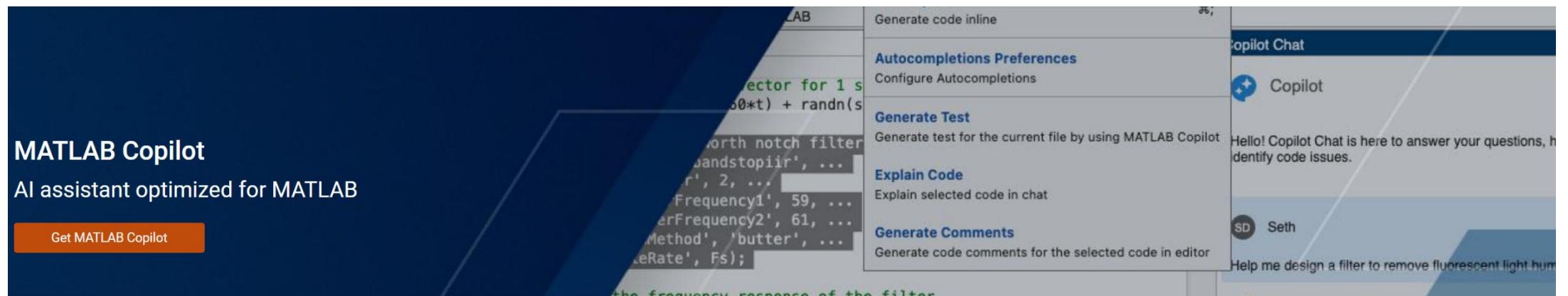
$\phi_2$

# Give it a try!

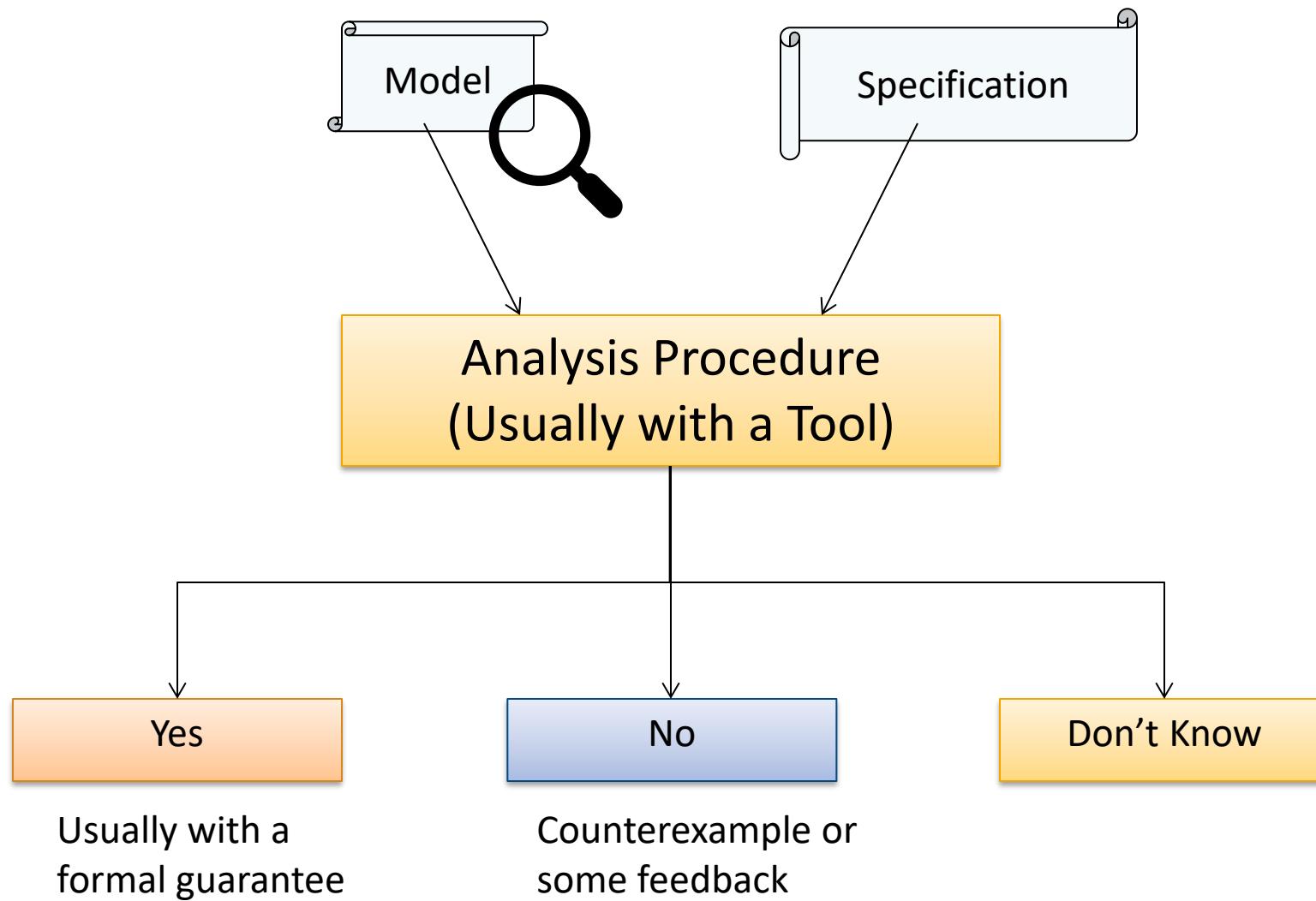
<https://www.mathworks.com/help/sltest/ug/temporal-assessments.html>

