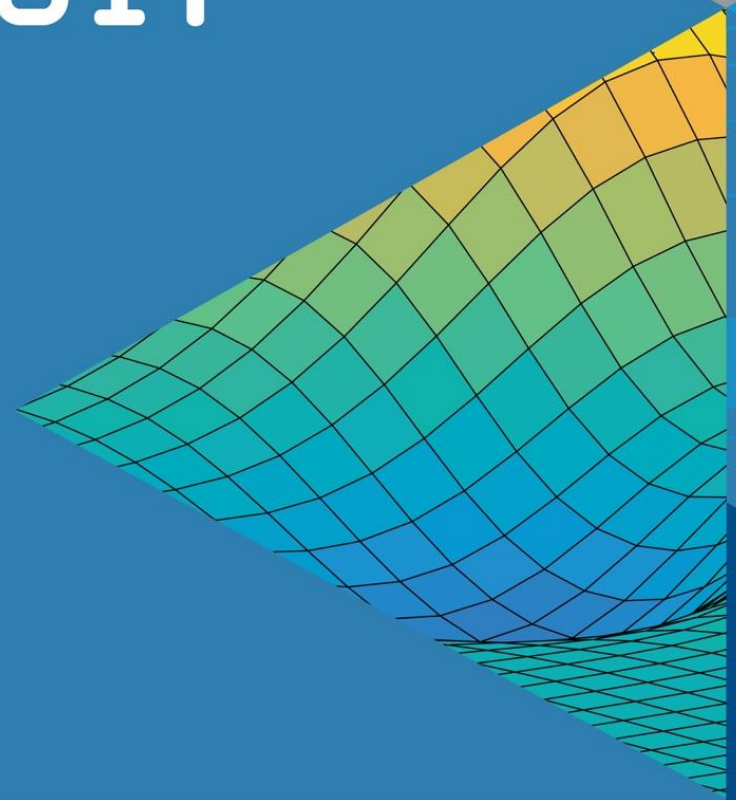
MATLAB EXPO 2017

Testing Simulink Models

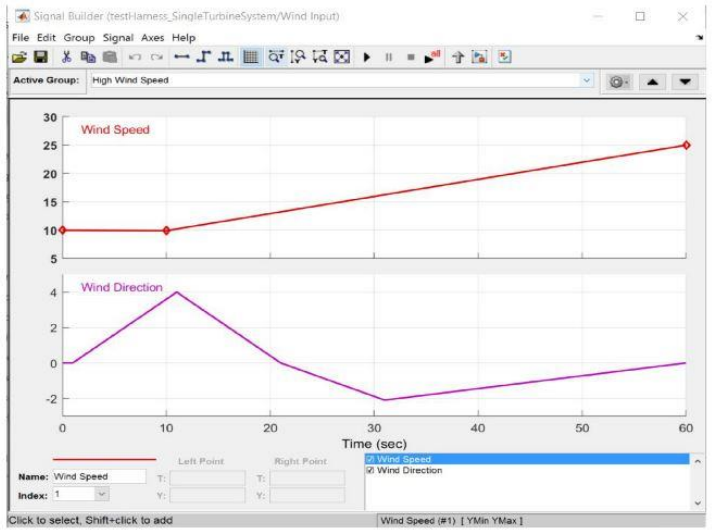
Fraser Macmillen



Test Infrastructure

- Model set-up — desired parameters, variants, operating point, etc.
  e.g. test start up script
- Model stimulus — desired inputs driving the model
  e.g. signal builder block, .xlsx, test sequence
- Views of behaviour — signal traces, read-outs, animations, etc.
  e.g. scopes, simulation data inspector
- Verification of behaviour — desired behaviour is checked
  e.g. verification blocks, post-simulation scripts

Wind Turbine



Common challenges:

- Problem: cannot do anything unless a particular script is run first
  Solution: use project startup, data dictionaries, models always “ready to go”

- Problem: model is tied to particular means of stimulus (from file, signal builder, etc)
- Solution: use test harnesses + variants

- Problem: changes to the design and test mixed together
- Solution: save test infrastructure externally to your design; separate source control

- Problem: one person’s system is another person’s component
- Solution: model referencing, suitable interfaces

- Problem: performance degraded by infrastructure not needed for “my test”
- Solution: multiple harnesses / variants

Simulink Test

Why Simulink Test?

Saves you time:

- Creating / managing test infrastructure
- generating & (re)-running multiple tests
- reporting results
- a common test environment –
  everyone doing things in a consistent manner

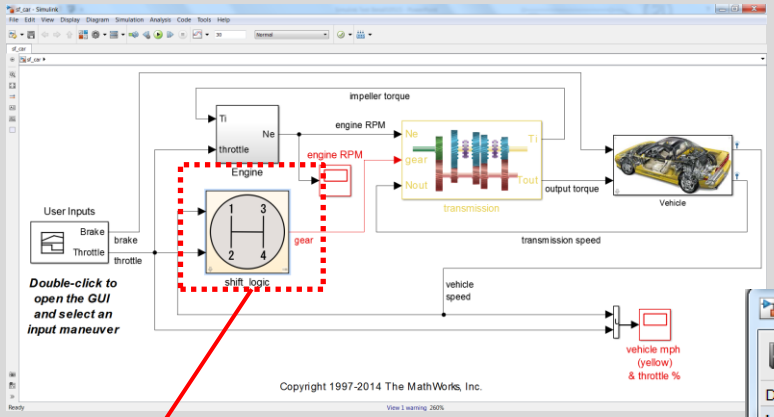
Gives you capability:

- new ways of authoring test scenarios
- easy integration with other tools
  (Requirements, Coverage, Test Generation, MATLAB Unit Test, Continuous Integration)


Time spent talking about Simulink Test at UK EXPO
Current Data
Projected Trend
Hours
Year
8
7
6
5
4
3
2
1
0
-1
2014
2015
2016
2017
2018
2019
2020
2021

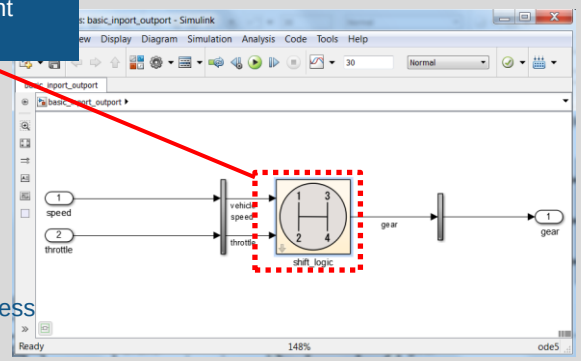
Simulink Test Overview

1. Test Harnesses	2. Test Stimulus Integration	3. Test Manager
• Synchronized, simulatable test environment	• Inputs and assessments based on logical, temporal conditions	• Author, execute, manage test cases • Review, export, report

