MATLAB EXPO 2019

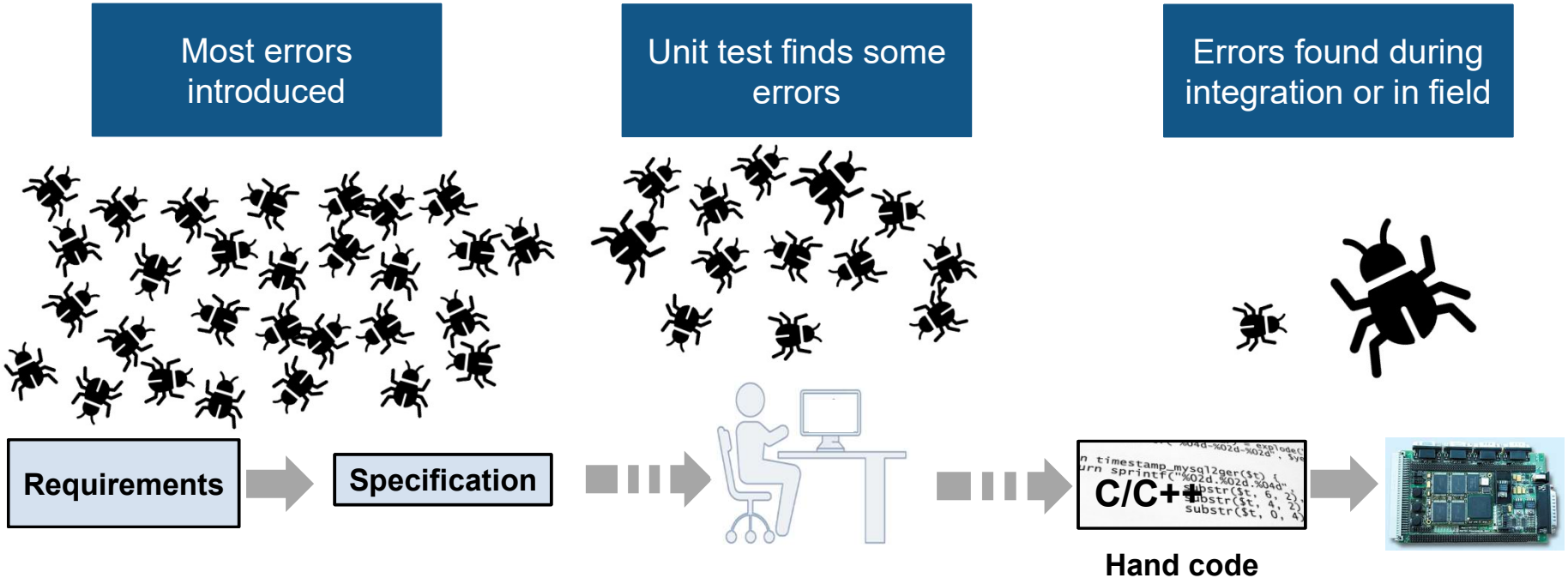
모델기반설계를 이용한 요구사항 기반 검증의 단순화

Application Engineer
홍 혁 기 부장

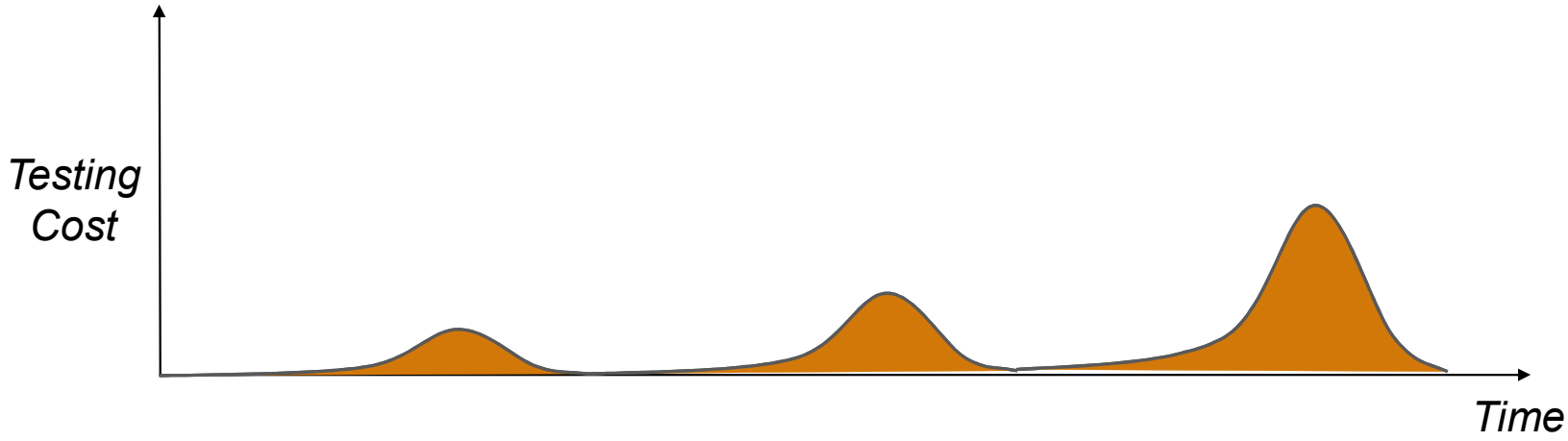
Key takeaways

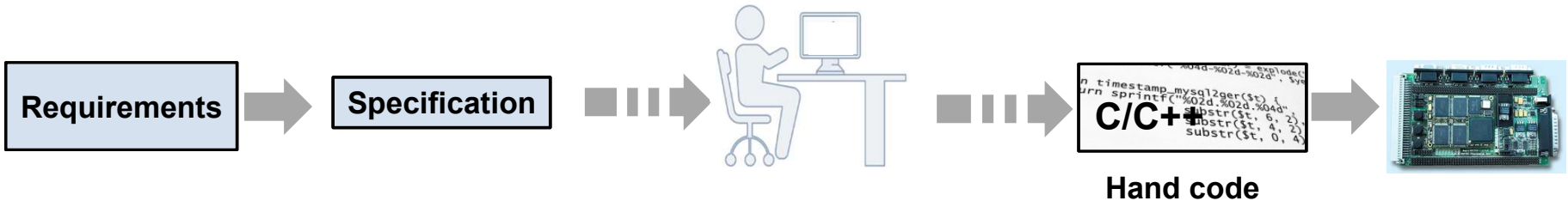
- Verify and validate requirements earlier
- Identify inconsistencies in requirements by using unambiguous assessments
- Traceability from requirements to design and test

Challenge: Errors introduced early but found late



Cost of finding errors increases over time





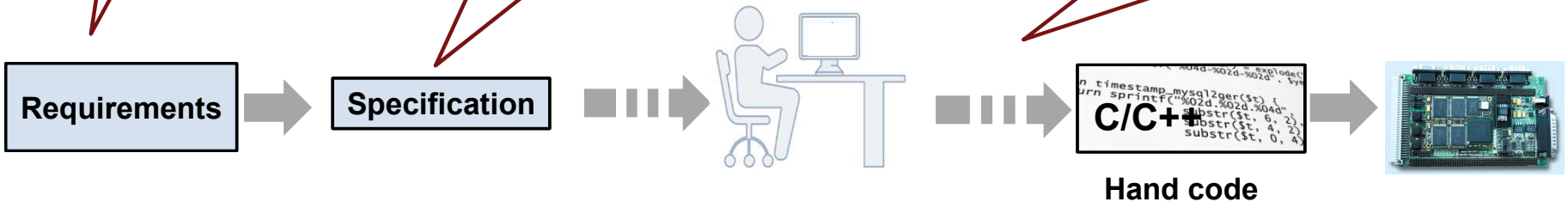
Challenges with requirements based verification

Are **all** requirements implemented?

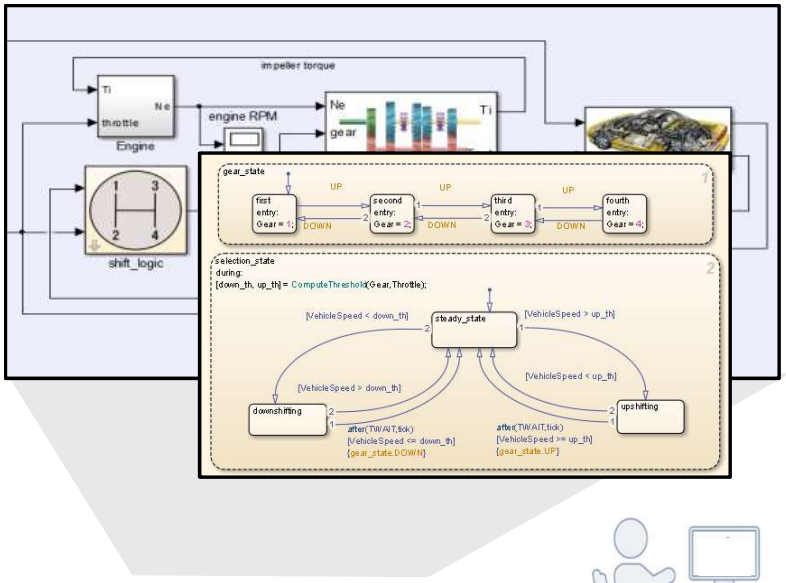
Is requirement interpreted correctly?

Is the implementation functioning correctly?

How to avoid modifying the design for test?

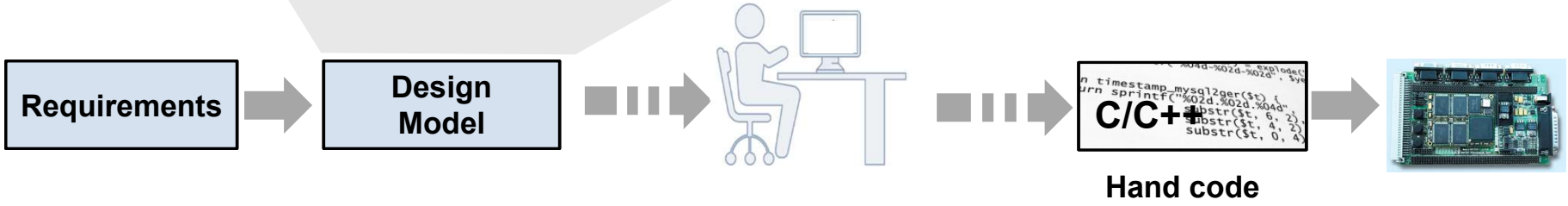


Simulink models for specification

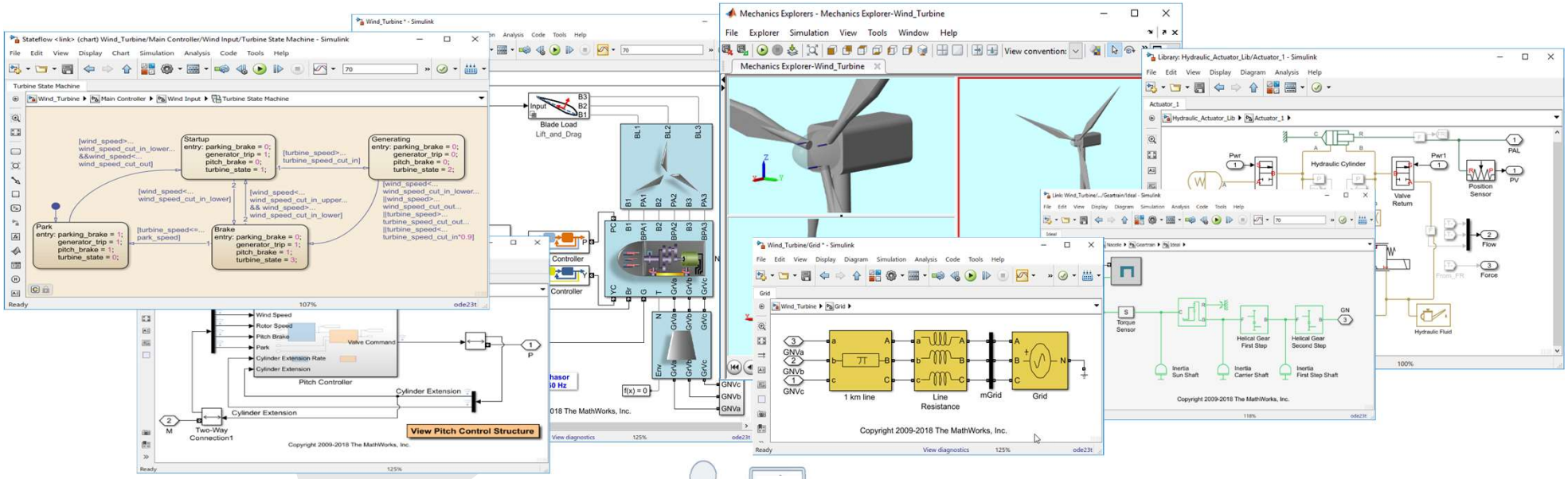


Model-Based Design enables:

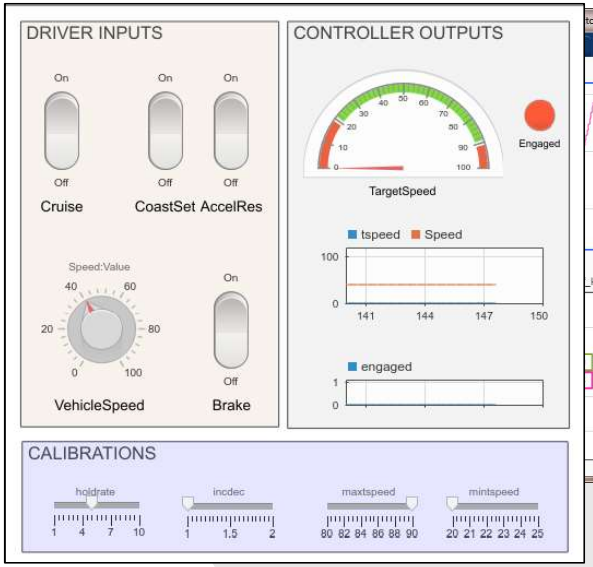
- *Early testing to increase confidence in your design*
- *Delivery of higher quality software throughout the workflow*

