MATLAB EXPO 2019

Simplifying Requirements Based Verification with Model-Based Design

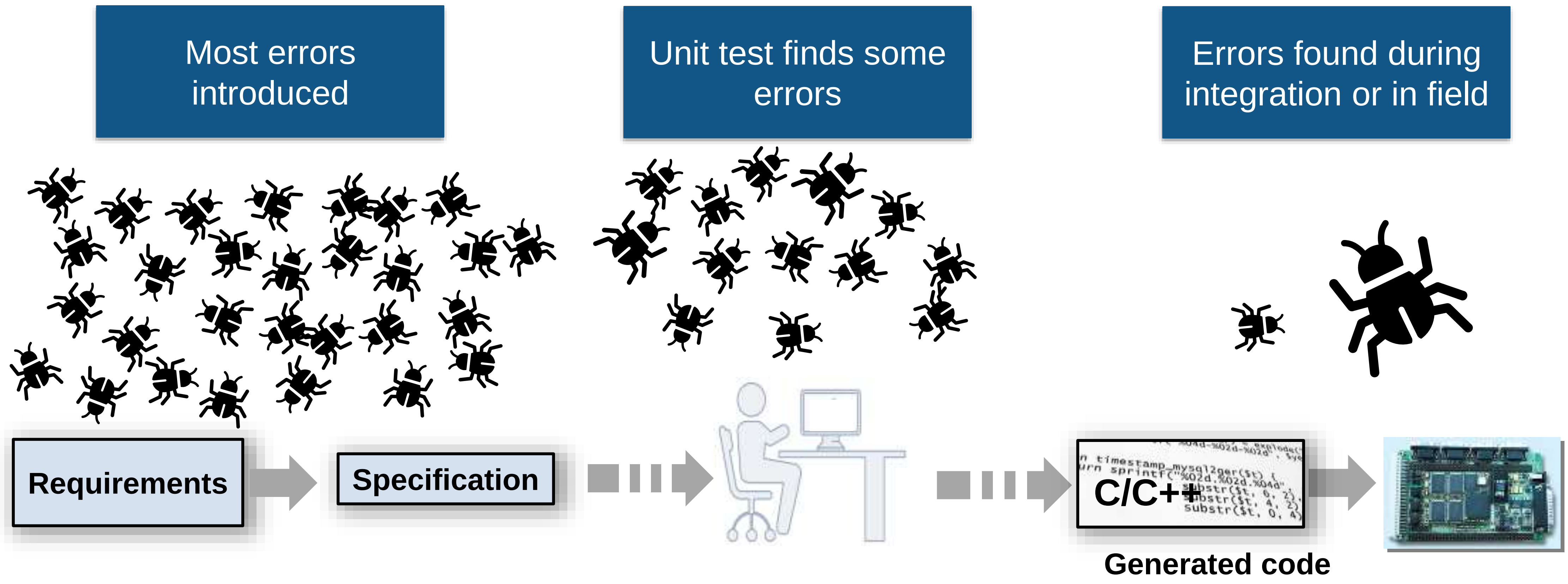
Vamshi Kumbham
Pilot Engineering

Key takeaways

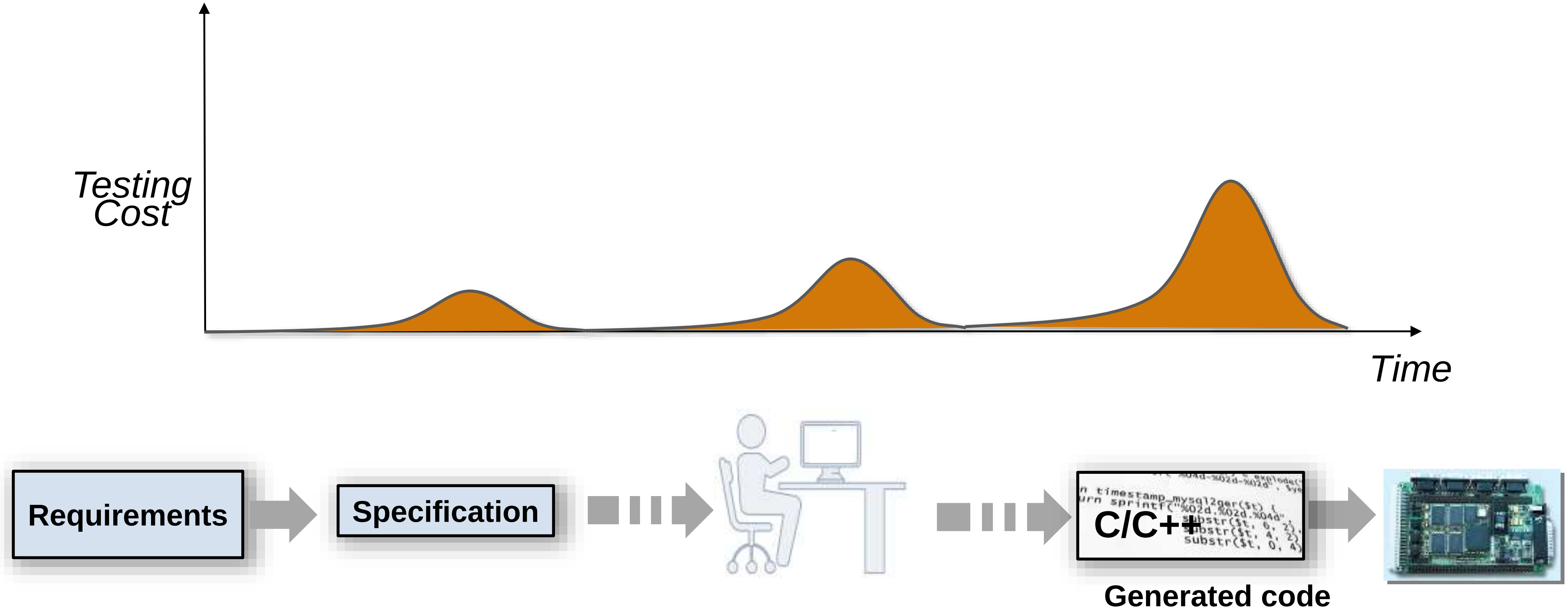
- Verify and validate requirements earlier
- Identify inconsistencies in requirements by using unambiguous assessments
- Traceability from requirements to design and test

> *“By enabling us to analyze requirements quickly, reuse designs from previous products, and eliminate manual coding errors, Model-Based Design has reduced development times and enabled us to shorten schedules to meet the needs of our customers.”*
>
> *- MyoungSuk Ko, LS Automotive*

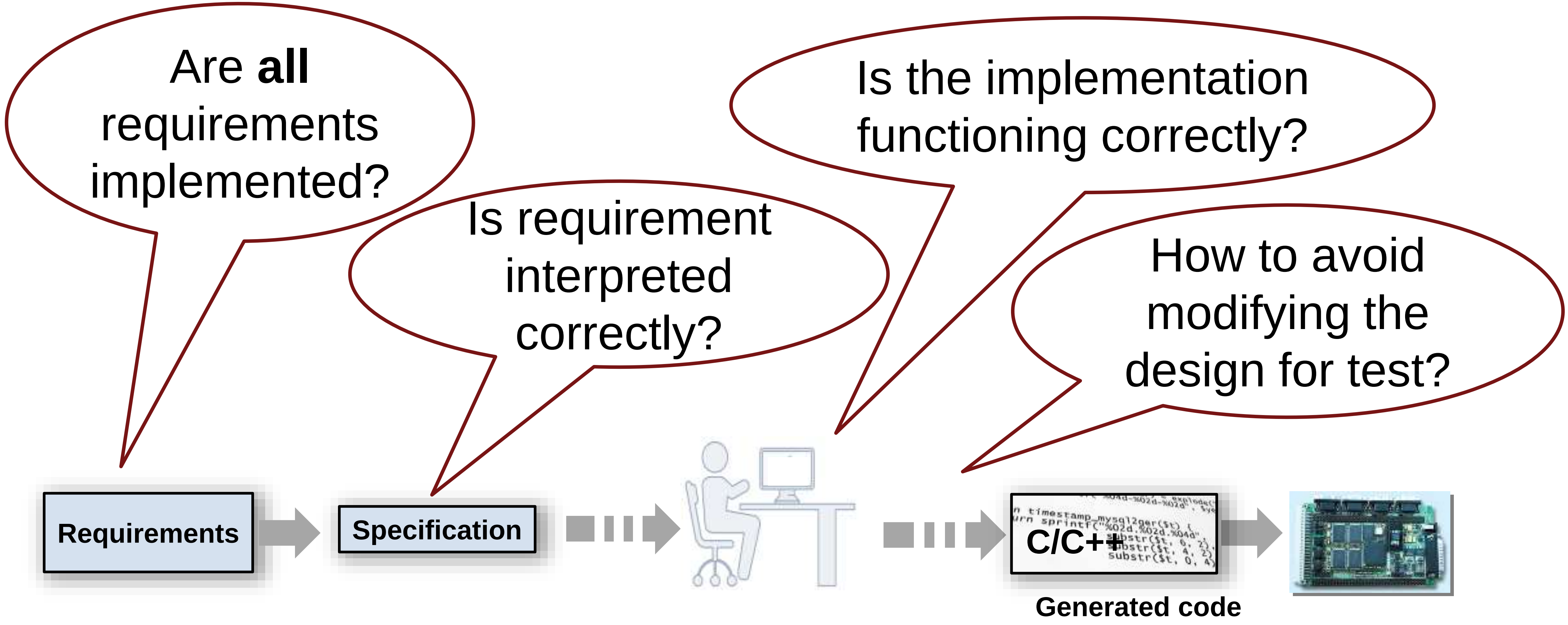
Challenge: Errors introduced early but found late



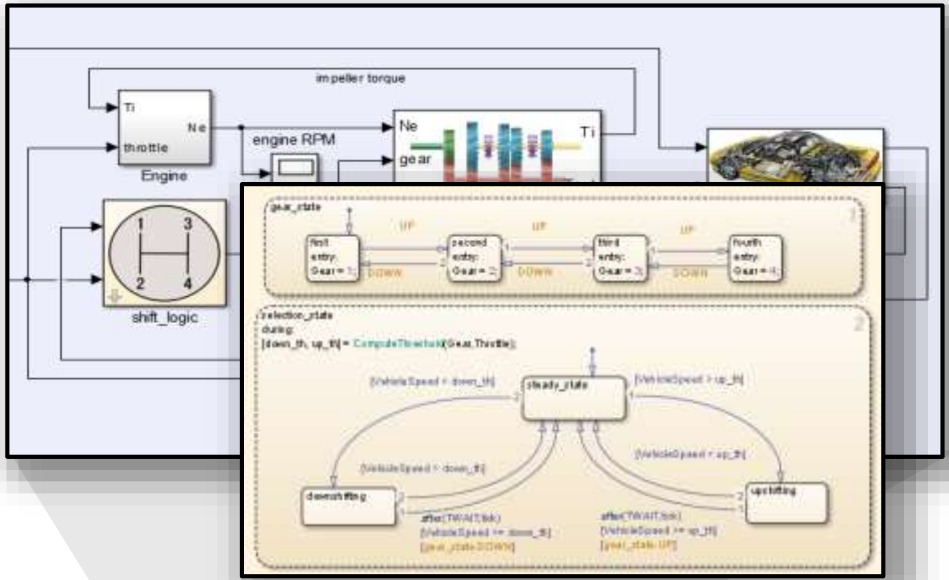
Cost of finding errors increases over time



Challenges with requirements based verification



Simulink models for specification

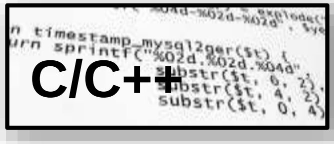


Model-Based Design enables:

- *Early testing to increase confidence in your design*
- *Delivery of higher quality software throughout the workflow*