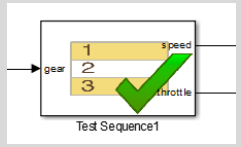


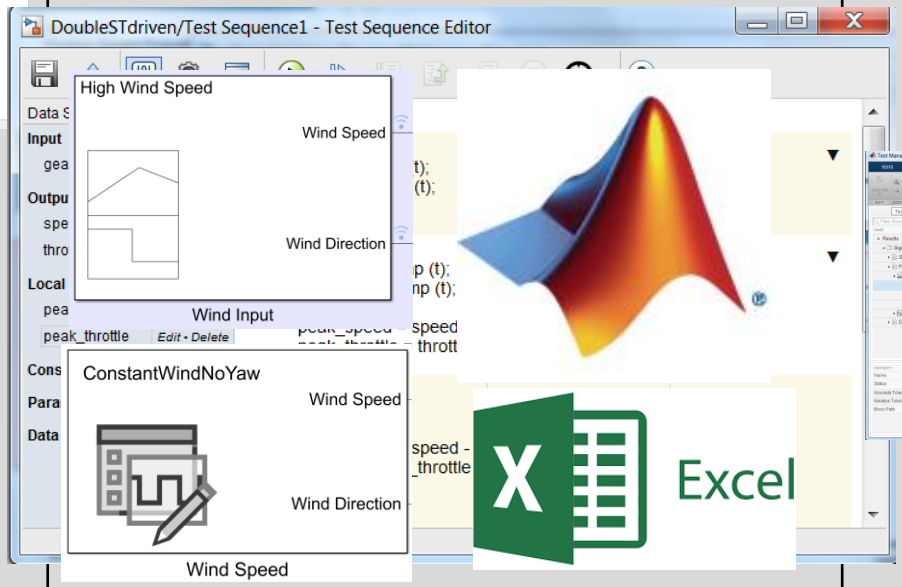
Main Model

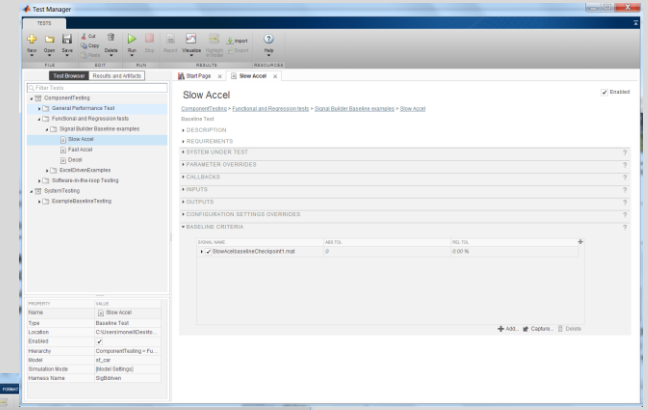
Component under test

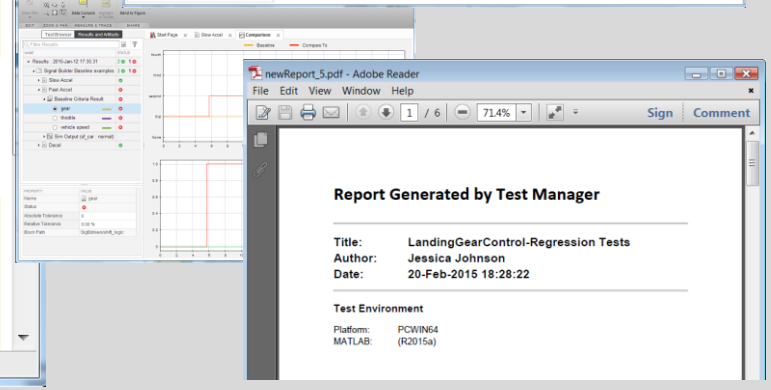


Test Harness





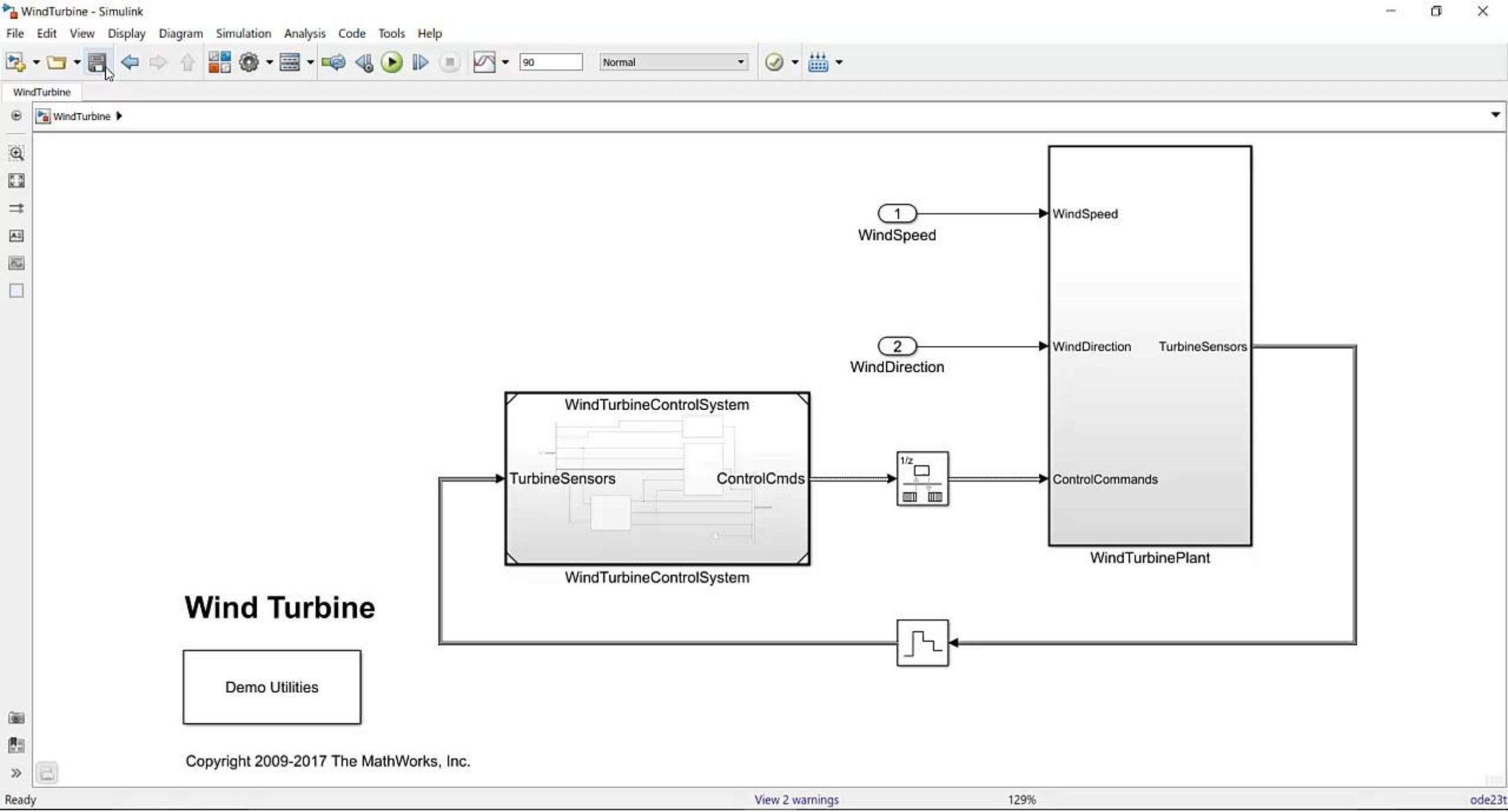




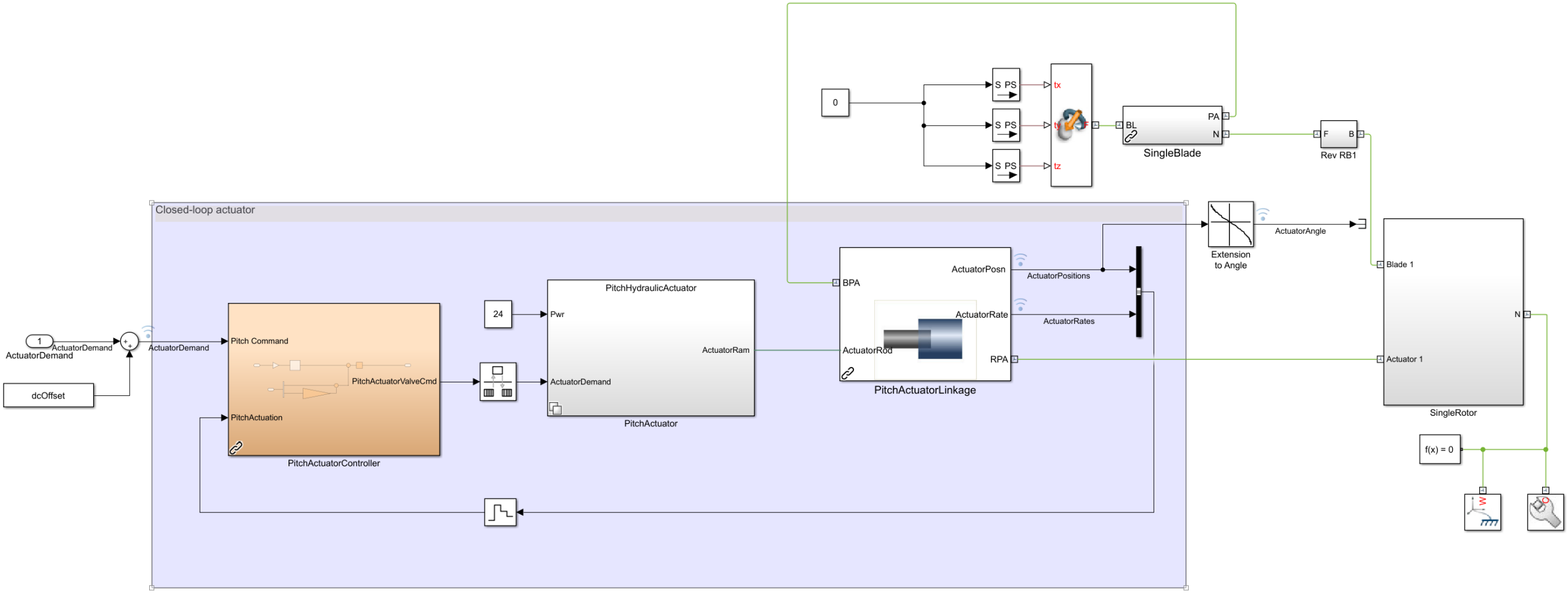
Agenda

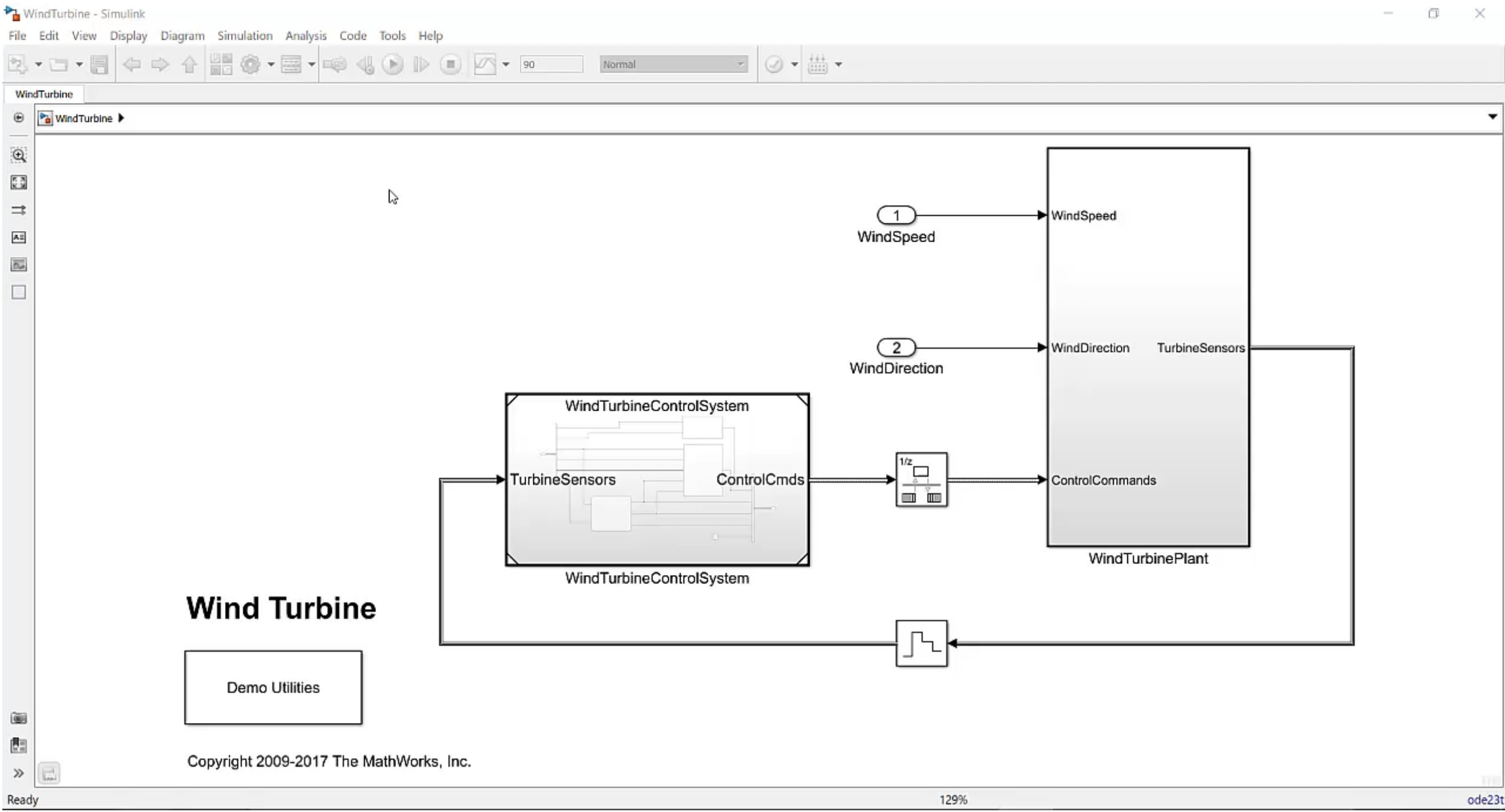
- Creating Test Harnesses
- Creating Test Cases
- Testing against Requirements
- Reporting
- Coverage analysis
- Multi-release regression testing
- Continuous integration

