MATLAB EXPO

Test Driven Development in Agile Model-Based Design

Paul Urban
Marco Dragic

MathWorks®

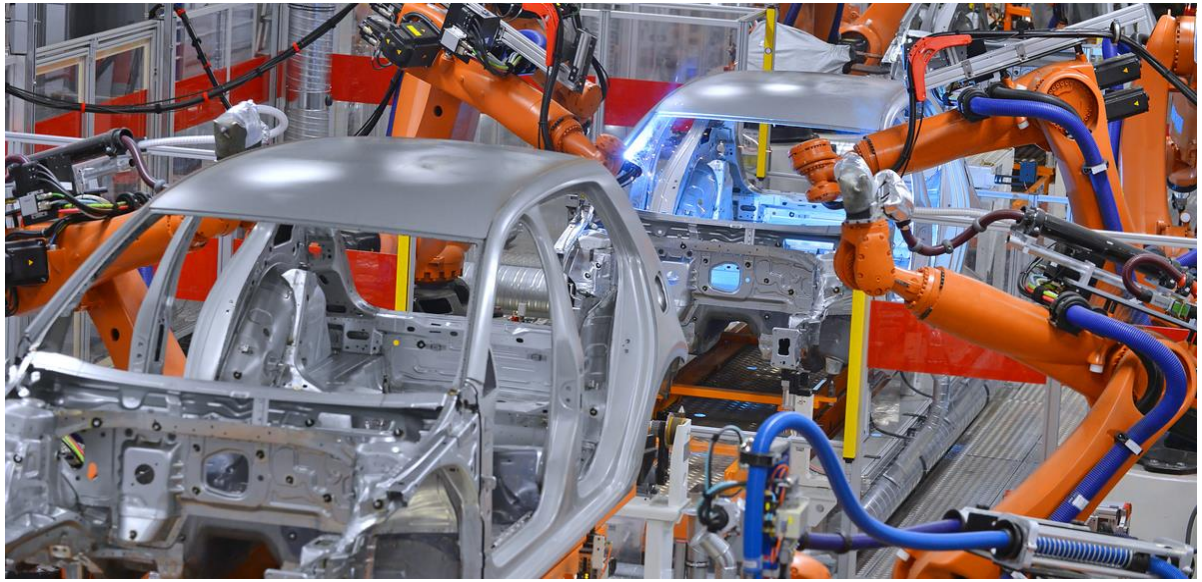
Marco Dragic

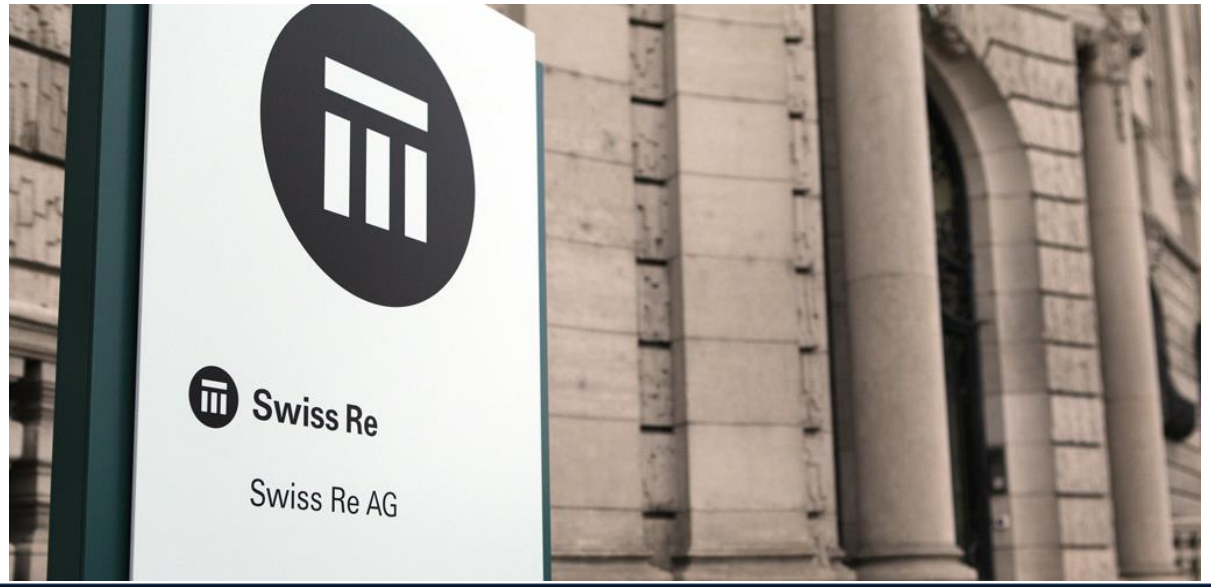
Senior Product Manager
Simulink Platform

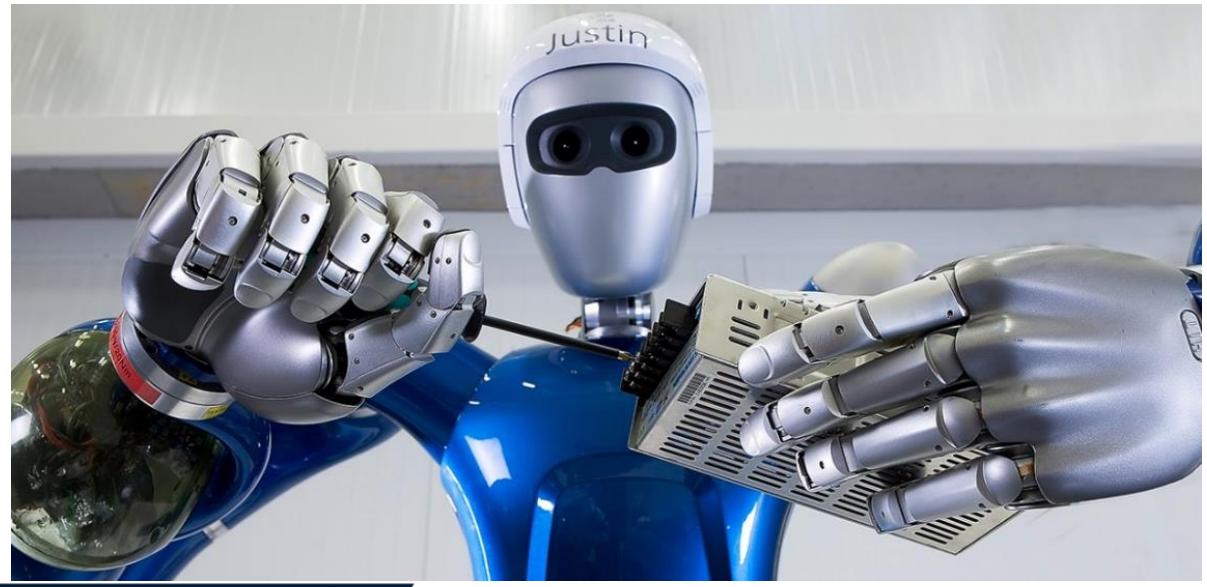
Paul Urban

Senior Product Manager
Simulink Verification and Validation

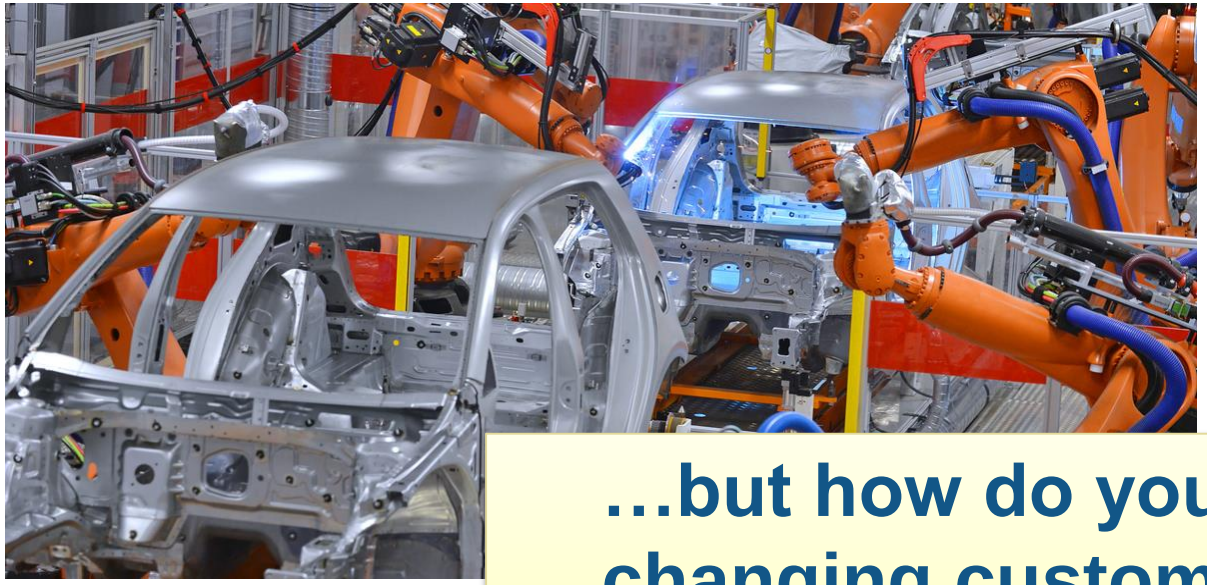
Building Algorithms in Everything…

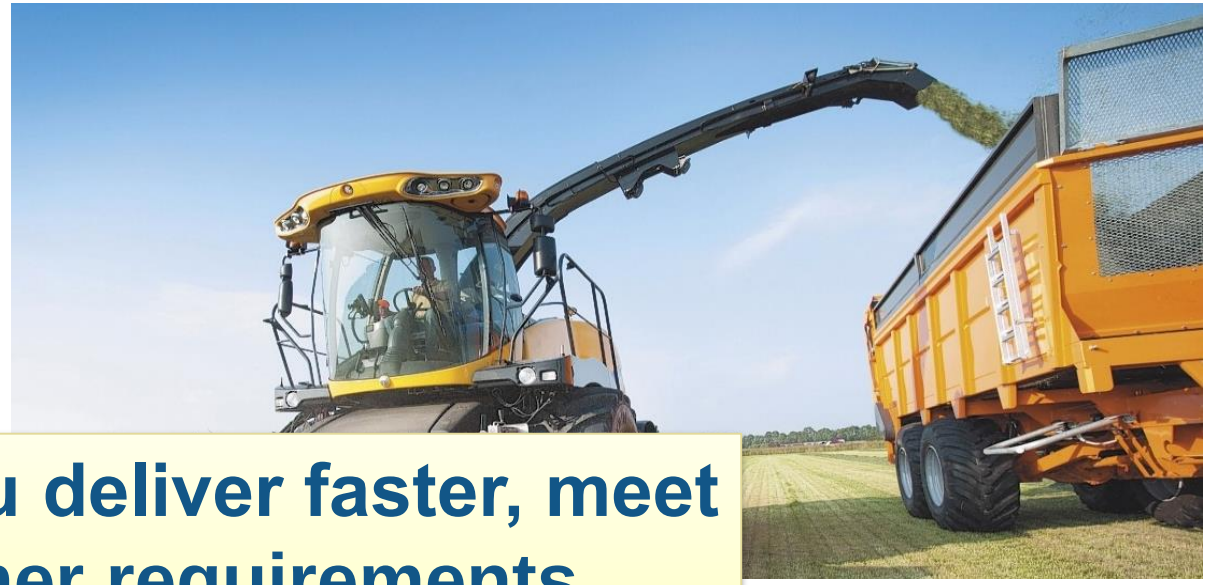




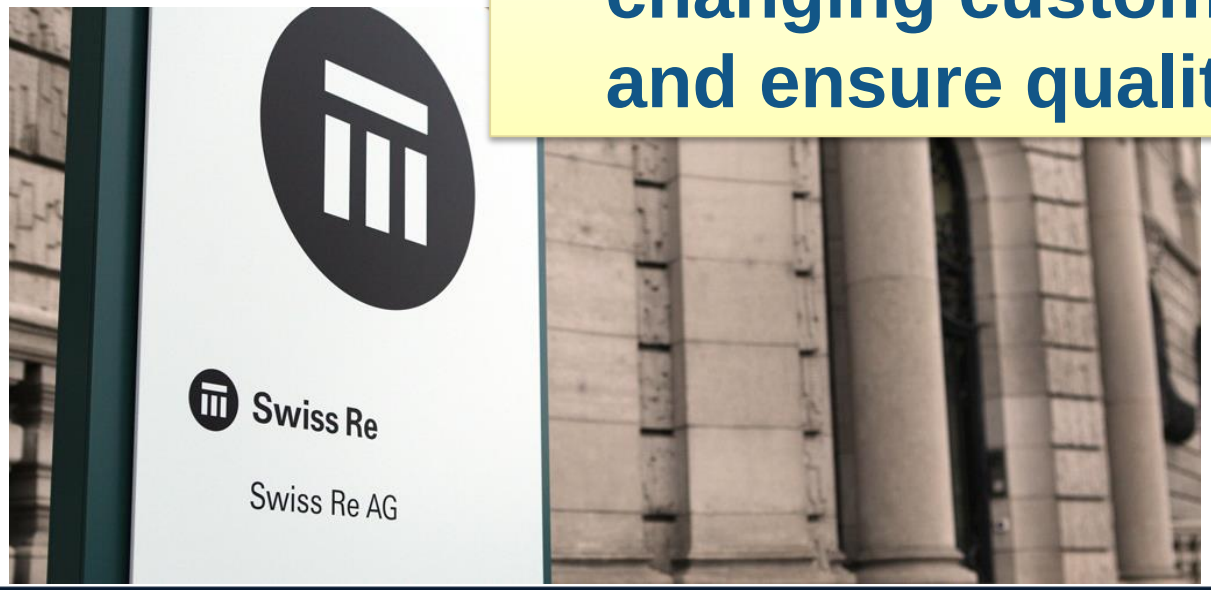


Building Algorithms in Everything…



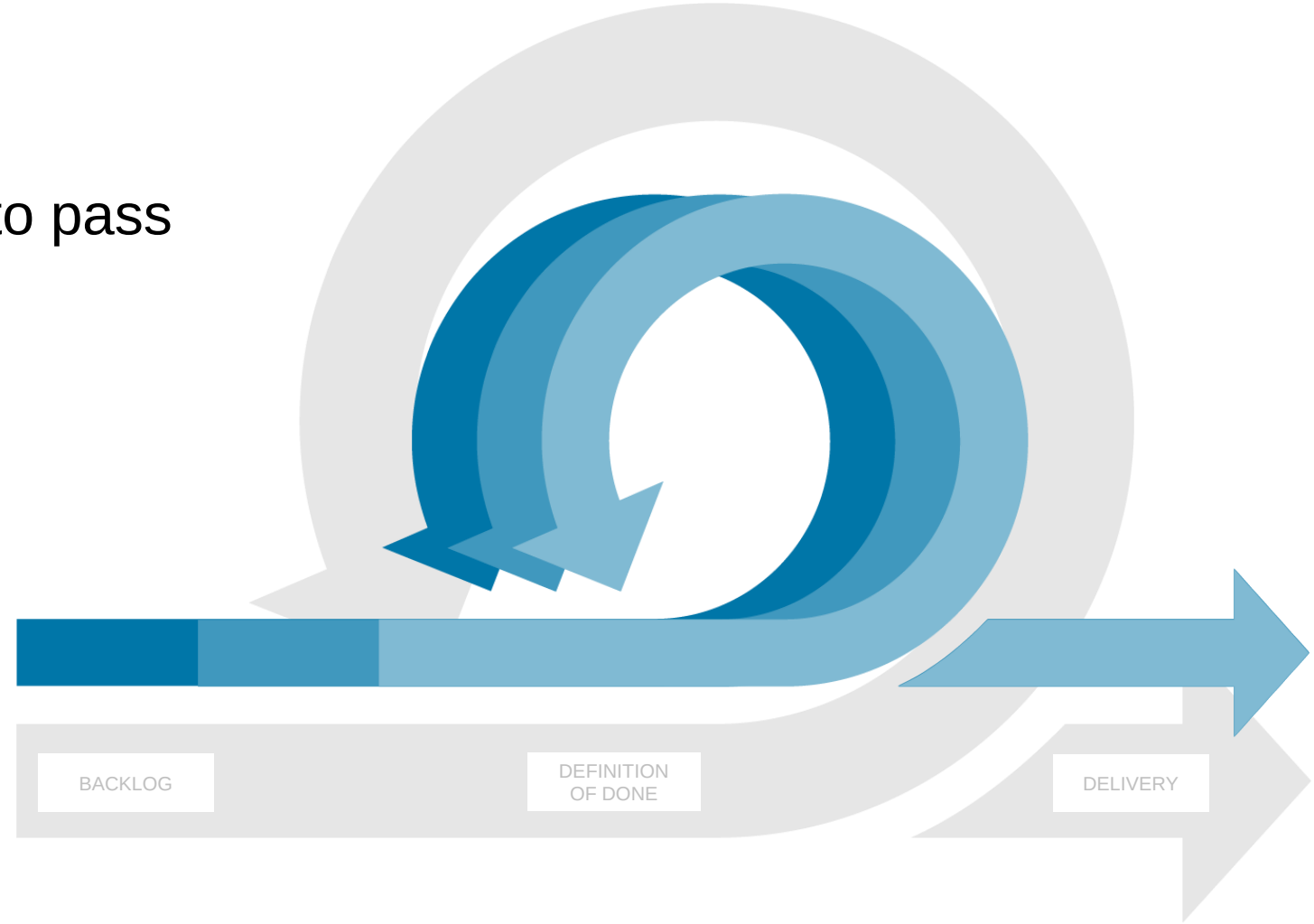


…but how do you deliver faster, meet changing customer requirements, and ensure quality?