Requirements → **Design Model** →

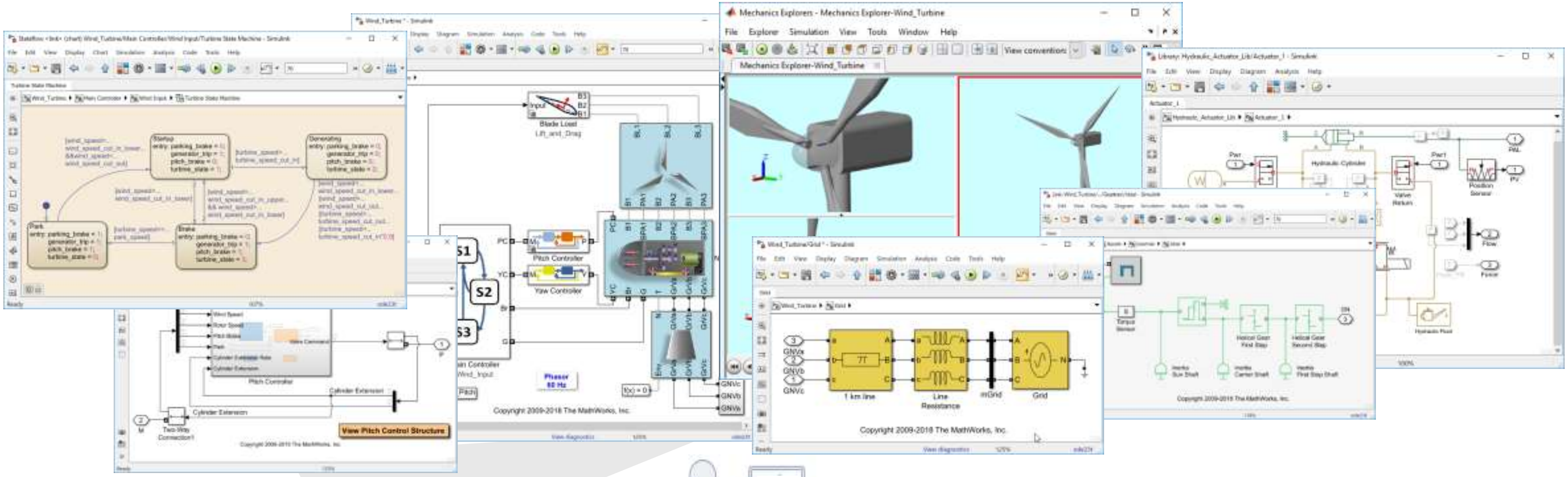
→



Generated code

→

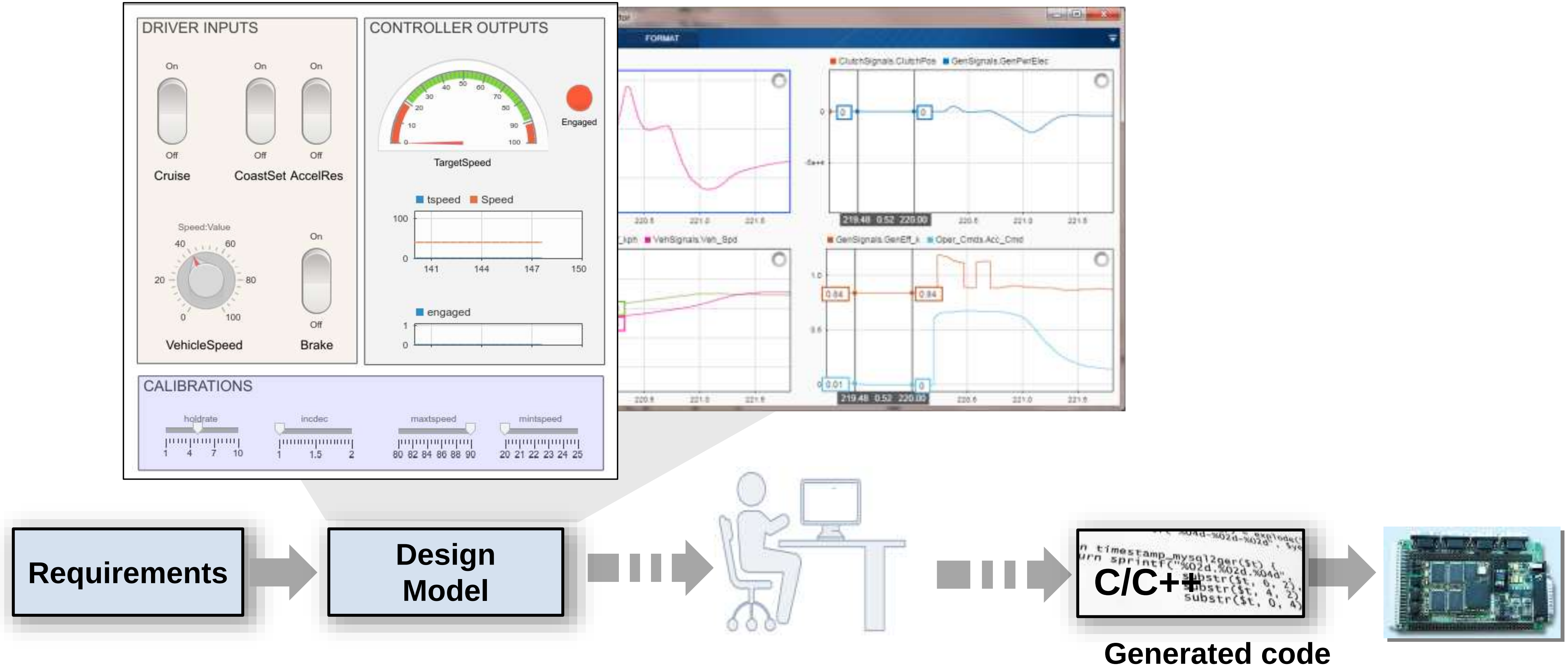
Multiple languages to describe complex systems



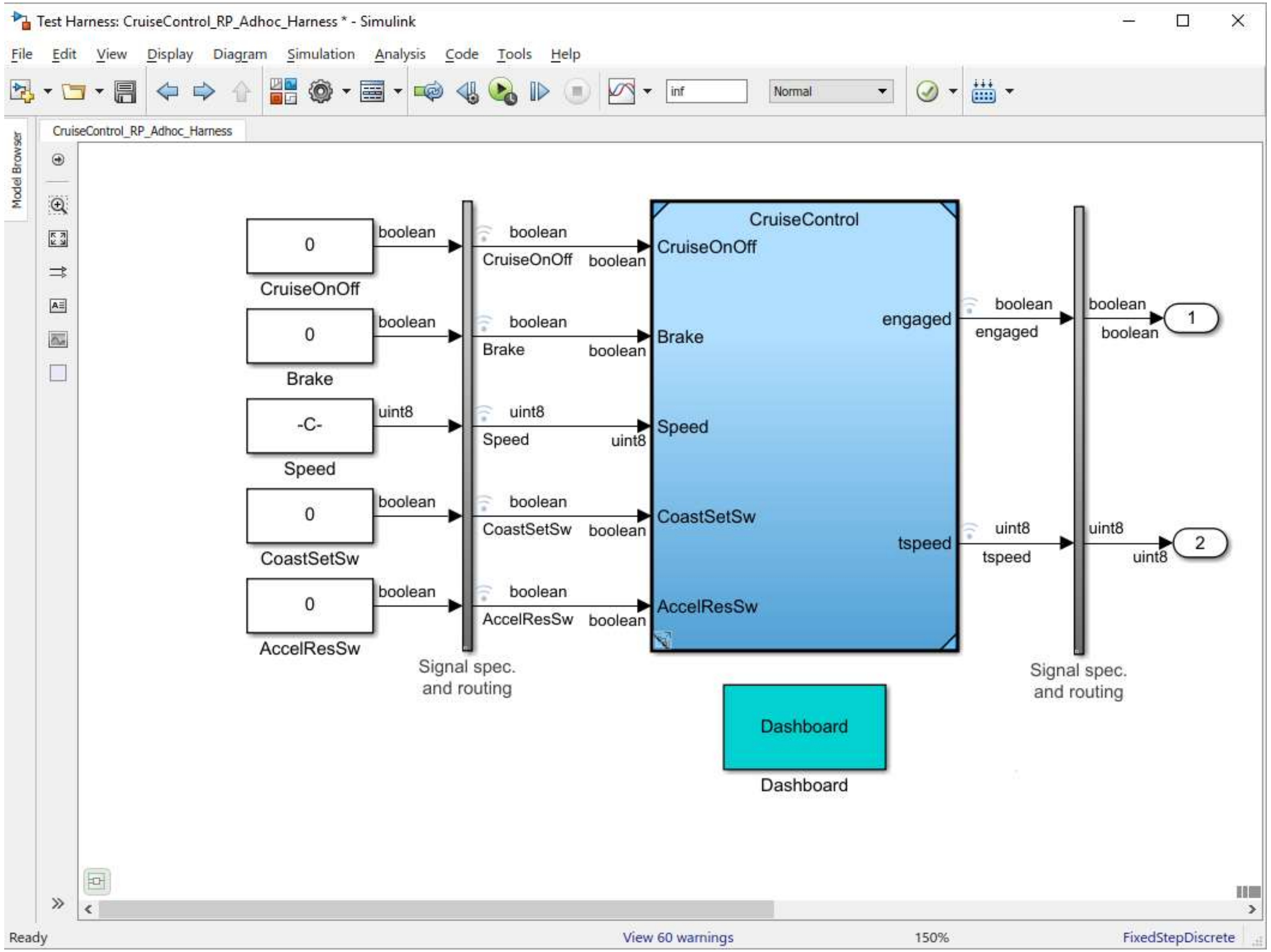
Requirements → **Design Model**

Generated code

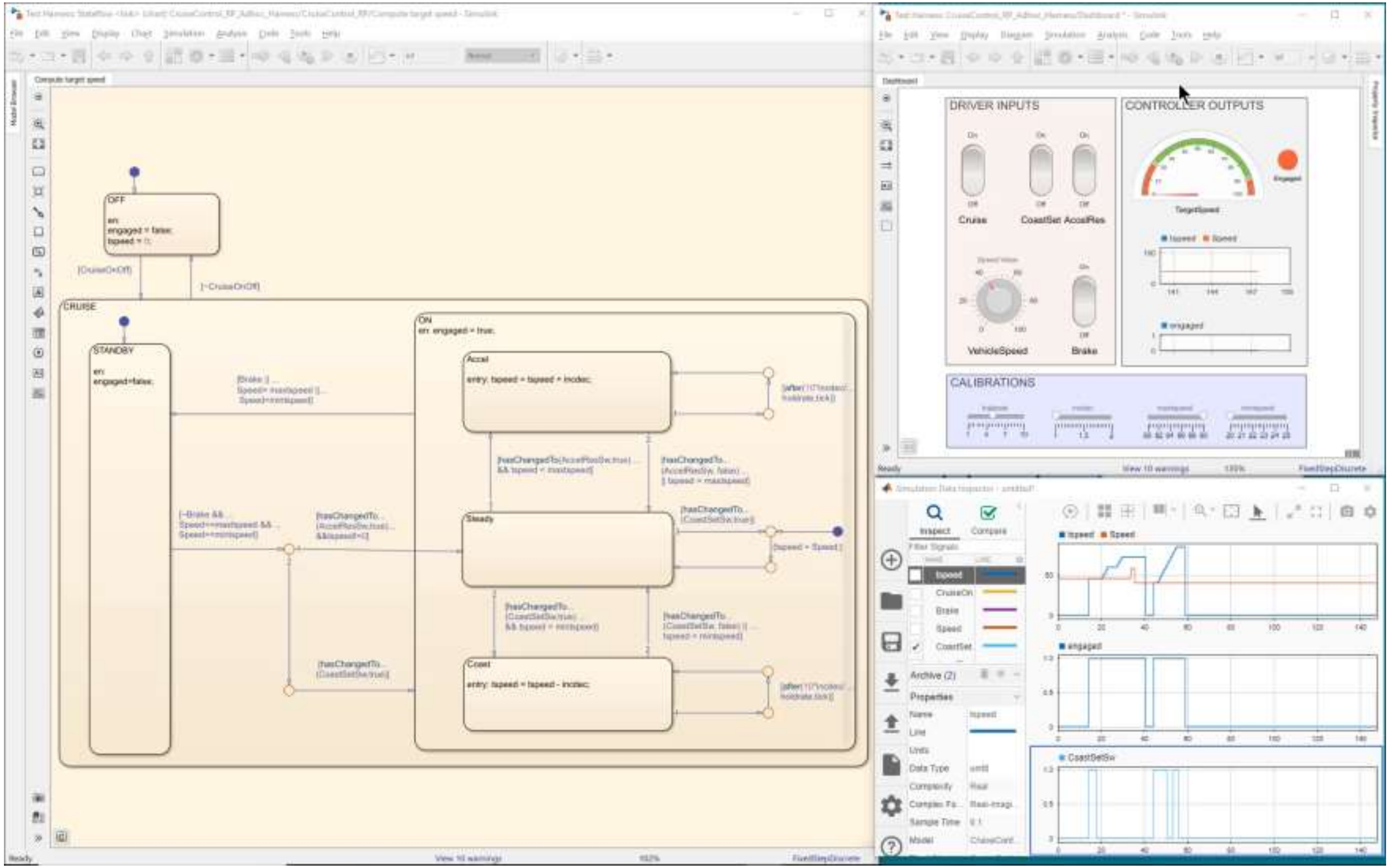
Ad-Hoc Testing: Explore behavior and design alternatives