Test Harnesses


WindTurbine - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
90
Normal
WindTurbine
WindTurbine
1
WindSpeed
WindSpeed
2
WindDirection
WindDirection
TurbineSensors
WindTurbineControlSystem
TurbineSensors
ControlCmds
WindTurbineControlSystem
1/z
WindSpeed
WindDirection
TurbineSensors
ControlCommands
WindTurbinePlant
Wind Turbine
Demo Utilities
Copyright 2009-2017 The MathWorks, Inc.
Ready
View 2 warnings
129%
ode23t

What if you already have a harness model....




WindTurbine - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
WindTurbine
WindTurbine
1
WindSpeed
WindSpeed
2
WindDirection
WindDirection
TurbineSensors
WindTurbineControlSystem
TurbineSensors
ControlCmds
1/z
ControlCommands
WindTurbinePlant
WindTurbineControlSystem
Wind Turbine
Demo Utilities
Copyright 2009-2017 The MathWorks, Inc.
Ready
129%
ode23t

Common questions...

Do I need a separate harness for each test?

Test Harness Release Highlights

R2017a:

- Test harness import
- Create harnesses for components with physical (Simscape) connections
- More control over synchronisation

R2017b:

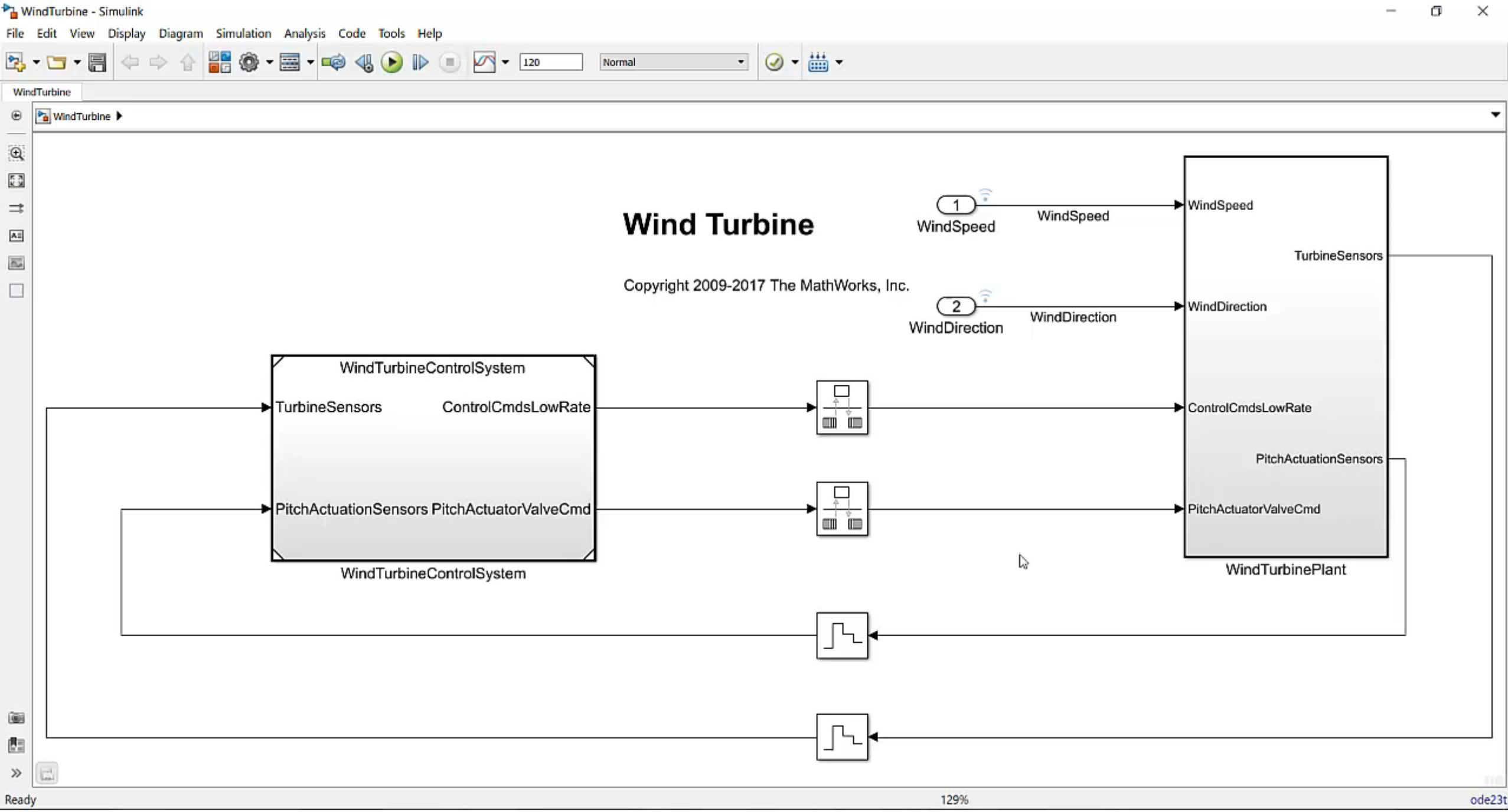
- Harness create/re-build callbacks
- Model comparison prior to synchronisation

Test Cases

&

Test Stimuli

Create a test case using the original signal builder

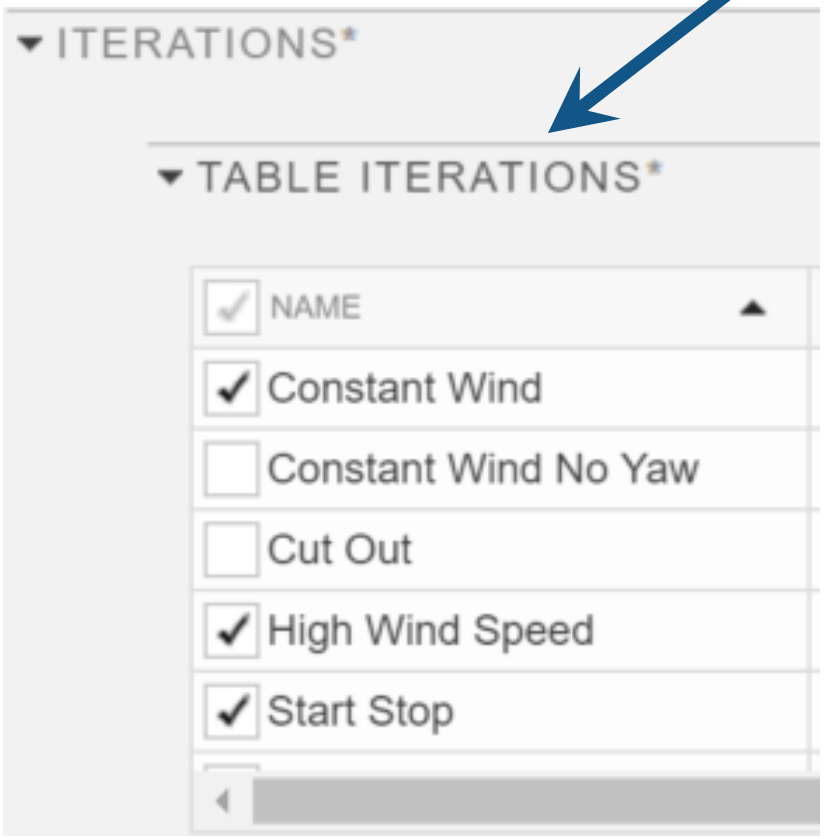

WindTurbine - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
120
Normal
WindTurbine
WindTurbine
Wind Turbine
Copyright 2009-2017 The MathWorks, Inc.
1
WindSpeed
WindSpeed
WindSpeed
2
WindDirection
WindDirection
WindDirection
WindTurbineControlSystem
TurbineSensors
ControlCmdsLowRate
PitchActuationSensors
PitchActuatorValveCmd
WindTurbineControlSystem
TurbineSensors
ControlCmdsLowRate
PitchActuationSensors
PitchActuatorValveCmd
WindTurbinePlant
Ready
129%
ode23t

