Using Model-Based Design in conformance with safety standards

Kristian Lindqvist
Senior Engineer

MATLAB EXPO 2014
High-Integrity Applications

Software-based systems that are designed and maintained such that they have a high probability of carrying out their intended function.
Why are we here?
I want to show:

- How can MathWorks tools help develop high-integrity systems?
- What are the standard requirements for using the tools?
- How can MathWorks simplify meeting standards?
How can MathWorks tools help develop high-integrity systems?
Exemplary verification and validation process for safety-related software created using Model-Based Design
Reference Workflow

Model-Based Design for High-Integrity Software

Model Verification
Detect design errors at design time

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
Verification by simulation
Test Coverage Analysis for Models
Tracing Requirements ↔ Model

- Creating links between textual documents and model objects

Simulation / model testing (Simulink)
Model coverage (Simulink V&V)
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
Model Static Review and Analysis
High-Integrity 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 ➔ ... ➔ Modeling ➔ Model used for production code generation ➔ Generated C code ➔ Object code

Code generation ➔ Compilation and linking
High-Integrity Modeling Guidelines Checks

Model Advisor checks to facilitate standard objectives at the model level
Reference Workflow

Model-Based Design for High-Integrity Software

Code Verification

Gain confidence in the generated code

- Review and static analysis at the model level
- Module and integration testing at the model level
- Equivalence testing
- Prevention of unintended functionality

Textual requirements → Executable specification → Modeling → Generated C code → Object code

Model used for production code generation
Code Verification in the context of ISO 26262
Processor In-the-Loop

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

Modeling

Code generation

Generated C code

Compilation and linking

Object code
Code Verification in the context of ISO 26262
Prevention of unintended functionality

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

Generate traceability information covering requirements, model elements, and generated code

- Use Microsoft Excel to track, and annotate generated traceability matrices for your project
- Fulfill requirements to document traceability information and to demonstrate absence of unintended functionality

```matlab
>> iec.ExportTraceReport('model_name')
```
Example: Door Locking System

- Small realistic example
- System model
- Modular development
- Automation
- Artifacts creation
Reference Workflow

**Model-Based Design for High-Integrity Software**

- Model standards checking (Simulink V&V)
- Simulation / model testing (Simulink)
- Model coverage (Simulink V&V)
- Req. Mgmt. Int. (Simulink V&V)

Preparation stages:

- Textual requirements
- Executable specification
- Modeling

Production stages:

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

Testing stages:

- PIL test (Embedded Coder)
- Traceability matrix analysis (IEC Certification Kit) or model vs. code coverage (third-party tool)
- Equivalence testing
- Prevention of unintended functionality

Review and static analysis at the model level

Module and integration testing at the model level

Simulink/Stateflow

Embedded Coder

Third-party tool
Advanced Reference Workflow
Additional Best Practices

- Model standards checking (Simulink V&V)
- Property Proving (Simulink Design Verifier)
- Simulation / model testing (Simulink)
- Model coverage (Simulink V&V)
- Req. Mgmt. Int. (Simulink V&V)
- Module and integration testing at the model level
- Review and static analysis at the model level
- Textual requirements
- Executable specification
- 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)
- Prevention of unintended functionality
- Equivalence testing
- Embedded Coder
- Third-party tool
What are the standard requirement for using the tools?
Tool Inventory and Classification

- Inventory of all the tools involved
- Document use cases of the tools
- Classify the tools for their use cases

*Generally, most tools will not have to be qualified*
II. Tool Qualification

Qualification methods for TCL3

Qualification methods for TCL2

Qualification not required

Increasing qualification requirements

ASIL

TCL 3

TCL 2

TCL 1

High

Medium

TI 1

TI 2

Tool use cases

Tool impact

Tool error detection

Tool confidence level

UC 1..n

I. Tool Classification
Reference Workflow

Model-Based Design for High-Integrity Software

Model standards checking (Simulink V&V)

Simulation / model testing (Simulink)
Model coverage (Simulink V&V)
Req. Mgmt. Int. (Simulink V&V)