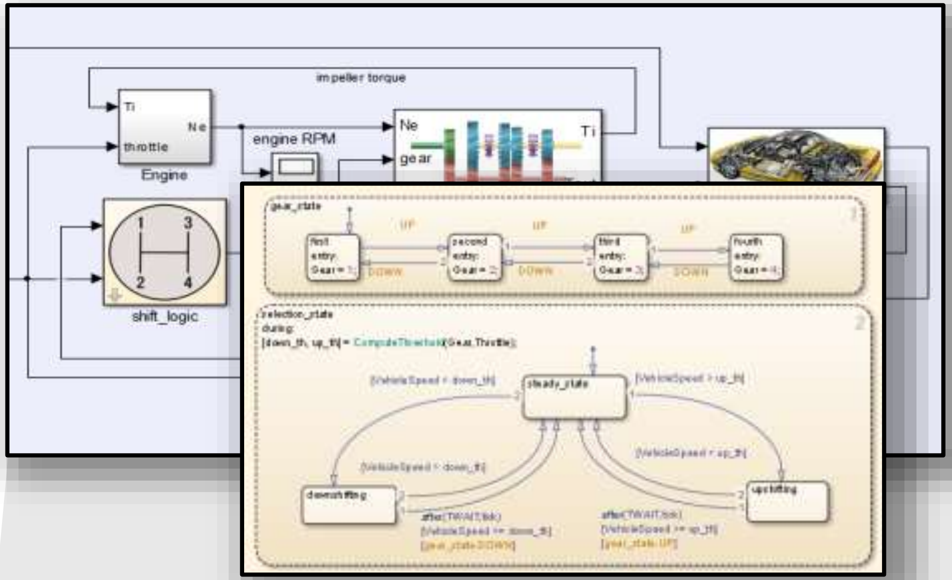
Validate behavior earlier with simulation



Validate Behavior Earlier with Simulation



Complete Model Based Design



Code Generation

Simulink Models

Requirements

Design Model

Model used for production code generation

Generated code

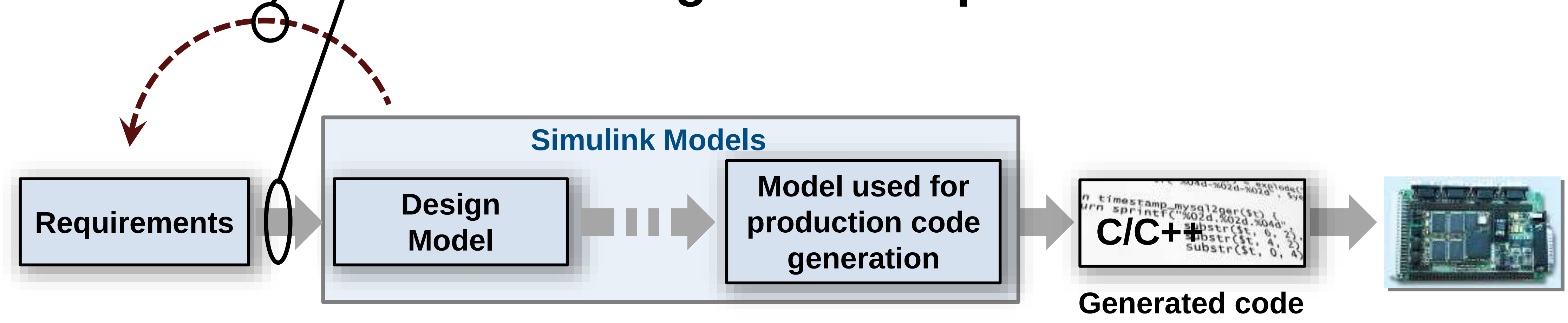
Systematically verify requirements

Are all requirements implemented?

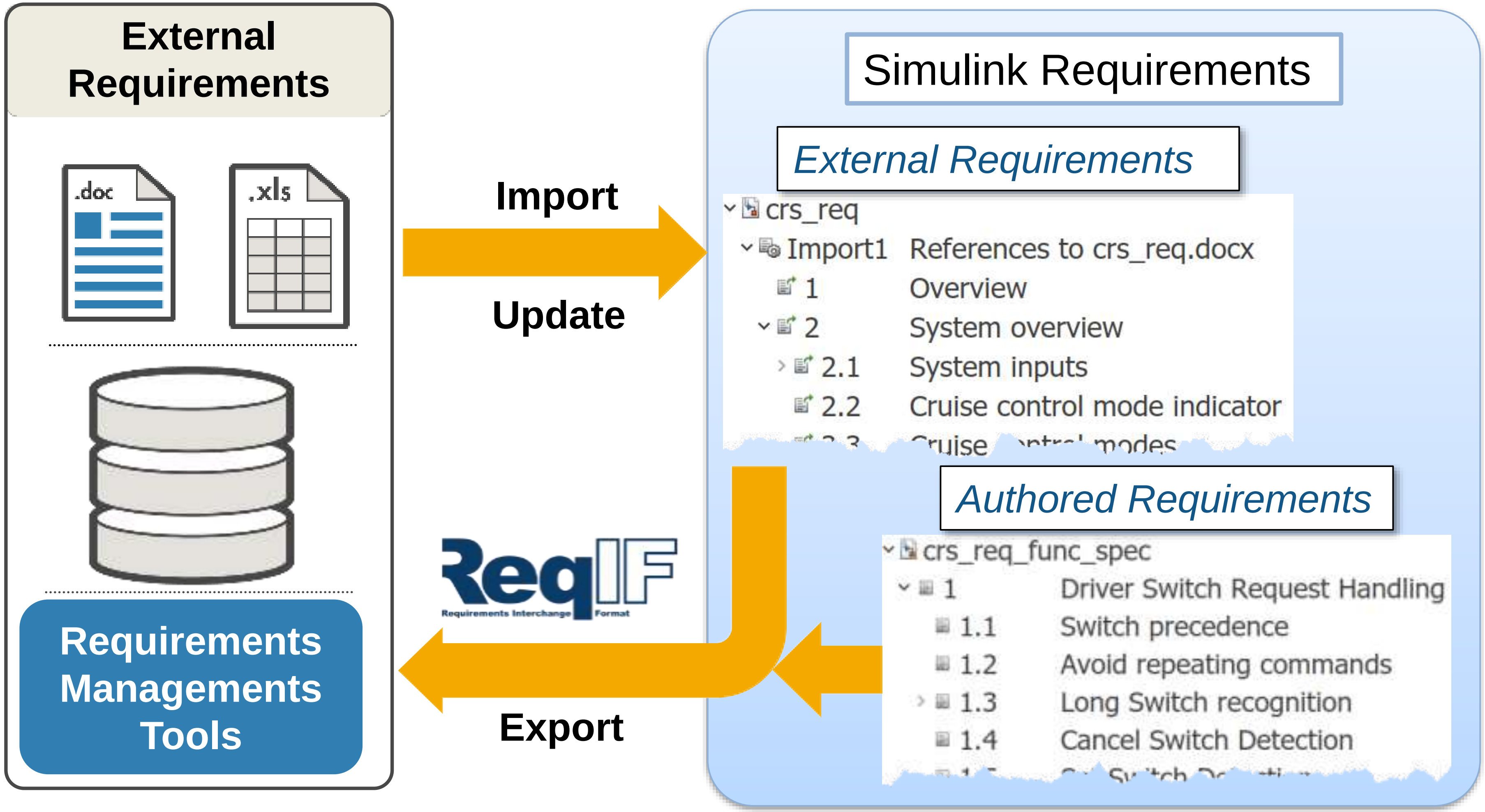
Is the implementation functioning correctly?

Are designs and requirements consistent?

Requirements Based Testing



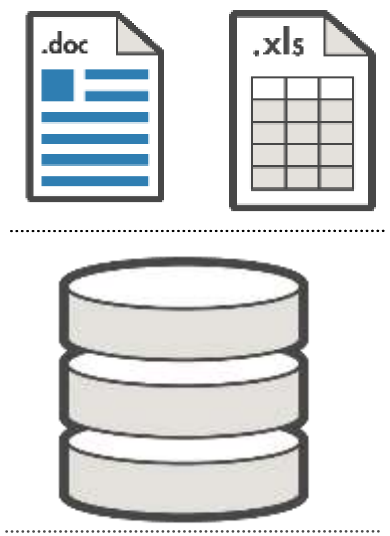
Integrate with requirements tools and author requirements



- Import from:
  - Word / Excel
  - IBM® Rational® DOORS®
  - ReqIF™ standard
- Update synchronizes changes from source
- Edit and add further details to import R2019a
- Author requirements
- Export ReqIF R2019a
  - Enables roundtrip with external tools

Roundtrip workflow with external tools thru ReqIF

External Requirements

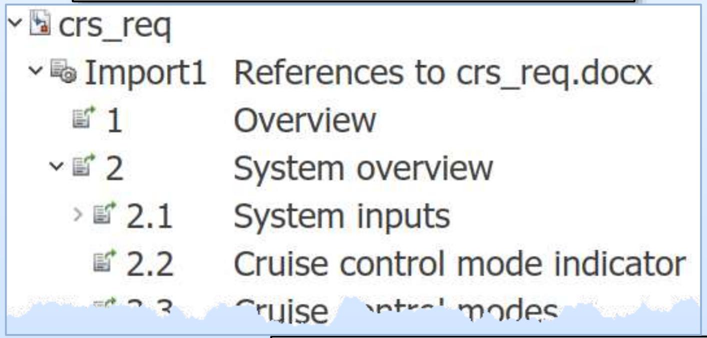


Requirements Managements Tools

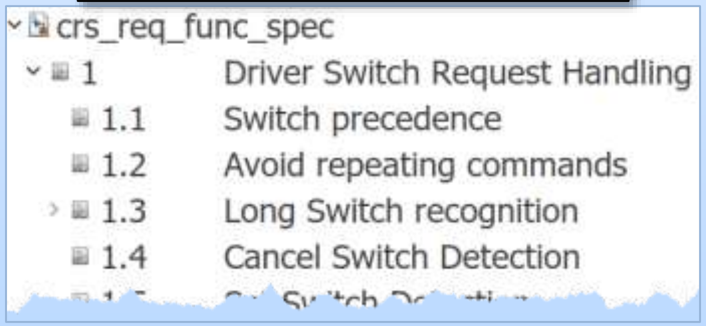
R2019a

Simulink Requirements

External Requirements



Authored Requirements



- Import from:
  - Word / Excel
  - IBM® Rational® DOORS®
  - ReqIF™ standard
- Update synchronizes changes from source
- Edit and add further details to import R2019a
- Author requirements
- Export ReqIF R2019a
  - Enables roundtrip with external tools

