SCADE Combined Testing Process and DO-178B/C

DGLR Workshop, 4.10.2011

Christian Schrader
Agenda

- Introduction
- Software Development Process
- Software Verification Processes
- Verification of Verification Process
- Conclusion
SCADE Suite 6.2 Certified Software Factory

System Design

SCADE SYSTEM™

Control Software Design

Prototyping, Design, Verification, Qualified Code Generation

Display Software Design

Prototyping, Design, Verification, Qualified Code Generation

Application Lifecycle Mgt

Certification Plans, Requirements & Configuration Management, Documentation Generation

System Architecture, System Checks
SCADE Suite

Control Software Design

- Model Checking
- Formal Verification
- Debug & Simulation
- Rapid Prototyping & Executable Spec
- Model Coverage Analysis
- Time & Stack Analysis

SCADE Suite KCG
- C & Ada

RTOS Adaptors
- Object Code & Compiler Verification

DO-178B
- IEC 61508
- EN 50128
- Certification Kits

PROTOTYPE & DESIGN

GENERATE

VERIFY
The combined testing process is part of a global efficient process with major effort/time savings in:

- **Development activities:**
  - Model-based design
  - Qualifiable autocode generation

- **Verification activities:**
  - Automated checks
  - No need to review the source code

- **Testing:** the combined testing process
Agenda

- Introduction
- Software Development Process
- Software Verification Processes
- Verification of Verification Process
- Conclusion
Generic DO-178B Development Processes with SCADE

Traceability of requirements: SCADE DOORS™ Link

System Requirements Process

SW Requirements Process with SCADE Suite™

High-Level Requirements

SW Design Process with SCADE Suite™

Low-Level Requirements & Architecture

SW Coding Process with SCADE KCG

Source Code

Integrated Executable

SW Integration Process
Design Process

Requirements Process

Global Architecture Design

Derived requirements

Textual Design

SCADE Architecture Design

SCADE Module A LLR

SCADE Module B LLR

SCADE Module X LLR

Global SCADE Model

In master design document or annex

Master design document

SRS

Global SCADE Model

Traceability of requirements: SCADE DOORS™ Link

System Requirements Process

SW Requirements Process

High-Level Requirements

Low-Level Requirements & Architecture

SW Coding Process with SCADE ICC

Source Code

Integrated Executable

SW Integration Process

SCADE design document

SCADE design document

In master design document or annex

Textual Design

Derived requirements

Global Architecture Design

Requirements Process

Master design document

SRS
SW Coding Process
using SCADE Suite KCG

SW Design Process

Textual design

SCADE Modeling

Low-Level Requirements & Architecture

SW Coding Process

Manual Coding

SCADE Suite KCG

Generated Code

Manual Code

SW Integration Process

Integrated Executable

System Requirements Process

High-Level Requirements

SW Requirements Process
with SCADE Suite™

Low-Level Requirements
Architecture

Source Code

SW Coding Process
with SCADE KCG

Integrated Executable

Traceability of requirements: SCADE DOORS™ Link
The qualifiable SCADE Suite KCG code generator produces simple C code that fits the constraints of safety-critical embedded software

- Portable (ANSI C, compiler, target and OS independent)
- Structured (by function or by blocks)
- Readable, traceable (name/annotation propagation)
- Static memory allocation
- No pointer arithmetic
- No recursion, bounded loops only
- Bounded execution time
- Size or speed optimisation
- Complies with MISRA 2004 coding rules
Agenda

- Introduction
- Software Development Process
- Software Verification Processes
- Verification of Verification Process
- Conclusion
Milestones in Software Life Cycle

Inception

1) Input Data Review
2) Planning Data Review
3) Software Requirements Acceptance Review
4) SCADE Global Architecture Review
5) Test Strategy Review

Elaboration

SCADE Plans & Standards
SDE & STE setup
SRS Acceptance
SCADE Global Architecture
Test Strategy

Construction

Design
Code & Integration
Test Cases & Procedures

Transfer

9) Test Results Review
10) Delivery Readiness Review
11) Software Acceptance Review

