Meet Certification Standards with Automated Requirements Based Testing

Paul Urban
Verification and Validation Product Manager
Challenge to Deliver Complex Systems and Meet Standards

- Need to meet industry or customer’s standards
  - DO-178C (Aero), ISO 26262 (Auto), IEC 62304 (Medical), IEC 61508 (Industrial), MISRA, etc.

- Time and cost for safety critical projects estimated 20-30 times more costly*

- Finding defects late increases cost and time

*Source: Certification Requirements for Safety-Critical Software
ISO 26262-6:2018 notes Simulink and Stateflow as Suitable for Software Architecture, Design and as basis for Code Generation

Table 5 — Notations for software unit design

<table>
<thead>
<tr>
<th>Notations</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1a Natural language</td>
<td>++</td>
</tr>
<tr>
<td>1b Informal notations</td>
<td>++</td>
</tr>
<tr>
<td>1c Semi-formal notations</td>
<td>+</td>
</tr>
<tr>
<td>1d Formal notations</td>
<td>+</td>
</tr>
</tbody>
</table>

a. Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or provide an explanation and rationale for decisions captured in the notations.

EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

b. Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design are applied to the model which serves as the basis for the code generation.

Table 2 Software Architecture Design Notations has similar suitability wording for use of Simulink and Stateflow
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases and procedures
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases and procedures
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases and procedures

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

Kostal’s electronic steering column lock module.

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Primary flight control computers from BAE Systems.
Conform to Certification Standards with Reference Workflow

Model Verification

Discover design errors at design time

Code Verification

Gain confidence in the generated code

Module and integration testing at the model level

Reviews and analysis at model level

Prevention of unintended functionality

Back to Back Testing

Textual requirements

Executable specification

Model used for production code generation

Generated code

Object code

Modeling

Code generation

Compilation and linking
Model Verification: Discover design errors at design time

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

- Module and integration testing at the model level
- Reviews and analysis at model level
- Prevention of unintended functionality
- Back to Back Testing
Code Verification: Gain Confidence in the Generated Code

**Code Verification**

- Trace code to model and requirements
- Measure code coverage
- SIL/PIL equivalence testing
- Generate 100% coverage test vectors

Module and integration testing at the model level

Reviews and analysis at model level

Prevention of unintended functionality

Back to Back Testing

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling → Code generation → Compilation and linking
Manage Requirements

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

Model Verification

Module and integration testing at the model level
Manage Requirements

- Ensure all requirements implemented
- Verify the implementation is correct
- Respond quickly to requirement changes
Work with Requirements, Architecture and Design Together

- **External Requirements**
  - .doc, .xls files

- **Authored Requirements**
  - Simulink Requirements
  - Implements
  - Verifies

- **Architecture / Design**
  - Tests

**Requirements Management Tools**

*MATLAB Expo*
Demo: Requirements Perspective
Test and Requirements Traceability

Verification Status
- Passed
- Failed
- Unexecuted
- Missing
Review and Analyze Traceability with Traceability Matrix

Requirement is missing link to Test Case
Review and Analyze Traceability with Traceability Matrix

- Review links between different requirements, model, test
- Filter view to manage large sets of artifacts
- Highlight missing links
- Directly add links to address gaps
Systematic Functional Testing of Model

Model Verification

- Manage requirements
- **Systematically test**
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

Module and integration testing at the model level

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

- Modeling
- Code generation
- Compilation and linking
Requirements Based Verification with Simulink Test

**Functional Requirements**
The flight control system shall ...

**Implemented By**
System Composer / Simulink / Stateflow

**Verified By**

**Test Case**

**Inputs**
- Signal Editor
  - Signal 1
  - MAT / Excel file (input)
- Test Sequence

**Assessments**
- MAT / Excel File (baseline)
- Test Assessments
- MATLAB Unit Test

**Test Harness**

**Simulink Test**
Measure completeness of testing

Model Verification

- Manage requirements
- Systematically test
- **Measure model coverage**
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

Module and integration testing at the model level

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling → Code generation → Compilation and linking
Coverage Analysis to Measure Testing

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

**Simulink**

**Stateflow**

**Code**

**Coverage Reports**
Test and Requirements Traceability in Coverage Results

![Graphical representation of a control system with labeled components and coverage results table]

<table>
<thead>
<tr>
<th>Index</th>
<th>ID</th>
<th>Summary</th>
<th>Implemented</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SET_SPEED</td>
<td>Set speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BRAKE</td>
<td>Brake disengages cruise control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ENABLE</td>
<td>Engage cruise control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>INCREMENT</td>
<td>Increment set speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DECREMENT</td>
<td>Decrement set speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>THROTTLE</td>
<td>Throttle to maintain set speed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Implemented Requirements**

- Brake disengages cruise control
- Engage cruise control
- Increment set speed
- Decrement set speed
- Throttle to maintain set speed

**Verified Requirements**

- Brake disengages cruise control
- Engage cruise control

**Required Testing Details**

- **Implemented**: Brake disengages cruise control, Engage cruise control
- **Verified**: Brake disengages cruise control, Engage cruise control

**Coverage Details**

- Metric: Coverage
- Cyclomatic Complexity: 6
- Conditions: 100% (4/4) condition outcomes
- Execution: 100% (1/1) objective outcomes
Scoping Model Coverage to Requirements-Based Tests

Create a coverage report from coverage results to justify or exclude missing coverage. The filters and updated coverage values will be displayed with this result.