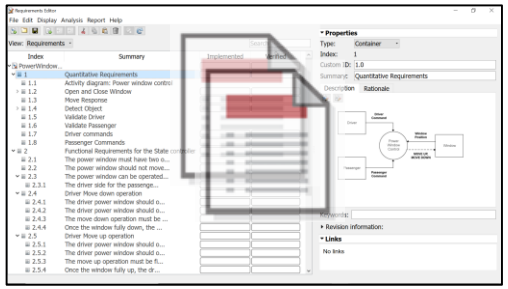
Test Driven Development Cycle

1. Create a test
2. Implement enough for test to pass
3. Refactor

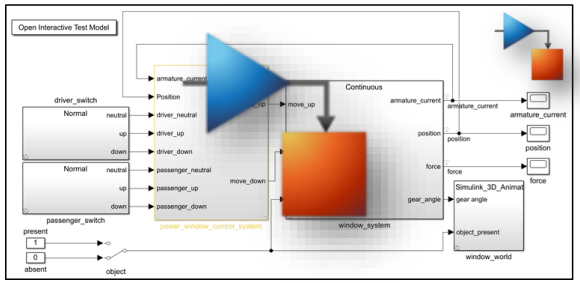


Simulink provides an integrated framework for TDD

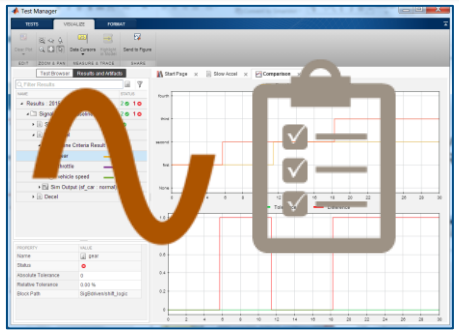
Requirements



Implementation



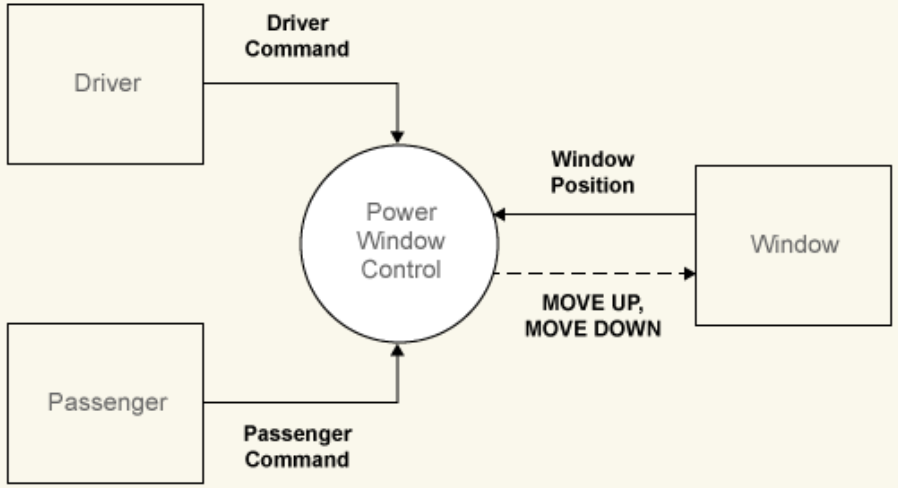
Test

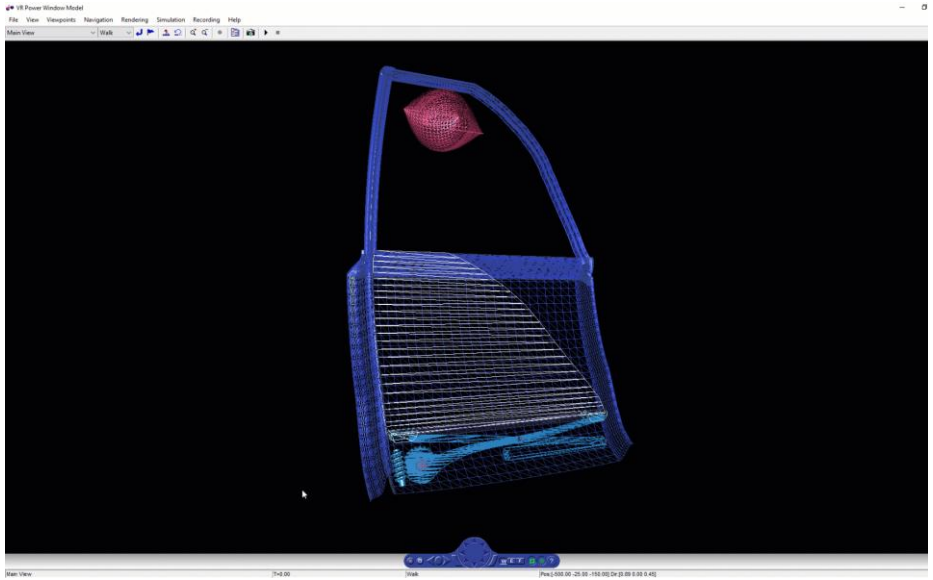


Starting with high level customer requirements

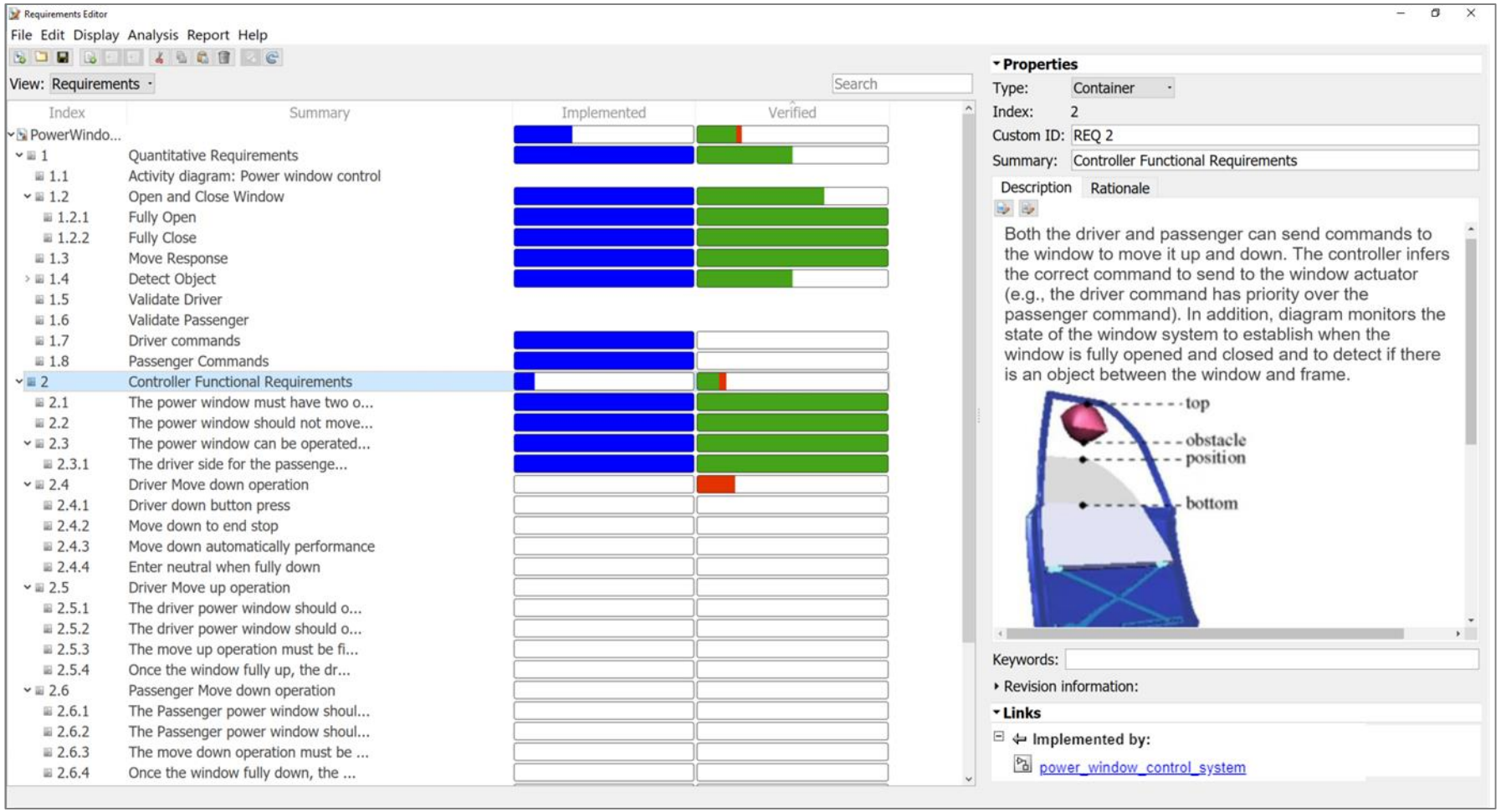
User Requirements:

- Both driver and passenger can control the window
- Window stops closing if an object is detected
- Window should have option to fully open and close