Combined Testing

6) Design Review
7) Code & Integration Review
8) Test Preparation Review

0) Kick-Off Meeting
Verification Objectives

- Verify LLR against HLR
- Verify code against LLR
- Verify correct object code behavior on target
- Show coverage of SCADE model using requirements-based test cases (test cases cover complete model)
Debugging & Simulation at Model Level
Role of Simulation in the Verification Process

- For Low-Level Requirements (DO-178B table A-4)
  - Verify compliance of LLR to HLR A-4.1 (SCADE design = LLR)
  - Verify algorithms accuracy HLR A-4.7
    - Precision
    - Convergence/stability

- Simulation detects design errors in a much more efficient way than design review and target testing
  - Every engineer has a PC
  - Model-level debugging
  - Real life experience on traditional processes shows that most errors have been introduced by design or hand-coding process
## SCADE Impact on Design Verification (1/2)

Colors:  
- **Supported by SCADE**  
- **Fully covered by SCADE**

<table>
<thead>
<tr>
<th>Objective</th>
<th>SCADE support</th>
<th>User verification activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> Low-level requirements comply with high-level requirements.</td>
<td>Simulation</td>
<td>Review and simulation</td>
</tr>
<tr>
<td><strong>2</strong> Low-level requirements are accurate and consistent.</td>
<td>Automated check by KCG of SCADE syntax and semantic rules defined in [SCADE_LRM].</td>
<td>Run KCG and analyze syntax/semantic checks log. Verify by review design rules that are not imposed by Scade language rules.</td>
</tr>
<tr>
<td><strong>3</strong> Low-level requirements are compatible with target computer.</td>
<td>KCG generates portable sequential code. Scade LLR are qualitatively compatible with most targets.</td>
<td>Quantitative aspects to be assessed by analysis /testing. <em>Note: testing is the ultimate means for this verification (DO-178B, A-6#5)</em></td>
</tr>
<tr>
<td><strong>4</strong> Low-level requirements are verifiable.</td>
<td>Ensured by scade notation formality</td>
<td>None</td>
</tr>
</tbody>
</table>
| **5** Low-level requirements conform to standards. | Automated check by KCG of SCADE syntax and semantic rules defined in [SCADE_LRM]. | 1) Run KCG and analyze syntax/semantic checks log.  
2) Review model for conformance to user-defined LLR standards |
| **6** Low-level requirements are traceable to high-level requirements. | RM Gateway | Review HLR/LLR matrix |
### SCADE Impact on Design Verification (2/2)

<table>
<thead>
<tr>
<th>Objective</th>
<th>SCADE support</th>
<th>User verification activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Algorithms are accurate.</td>
<td>Simulation</td>
<td>Review and simulation</td>
</tr>
<tr>
<td>8 Software architecture is compatible with high-level requirements.</td>
<td>Review eased by functional nature of Scade</td>
<td>Review</td>
</tr>
<tr>
<td>9 Software architecture is consistent.</td>
<td>For inner consistency of SCADE model: automated check by KCG of scade syntax and semantic rules.</td>
<td>1) Run KCG and analyze syntax/semantic checks log. 2) Review consistency between scade model and other software</td>
</tr>
</tbody>
</table>
| 10 Software architecture is compatible with target computer.             | KCG generates portable sequential code. Scade LLR are qualitatively compatible with most targets. | Quantitative aspects to be assessed by analysis /testing.  
Note: testing is the ultimate means for this verification (DO-178B, table A-6, objective 5) |
| 11 Software architecture is verifiable.                                  | Ensured by scade notation formality                                            | None                                                                                      |
| 12 Software architecture conforms to standards.                          | Automated check by KCG of scade syntax and semantic rules.                    | 1) Run KCG and analyze syntax/semantic checks log. 2) Review model for conformance to user-defined architecture standards |
| 13 Software partitioning integrity is confirmed.                         | None                                                                          | Review                                                                                    |
Verification Objectives

- Verify LLR against HLR
- Verify code against LLR
- Verify correct object code behavior on target
- Show coverage of SCADE model using requirements-based test cases (test cases cover complete model)
Impact of SCADE Suite KCG Qualification

- When a code generator is qualified as a development tool (DO-178B; section 12.2),
  - The conformance of the code to the input model is trusted
  - The verification activities related to the coding phase are eliminated

