MathWorks®

Accelerating the pace of engineering and science

DO-178C Workflow with Qualified Code Generation

Simulink Requirements — Requirements Authoring → **SPECIFICATION** **High Level Reqs**

Simulation, Processor and Hardware in the Loop Test Cases Traceability to HLR

LLR and Source Code Traceability to HLR

Design Implementation - MATLAB, Simulink, Stateflow, Simevents, Simscape

Simulation Cases Results → **Simulink Coverage**

DO-331: Table MB.A-3 Verification of Requirements Process (Obj 1, 4 and 5)
Table MB.A-4 Verification of Design Process (Obj 1, 4, 5, 11 and 12)
Table MB.A-7 Verification of Verification Process Results (Obj 5 to 7)

Coverage Metrics

Design constraints (Equivalence classes, Boundary Values, Derived Requirements) → **Simulink Design Verifier**

(FM-B) Design Error Detection and (FM-C) Property Proving

DO-331: Table MB.A-3 and MB.C-3 Verification of Requirements Process (Obj 2, 4 and 7)
Table MB.A-4 and MB.C-4 Verification of Design Process (Obj 2, 4, 7, 9,11)
DO-333: Table FM.A-3 and FM.C-3 Verification of Requirements Process (Obj 8 to 11)
Table FM.A-4 and FM.C-4 Verification of Design Process (Obj 14 to 17)

Note: Formal Methods allow to detect errors in the Model including dead logic, integer overflow, division by zero, and violations of design properties and assertions, out-of-bounds array access and certain other run-time errors in source code

(FM-A) Automatic Test Case Generation

Requirement Based Functional Test Cases + Test Cases identified using Formal Methods

Simulation Test Cases Traceability

Testing Environment Settings → **Simulink Test**

DO-331: Table MB.A-3 and MB.C-3 Verification of Requirements Process (Obj 10)
Table MB.A-4 and MB.C-3 Verification of Design Process (Obj 16)

Model in the Loop (MIL) Functional Testing

Simulink Requirements

Test Cases Traceability to LLR

Simulink Check

Model Conformance Checks

DO-331: Table MB.A-3 Verification of Requirements Process (Obj 2 to 7)
Table MB.A-4 Verification of Design Process (Obj 2 to 7 and 9 to 12)
Table MB.A-5 Verification of Coding and Integration Process (Obj 5)

Modelling Standards

Model Coverage Analysis

DESIGN MODEL

Low Level Reqs

LLR Traceability to HLR → **Simulink Requirements**

Autocoding Settings → **Embedded Coder**

Note: If qualified, this product eliminates or reduces manual code reviews, same as a qualified code generator would also do

DO-331: Table MB.A-5 and C.5 Verification of Coding and Integration Process (Obj 1, 2, 3, 5 and 6)

Simulink Code Inspector

Automatic Code Generation (and Source Code traceability to LLR)

Automatic Code Inspection (verification of structure and Source Code traceability to LLR)

Process that generates the life-cycle data	MB Example 1	MB Example 2	MB Example 3	MB Example 4	MB Example 5
System Requirement and System Design Processes	Requirements allocated to software	Requirements from which the Model is developed	Requirements from which the Model is developed	Requirements from which the Model is developed	Requirements from which the Model is developed
Software Requirement and Software Design Processes	Requirements from which the Model is developed	Specification Model	Specification Model	Design Model	Design Model
	Design Model	Design Model	Textual description		
Software Coding Process	Source Code	Source Code	Source Code	Source Code	Source Code

DO-331 Table MB.1-1 Model Usage Examples

Specification Model is a model representing high-level requirements that provides an abstract representation of functional, performance, interface, or safety characteristics of the software components. A Specification Model does not define software design details such as internal data structures, internal data flow, or internal control flow.

