The Validation Suite Approach to Safety Qualification of Tools
Automotive Safety and Security 2008

BMW AG, Dr. Stefan-Alexander Schneider
PMFS IT Consulting, Pierre R. Mai
TÜV-Nord IFM, Dr. Tomislav Lovric

The Validation Suite Approach
Tool Qualification Introduction

Model-Based Software Development

**Advantages**

+ Efficient Software Design and Development
+ Early Validation of Executable Specifications
+ …

**Challenges**

– Testability of large Models (needs to be assured)
– Models too abstract for Target
– complex translation steps - details not known
– translation integrity level - not known
– resulting code quality - not known a priori
– …
Safety-Related Model-Based Software Development:

Due Diligence of Developer to assure:
– Language is suitable for safety
– Developed models are suitable for safety (diagnosis)
– Translation steps are suitable for safety
– Translation Results are suitable
…

Systematic validation of translator suitability needed (see 61508/26262…) - tool validation difficult for individual projects

-> Do this reproducibly and thoroughly: Validation Suite Approach

The Validation Suite Approach

Agenda

1. Tool Qualification Introduction
2. Validation Suite Approach to safety Qualification of Tools
3. Validation Suite Requirements
4. Assessment steps of the Validation Suite
5. Experience with validation suite and tool validation
6. Outlook
Basic Idea for Validation Suite

EW<sub>t</sub>: Transformation of a model m by development tool EW to a binary B<sub>t</sub>

S<sub>m</sub>: Set of stimuli for a model m ∈ M (= suitable models (i.e. EW Inputs))

E<sub>m</sub>: Set of possible results e<sub>m</sub> for evaluations of model m

eval<sub>x</sub>: evaluation function for an execution environment x

To be shown by VS for all Models and Stimuli:

distance d(e<sub>m</sub>, e<sub>m'</sub>) = 0 (∀ e<sub>m</sub> ∈ E<sub>m</sub>; e<sub>m'</sub> ∈ E<sub>m'</sub>)

- Function d defines “Correctness” (here: Bit-Equality of all results).
- “Teststrategy” defines required models and stimuli for the VS

Central to BMW for VS Strategy:

1. Automated testing and result evaluation of many test
   - database with various models and stimuli
   - powerful automated test environment
   - efficient evaluation of result correctness
   - validity of validation results (incl. “golden results”)

2. State-Of-The-Art validation strategy:
   - evaluation of safety standards requirements
   - independent derivation: requirements for tools
   - independent derivation: requirements for VS
   - include tool requirements into VS
   - implement VS requirements
The Validation Suite Approach
Approach to safety Qualification of Tools

Basic structure of VS-TL-0001:

State of the art requirements for VS

TVS-Input:
EW-TL-xxxx

TVS User Manual (with Checklists)
TVS Maintenance Manual

Review-Part
Review Manual

Test-Part
Test cases (Models, Stimuli, Configurations)
Test scripts
Test Environment

TVS-Operating Environment
HW (Evalboard, PC), OS, Tools

TVS-Out:
Qualification Report

Define TVS-Input early in Project

Fix basic configuration of tool to be validated:

1. Operating Environment,
2. Code generator incl. relevant Hooks,
3. Compiler with Source files (if req.)
4. Development guidelines

E.g.
Windows, JAVA, MATLAB, TargetLink, Windriver, and Modelling Guidelines

Document “EW-Def” (Tool Definition)
The Validation Suite Approach
Approach to safety Qualification of Tools

**Basic structure of VS-TL-0001:**

State of the art requirements for VS

- TVS User Manual (with Checklists)
- TVS Maintenance Manual

**TVS-Input:**
- EW-TL-xxxx

**Review-Part**
- Review Manual

**Test-Part**
- Test cases (Models, Stimuli, Configurations)
- Test scripts
  - Test Environment

**TVS-Operating Environment**
- HW (Evalboard, PC), OS, Tools

**TVS-Out:**
- Qualification Report

---

**Test Environment:**
- thorough testing and analysis, certified in advance

- Model + Stimuli
- C-Code Generator
- C code

- Referenz Interpreter + Simulator
- References
  - MIL results

- PC + Target
  - SIL results
  - PIL results

- ..0100110

- Bit-by-Bit comparison
The Validation Suite Approach
Approach to safety Qualification of Tools