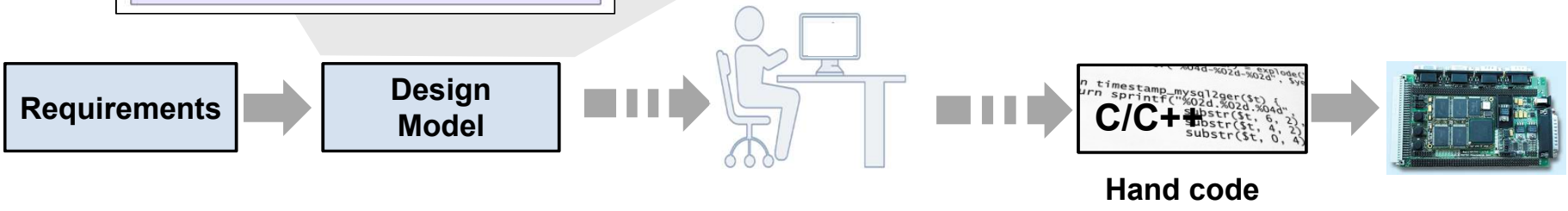


Multiple languages to describe complex systems

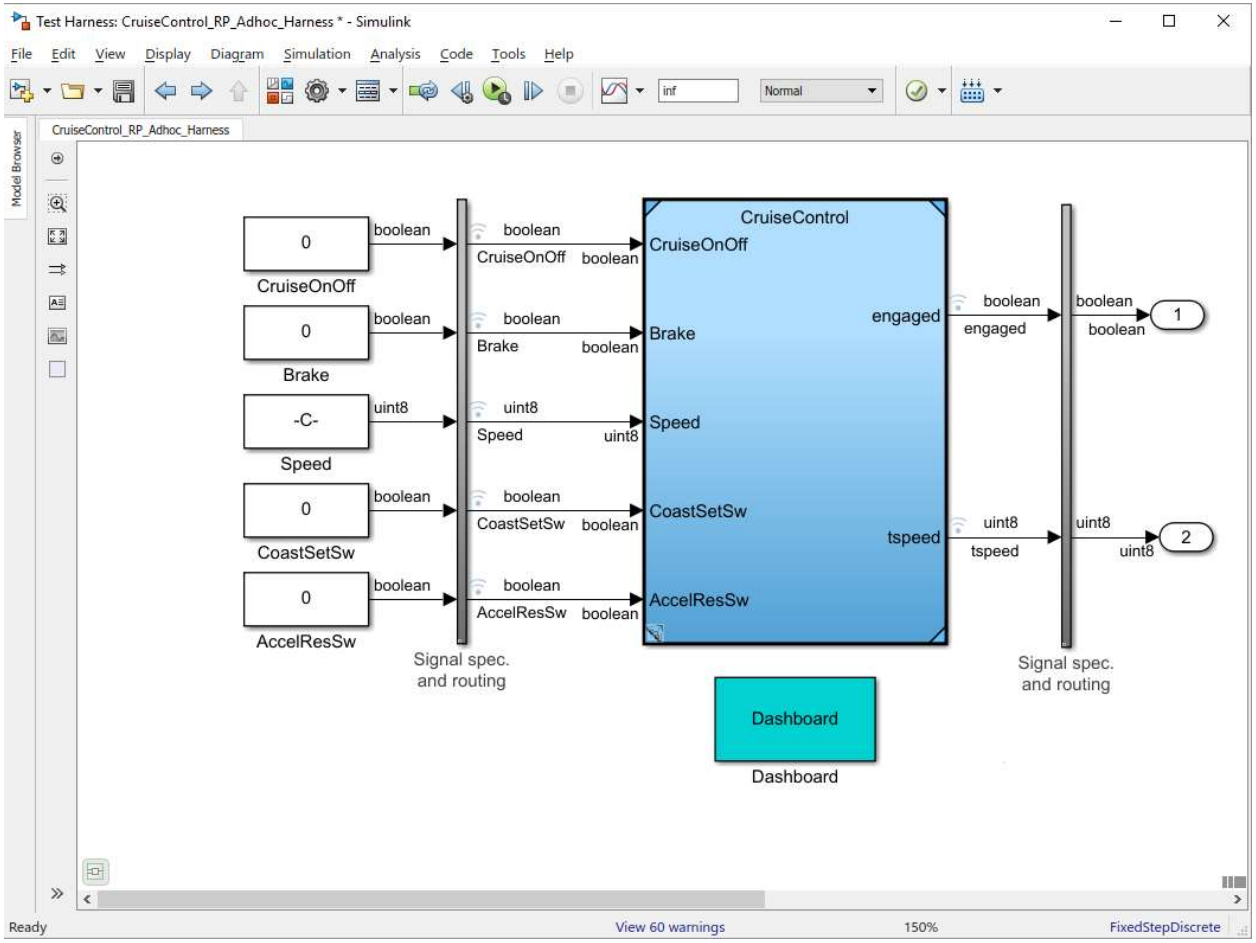


Ad-Hoc Testing: Explore behavior and design alternatives

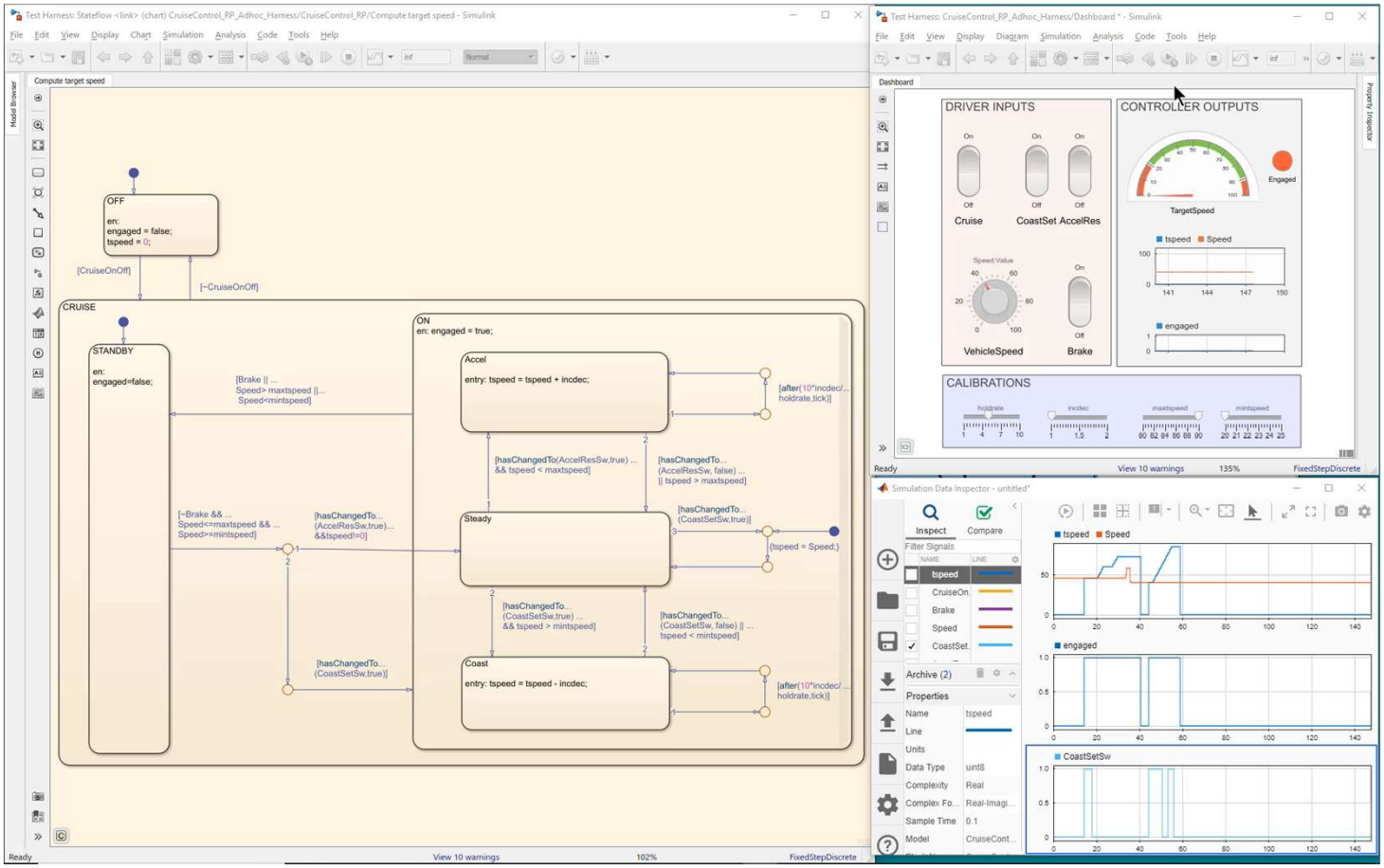




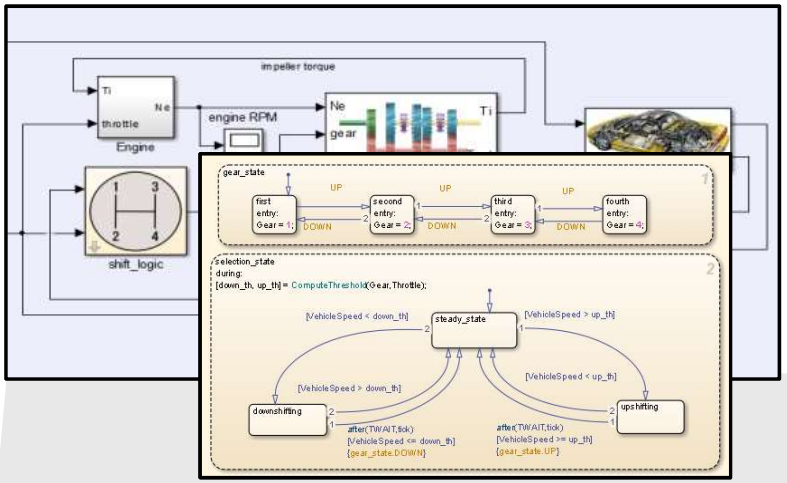
Validate behavior earlier with simulation



Validate Behavior Earlier with Simulation



Complete Model Based Design



Simulink Models

Requirements

Design Model

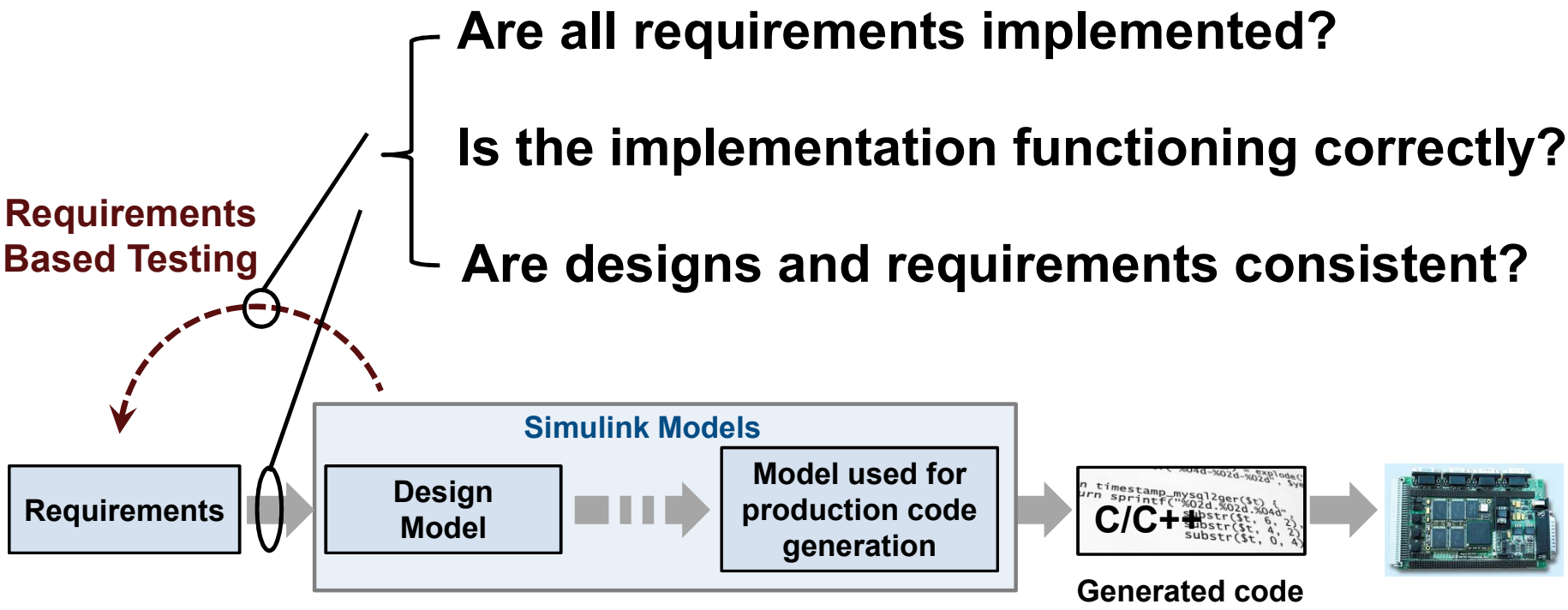
Model used for production code generation