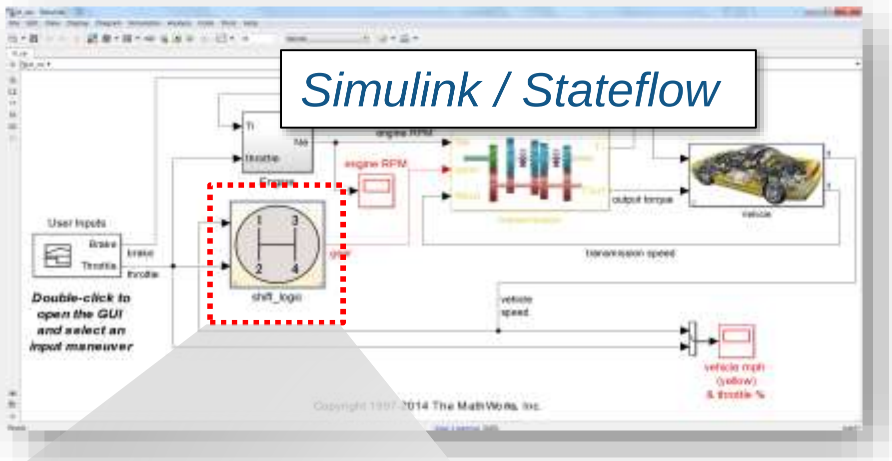
Requirements Verification with Simulink

Requirements

TransmissionReq

1	Transmission Operating Modes
1.1	Reverse cannot be entered from drive
1.2	Engine only starts in Park

Implemented By



Verified By

Test Case

Inputs

MAT / Excel file (input)

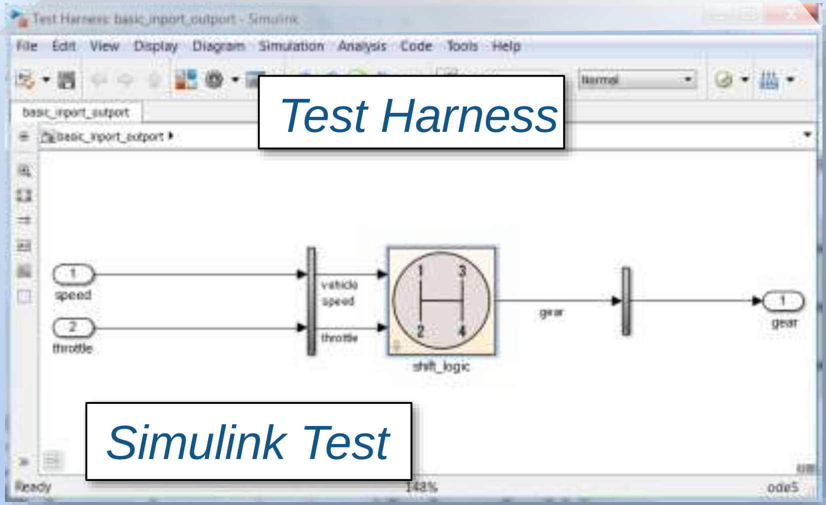
Scenario

Signal 1

Signal Editor

1
2 Output
3

Test Sequence



Assessments

MAT / Excel File (baseline)

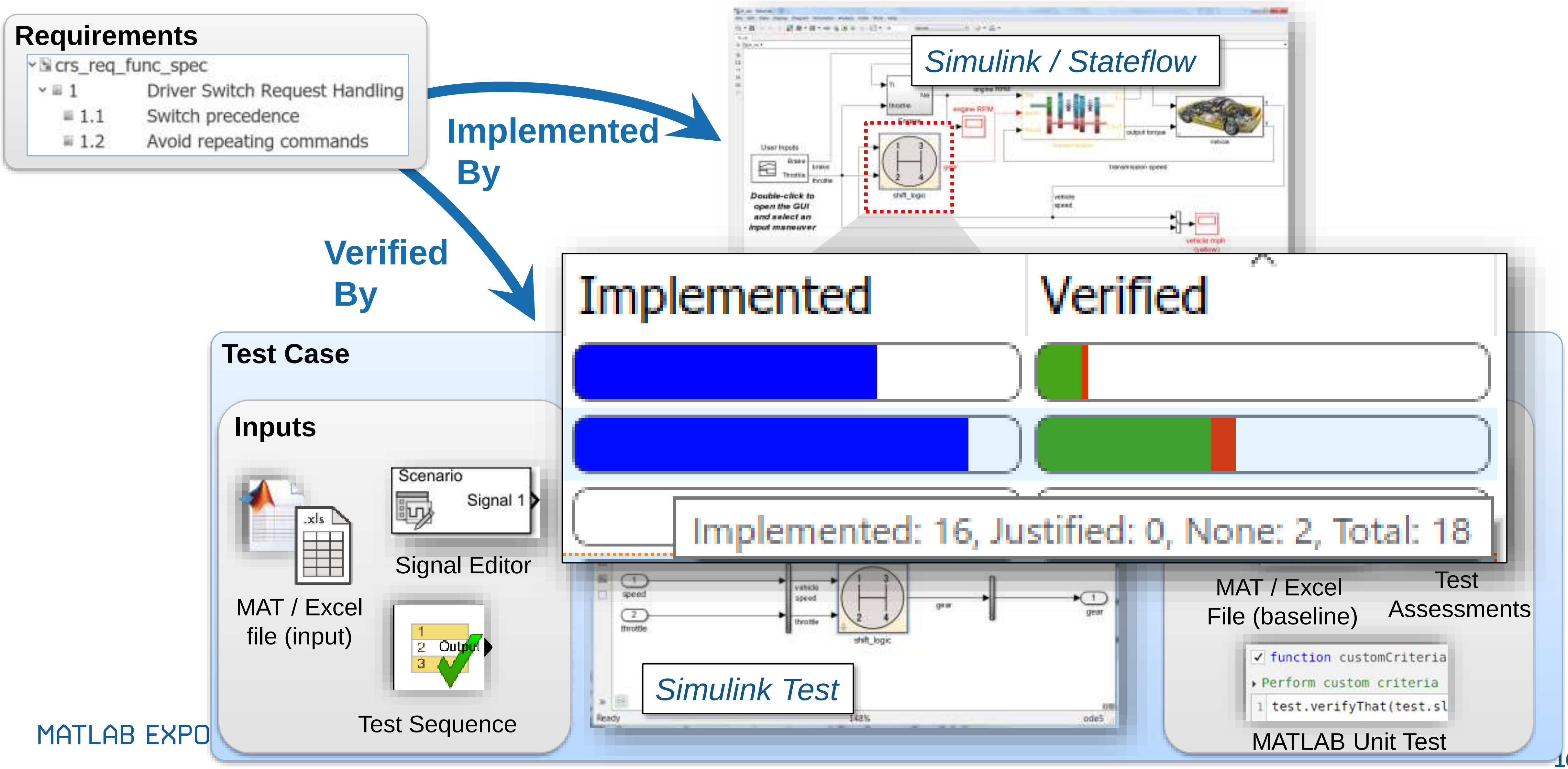
1
Input
3

Test Assessments

```
function customCriteria
Perform custom criteria
1 test.verifyThat(test.sl
```

MATLAB Unit Test

Requirements Verification with Simulink



Example: Verifying Heat Pump Controller Requirements

1 Requirements for the basic Heatpump Controller

Temperature difference is defined as the difference between the room and the set temperature. The controller shall turn the fan on when the temperature difference has reached a certain level, to circulate the air. The controller shall turn the heatpump on when the temperature difference has reached another level, to heat or cool the space.

1.1 Idle when Temperature in Range

If the temperature difference is less than 1 degrees, the system shall be idle with all signals off.

1.2 Activate Fan

The fan shall activate when the temperature difference is greater than or equal to 1 degrees.

1.3 Activate Heat Pump

The pump shall activate when the temperature difference is greater than or equal to 2 degrees for more than 2 seconds and stay active for at least 2 seconds.

1.3.1 Cool Mode

If the room temperature is greater than the set temperature, the system shall cool the space.

1.3.2 Heat Mode

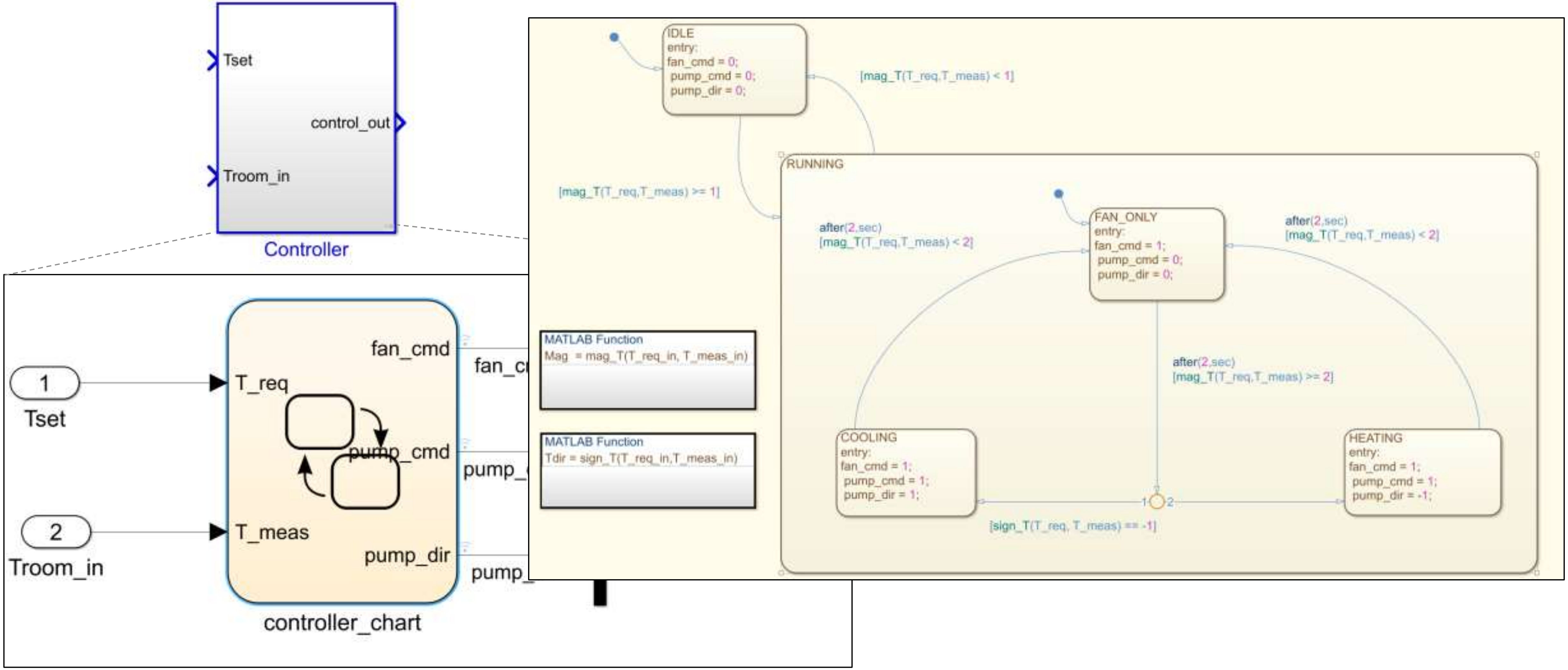
If the room temperature is less than the set temperature, the system shall heat the space.