Test Environment:
– Derivation of EW specific characteristics

Abbreviations
TVS … TargetLink-VS
NOK … not OK
ERR … Abort during Codegeneration

Most critical scenario: Erroneous code generated, despite:
1. Conformance to Modelling-Guidelines
2. Confirmation by ES-Model-checkers
3. Model preparation via TL-Tool chain
4. Consideration of all notes from Log-file

The Validation Suite Approach
Agenda

1. Tool Qualification Introduction
2. Validation Suite Approach to safety Qualification of Tools
3. Validation Suite Requirements
4. Assessment steps of the Validation Suite
5. Experience with validation suite and tool validation
6. Outlook
The Validation Suite Approach. 
Validation Suite Requirements

Development Tools are complex,
• potentially error prone
• without evidence of integrity
• without „proven history”
• they need validation
  prior use in safety related projects
  (e.g. ISO CD 26262)
• No specific test criteria exist in standards

Task: Derive requirements
• top-down-filtering process
• derive and refinement existing requirements:
  currently: “VS-Anforderungen V2.0”

Result: 113 categorized requirements
• functional
• non-functional

General structure of requirements:
• … for VS for VS Main Test (= main part for test based VS)
• … for Pre Test of EW
  (= Annex; general suitability criteria for language and tool)
The Validation Suite Approach

Validation Suite Requirements

Structure of VS Requirements, with requirements on …

1. …Functionality of VS,
   e.g. test execution, logging of Configuration data
2. …Documentation and Report generation
   e.g. Generation of a tool Qualification Report
3. …Structure and Constituents of VS
   e.g. test database; different test categories with systematic tests test of language
4. …Validation of correct transformation of EW-Input
   e.g. for modular EW-Input
5. …Validation of intermediate code representations
   e.g. readability
6. …other Functional checks of the tool
   e.g. integration of other modules into the code
7. … Verification and Validation of VS itself
   e.g. structured and systematically planned V&V activities
8. … Qualität of VS itself
   e.g. Config. management; planning and execution of QA measures

Structure of Requirements for pre test of tools:
1. Support on and facilitation of suitable tool-Input by tool
2. Basic properties of intermediate code representations (C) and tool-Output
3. Other functional properties of the tool
4. Tool quality

The Validation Suite Approach

Agenda

1. Tool Qualification Introduction
2. Validation Suite Approach to safety Qualification of Tools
3. Validation Suite Requirements
4. Assessment steps of the Validation Suite
5. Experience with validation suite and tool validation
6. Outlook
The Validation Suite Approach
Assessment steps of the VS

**Structured stepwise procedure for TVS-Assessment**

<table>
<thead>
<tr>
<th>Test-Phase</th>
<th>Test object</th>
<th>Test aim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pre-Test (general suitability of tool)</td>
<td>EW Definition</td>
<td>e.g. tool allows suitable sw development (e.g. modularity)?</td>
</tr>
<tr>
<td></td>
<td>EW + Documentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EW-Developer (Audits)</td>
<td></td>
</tr>
<tr>
<td>2. Test of VS Test-Engine (TAU) (suitability of VS Testenvironment)</td>
<td>TAU + Documentation</td>
<td>e.g. TAU logs all relevant data?</td>
</tr>
<tr>
<td></td>
<td>TAU-Developer (Audits)</td>
<td></td>
</tr>
<tr>
<td>3. Test of VS Testcases (suitability of VS test case planning)</td>
<td>VS-Test strategy</td>
<td>Test categories for each language feature? ...; for modular progr. building blocks?</td>
</tr>
<tr>
<td></td>
<td>VS-Test plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VS-Test specification</td>
<td></td>
</tr>
<tr>
<td>4. Main VS Test (suitability of VS as a whole)</td>
<td>VS-Documentation</td>
<td>Test cases generation and executions for language features; VS-V&amp;V plan / Rep. ...</td>
</tr>
<tr>
<td></td>
<td>VS-Development, V&amp;V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VS-Developer Audits</td>
<td></td>
</tr>
</tbody>
</table>

