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

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>1a</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></td>
<td></td>
<td>Embedded Code – Code generation report</td>
<td>Code walkthroughs can be based on HTML code generation reports or code generation reports with an integrated Web View of the model.</td>
</tr>
<tr>
<td>1b</td>
<td>+</td>
<td>Simulink Report Generator – Web View, System Design Description (SDD) report</td>
<td>Unit design inspections can be based on a model, a generated Web View, or an SDD report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulink Verification and Validation – Model Advisor checks</td>
<td>Unit design inspections can be supported by ISO 26262.</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>2a</td>
<td></td>
<td>Simulink Verification and Validation – Requirements Management Interface (RMI)</td>
<td>RMI can be used to establish bidirectional links between textual requirements and models.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IEC Certification Kit – Traceability matrix</td>
<td>Generated traceability matrices can be used to document and review existing links between textual requirements, models, and code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulink – Signal Builder block</td>
<td>Signal Builder blocks can be used to create open-loop model tests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StateFlow – Dynamic test vector charts</td>
<td>Dynamic test vector charts can be used to create closed-loop, reactive model tests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulink Verification and Validation – Component testing capabilities</td>
<td>Component testing capabilities can be used to create model test harnesses. They also enable a requirements pane in the Signal Builder that can be used to link tests with textual requirements.</td>
</tr>
</tbody>
</table>
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 → ...

Model used for production code generation

Generated C code → Object code

Modeling

Code generation
Compilation and linking
ISO 26262 Modeling Guidelines Checks

Model standards checking (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 → Model used for production code generation → Generated C code → Object code

Modeling → Code generation → Compilation and linking
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>1a Requirement-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b Interface test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1c Fault injection test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1d Resource usage test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1e Back-to-back test between model and code, if applicable</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>

- This method is based on injecting arbitrary faults into the software in order to test its ability to maintain or restore its functional behavior under fault conditions.
- Some aspects of the resource usage test can only be verified properly when the software unit tests are executed on the target processor or if the software unit supports resource usage tests.
- This method requires a model that can simulate the functionality of the software unit. 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>1a Requirement-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b External interface test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1c Fault injection test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1d Resource usage test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1e Back-to-back test between model and code, if applicable</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>

- This method is based on injecting arbitrary faults into the software in order to test its ability to maintain or restore its functional behavior under fault conditions.
- To ensure the fulfillment of requirements influenced by the hardware architecture, such as average and maximum processor performance, minimum or maximum execution times, storage usage (e.g., RAM for stack and heap, ROM for program and data) and the bandwidth of communication links (e.g., data buses) have to be determined.
- 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.

Equivalence testing

Prevention of unintended functionality

Model used for production code generation

Generated C code

Object code

Code generation

Compilation and linking
Traceability and Code Coverage

Module and integration testing at the model level

Review and static analysis at the model level

Model used for production code generation

Equivalence testing

Prevention of unintended functionality

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

Textual requirements

Executable specification

Modeling

Generated C code

Object code

Code generation

Compilation and linking

Third-party tool

Embedded Coder
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

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

Equivalence testing
Prevention of unintended functionality

Model used for production code generation

Object code

Generated C code

Code generation

Compilation and linking

Third-party tool

Model standards checking (Simulink V&V)
PIL test (Embedded Coder)

Executable specification

Textual requirements

Modeling

Simulink/Stateflow

Embedded Coder
Advanced Reference Workflow

Additional Best Practices

Model standards checking (Simulink V&V)
Property Proving (Simulink Design Verifier)

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)

Traceability matrix analysis
(IEC Certification Kit) or
model vs. code coverage (third-party tool)
MISRA-C checking (Polyspace products)

Equivalence testing

Prevention of unintended functionality

Textual requirements

Executable specification

Modeling

Simulink/Stateflow

Code generation

Compilation and linking

Embedded Coder

Third-party tool
ISO 26262 Tool Qualification Approach - Details

I. Tool Classification

- Tool use cases
  - UC 1..n
- Tool impact
  - TI 1
  - TI 2
- Tool error detection
  - TD 1
  - TD 2
  - TD 3
- Tool confidence level
  - High
  - Medium

II. Tool Qualification

- 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
Independent Assessment by TÜV SÜD

Example

Certificate

No. Z10 11 01 67052 008


24 MathWorks Plaza
Natick, MA 01760-2091
USA

Product(s): Software Tool for Safety Related Development

Model(s): Simulink® Verification and Validation™
Simulink® Design Verifier™

Parameters:

The verification tools are intended for use on verified, validated software according to IEC 61508, ISO 26262, EN 50128, and derivative standards.

Toolset:

IEC 61508-3:2011 (C664810)
IEC 61508-4 (C664811)

Toolset according to:

IEC 61508-3:2011 (C664810)
IEC 61508-4 (C664811)

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
- Review and static analysis at the model level

Simulink Design Verifier
- Equivalence testing
- Prevention of unintended functionality

PolySpace
- Client/Server for C/C++

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

Modeling → Code generation

Generated C code → Object code

Compilation and linking
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