- The SCADE Suite KCG automatic C code generators has been qualified as development tools at DO-178B level A (see DO-178B; section 12.2)
  - KCG has been developed with the same objectives (in particular verification, such as MC/DC code coverage) as level A embedded code

- Qualification for each specific project combines
  - Prequalification activities (99% of the effort): by Esterel Technologies
  - Inclusion of SCADE usage process in planning documents: by user
Coding and Integration
Reviews and Analyzes

- SCADE Code Generation Verification
  - SCADE Suite KCG is qualifiable
  - It is just required to check SCADE Suite KCG execution log

- Manual C code Review and Analysis
  - As for other manual code

- Software Build Verification
  - Simulation build is automated by qualifiable MTC
  - Target build is specific to user project
## KCG Impact on Coding Output Verification

<table>
<thead>
<tr>
<th>Objective</th>
<th>Verification method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Code complies with low-level requirements</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Source Code complies with software architecture</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Source Code is verifiable</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Source Code conforms to standards</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Source Code is traceable to low-level requirements</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Source Code is accurate and consistent</td>
<td>Ensured by SCADE KCG qualification</td>
</tr>
<tr>
<td>Output of software integration process is complete and correct</td>
<td>Analysis of the build and loading data</td>
</tr>
</tbody>
</table>
Verification Objectives

- Verify LLR against HLR
- Verify code against LLR
- Verify correct object code behavior on target
- Show coverage of SCADE model using requirements-based test cases (test cases cover complete model)
“Traditional” Testing Process (without SCADE)

System Requirements Based Testing focus

Application

Application Code

Low-Level Testing

User written C libraries

Low-Level Testing
Combined Testing Process with SCADE/KCG

- System Requirements Based Testing focus
- SW Requirements Based Low-Level Testing saved with KCG
- KCG-generated C sample
- Low-Level Testing
- Application Code Generated by SCADE KCG
- User written C libraries
SCADE Suite Compiler Verification Kit (CVK)

Customer’s Development Project (SCADE Suite)

KCG → C code → Compiler → Object code

Customer’s Environment Qualification

Integrate KCG in the Certification process

Verify Compiler

KCG Certification Kit

Compiler Verification Kit
SCADE Suite CVK is:

- A complete subset of C constructs that can be generated from any Scade model
- A series of inputs & reference output vectors ensuring MC/DC coverage at source code level

This subset approach is accepted by safety authorities (CAST-12 §h, CAST 25)
SCADE Qualifiable Test Environment (QTE)

- Test Preparation
- Test Execution & Results on host
- Test Harnesses generation for target testing

Software Requirements Specification

Specify Test Cases

Prepare Test Vectors

Execute & Report

Test Preparation Review

Test Results Review

DOORS
ACCESS
other

CSV Direct

QMTC

RTRT
other
SCADE QTE/QMTC Workflow

- Fully **Automated** tool-chain
- Run test cases on host using QMTC
  - MC/DC Model Coverage assessment
  - Report generation
- Perform **Conformance Testing**
  - Report generation
SCADE QTE/RTRT Workflow

- Generate equivalent test procedures for RTRT (PTU)
  - Including directives for Conformance Testing
- Can be adapted to other target testing solutions

- Generate Conformance Report
- Test Conformance Report
  - Actual Output

SCADE model

KCG

- Requirements-based test development (beyond qualification scope)

Test Cases
- Node identification
- Input scenarios
- Expected outputs
- Tolerance information

QTE/RTRT
- RTRT Test Driver

Execute Code

RTRT Component Testing

<table>
<thead>
<tr>
<th>Objective</th>
<th>SCADE Support</th>
<th>User Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Executable Object Code complies with high-level requirements</td>
<td>Normal test cases simulation. Possibility to re-run test cases on target (QTE)</td>
<td>Normal test cases simulation. Re-run test cases on target</td>
</tr>
<tr>
<td>2  Executable Object Code is robust with high-level requirements</td>
<td>Robustness test cases simulation. Possibility to re-run test cases on target (QTE)</td>
<td>Robustness test cases simulation Re-run test cases on target</td>
</tr>
<tr>
<td>3  Executable Object Code complies with low-level requirements</td>
<td>CVK in context of qualified code generation</td>
<td>CVK on target</td>
</tr>
<tr>
<td>4  Executable Object Code is robust with low-level requirements</td>
<td></td>
<td>Dedicated Robustness analysis/testing</td>
</tr>
<tr>
<td>5  Executable Object Code is compatible with target computer</td>
<td></td>
<td>HW/SW Integration Testing</td>
</tr>
</tbody>
</table>
Verification Objectives