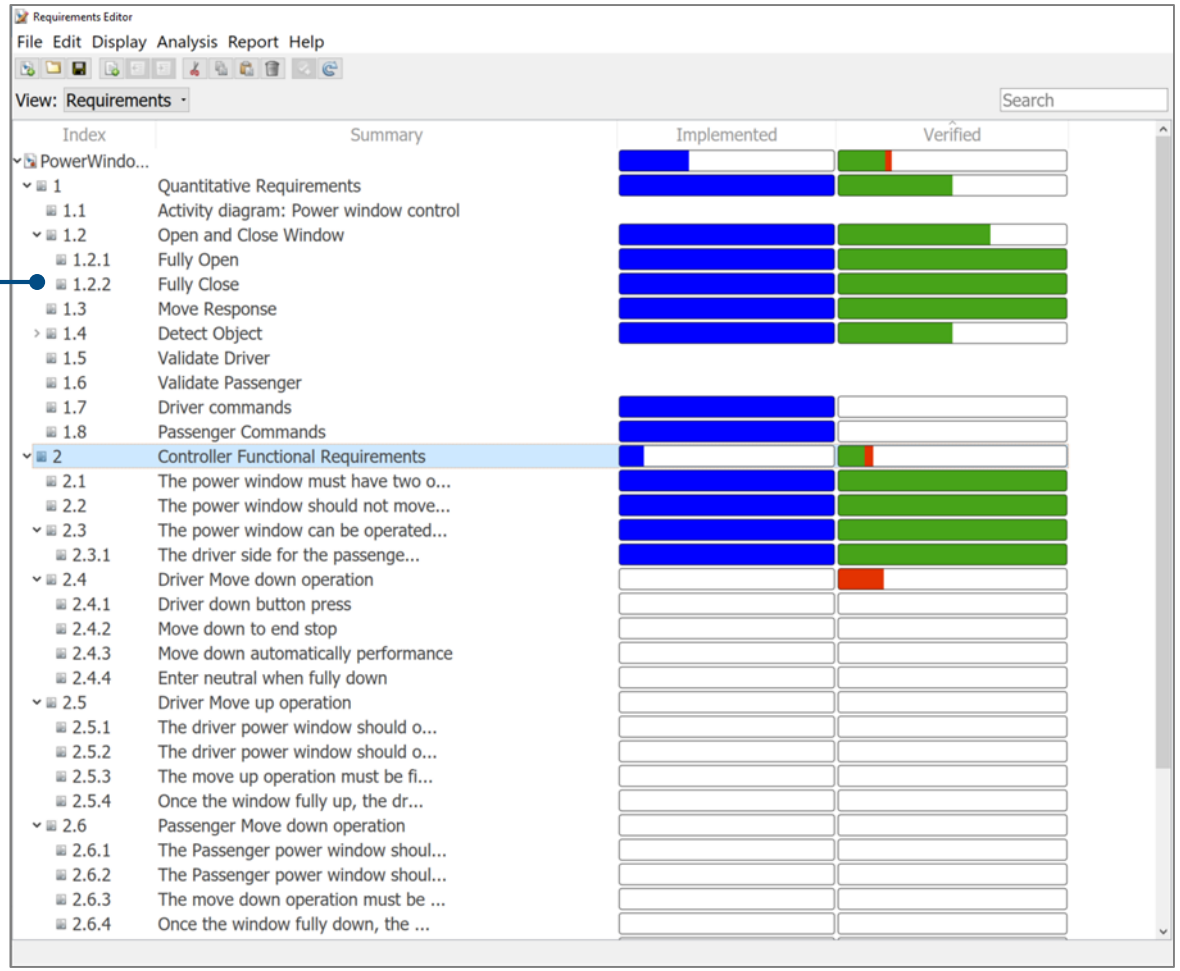


Capturing requirements

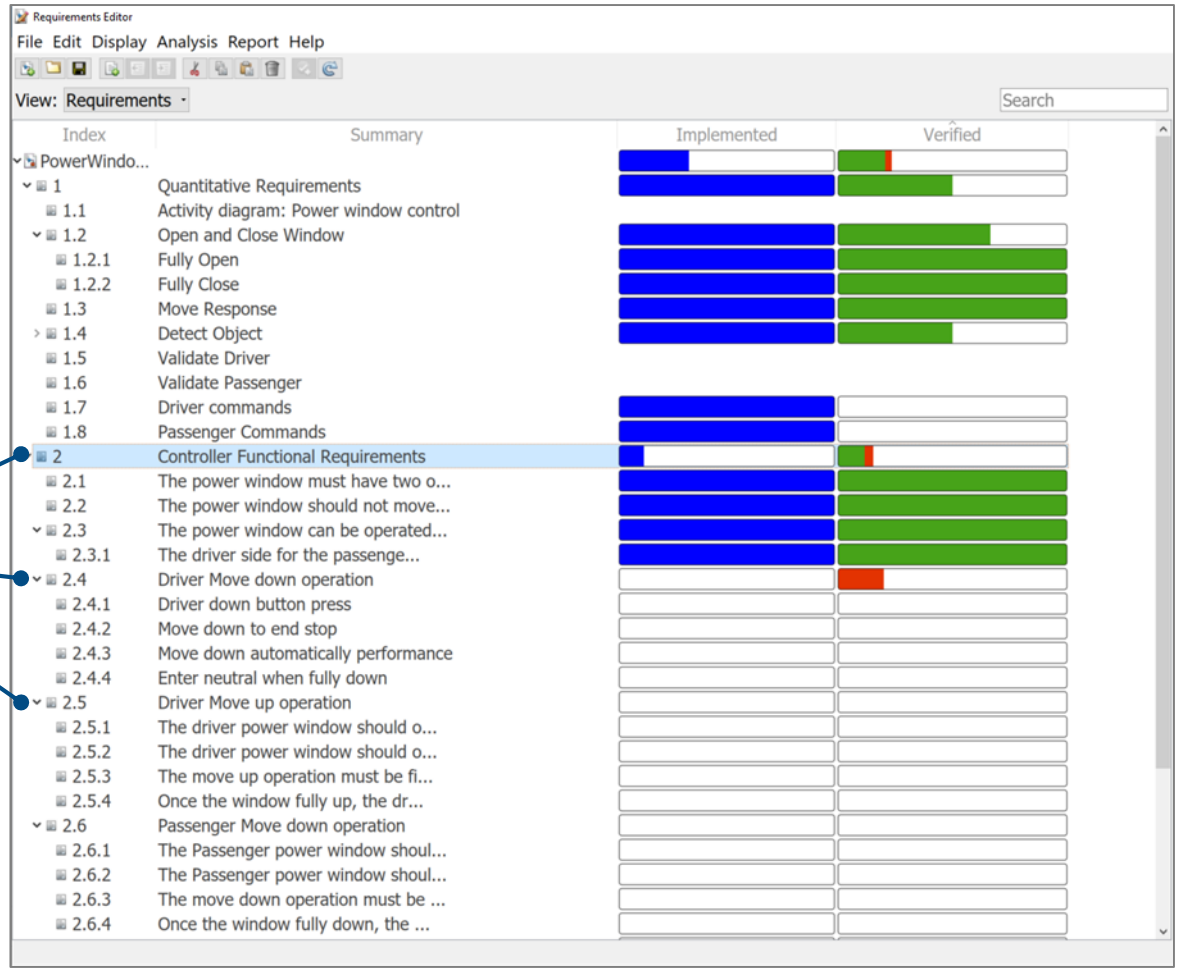


Viewing details

Requirement Details

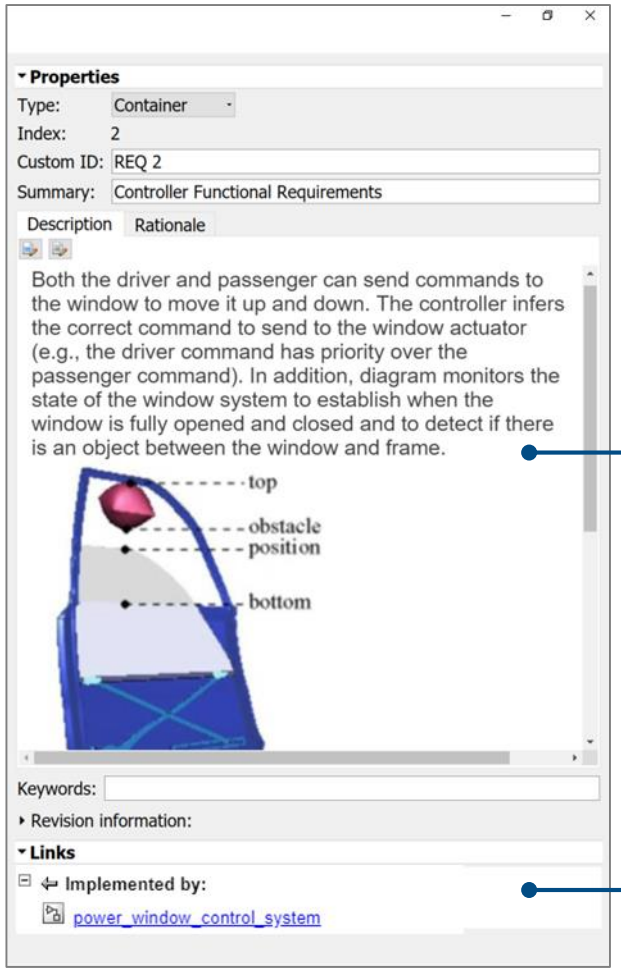


Organizing and creating requirement hierarchies



Requirement Hierarchies

Specifying details



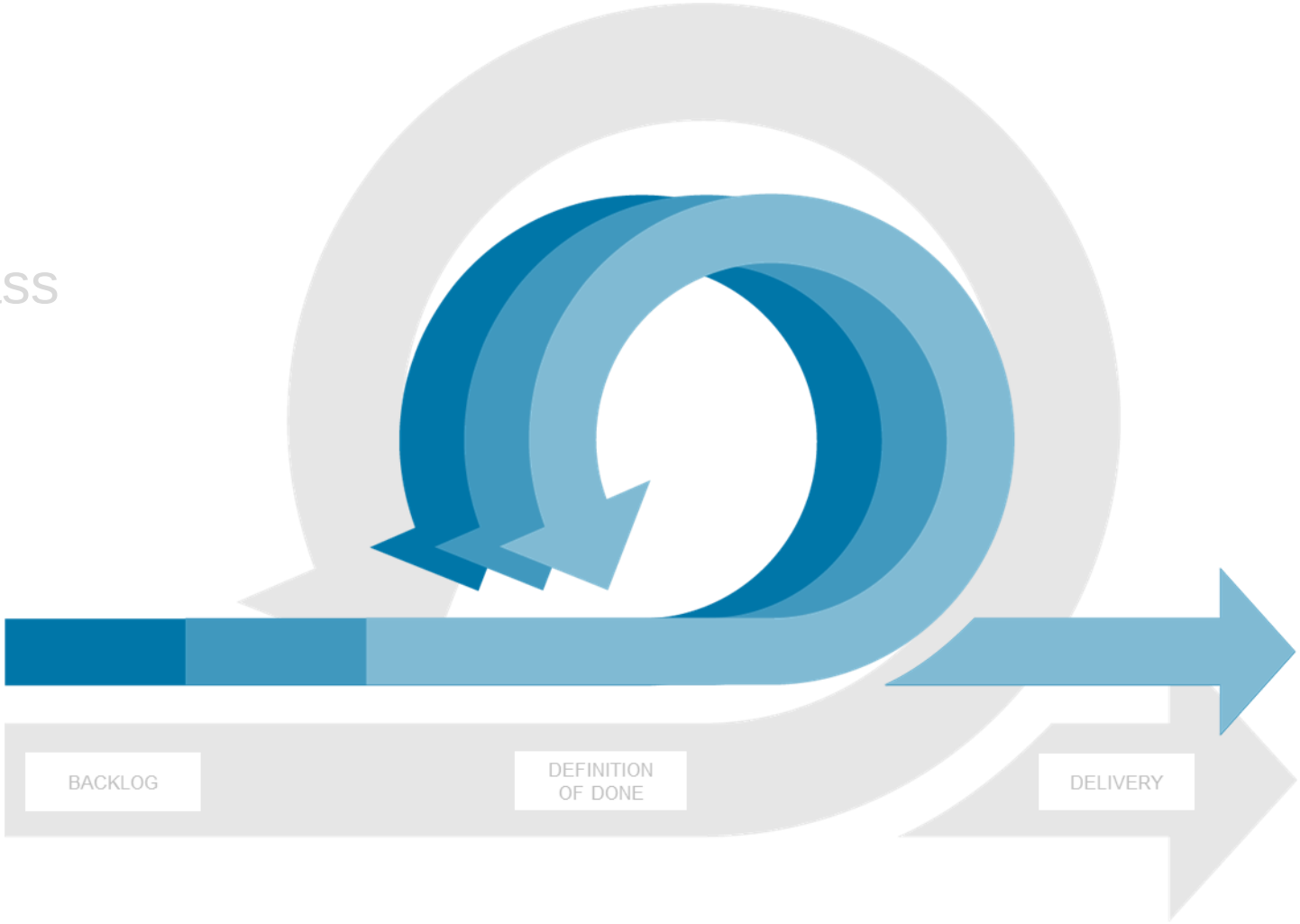
Description / Rationale Fields

Links Pane

1. **Create a test**

2. Implement enough for test to pass

3. Refactor

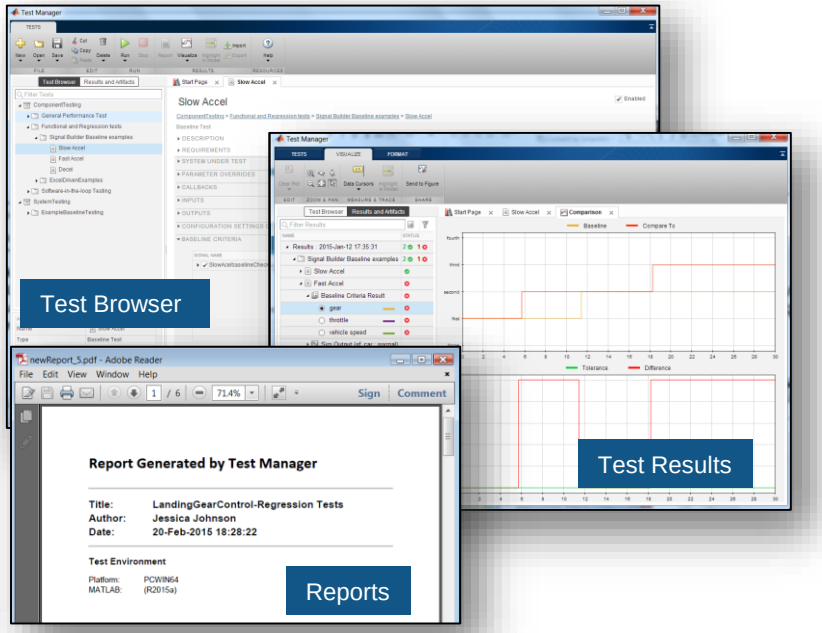


Develop, manage, and execute simulation-based tests

Simulink Test

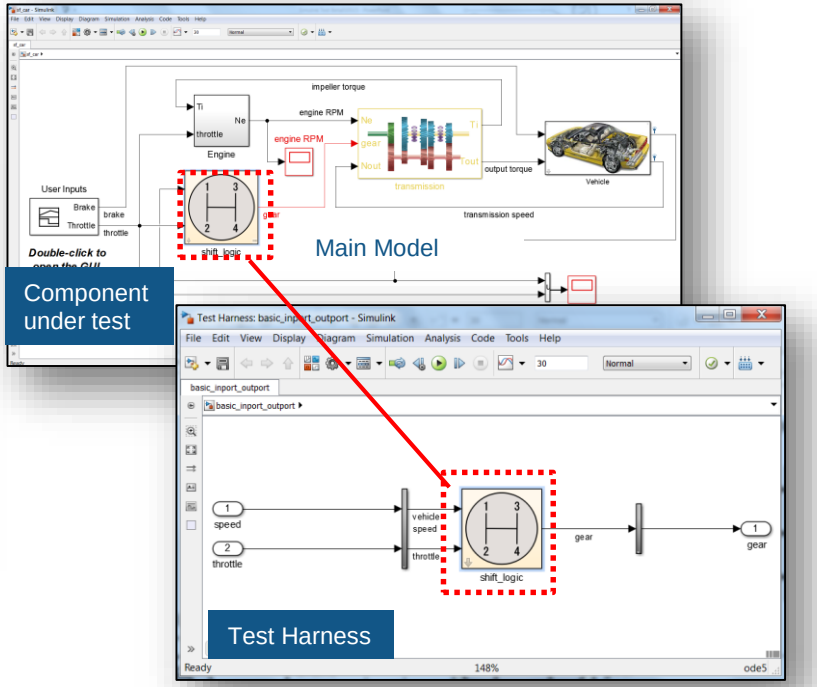
Test Manager

- Author, manage, organize tests



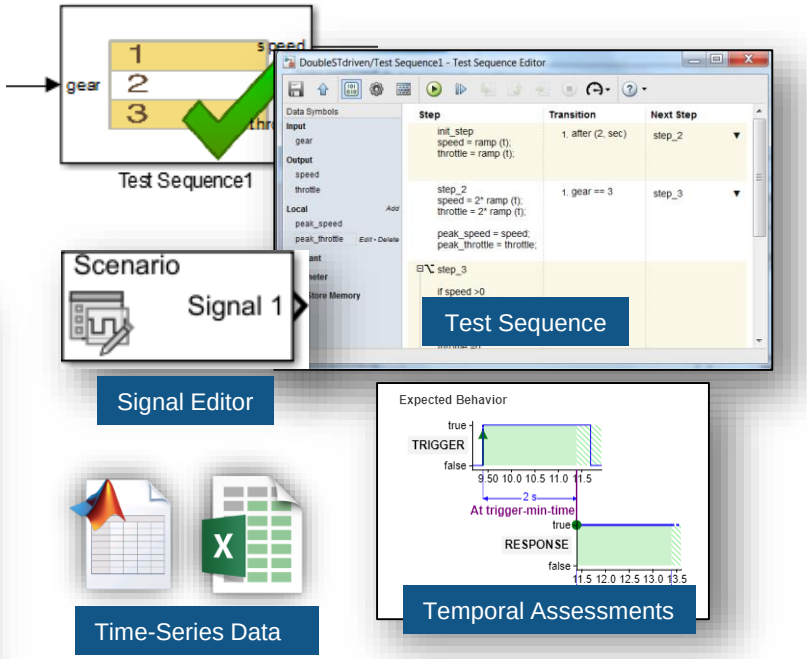
Test Harnesses

- Isolate Component Under Test

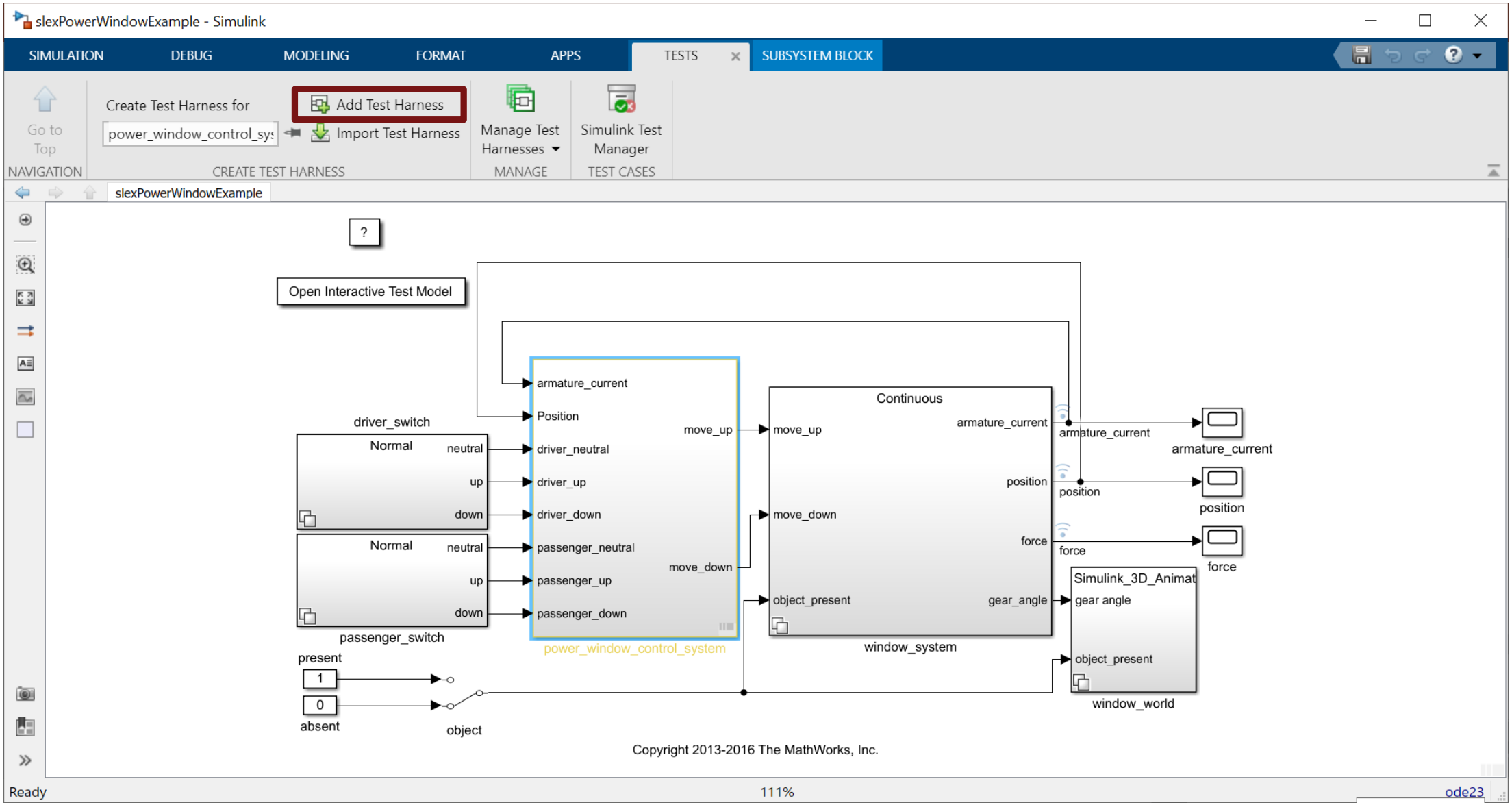


Test Authoring

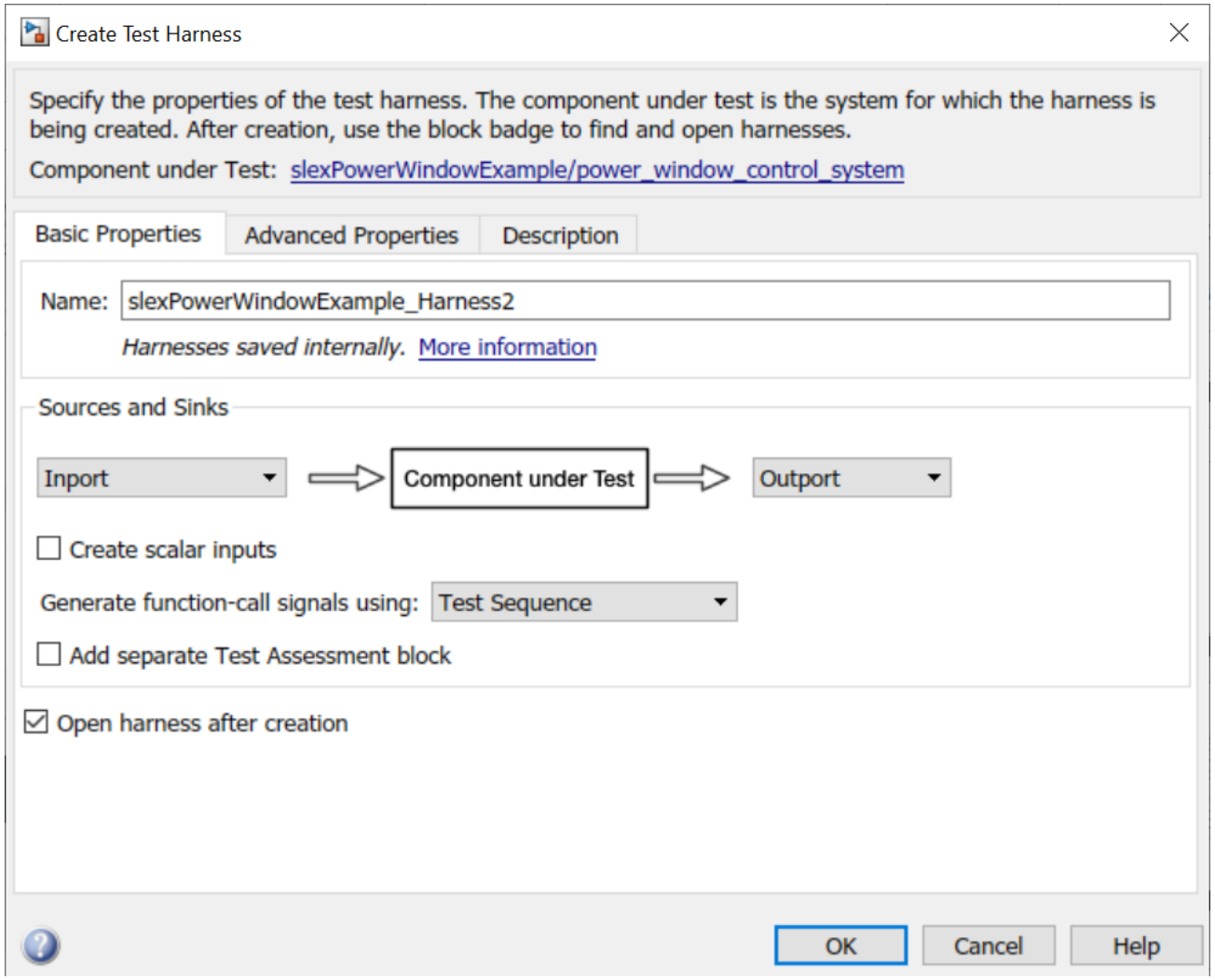
- Specify test inputs, expected outputs, and tolerances



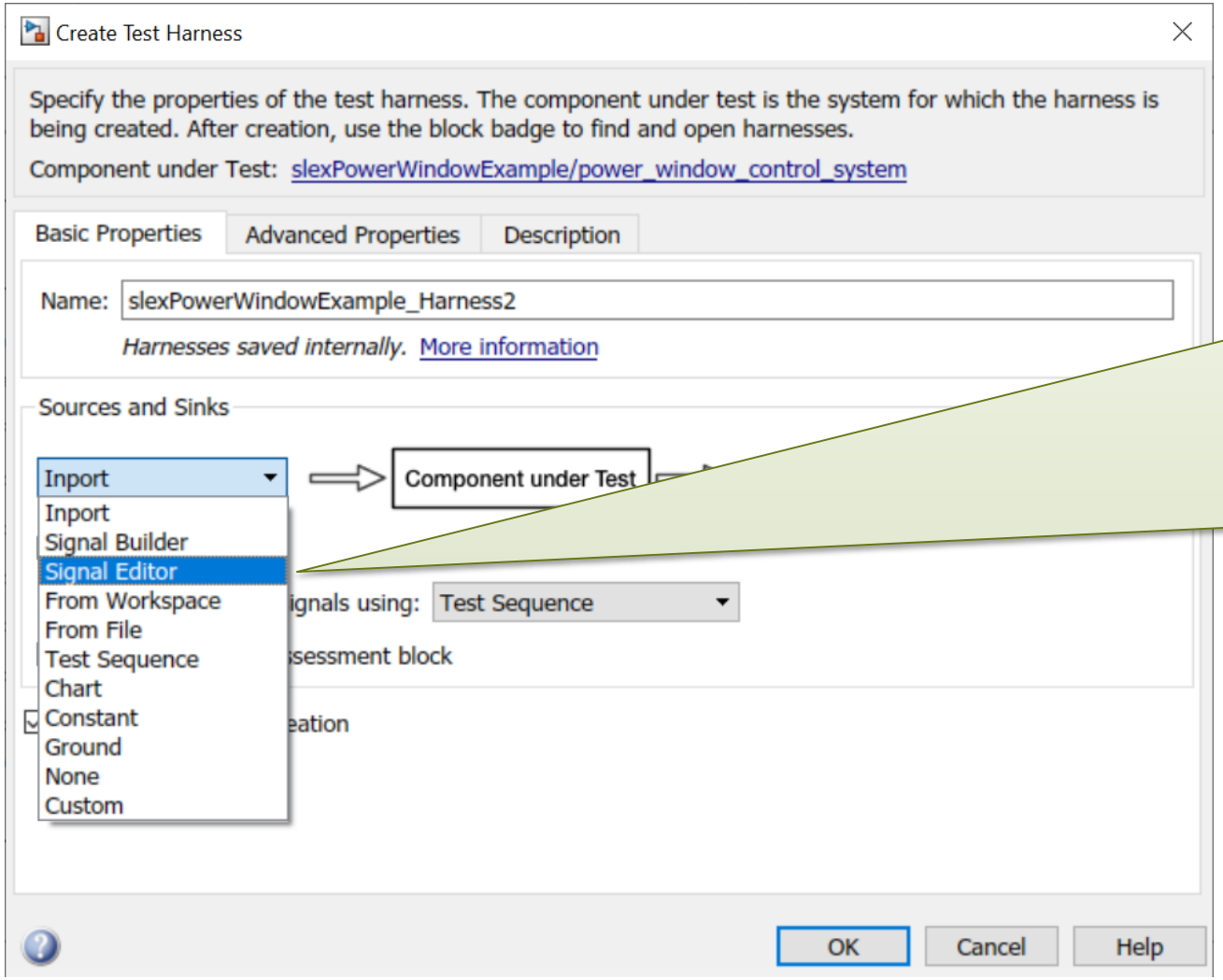
Creating a Test Harness to isolate Component Under Test

