

Verification & Validation of an Autonomous Quadcopter System

Jeremy Ross
Senior Application Engineer
November 7th, 2017



Agenda

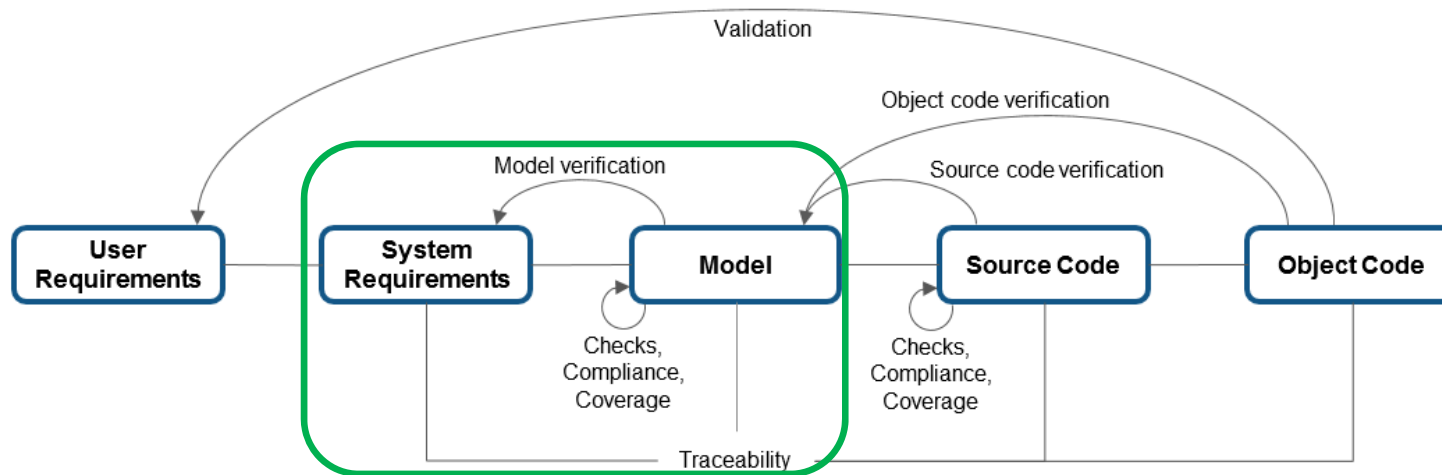
- “*Why do verification & validation at all?*”
- Our Quadcopter Story
- Implementing Requirements and Handling Changes
- Verifying Requirements Through Regression Testing
- “*When am I finally done testing!?*”
- Using Static Analysis to Complement Dynamic Testing

Why Do Verification & Validation? → Failure is Not an Option!

- Increasing product complexity
 - Manual testing takes too long and may be incomplete
- Finding defects late
 - Increased costs for rework or significant costs of recalls
- Meeting Industry or Customer's Standards
 - DO-178 (Aero), ISO 26262 (Auto), IEC 62304 (Medical), MAAB, MISRA, etc.
 - Time and cost for safety critical projects estimated 20-30 times more costly*



Where MathWorks V&V Products Fit into a Design Workflow



Simulink Requirements, Coverage and Check

- Requirement Traceability
- Model and Code Coverage
- Standards Checking and Metrics

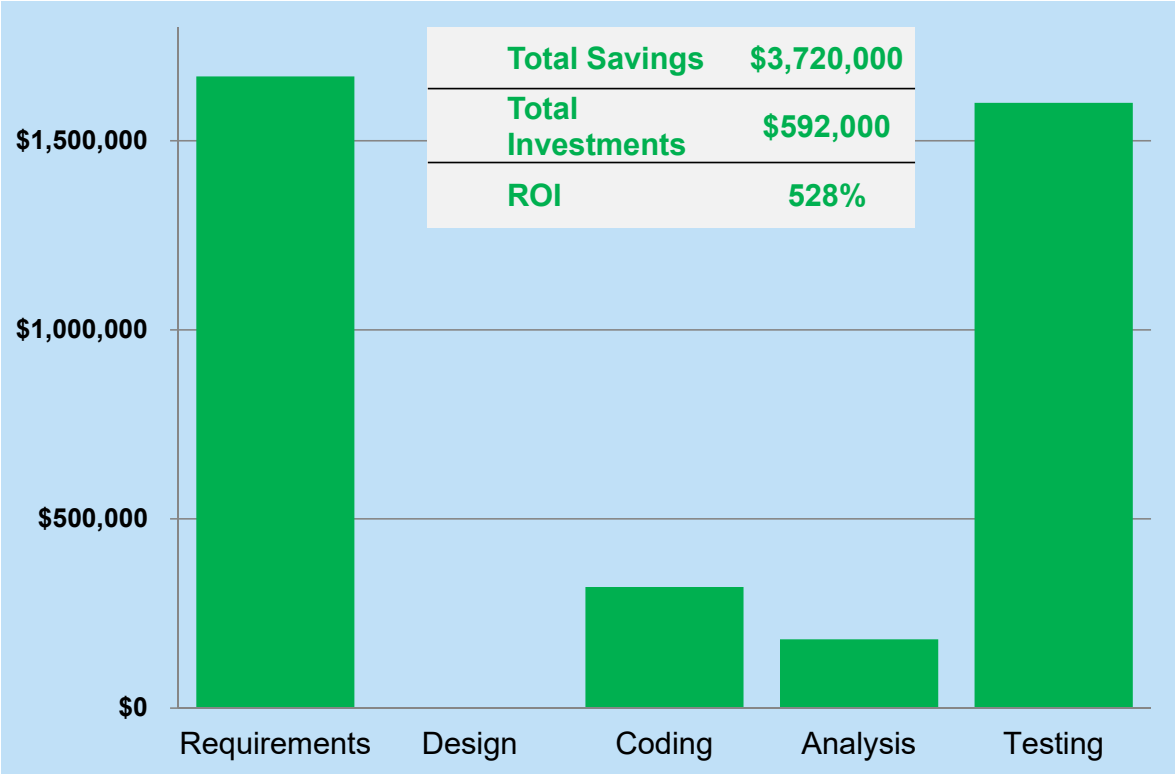
Simulink Test

- Test Harnesses
- Test Sequences
- Test Manager

Simulink Design Verifier

- Test Generation
- Design Error Detection
- Requirements Proving
- Model Slicer

Where Customers Measure the Biggest ROI with V&V Tools



Aerospace Customer Data Shared with MathWorks

Customer User Stories – Not Just Big Aerospace!



Bell Helicopter

Traceability enabled the team to perform an impact analysis to identify areas of the Simulink model that would be affected if requirements were updated later in the project

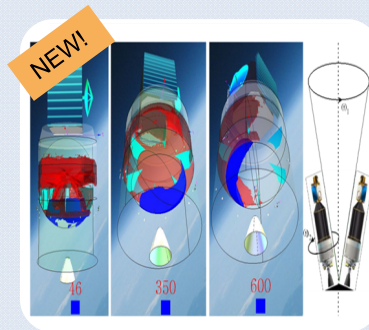
[Link to User Story](#)



Chery Automobile

Established bidirectional links between requirements and the model elements for Engine Management System software in Simulink that implemented the requirements

[Link to User Story](#)



ESA and Airbus

Linked elements of the model to system requirements. Automated documentation that incorporated the linked requirements, and the simulation results for each requirement.