- Verify LLR against HLR ✓
- Verify code against LLR ✓
- Verify correct object code behavior on target ✓

Show coverage of SCADE model using requirements-based test cases (test cases cover complete model)
Agenda

- Introduction
- Software Development Process
- Software Verification Processes
- Verification of Verification Process
- Conclusion
Testing Process Analyses and Reviews

- **Test Cases and Procedures Review and analysis**
  - Test cases and procedures need to be reviewed independently of
    - The developers
    - The authors of test cases and procedures

- **LLR coverage analysis**
  - Manually for manual code
  - Model coverage analysis for SCADE

- **Structural code coverage analysis**
  - For manual code
  - Not needed for KCG generated code

- **Test results analysis**
“Model coverage analysis determines which requirements expressed by the model were not exercised by verification based on the requirements from which the model was developed.

This analysis contributes to the detection of unintended functions in the model and potentially in the final executable object-code.”
\[ U = \text{unintended function: e.g. filter reset} \]

Not detected by traceability analysis (e.g. because granularity not going down to atomic element)

Traceability exists between LLR and code (no detection possible)

Detected by Model Coverage Analysis (100% model structure covered by HLR based test/simu cases)

Not detected by SW code structural coverage analysis (because covered by a low level test case)
Role of Model Coverage (DO-178C MBD Supplement §6.7)

- MBD supplement table A-4
  - Model coverage is a required activity for objective 1 (compliance of a design model to its higher-level requirements)

- MBD supplement table A-7
  - Model coverage is a required activity for objective 4 (coverage of low-level requirements is achieved)

- See also MBD FAQ#11 and Tool Qual FAQ D.8
Unintended Functions In Traditional Process

**HLR**
- F1
- F2

**LLR**
- F1
- F2

**Code**
- f1a
- f1b
- F2

F3 unintended

Missed by traditional process

designer

programmer

Missed by traditional process
Model Test Coverage (MTC)
Detect/Eliminate Unintended Functions from LLRs

Eliminated by MTC resolution in context of complete HLR testing

KCG does not introduce unintended code
<table>
<thead>
<tr>
<th>Objective</th>
<th>Verification method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Test procedures are correct</td>
<td>Test Procedures review</td>
</tr>
<tr>
<td>2  Test results are correct and discrepancies explained</td>
<td>Test results analysis</td>
</tr>
<tr>
<td>3  Test coverage of high-level requirements is achieved</td>
<td>Coverage analysis of HLR by test cases</td>
</tr>
<tr>
<td>4  Test coverage of low-level requirements is achieved</td>
<td>Coverage analysis of SCADE LLR with QMTC</td>
</tr>
</tbody>
</table>
| 5  Test coverage of software structure (modified condition/decision) is achieved | KCG qualification  
+ structural code coverage (MC/DC) on CVK C sample  
+ MC/DC coverage on SCADE model                                                                |
| 6  Test coverage of software structure (decision coverage) is achieved    | Guaranteed by achievement of objective 5                                                                                                          |
| 7  Test coverage of software structure (statement coverage) is achieved    | Guaranteed by achievement of objective 5                                                                                                          |
| 8  Test coverage of software structure (data coupling and control coupling) is achieved | **SCADE Code with SCADE Code**: Semantic check of  
SCADE model+ KCG qualification+ SCADE model coverage with QMTC  
**SCADE code with manual code**: verification of integration with manual code  |
Verification Objectives

- Verify LLR against HLR ✔
- Verify code against LLR ✔
- Verify correct object code behavior on target ✔
- Show coverage of SCADE model using requirements-based test cases (test cases cover complete model) ✔
Agenda

- Introduction
- Software Development Process
- Software Verification Processes
- Verification of Verification Process
- Conclusion
Key Verification Drivers and Benefits

