Software Detailed Design for Model-Based Development – Obligatory or Superfluous?
Outline

- Introduction
- Necessity of Software Detailed Design
- Requirements on Detailed Design
- Challenges Model Based Development and Detailed Design
Introduction VW Software Quality Assurance

- VW Group Supplier Quality Assurance Electric/Electronics
- Responsible for quality assurance of VW group suppliers
  - Potential analysis before nomination
  - Full ASPICE assessments for focus projects
  - Technical revisions and supplier improvement program support
- 10 ASPICE assessors (+ colleagues at AUDI, MAN, Porsche, CARMEQ)
- Approx. 100 Software assessments/audits per year
- Focus on critical Software/ECU-projects for series with tier 1 suppliers
- Specification of VW Group Basic Software Requirements
**ASPICE and ISO 26262**

- requirements for development processes and quality criteria for automotive system and software development
- in general not specific to any programming language, but defined with the mindset of classic c-code implementation.
Model based development for series projects

- used mostly for functional application software, e.g. engine control, steering, suspension, climate control for series ECU development
- fast growing in new projects
- job split - functional modelling at OEM and industrialization / code generation at supplier

use of model based development in series projects*

- 70% with code generation
- 25% as design tool
- 5% w/o model based

*internal survey, projects of VW group suppliers 2013-2016
Software Design Understanding

Architectural Design  High-Level Design

Low-Level Design  Detailed Design

Unit requirements specification

Software Design Description (IEEE 1016)

Design

software unit design specifications  (ISO 26262)
Why Software Design?

- Modularity
- Reuseability
- Modifiability
- Testability
- Analyzability

ISO / IEC 25010:2011
Automotive SPICE® v3.0 and implementation model

Figure D.3 — Element, Component, Unit, and Item
Requirements from Automotive SPICE® v3.0 (extract)

As a result of successful implementation of process SWE.3 “Software Detailed Design and Unit Construction”:

- A detailed design is developed that describes software units.
- Interfaces of each software unit are defined.
- The dynamic behavior of the software units is defined.
- Consistency and bidirectional traceability are established between:
  - Software requirements and software units.
  - Software architectural design and software detailed design.
  - Software detailed design and software units.
- Software units defined by the software detailed design are produced.
Thesis:
„My model is my detailed design!“
Why a model may not be a Detailed Design?

Why a model **may not be** a Detailed Design (typical challenges):

- Missing design decisions - no answer why something is implemented that way (ISO 25010: functional suitability, maintainability, portability, etc.) (SWE3.BP4)

- No distinction between architectural and detailed design (sometimes)

- No distinction between specification and implementation model (ISO 26262)

- No specification of non-functional requirements (e.g. RAM, ROM usage)
Why a model may be a Detailed Design?

Why a model may be a detailed design (typical issues):

- Describes structural break down and allows definition of smallest unit (e.g. submodel), which can be run dedicated.

- Consistency of interfaces is ensured inside of the model by use of data dictionary (SWE3.BP2, SWE3.BP6).

- Visualization of dataflow supported by graphical representation directly in the model (SWE3.BP1).

- Description of dynamic behavior (SWE3.BP3) by using synchronization elements, internal scheduler and sample timing definition.
Challenge – find the optimum!

- suitable extent of detailed design, no unnecessary overlap with model
So, how do YOU find the “optimum”?

And still achieve Automotive SPICE Compliance?
Automotive SPICE
SWE.3 Software Detailed Design - Typical Challenges

- All development activities must add value to the model.

- Activities’ effort has to be sustainable (and realistic) along the whole project lifecycle.

- Find the optimum and avoid duplicate work!

- Since end of 2014 we have been working on this topic together with Volkswagen to define a solution.
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)

BP1: Develop software detailed design.
BP2: Define interfaces of software units.
BP3: Describe dynamic behavior.
BP4: Evaluate software detailed design.
BP5: Establish bidirectional traceability.
BP6: Ensure consistency.
BP7: Communicate agreed software detailed design.
BP8: Develop software units.

SWE 1: Software Requirements Analysis
SWE 2: Software Architectural Design
SWE 3: Software Detailed Design

Traceability
Iterative Process

SWE.3 - Base Practices
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)

SWE 1
Software Requirements Analysis
RM Tool

SWE 2
Software Architectural Design
RM Tool

SWE 3
Software Detailed Design

BP1: Develop software detailed design.
BP2: Define interfaces of software units.
BP3: Describe dynamic behavior.
BP4: Evaluate software detailed design.
BP5: Establish bidirectional traceability.
BP6: Ensure consistency.
BP7: Communicate agreed software detailed design.
BP8: Develop software units.

SWE.3 - Base Practices

Traceability

Iterative Process
Automotive SPICE
SWE.3 BP1: Develop software detailed design.

- Develop a detailed design for each software component
  - Use Simulink, Stateflow and toolboxes.
  - Involve functional and non-functional requirements.

- Develop Specification Model
  - Assess the impact of requirements and design changes through simulation.

- Derive an Implementation Model
  - Fulfills all automotive relevant Model-Advisor checks (e.g. MISRA C, ISO 26262, MAAB, …).
  - Is ready for production code generation (e.g. uses Fixed-Point Data types, …).

- Manage and document design decisions
  - Directly in the model or (if applicable) in the RM Tool.
  - Establish bidirectional linking between relevant blocks and satisfied requirements.
Model-Based Design & Automotive SPICE

SWE.3 BP1: Develop software detailed design (2)

Document Design Decisions textually in Model (or in RM tool)

Bidirectional traceability with Software Requirements

Generate Software Design Document
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)

- Software Requirements Analysis
- Software Architectural Design
- Software Detailed Design

SWE 1

SWE 2

SWE 3

BP1: Develop software detailed design.
BP2: Define interfaces of software units.
BP3: Describe dynamic behavior.
BP4: Evaluate software detailed design.
BP5: Establish bidirectional traceability.
BP6: Ensure consistency.
BP7: Communicate agreed software detailed design.
BP8: Develop software units.