Derivation of compliance level for each defined VS requirement by TN

Overall results of assessment activities:

**VS-TL-0001 complies satisfactory to the VS Requirements**

- The VS requirements
  - could be traced into the BMW TVS design
  - were refined for the TargetLink VS
  - Have been successfully implemented to a large extend

- The TVS has been successfully applied to a TargetLink / WindRiver based tool chain EW-TL-0001
- Several potential issues have been identified and suitably handled
- Nevertheless, overall the results have confirmed the suitability of the tool chain and maturity of the contained tools for intended use
The Validation Suite Approach

Agenda

1. Tool Qualification Introduction
2. Validation Suite Approach to safety Qualification of Tools
3. Validation Suite Requirements
4. Assessment steps of the Validation Suite
5. Experience with validation suite and tool validation
6. Outlook

Important Note:
The sample detected problems presented here

...are indications of
– a thorough testing approach by the TVS
– comprehensive test database and test engine
– with over 150.000.000 test result comparisons

The sample issues discovered
– are typical for complex software tools (>> 100.000 LOC).
– Similar issues can be found in most tools
– Other tools may show up many more issues …

The samples SHALL motivate tool validation

The examples SHALL NOT be used to
– question the maturity of the specific tested tool
– question the general suitability of the tested tool
The Validation Suite Approach.
Experience with tool validation
Sample problem with Subtraktion and Cast (1/3)

Model

1
single
InPort

2
InPort

Test Results

MIL

4
Constant

SIL/Ref

4
Constant

PIL

4
Constant

Conversion with
float to **unsigned** long and round to zero
(neg. input -4 convert to zero)
Instead of
float to **signed** long and round to zero?
The Validation Suite Approach.
Experience with tool validation
Problem with Subtraktion and Cast (3/3)

Result is reproducible …:

```c
#include <stdio.h>
typedef signed long sint32;
typedef unsigned long uint32;
typedef unsigned char uint8;
typedef float float32;

float32 int1;
uint8 in32;
sint32 out2;

void main () {
    int int_23 = 1;
    out = (sint32)((sint32)(-int_23) - (uint32)in32);
    out2 = (sint32)(-int_23 - in32);
    printf("%2d, %5d\n", out, out2);
}
```

No compiler failure…:

6.3.1.4 Real floating and integer

When a finite value of real floating type is converted to an integer type other than _Bool, the fractional part is discarded (i.e., the value is truncated toward zero). If the value of the integral part cannot be represented by the integer type, the behavior is undefined. 

The Validation Suite Approach.
Experience with tool validation
Wrong rounding of constants with MinMax

Calculation of int. maximum of three constants (-32768,-32768, 0.5) with three inputs block
Code generator generates
• at compile time the maximum for first part (-32768,-32768)
• a runtime calculation code for the second part

/* MinMax: P_Model/MinMax_3 */
P_Model/MinMax_3: folded operation max to constant value -32768
Variable 'Sa1_MinMax_3' replaced by 'Aux_S16' */

Aux_S16 = -32768 /* -32768. */;
if (0 /* was "0.5" in Model */ > Aux_S16) {
    /* Variable 'Sa1_MinMax_3' replaced by 'Aux_S16' */
    Aux_S16 = 1 /* was "0.5" in Model */;
}

/* TargetLink outport: P_Model/Out1__3 */
Sa1_Out1__3 = (sint32) Aux_S16;

Note: different rounding of 0.5 which may lead to issues in other context...
• 0 in if-statement
• 1 in assignment
shift operator e.g. on Uint16 has unexpected behaviour:

- if shifted more than 16 bits result is not always 0 (as might be expected and computed by reference interpreter)

Example:

46 << 204
= 57344 on SIL, MIL and PIL!