1.4 Max Tempera

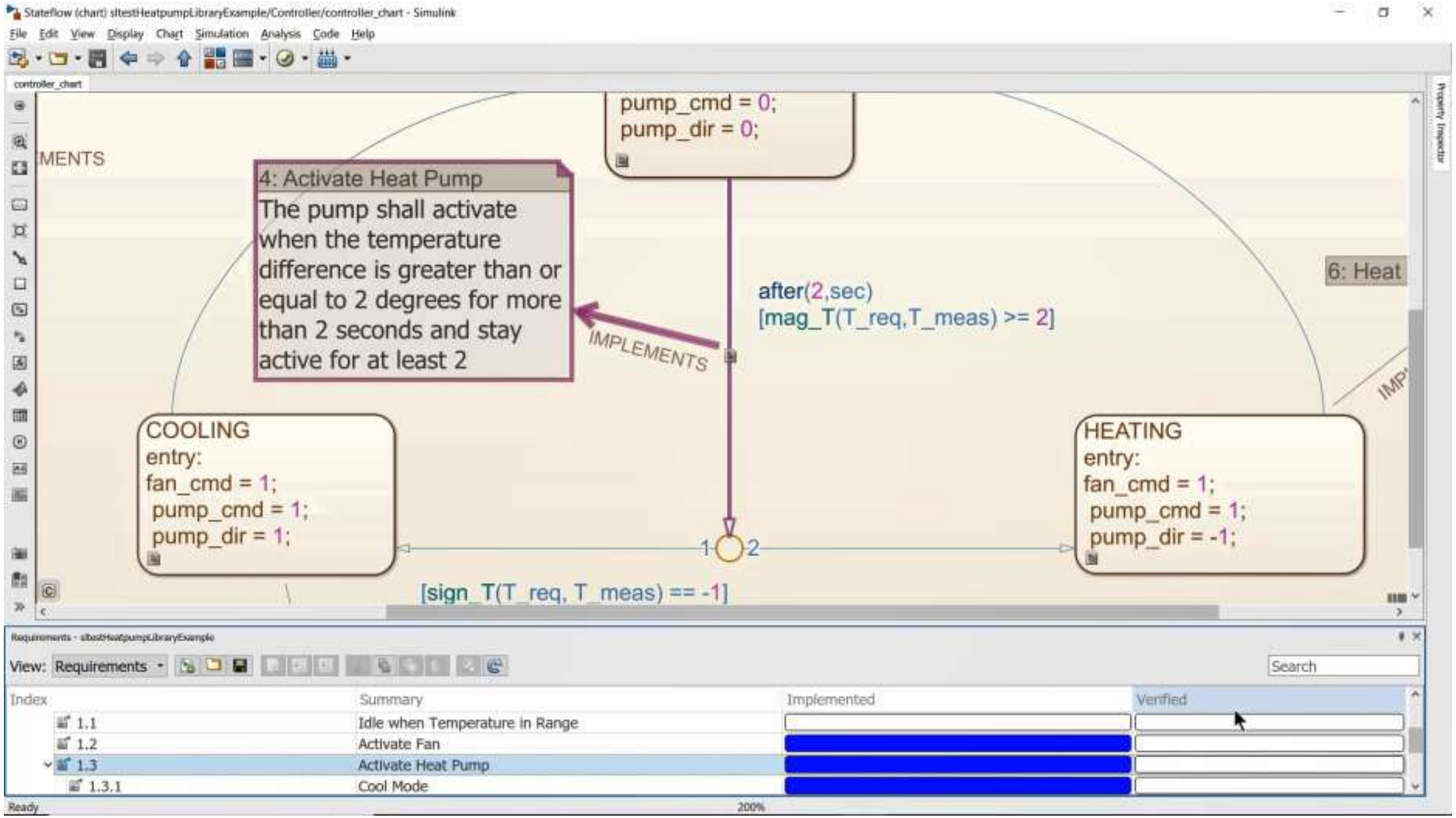
Requirements in DOORS

The difference between the room temperature and the set temperature should never exceed 6 degrees

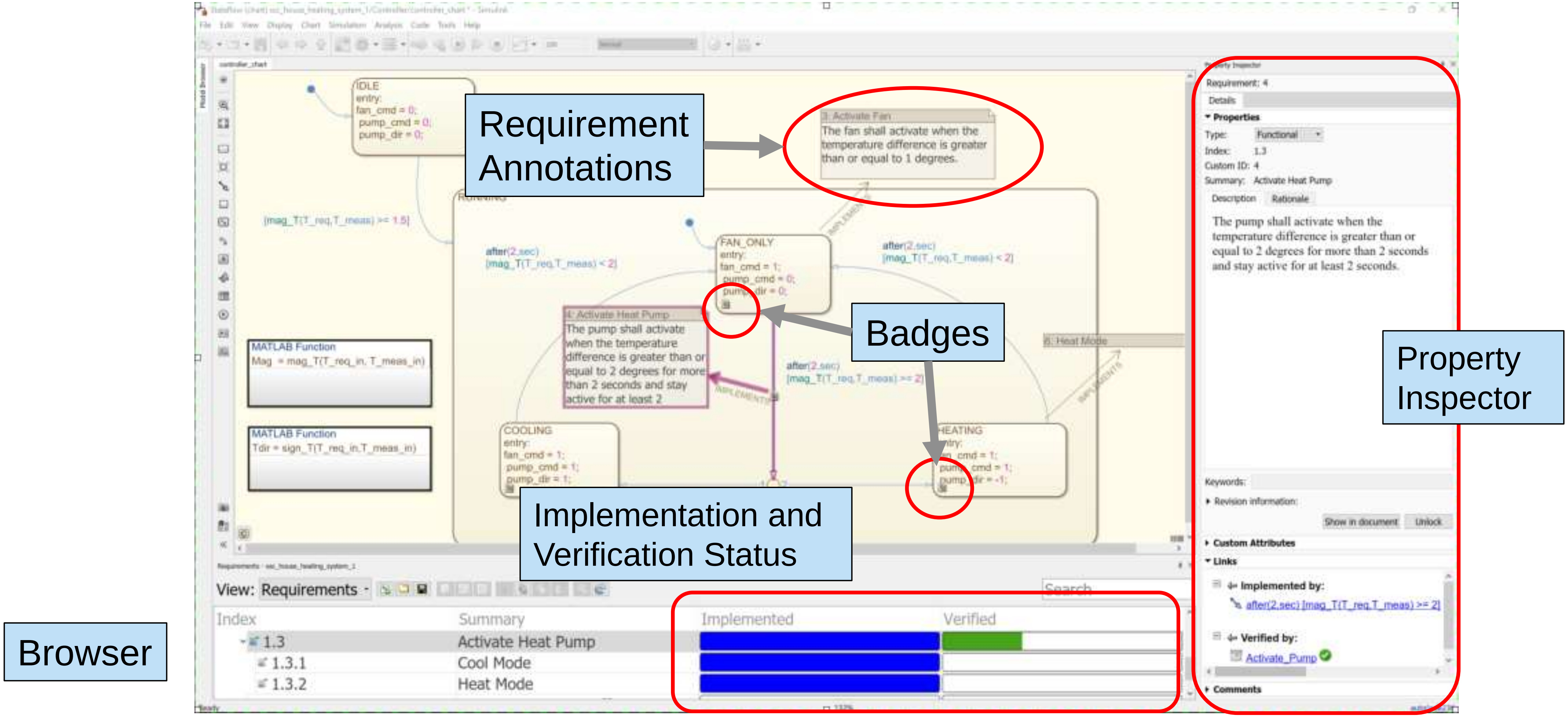
Example: Heat Pump Controller Implementation



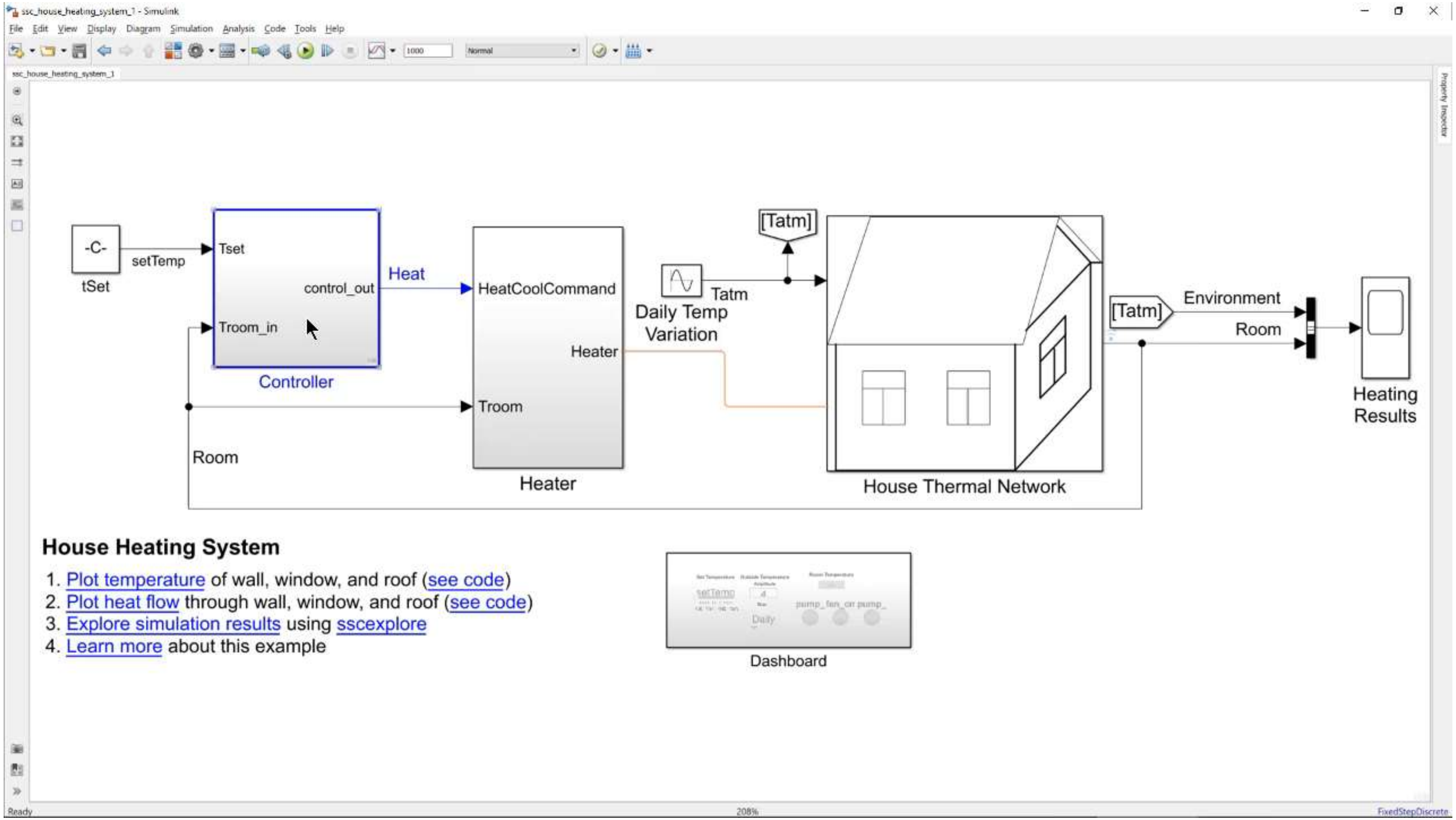
Link requirements to implementation in model