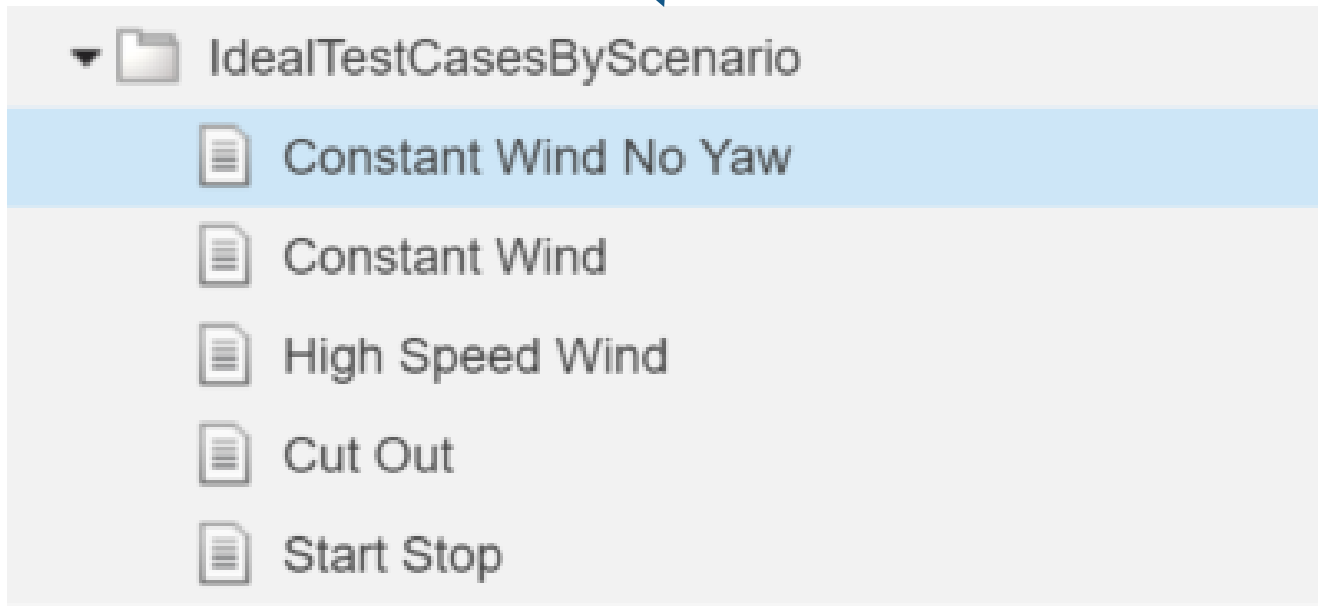
What have we done so far....

- Created and imported test harnesses
- Created a test case for running multiple simulations (iterations) with different scenarios

Common questions...

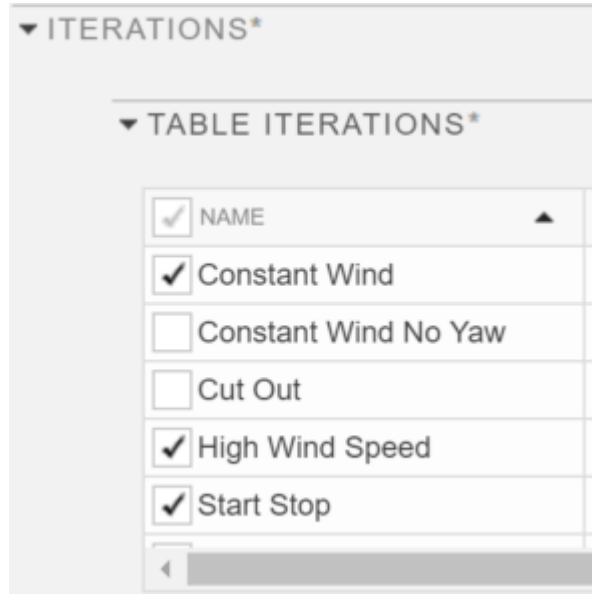
When should I use iterations vs multiple test cases?





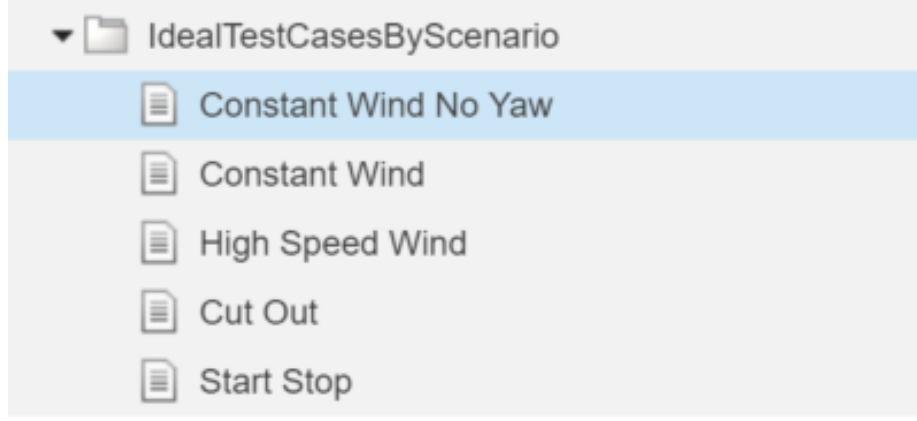
Use iterations if:

- Same model/harness & test type
- Same set-up (callbacks)
- Usually run together
- Relate to same requirements(s)
- Can use fast-restart



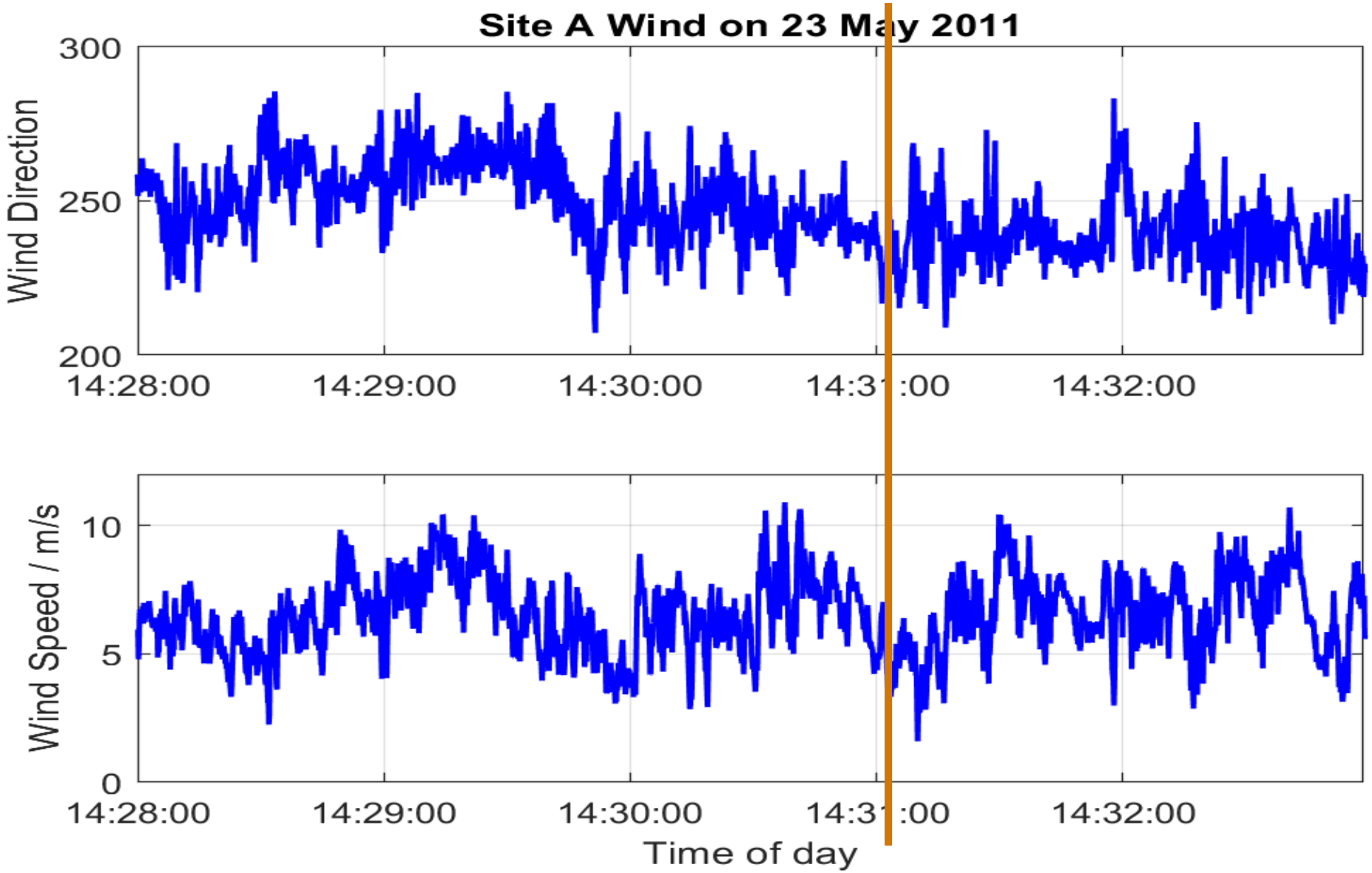
Use separate test cases if:

- Need independent configuration control
- Different model/harness/test type or callbacks
- Relate to distinct requirements
- Distinct control of coverage



