ISO 26262 Conforming Tool Qualification for Modular Tools
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- Modular Tool Qualification
- Example Tool Architecture
- Summary
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- Modular Tool Qualification
- Example Tool Architecture
- Summary
Motivation & Goal

Motivation:

- Many modular tools (Eclipse-based) in automotive industry
  Some of them are frameworks
- ISO 26262 requires tool confidence. Achieved by
  - Tool qualification or
  - Evidence that potential errors would not affect safety

Goal:

- Development modular method to provide tool confidence
  - Qualification kit
  - Information ("Safety Manual") to work with the tool in a safe way
ISO 26262-8, Chapter 11: „Confidence in the use of software tools“

Classification in „Tool Confidence Level (TCL)“

Tool Impact (TI)
- TI1: no impact => Tool is TCL1
- TI2: some impact
  - Tool Error Detection/prevention probability (TD)
  - TD1: high confidence => Tool is TCL1
  - TD2: medium confidence => Tool is TCL2
  - TD3: low/unknown confidence => Tool is TCL3

Justification? Dokumentation? Confirmation Review!

Table 2 — Qualification of software tools classified TCL3

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased confidence from use according to 11.4.7</td>
<td>++</td>
</tr>
<tr>
<td>Evaluation of the tool development process according to 11.4.8</td>
<td>++</td>
</tr>
<tr>
<td>Validation of the software tool according to 11.4.9</td>
<td>+</td>
</tr>
<tr>
<td>Development in compliance with a safety standard³</td>
<td>+</td>
</tr>
</tbody>
</table>
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- Modular Tool Qualification
- Example Tool Architecture
- Summary
Tool Chain Analysis: Method & Tool

- Computes automatically the TCL of tools and checks qualifications for ASIL
- Based on a simple but formal model of tools, use cases, errors, checks, etc.
- Many helpful features
  - Report generation
  - Assumptions modeling
  - Model validation (no empty descriptions, no use case without errors,..)
  - Modularity and reuse concepts
  - Excel interface (check list generation,..)
- Supports systematic error analysis
- Prototype developed from Validas AG in research project
- Bases on Eclipse, EMF, graphviz and docx4j
- Download from http://www.validas.de/TCA152.zip
Result: Tool Confidence Levels

- Tool Chain: 39 tools
- Only one tool with TCL 2
- Improvement: Reviewed -> Final
  - Added wins & quick wins (Process changes)
- Input for tool qualification
  - Critical features
  - Undetected errors
- ISO 26262 compliant
- Better than
  - Defaults (from literature)
  - IEC / DO classifications
Why Can we Trust?

- Independent/external analysis performed from experts (Validas)
- TCLs are computed with a calculus (based on a formal model)
- Systematic error model for tools applied (black-box & white-box)
- Tool chain model and error model have been reviewed
- Analysis was tool supported (Tool Chain Analyzer)
- Detailed report (14 pages per tool in the example) explaining every error and every check/restriction
Modeling Method / Process for TCA

1. Planning:
   1. Build a formal model of tool usage based on tool and process information
   2. Validate the model (Review, Checklists)

2. Tool Evaluation
   1. Systematically build an error model (black box / glass box)
   2. Model detection and prevention (including assumptions)
   3. Validate **assumptions** and error/detection/prevention model
   4. Compute the TCL (with/without assumptions)
   5. Generate tool evaluation report

3. Tool Qualification
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- **Modular Tool Qualification**
- Example Tool Architecture
- Summary
Modular Tool Qualification

- **Goal:** use tool modules with confidence (in many tools)
- **Similar to “Safety Element out of Context” (SEooC)**
  - Elements: “Tool Modules”, e.g. Libraries / Plugins
  - Assumptions on the usage of the modules in the context (tool)
- **A tool module description should contain**
  - List of available functions e.g. actions, methods,..
  - Input and output elements of the functions e.g. model elements, parameters
  - Potential errors in functions
  - Proposals for error mitigations (detection/prevention) with probabilities
  - Set of tests (for each function) that shows the correctness
    - Eventually restricted to a subset of models
    - Eventually restricted to the absence of some potential error classes
  - Documentation (including known bugs and used modules)

- **Using a module in a tool requires**
  - To check the assumptions of the module
    - Mitigate them in the tool or
    - Add them to the assumptions of the tool (“Safety Manual”)
Module Qualification Kits

Qualification kit consists of

- Test automation e.g. JUnit, Scripts for verifying the results,..
- Tests with expected results for
  - Functions and
  - Inputs
- Should cover the relevant code (in the module)
- Should mention the requirements of used plugins (functions & inputs)
- Safety Manual to avoid/detect potential tool errors

A qualification kit is an additional product for every module
Assumption Examples

Depend on tool use cases and potential error mitigations

- **User Requirements ("Safety Manual")**
  - Reopen saved models to verify persistency
  - React on warnings
  - Check log files
  - Make a training/tutorial
  - Execute an installation test
  - Verify environment variables & resources
  - Apply redundancy
  - Review results
  - Check known bugs

- **Developer Requirements**
  - Catch exceptions
  - Check return codes, parameters values
  - Consider programming rules
  - Provide qualification kit
Vision: Eclipse Qualification Process

- Determine requirements for every used plugin:
  - Used functions, inputs
  - Check assumptions of plugins
- Show that assumptions of modules are satisfied
- Collect assumptions for the tool (user)
- Apply qualification kits for every plugin according to the identified requirements
- Measure code coverage during qualification

- Measure code coverage of the application during usage
- Compare it with sum of all qualification coverage
  - Application coverage ⊆ qualification coverage ⇒ OK
  - Application coverage ⊄ qualification coverage ⇒ NOK
    - Refine requirements & improve qualification kits (coverage)
    - Manually check correctness
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- Modular Tool Qualification
- Example Tool Architecture
- Summary
Tool Chain Analyzer: Developer View (White-Box)

- Eclipse structure: Plugin Projects (RCP)
  
  - Many dependencies to Eclipse plugins
  - Furthermore
    - Integration of graphviz
    - Excel-Interface (Read & Write)
    - Word Generator
Content

- Motivation: ISO 26262
- Tool Chain Analysis
- Modular Tool Qualification
- Example Tool Architecture
- Summary
Summary

- ISO 26262 requires to classify (evaluate) all and qualify some tools
- Tool chain analysis
  - Can reduce qualification needs
  - Determines critical use cases and potential errors of tools
- Modular Qualification Approach (like SEooC)
- Tool chain analyzer as example for discussion of qualification methods
Thank You!
Possible Next Steps

- Analyze more Eclipse architectures & middleware
- Build examples for modular qualification kits
- Build reference architecture, based on EMF?
- Build qualification infrastructure for EMF?
  - Test automatization
  - Coverage analysis
    - Code coverage
    - Model coverage