Leveraging Model-Based Design to Meet High-Integrity Software Certification Objectives

Prashant Mathapati
Application Engineering Group
High-Integrity Applications

Definition: cf. Buncefield Investigation Glossary
http://www.buncefieldinvestigation.gov.uk/glossary.htm

Software-based systems that are designed and maintained so that they have a high probability of carrying out their intended function.
Development Processes for High-Integrity Applications
Development Processes for High-Integrity Applications

- High integrity applications development follows standards and guidelines.

- Standards and Guidelines have objectives for development process activities:
  - Impose additional constraints on development
  - Require creation of additional artifacts
  - Require more thorough verification, validation and testing activities

- Standards and Guidelines require evidence that the objectives were met to certify:
  compliance demonstration
Development Process for High-Integrity Applications: Compliance Demonstration

- System certification involves a compliance demonstration:
  - applicant must provide evidence that the objectives of the standard were met

- Compliance demonstration is a lengthy and labour-intensive process
Standards Landscape

- **Generic Standards**
  - IEC 61508* (= EN 61508)

- **Automotive Standards / Guidelines**
  - ISO 26262
  - MISRA-C
  - MAAB Guidelines

- **Aerospace Standards**
  - DO-178C
  - DO-254
ISO 26262 “Road Vehicles - Functional Safety”

- Emerging functional safety standard for passenger cars
- Facilitates modern software engineering paradigms, e.g.
  - Model-Based Design
  - Early Verification and Validation
  - Code generation
- **Automotive Safety Integrity Levels** (ASIL A…D) for systems/software
- **Tool Confidence Levels** (TCL 1…3) for tools
## ISO 26262 Structure

<table>
<thead>
<tr>
<th>ISO 26262-n</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 26262-1</td>
<td>• Vocabulary</td>
</tr>
<tr>
<td>ISO 26262-2</td>
<td>• Management of functional safety</td>
</tr>
<tr>
<td>ISO 26262-3</td>
<td>• Concept phase</td>
</tr>
<tr>
<td>ISO 26262-4</td>
<td>• Product development: system level</td>
</tr>
<tr>
<td>ISO 26262-5</td>
<td>• Product development: hardware level</td>
</tr>
<tr>
<td>ISO 26262-6</td>
<td>• Product development: software level</td>
</tr>
<tr>
<td>ISO 26262-7</td>
<td>• Production and operation</td>
</tr>
<tr>
<td>ISO 26262-8</td>
<td>• Supporting processes</td>
</tr>
<tr>
<td>ISO 26262-9</td>
<td>• ASIL-oriented and safety-oriented analyses</td>
</tr>
<tr>
<td>ISO 26262-10</td>
<td>• Guideline</td>
</tr>
</tbody>
</table>

- **Definitions**
  - Model-Based Design
  - Early verification and validation
  - Code generation
  - Tool classification and qualification
ISO 26262 and Model-Based Design

- Model-Based Design, early verification & validation, and code generation are **integral parts of ISO 26262** (Examples from part 1)

---

**ISO/DIS 26262-1**

1.74 **model-based development**

development that uses models to describe the functional behavior of the elements which are to be developed

**NOTE** Depending on the level of abstraction used for such a model it can be used for simulation or **code generation** or both.

---

1.112 **semi-formal notation**

description technique that has its syntax completely defined, but its semantics definition may be incomplete

**EXAMPLE** Graphical modelling approaches such as UML use case diagrams, UML class diagrams, **block diagrams** and state charts.

---

1.113 **semi-formal verification**

verification that is based on a description using semi-formal notation
ISO 26262 and Model-Based Design

- Model-Based Design is one of the software development paradigms directly addressed in ISO 26262
ISO26262 Reference Workflow for Verification and Validation and Code Generation

Model Verification
- Module and integration testing at the model level
- Review and static analysis at the model level

Code Verification
- Equivalence testing
- Prevention of unintended functionality

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

Modeling → Code generation → Compilation and linking
ISO26262 Example Tool Chain for Verification and Validation and Code Generation

Model Advisor (SLVnV), Property proving (SLDV)

Traceability report (Embedded Coder), Traceability matrix generation (IEC Cert Kit), Bullseye code coverage integration (Embedded Coder)

PIL testing / CGV (Embedded Coder), Test generation (SLDV)

Simulink, Model coverage (SLVnV), Requirements Management Interface (SLVnV)