[Link to User Story](#)



Baker Hughes

Checked compliance of Oil and Gas Drilling Equipment with MathWorks Automotive Advisory Board (MAAB) modeling standards and measure model coverage of their test cases

[Link to User Story](#)



ITK Engineering

Produced model coverage reports for MATLAB unit testing scripts for IEC 62304 Compliant Dental Drill Motor

[Link to User Story](#)

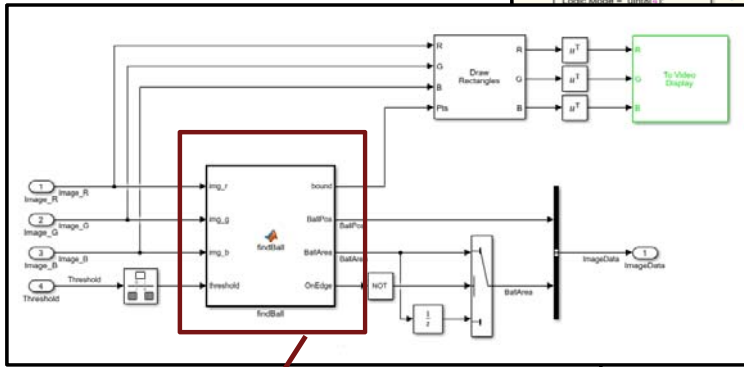
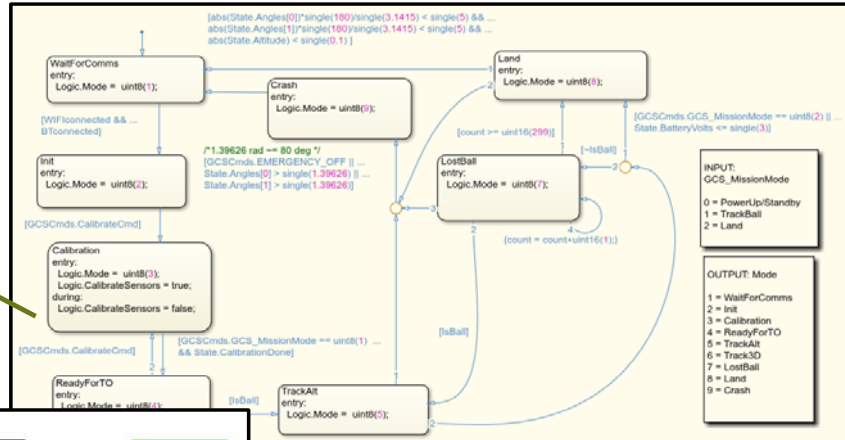
Initial Quadcopter Design Problem

- Control a quadcopter to track a ball up and down

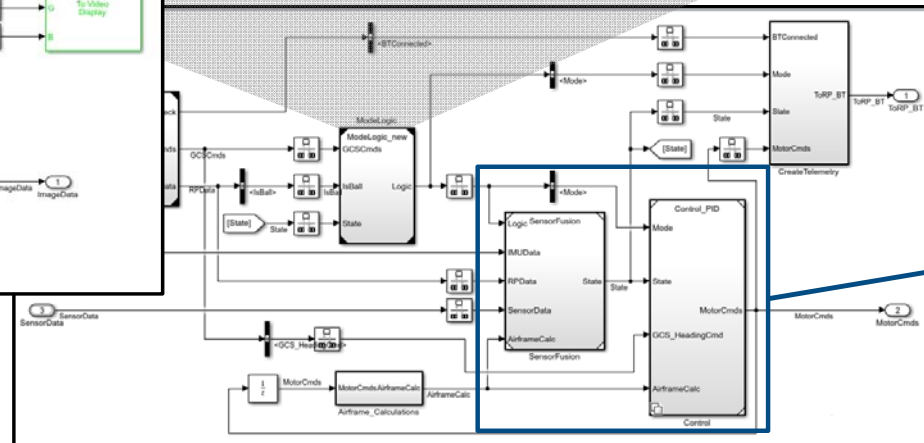


Quadcopter Design Model

Stateflow
Supervisory Logic

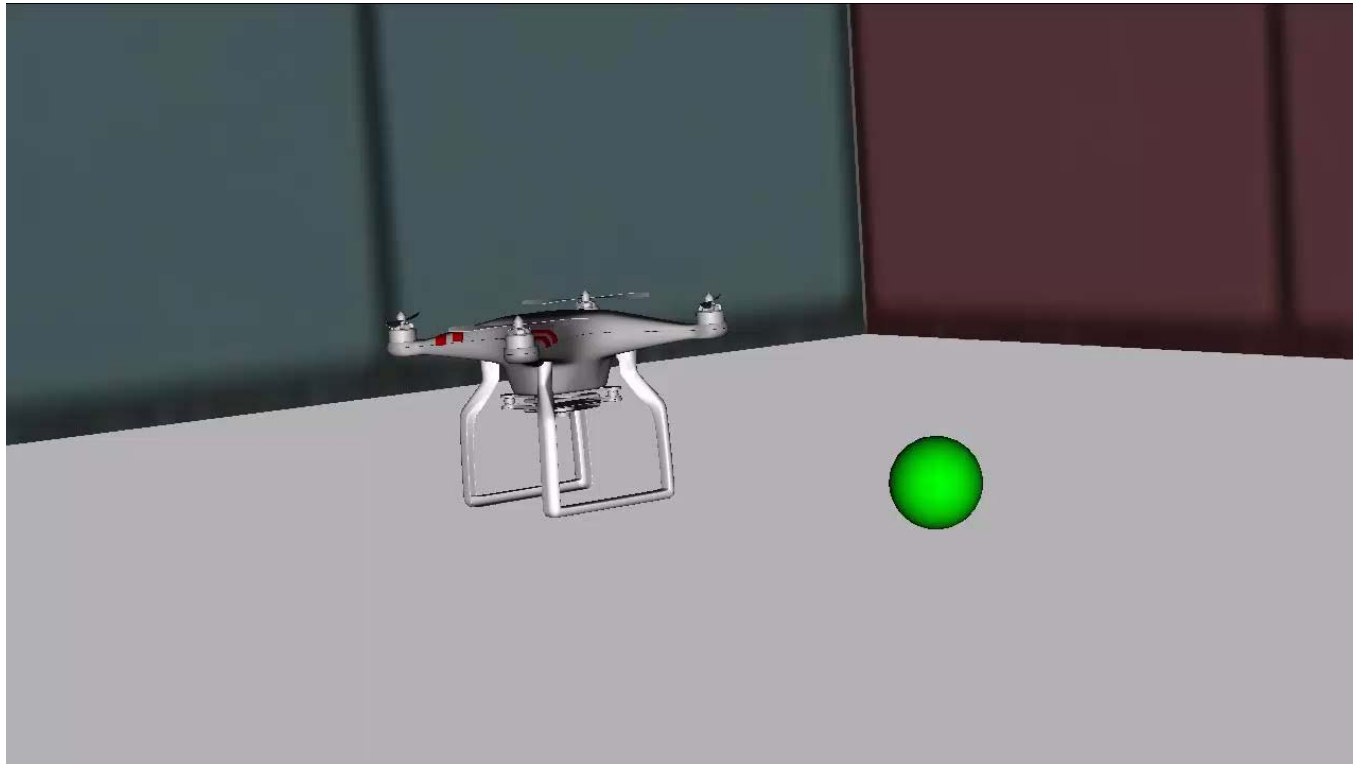


MATLAB
Image Processing and
Object Detection



Simulink
Kalman Filtering and PID
Airframe Control

Simulation Results



Modified Quadcopter Design Problem

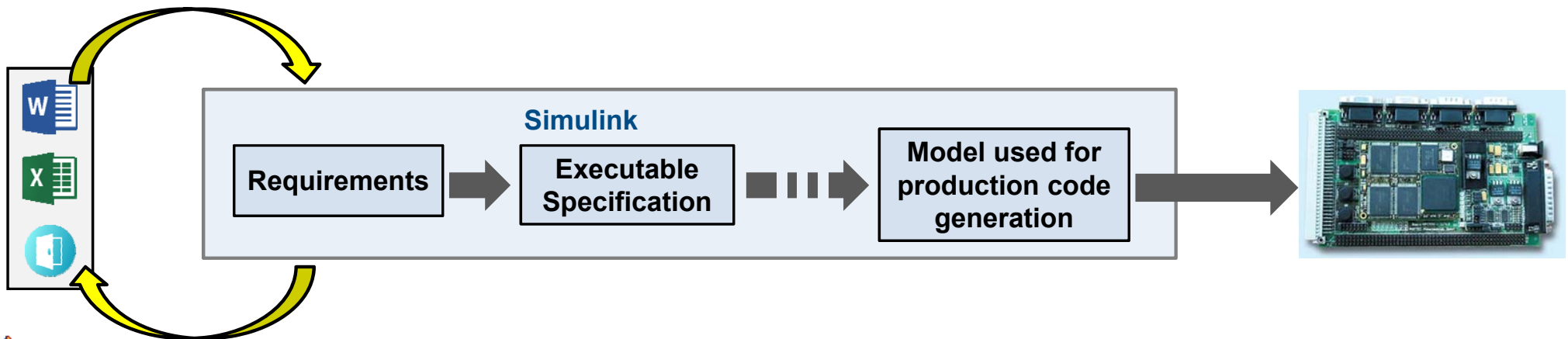
- Requirement Change: Control a quadcopter to track a ball **all around** ~~up and down~~



Simulink Requirements

Work with Requirements without Leaving Simulink!