Code Generation

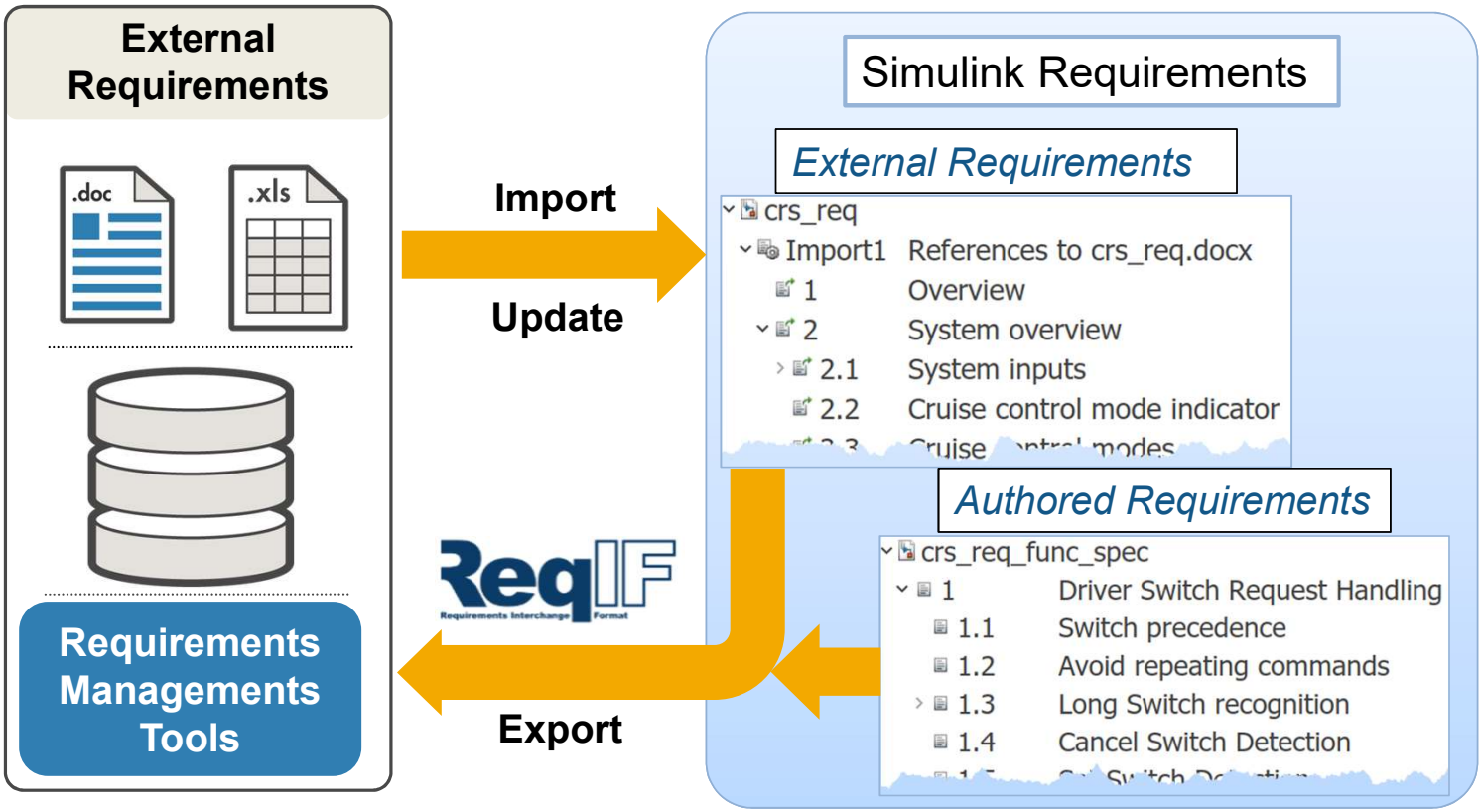
C/C++

Generated code

Systematically verify requirements

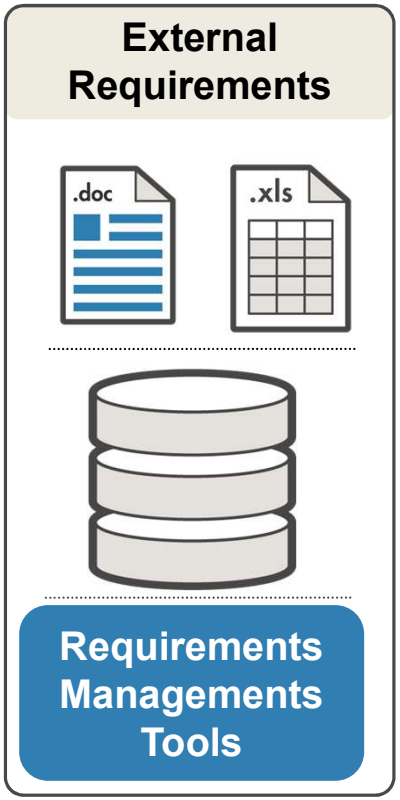


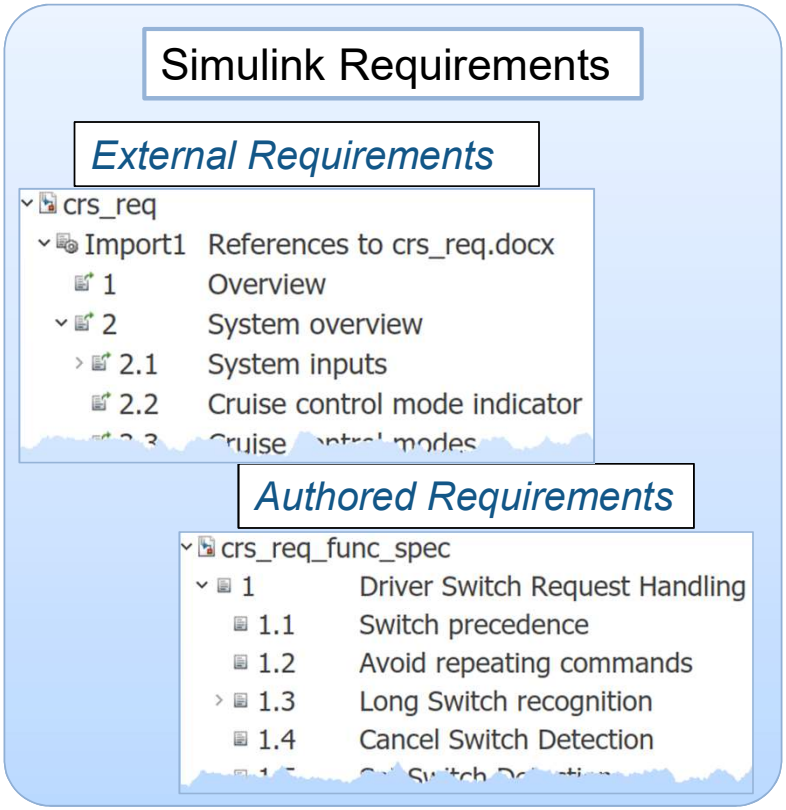
Integrate with requirements tools and author requirements



- 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





- 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

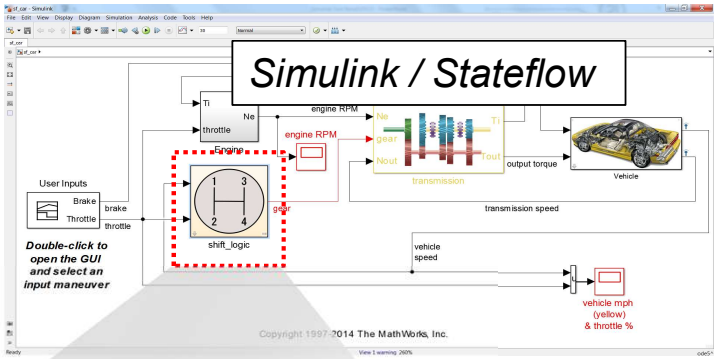
Requirements Verification with Simulink

Requirements

TransmissionReq

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

Implemented By



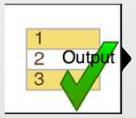
Verified By

Test Case

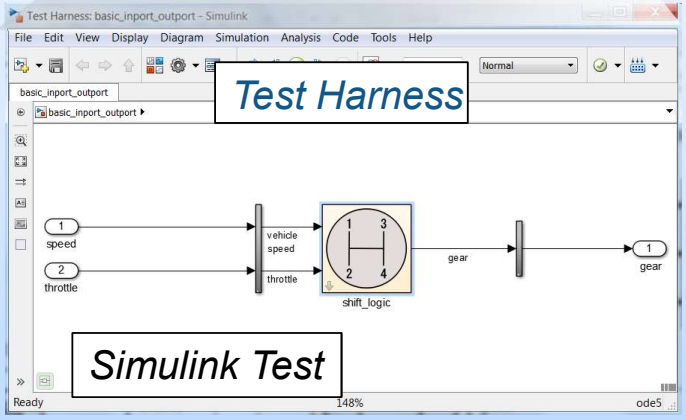
Inputs

MAT / Excel file (input)

Signal Editor

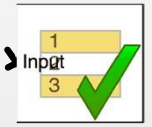


Test Sequence

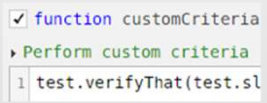


Assessments

MAT / Excel File (baseline)



Test Assessments



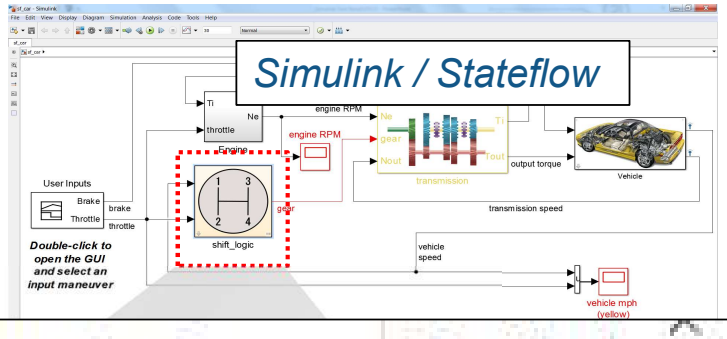
MATLAB Unit Test

Requirements Verification with Simulink

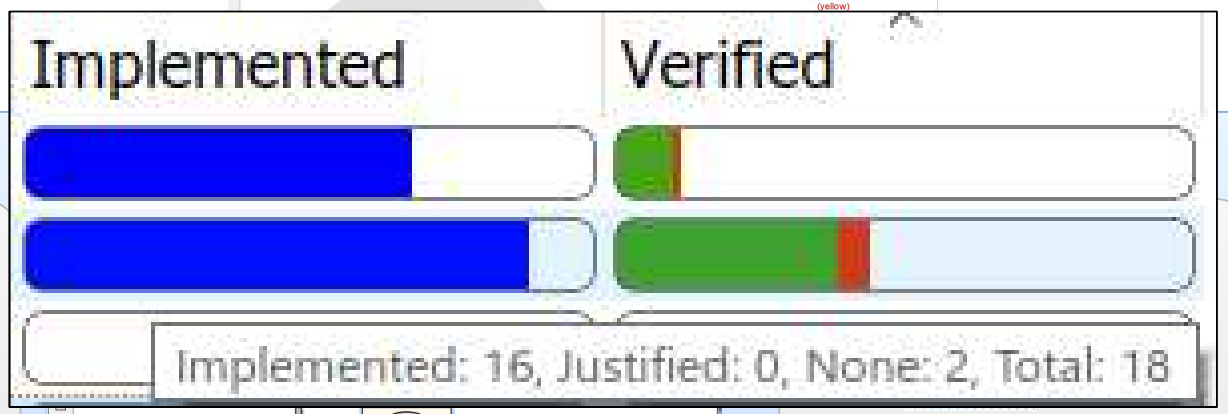
Requirements

- crs_req_func_spec
  - 1 Driver Switch Request Handling
    - 1.1 Switch precedence
    - 1.2 Avoid repeating commands

Implemented By



Verified By

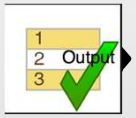


Test Case

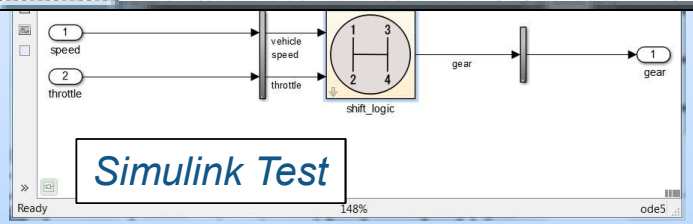
Inputs

MAT / Excel file (input)

Signal Editor

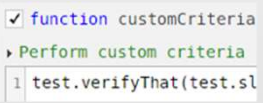


Test Sequence



MAT / Excel File (baseline)

