Standardkonforme Absicherung mit Model-Based Design

MATLAB EXPO 2014

Dr. Marc Segelken
Principal Application Engineer
Safety Standards for Embedded Systems

- IEC 61508
- ISO 26262
- EN 50128
- DO-178C
- DO-278A
- DO-330
- DO-254
- ...

IEC Certification Kit

DO Qualification Kit
IEC Certification Kit

① Supports tool qualification

② Streamlines ISO 26262 compliant development of embedded systems

www.mathworks.com/products/iec-61508/
# Process Compliance Demonstration

Annotated method tables with suggestions on how to use Model-Based Design processes and tools to apply the methods listed in ISO 26262-6

## Table 9 - Methods for Verification of Software Unit Design and Implementation

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>Applicable Model-Based Design Tools and Processes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Simulink Report Generator – Web View, System Design Description (SDD) report</td>
<td>Unit design walkthroughs can be based on a model, a generated Web View, or an SDD report.</td>
</tr>
<tr>
<td>1a Walkthrough</td>
<td>++</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>1b Inspection</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

## Table 10 - Methods for Software Unit Testing

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>Applicable Model-Based Design Tools and Processes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Requirements-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1a IEC Certification Kit – Traceability matrix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink – Signal Builder block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink Verification and Validation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink Verification and Validation – Component testing capabilities</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MATLAB EXPO 2014
Deutschland
9. Mai – München
Reference Workflow

Model-Based Design for ISO 26262

Exemplary verification and validation process for safety-related software created using Model-Based Design
Reference Workflow

*Model-Based Design for ISO 26262*

Simulation / model testing
(Simulink)

Model coverage
Req. Mgmt. Int.
(Simulink V&V)

Module and integration
testing at the model level

Review and static analysis
at the model level

Equivalence testing

Prevention of
unintended functionality

Textual requirements

Executable specification

Modeling

Model used for
production code generation

Generated C code

Object code

Code generation

Compilation and linking
ISO 26262 Modeling Guidelines Checks

Model standards checking (Simulink V&V)

Module and integration testing at the model level

Equivalence testing

Prevention of unintended functionality

Review and static analysis at the model level

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

- Code generation
- Compilation and linking

Modeling
Code Generation Verification in the context of ISO 26262

### Table 12 — Methods for software unit testing

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Interface test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Fault injection test</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Resource usage test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Back-to-back test between model and code, if applicable</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

Notes:
- Some aspects of the resource usage test can only be evaluated properly when the software unit tests are executed on the target hardware or if the simulator for the target processor supports resource usage tests.
- This method requires a model that can simulate the functionality of the software units. Here, the model and code are simulated in the same way and results compared with each other.

### Table 15 — Methods for software integration testing

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>External interface test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Fault injection test</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Resource usage test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Back-to-back test between model and code, if applicable</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

Notes:
- This method requires a model that can simulate the functionality of the software components. Here, the model and code are simulated in the same way and results compared with each other.
Traceability and Code Coverage

Module and integration testing at the model level

- Textual requirements
- Executable specification
- Modeling

Review and static analysis at the model level

Model used for production code generation

Generated C code

Object code

Equivalence testing

Prevention of unintended functionality

Traceability matrix analysis (IEC Certification Kit) or model vs. code coverage (third-party tool)

Code generation

Compilation and linking

Embedded Coder

Third-party tool
Reference Workflow
Model-Based Design for ISO 26262

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

Modeling → Simulink/Stateflow

PIL test (Embedded Coder)

Model standards checking (Simulink V&V)

Simulation / model testing (Simulink)
Model coverage (Simulink V&V)

Module and integration testing at the model level

Requirement Int. (Simulink V&V)

Equivalence testing
Prevention of unintended functionality

Traceability matrix analysis (IEC Certification Kit) or model vs. code coverage (third-party tool)