Module and integration testing at the model level

Equivalence testing

Prevention of unintended functionality

Textual requirements

Executable specification

Modeling

Simulink/Stateflow

Model used for production code generation

Generated C code

Object code

Modeling

Embedded Coder

Third-party tool

Code generation

Compilation and linking
How can MathWorks simplify meeting the standard?
MathWorks Certification Kits

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

IEC Certification Kit

DO Qualification Kit
MathWorks Certification Kits

① Supports tool qualification
② Streamlines standard compliant development of embedded systems

www.mathworks.com/products/iec-61508/
www.mathworks.com/products/do-178/
Independent Assessment by TÜV SÜD

Example

Certificate

Assessment Report
Qualification of Model-Based Design Tools

- Embedded Coder, Simulink Design Verifier, Simulink Verification and Validation, and Polyspace are pre-qualified for all ASILs according to ISO 26262

Note: Embedded Coder, Simulink Design Verifier, Simulink Verification and Validation, and Polyspace products were not developed using certified processes.
Annotated method tables with suggestions on how to use Model-Based Design processes and tools to apply the methods listed in the standard

### 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</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>Simulink Report Generator – Web View, System Design Description (SDD) report</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Embedded Coder – 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</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 Report Generator – Web View, System Design Description (SDD) report</td>
<td></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>1a</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. Dynamic test vector charts can be used to create closed-loop, reactive model tests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stateflow – Dynamic test vector charts</td>
<td></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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reference Workflow - **Compliance Demonstration Templates**

**Checklist 1: Design Verification**

<table>
<thead>
<tr>
<th>Technique / Method</th>
<th>Associated Requirements</th>
<th>Used / Used in a limited range?</th>
<th>Interpretation in this application, evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Model review</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Indication of all model components</td>
<td></td>
</tr>
<tr>
<td>2. Adherence to modeling standard</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Designation of a modeling standard</td>
<td></td>
</tr>
<tr>
<td>3.度量对模型的评估</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Ensure the modeling standard is accessible for use</td>
<td></td>
</tr>
<tr>
<td>4. Testing activities</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Evidence for using static analysis</td>
<td></td>
</tr>
</tbody>
</table>

**Checklist 2: Code Verification**

<table>
<thead>
<tr>
<th>Technique / Method</th>
<th>Relevant Requirements</th>
<th>Used / Used in a limited range?</th>
<th>Interpretation in this application, evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Code review</td>
<td>Code review must be performed</td>
<td>Indication of the code</td>
<td></td>
</tr>
<tr>
<td>2. Adherence to modeling standard</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Designation of a modeling standard</td>
<td></td>
</tr>
<tr>
<td>3.度量对代码的评估</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Ensure the modeling standard is accessible for use</td>
<td></td>
</tr>
<tr>
<td>4. Testing activities</td>
<td>Use Metrics and Static Analysis at the Model Level</td>
<td>Evidence for using static analysis</td>
<td></td>
</tr>
</tbody>
</table>

Module and integration testing at the model level

Review and static analysis at the model level

Prevention of unintended functionality

Equivalence testing

Textual requirements

Executable specification

Modeling

Model used for production code generation

Generated C code

Object code

Code generation

Compilation and linking
Test Cases for Tool Qualification/Validation

- Test automation provided to validate tools within the project
Process Deployment Advisory Service

MathWorks Consulting services to quickly adopt Model-Based Design for standards

- Objectives
  - Identify gaps in current processes
  - Provide a roadmap to an optimized standard compliant process
  - Assist with deployment of that roadmap
  - Educate on the standard
  - Help with software tool qualification

www.mathworks.com/services/consulting/areas/iso26262-process-deployment.html
Summary

- **How can MathWorks tools help develop high-integrity systems?**
  Model-Based Design offers many Verification and Validation methods.
  Tool automation help meet the process documentation requirements.

- **What are the standard requirements for using the tools?**
  The Reference Workflow is guiding through the development process.

- **How can MathWorks simplify meeting the standard?**
  Certification/Qualification Kits and Consulting expertise eases the Tool Qualification process.