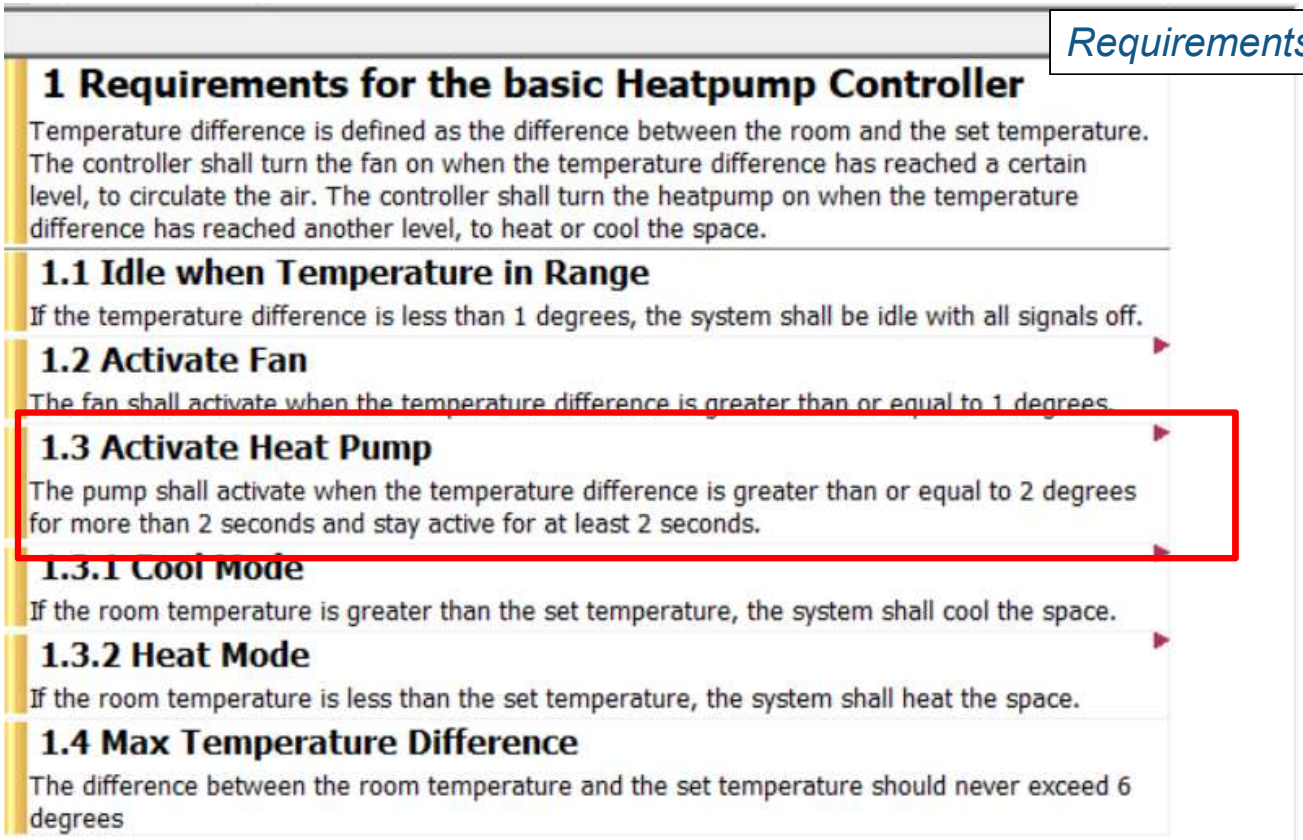
Test Assessments



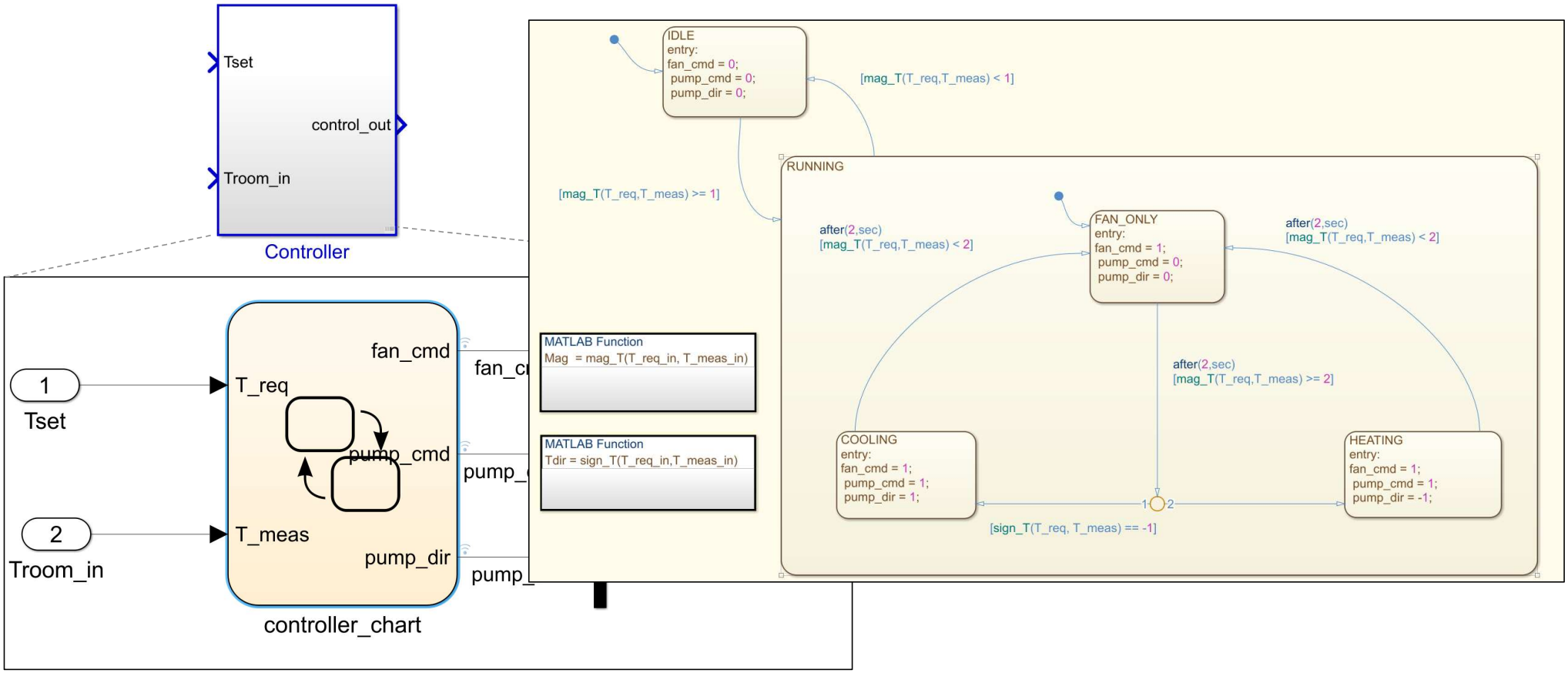
MATLAB Unit Test

Example: Verifying Heat Pump Controller Requirements

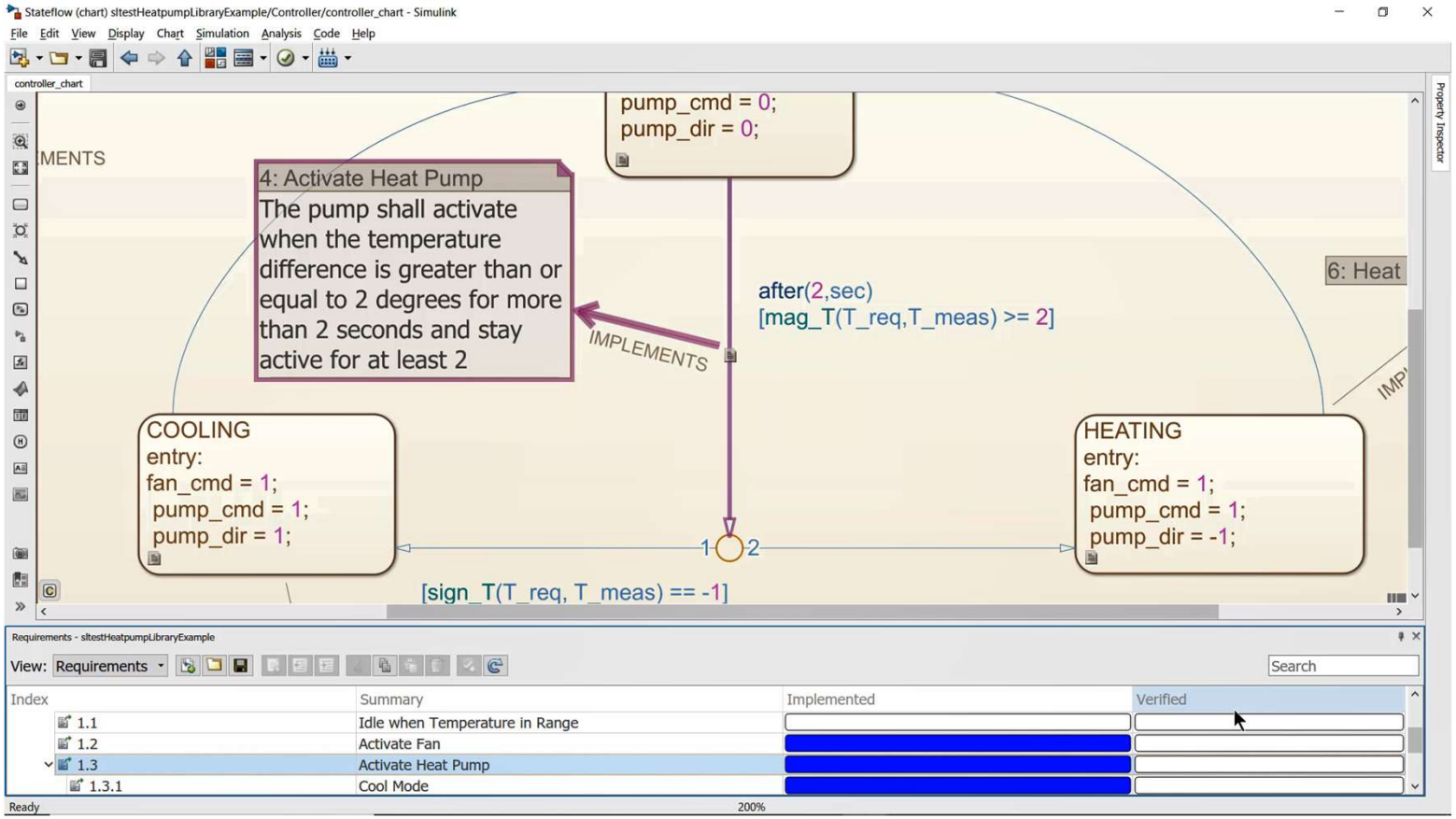
Requirements in DOORS



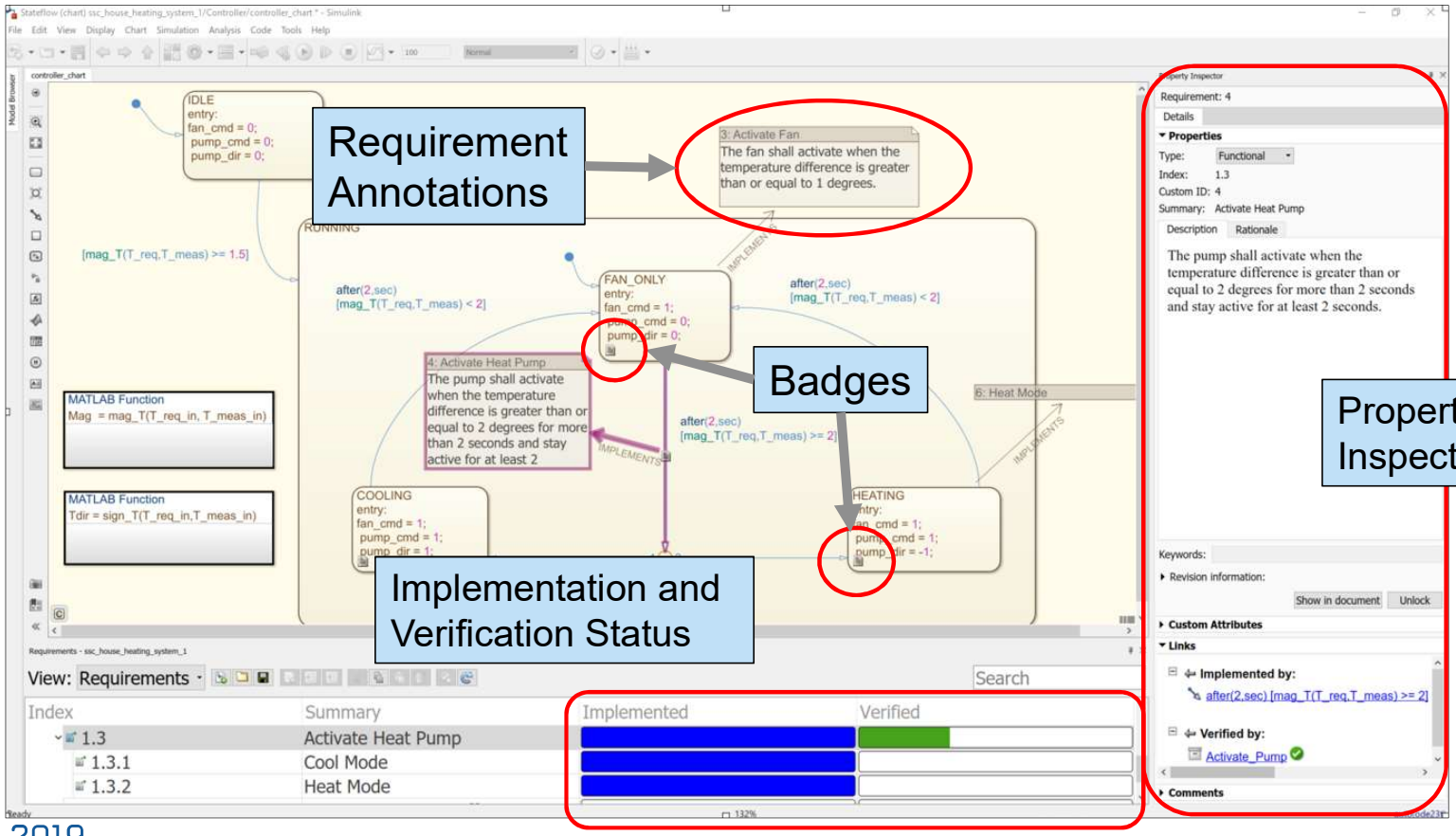
Example: Heat Pump Controller Implementation