Traceability

SWE.3 - Base Practices

Iterative Process
Model-Based Design & Automotive SPICE

BP4: Evaluate software detailed design.

- Review models, design decisions and requirements linking.
- Execute test cases and model coverage analysis.
- Assess size and complexity of software units with model-metrics.
- Assess conformance to standards at model level (ISO 26262, MISRA, etc.).
  - Justify non-conformities through model annotations.
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)

- SWE 1: Software Requirements Analysis
  - RM Tool

- SWE 2: Software Architectural Design
  - RM Tool

- SWE 3: Software Detailed Design
  - BP1: Develop software detailed design.
  - BP2: Define interfaces of software units.
  - BP3: Describe dynamic behavior.
  - BP4: Evaluate software detailed design.
  - BP5: Establish bidirectional traceability.
  - BP6: Ensure consistency.
  - BP7: Communicate agreed software detailed design.
  - BP8: Develop software units.

Traceability
Iterative Process
Model-Based Design & Automotive SPICE

BP5: Establish bidirectional traceability.

- Establish bidirectional traceability between software requirements and the software detailed design.

- Bidirectional traceability
  - Requirements
  - Design decisions
  - Model

- These can include:
  - Parametrization and interface requirements on a high-level of abstraction
  - Specific requirements, e.g. for a start-up task

- Ensure traceability through traceability report or traceability matrix
Model-Based Design & Automotive SPICE
Software Engineering Process Group (SWE)

SWE 1
Software Requirements Analysis

SWE 2
Software Architectural Design

SWE 3
Software Detailed Design

BP1: Develop software detailed design.
BP2: Define interfaces of software units.
BP3: Describe dynamic behavior.
BP4: Evaluate software detailed design.
BP5: Establish bidirectional traceability.
BP6: Ensure consistency.
BP7: Communicate agreed software detailed design.
BP8: Develop software units.

SWE 3 - Base Practices
Model-Based Design & Automotive SPICE

BP6: Ensure consistency.

- Ensure consistency between software requirements and software units.

- Ensure consistency between the software detailed design and software units.

- Consistency check
  - Missing documents
  - Invalid links
  - Modified requirements
  - Unidirectional links

Traceability Report

Requirements Consistency Check
Model-Based Design & Automotive SPICE

Software Engineering Process Group (SWE)

SWE 1
Software Requirements Analysis

SWE 2
Software Architectural Design

SWE 3
Software Detailed Design

RM Tool

Traceability

BP1: Develop software detailed design.
BP2: Define interfaces of software units.
BP3: Describe dynamic behavior.
BP4: Evaluate software detailed design.
BP5: Establish bidirectional traceability.
BP6: Ensure consistency.
BP7: Communicate agreed software detailed design.
BP8: Develop software units.

SWE 3 - Base Practices
Model-Based Design & Automotive SPICE

BP8: Develop software units.

- Code generation for MBD
  - Implementation model (consideration of all production code parameters as fixed-point arithmetic, etc.)
  - Coder Configuration
    - Target hardware
    - Resources optimization
    - Function prototypes and variables allocation

- Automatic report with bidirectional traceability
  - Requirements
  - Design Decisions
  - Model
  - Code
Thesis: “My model is my detailed design!”

Model = Detailed Design, if fulfills:
- Design Decisions documentation
- Interfaces definition
- Dynamic behavior description
- Design review
- Bidirectional requirements traceability
- Consistency check

Software Units
- Implementation model
- Code generation
- Model has much more value than a static drawing

Result of collaboration:
- Guideline for efficient ASPIE-conform Model-Based Design development.
- MathWorks Expertise for customer support.

<table>
<thead>
<tr>
<th>Base Practice</th>
<th>Measure</th>
<th>Recommended Tool or Functionality</th>
<th>Anti</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP1: Develop software detailed design.</td>
<td>+ Use Model Reference Blocks, Atomic Subsystems, Function-Call Subsystems or Simulink functions to achieve functional decomposition into testable units</td>
<td>Simulink®</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>+ Use Interface view to assess signal flow and decomposition</td>
<td>Stateflow®</td>
<td>Port and Desc</td>
</tr>
<tr>
<td></td>
<td>+ Adhere to MAAB Modeling Standards, e.g. avoid mixing basic blocks and subsystems</td>
<td>Simulink Verification and Validation® - Model Advisor MAAB Checks</td>
<td></td>
</tr>
<tr>
<td>BP2: Define interfaces of software units.</td>
<td>+ Use unambiguous names for Signals and Ports</td>
<td>Simulink®</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>+ Definition of complex interfaces with multiple signals through non-virtual buses (Bus Objects)</td>
<td>Simulink® - Data Dictionary</td>
<td>Port and Desc</td>
</tr>
<tr>
<td></td>
<td>+ Link Interface Requirements to Data Dictionary Elements</td>
<td>Simulink Verification and Validation® - Requirements</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion and Outlook

- VW Quality Goal: Improvement of “VW Group Basic Software Requirements” to consider a Model-Based Design development workflow
- VW and MathWorks successfully collaborated to craft a Model-Based Design process that is targeted towards reaching compliance with important industry quality standards
- MATLAB & Simulink provides a documented and traceable workflow aligned with the requirements of Automotive SPICE and ISO 26262-6
- Auditor community needs to adopt a common approach for assessments with Model-Based Design
- Definition of industry-wide standards for model quality criteria, e.g. complexity indicators and limits (like HIS-MISRA for C).