Create a test case using real-world recorded data

My data

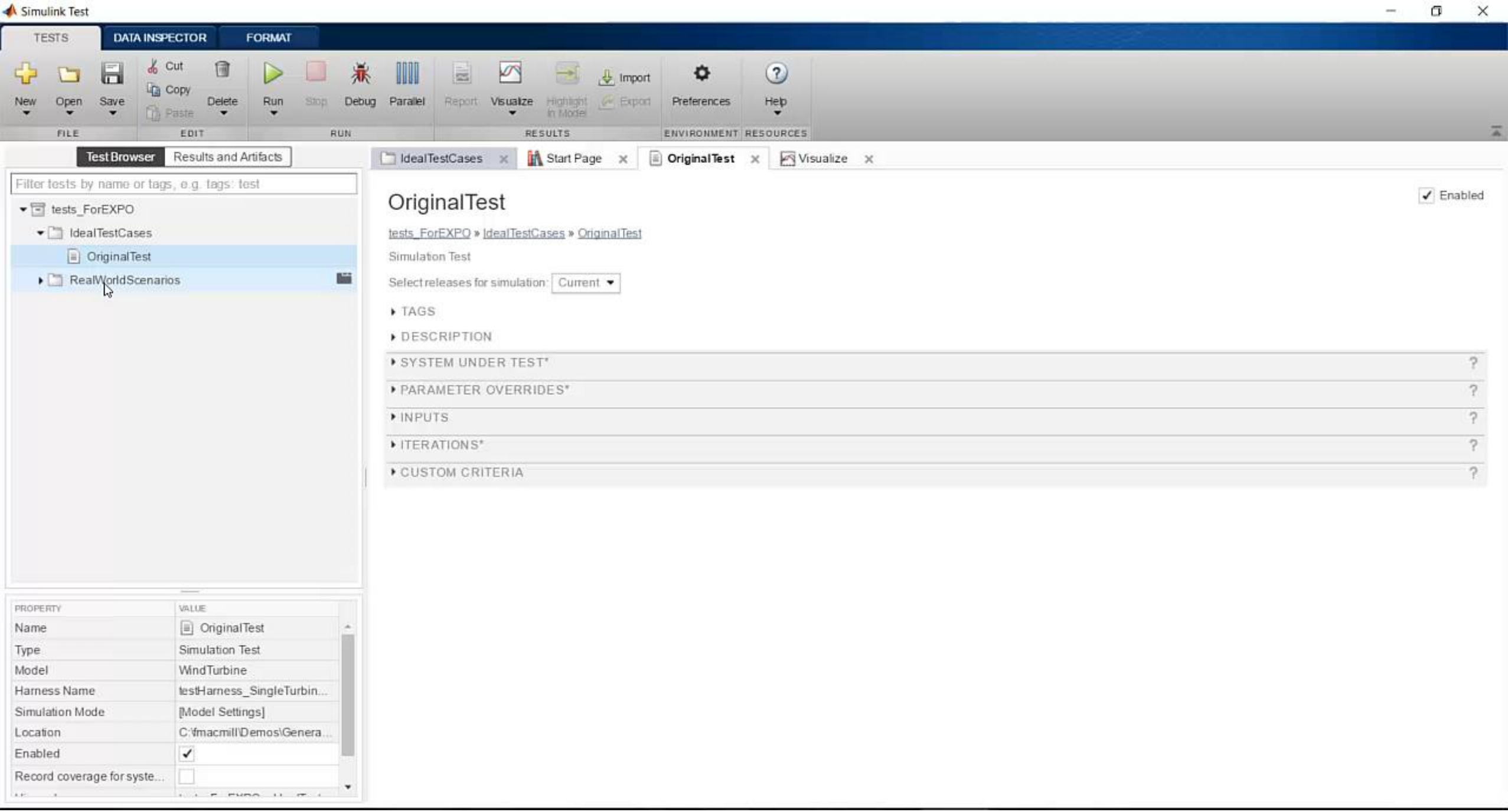

Site A Wind on 23 May 2011
Wind Direction
300
250
200
14:28:00
14:29:00
14:30:00
14:32:00
Wind Speed / m/s
10
5
0
Time of day

Importing time-stamped data from Excel or text files

Time	WindSpeed	WindDirection
00:00:00.175	14.59	214.9
00:00:00.306	14.47	212.3
00:00:00.437	16.1	208.5
00:00:00.568	17.94	209.4
00:00:00.700	17.53	210.9
00:00:00.831	16.93	219.6
00:00:00.962	15.25	218.2
00:00:01.093	12.73	220.1
00:00:01.224	13.71	212.2
00:00:01.355	11.89	218.6
00:00:01.486	15.94	212.2
00:00:01.617	16.51	208.1
00:00:01.748	17.11	211.8

```
% pre-process .xlsx file
% get import options
importOptions = detectImportOptions('SiteWindDataRecorded.xlsx')
% set sheet
importOptions.Sheet = '2011_05_23';
% tell it that Time is in a date-time format
importOptions = setvartype(importOptions,'Time','datetime');
importOptions = setvaropts(importOptions,'Time', 'DatetimeFormat', 'HH:mm:ss.SSS');
% read data in
T = readtable('SiteWindDataRecorded.xlsx',importOptions);
% convert to timetable
TT = table2timetable(T);
% re-sample to 1sec intervals
TTT = retime(TT,'secondly','nearest');
```

Time	WindSpeed	WindDirection
0	14.59	214.9
1	15.25	218.2
2	16.46	212.2
3	16.08	207.3


Simulink Test
TESTS
DATA INSPECTOR
FORMAT
New
Open
Save
Cut
Copy
Paste
Delete
Run
Stop
Debug
Parallel
Report
Visualize
Highlight in Model
Import
Export
Preferences
Help
FILE
EDIT
RUN
RESULTS
ENVIRONMENT
RESOURCES
Test Browser
Results and Artifacts
Filter tests by name or tags, e.g. tags: test
tests_ForEXPO
IdealTestCases
OriginalTest
RealWorldScenarios
IdealTestCases
Start Page
OriginalTest
Visualize
OriginalTest
Enabled
tests_ForEXPO » IdealTestCases » OriginalTest
Simulation Test
Select releases for simulation: Current
TAGS
DESCRIPTION
SYSTEM UNDER TEST*
PARAMETER OVERRIDES*
INPUTS
ITERATIONS*
CUSTOM CRITERIA
PROPERTY
VALUE
Name
OriginalTest
Type
Simulation Test
Model
WindTurbine
Harness Name
testHarness_SingleTurbin...
Simulation Mode
[Model Settings]
Location
C:\fmacmill\Demos\Genera...
Enabled
Record coverage for syste...

What have we done so far....

- Created and imported test harnesses
- Created a test case for multiple simulations (iterations)
- Created a test case importing real-world data from Excel using root import mapping

Testing Against Requirements (Verification)

Good quality textural requirements....

#	Property	Description
1	Correct	Requirement has no errors and is not an error
2	Compliant	with one or more documented upper level requirements (operational, customer needs, etc)
3	Complete	Each requirement covers all aspects of the requirement's intent.
4	Consistent	Is not in conflict with any other requirement. Is consistent with the environment
5	Validated	Ensures the requirement will lead to the right design, i.e. reflects fully, correctly and objectively system objectives, scope, operational use, etc
6	Achievable	Can be implemented in a cost-effective manner that considers cost and schedule constraints
7	Unambiguous	The requirement has only one possible interpretation. Questions are: Could the requirement be read different ways by different people? What are the different interpretations of the requirement?
8	Verifiable	Expected performance or functionality expressed in a manner that allows verification to be objective, preferably as a result of an observable, ideally measurable, effect
9	Singular	Use a unique "shall" in each textual requirement to express a single design Demand (unique intent).
10	Positive	Negative requirements are very difficult, if not impossible, to verify. Negative requirement may be used only for safety requirements
11	Adequate	Each requirement is expressed as a problem statement i.e. it defines what is needed, not a solution, except if a particular implementation is a constraint to be resolved by design and test

This model had requirements such as...

9. Pitch Controller Requirements

Track angle within	0.2 degree
Rotor speed	Wthin 10% of nominal
Rise Time	3 seconds
Settling Time	5 seconds

These are ambiguous, incomplete, and not clearly verifiable

Hopefully a bit better is...

REQ001: when in power generation mode the rotor speed shall be maintained within ±10% of the [RotorNominalSpeed]

Definition: [RotorNominalSpeed] shall be calculated for a given turbine design to correspond to the rated generator speed converted to rotor speed based upon gearing implementation to requirements [ref. TBD].

Rationale: control average power, shed aerodynamic load

Verification: by system level simulation

REQ002: under inertial load only (zero aerodynamic load) the rise time of the blade pitch angle to a ±10° step change in pitch angle demand shall be less than 3 seconds

Definition: rise time shall be measured as the time elapsed from initiation of the step to 90% of the expected response.

Rationale: design experience [Ref. TBD] indicates meeting this requirement is a prerequisite for meeting REQ001 & braking requirements of [TBD]

Verification: by sub-system simulation

Example 1:
Using verify() to test against a requirement

Simulink Test

TESTS | DATA INSPECTOR | FORMAT

New | Open | Save | Cut | Copy | Paste | Delete | Run | Stop | Debug | Parallel | Report | Visualize | Highlight in Model | Import | Export | Preferences | Help

FILE | EDIT | RUN | RESULTS | ENVIRONMENT | RESOURCES

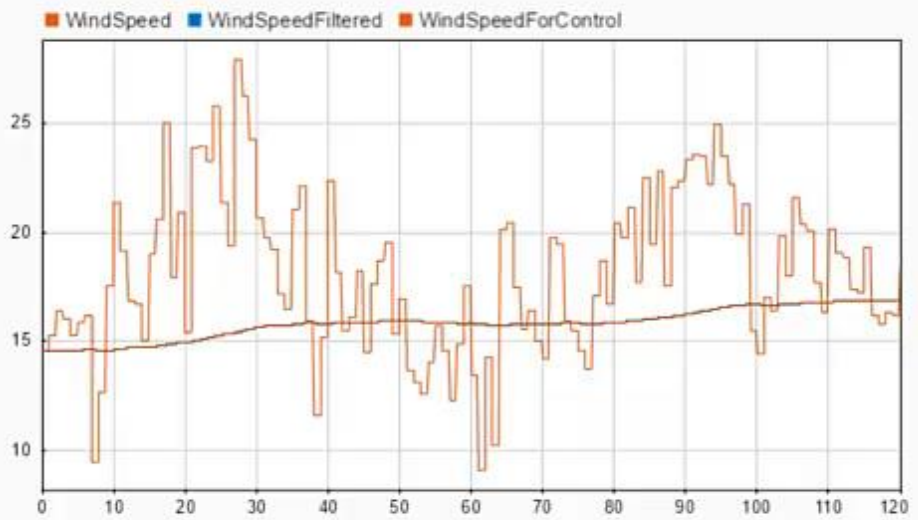
Test Browser | Results and Artifacts

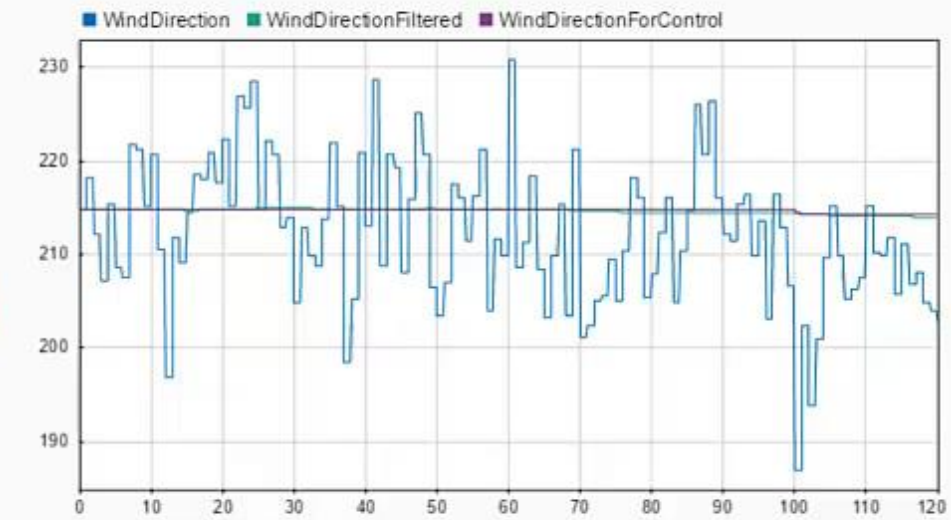
Filter tests by name or tags, e.g. tags: test