# Advanced topic

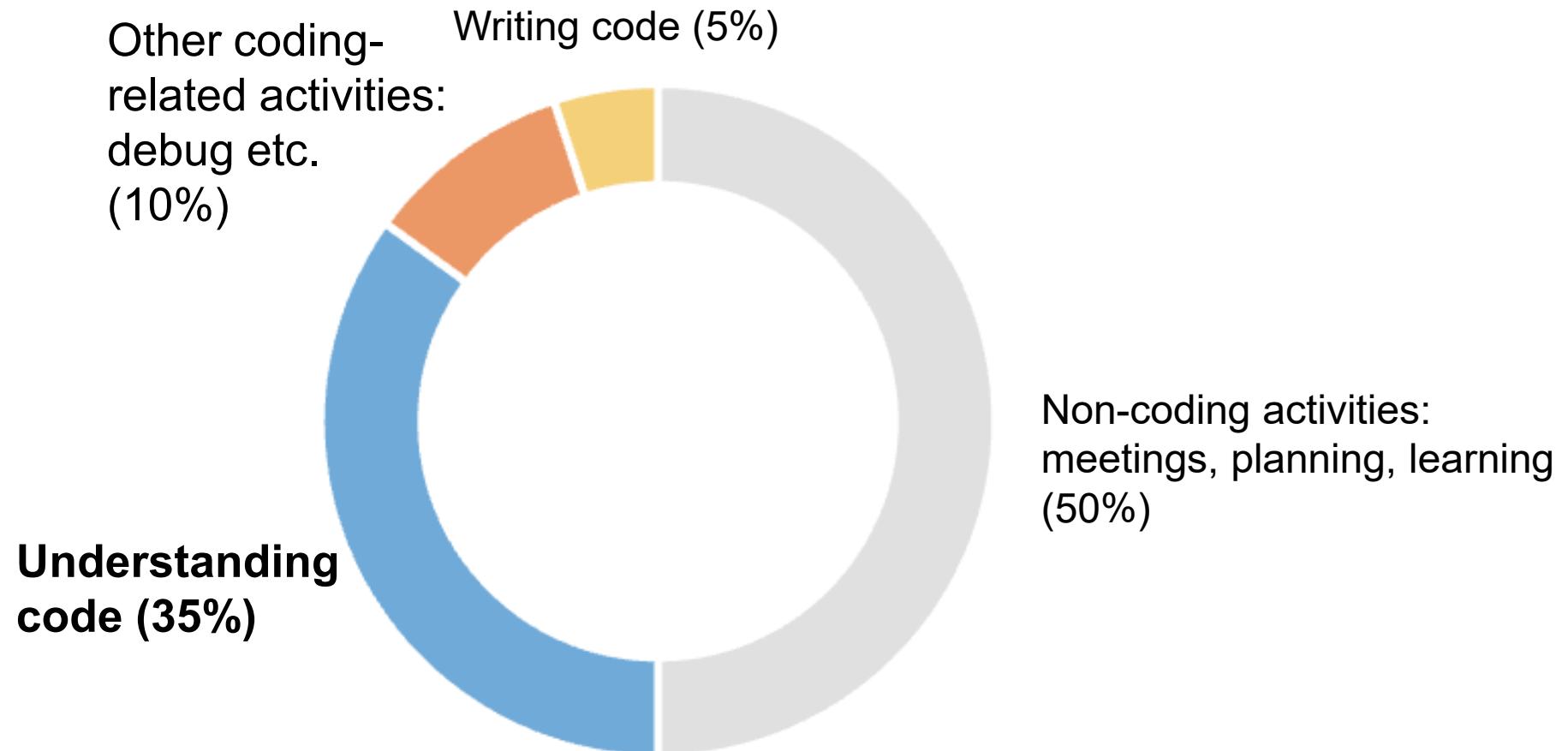
## Generative AI for explainability



<https://www.mathworks.com/products/matlab-copilot.html>



# Much of developer time is spent understanding and evaluating code



# Actions provide timely assistance in common workflows

```
1 function str = formatNumbersAsStrings(num)
2     num = round(num);
3     str = string(num);
4     str = regexp替換(str,"[0-9] (?=(?:[0-9]{3})+(?! [0-9]))","$&,");
5 end
```

MATLAB R2025a

HOME PLOTS APPS EDITOR PUBLISH VIEW

New Open Save Print Compare Go To Find Copilot Refactor Profiler Code Issues Debugger Generate Find Run Section Run Step Stop

FILE NAVIGATE CODE ANALYZE TEST SECTION RUN

/ Users > sdeland > Documents > MATLAB > Work

formatNumbersAsString.m ×

```
1 function str = formatNumbersAsString(num)
2     num = round(num);
3     str = string(num);
4     str = regexprep(str, '[0-9](?=(?:[0-9]{3})+([?![0-9]))','$&,');
5 end
```

Copilot Chat

Copilot

Hello! Copilot is here to answer your questions, help write and explain code, and even identify code issues. Learn more about [Copilot](#).

Remember that Copilot sometimes writes code and text that seems accurate, but is not. Make sure to verify any received code and give feedback on the results to help improve the responses.

Shuffle Example Prompts

Read "peppers.png" and display it as grayscale

Plot a sine wave and add a vertical line using vline

Ask Copilot

Validate generated output before use.

Editor: 110% UTF-8 LF formatNumbersAsString Ln 4 Col 66



Copilot

## What

**Brief Overview:** This code adds commas to a string representation of a number to format it with thousands separators.

**Summary:** The code uses a regular expression to find positions in a string where a digit is followed by groups of three digits, and it inserts a comma at those positions. This is particularly useful for formatting large numbers for better readability, such as converting "1000000" into "1,000,000".