Design Model is a model that defines any software design such as low-level requirements, software architecture, algorithms, component internal data strictures, data flow and/or control flow. A model used to generate Source Code is a Design Model.

```
static void arithm_pointer (void){
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
      *p = 0;
      p++;
    }

  if (get_bus_status() > 0) {
    if (get_oil_pressure() > 0) {
      *p = 5;
    } else {
      i++;
    }
  }

  i = get_bus_status();

  if (i >= 0) {
    *(p - i) = 10;
  }
}
```

Green: reliable — safe pointer access

Range data — tool tip

Red: faulty — out of bounds

Gray: dead — unreachable code

Orange: unproven — may be unsafe for some conditions

Purple: violation — MISRA-C code rules

variable 'i' (int32): [0 .. 99]
assignment of 'i' (int32): [1 .. 100]

Coding Standards → **Polyspace Bug Finder** → Code Conformance (MISRA,…)

DO-331: Table MB.A-5 and C.5 Verification of Coding and Integration Process (Obj 3 to 6)
DO-332: Table OO.A-5 and C.5 Verification of Coding and Integration Process (Obj 3 to 6)

Doc Templates Scripts → **Simulink Report Generator**

DO-331: Table MB.A-3 Verification of Requirements Process (Obj 1 to 7)
Table MB.A-4 Verification of Design Process (Obj 1 to 6, 8 to 12)

- DO-331 MB.A-4 Simulation Results Report
- DO-331 MB.A-4 Model Coverage Report
- DO-331 MB.A-4 Model Standards Report
- DO-333 FM.A.4 Model Design Error Report
- DO-331 MB.A-4 System Design Description
- DO-331 MB.A.5 Source Code Report
- DO-331 MB.A.5 Coding Standards Report
- DO-331 MB.A.5 Code Inspection Report
- DO-331 MB.A.6 Code Coverage Report
- DO-333 FM.A.6 Run-Time Error Report
- DO-331 MB.A.7 Low Level Test Cases
- DO-331 MB.A.7 EOC Test Results Report

DO-331: Table MB.A-5 and C.5 Verification of Coding and Integration Process (Obj 2, 3, 6)
DO-332: Table OO.A-5 and C.5 Verification of Coding and Integration Process (Obj 2, 3, 6)
DO-333: Table FM.A-5 and C.5 Verification of Coding and Integration Process (Obj 2, 3, 6 and 10 to 13)
Table FM.A-6 and C.6 Testing of Outputs of Integration Process (Obj 1-4)
Table FM.A-7 and C.7 Verification of Verification Process Results (Obj 1, 2 and 5 to 10)

Verification Objectives Settings → **Polyspace Code Prover** → Prove Absence of Run-Time Errors

SOURCE CODE

Software in the Loop (SIL) Testing

SIL and PIL Test Cases Traceability to LLR and HLR

Supported coverage types:
- Decision coverage
- Condition coverage
- MC/DC
- Lookup table coverage
- Signal range coverage

SIL Test Cases Results → **Simulink Coverage** → Code Coverage

- Table A-7 Verification of Verification Process Results (Obj 5 to 7)
- Table M.B.A-7 Verification of Verification Process Results (Obj 5 to 7)

SW DAL	MC/DC	Decision coverage	Statement coverage	data coupling and control coupling
A	100% (Ind)	100% (Ind)	100% (Ind)	100% (Ind)
B		100% (Ind)	100% (Ind)	100% (Ind)
C			100%	100%

Testing Environment Settings

Reuse of the Simulation Test Cases
Adding EOC specific Test Cases → **Simulink Test**

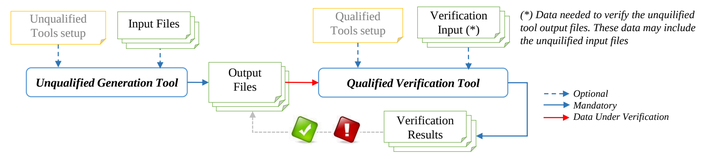
DO-178C: Table A-6 Table A-6: Testing of Outputs of Integration Process (Obj 1 to 4)
DO-178C: Table A-7 Verification of Verification Process Results (Obj 3 and 4)

Compiler

Processor and Hardware in the Loop (PIL and HIL) Testing

EXECUTABLE OBJECT CODE

DO-178C: Section 12.2 – FAQ D.7: How Might One Use a Qualified Tool to Verify the Outputs of an Unqualified Tool?