Module and integration testing at the model level

Equivalence testing

Review and static analysis at the model level

Prevention of unintended functionality

PolySpace for code verification

Textual requirements

Executable specification

Modeling

Simulink, Stateflow, Simulink Fixed Point

Model used for production code generation

Generated C code

Object code

Code generation

Compilation and linking

Embedded Coder
Coverage of ISO 26262–6 and -8 Requirements

### Table 7 – Notations for software unit design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>Applicable Model-Based Design Tools / Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Natural language</td>
<td>++</td>
<td>- Simulink – Model Info block, DocBlock</td>
</tr>
<tr>
<td>1b Informal notations</td>
<td>++</td>
<td>- Simulink Verification and Validation – System Requirements block, Requirements Management Interface (RMI)</td>
</tr>
</tbody>
</table>

### Table 9 – Methods for the verification of the software unit design and implementation (1/2)

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>Applicable Model-Based Design Tools / Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Walk-through</td>
<td>++</td>
<td>- Simulink Report Generator – Web View, System Design</td>
</tr>
<tr>
<td>1b Inspection</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>1c Semi-formal verification</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>1d Formal verification</td>
<td>o</td>
<td></td>
</tr>
</tbody>
</table>

### Table 10 – Methods for software unit testing (2/2)

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>Applicable Model-Based Design Tools / Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b Interface test</td>
<td>++</td>
<td>- Simulink Design Verifier - Test case generation</td>
</tr>
<tr>
<td>1c Fault injection test</td>
<td>+</td>
<td>- Simulink, Stateflow</td>
</tr>
<tr>
<td>1d Resource usage test</td>
<td>+</td>
<td>- Simulink Design Verifier - Test case generation</td>
</tr>
<tr>
<td>1e Back-to-back test between model and code, if applicable</td>
<td>+</td>
<td>- Embedded Coder – Procesor-in-the-Loop (PIL) testing, Code metrics report</td>
</tr>
</tbody>
</table>

www.mathworks.com/automotive/standards/iso-26262.html (=> Free technical kit)
TÜV Certification of MathWorks Products

- **TÜV SÜD certified:**
  - **Embedded Coder**
    for use in development processes which need to comply with IEC 61508, or ISO 26262
  - **Polyspace products**
    for use in development processes which need to comply with IEC 61508, ISO 26262, or EN 50128
  - **Simulink PLC Coder**
    for use in development processes which need to comply with IEC 61508, or IEC 61511

Note:
Real-Time Workshop Embedded Coder, Polyspace products for C/C++, and Simulink PLC Coder were not developed using certified processes
Standards Landscape

- **Generic Standards**
  - IEC 61508* (= EN 61508)

- **Automotive Standards / Guidelines**
  - ISO 26262
  - MISRA-C
  - MAAB Guidelines

- **Aerospace Standards**
  - DO-178C
  - DO-254
RTCA/DO-178 Software Considerations in Airborne Systems and Equipment Certification

Purpose - Provides guidelines for the production of airborne software that performs its intended function and avoids unintended function.

Guidance includes

- Software life cycle objectives
- Development and verification activity descriptions
- Descriptions of evidence needed to indicate the objectives were satisfied

http://www.rtca.org/
## DO-178 History

<table>
<thead>
<tr>
<th>Version</th>
<th>RTCA Publication*</th>
<th>FAA Acceptance**</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO-178</td>
<td>1982</td>
<td>1982</td>
</tr>
<tr>
<td>DO-178A</td>
<td>1985</td>
<td>1986</td>
</tr>
<tr>
<td>DO-178B</td>
<td>1992</td>
<td>1993</td>
</tr>
<tr>
<td>DO-178C</td>
<td>Dec. 2011</td>
<td>TBD</td>
</tr>
</tbody>
</table>

*Published in Europe by EUROCAE as ED-12
**Accepted by EASA in Europe as AMC 20-115B

AC 20-115B
http://rgl.faa.gov/
Model Based Development Issues - DO-178B Mapping

• Model based development merges the following development steps:
  – System Design
  – High-Level Software Requirements
  – Low-Level Software Requirements

• Some people view the graphical model as a higher level software language, the code generator is simply another compiler stage

• DO-178B and ARP-4754 do not address this possibility
  – Currently it is difficult to get consensus about mapping to DO-178B
  – Certification authorities need to start addressing this issue
DO-178C – What’s New

- Clarifies that ARP4754A should be used for Systems Engineering

SYSTEM ASPECTS RELATING TO SOFTWARE DEVELOPMENT

This section discusses those aspects of the system life cycle processes necessary to understand the software life cycle processes. System life cycle processes can be found in other industry documents (for example, SAE ARP4754A).