- Author Requirements and Models Together
- View and Link Requirements within the Model
- Track Status and Quickly Manage Requirement Changes
- Trace Requirements to Models and Test Cases → Generated Code




Author or Import Requirements

The screenshot displays a Microsoft Word document titled 'aircraft_spec.docx'. The document content includes:

- 1 System Description**
 - 1.1 Overview**


The MR1 is a micro unmanned Aerial Vehicle (UAV) to demonstrate software capabilities.
 - 1.2 Physical Dimensions**
 - 1.2.1 Rotor Spacing**

Distance between rotor centers shall be 142 mm



 - 1.2.2 Rotor Diameter**

The rotors shall be 75mm in length.
 - 1.3 Hardware**
 - 1.3.1 Frame**

The UAV frame shall be a bare two-layer PCB. This will reduce the expense and weight compared to a plastic enclosure.


 - 1.3.2 Battery**

Power will be provided by a standard 650 mAh 3.7v Lithium-Polymer battery. Fully charged the battery should sustain flight for 5 minutes.



 - 1.3.3 CPU**

The onboard flight control microprocessor shall be the STM32F105T8U6 ARM® 32-bit Cortex®-M3 CPU Core

The Properties pane on the right shows details for the selected requirement:

- Properties**
 - Index: 1.1.2.1
 - Custom ID: Sys_RotSpc
 - Summary: Rotor Spacing
- Description**

1.2.1 Rotor Spacing
Distance between rotor centers shall be 142 mm


- Rationale**
- Keywords:**
- Revision information:**
 - SID: 5
 - Revision: 1
 - Refreshed on: 15-May-2017 12:22:48
- Links**

Author Requirements

- Supports Rich Text
 - Images
 - Tables
 - Bullets
 - ...

OR

Import External Docs

- Word
- Excel
- DOORS

Requirements Perspective: Combining Design and Requirements

Requirement Text on Simulink Canvas

The screenshot displays the Simulink Requirements Perspective interface. The main canvas shows a Simulink model with a requirement block labeled "#1: Driver Switch Request Handling" containing the text "Handle switch operations by the driver to determine the command for the cruise control system to operate upon." The model includes blocks for "DriverSwRequest" and "reqDrv".

The Requirements Browser at the bottom shows a table of requirements:

| Index | ID | Summary | Implemented | Verified |
|-------|-----|-----------------------------------|--|---|
| > | | | | |
| > | 1 | #1 Driver Switch Request Handling | <div style="width: 100%; height: 10px; background-color: blue;"></div> | <div style="width: 0%; height: 10px; background-color: green;"></div> |
| > | 2 | #19 Cruise Control Mode | <div style="width: 100%; height: 10px; background-color: blue;"></div> | <div style="width: 0%; height: 10px; background-color: green;"></div> |
| > | 2.1 | #20 Disable Cruise Control system | <div style="width: 100%; height: 10px; background-color: blue;"></div> | <div style="width: 0%; height: 10px; background-color: green;"></div> |
| > | 2.2 | #24 Operation mode determination | <div style="width: 100%; height: 10px; background-color: blue;"></div> | <div style="width: 0%; height: 10px; background-color: green;"></div> |

The Property Inspector on the right shows details for Requirement #1, including a description, rationale, and a list of links: "Implemented by: DriverSwRequest" and "Verified by: DriverSwRequest_Tests".

Property Inspector to Modify Requirement and View Links

Enter and Exit Requirements Perspective

Navigate to associated links (design and test cases)

Requirements Browser with Implementation and Verification Status

Track Requirements' Implementation and Verification Status

Requirements Editor

View: Requirements

| Index | Summary | Implemented | Verified |
|-------|-------------------------------|-------------|----------|
| 1 | Modes of Operation | Blue | Green |
| 2 | Establish Communications | Blue | Green |
| 2.1 | Crashed or landed time | Blue | Green |
| 3 | Initialization | Blue | Green |
| 3.1 | Turn off motors | Blue | Green |
| 4 | Calibrate the Sensors | Blue | Red |
| 5 | Ready for Flight | Blue | Green |
| 6 | Track Altitude | Blue | Green |
| 7 | Lost Ball | Blue | Green |
| | Accepted modes for entering | Blue | Green |
| 7.2 | Transition to Land mode af... | Blue | Green |
| 8 | Landing | Blue | Green |
| 9 | Landing condition | Blue | Green |
| 9.1 | Landing condition | Blue | Green |
| 10 | Crash | Blue | Green |

Properties

Index: 4
Custom ID: #7
Summary: Calibrate the Sensors

Description: The Calibrate Sensors mode shall be entered only from the Initialization mode or the Ready for Flight mode when commanded by the ground station. In this mode the quadcopter shall calibrate the attitude and barometer sensors.

Links

- Implemented by: Calibration
- Verified by: testCalibrationLightOn, testCalibrationLightOff

Implemented by

Verified by

Verified by:
testCalibrationLightOn
testCalibrationLightOff

Test Manager

TESTS

DetectGreenPixels

Test results

| Test Name | Status |
|-----------------------------|--------|
| SectionTest | 6 of 1 |
| Noise reduction unit tests | 2 of 2 |
| Noise reduction test 1 | 1 of 1 |
| Noise reduction test 2 | 1 of 1 |
| Pixel detection tests | 2 of 2 |
| Screen pixel identification | 1 of 1 |
| Threshold test 1 | 1 of 1 |

Implemented by:
Calibration



Respond to Changes

Identifying Modified Requirements Quickly

- Control a quadcopter to track a ball up and down all around.

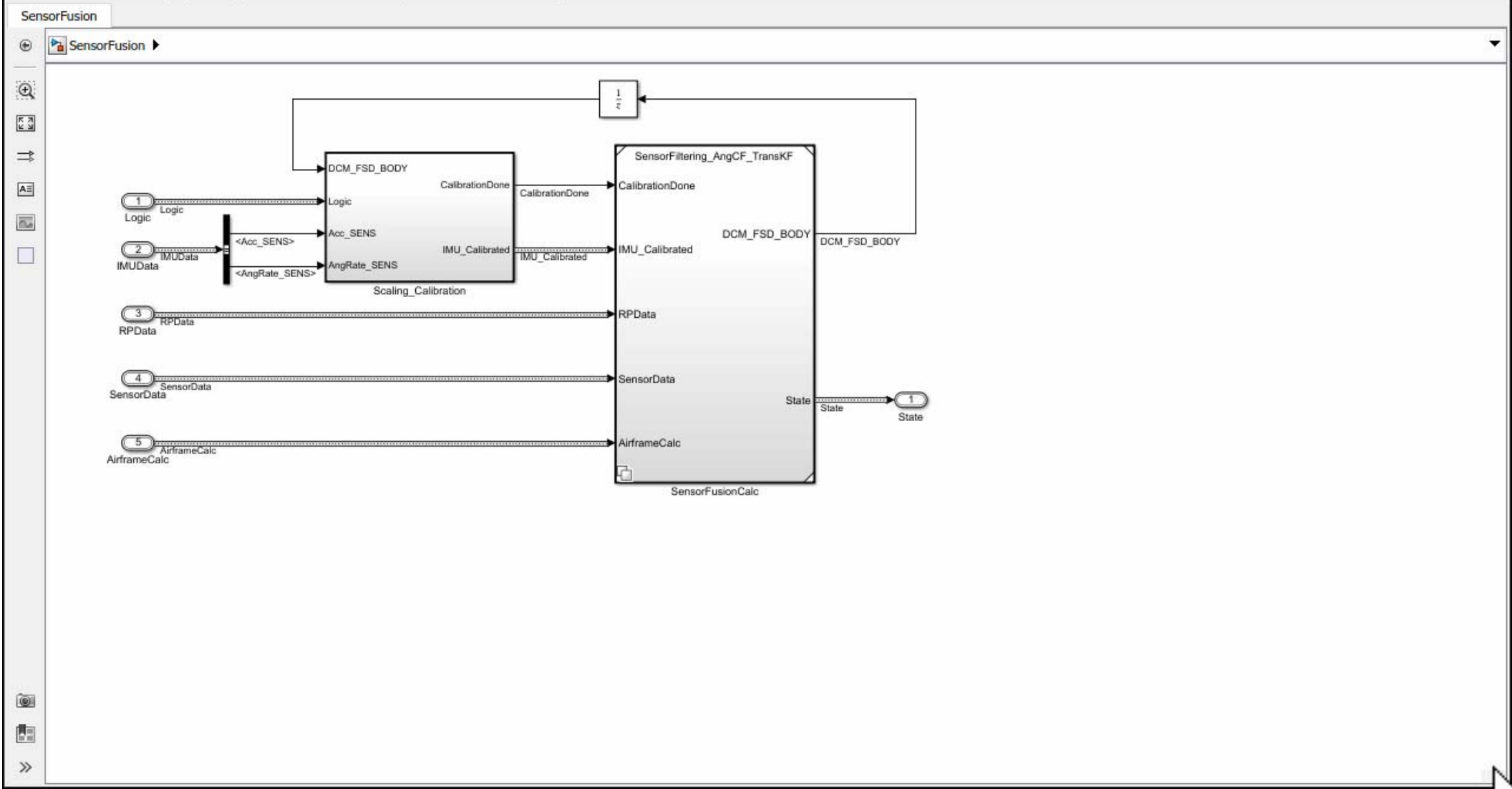
| Index | Summary | Implemented |
|---------------|----------------------------------|-------------|
| aircraft_spec | | |
| 1 | References to aircraft_spec.docx | |
| 1.1 | System Description | |
| 1.2 | General Characteristics | |
| 1.3 | Performance Requirements | |
| 1.4 | Systems Requirements | |
| 1.5 | Flight Control Requirements | |
| 1.5.1 | Mode Logic Flight Mode | |
| 1.5.1.1 | Wait for Communications | |
| 1.5.1.2 | Initializatic | |
| 1.5.1.3 | Calibration | |
| 1.5.1.4 | Ready for Takeoff | |
| 1.5.1.5 | Track Altitu | |
| 1.5.1.6 | Track 3D | |
| 1.5.1.7 | Land | |
| 1.5.1.8 | Crash | |

