MATLAB EXPO 2019

Planning Simulink Model Architecture and Modeling Patterns for ISO 26262 Compliance

Dave Hoadley
ISO 26262 “Road Vehicles - Functional Safety”

- ISO 26262 is a functional safety standard for road vehicles

- MathWorks has seen an increased interest in ISO 26262 compliant workflows
  - Increase in System Complexity
  - Demand from ADAS and AD related applications

- ISO 26262 facilitates modern software engineering concepts
Challenges with ISO 26262

- Do I have an ISO 26262 compliant workflow?
- How to efficiently reach unit testing coverage criteria?
- How to achieve Freedom from Interference?
- Can we use AUTOSAR and meet ISO 26262 at the same time?
- Is Simulink suitable for use for ISO 26262?
ISO 26262-6:2018 notes Simulink and Stateflow as Suitable for Software Architecture, Design and as basis for Code Generation

Table 5 — Notations for software unit design

<table>
<thead>
<tr>
<th>Notations</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural language(^a)</td>
<td>++</td>
</tr>
<tr>
<td>Informal notations</td>
<td>++</td>
</tr>
<tr>
<td>Semi-formal notations(^b)</td>
<td>+</td>
</tr>
<tr>
<td>Formal notations</td>
<td>+</td>
</tr>
</tbody>
</table>

\(^a\) Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or provide an explanation and rationale for decisions captured in the notations.

EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

\(^b\) Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design are applied to the model which serves as the basis for the code generation.
LG Chem Develops ISO 26262 ASIL C AUTOSAR-compliant Software for a Hybrid Vehicle Battery Management System for the Volvo XC90

“Model-Based Design enables us to increase component reuse, reduce manual coding, improve communication with our customers, and ultimately deliver higher-quality BMS in less time.”

- Won Tae Joe, LG Chem
MathWorks Support for ISO 26262 Certification Kit

- Applicable Model-Based Design Tools and Processes

Mapping between ISO requirements to Model-Based Design toolchain
MathWorks Support for ISO 26262 Certification Kit

- Applicable Model-Based Design Tools and Processes

- Model-Based Design reference workflow

- Overall MBD workflow
- Tools/Features:
  - Embedded Coder
  - Simulink Check
  - Simulink Coverage
  - Simulink Test
  - Polyspace Bug Finder
  - Polyspace Code Prover
MathWorks Support for ISO 26262 Certification Kit

- Applicable Model-Based Design Tools and Processes
- Model-Based Design reference workflow
- Tool Qualification Package
  - Software Tool Criteria Evaluation Report
  - Software Tool Qualification Report

IEC Certification Kit
Simulink® Test™ ISO 26262 Tool Qualification Package

Tool Confidence Level determination + Other qualification artifacts

<table>
<thead>
<tr>
<th>Potential Malfunction or Erroneous Output</th>
<th>Use Cases</th>
<th>TI</th>
<th>Justification for TI</th>
<th>Prevention / Detection Measures</th>
<th>TD</th>
<th>Justification for TD</th>
<th>TCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Simulink] Incorrect behavior of test harness</td>
<td>[Simulink] Incorrect behavior of [Simulink] Incorrect behavior of [Simulink] Incorrect behavior of [Simulink] Incorrect behavior of test harness</td>
<td>T12</td>
<td>Incorrect behavior of test harness could prevent errors in an object under test from being detected.</td>
<td>[Simulink] Requirements-based testing</td>
<td>TD1</td>
<td>The test cases and expected results are derived from requirements independent of the model under test and the test environment. The independence provides a high degree of confidence that errors will be detected using the actual results from the model under test in the test environment</td>
<td>TCL1</td>
</tr>
<tr>
<td>[Simulink] Incorrect run of test procedure</td>
<td>[Simulink] Incorrect run of test procedure</td>
<td>T12</td>
<td>Incorrect run of test procedure could prevent errors in an object under test from being detected.</td>
<td>[Simulink] Requirements-based testing</td>
<td>TD1</td>
<td>Requirements-based testing will detect incorrect run of test procedure, see TD justification for [Simulink] Incorrect run of test procedure</td>
<td>TCL1</td>
</tr>
<tr>
<td>[Simulink] Incorrect assessment of test results - passed test indicated as failed</td>
<td>[Simulink] Incorrect assessment of test results - passed test indicated as failed</td>
<td>T11</td>
<td>Nuance only, failed tests have to be manually reviewed and explained by user</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TCL3</td>
</tr>
<tr>
<td>T12</td>
<td>None</td>
<td>TD9</td>
<td>-</td>
<td>TCL1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Modeling Best Practices for ISO 26262

- Architecture
- Signal Routing
- Data Definition
- Code Generation Configuration

(Excerpts from our white paper
  – Please request www.mathworks.com/services/consulting/contact.html)
Use Model Metrics to Monitor Unit Complexity