SW Level	Tool Qualification Criteria		
	1	2	3
A	*TQL-1*	*TQL-4*	*TQL-5*
B	*TQL-2*	*TQL-4*	*TQL-5*
C	*TQL-3*	*TQL-5*	*TQL-5*
D	*TQL-4*	*TQL-5*	*TQL-5*

Tool Criteria Definition

1: Development Tool whose output is part of the resulting SW and thus could insert and error

2: Verification Tool that automates verification process (es) and thus could fail to detect and error, and whose output is used to justify the elimination or reduction of:
- Verification process (es) other than that automated by the tool, or
- Development process (es) that could have an impact on the airborne (or NS/ATM) SW

3: Verification Tool that automates verification process(es) and thus could fail to detect and error

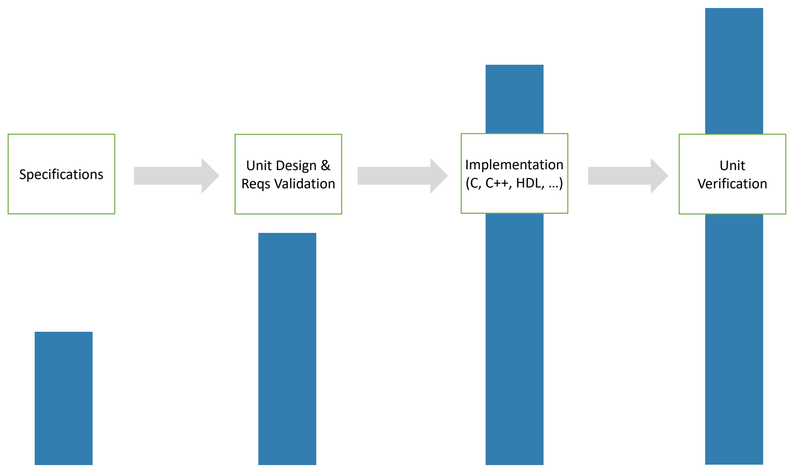
DO Qualification Kit

- Tools Requirements, User Manual and other MathWorks documentation
- Workflow Documentation and Tool Qualification Plans templates
- Verification Inputs Test Cases and Expected Results

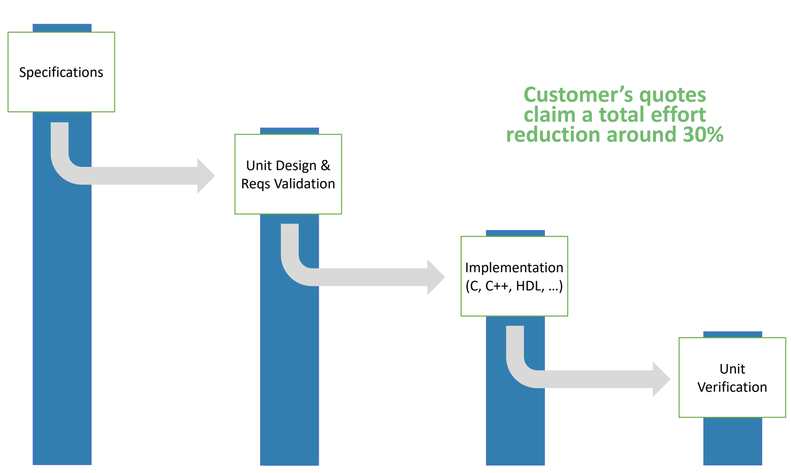
Unqualified Tools
Qualified Tools
Configuration Inputs
Artifacts

DO-178C supplements
- DO-330 for Tool Qualification
- DO-331 for Model-Based Design
- DO-332 for Object Oriented Techniques
- DO-333 for Formal Methods

Effort Distribution in Traditional Development Workflows



Effort Distribution in Model-Based Design Workflows



MATLAB® & SIMULINK®

Prepared by: Albert Ramirez Perez, MathWorks
AlbertR@MathWorks.com
+49-89-45235-6772
January 2018