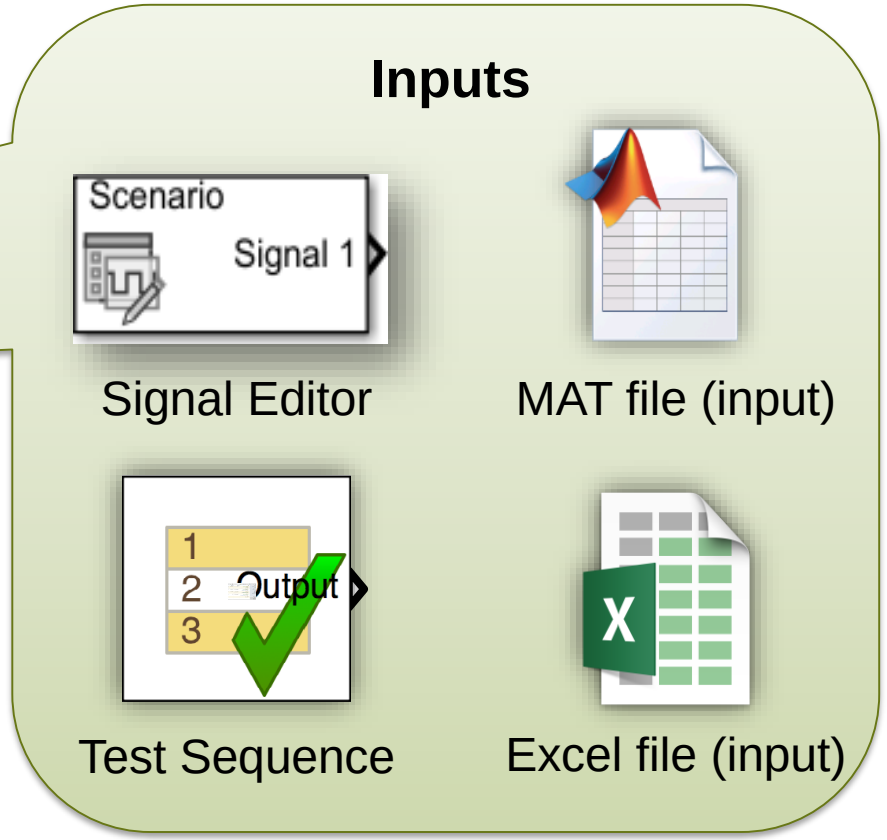


Specify properties of the Test Harness

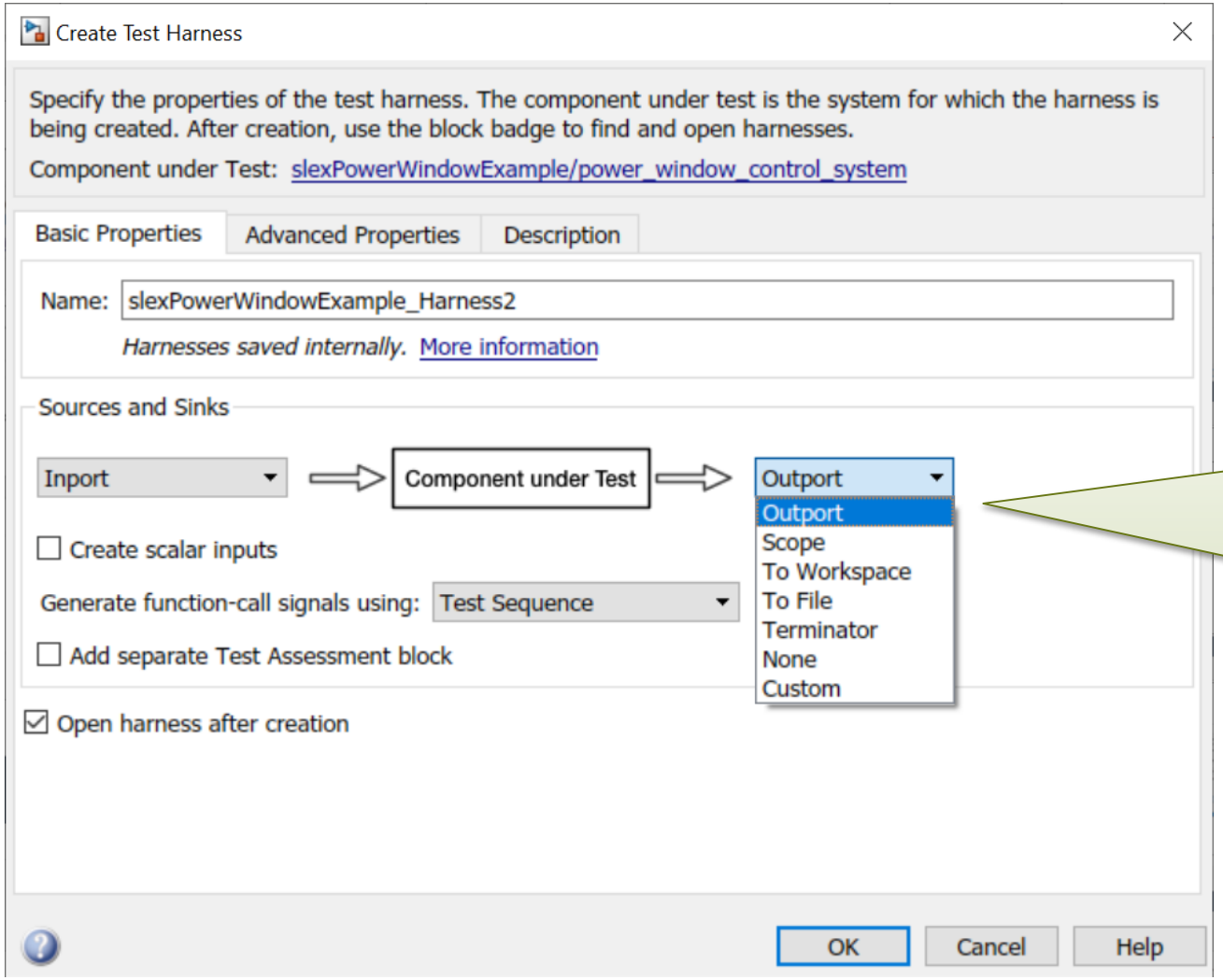


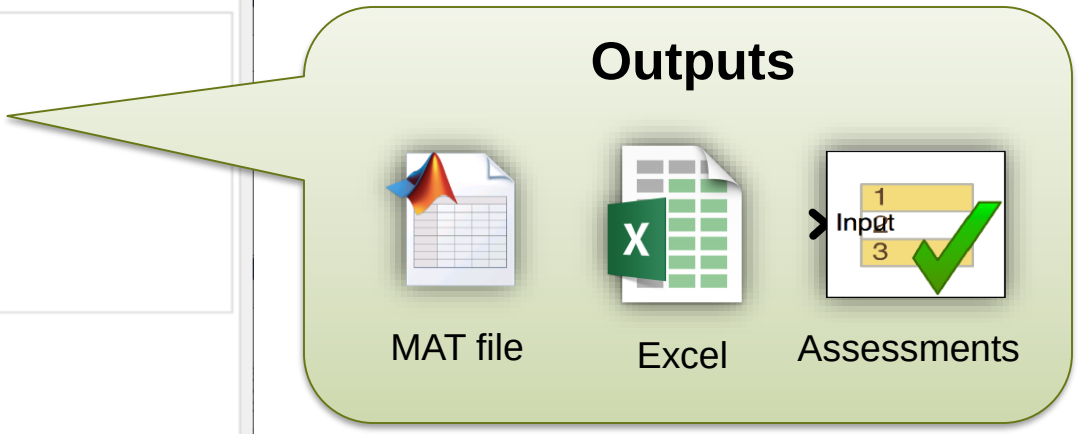
Specify inputs



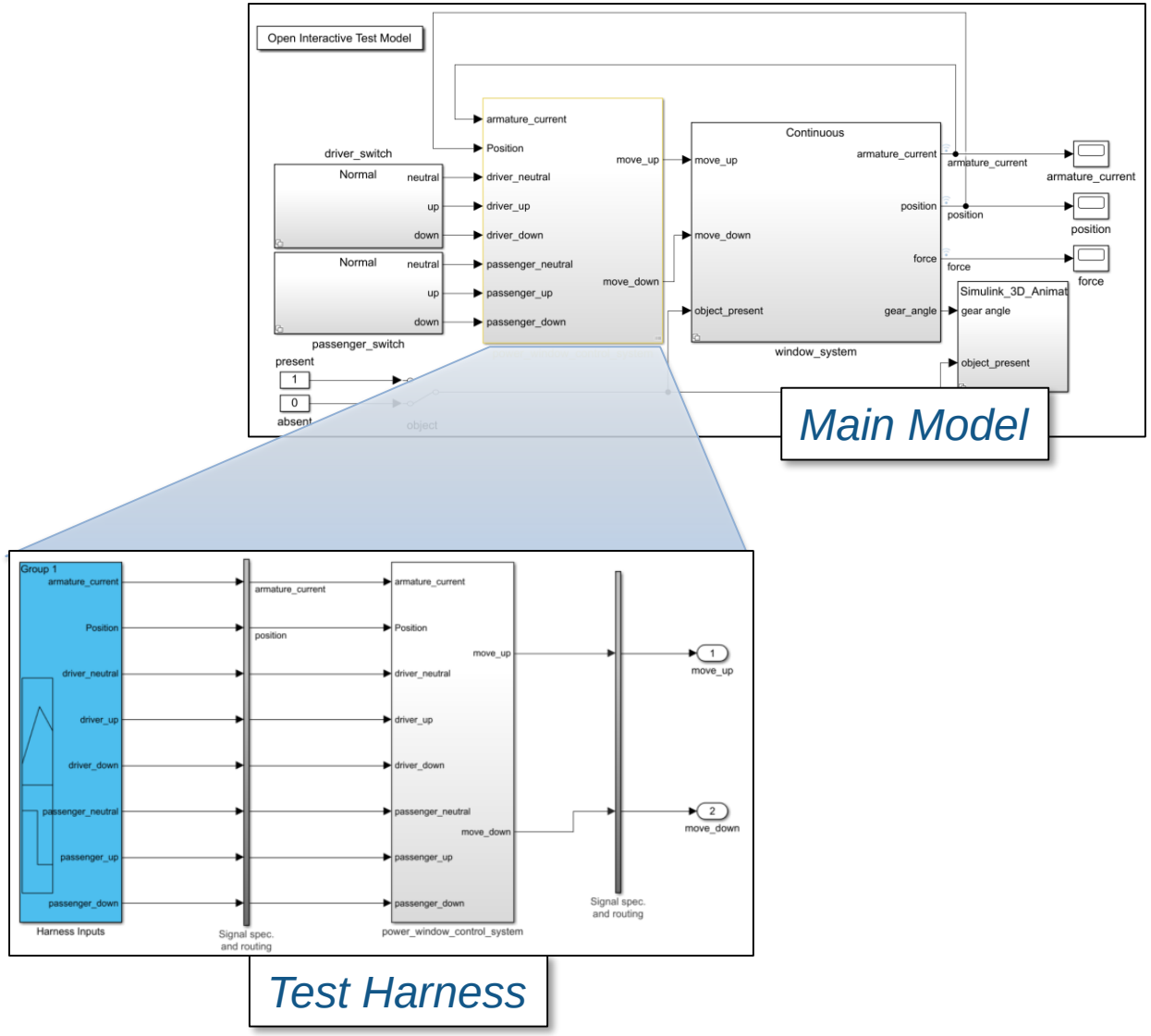


Specify outputs

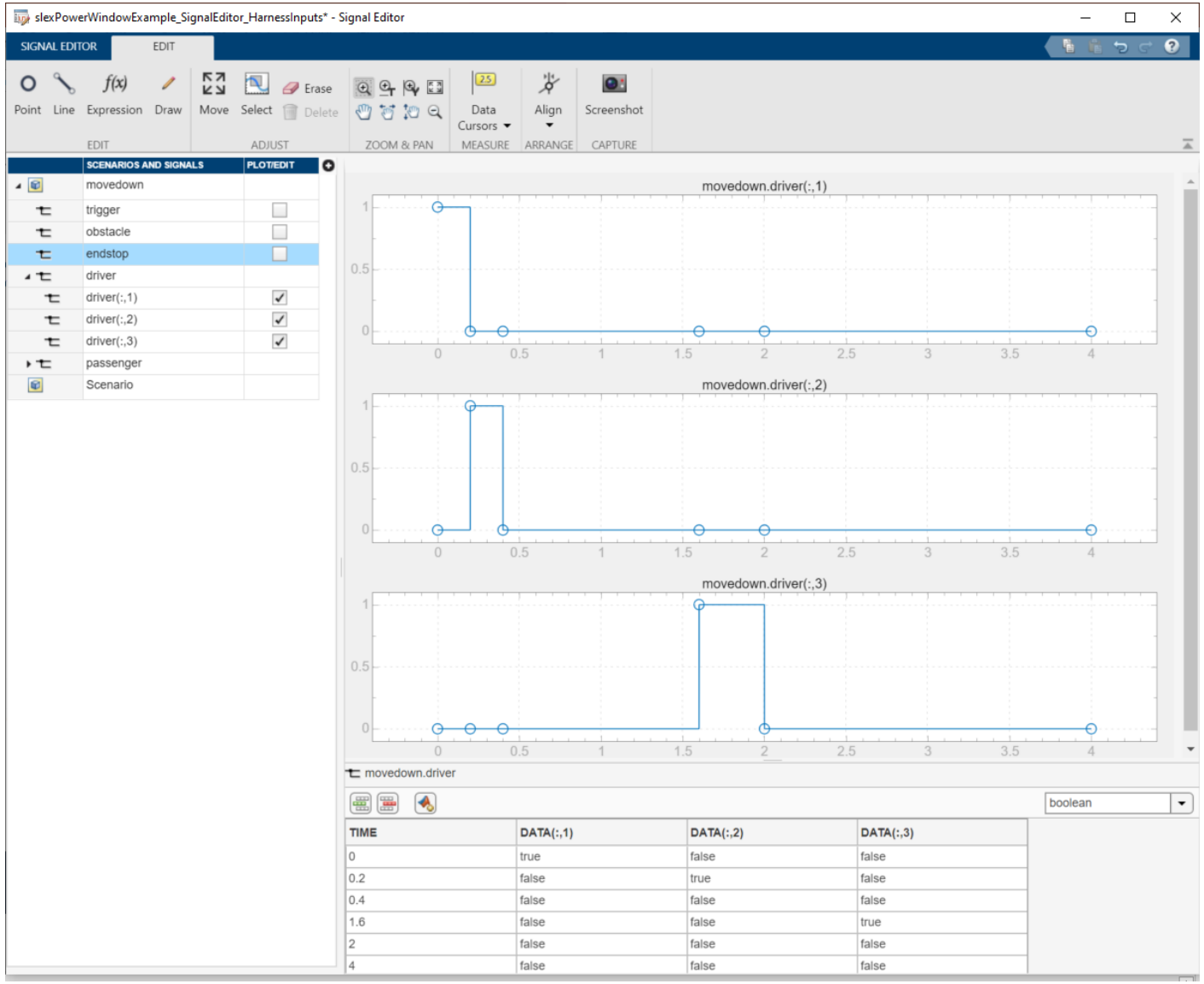




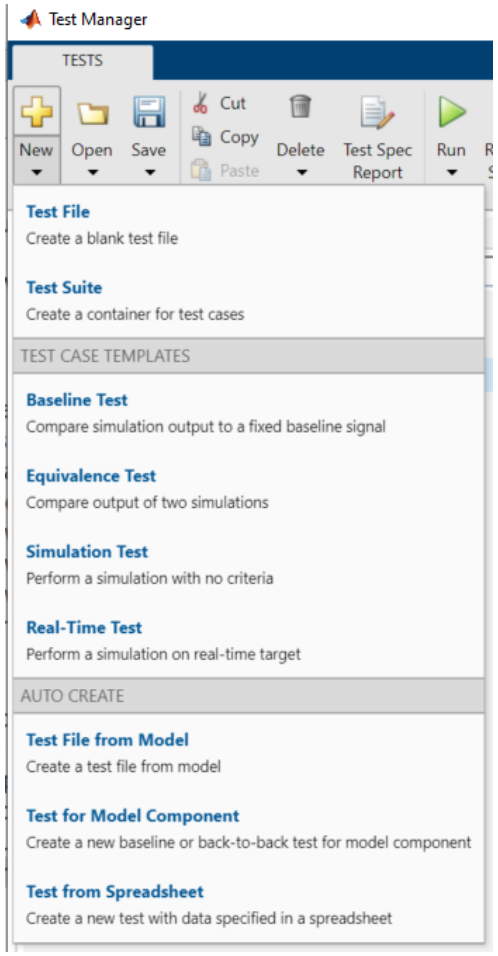
Created Test Harness to isolate Component Under Test



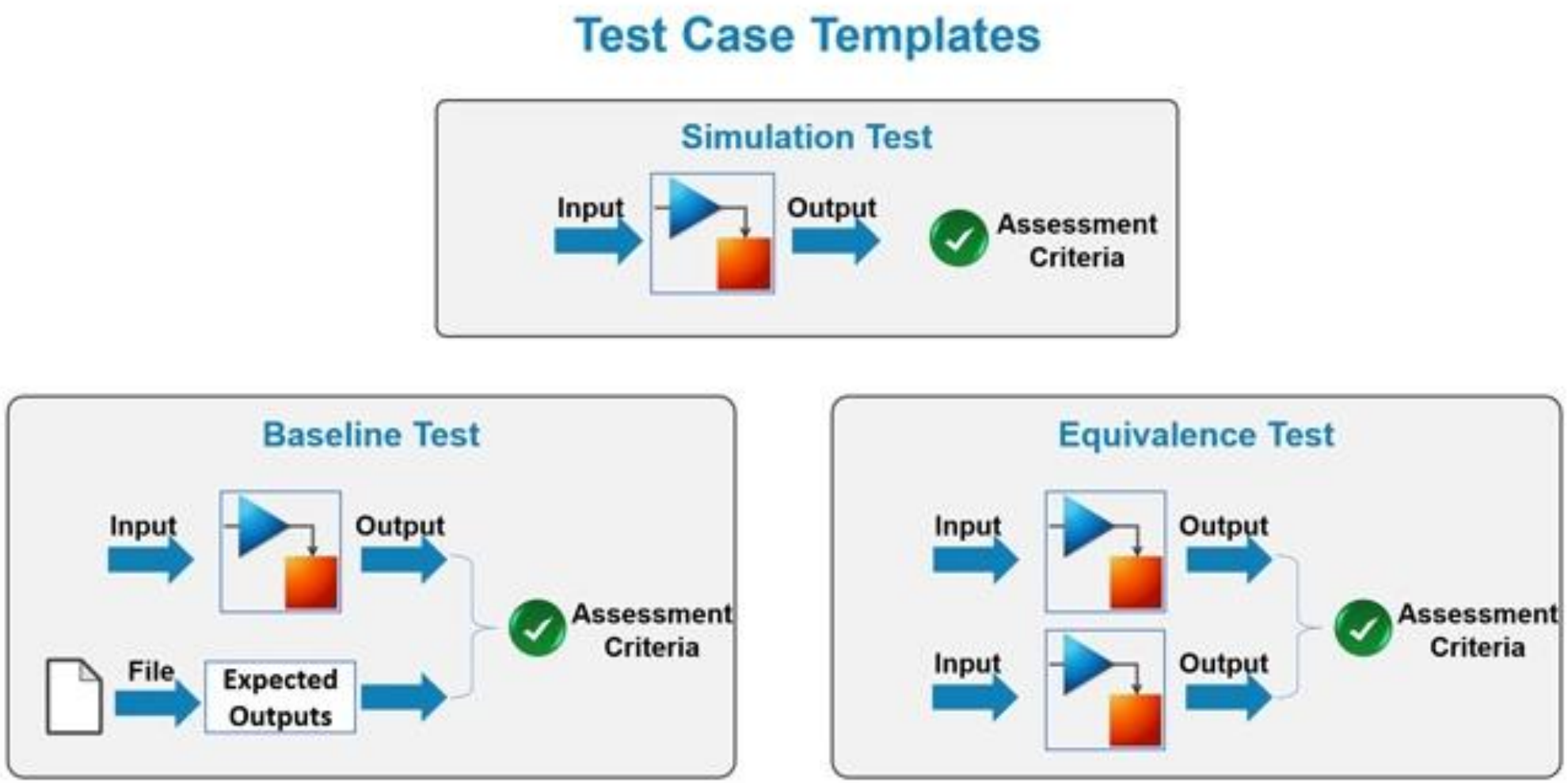
Authoring tests using Signal Editor



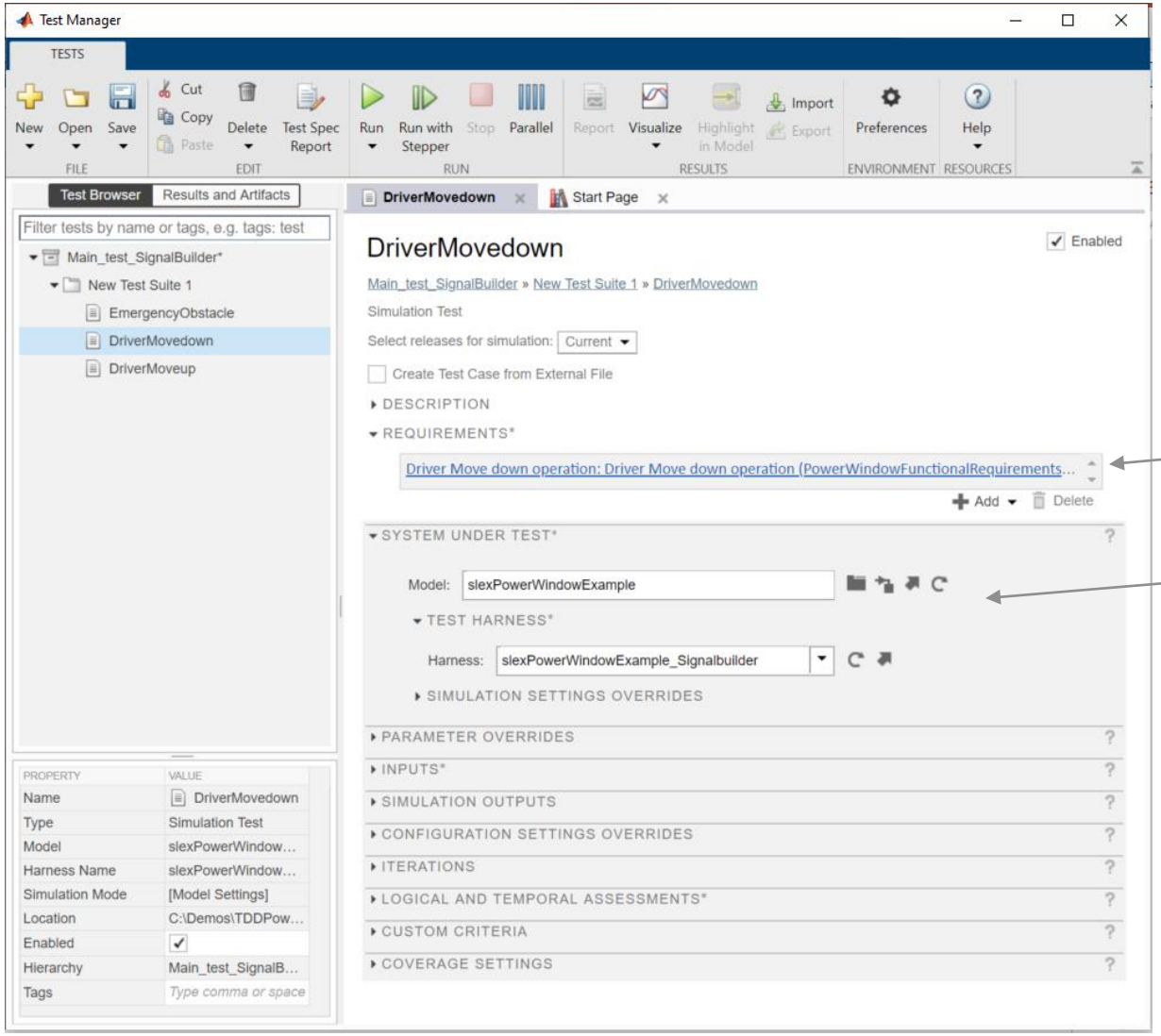
Use templates and wizards to automate test case creation