Scope coverage results to linked requirements
Scoping Model Coverage to Requirements-Based Tests

Missing requirement links identified
Test and Requirements Traceability in Coverage Results

3. SubSystem block "PI Controller"

Justify or Exclude

Requirement Testing Details

<table>
<thead>
<tr>
<th>Implemented Requirements</th>
<th>Verified by Tests</th>
<th>Associated Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throttle to maintain set speed</td>
<td>Throttle Test</td>
<td>T6</td>
</tr>
</tbody>
</table>

MATLAB EXPO
Decision covered but not by linked test
Address missing Requirements Based Test Coverage

- Add missing implementation links to requirements

- Update test to increase target speed
100% Coverage but Testing Identified Error in Implementation
Additional Testing Identified Error in Implementation
Scoped Model Coverage to Requirements-Based Tests

DO-178C 6.4.4.2
... coverage information collected during requirements-based testing to confirm that ...

Hit by linked RBT -- Satisfied
Hit, but not by linked RBT -- Unsatisfied
Check standard compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

Reviews and analysis at model level

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling → Code generation → Compilation and linking
Verify Design to Guidelines and Standards

Check for:
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more……
Built in checks for industry standards and guidelines

- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128
- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAB (MathWorks Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)
Shift Verification Earlier With Edit-Time Checking

- Highlight violations as you edit
- Fix issues earlier
- Avoid rework
Detect Design Errors with Formal Methods

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- **Detect design errors**
- Prove model behavior compliance

Reviews and analysis at model level

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling → Code generation → Compilation and linking
Detect Design Errors Using Formal Methods

- Find design errors
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations

- Generate counter example to reproduce error
Prove Model Behavior Compliance

Model Verification

- Manage requirements
- Systematically test
- Measure model coverage
- Check standard compliance
- Detect design errors
- Prove model behavior compliance

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Reviews and analysis at model level

Modeling

Code generation

Compilation and linking
Safety Requirement:

When the brake is applied for three consecutive steps, the throttle shall go to zero.

- Need to ensure the design performs correctly
Model functional and safety requirements
Link requirements to properties

#1: Disable Throttle when Braking
When the brake is applied for three consecutive steps, the throttle shall go to zero.
Prove That Design Meets Requirements
Debugging Property Proving Violations

Design Model

Safety Properties

Model Slicer

Safety Properties/BrakeAssertion

Debug Using Slicer

MATLAB EXPO
Resolve unexpected behavior in a model with Model Slicer

Isolate
Find the area of the model responsible for unexpected behavior

Analyze dependencies
Understand data & control dependencies in large or complex models

Inspect slice regions
Highlight model slices for time windows or failure states & transitions for state flow.

Debug simulation behavior
Step through precompiled slices to understand signal and port value propagation

Correct Model
Code Verification: Gain Confidence in the Generated Code

**Code Verification**

- Trace code to model and requirements
- Measure code coverage
- SIL/PIL equivalence testing
- Generate 100% coverage test vectors

Module and integration testing at the model level

Reviews and analysis at model level

Prevention of unintended functionality

Back to Back Testing

Textual requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling → Code generation → Compilation and linking
Back-to-Back Testing

- Automate SIL testing using Simulink Test
- Testing across releases
Automate Test Creation using Test Manager Wizard

- Guided steps to define component to test, inputs, type of test and format for output
- Wizard generates required test harness
- Auto generate tests using Simulink Design Verifier
Cross Release SIL/PIL Test Harness Generation

- Create a SIL/PIL test harness using code that was generated in a previous release.
- Modify existing SIL/PIL test harnesses to store the build folder path information which can be used for rebuild.
Reference Workflow for Generated Code

Software requirements → Executable specification → Model used for production code generation → Generated code → Object code

Modeling:
- Simulink / Stateflow / AUTOSAR Blockset
- Embedded Coder*

Code generation:
- Simulink Requirements*
- Simulink Test and Simulink Coverage (for MIL)*
- Simulink Check*
- Simulink Design Verifier*
- Simulink Test and Simulink Coverage (for SIL)*
- IEC Cert Kit (for trace)
- Simulink Test and Simulink Coverage (for SIL)*
- Simulink Test and Simulink Coverage (for MIL)*
- Simulink Test (for PIL)*
- Simulink Test (for PIL)*

Compilation and linking:
- Simulink Check*
- Simulink Design Verifier*
- Simulink Test*
- Simulink Coverage (for SIL)*
- IEC Cert Kit (for trace)

*Qualifiable
Customer References and Applications

Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design
Software testing time cut by two-thirds

LS Automotive Reduces Development Time for Automotive Component Software with Model-Based Design
Specification errors detected early

Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks
Verification time cut by up to 50 percent

More User Stories: www.mathworks.com/company/user_stories.html
Use reference workflow to conform to standards

- Shift verification earlier
- Automate manual verification tasks (coding, compiling, back-to-back)
- Measure completeness of Requirements Based Testing
Learn More

- Verification, Validation, and Test Solution Page
- Requirements-Based Testing Workflow Example
- Verifying Models and Code for High-Integrity Systems
- Getting Started with Model Verification and Validation
Thank You!