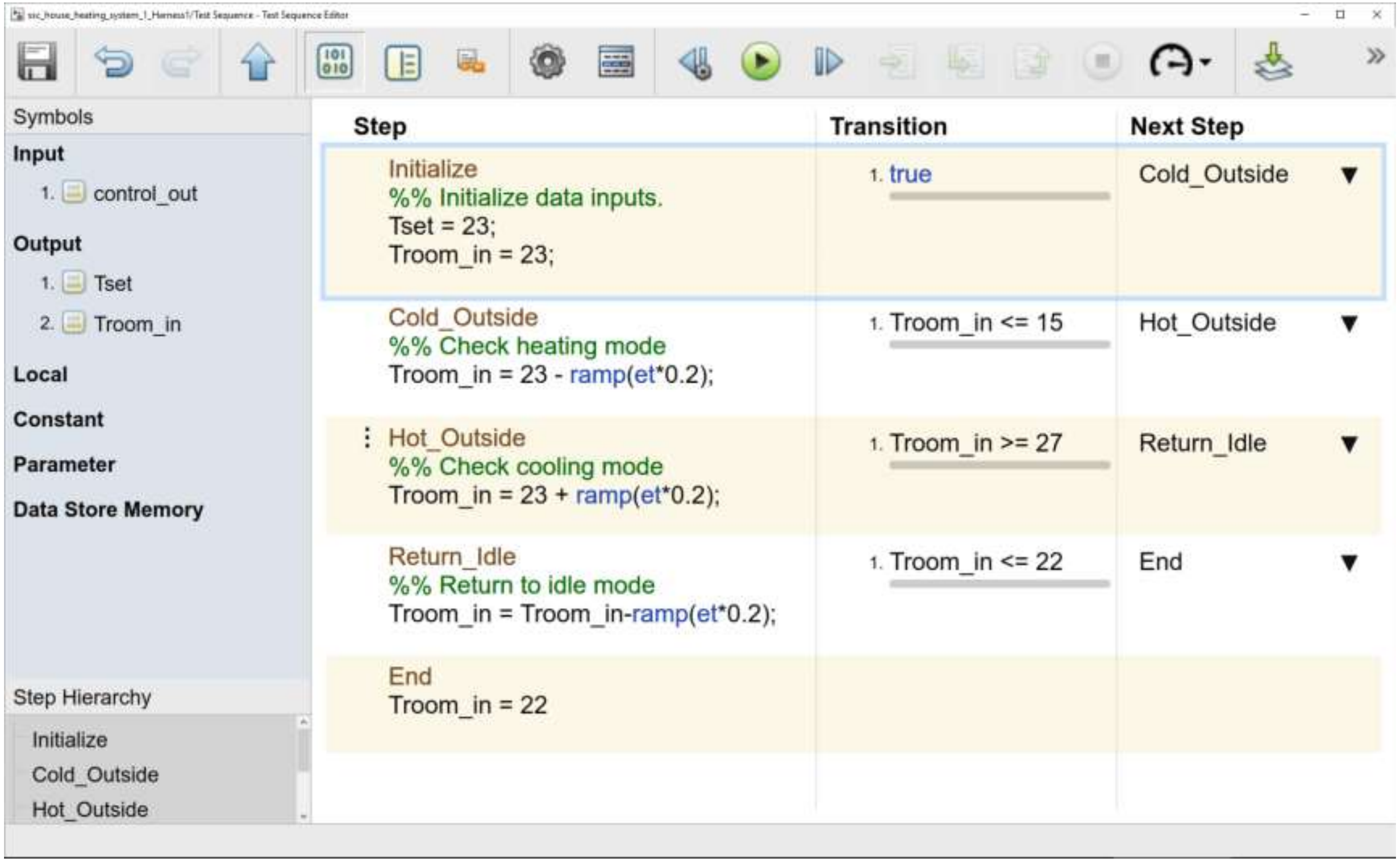
Work with Model and Requirements with Requirements Perspective



Isolate Component Under Test with Test Harness



Test Sequence Block: Step-based and temporal test sequences



Test Assessments: Formalize and execute requirements

R2019a

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

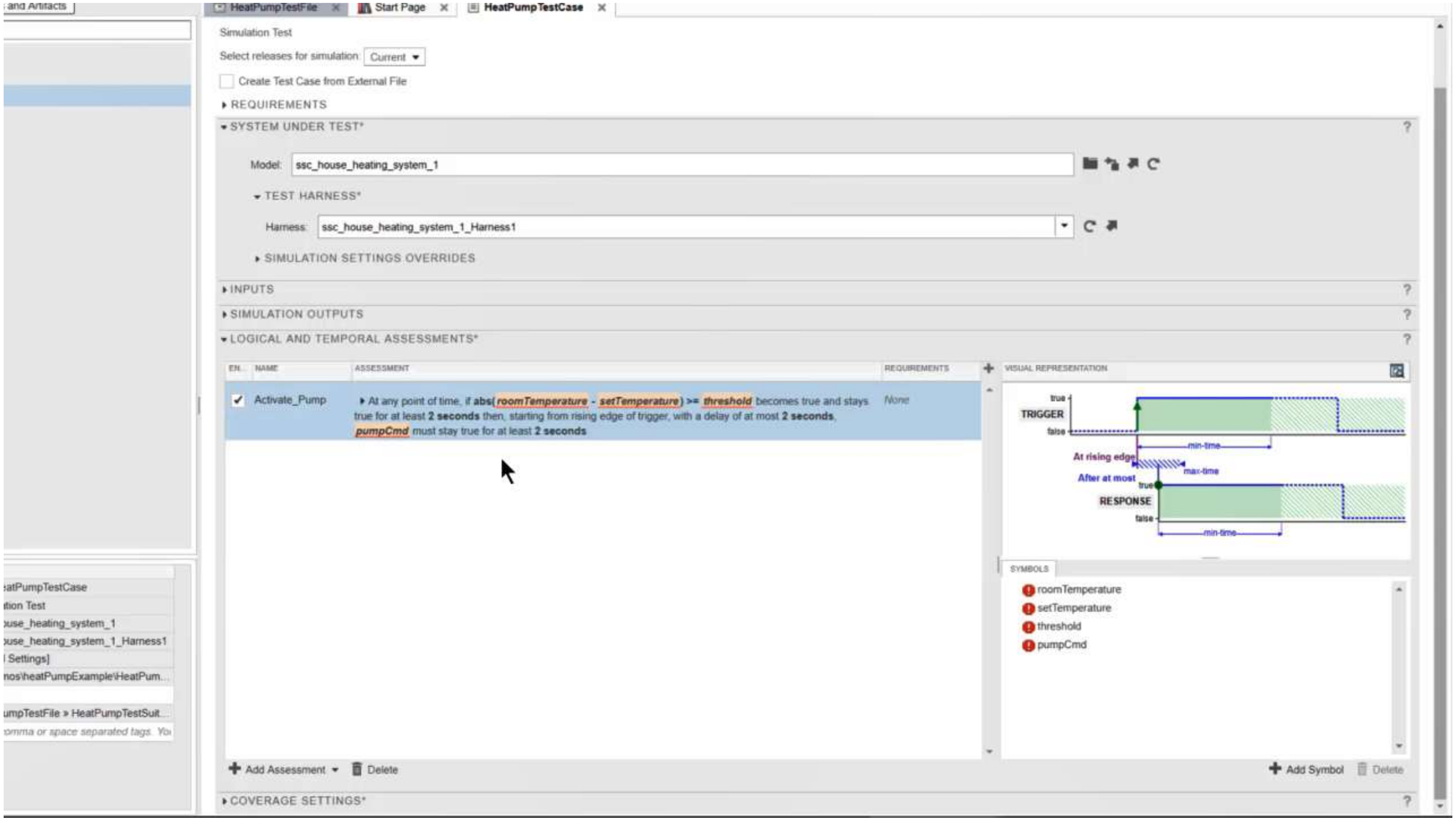
$$(|x_1 - x_2| \geq x_3)^{\overset{\varepsilon}{\leftarrow}} \wedge \square_{[0,t_1)}(|x_1 - x_2| \geq x_3) \rightarrow \square_{[0,t_2)} x_4$$

Hard to formalize

MTL logic

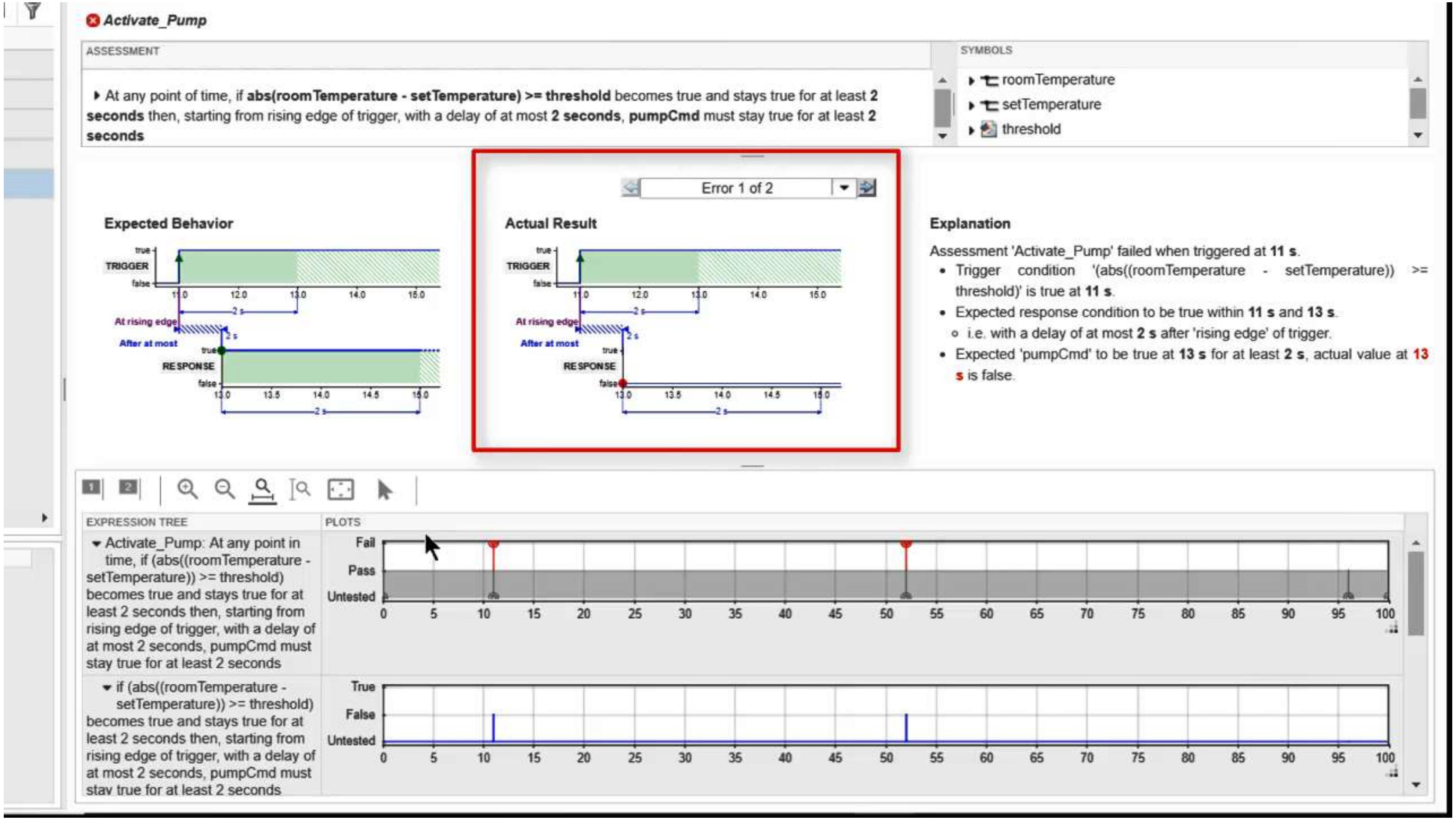
Author temporal assessments using form based editor

