Model-Based Design for ISO 26262 Applications

April 2010
Agenda

- Introduction
  - Certification, Standards, and Compliance Demonstration
  - ISO 26262 & Qualification of Software Tools
- Verification & Validation Workflow
  - V&V at Model Level
  - V&V at Code Level
- Conclusion
High-Integrity Applications

Software-based systems that are designed and maintained so that they have a high probability of carrying out their intended function.

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

- **Aerospace Standards**
  - DO-178B (= JAA EUROCAE ED-12B)
  - DO-254

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

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

* Used e.g. in automotive and industrial automation
IEC 61508 Derivative Standards

Generic safety standard

IEC 61508
1998-2000

EN 5012x

Derivative standards

IEC 61511

IEC 61513

IEC 60601

ISO/CD 26262
2008

ISO/DIS 26262
2009
ISO 26262
Road vehicles - Functional safety

- Draft International standard ISO/DIS 26262, published 2009
  - References to modern software engineering paradigms such as Model-Based Design and code generation
  - IEC 61508 Derivative
- Sector specific safety standard for automotive
- Four Automotive Safety Integrity Levels (ASILs) A…D
- Already used by some automotive companies on a voluntary basis
- Tool Qualification used when use of software tool simplifies or automates activities and tasks required for the development of a safety-related item or element by ISO 26262
ISO 26262 Tool Qualification of Real-Time Workshop Embedded Coder

- Real-Time Workshop Embedded Coder has been pre-qualified by TÜV SÜD for all ASILs according to ISO/DIS 26262

- MathWorks created tool qualification artifacts were assessed by TÜV SÜD; Qualification assessment is documented in the certificate report

- Tool qualification can be claimed by customizing the tool qualification package and referencing the certificate/certification report

Includes templates for:
- SW Tool Qualification Plan
- SW Tool Documentation
- SW Tool Classification Analysis
- SW Tool Qualification Report
ISO 26262 Tool Qualification

- **Tool functionality and usage**
- **Tool impact**
- **Tool error detection**
- **Tool confidence level**

- **TI 0**
  - **TD 1**: High
    - **TCL 1**: No additional qualification methods required
  - **TD 0**: No
    - **TCL 0**: Increasing qualification requirements

- **TI 1**
  - **TD 2**: Medium
    - **TCL 2**: No additional qualification methods required
  - **TD 3**: Low
    - **TCL 3**: Additional qualification methods required
  - **TD 4**: No
    - **TCL 4**: Increasing qualification requirements
ISO 26262 Tool Qualification of Real-Time Workshop Embedded Coder

<table>
<thead>
<tr>
<th>Tool functionality and usage</th>
<th>Tool impact</th>
<th>Tool error detection</th>
<th>Tool confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI 0</td>
<td>TD 1</td>
<td>No</td>
<td>TCL 1</td>
</tr>
<tr>
<td></td>
<td>TI 1</td>
<td>Low</td>
<td>TCL 2</td>
</tr>
<tr>
<td></td>
<td>TD 2</td>
<td>Medium</td>
<td>TCL 3</td>
</tr>
<tr>
<td></td>
<td>TD 3</td>
<td>No</td>
<td>TCL 4</td>
</tr>
<tr>
<td></td>
<td>TD 4</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Increasing qualification requirements

- Complete verification and validation workflow
- Suitable subset of verification and validation workflow
- Additional qualification methods required
- No additional qualification methods required
TÜV Certificate for Real-Time Workshop Embedded Coder

- Certificate based on:
  - Focused audit by TÜV of MathWorks development and quality assurance processes for Real-Time Workshop Embedded Coder
  - Review by TÜV of MathWorks document describing example workflow for verification and validation of models and generated code

- Certification includes:
  - Real-Time Workshop Embedded Coder (R2009a, R2009b, R2010a)
  - PolySpace Client / Server for C/C++ (R2009a+, R2009b, R2010a)

Note: Real-Time Workshop Embedded Coder and PolySpace products for C/C++ were not developed using certified processes.
Verification & Validation Workflow
Model Testing

- Model components should be functionally tested using systematically derived test vectors
  - Demonstrate that each model component performs its intended function and does not perform any unintended functions

- After component testing is completed, model integration testing should be performed with predefined test vectors
  - Demonstrate that each model components with their integrated subsystems interact correctly to perform their intended function and do not perform unintended functions.
Model Review and Static Analysis

- Model components should be reviewed
- Manual reviews should be supported by automated static analyses of the model
- Modeling guidelines should be used, and adherence with the guidelines should be assessed
Code Testing

- The workflow
  - Use translation validation through systematic testing
  - Demonstrates that the execution semantics of the model is being preserved during code generation, compilation, and linking

- Numerical Equivalence Testing
  - Equivalence Test Vector Generation
  - Equivalence Test Execution
  - Signal Comparison
Prevention of Unintended Functionality

- Traceability Review
  - Traceability analysis of the generated C source ensures that all parts of this code can be traced back to the model used for production code generation
  - The generated code is subjected to a limited review that exclusively focuses on traceability aspects
  - Non-traceable code shall be assessed
Prevention of Unintended Functionality

- Model versus Code Coverage Comparison
  - Structural coverage metrics should be used on the model and code level respectively
  - Decision coverage at the model level and branch coverage (C1) at the code level can be used in combination
  - Discrepancies between model and code coverage shall be assessed.

- If the code coverage achieved is less than the model coverage, unintended functionality could have been introduced
Example V&V Workflow with MathWorks Products

Model Advisor, Modeling standards checking
Simulation (model testing), Model coverage, RMI
Module and integration testing at the model level
Review and static analysis at the model level

PIL testing using Embedded IDE Links
Real-Time Workshop Embedded Coder traceability report or Model vs. code coverage comparison
Equivalence testing
Prevention of unintended functionality

Model used for production code generation
Generated C code
Object code

Textual requirements
Executable specification

Modeling

Simulink / Stateflow / Simulink Fixed Point
Real-Time Workshop Embedded Coder

19
Conclusion

- Model-Based Design is used for many systems, including high-integrity applications
- An example V&V workflow and tools were described based on IEC 61508 and ISO 26262
  - But, any application can benefit from rigorous V&V
- MathWorks offers variety of V&V and PCG workshops, master classes, and additional support materials
  - Visit our website or contact us for details
The MathWorks

Change the world by

Accelerating the pace
of discovery, innovation, development, and learning

in engineering and science