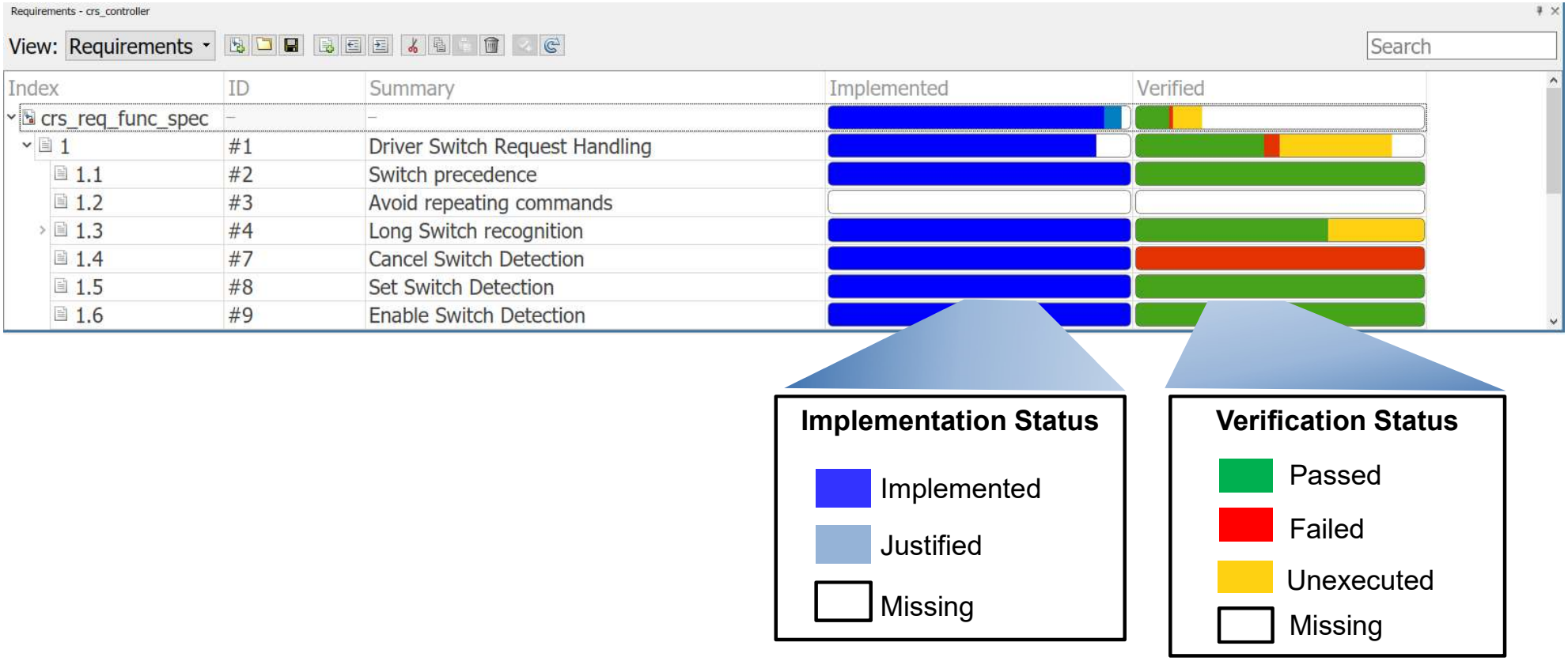
Link requirements to implementation in model



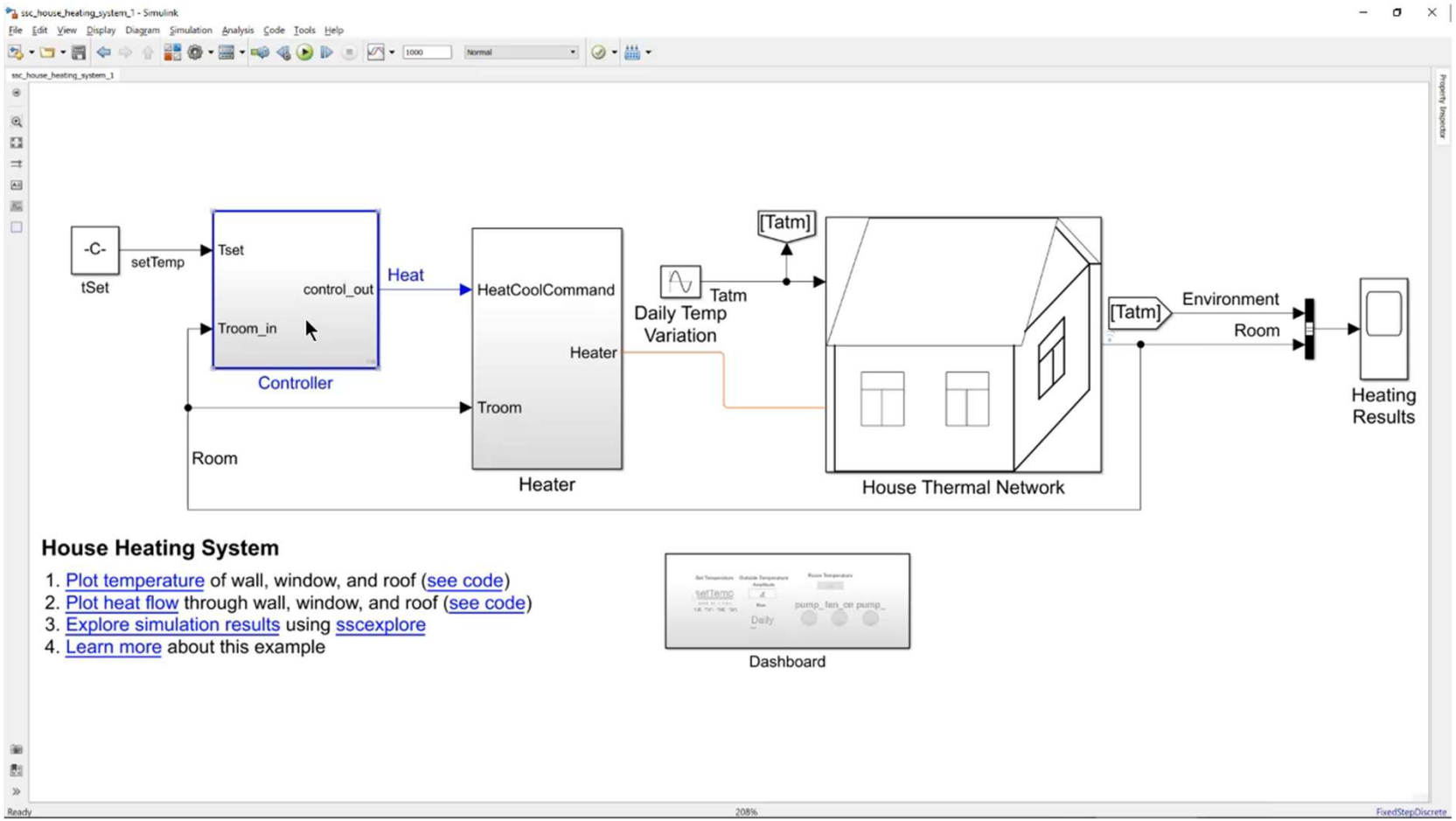
Work with Model and Requirements with Requirements Perspective



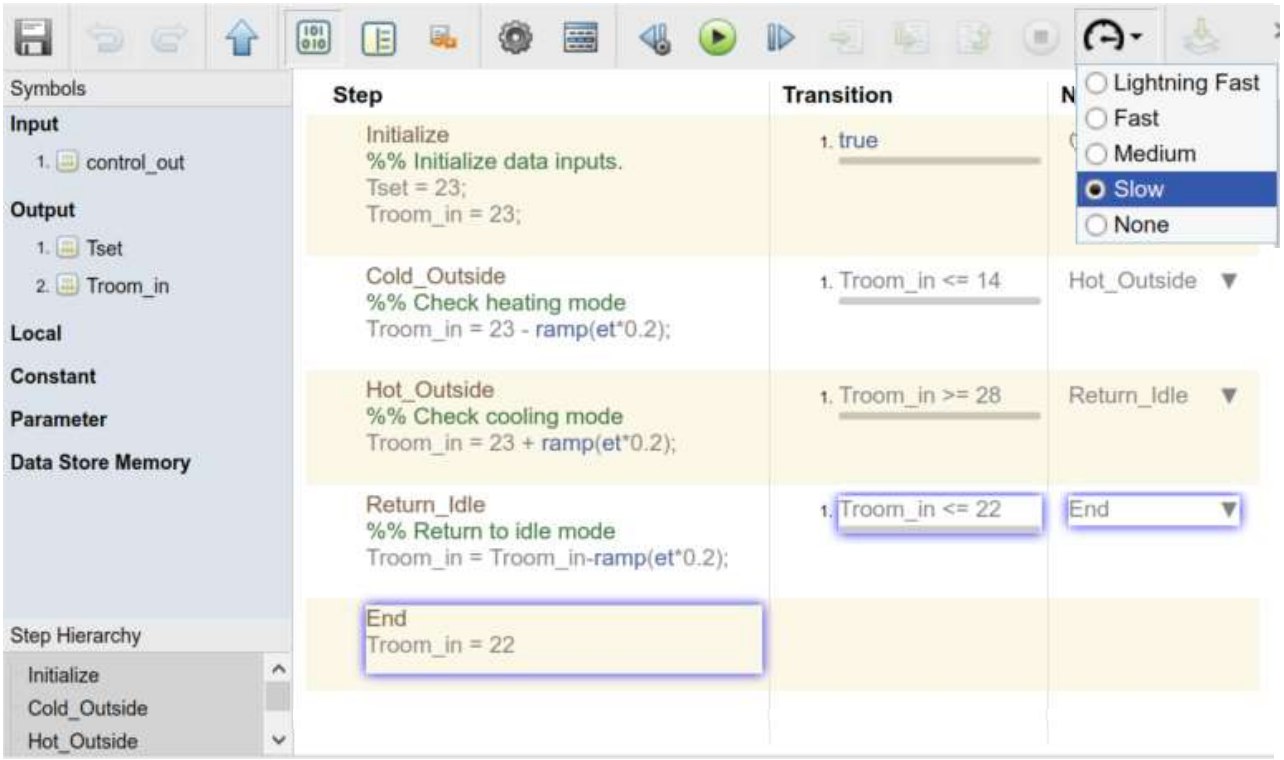
Track Implementation and Verification



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

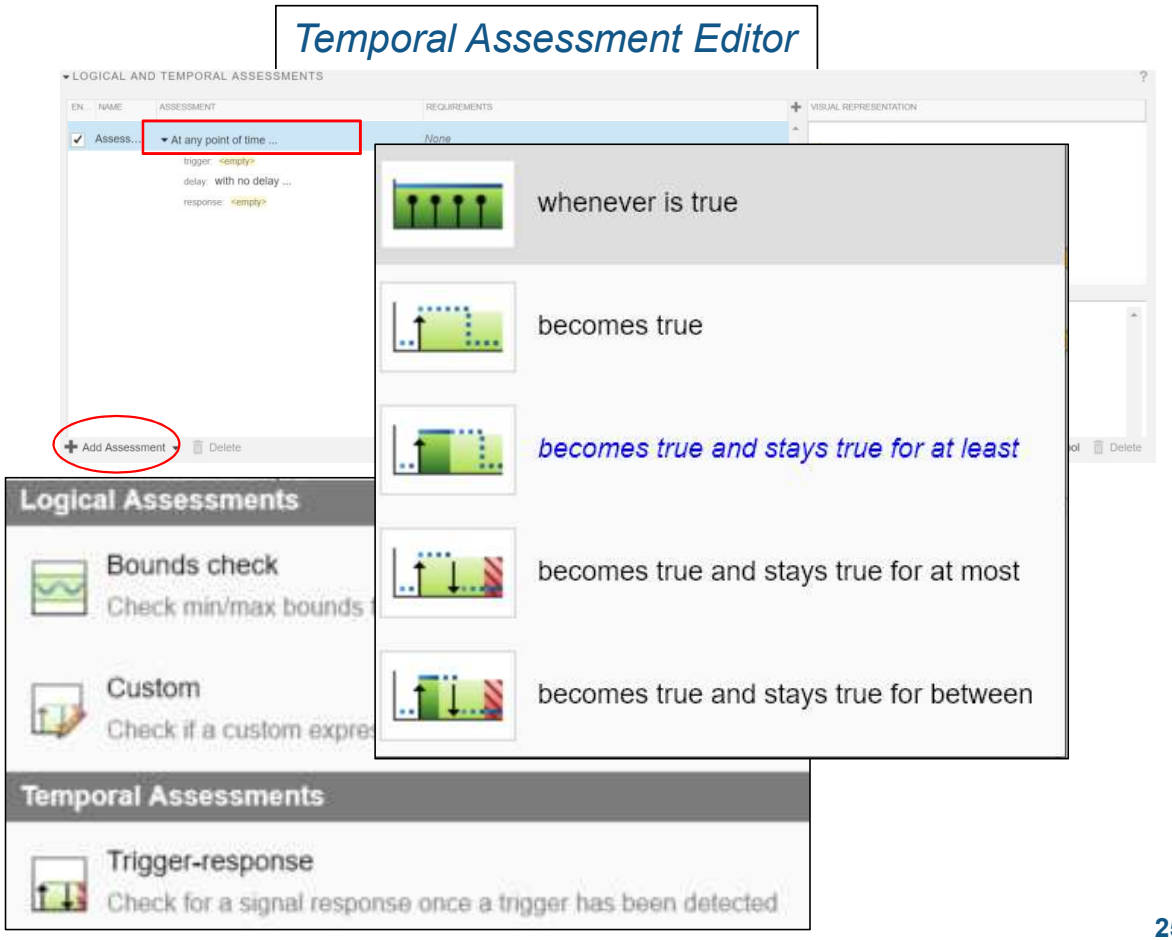
$$(|x_1 - x_2| \geq x_3)^{\overleftarrow{\varepsilon}} \wedge \square_{[0,t_1)}(|x_1 - x_2| \geq x_3) \rightarrow \square_{[0,t_2)} x_4$$