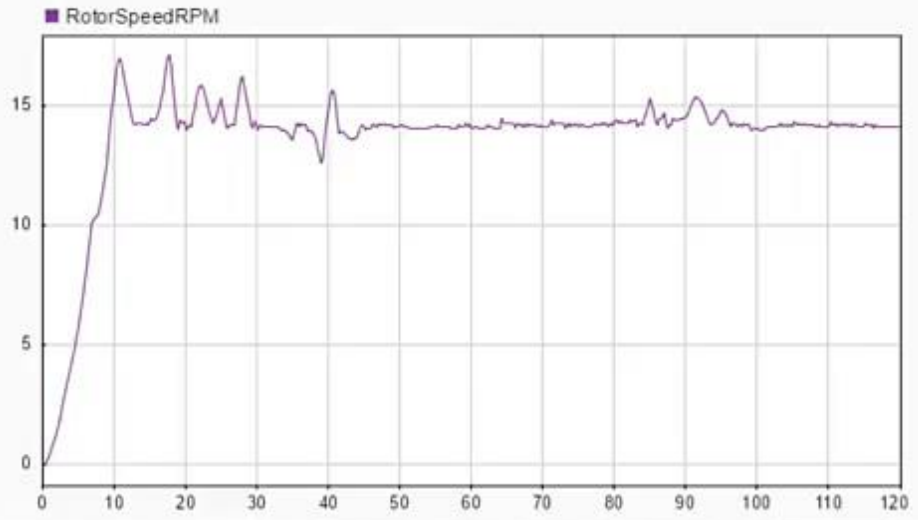
- tests_ForEXPO
  - IdealTestCases
    - OriginalTest
  - RealWorldScenarios
    - 2011_05_23

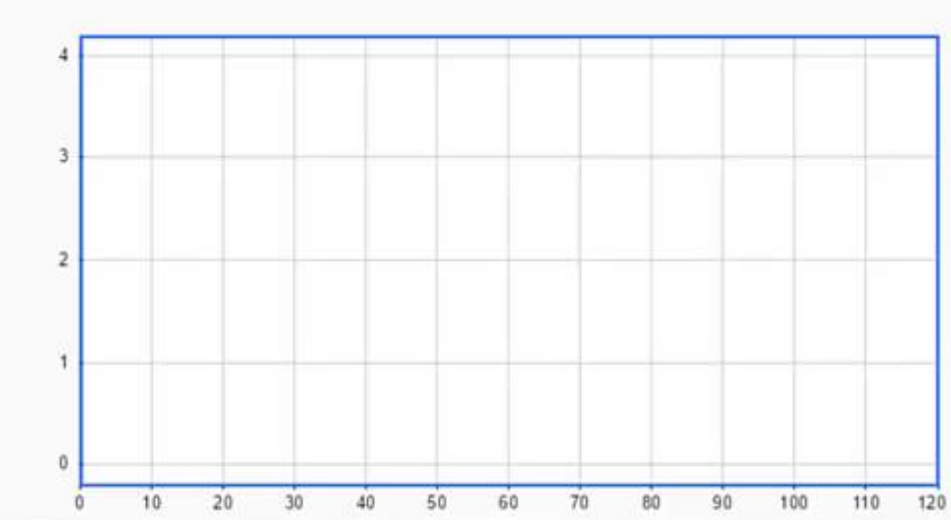
PROPERTY	VALUE
Name	tests_ForEXPO
Location	C:\fmacmill\Demos\Genera...
Enabled	✓

2011_05_23 | Start Page | Visualize









Further considerations...

- Testing at an appropriate level

  i.e. system – sub system – component

- Verification of more complex requirements

Example 2:
Using custom criteria to test against a requirement

REQ002: under inertial load only (zero aerodynamic load) the rise time of the blade pitch angle to a ±10° step change in pitch angle demand shall be less than 3 seconds

Definition: rise time shall be measured as the time elapsed from initiation of the step to 90% of the expected response.

Rationale: design experience [Ref. TBD] indicates meeting this requirement is a prerequisite for meeting REQ001 & braking requirements of [TBD]

Verification: by sub-system simulation

REQ003: under inertial load only (zero aerodynamic load) the settling time of the blade pitch angle to a ±10° step change in pitch angle demand shall be less than 5 seconds

Definition: settling time shall be measured as the time elapsed from initiation of the step to the response remaining within 5% of the expected response.

Rationale: design experience [Ref. TBD] indicates meeting this requirement is a prerequisite for meeting REQ001 & braking requirements of [TBD]

Verification: by sub-system simulation

Types of Qualifications

R2017a

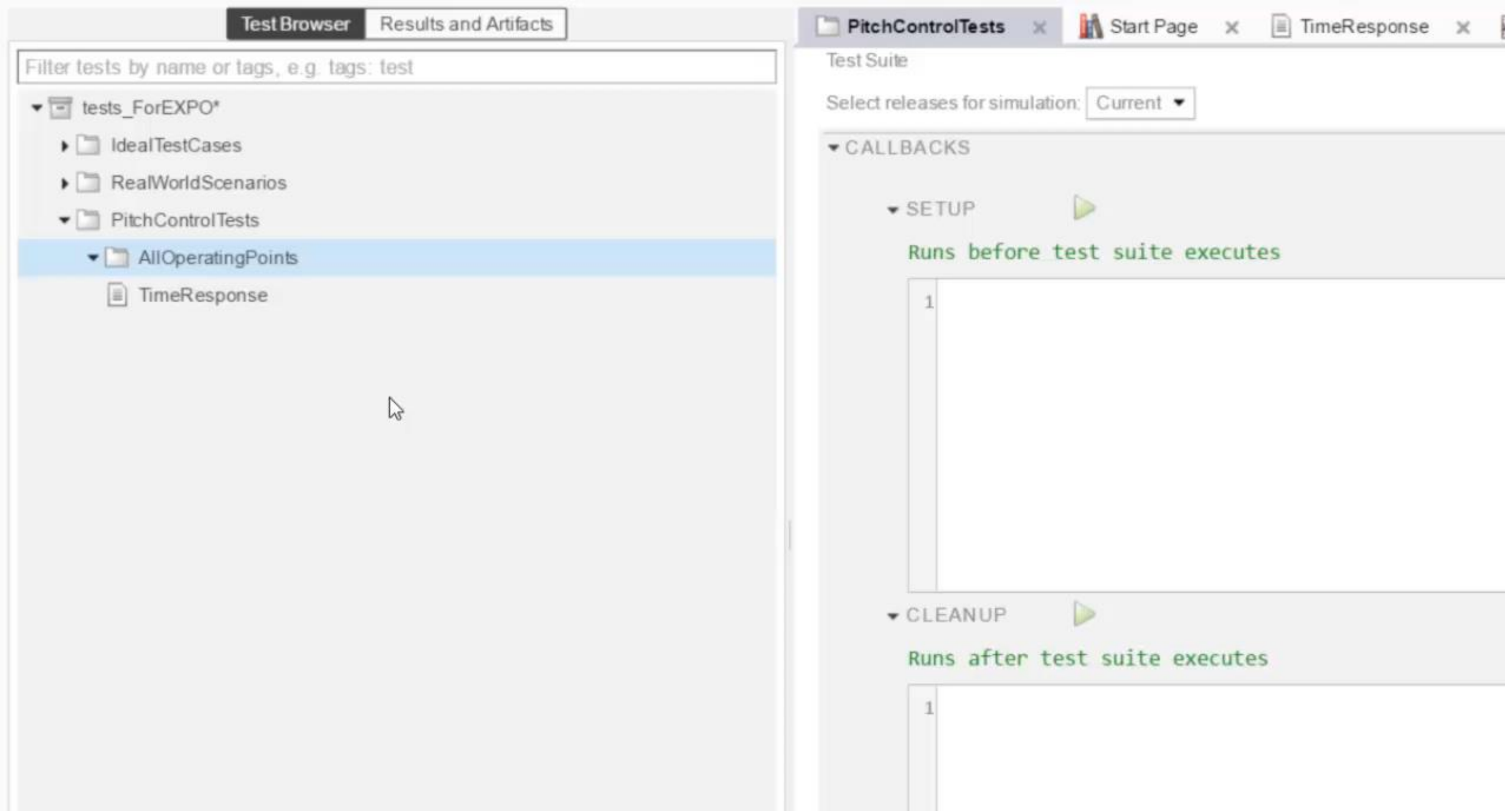
Qualifications are functions for testing values and responding to failures. There are four types of qualifications:

- Verifications — Produce and record failures without throwing an exception, meaning the remaining tests run to completion.
- Assumptions — Ensure that a test runs only when certain preconditions are satisfied and the event should not produce a test failure. When an assumption failure occurs, the testing framework marks the test as filtered.
- Assertions — Ensure that the preconditions of the current test are met.
- Fatal assertions — Use this qualification when the failure at the assertion point renders the remainder of the current test method invalid or the state is unrecoverable.