R2019a

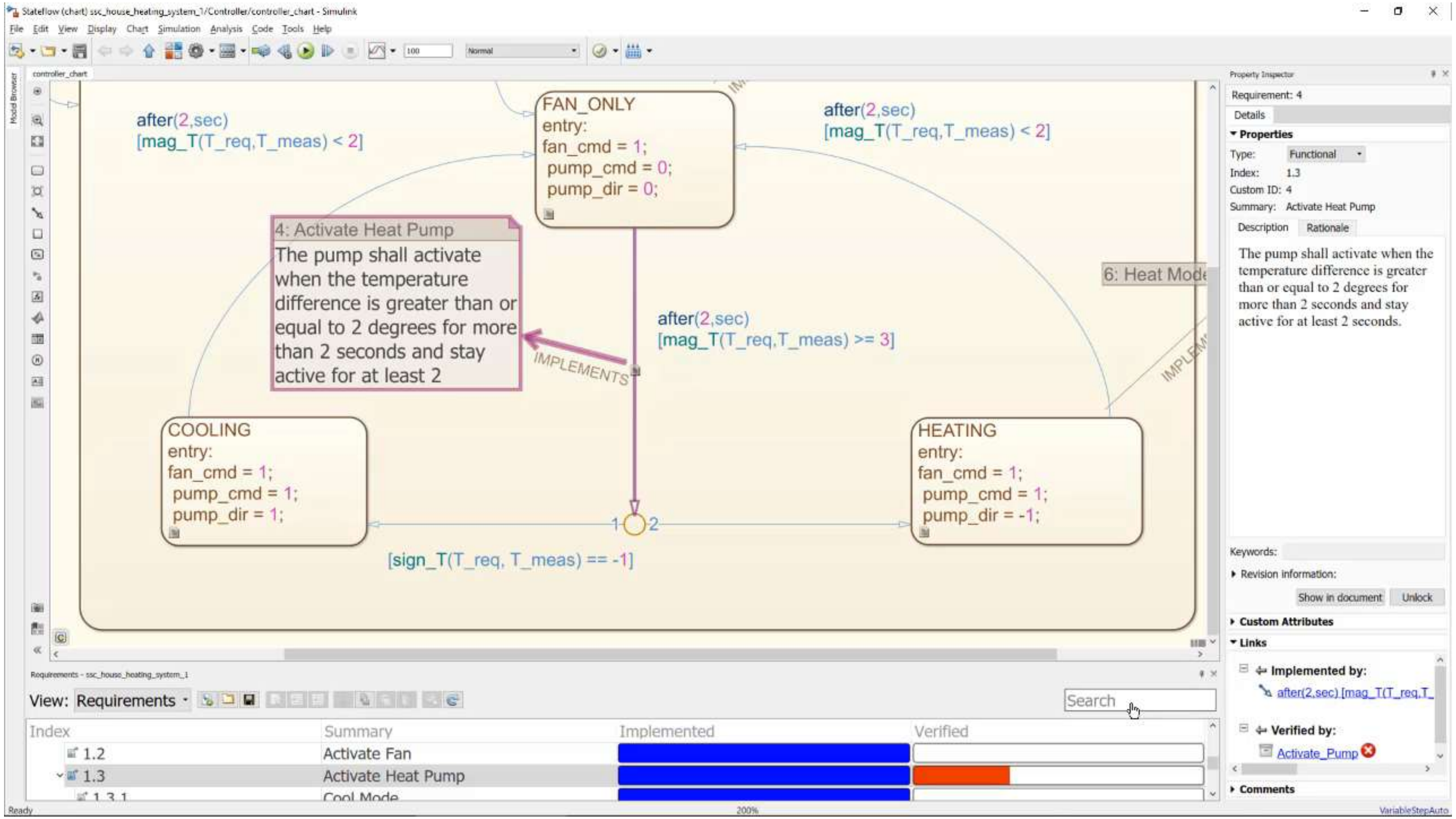


Execute assessments to verify requirements

R2019a



Locate implementation of requirement using link

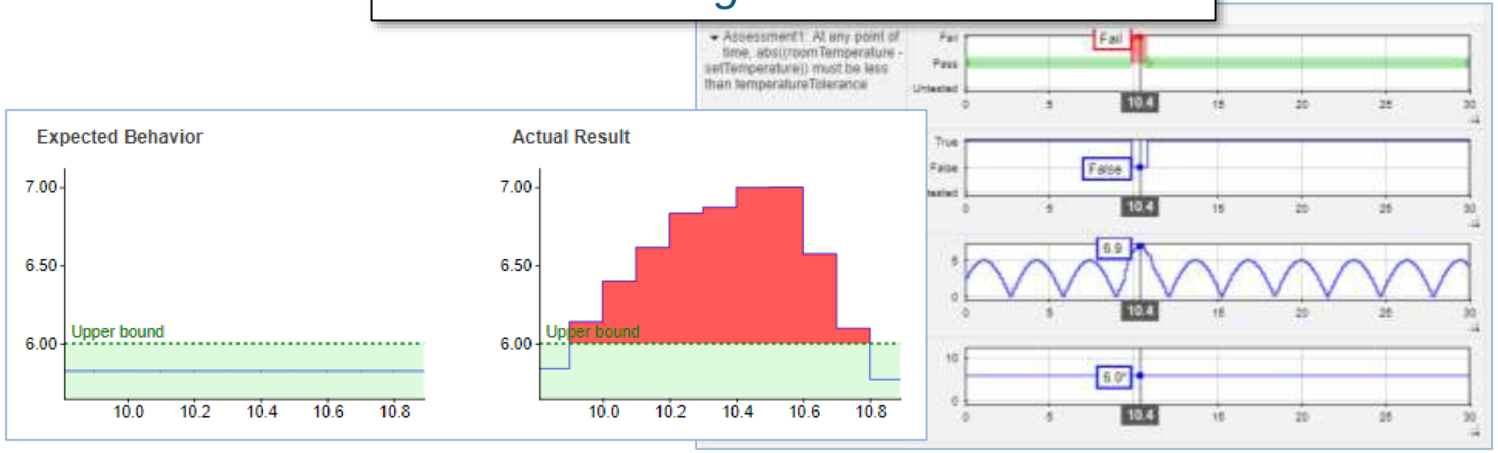


Translate textual requirements into unambiguous Temporal Assessments

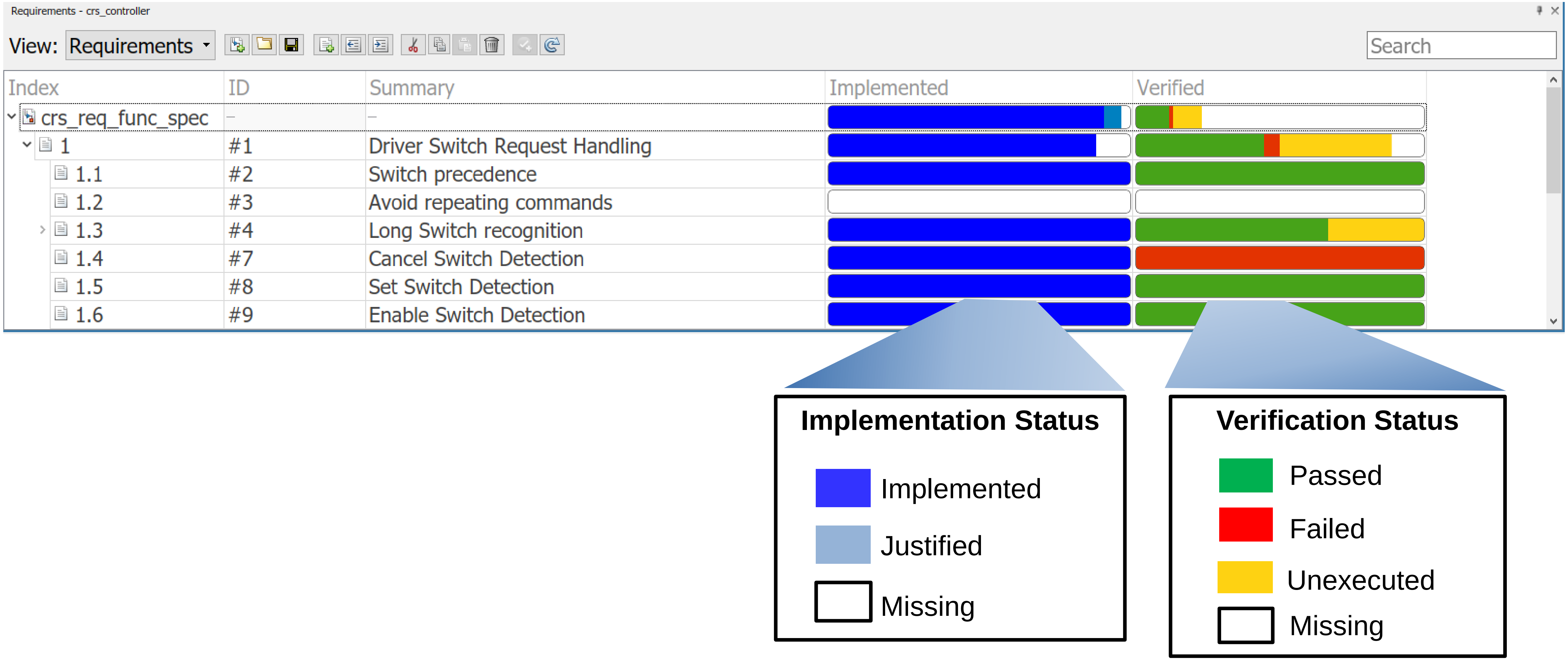
- Compose assessments using form based editor
- View assessments as English-like sentence
- Review and debug temporal assessment results
- Link to requirements