Hard to formalize

MTL logic

Translate textual requirements into unambiguous Temporal Assessments

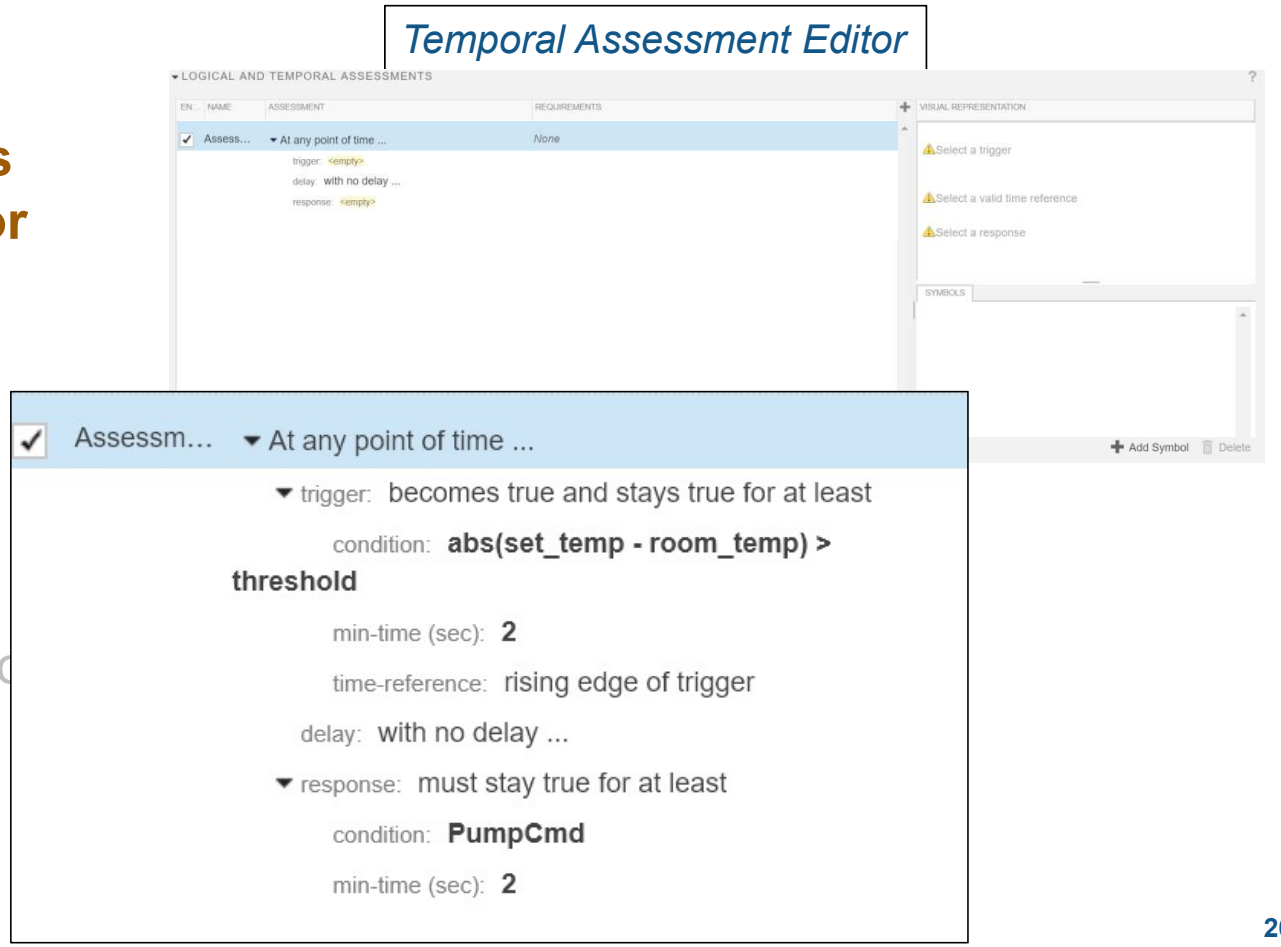
- **Compose assessments using form based editor**
- View assessments as English-like sentence
- Link to requirements
- Review and debug temporal assessment results



Translate textual requirements into unambiguous Temporal Assessments

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

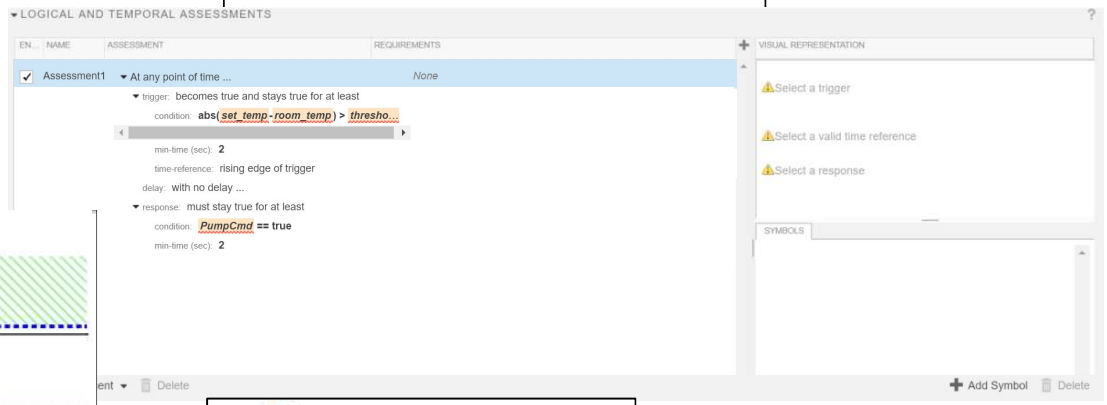
Temporal Assessment Editor

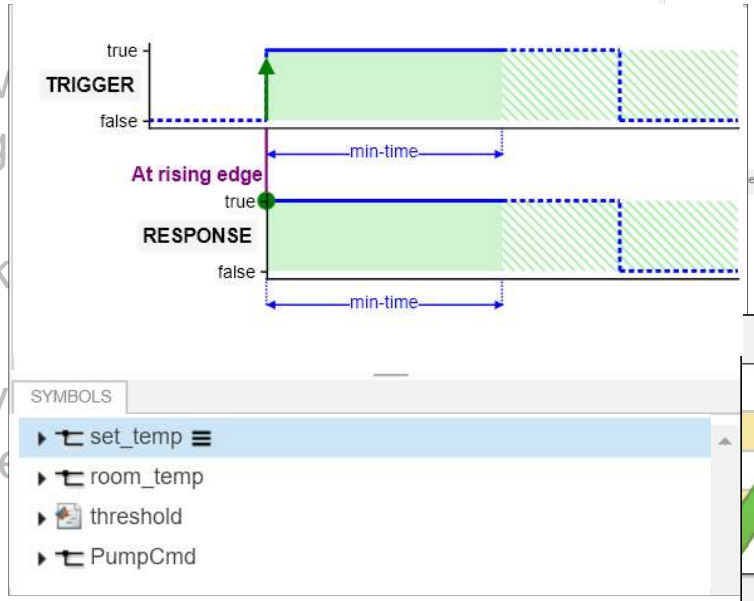


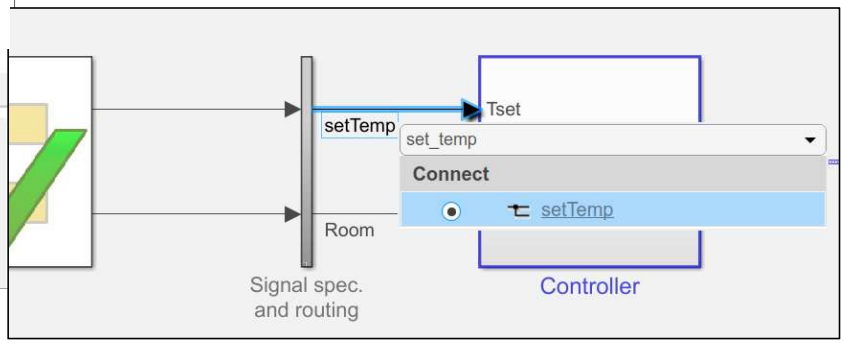
Translate textual requirements into unambiguous Temporal Assessments

- **Compose assessments using form based editor**
- Vie
  Eng
- Link
- Rev
  ass

Temporal Assessment Editor



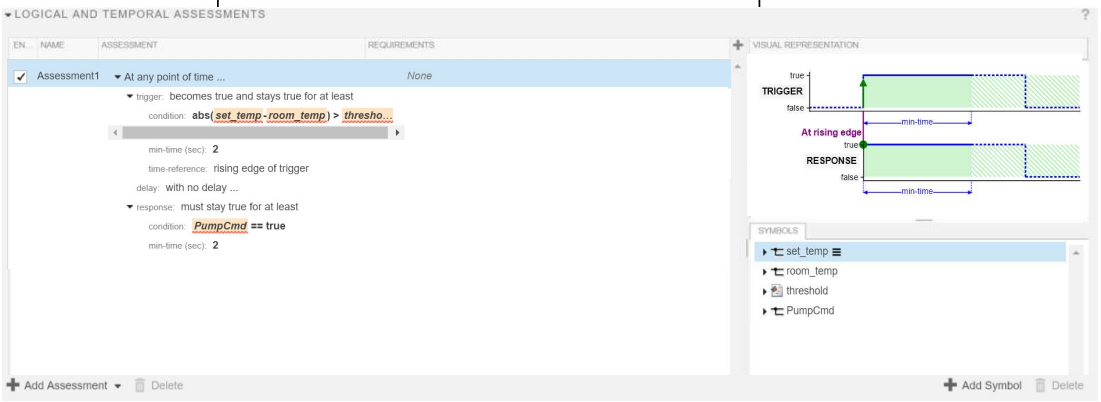




Translate textual requirements into unambiguous Temporal Assessments

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

Temporal Assessment Editor

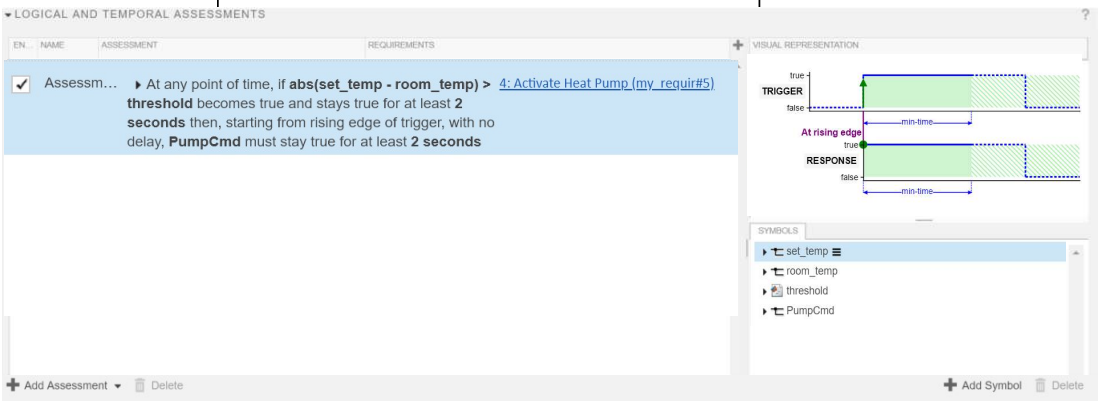


Assessm… ▸ At any point of time, if **abs(set_temp - room_temp) > threshold** becomes true and stays true for at least **2 seconds** then, starting from rising edge of trigger, with no delay, **PumpCmd** must stay true for at least **2 seconds**

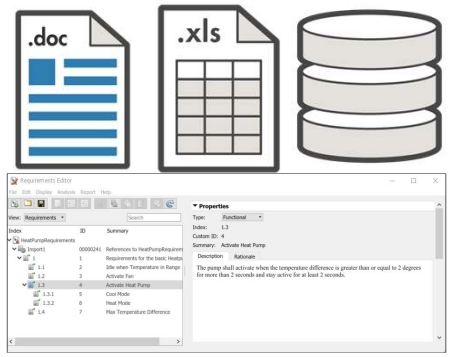
Translate textual requirements into unambiguous Temporal Assessments

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

Temporal Assessment Editor



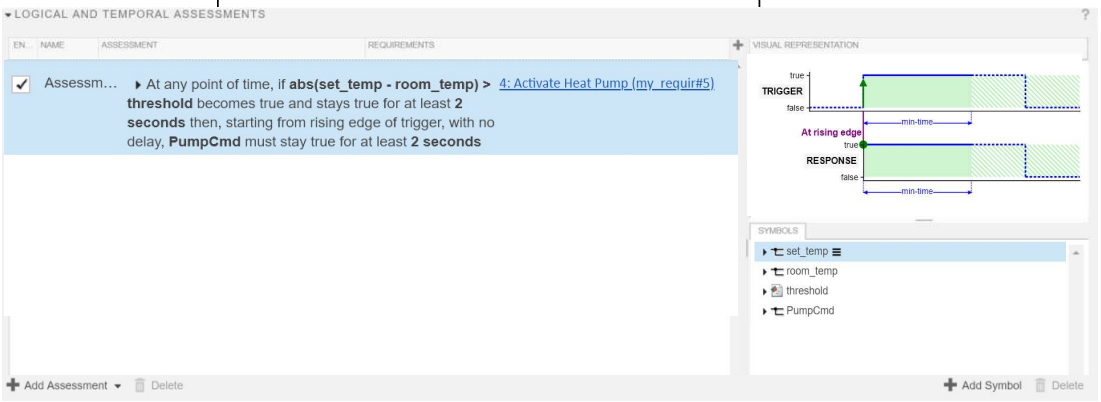
Requirements



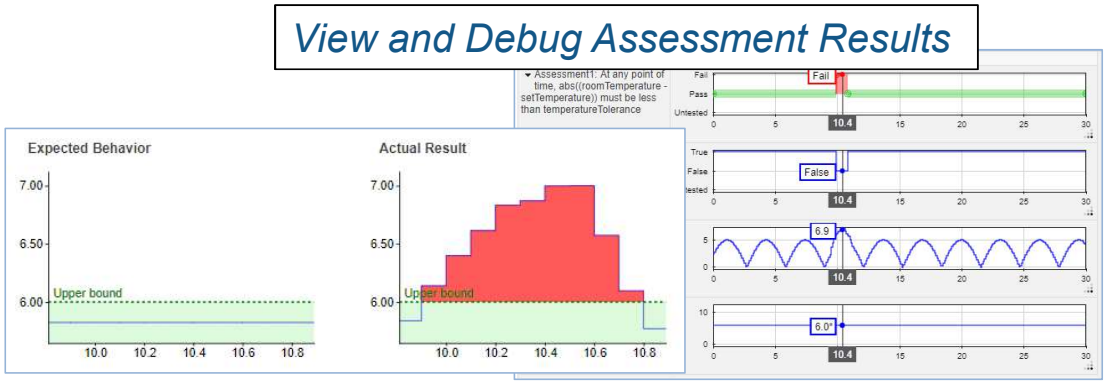
Translate textual requirements into unambiguous Temporal Assessments