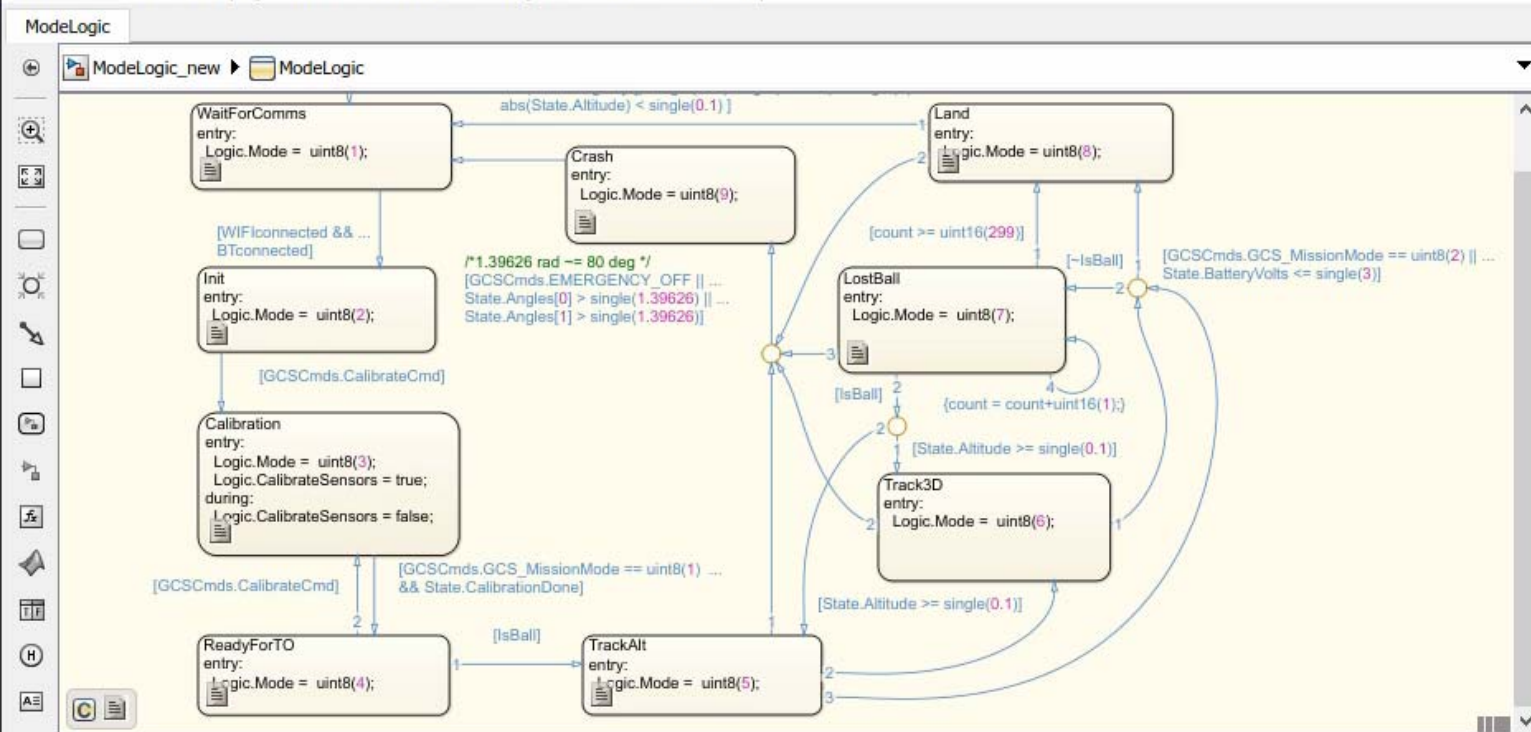


Updated Requirement



New Requirement





Property Inspector

ModeLogic

Properties Info

Update method: Inherited

Sample Time: 0.01

Create output for monitoring

Advanced

Requirements - ModeLogic_new

View: Requirements

| Index | Summary | Implemented |
|---------------|----------------------------------|-------------------------------------|
| aircraft_spec | — | <input checked="" type="checkbox"/> |
| 1 | References to aircraft_spec.docx | <input checked="" type="checkbox"/> |
| 1.1 | System Description | <input checked="" type="checkbox"/> |
| 1.2 | General Characteristics | <input checked="" type="checkbox"/> |
| 1.3 | Performance Requirements | <input checked="" type="checkbox"/> |
| 1.4 | Systems Requirements | <input checked="" type="checkbox"/> |

Respond to Changes

Qualifying with Regression Tests

The screenshot shows the Requirements Editor interface. On the left, a tree view shows a hierarchy of requirements under 'aircraft_spec'. The main table displays the following data:

| Index | Summary | Implemented | Verified |
|---------------|----------------------------------|-------------|-------------|
| aircraft_spec | | [Blue bar] | [Green bar] |
| 1 | References to aircraft_spec.docx | [Blue bar] | [Green bar] |
| 1.1 | System Description | [Blue bar] | [Green bar] |
| 1.2 | General Characteristics | [Blue bar] | [Green bar] |
| 1.3 | Performance Requirements | [Blue bar] | [Green bar] |
| 1.4 | Systems Requirements | [Blue bar] | [Green bar] |
| 1.5 | Flight Control Requirements | [Blue bar] | [Green bar] |
| 1.5.1 | Mode Logic Flight Mode | [Blue bar] | [Green bar] |
| 1.5.1.1 | Wait for Communications | [Blue bar] | [Green bar] |
| 1.5.1.2 | Initialization | [Blue bar] | [Green bar] |
| 1.5.1.5 | Track Altitude | [Blue bar] | [Red bar] |
| 1.5.1.6 | Track 3D | [Blue bar] | [Green bar] |
| 1.5.1.7 | Land | [Blue bar] | [Green bar] |
| 1.5.1.8 | Crash | [Blue bar] | [Green bar] |

The 'Properties' pane on the right shows details for requirement 1.5.1.2:

- Index: 1.5.1.2
- Custom ID: FCS_Init
- Summary: Initialization
- Description: 5.2.2 Initialization. The air vehicle shall remain in an initialization state until commanded to commence sensor calibration.