Temporal Assessment Editor

View and Debug Assessment Results

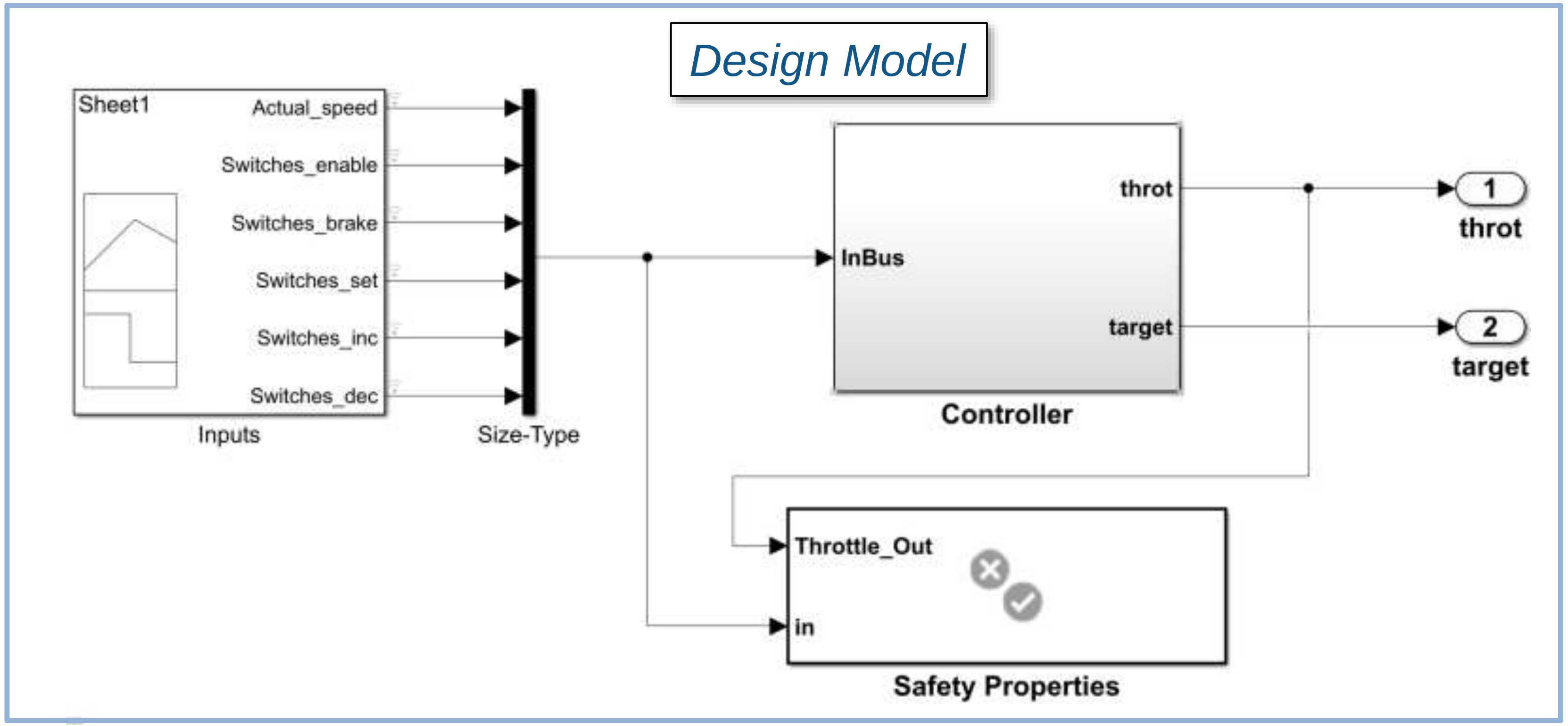


Track Implementation and Verification



Observers: Separate test/verification logic from design

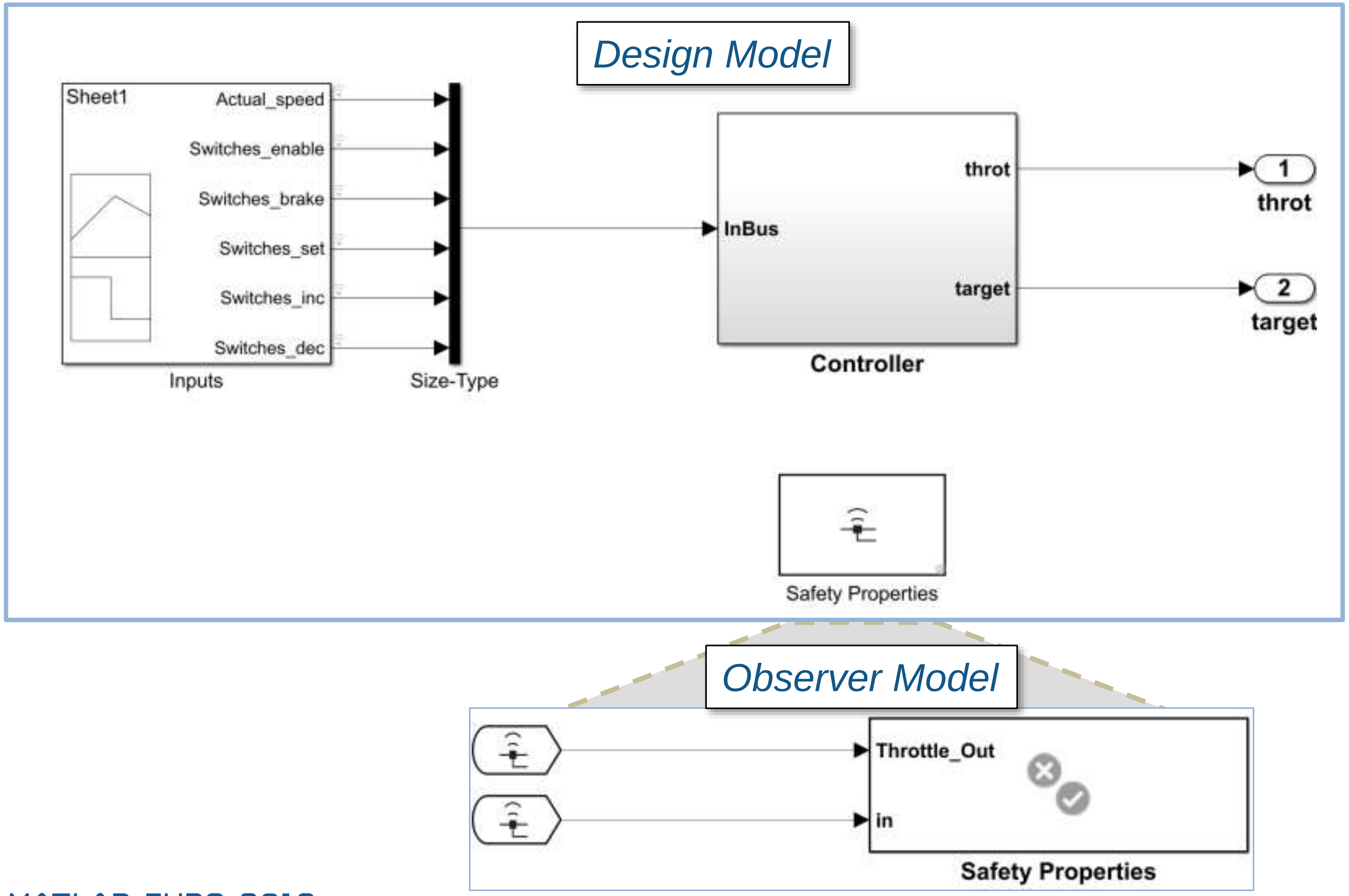
R2019a



- Access nested signals without signal lines or changing dynamic response
- Avoid modifying interface for testing
- Simplify design and test by avoiding additional signal lines

Observers: Separate test/verification logic from design

R2019a



- Access nested signals without signal lines or changing dynamic response
- Avoid modifying interface for testing
- Simplify design and test by avoiding additional signal lines

LS Automotive Reduces Development Time for Automotive Component Software with Model-Based Design

Challenge

Shorten development times for embedded control software used in automotive switches and components

Solution

Use Model-Based Design to model controller designs, run simulations, verify customer specifications, and generate error-free production code

Results

- Specification errors detected early
- Proven development approach established
- 80% Coding errors eliminated

An LS Automotive door area unit.

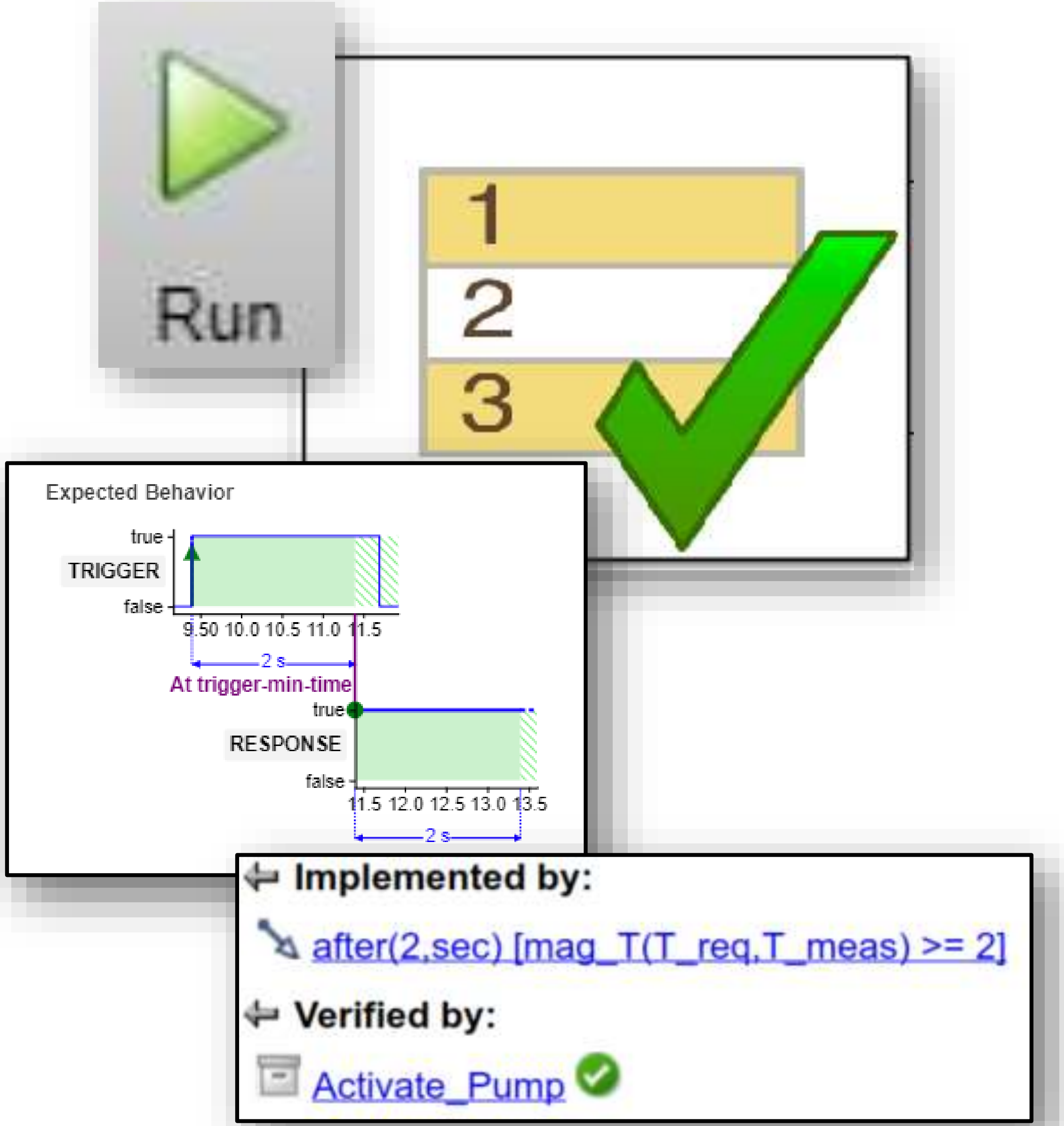
"By enabling us to analyze requirements quickly, reuse designs from previous products, and eliminate manual coding errors, Model-Based Design has reduced development times and enabled us to shorten schedules to meet the needs of our customers."

- MyoungSuk Ko, LS Automotive

Link to user story

Summary

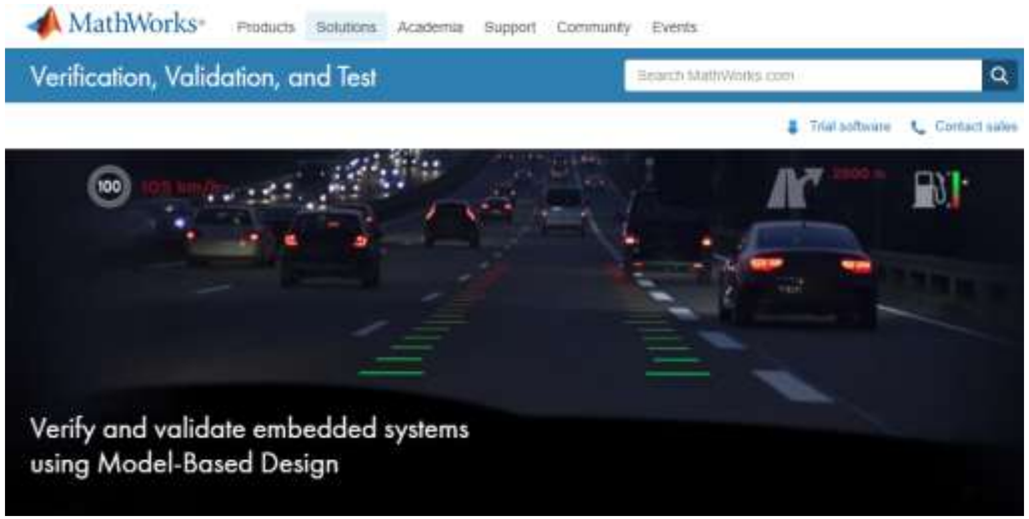
- Verify and validate requirements earlier
- Identify inconsistencies in requirements by using unambiguous assessments
- Traceability from requirements to design and test



Learn More

Key products covered in this presentation:

- [Simulink Requirements]
- [Simulink Test]



Learn more at Verification, Validation and Test Solution Page:

mathworks.com/solutions/verification-validation.html