- Notes additional supplement documents should be used for specific techniques

One or more supplements to this document exist and extend the guidance in this document to a specific technique. Supplements are used in conjunction with this document and may be used in conjunction with one another. Unless alternatives are used (see 1.4.1), if a supplement exists for a specific technique, the supplement should
DO-178C Supplements

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
Key Development and Verification Tools for DO

DO-178
- Simulink Code Inspector
- Polyspace

DO-254
- HDL Verifier

DO Qualification Kit

Embedded Coder
- Simulink Design Verifier
- Simulink Verification and Validation
- Simulink Report Generator
- Simulink, Stateflow & Libraries

HDL Coder
- HDL Code

C Code

Simulink, Stateflow & Libraries
DO-178C Supplements

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
DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A

- Specification model

 Specification Model – A model representing high-level requirements that provides an abstract representation of functional, performance, interface, or safety characteristics of 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

 Glossary

 Design Model – A model that defines any software design such as low-level requirements, software architecture, algorithms, component internal data structures, data flow and/or control flow. A model used to generate Source Code is a Design Model.
DO-331- Analysis Definitions

Introduces new modeling activities

- Simulation

**Model simulation** – The activity of exercising the behavior of a model using a model simulator.

- Model Coverage

**Model coverage analysis** – An analysis that determines which requirements expressed by the Design Model were not exercised by verification based on the requirements from which the Design Model was developed. The purpose of this analysis is to support the detection of unintended function in the Design Model, where coverage of the requirements from which the model was developed has been achieved by the verification cases.
DO-331 – MathWorks Product Support

Products impacted by DO-331:

- **DO Qualification Kit (for DO-178)**
  - Qualification kits for verification tools

- **Simulink family of products, incl:**
  - Design: Simulink, Stateflow
  - Code Generation: Embedded Coder

- **Polyspace code verifiers**
  - Polyspace Client C/C++
  - Polyspace Server C/C++
DO-178C Supplements

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
DO-333 – MathWorks Product Support

Products impacted by DO-333 include:

- **DO Qualification Kit** (for DO-178C)
  - Qualification kits for MathWorks verification tools

- **Simulink Design Verifier**
  - Verifies designs using formal methods
  - Detect errors including dead logic, integer overflow, division by zero, and violations of design properties and assertions

- **Polyspace code verifiers**
  - Prove absence of run-time errors in source code using formal methods (abstract interpretation)
  - Proves absence of overflow, divide-by-zero, out-of-bounds array access, and certain other run-time errors in source code using static code analysis
  - Supports code generated automatically or manually written
Model-Based Design Workflow With Qualified Tools

- **Trace:**
  - **System Design Description**
  - **Model/Code Trace Report**
  - **Simulink Code Inspector**
  - **One Time Trace for Level A**

Legend:
- Manual steps are normal font
- Tools or features are shown in italics
- Qualified tools are in bold/color

**Models**
- **Simulink & Stateflow**
- Conformance: Model Advisor
- **Verify:** Test Cases & Review Model Coverage
- **Verify:** Testing with Embedded IDE Link PolySpace Code Coverage Tool

**Source Code**
- Embedded Coder
- Conformance: Polyspace
- **Verify:**
  - Simulink Code
  - Inspector

**Object Code**

**Requirements**
- Validate

**Compiler/IDE**

**Report**

Simulink Code Inspector
National Aerospace Laboratories Proves Benefits of Model-Based Design for DO-178B Flight Software Development

Challenge
Accelerate the development of DO-178B Level A certified flight software

Solution
Complete a stall warning system pilot project using Simulink and Embedded Coder, quantify improvements in development efficiency, and adopt Model-Based Design for future DO-178 projects

Results
• Code analysis and design time cut in half
• Integrated workflow established
• Consistent, high-quality code generated

"Simulink and Model-Based Design reduced the effort needed to upgrade functionality, code analysis time, and design time for the safety-critical embedded system. The compatibility of Simulink with the DO-178 process gave us confidence to use Model-Based Design for our upcoming DO-178 projects. “

Manju Nanda,
CSIR-National Aerospace Laboratories

SARAS, CSIR-NAL's multirole light transport aircraft.
Find more Info on:

- **DO Qualification Kit**

- **IEC Certification Kit**

- **Success Stories**
  - [http://www.mathworks.in/company/user_stories/industry.html](http://www.mathworks.in/company/user_stories/industry.html)