Use templates and wizards to automate test case creation



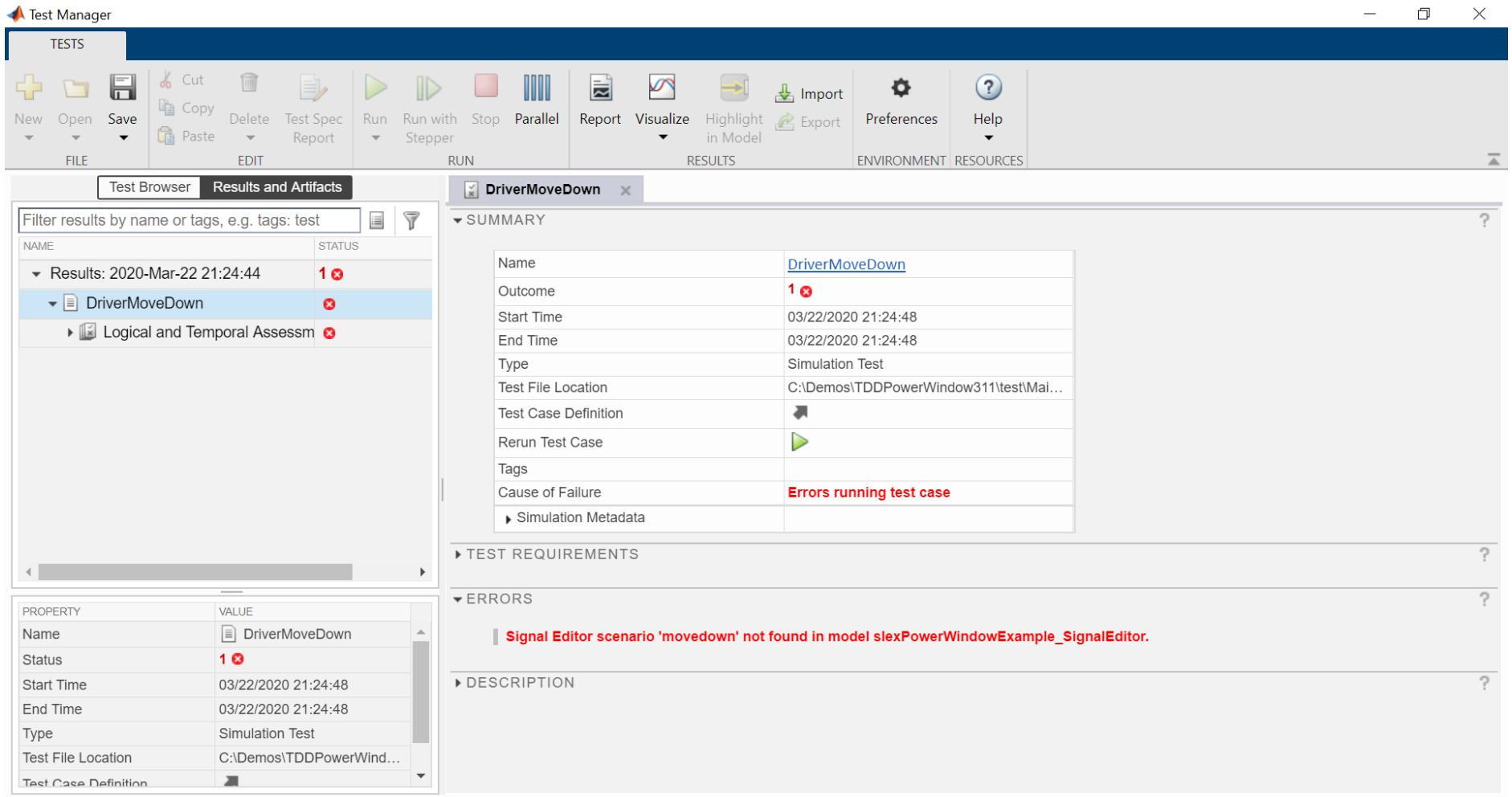
Create Simulation Test and link to requirement



Link to requirements

Specify model to test

Test fails due to compilation error

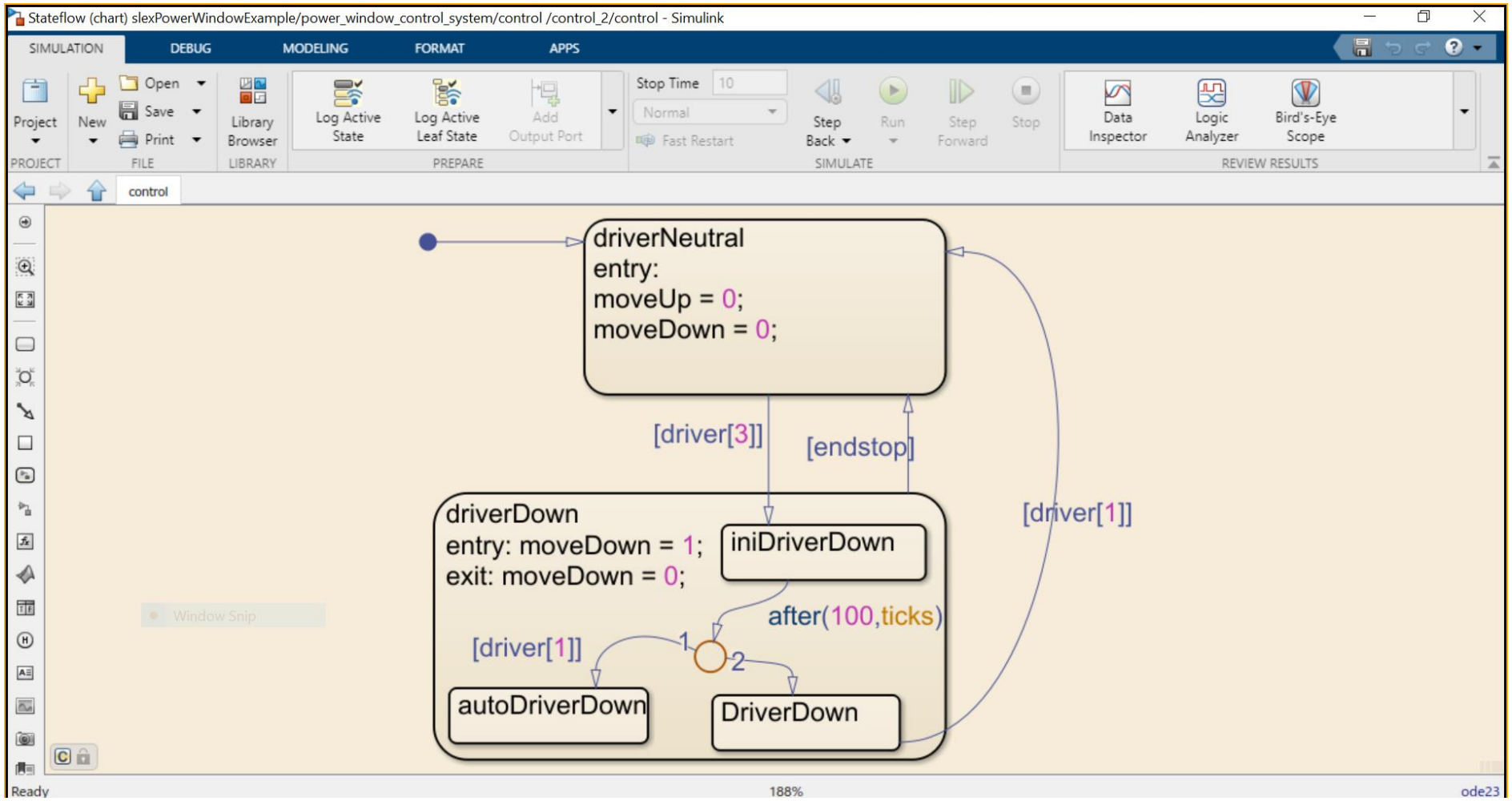


1. Create a test

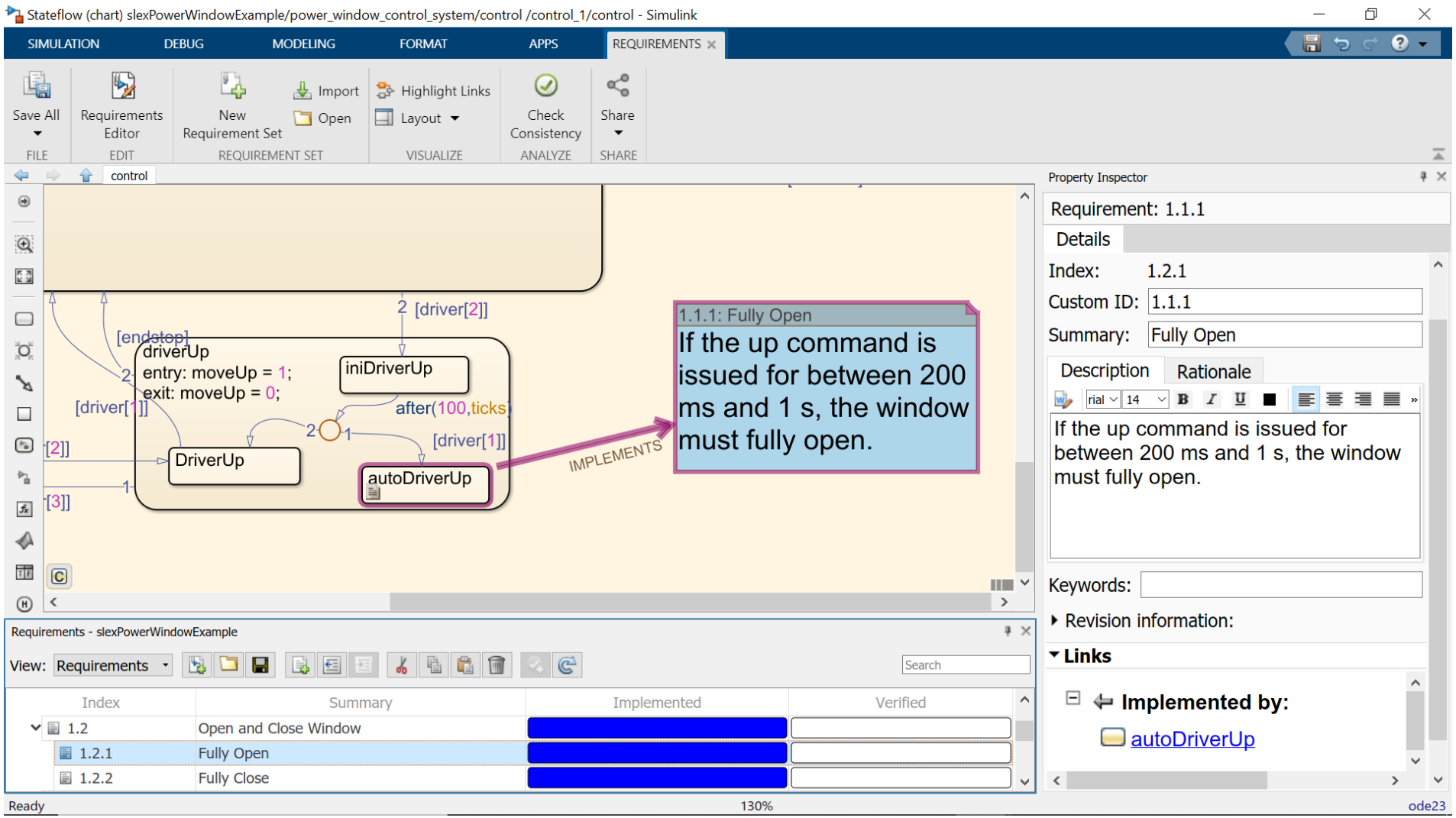
2. **Implement enough for test to pass**