Two red boxes highlight the 'Verified' column for requirements 1.5.1.5 and 1.5.1.6. The box for 1.5.1.5 is labeled 'Test Failed' and the box for 1.5.1.6 is labeled 'Not Verified'.

Test Failed

Not Verified

New Requirements → New Test Cases

The image shows two overlapping windows from the MathWorks environment. The top window is the 'Requirements Editor' and the bottom window is the 'Simulink Test' interface.

Requirements Editor:

- View: Requirements
- Search: Track 3D
- Table with columns: Index, Summary, Implemented, Verified

| Index | Summary | Implemented | Verified |
|---------------|----------------------------------|-------------|----------|
| aircraft_spec | | | |
| 1 | References to aircraft_spec.docx | | |
| 1.5 | Flight Control Requirements | | |
| 1.5.1 | Mode Logic Flight Mode | | |
| 1.5.1.6 | Track 3D | | |

Simulink Test:

- TESTS toolbar with icons for New, Open, Save, Delete, Run, Stop, Debug, Parallel, Report, Visualize, Highlight in Model, Preferences, Help.
- Test Browser: Filter tests by name or tags, e.g. tag. Tree view showing quadCopterPositionTests > 3DTracking.
- Property Table:

| PROPERTY | VALUE |
|-----------|------------------|
| Name | 3DTracking |
| Location | S:\MAB2017Qu... |
| Hierarchy | quadCopterPos... |

The '3DTracking' test suite is shown as 'Enabled' with a dropdown for 'Select releases for simulation: R2017bPrerelease'. It includes sections for TAGS, DESCRIPTION, REQUIREMENTS, CALLBACKS, and COVERAGE SETTINGS*.

A red arrow points from the 'Track 3D' requirement in the Requirements Editor to the '3DTracking' test suite in the Simulink Test interface, illustrating the mapping from requirements to test cases.

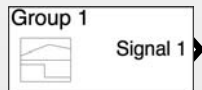
Testing in Simulink Test

Test Case

Inputs



MAT file (input)



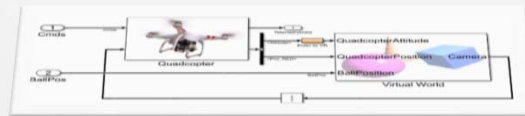
Signal Builder



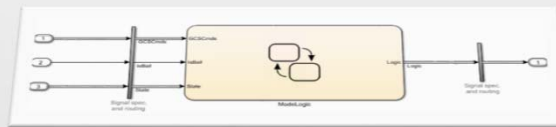
Test Sequence

and more!

Test Model



System-level test harness



Unit-level test harness

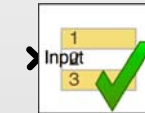
Assessments



MAT file (baseline)

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

MATLAB Unit Test



Test Assessment

and more!



Excel file (input)

R2017b



Excel file (baseline)

R2017b

Running Tests with Simulink Test

The screenshot displays the Simulink Test environment. The main window is titled 'Simulink Test' and features a toolbar with icons for New, Open, Save, Cut, Copy, Paste, Delete, Run, Stop, Debug, Parallel, Report, Visualize, Highlight in Model, Import, Export, Preferences, and Help. Below the toolbar, there are tabs for 'Test Browser' and 'Results and Artifacts'. The 'Test Browser' shows a tree view of test results for '2017-May-17 11:20:50', including '3DTracking' and 'Excel Test Case'. The 'Excel Test Case' is highlighted, and its status is '3' with a green checkmark. A red box highlights this status, and a red arrow points from it to the 'Requirements Editor' window.

The 'Results and Artifacts' window shows a 'SUMMARY' section with the following data:

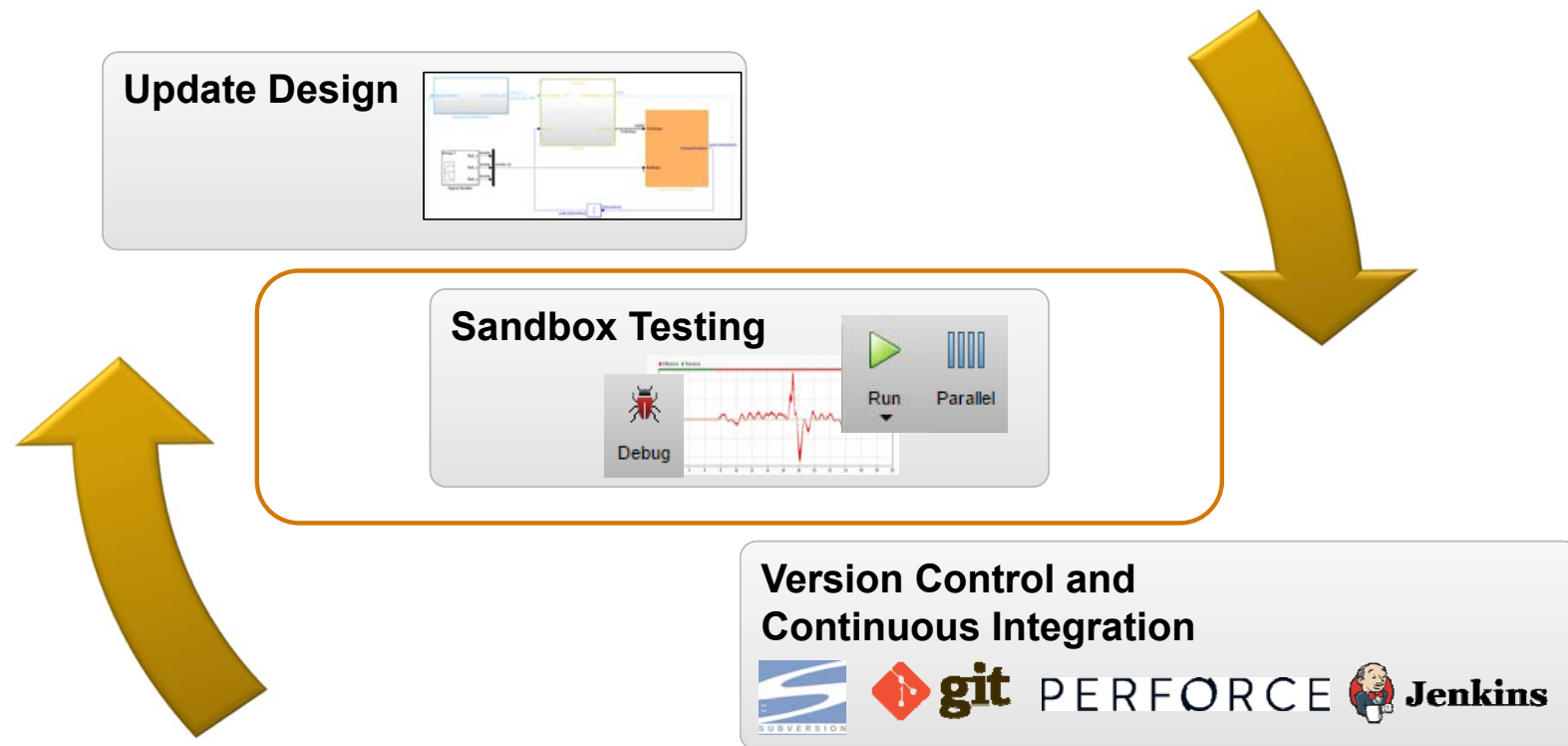
| Property | Value |
|------------|---------------------|
| Name | Excel Test Case |
| Outcome | 3 |
| Start Time | 05/17/2017 11:20:50 |
| End Time | 05/17/2017 11:22:52 |
| Type | Baseline Test |

The 'Requirements Editor' window is open, showing a table of requirements for 'Track 3D':