13 << 237
- = 40960 on SIL and MIL,
- = 0 on PIL!

Reason:
- Model-shift … is transformed directly to c-language-shift ... to asm-shift
- C-shift is undefined for shifts larger bit length
- Asm shift reg may use only 4 bits (lsb) for shift register …
- e.g. shift “<<17” may equal shift “<<1” (or ==0; result is fully undefined…)
Especially for **Model Based** development

- Tools have many **useful** functionalities … allow various design **improvements**
- Tools are **complex** (>>100,000 LOC)
- Tools are **combined differently** (use of different tool chains, tool usage guidelines)
- Tools “take over” design tasks previously performed “manually”
- Tools may **translate** “similar looking” blocks **differently**

**Tools need systematic validation** if used for safety

**Many parts of Validation may be automated by VS Approach**

**Use of a Validation Suite** is recommended in

- IEC 61508-3, 2nd Edition, note 3 on 7.4.4.11
- ISO WD 26262-8, 10.4.5 with tables 10.1 und 10.2.

**The documents “VS-Anforderungen”**

1. Herleitung und Leitfaden zur Anwendung
2. Anforderungen an eine Validierungssuite für Entwicklungs-werkzeuge

- are **systematically derived** from existing and upcoming standards (ISO 26262)
- have been already **used in practice** and improved
- **close a normative gap** for development **tool validation** suites
- the requirements define and summarise current state-of-the-art

- can be used by anyone to improve
- will be **further maintained**, translated, published and used
The Validation Suite Approach.

Summary / Outlook

BMW

• has successfully used the TVS
  • for qualification of TargetLink based Toolchains
  • for safety related series developments projects

• plans use of VS for further general qualification of tools
  • for further QA improvement reasons
  • for other tool chains (other targets, other tools)

• has gained positive experience from TVS use by
  • revealing potential difficulties otherwise not seen early
  • understanding potential and limits of tool chains in detail
  • improving guidelines for modelling and coding
  • improving quality of software and tools
  • Improving awareness by developers of potential tool limits

Is VS Approach applicable to your tool based project?

Improve it!

The Validation Suite Approach.

Finish: THANK YOU !

Thanks to all supporting parties, especially
• VALIDAS
• dSPACE
• WIND RIVER
• Audi Electronic Venture

Thank you very much for your attention

Contact
Dr. Stefan-Alexander Schneider, BMW AG
Stefan-alexander.schneider@bmw.de
+49 (0) 89 382 - 54566

Dr. Tomislav Lovric, TÜV NORD
TLovric@tuev-nord.de
+49 (0) 201 825 - 4101

Herr Pierre Mai, PMSF IT Consulting
pmai@pmsf.de
+49 (0) 8332 93669 – 13
Topics covered here include

- Sample validation suite and validated tools
- Samples of identified tool weaknesses during validation
- Adoption of validation suite to application project needs
- Maintenance of validated tool versions
- Necessary project validation activities
- Tool re-validation activities
Nutzen einer VS

IEC 61508

Projekt

Nicht EW

Sonstige Validierte Tools

Historie

Ascet Codegen
–2006 TN zertifiziert

Scade Codegen
–2007 TS Zertifiziert,

CoDeSys
–2007 TR Zertifiziert (?)

Mathworks RTW
–2008 TS Zertifiziert (prozess, prinzipielle eignung)
–2008 TR Validiert (vs mit einigen testfällen, systematik??)

BMW AVS + EW-0001 (TargetLink / WindRiver)
-2008 TN geprüft / bestätigt

Validierungsanforderungen
-2008 TN/BMW veröffentlicht