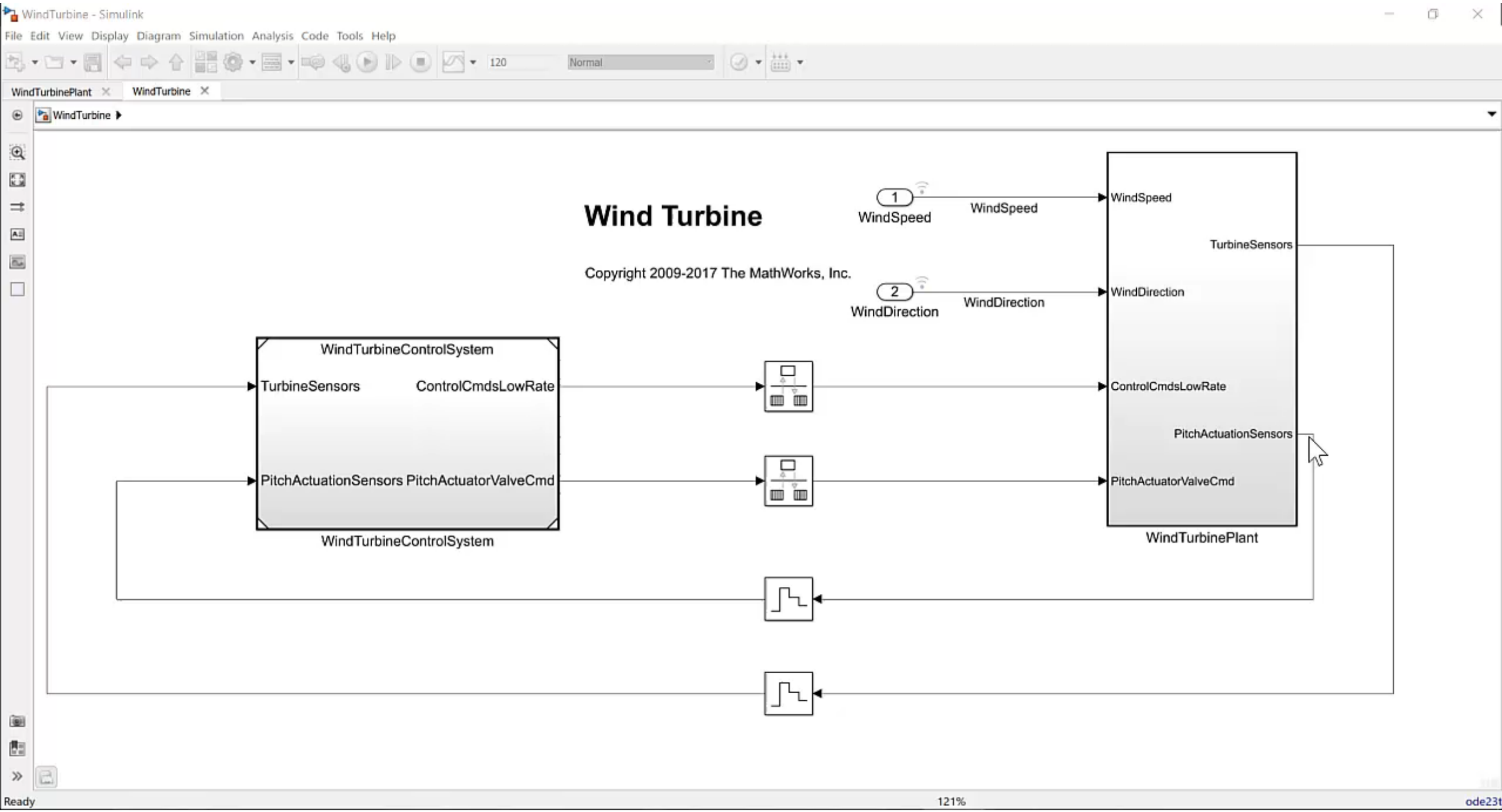
Type of Test	Verification	Assumption	Assertion	Fatal Assertion
Value is true.	`verifyTrue`	`assumeTrue`	`assertTrue`	`fatalAssertTrue`
Value is false.	`verifyFalse`	`assumeFalse`	`assertFalse`	`fatalAssertFalse`
Value is equal to specified value.	`verifyEqual`	`assumeEqual`	`assertEqual`	`fatalAssertEqual`
Value is not equal to specified value.	`verifyNotEqual`	`assumeNotEqual`	`assertNotEqual`	`fatalAssertNotEqual`
Two values are handles to same instance.	`verifySameHandle`	`assumeSameHandle`	`assertSameHandle`	`fatalAssertSameHandle`
Value is not handle to specified instance.	`verifyNotSameHandle`	`assumeNotSameHandle`	`assertNotSameHandle`	`fatalAssertNotSameHandle`
Function returns true when evaluated.	`verifyReturnsTrue`	`assumeReturnsTrue`	`assertReturnsTrue`	`fatalAssertReturnsTrue`
Test produces unconditional failure.	`verifyFail`	`assumeFail`	`assertFail`	`fatalAssertFail`
Value meets given constraint.	`verifyThat`	`assumeThat`	`assertThat`	`fatalAssertThat`
Value is greater than specified value.	`verifyGreaterThan`	`assumeGreaterThan`	`assertGreaterThan`	`fatalAssertGreaterThan`
Value is greater than or equal to specified value.	`verifyGreaterThanOrEqual`	`assumeGreaterThanOrEqual`	`assertGreaterThanOrEqual`	`fatalAssertGreaterThanOrEqual`
Value is less than specified value.	`verifyLessThan`	`assumeLessThan`	`assertLessThan`	`fatalAssertLessThan`
Value is less than or equal to specified value.	`verifyLessThanOrEqual`	`assumeLessThanOrEqual`	`assertLessThanOrEqual`	`fatalAssertLessThanOrEqual`
Value is exact specified class.	`verifyClass`	`assumeClass`	`assertClass`	`fatalAssertClass`
Value is object of specified type.	`verifyInstanceOf`	`assumeInstanceOf`	`assertInstanceOf`	`fatalAssertInstanceOf`
Value is empty.	`verifyEmpty`	`assumeEmpty`	`assertEmpty`	`fatalAssertEmpty`
Value is not empty.	`verifyNotEmpty`	`assumeNotEmpty`	`assertNotEmpty`	`fatalAssertNotEmpty`
Value has specified size.	`verifySize`	`assumeSize`	`assertSize`	`fatalAssertSize`
Value has specified length.	`verifyLength`	`assumeLength`	`assertLength`	`fatalAssertLength`
Value has specified element count.	`verifyNumElements`	`assumeNumElements`	`assertNumElements`	`fatalAssertNumElements`
String contains specified string.	`verifySubstring`	`assumeSubstring`	`assertSubstring`	`fatalAssertSubstring`
Text matches specified regular expression.	`verifyMatches`	`assumeMatches`	`assertMatches`	`fatalAssertMatches`
Function throws specified exception.	`verifyError`	`assumeError`	`assertError`	`fatalAssertError`
Function issues specified warning.	`verifyWarning`	`assumeWarning`	`assertWarning`	`fatalAssertWarning`
Function issues no warnings.	`verifyWarningFree`	`assumeWarningFree`	`assertWarningFree`	`fatalAssertWarningFree`

Test across multiple operating points?

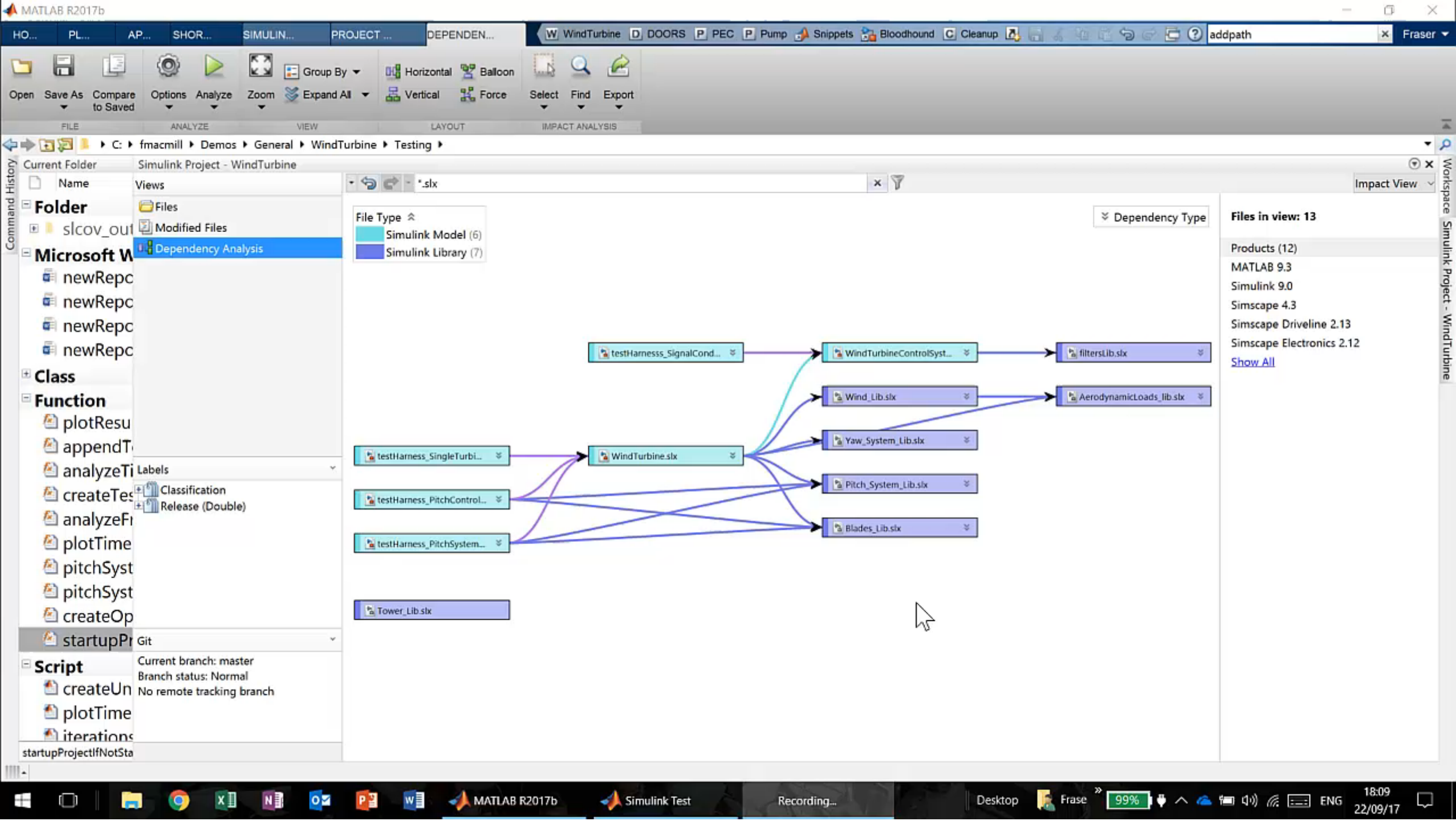
(trim conditions)