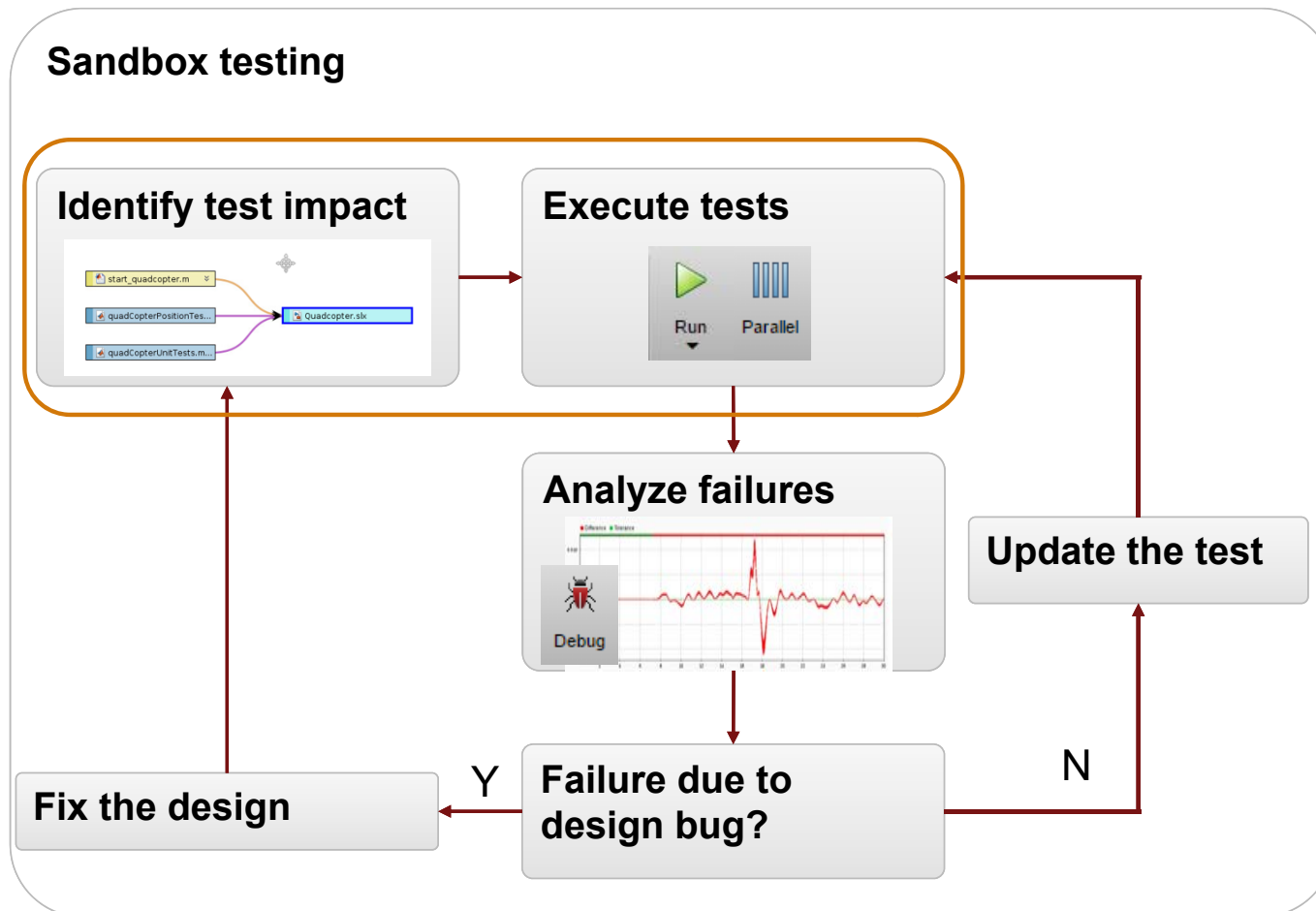
| Index | Summary | Implemented | Verified |
|---------------|----------------------------------|-------------|----------|
| aircraft_spec | | | |
| 1 | References to aircraft_spec.docx | | |
| 1.5 | Flight Control Requirements | | |
| 1.5.1 | Mode Logic Flight Mode | | |
| | Track 3D | | |

A red box highlights the 'Verified' column for the 'Track 3D' requirement, which contains a green checkmark. A green checkmark icon is also visible in the bottom right corner of the Requirements Editor window.

Regression Testing Process



Testing in a Sandbox



Simulink Project - quadcopter_reference

SIMULINK PROJECT | PROJECT SHORTCUTS | DEPENDENCY ANALYSIS

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

Views: Impacted by 'Quadcopter.slreqx'

File Type

- Script (1)
- Simulink Model (1)
- Requirements data file (1)
- SimulinkTest File (2)

Dependency Type

Files in view: 5

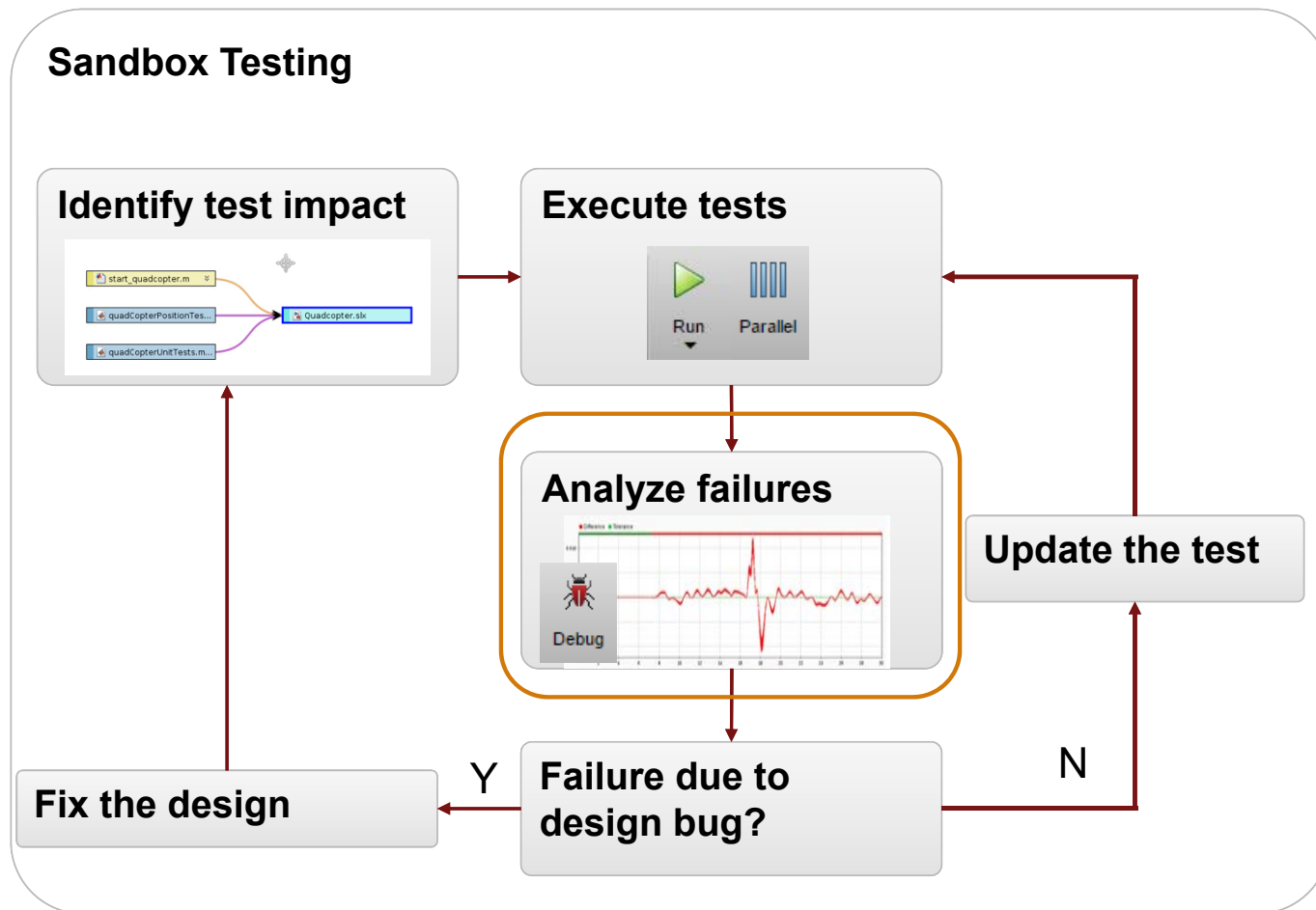
- Products (9)
- MATLAB 9.3
- Simulink 9.0
- Aerospace Blockset 3.20
- Embedded Coder 7.0
- Simulink 3D Animation 7.8
- Show All
- Problems (2)
- Requires Simulink Requirements ...

```

    graph LR
      A[Quadcopter.slreqx] --> D[Quadcopter.slx]
      B[start_quadcopter.m] --> D
      C[quadCopterPositionTests.m] --> D
      E[quadCopterUnitTests.mldatx] --> D
  