- **DO-178B Table A-5**
  - No code reviews
    - SCADE Suite KCG is qualified

- **DO-178B Table A-6**
  - Testing can be achieved through a combination of:
    - HLR-based tests early in the lifecycle
      - Running HLR tests on Host and Target
    - and Testing of a Representative Sample of the Source Code
      - Running CVK on Target

- **DO-178B Table A-7**
  - Structural Coverage objectives may be achieved by performing Model Coverage Analysis
    - SCADE Suite MTC
Model based technology evolved after release of DO-178B

DO-178C and ist Model-Based and Tool Qualification supplements fully recognize the „SCADE way“

- Qualified code generation
  - SCADE Suite KCG
- HLR based testing on model level
  - SCADE Suite Simulation/qualifiable MTC
- C-sample testing
  - SCADE Suite CVK
- Model test coverage
  - SCADE Suite qualifiable MTC
SCADE SECURES

ACHIEVEMENT OF

CERTIFICATION OBJECTIVES

WITHIN TIME AND BUDGET
BACKUP SLIDES
SC-205/WG-71 (DO-178C) and Model Coverage
SC-205/WG-71 and Model Coverage

- Model Coverage has been handled in two Sub-Groups
  - SG3 (Tool Qualification):
    - **DP#337**: Using a Qualified Code Generator
  - SG4 (Model Based Development and Verification):
    - **FAQ#11**: Model Coverage Analysis vs. Structural Coverage Analysis

- Both papers have been approved in Plenary Session at the Marseille meeting of the SC-205/WG-71 Committee (June 10th 2010)
Discussion Paper (DP#337) discusses the potential benefits of using a Qualified Automatic Code Generator

- It is largely based on CAST Paper 25
- It presents a full user development flow based on a Qualified Code Generator and benefits that can be claimed
- It brings a common understanding of these issues between FAA, EASA and the applicants
Expected Benefits

- No code reviews (Table A-5)
- Testing can be achieved through HLR-based tests and Testing of a Representative Sample of the Source Code (Table A-6)
- Structural Coverage objectives may be achieved by performing Model Coverage Analysis (Table A-7)
FAQ#11: Model Coverage vs. Code Coverage

FAQ#11: May the applicant take credit for model coverage analysis instead of structural coverage analysis?

Answer: The applicant may take credit for model coverage analysis instead of structural coverage analysis, if the applicant can demonstrate that:

- The tool used to generate code from the model is qualified
- Any libraries used in the generation of code from the model are verified
- The verification technique(s) used to collect the model coverage data is representative of the execution of executable object code on the target computer
- The model coverage analysis criteria hold the same properties that the structural coverage analysis criteria hold
Benefits when using SCADE Suite 6.2.1

- No code reviews (Table A-5)
  - \( \rightarrow \) KCG 6.1.3 is qualifiable

- Testing can be achieved through HLR-based tests and Testing of a Representative Sample of the Source Code (Table A-6)
  - \( \rightarrow \) Running HLR-based tests on host and target
  - \( \rightarrow \) Running CVK 6.1.3 on host and target

- Structural Coverage objectives may be achieved by performing Model Coverage Analysis
  - \( \rightarrow \) Model Coverage Analysis is performed with qualifiable MTC 6.2.1 (using HLR-based tests)
The tool used to generate code from the model is qualified -> KCG 6.1.3 is qualifiable

Any libraries used in the generation of code from the model are verified
-> This is under the user’s responsibility

The verification technique(s) used to collect the model coverage data is representative of the execution of executable object code on the target computer
-> Representatativity of PC-host Simulation is verified by running HLR-based tests both on host and target

The model coverage analysis criteria hold the same properties that the structural coverage analysis criteria hold -> MTC 6.2.1 implements masking MC/DC
Conclusion on Model Coverage

Considering the 3 points below:

- Qualification of KCG
- Running CVK tests on host and target
- Running HLR-based tests on host and target

it is indeed the case that Structural Coverage objectives may be achieved at model level with MTC.
Possible Simulation and Testing Strategies

Do not consider testing strategy as a model coverage issue: model coverage is just a final completion check.

It is a **requirements** coverage challenge: a test case can be at any level, if and only if it is requirements-based.