3. Refactor

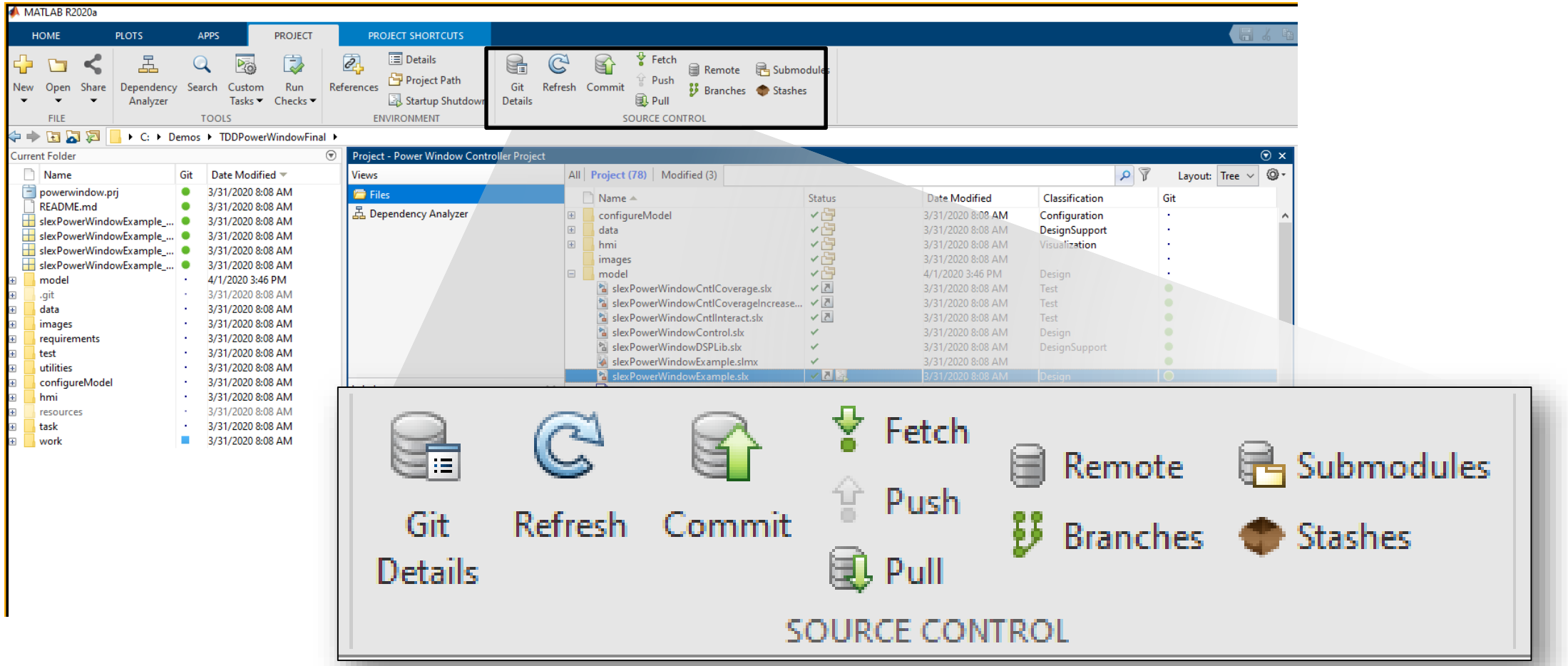
Implement enough to get test to pass



Linking implementation to requirements

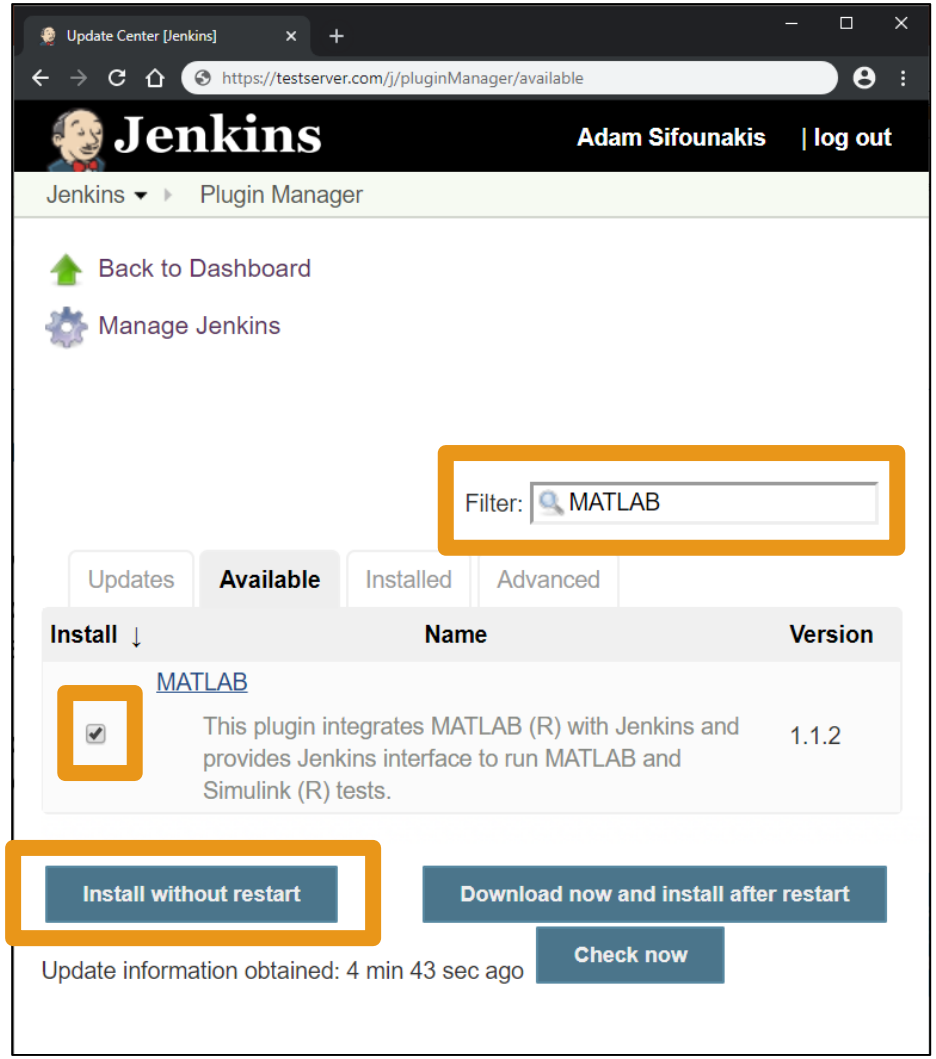


Managing artifacts with source control directly from Projects



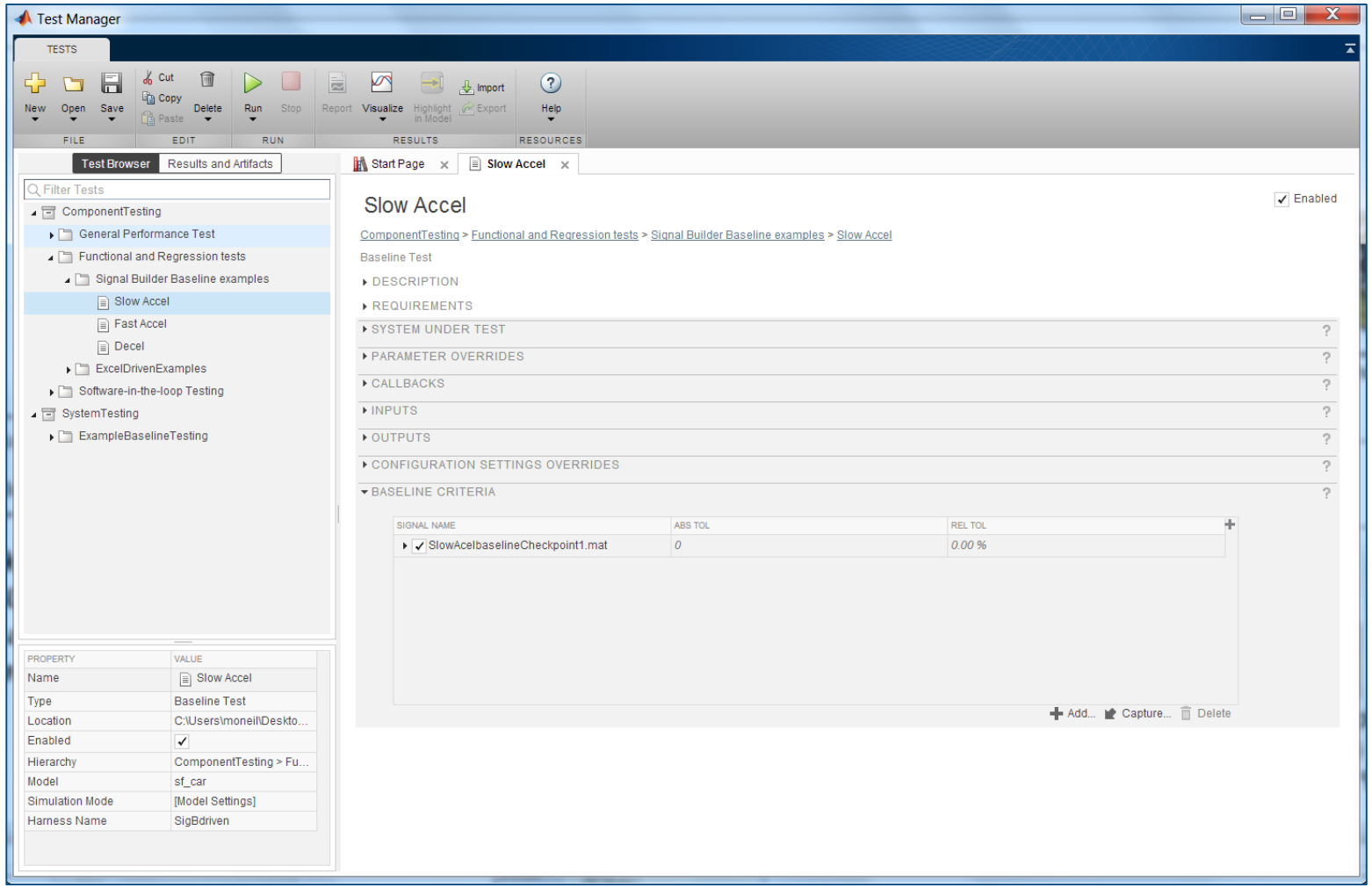
Scale and automate testing with Continuous Integration

- Schedule automatic code and model testing
- Access MATLAB Plugin for Jenkins



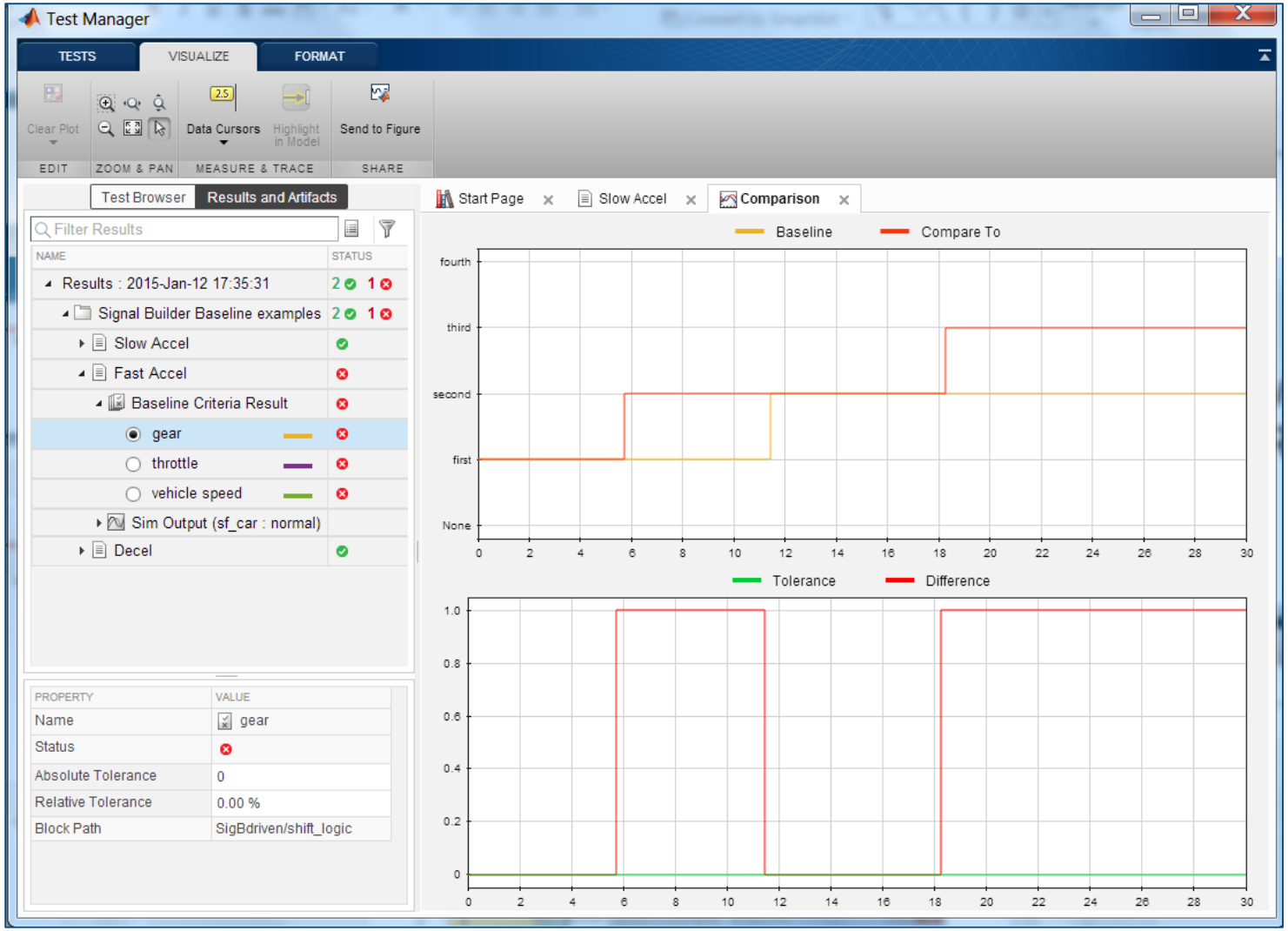
Executing test with Test Manager

- Group into suites and test files
- Execute individual or batch



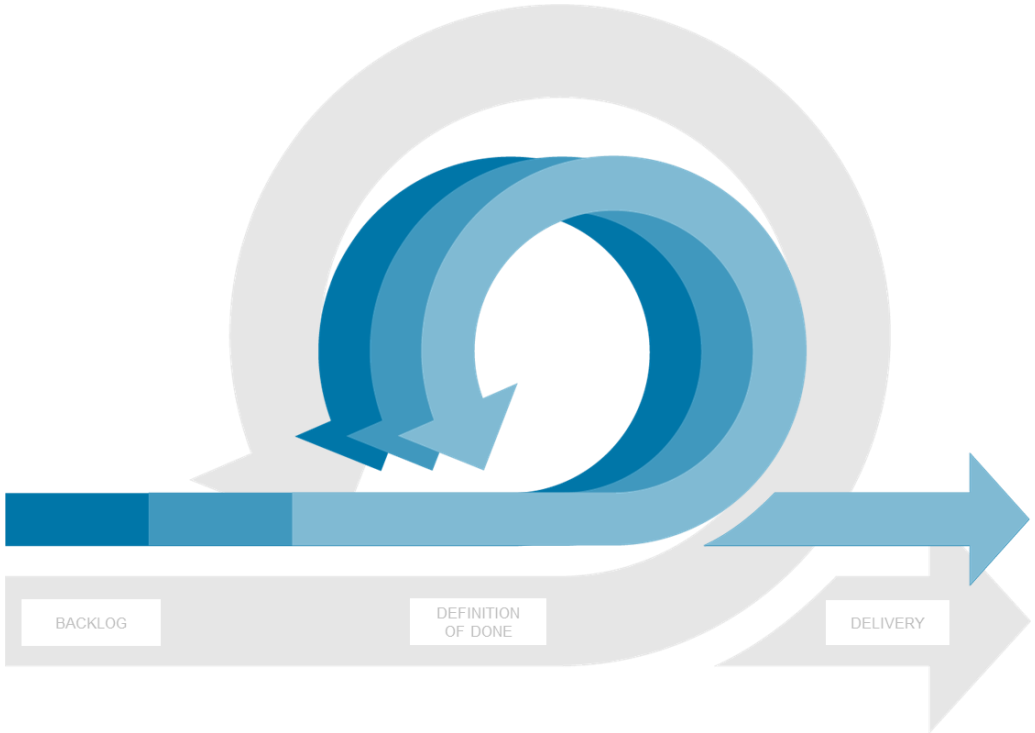
Analyzing and debugging results with Test Manager

- View result summary
- Debug using Simulation Data Inspector
- Archive, export, and report results



Executing all tests until they pass

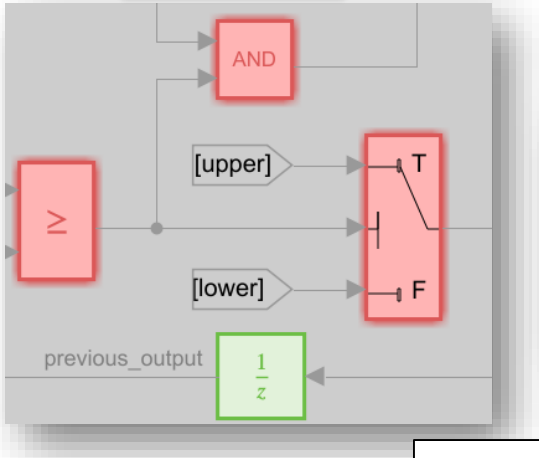
PowerWindowControlUnitTest_sig	6
DriverMoveDown	
DriverMoveUp	
EmergencyObstacle	
Detect Object	
Fully Close Window	
PassengerMovedown	



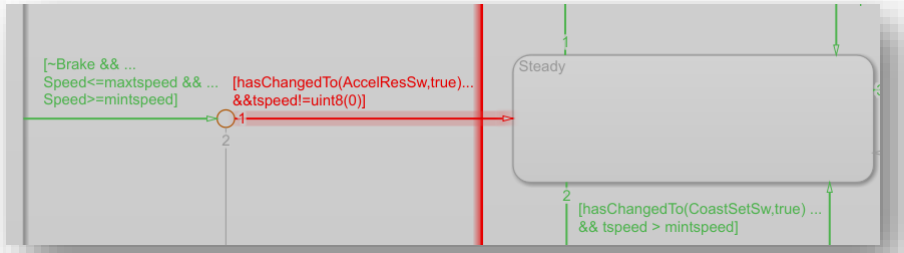
Measuring testing completeness with coverage

- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors

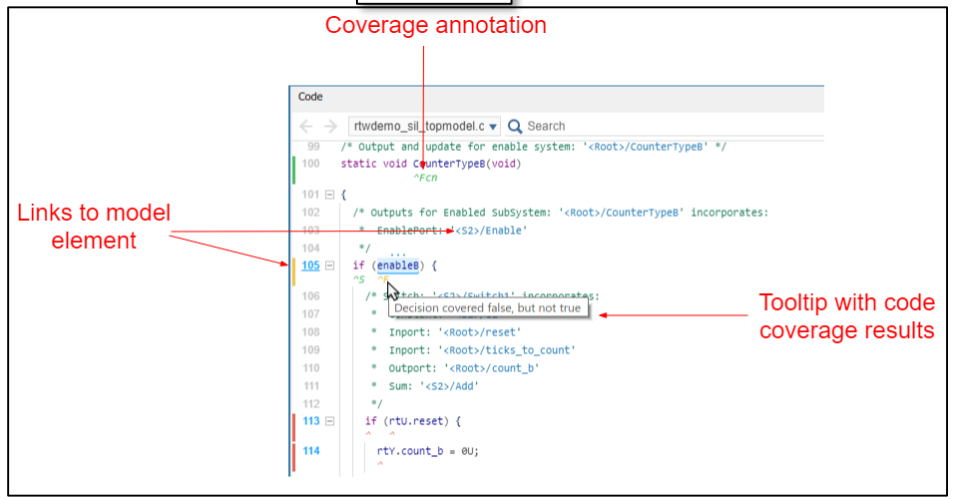
Simulink



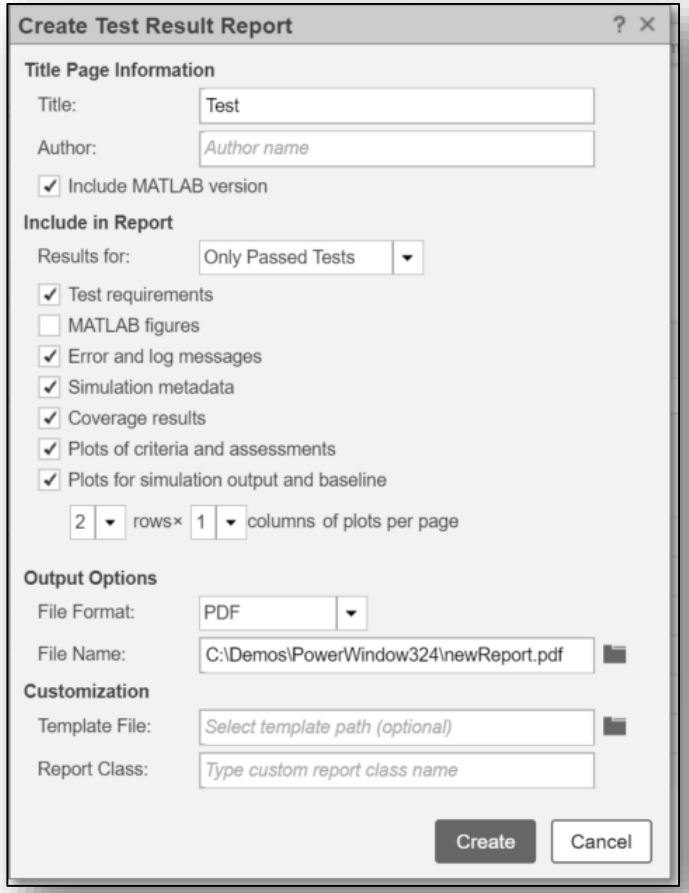
Stateflow

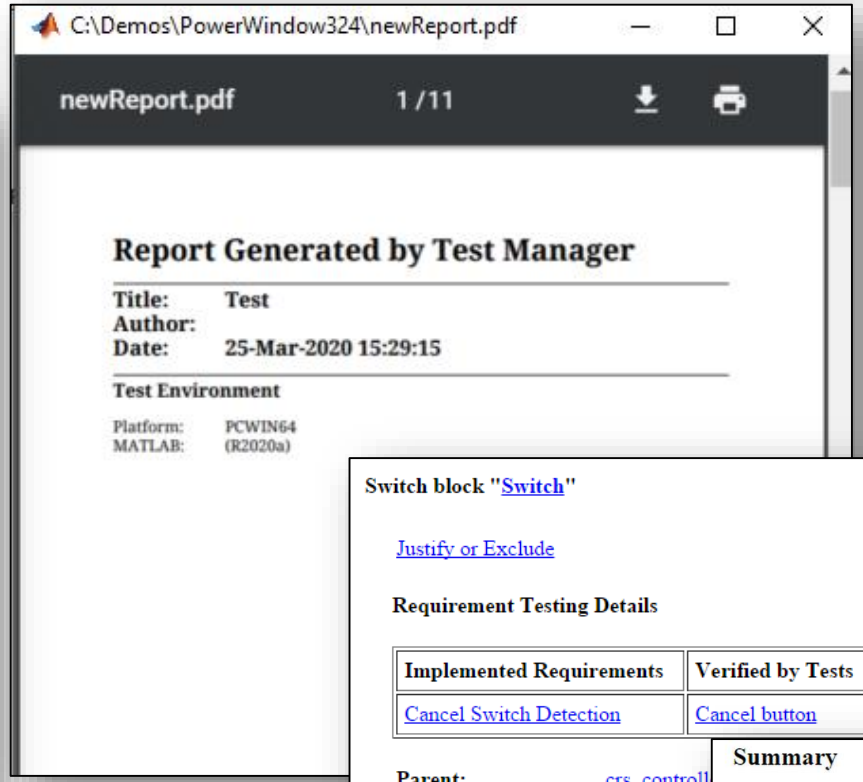


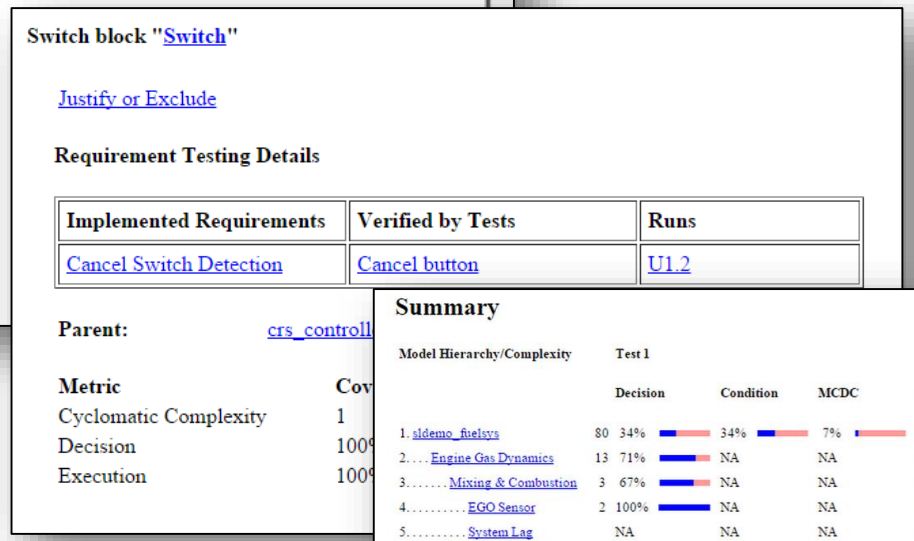
Code

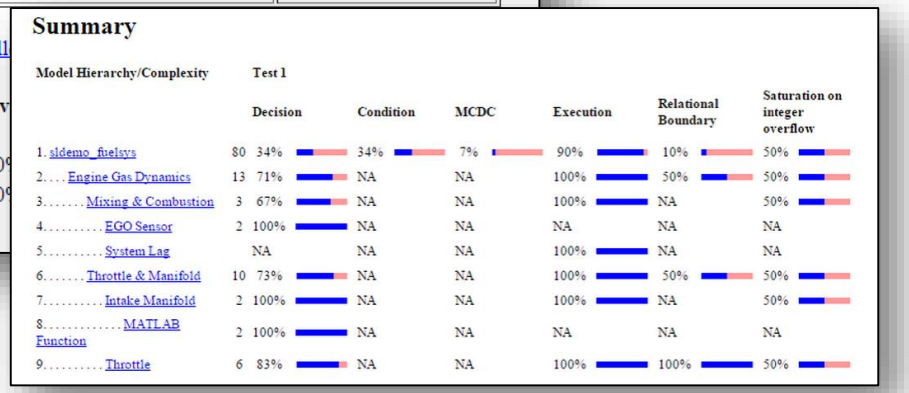


Generating test reports for audits and reviews





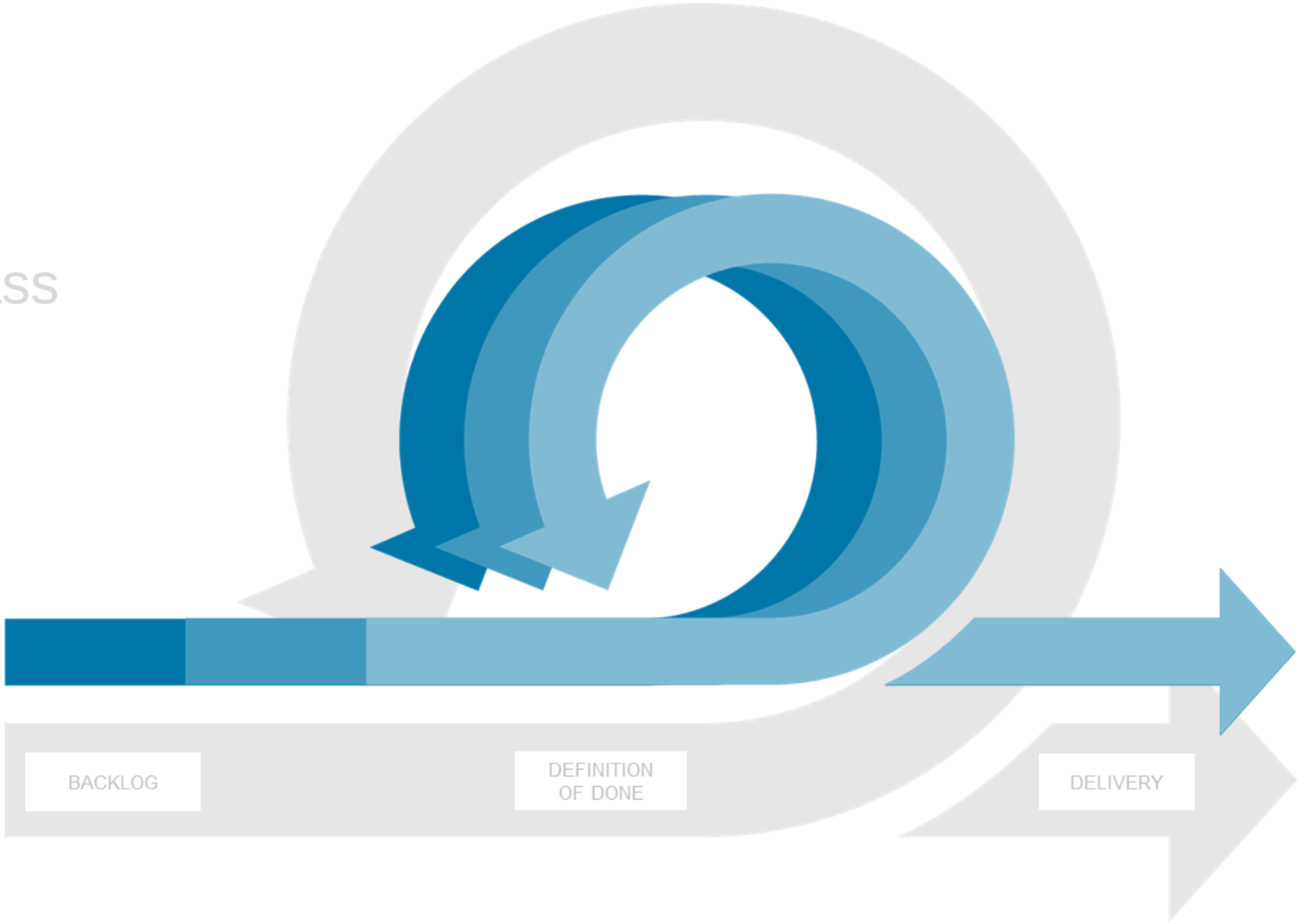




1. Create a test

2. Implement enough for test to pass

3. **Refactor**

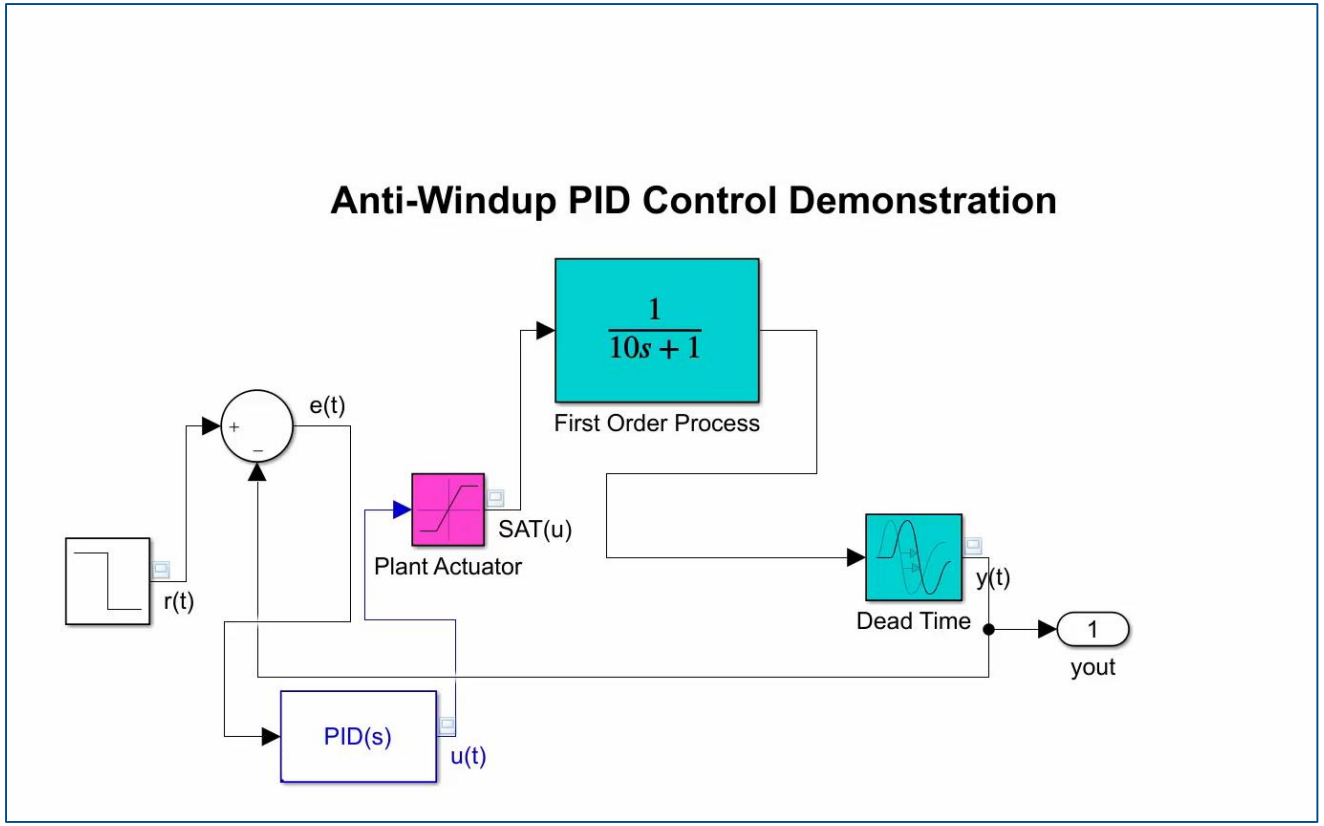


Refactoring

- Refactoring is the process of changing software in such a way that it does not alter the external behavior of the code yet improves its internal structure