```

Labels: Classification

Testing in a Sandbox



Simulink Test

TESTS

FILE EDIT RUN RESULTS ENVIRONMENT RESOURCES

Start Page x

Getting Started

New Test File Open Test File

RECENT FILES

- [quadCopterUnitTests](#)
- [quadCopterPositionTests](#)
- [f1](#)
- [tempGenReport](#)
- [sltestProjectorFanSpeedTestSuite](#)
- [tsltestProjectorFanSpeedExample](#)
- [tsltestProjectorFanSpeedExample](#)
- [tsltestProjectorFanSpeedExample](#)
- [f1](#)
- [test](#)

HELP LINKS

- [Get Started With the Test Manager](#)
- [How to Create and Run a Test Case](#)
- [View Test Results](#)
- [Export Test Results and Generate Reports](#)

Test Browser Results and Artifacts

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

| NAME | STATUS |
|-------------------------------|--------|
| Results: 2017-May-12 14:55:05 | 20 1 |
| quadCopterPositionTests | 15 1 |
| AltitudeTracking | 5 1 |
| scenario1 | |
| scenario2 | |
| scenario3 | |
| scenario4 | |
| scenario5 | |
| scenario6 | |
| Telemetry | 5 |
| Vehicle | 5 |

PROPERTY VALUE

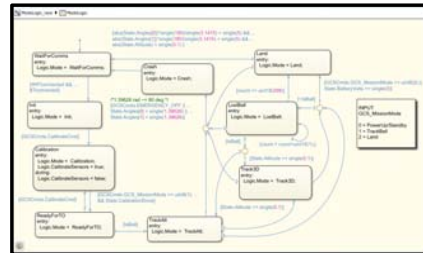
Requirements are Fully Implemented and Verified

| Index | Summary | Implemented | Verified |
|-----------------|----------------------------------|-------------|----------|
| ▼ aircraft_spec | | | |
| ▼ 1 | References to aircraft_spec.docx | | |
| > 1.1 | System Description | | |
| > 1.2 | General Characteristics | | |
| > 1.3 | Performance Requirements | | |
| > 1.4 | Systems Requirements | | |
| ▼ 1.5 | Flight Control Requirements | | |
| ▼ 1.5.1 | Mode Logic Flight Mode | | |
| 1.5.1.1 | Wait for Communications | | |
| 1.5.1.2 | Initialization | | |
| 1.5.1.3 | Calibration | | |
| 1.5.1.4 | Ready for Takeoff a | | |
| 1.5.1.5 | Track Altitude | | |
| 1.5.1.6 | Track 3D | | |
| 1.5.1.7 | Land | | |
| 1.5.1.8 | Crash | | |

“But, how do I know when we’ve done enough testing?”

Model Coverage in Dynamic Testing

Stateflow Logic



Address Missing Coverage using Static Analysis

Simulink Coverage Analysis

Results: 2017-May-10 15:08:09 x

SUMMARY

AGGREGATED COVERAGE RESULTS

| ANALYZED MODEL | REPORT | CO... | DECISION | CONDITION |
|----------------|--------|-------|----------|-----------|
| Comms | 0 | -- | 83% | 83% |
| Control_PID | 38 | 81% | 94% | 94% |
| FCS | 9 | -- | -- | -- |
| ModelLogic_new | 30 | 49% | 46% | 46% |
| Motors | 0 | -- | -- | -- |
| PiZero | 20 | 89% | 70% | 70% |
| PlantModel | 12 | 100% | 75% | 75% |

+ Add Test

Simulink Design Verifier

Simulink Design Verifier Results Summary: ModelLogic_new

Progress

Objectives processed 54/55
Satisfied 54
Unsatisfiable 0
Elapsed Time: 0:53

19-Apr-2017 13:36:48
Checking compatibility for test generation: model 'ModelLogic_new'
Compiling model...done
Checking compatibility...done

19-Apr-2017 13:37:00
'ModelLogic_new' is **compatible** for test generation with Simulink Design Verifier.

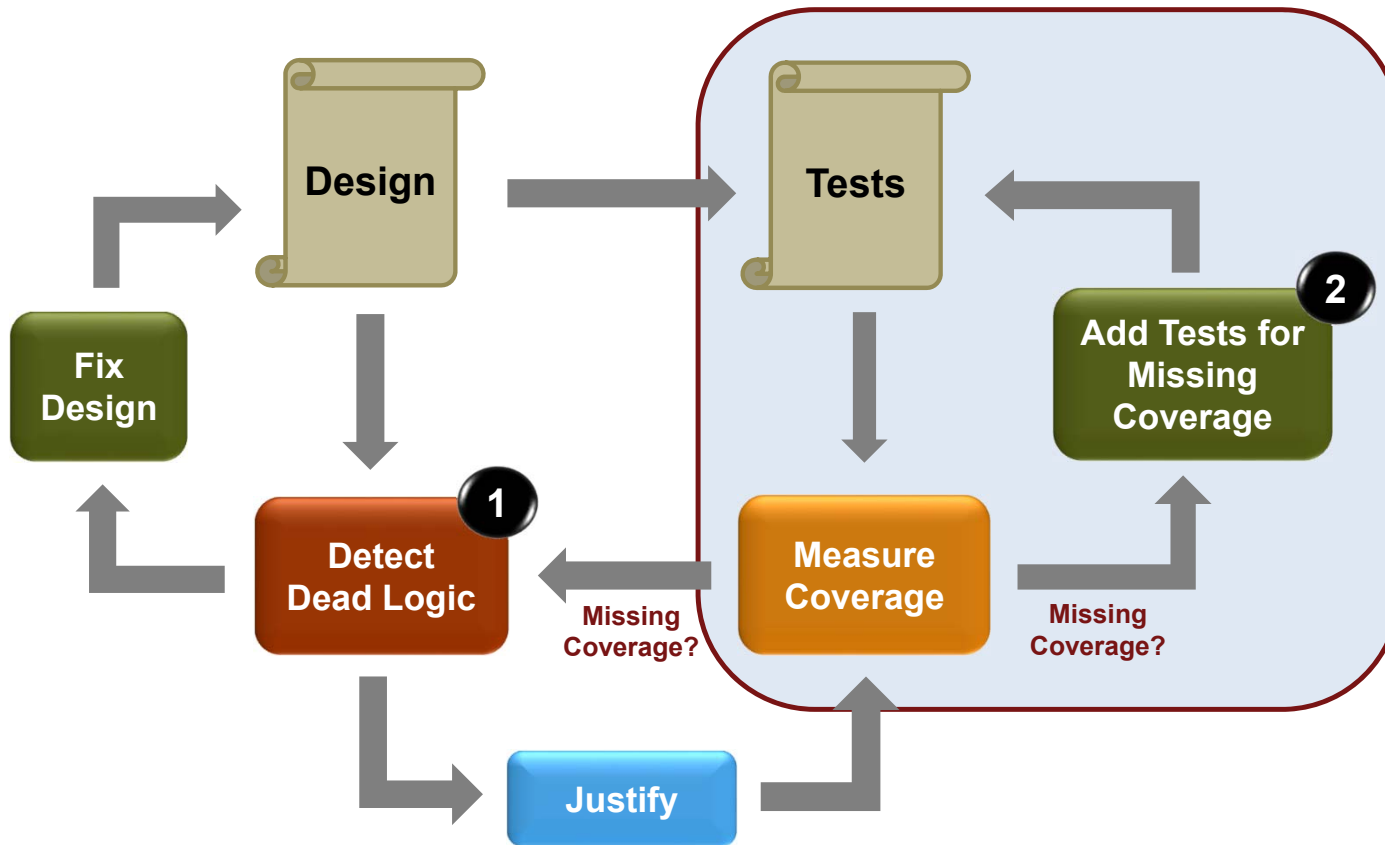
Generating tests using compatibility results from 19-Apr-2017 13:37:00...