**Architecture**

- **Issues:**
  - Model verification gets increasingly difficult
  - Unable to efficiently achieve unit coverage

- **Best Practice:**
  - Monitor complexity metrics
    - Interfaces
    - Reusable libraries
    - Cyclomatic complexity ($\leq 30$)*
    - Number of elements ($< 500$)*
    - Style and standards conformance

- **Reference:**
  - *Paper: Model Quality Objectives*
    - Authors: Jérôme Bouquet(Renault), Stéphane Faure(Valeo), Florent Fève(Valeo), Ursula Garcia(Bosch), François Guérin(MathWorks), Thierry Hubert(PSA), Florian Levy(Renault), Stéphane Louvet(Bosch), Patrick Munier(MathWorks), Pierre-Nicolas Paton(Delphi), Alain Spiewek(Delphi), and Yves Touzeau(Renault)
Use Model Reference for Unit Level Model
Simulink Architecture

- **Issues:**
  - Poor modularity of algorithm (reuse)
  - Unable to perform unit level testing
  - Configuration Management difficulties
  - Unable to achieve Freedom from Interference

- **Best Practice**
  - Use Model Reference for unit level model
  - Group units to form functional hierarchy (features/components) with virtual Subsystems

Model block as Unit containers
Group Units with Subsystem block
Split ASIL and QM Levels at Top Level of Control Model

Simulink Architecture

- **Issues:**
  - Difficulty in achieving Freedom from Interference
  - Complexity in code integration

- **Best Practice:**
  - Code generation should be done at as high as level as possible.

<table>
<thead>
<tr>
<th>Model Hierarchy</th>
<th>Modeling Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top level (ASIL / QM)</td>
<td>Model Reference</td>
</tr>
<tr>
<td>Integration</td>
<td>Subsystem (multiple layers)</td>
</tr>
<tr>
<td>Unit</td>
<td>Model Reference</td>
</tr>
</tbody>
</table>

MATLAB EXPO 2019
Data Protection Between ASIL and QM Levels

Code Generation Configuration

- Issues:
  - How to provide signal protection between ASIL and QM functions?
Data Protection Between ASIL and QM Levels

Code Generation Configuration

- Issues:
  - How to provide signal protection between ASIL and QM functions?

- Best Practice
  - Use Get/Set storage class for signals between ASIL and QM levels

Get/Set Storage Class

MATLAB EXPO 2019
Eliminate Algorithm Content at Integration Level

Architecture

- Issues:
  - Complexity in integration level testing
  - Difficult tracing of requirement $\Leftrightarrow$ design $\Leftrightarrow$ test

- Best Practice:
  - Ensure only virtual blocks are at the integration level
  - Reference (MAAB/JMAAB): db_0143: Similar block types on the model levels
Use Different Name Token for Shared Utility

Code Generation Configuration

- Issues:
  - ASIL and QM level uses the same shared utility code
- Best Practice:
  - Configure Shared Utility Identifier
Design Bus Hierarchy
Signal Routing

- **Issues:**
  - Inefficient bus segmentation
  - Inconsistent bus grouping by developers
  - Modeling difficulty from splitting and recreating bus signals

- **Best Practice:**
  - Bus hierarchy should be designed as a function of ASIL levels, QM, and rates at a minimum.
Pass Only Used Signal into Unit

Signal Routing

- **Issues:**
  - Hundreds/Thousands input signals causing difficulties in verification flow

- **Best Practice:**
  - Use Bus Selector to send only used signals into Unit
  - Add additional virtual Subsystem to encapsulate the Bus manipulation before and after the unit
AUTOSAR Implications

- **AUTOSAR**
  - adds complexity due to additional tool ecosystem
  - but makes some things simpler
    - Get/Set function would be implemented using Send/Receiver port with RTE protection

- Best Practices discussed are consistent with our AUTOSAR Blockset

- Reference Workflow shown in IEC Certification Kit supports AUTOSAR
  - Simulation
  - Code generation
  - Verification
MathWorks Support

ISO 26262 Consulting Services

- Process establishment
  - Development Processes
  - Verification process
  - Gap analysis

- Tool qualification support
  - Analyze customer specific tools
  - Provide guidance on tool qualification activities
Summary

- Modeling Best Practice for ISO 26262
  - Use Model Reference for Unit Level Model
  - Split ASIL and QM Levels at Top Level of Model
  - Eliminate Algorithm Content at Integration Level
  - Use Model Metrics to Monitor Unit Complexity
  - Pass Only Used Signal into Unit
  - Design Bus Hierarchy

- Further information?
  - Please see mathworks.com/services/consulting/proven-solutions/iso26262.html
  - Contact me dhoadley@mathworks.com
  - Stop by the ISO 26262 table