**Walkthrough:**

### 1. Regular Expression Replacement:

```
str = regexp替換(str, '[0-9](?=(?:[0-9]{3})+(?! [0-9]))', '$&,');
```

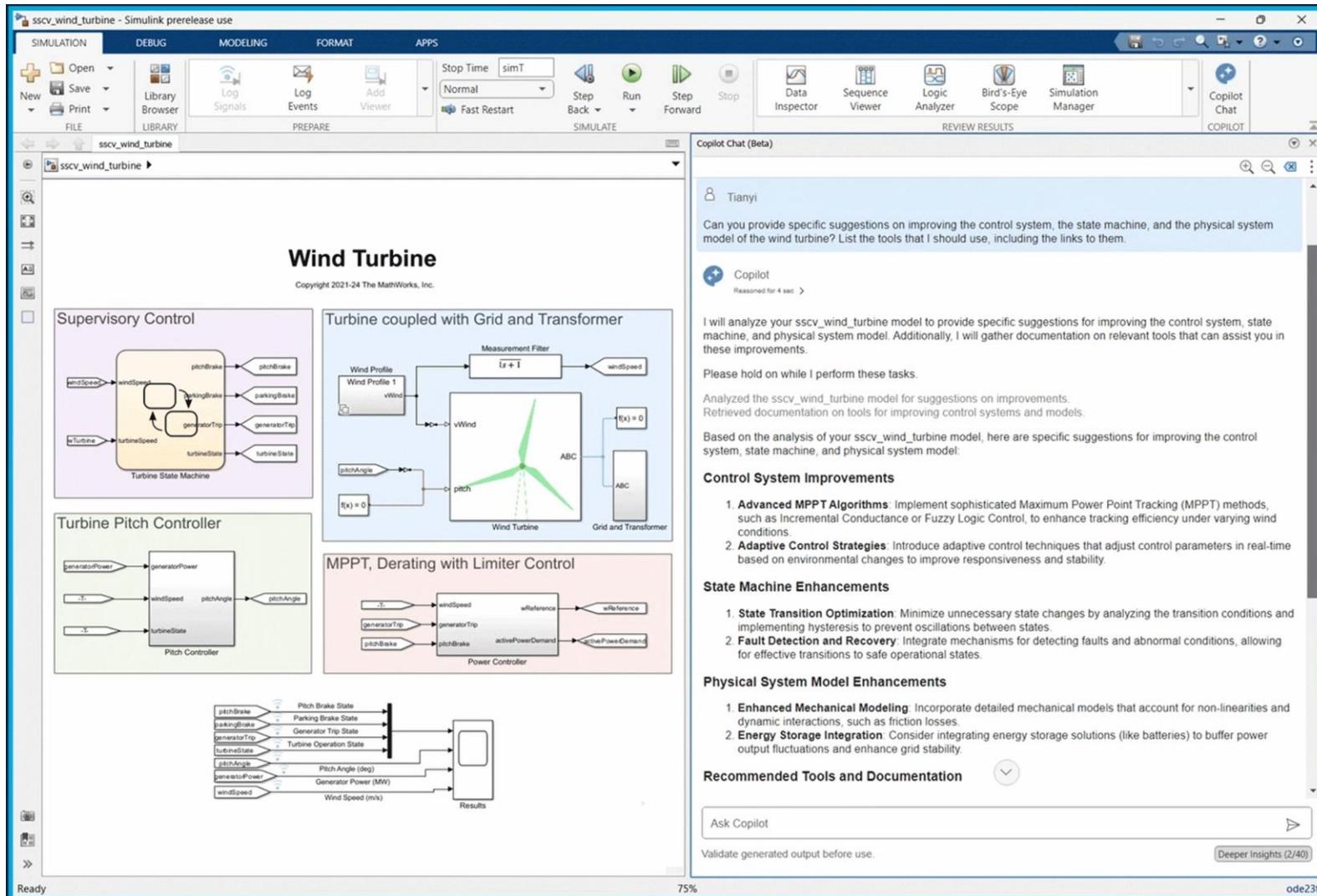


## How

- `regexp替換` is a MATLAB function that replaces parts of a string based on a regular expression pattern.
- The pattern '`[0-9](?=(?:[0-9]{3})+(?! [0-9]))`' is used to identify digits that should be followed by a comma.
  - `[0-9]` matches any single digit.
  - `(?=(?:[0-9]{3})+(?! [0-9]))` is a lookahead assertion that checks if the matched digit is