SATISFIED
ModelLogic
Chart: Substate executed State "WaitForComms"
Analysis Time = 00:00:08

SATISFIED
ModelLogic: "[CommsCheck, GoodComms]"

Highlight Stop

Addressing Missing Coverage Workflow



Simulink Design Verifier:

- 1 Dead Logic Detection
- 2 Test Generation

Generate Tests for Missing Coverage

Simulink Test

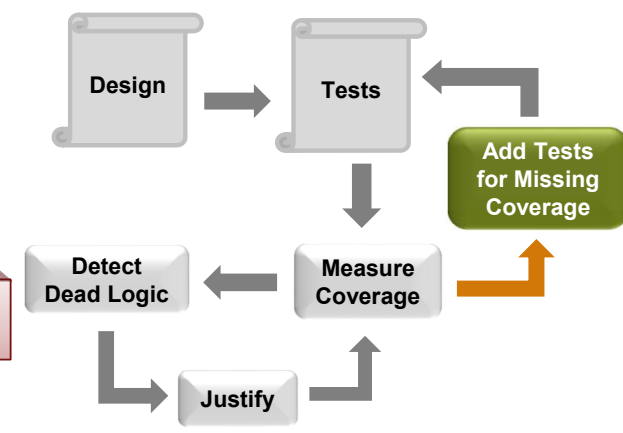
TESTS

Filter results by name or tags

| ANALYZED MODEL | REPORT | CO... | DECISION | CONDITION | EXECUTION |
|----------------------|-----------|------------|------------|-----------|-----------|
| Comms | 0 | -- | 83% | 100% | |
| Control_PID | 38 | 81% | 94% | 98% | |
| FCS | 9 | -- | -- | 100% | |
| ModeLogic_new | 30 | 49% | 46% | -- | |
| Motors | 0 | -- | -- | 100% | |
| PIZero | 20 | 89% | 70% | 100% | |
| PlantModel | 12 | 100% | -- | -- | |

+ Add Tests for Missing Coverage

+ Add Tests for Missing Coverage



Generate Tests for Missing Coverage

Coverage: ModeLogic_new

Transition "[count >= uint16(299)]" from "LostBall" to "Land"

Transition was never **evaluated**.

Test Generation

Results: ModeLogic_new

Back to summary

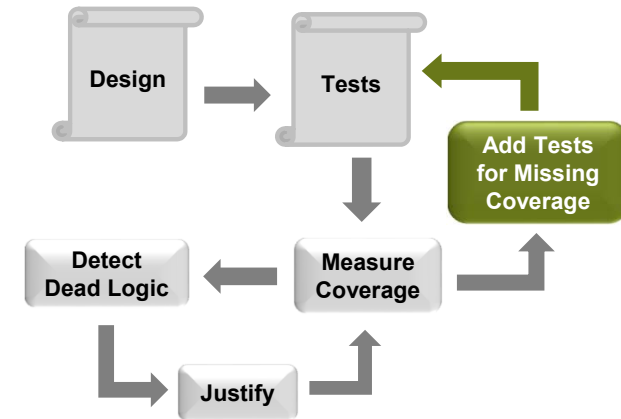
ModeLogic."[count >= uint16(299)]"

Transition: Transition trigger expression **SATISFIED** - View test case

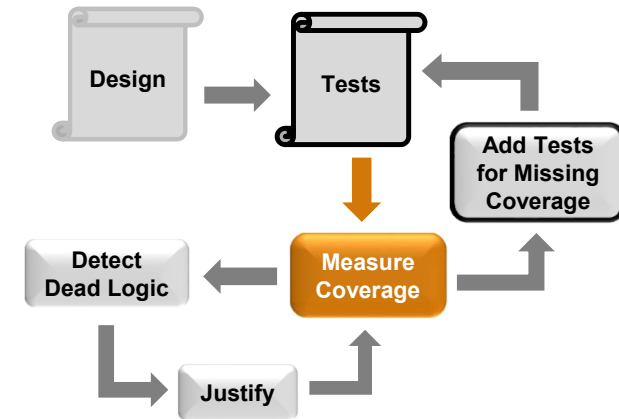
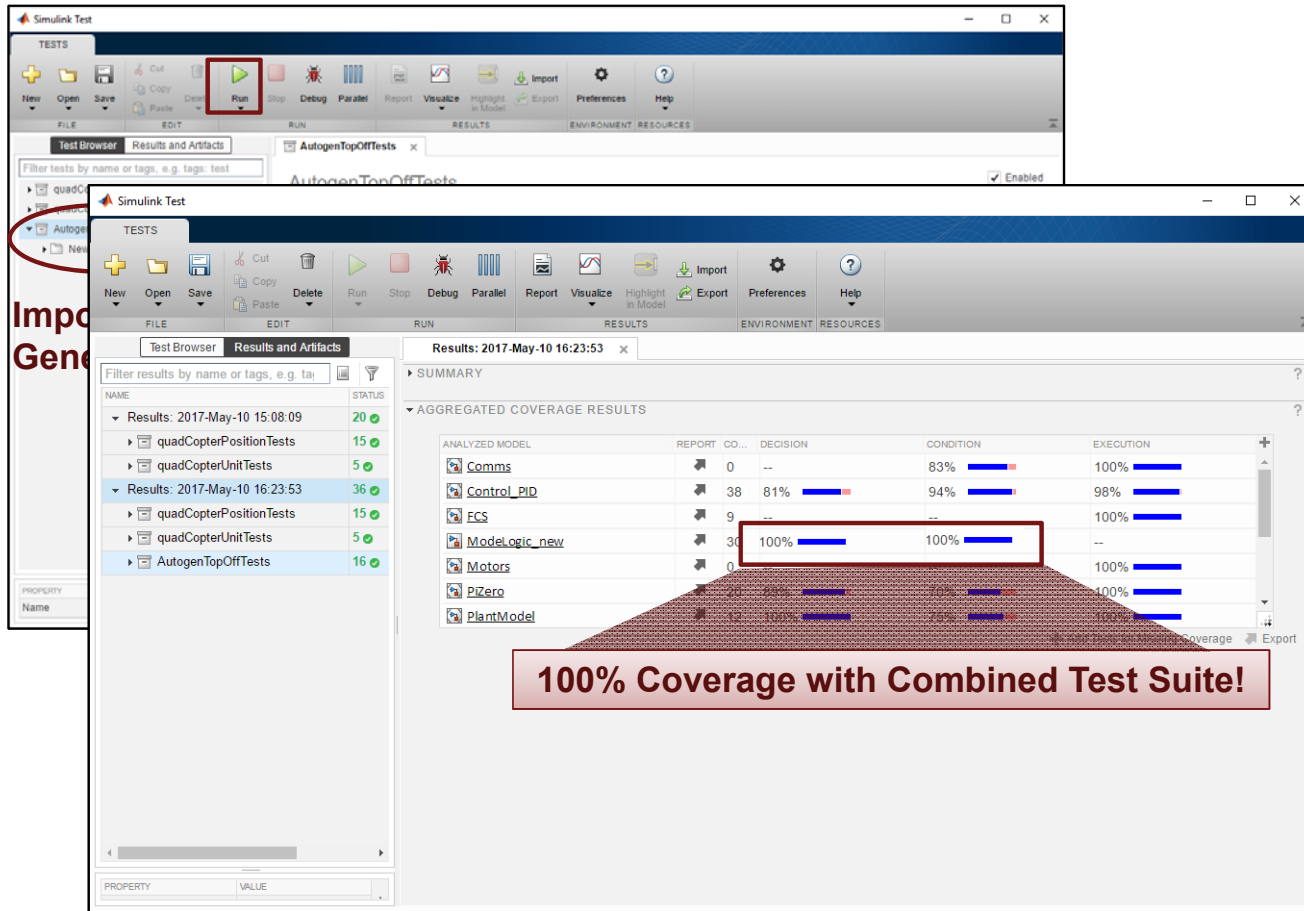
F

Transition: Transition trigger expression **SATISFIED** - View test case

T



Final Coverage Results



Quadcopter Verification & Validation Workflow Summary

- **Implement requirements without leaving Simulink**
 - Edit requirements and models together
 - Track implementation and verification of requirements
 - Respond to changes faster
- **Verify requirement changes through regression testing**
 - Find impacted tests through file dependency analysis
 - Update test's pass/fail criteria (when design is correct!)
 - Automate tests (in parallel!) to ensure all tests still pass
- **Measure model coverage from test cases**
 - Identify unreachable design content via dead logic analysis
 - Fix design or justify dead logic when it's acceptable
 - Generate additional tests to help fill coverage gaps

MathWorks V&V Solutions Page: <https://www.mathworks.com/solutions/verification-validation.html>