Model used for production code generation → Generated C code → Object code

Code generation

Compilation and linking

Third-party tool

Embedded Coder
Advanced Reference Workflow  
**Additional Best Practices**

- Simulation / model testing (Simulink)
- Model coverage Req. Mgmt. Int. (Simulink V&V)
- Module and integration testing at the model level
- Review and static analysis at the model level
- Model used for production code generation
- Generated C code
- Object code
- PIL test (Embedded Coder)
- Test generation (Simulink Design Verifier)
- Run-time error detection (Polyspace products)
- Equivalence testing
- Prevention of unintended functionality
- Traceability matrix analysis (IEC Certification Kit) or model vs. code coverage (third-party tool)
- MISRA-C checking (Polyspace products)
- Property Proving (Simulink Design Verifier)
- Req. Mgmt. Int. (Simulink V&V)
- Model standards checking (Simulink V&V)
- Test generation (Simulink Design Verifier)
- Run-time error detection (Polyspace products)
- Code generation
- Compilation and linking
- Third-party tool
- Embedded Coder
- Simulink/Stateflow
- Textual requirements
- Executable specification

---

**Simulation / model testing**
- Model coverage
- Req. Mgmt. Int.
- Module and integration testing
- Review and static analysis

**Model used for production code generation**
- Generated C code
- Object code

**Traceability matrix analysis**
- (IEC Certification Kit)
- Model vs. code coverage

**MISRA-C checking**
- (Polyspace products)

---

**Textual requirements**
- Modeling

**Modeling**
- Simulink/Stateflow
- Embedded Coder
- Third-party tool
ISO 26262 Tool Qualification Approach - Details

I. Tool Classification

<table>
<thead>
<tr>
<th>Tool use cases</th>
<th>Tool impact</th>
<th>Tool error detection</th>
<th>Tool confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC 1..n</td>
<td>TI 2</td>
<td>TD 3 (Medium)</td>
<td>TCL 3</td>
</tr>
<tr>
<td></td>
<td>TI 1</td>
<td>TD 2 (High)</td>
<td>TCL 2</td>
</tr>
</tbody>
</table>

II. Tool Qualification

- Increasing qualification requirements
- ASIL
- Qualification methods for TCL3
- Qualification methods for TCL2
- Qualification not required
Qualification of MathWorks Tools

Tool qualification may involve multiple parties

- **Tool user**
  - Responsible for final tool qualification in the context of the application

- **Tool vendor**
  - Conducts generic pre-classification and pre-qualification based on reference use cases / reference workflow

- **3rd party assessor** (optional)
  - Provides independent assessment of reference workflow and pre-qualification artifacts

MathWorks
Pre-classification / qualification based on typical use cases (reference workflows)

TÜV SÜD
Independent Assessment

ISO 26262 Tool Qualification Kit

Tool User
Project-specific adaptation
Independent Assessment by TÜV SÜD

Example

Certificate

Assessment Report

5.2 Usage considerations for development processes which need to comply with IEC 61508, ISO 26262, EN 50128, or derivative standards

The capabilities of Simulink® Verification and Validation™ and Simulink® Design Verifier™ listed in sections 2.1.2 and 2.2.2 respectively are certified for use in development processes which need to comply with IEC 61508, ISO 26262, EN 50128, or derivative standards. The two verification tools allow the automation of core verification and validation activities for Simulink models and generated code.
Qualification of Model-Based Design Tools

☑ pre-qualified for all ASILs according to ISO 26262

Simulink Verification and Validation ☑
- Module and integration testing at the model level

Simulink Design Verifier ☑
- Equivalence testing

PolySpace Client/Server for C/C++
- Prevention of unintended functionality

Textual requirements ➔ Executable specification ➔ ...

Modeling ➔ Model used for production code generation ➔ Generated C code ➔ Object code

Code generation ➔ Compilation and linking

Embedded Coder ☑
DO Qualification Kit

① Qualify Simulink and Polyspace verification tools for DO-178 and DO-278

② Streamline certification of embedded systems

www.mathworks.com/products/do-178/
Simulink Code Inspector
Automate DO-178C Code Reviews

Independently verify that Embedded Coder generated code traces to and complies with low-level requirements

- Demonstrate that model and source code match structurally
- Provide model ⇔ code traceability data
- Eliminate/reduce manual code reviews for DO-178C software
- Same certification credits as qualified code generator
Summary

- Model-Based Design offers many Verification and Validation methods to address objectives of standards

- Complete Reference Workflows are guiding through the development process

- Certification and Qualification Kits easing Tool Qualification process