followed by groups of three digits (i.e., thousands) without any additional digits after those groups.
- `'$&,'` specifies that the matched digit (denoted by `$&`) should be followed by a comma.
- The result is that every appropriate digit in the string `str` is followed by a comma, effectively formatting the number with thousands separators.

This code is a concise way to enhance the readability of numerical strings by formatting them appropriately.

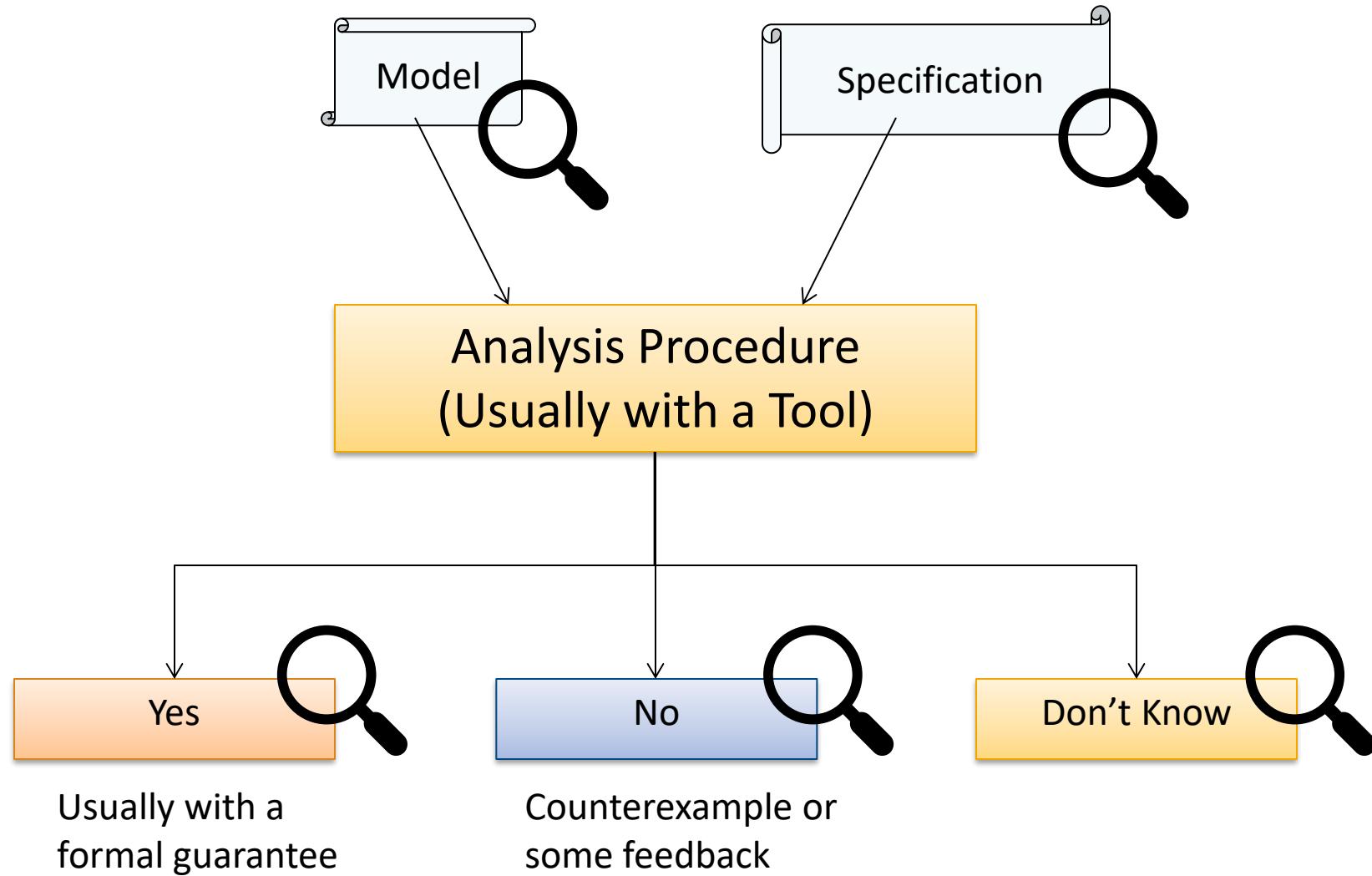
# Simulink Copilot coming soon...