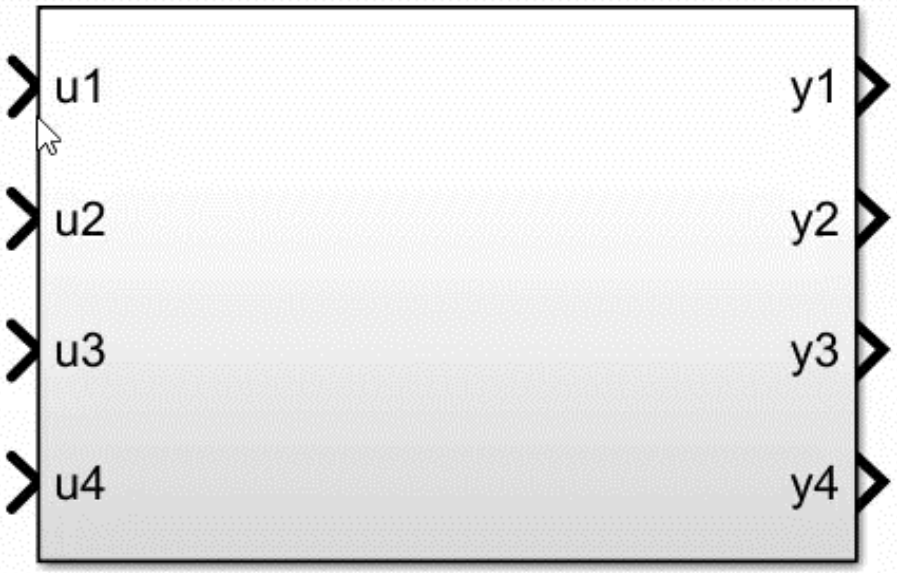
Refactoring takes many shapes and forms

- Rearranging Layout



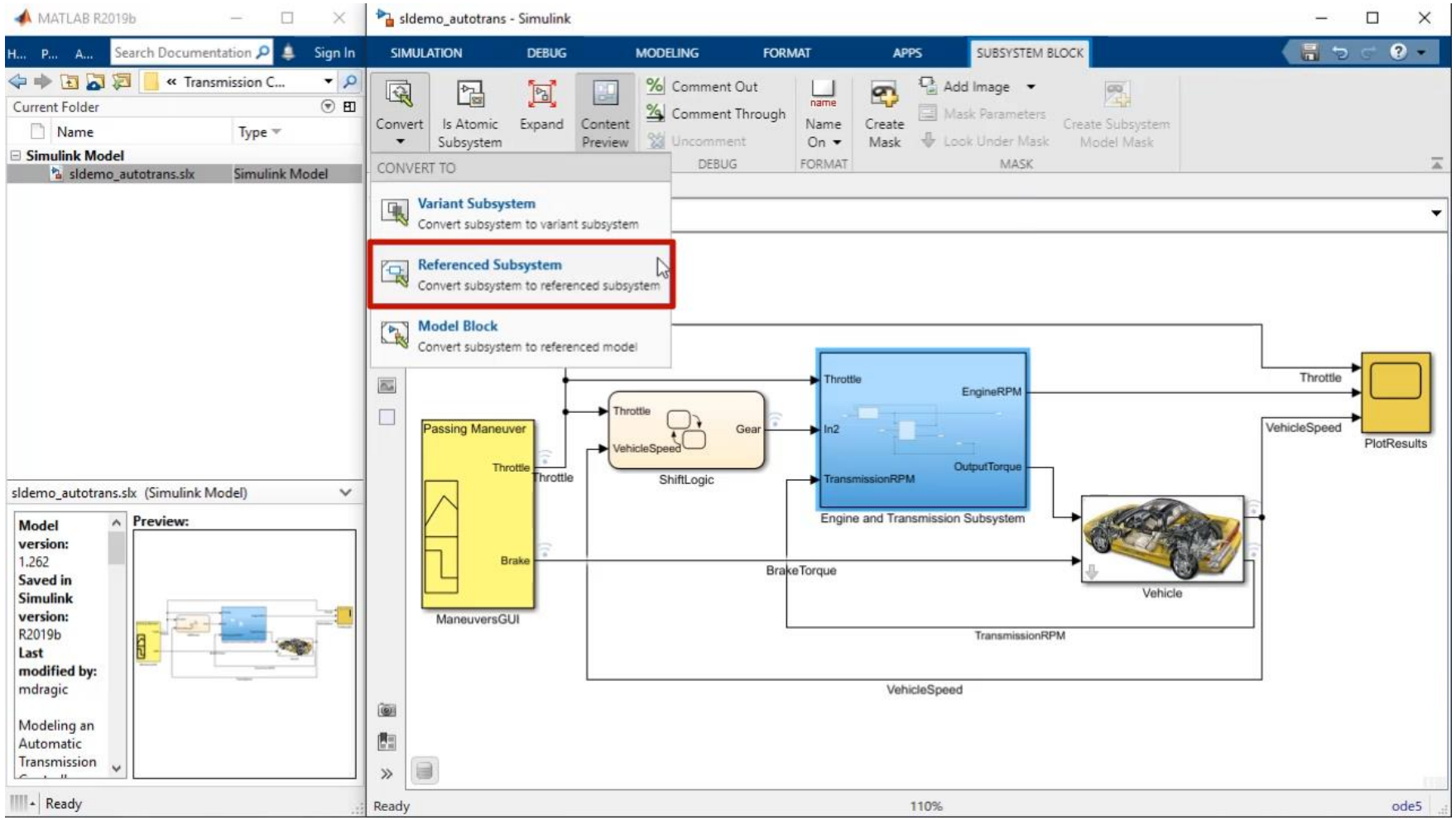
Refactoring takes many shapes and forms

- Rearranging Layout



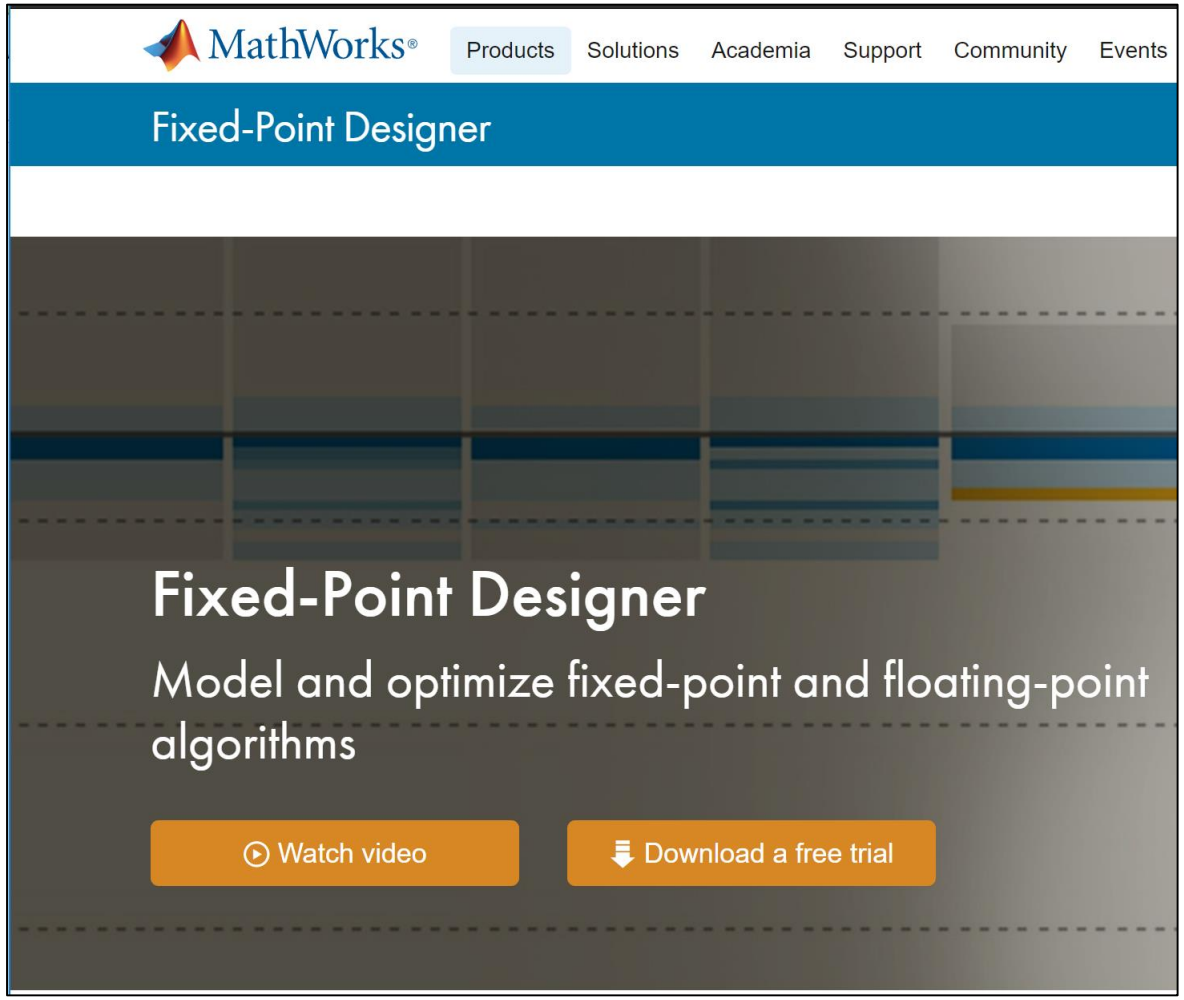
Refactoring takes many shapes and forms

- Rearranging Layout
- Restructuring Hierarchy



Refactoring takes many shapes and forms

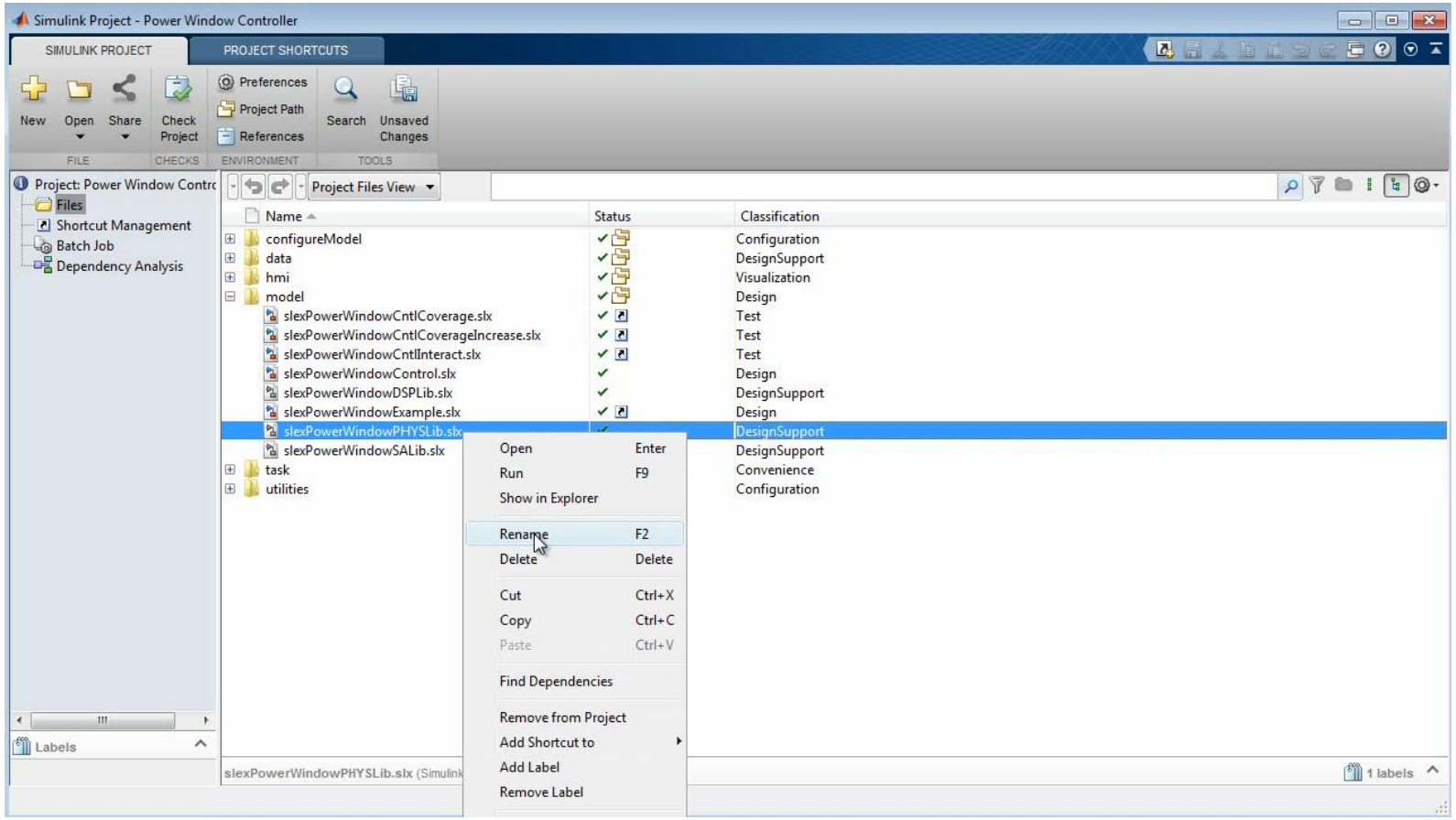
- Rearranging Layout
- Restructuring Hierarchy
- Optimizing Implementation



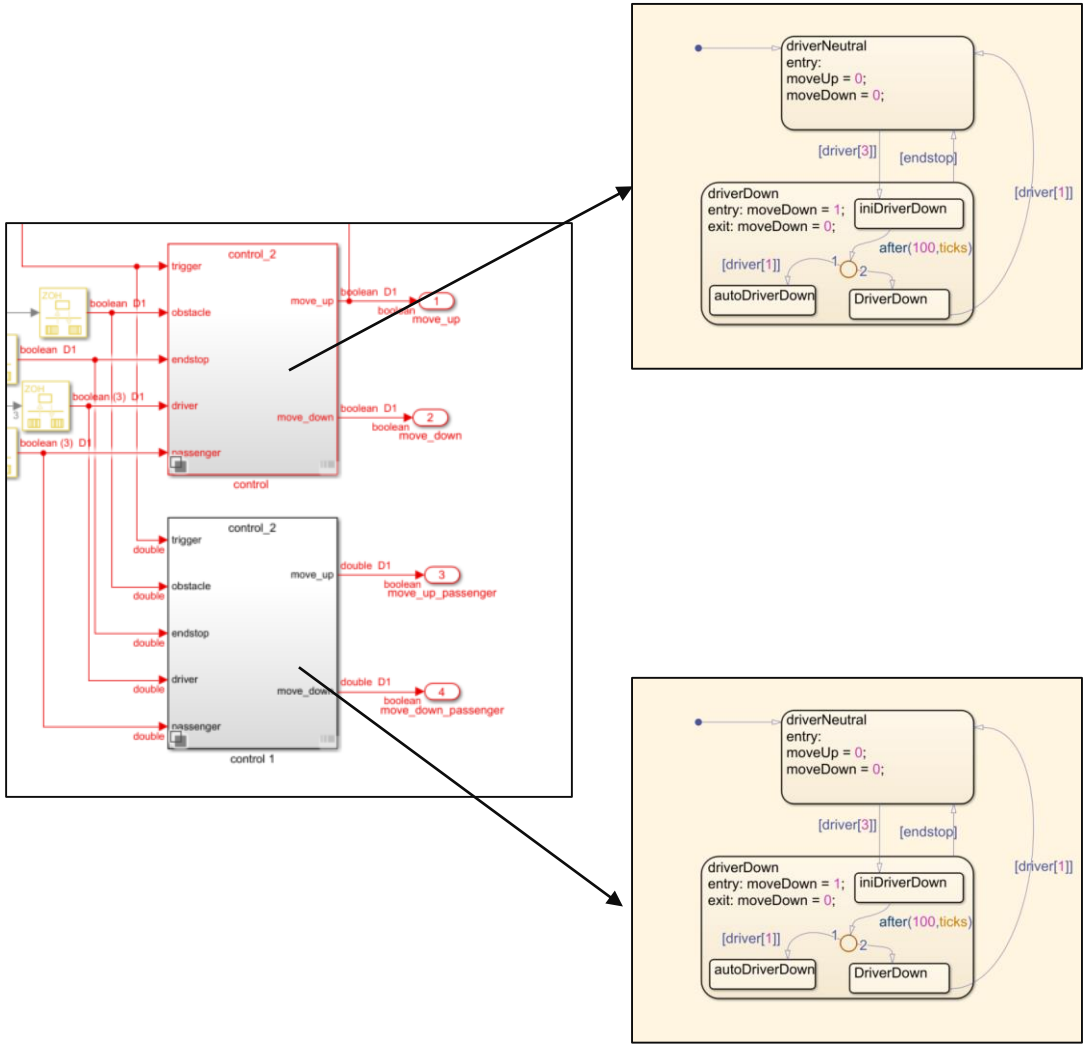
Refactoring takes many shapes and forms

- Rearranging Layout
- Restructuring Hierarchy
- Optimizing Implementation
- Project-wide Renaming

…. and many more!

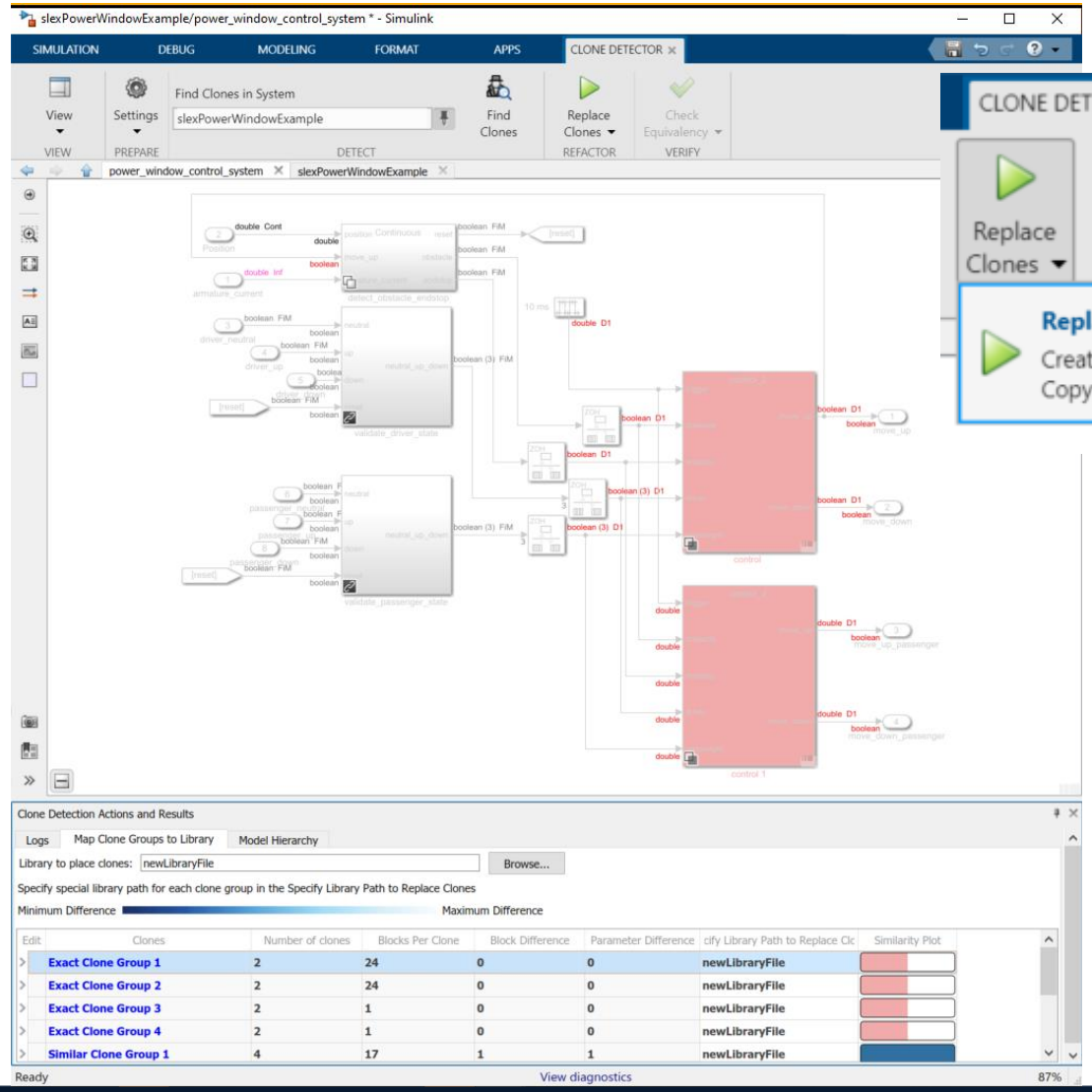


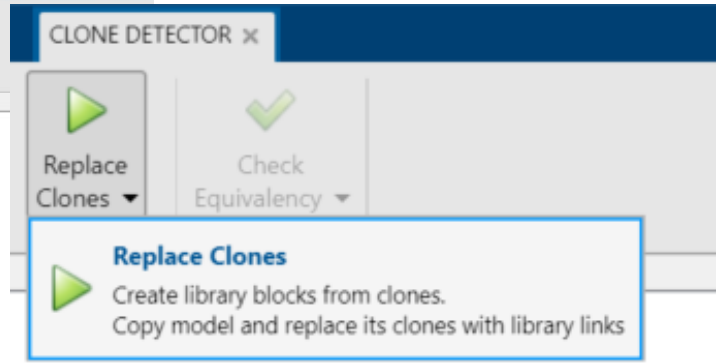
Refactor by consolidating redundant Stateflow chart



Driver and Passenger Controls are identical

Detecting clones with Clone Detector App





Test Driven Development Cycle

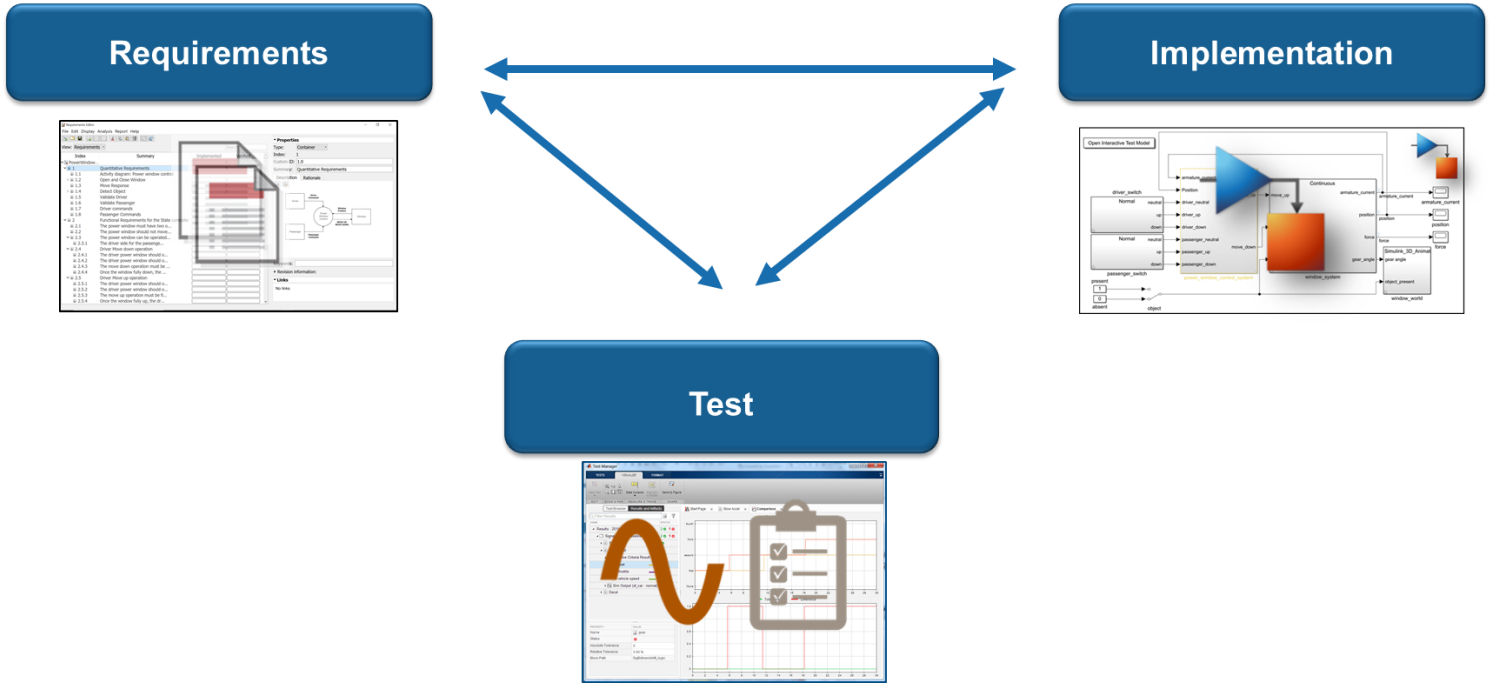
1. Create a test
2. Implement enough for test to pass
3. Refactor

Conclusion and key takeaways

Simulink provides an integrated framework for TDD

Systematically verify requirements

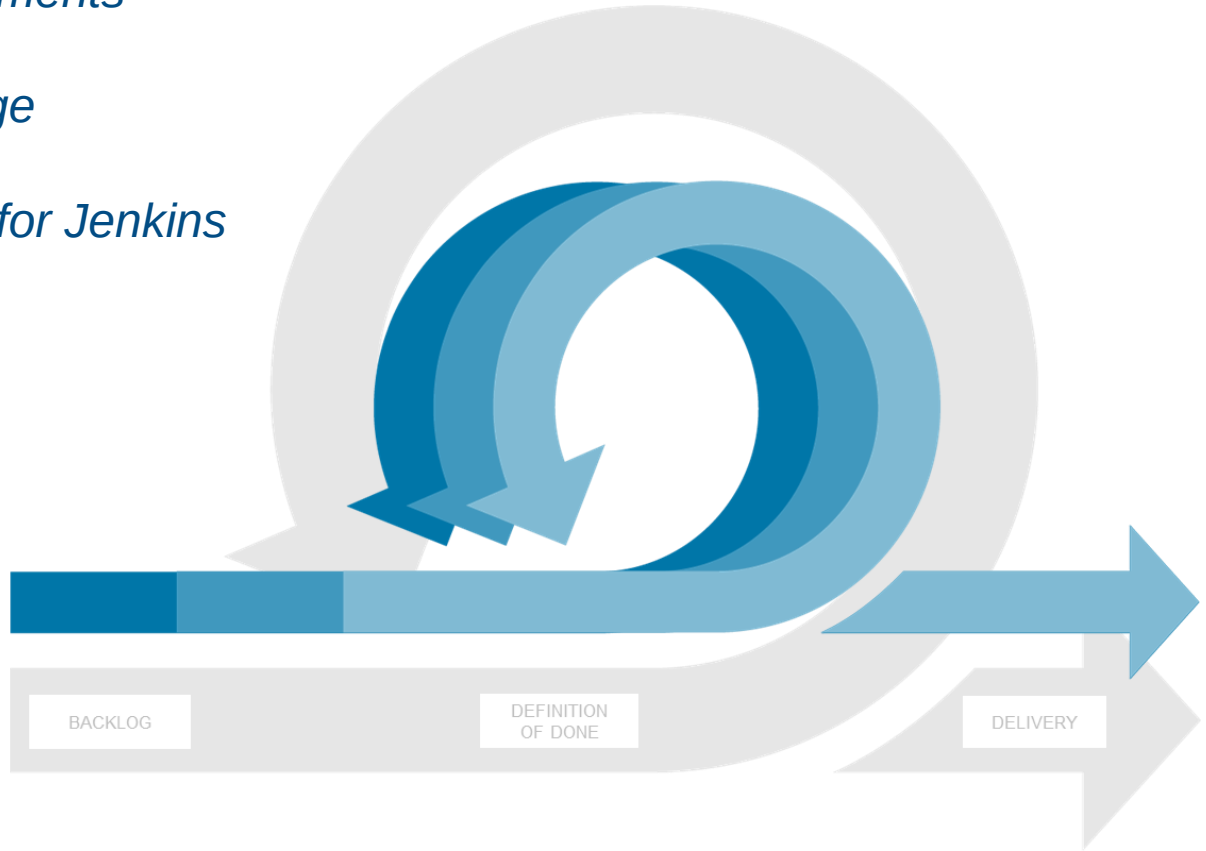
Automate testing to deliver working systems faster



Test Driven Development

powered by MATLAB and Simulink

- **Model-Based Design** – *Simulink and Stateflow*
- **Manage Requirements** – *Simulink Requirements*
- **Author and Execute Tests** – *Simulink Test*
- **Measure Test Completeness** – *Simulink Coverage*
- **Refactor and Verify Compliance** – *Simulink Check*
- **Continuous Integration** – *MATLAB Plug in for Jenkins*
- **Organize, Manage and Share** – *Projects*



Learn more

- Agile System Development with Model-Based Design
- Agile Model-Based Design: Accelerating Simulink Simulations in Continuous Integration Workflows
- Verification, Validation, and Test Solution Page
- Continuous Integration Solution Page