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

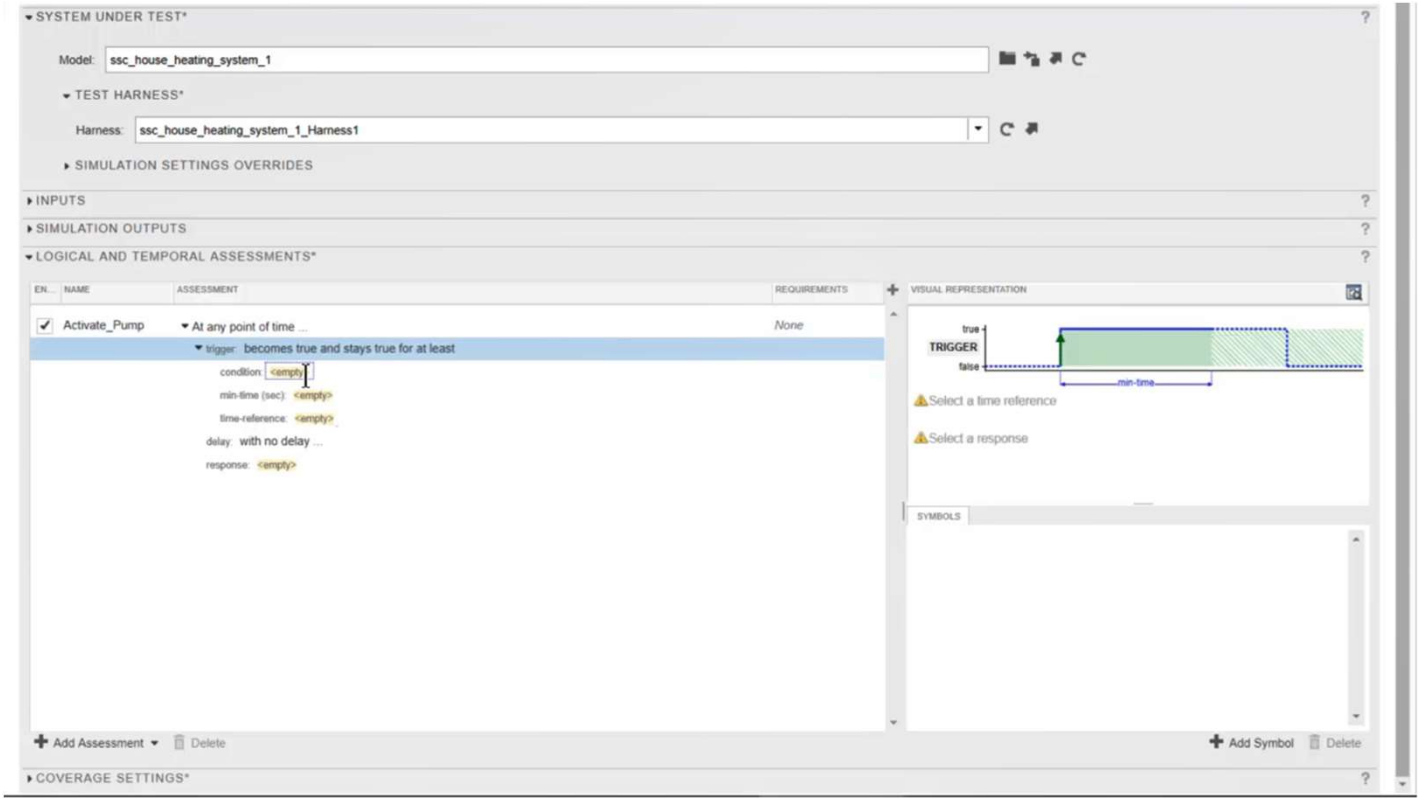
Temporal Assessment Editor



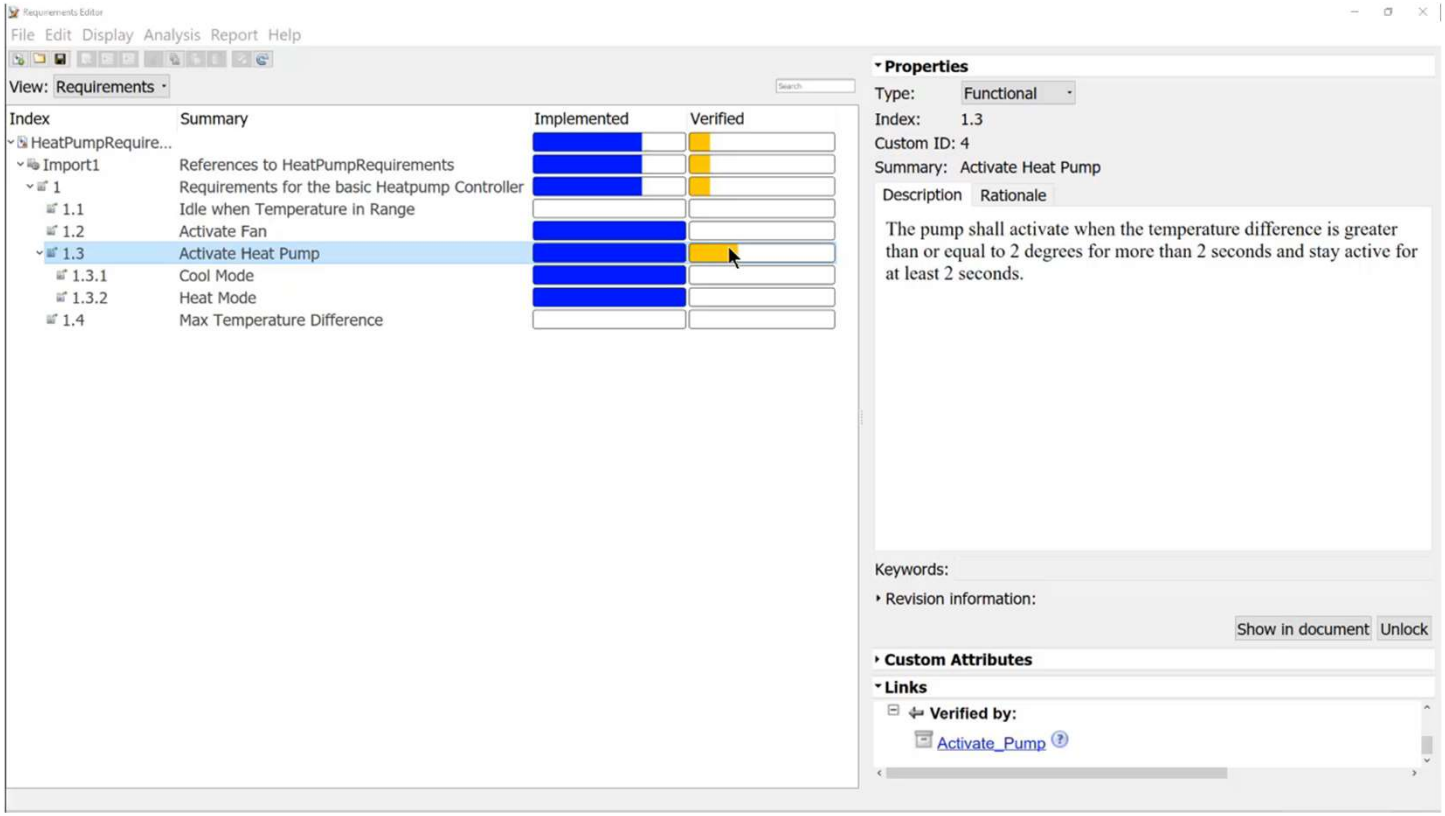
View and Debug Assessment Results



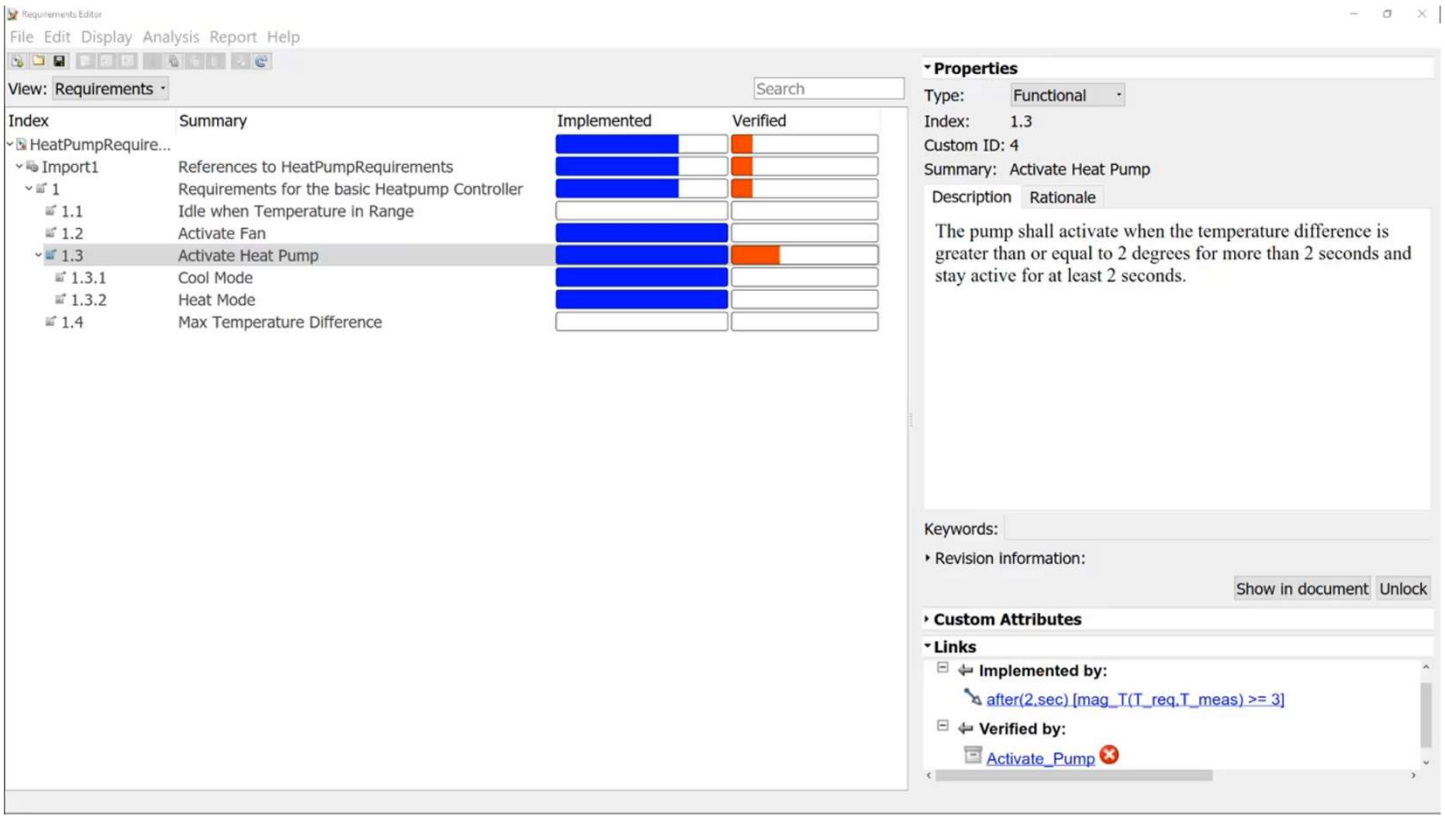
Demo : Temporal Assessments



Execute assessments to verify requirements

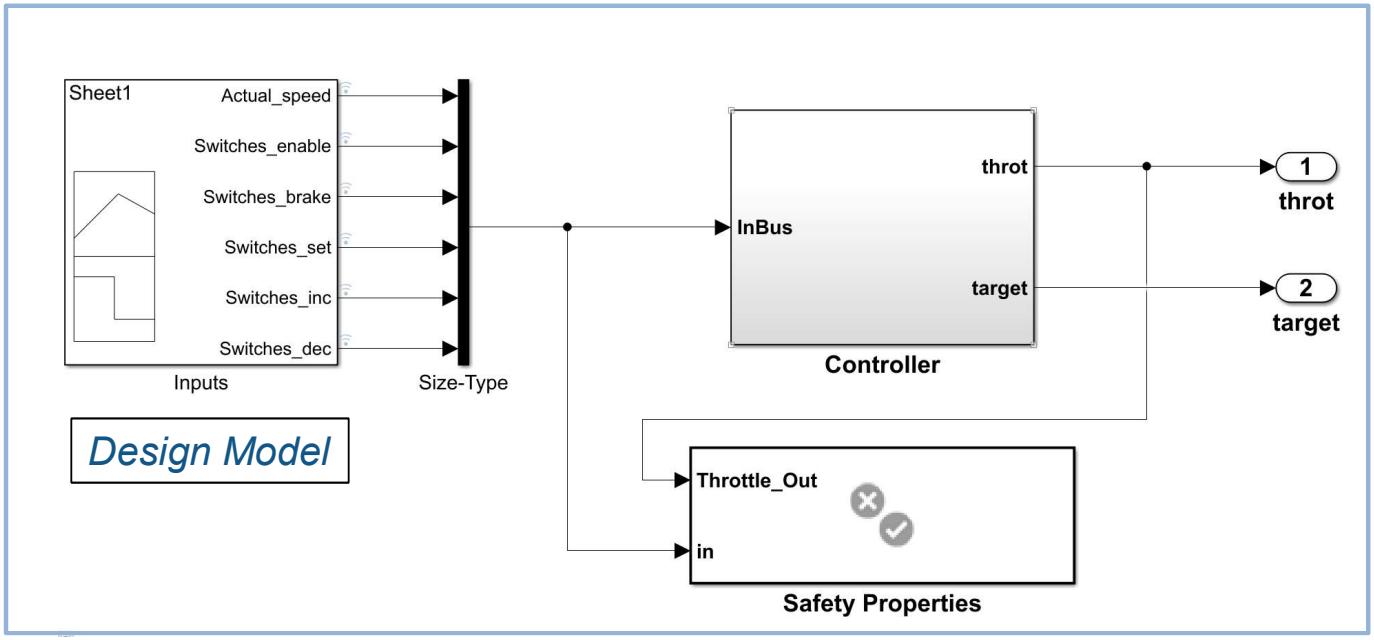


Locate implementation of requirement using link



Observers: Separate test/verification logic from design

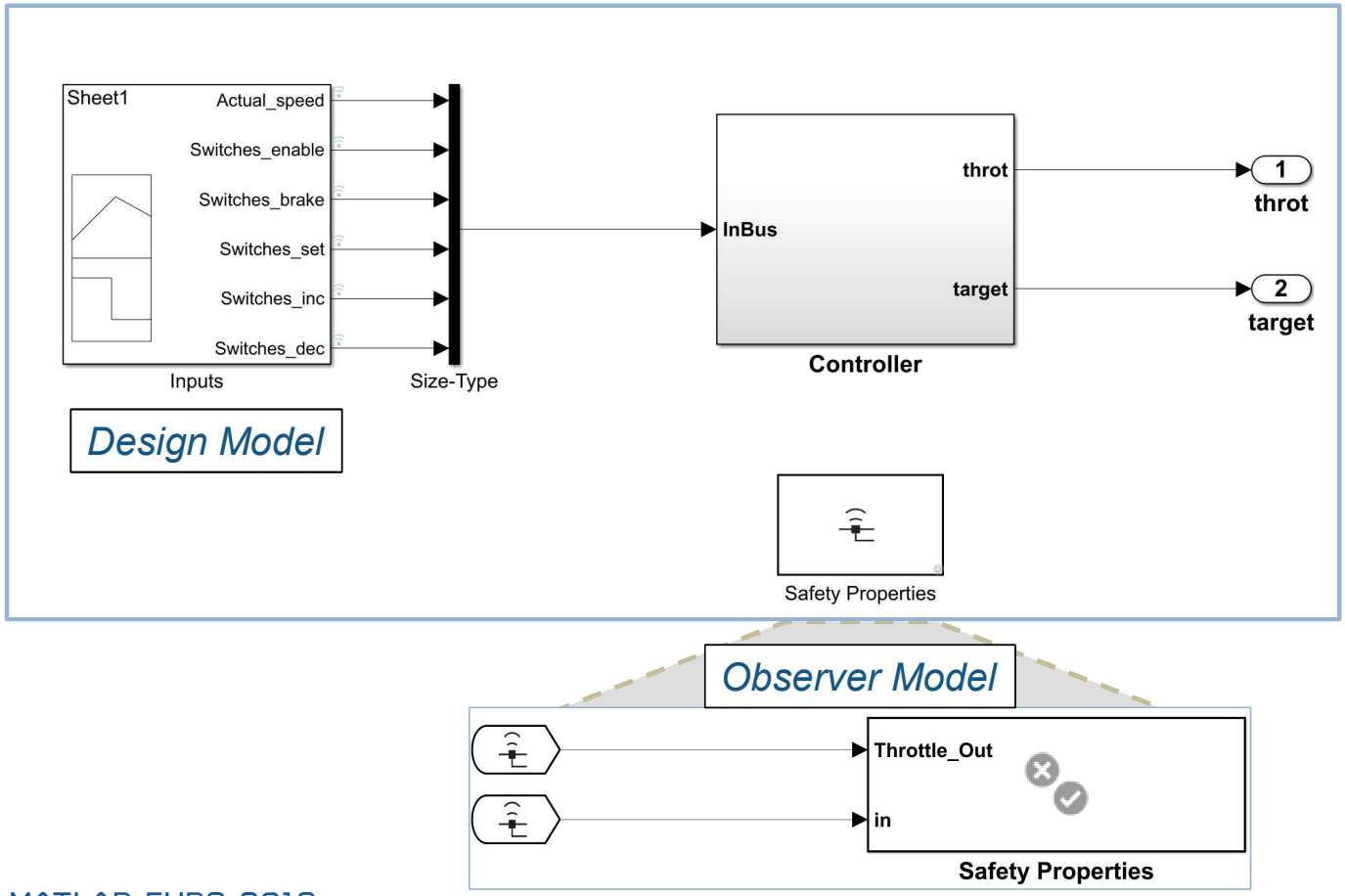
R2019a



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

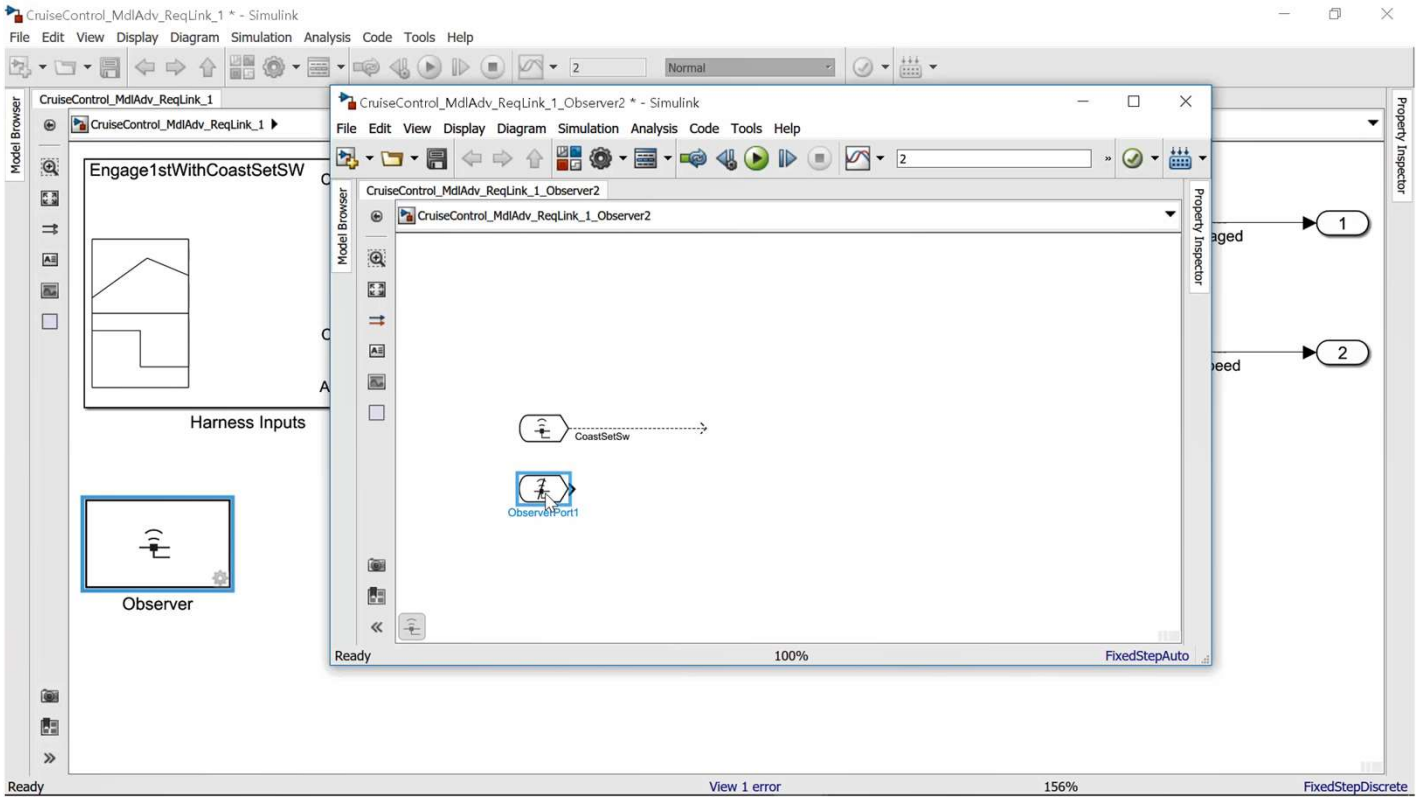
Observers: Separate test/verification logic from design

R2019a