Test Browser
Results and Artifacts
Filter tests by name or tags, e.g. tags: test
tests_ForEXPO*
IdealTestCases
RealWorldScenarios
PitchControlTests
AllOperatingPoints
TimeResponse
PitchControlTests
Start Page
TimeResponse
Test Suite
Select releases for simulation:
Current
CALLBACKS
SETUP
Runs before test suite executes
1
CLEANUP
Runs after test suite executes
1

Incorporating coverage analysis


WindTurbine - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
WindTurbinePlant
WindTurbine
WindTurbine
Wind Turbine
Copyright 2009-2017 The MathWorks, Inc.
1
WindSpeed
WindSpeed
2
WindDirection
WindDirection
WindTurbineControlSystem
TurbineSensors
ControlCmdsLowRate
PitchActuationSensors
PitchActuatorValveCmd
WindTurbineControlSystem
WindSpeed
TurbineSensors
WindDirection
ControlCmdsLowRate
PitchActuationSensors
PitchActuatorValveCmd
WindTurbinePlant
Ready
121%
ode23t

Regression & cross-release testing


MATLAB R2017b
addpath
Fraser
Open
Save As
Compare to Saved
Options
Analyze
Zoom
Group By
Expand All
Horizontal
Vertical
Balloon
Force
Select
Find
Export
FILE
ANALYZE
VIEW
LAYOUT
IMPACT ANALYSIS
C: ▸ fmacmill ▸ Demos ▸ General ▸ WindTurbine ▸ Testing ▸
Current Folder
Simulink Project - WindTurbine
Name
Views
Files
Modified Files
Dependency Analysis
*.slx
Impact View
File Type
Simulink Model (6)
Simulink Library (7)
Dependency Type
Files in view: 13
Products (12)
MATLAB 9.3
Simulink 9.0
Simscape 4.3
Simscape Driveline 2.13
Simscape Electronics 2.12
Show All
testHarnesss_SignalCond...
WindTurbineControlSyst...
filtersLib.slx
Wind_Lib.slx
AerodynamicLoads_lib.slx
Yaw_System_Lib.slx
testHarness_SingleTurbi...
WindTurbine.slx
Pitch_System_Lib.slx
testHarness_PitchControl...
Blades_Lib.slx
testHarness_PitchSystem...
Tower_Lib.slx
Labels
Classification
Release (Double)
Git
Current branch: master
Branch status: Normal
No remote tracking branch
Folder
Microsoft W
Class
Function
Script
startupProjectIfNotSta
Command History
Workspace
Simulink Project - WindTurbine
Recording...
Desktop
99%
ENG
18:09
22/09/17

Continuous Integration

WindTurbine [Jenkins] | Egencia Search result pa

localhost:8080/job/WindTurbine/

Apps | Bookmarks | Google Advanced Se | Salesforce | Jenkins | WebEx | MathWorks | fmacmill - GitLab | Personal | Travel | LinkedIn | Wx | Rain Radar | xkcd | WxCh | Other bookmarks

Jenkins 2 search

Jenkins › WindTurbine ›

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Git Polling Log

Build History trend

find

#7	Sep 25, 2017 4:44 PM	
#6	Sep 25, 2017 4:05 PM	
#5	Sep 25, 2017 4:01 PM	
#4	Sep 25, 2017 2:01 PM	
#3	Sep 25, 2017 1:29 PM	

RSS for all RSS for failures

Project WindTurbine

Run tests for CI

edit description

Disable Project

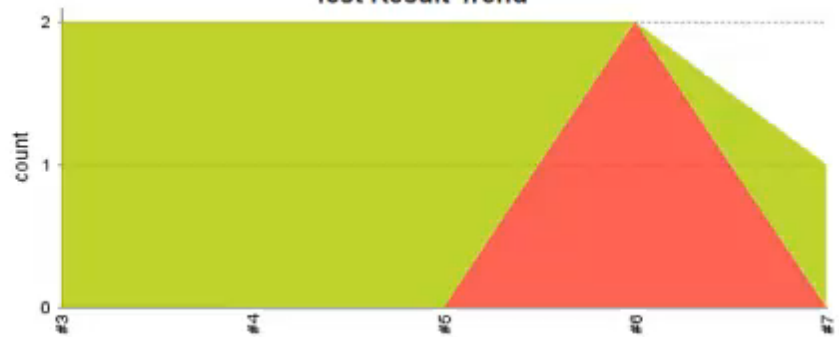
Workspace

Last Successful Artifacts
ciTestResults.pdf 98.40 KB view

Recent Changes

Latest Test Result (no failures)

Test Result Trend



(just show failures) enlarge

Permalinks

- Last build (#7), 6 min 10 sec ago
- Last stable build (#7), 6 min 10 sec ago
- Last successful build (#7), 6 min 10 sec ago
- Last unstable build (#6), 45 min ago
- Last unsuccessful build (#6), 45 min ago
- Last completed build (#7), 6 min 10 sec ago

Help us localize this page

Page generated: Sep 25, 2017 4:50:29 PM BST REST API Jenkins ver. 2.55

Conclusions

Benefits of Simulink Test

- Ease of creation, organisation & control of test harnesses
- Ease of driving your models with data from various sources
- Ease of verify() for in-harness/model verification of requirements
- Ease of test case set-up for multiple inputs, parameters, operating points, etc.
- Ease of reporting
- Ease of integration: requirements, coverage, MATLAB Unit Test, continuous integration, …

Test Manager

TESTS | VISUALIZE | FORMAT

Subplots | Clear Subplot | Legend | Data Cursors | Highlight in Model | Send to Figure

VIEWS | ZOOM & PAN | MEASURE & TRACE | SHARE

Test Browser | Results and Artifacts

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

NAME	STATUS
Results: 2016-Sep-19 14:06:01	2 1
testMotorPlant	1
Open Loop Tests	1
Thermal requirement test	
testMotorSystem	1 1
System Tests	1 1
chirpCustomCriteriaTest	
Sim Output (myMotorSystem ...)	
Custom Criteria Result	
Gain upper bound e...eedence:	
Gain lower bound ex...edence:	
Phase upper bound exc...	
Phase lower bound exceedence	
chirpTempTests Table Iterations	

chirpCustomCriteriaTest | Start P...

chirpCustomCriteriaTest

testMotorSystem » System Tests » chirpCustom...

Baseline Test

TAGS

myMotorSystem

DESCRIPTION

REQUIREMENTS

SYSTEM UNDER TEST

Model: m...

TEST HARNESS

SIMULATION SETTINGS OVERR...

PARAMETER OVERRIDES

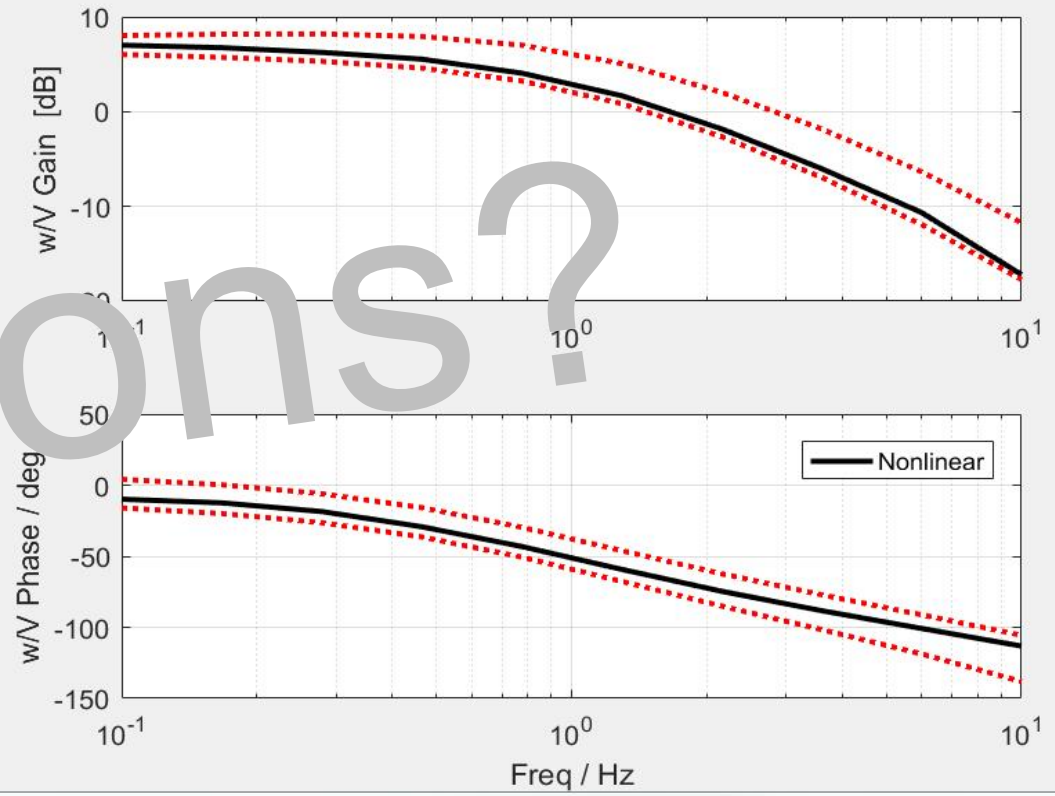
CALLBACKS

INPUTS

OUTPUTS

Figure 1: chirpFreq_

File Edit View Insert Tools Desktop Window Help



Questions?