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
- TestHarnesses
- 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

Total Savings: $3,720,000
Total Investments: $592,000
ROI: 528%

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.

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

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

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.

ITK Engineering
Produced model coverage reports for MATLAB unit testing scripts for IEC 62304 Compliant Dental Drill Motor.
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 up and down all around.
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

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
- Requirements Browser with Implementation and Verification Status
- Property Inspector to Modify Requirement and View Links
- Enter and Exit Requirements Perspective
- Navigate to associated links (design and test cases)
Track Requirements’ Implementation and Verification Status

- Implemented by: Calibration

- Verified by: TestCalibrationLightOn, TestCalibrationLightOff

- Test results
### Respond to Changes
#### Identifying Modified Requirements Quickly

<table>
<thead>
<tr>
<th>Index</th>
<th>Summary</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>References to aircraft_spec.docx</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>System Description</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>General Characteristics</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Performance Requirements</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Systems Requirements</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Flight Control Requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode Logic Flight Mode</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5.1</td>
<td>Track Altitude</td>
<td></td>
</tr>
<tr>
<td>1.5.1.1</td>
<td>Track Altitude</td>
<td></td>
</tr>
<tr>
<td>1.5.1.2</td>
<td>Calibration</td>
<td></td>
</tr>
<tr>
<td>1.5.1.3</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td>1.5.1.4</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td>1.5.1.5</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td>1.5.1.6</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td>1.5.1.7</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td>1.5.1.8</td>
<td>Track 3D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Crash</td>
<td></td>
</tr>
</tbody>
</table>

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

- **Updated Requirement**
- **New Requirement**
Respond to Changes
Qualifying with Regression Tests

Test Failed
Not Verified
New Requirements → New Test Cases
Testing in Simulink Test

**Test Case**

**Inputs**
- MAT file (input)
- Signal Builder
- Test Sequence

**Test Model**
- System-level test harness
- Unit-level test harness

**Assessments**
- MATLAB Unit Test
- Test Assessment

And more!

**Inputs**
- MAT file (input)
- Excel file (input)

**Assessments**
- MATLAB Unit Test
- Test Assessment
- Excel file (baseline)
Running Tests with Simulink Test
Regression Testing Process

Update Design

Sandbox Testing

Version Control and Continuous Integration
Testing in a Sandbox

Sandbox testing

Identify test impact

Execute tests

Analyze failures

Update the test

Fix the design

Failure due to design bug?

Y

N
Testing in a Sandbox

Sandbox Testing

Identify test impact → Execute tests → Analyze failures

- Failure due to design bug?
  - Y: Fix the design
  - N: Update the test

- Analyze failures
  - Failure due to design bug?
    - Y: Fix the design
    - N: Update the test

- Execute tests
  - Run

- Identify test impact
  - Analyze failures

- Update the test
  - Fix the design
  - Failure due to design bug?

- Identify test impact
  - Execute tests
  - Analyze failures
Requirements are Fully Implemented and Verified

<table>
<thead>
<tr>
<th>Index</th>
<th>Summary</th>
<th>Implemented</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>aircraft_spec</td>
<td>References to aircraft_spec.docx</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>System Description</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>General Characteristics</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Performance Requirements</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Systems Requirements</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Flight Control Requirements</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>1.5</td>
<td>Mode Logic Flight Mode</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Wait for Communications</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Initialization</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Calibration</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Ready for Takeoff a</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Track Altitude</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>1.5.1.6</td>
<td>Track 3D</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>1.5.1.7</td>
<td>Land</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>1.5.1.8</td>
<td>Crash</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

“But, how do I know when we’ve done enough testing?”
Model Coverage in Dynamic Testing

Simulink Coverage Analysis

Stateflow Logic

Address Missing Coverage using Static Analysis

Simulink Design Verifier
Addressing Missing Coverage Workflow

1. Detect Dead Logic
2. Add Tests for Missing Coverage

Simulink Design Verifier:
1. Dead Logic Detection
2. Test Generation
Generate Tests for Missing Coverage
Generate Tests for Missing Coverage

Test Generation

Design → Tests → Add Tests for Missing Coverage → Detect Dead Logic → Measure Coverage → Justify
Final Coverage Results

100% Coverage with Combined Test Suite!
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