<https://www.mathworks.com/products/simulink-copilot.html>

# Summary

- Building confidence in the design of complex engineered systems
  - Models, specifications, and various analysis problems
  - The role of tooling in enabling model-based design, analysis, and operation
- Temporal Assessments:
  - Logical formulas that read like English sentences
  - Templates for *bounds check*, *custom formula*, and *trigger-response* patterns
  - Graphical help with fictitious pass/fail test during authoring time
  - Graphical and textual help for pass/fail test results during simulation
  - Expression tree to help narrow down failures
- Advanced topic: AI for explainability
  - For MATLAB code and error messages
  - For Simulink models and error messages coming soon...



# User feedback and beta testing are key steps in feature design

- Events such as *MathWorks Research Summits* and *MathWorks Advisory Board* enable us to provide a preview of future functionality to solicit engineering feedback
- There may be specific *beta programs* for new products, e.g., for [Simulink Copilot](#)
- Sometimes we work closely with a *lead customer* on a key new feature and gather user feedback that way
- Other *existing mechanisms* for customer-facing engineers to record and share feedback with engineering teams

# References

- C. Menghi, E. Balai, D. Valovcin, C. Sticksel, [A. Rajhans](#), "Completeness and Consistency of Tabular Requirements: an SMT-Based Verification Approach", IEEE Transactions on Software Engineering, Vol: 51, Issue: 2, February 2025. [[Paper \(Open Access\)](#)]
- [A. Rajhans](#), A. Mavrommati, P.J. Mosterman, and R.G. Valenti, "Specification and Runtime Verification of Temporal Assessments in Simulink", 21st International Conference on Runtime Verification (RV) 2021. [[Preprint \(PDF\)](#) | [Accompanying Video](#)]
- S. Castro, P. J. Mosterman, [A. H. Rajhans](#), and R. G. Valenti, "Challenges in the Operation and Design of Intelligent Cyber-Physical Systems", Book Chapter. In Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy, Saurabh Mittal and Andreas Tolk, Eds., Wiley, January 2020. [[Preprint \(PDF\)](#)]
- [A. Rajhans](#), S. Avadhanula, A. Chutinan, P. J. Mosterman, and F. Zhang, "Graphical Hybrid Automata with Simulink and Stateflow", 21st International Conference on Hybrid Systems: Computation and Control, 2018. [[Preprint \(PDF\)](#)]
- <https://www.mathworks.com/help/sltest/ug/temporal-assessments.html>
- <https://www.mathworks.com/help/slrequirements/ref/requirementstable.html>
- <https://www.mathworks.com/products/matlab-copilot.html>