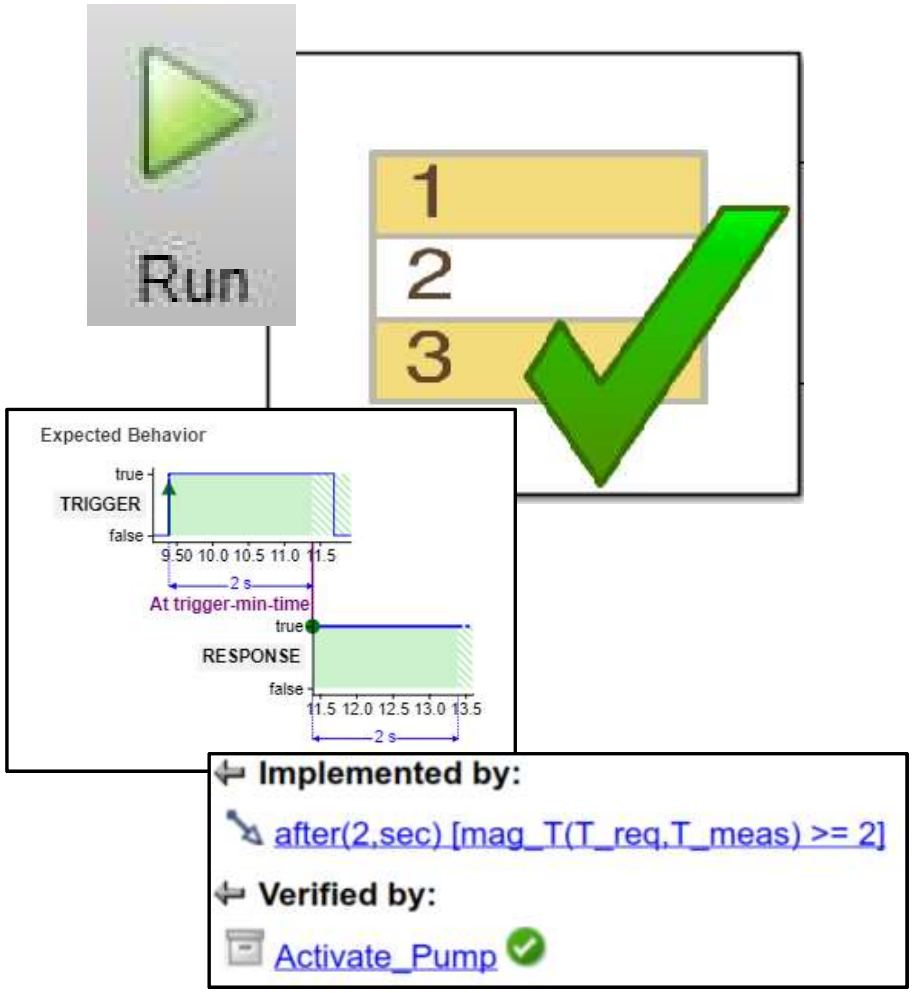
- Access nested signals without signal lines or changing dynamic response
- Avoid modifying interface for testing
- Simplify design and test by avoiding additional signal lines

Demo : Observers



Summary

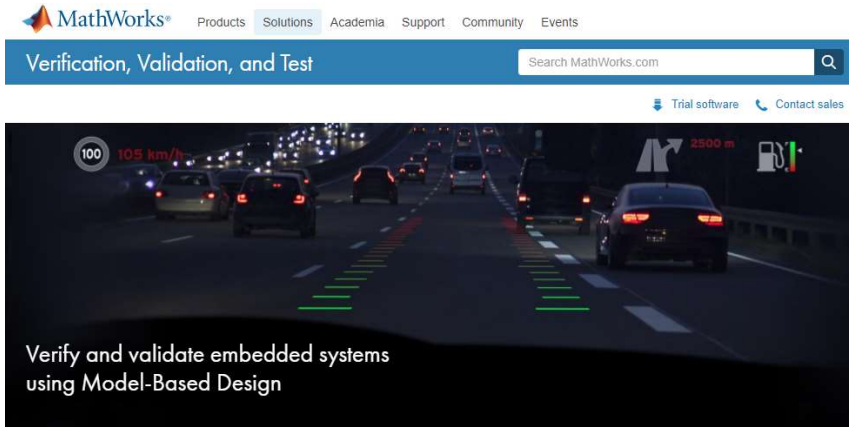
- Verify and validate requirements earlier
- Identify inconsistencies in requirements by using unambiguous assessments
- Traceability from requirements to design and test



Learn More

Key products covered in this presentation:

- Simulink Requirements
- Simulink Test
- Simulink Real-Time



Learn more at Verification, Validation and Test Solution Page:
mathworks.com/solutions/verification-validation.html