Qualification of Test Tools for Safety-Critical Systems with Fault Injection and a Monitor

Viacheslav “Slava” Izosimov, EIS By Semcon AB, Sweden
7 March 2012
Motivation

- Modern vehicles are ultimate creations of advanced electronic components.
- However, complexity grows dramatically and “bugs” are more difficult to find...
- **ISO 26262** should help with respect to quality of automotive safety-critical systems!
- But the part on the requirements on qualification of software tools and, in particular, test systems is weak and needs to be adapted to each particular project!
- **Approaches proposed so far demand a lot of process and qualification effort.**
- **Internal tool support** for automating qualification process is necessary to reduce qualification effort and ensure completeness of the tool qualification process.
Outline

• Motivation
• DFEA2020 and FUSS projects
• Classification of software tools acc. to ISO 26262
• Example: Test tool for an ASIL D component
• Existing tool qualification methods
• EIS By Semcon tool qualification approach
  – Safety arguments
  – Monitor and fault injection
  – Static and dynamic tool validation steps
• Lessons learned and summary
DFEA 2020 – Dependable Flexible Electrical Architecture

- Develop next generation of electrical & electronic architectures for automotive vehicles:

- Green, safe and connected!

- Including model-based development, new communication technologies, architecture models for early verification, state-of-the-art control systems, electronic fault model, safety patterns and ISO 26262

- Increase Swedish competence in the area of electrical & electronic architectures
DFEA 2020 – Dependable Flexible Electrical Architecture

- Partially supported by
  - VINNOVA, Swedish Governmental Agency for Innovation Systems
- Lead by
  - Volvo Car Corporation (VCC)
- Project partners:
  - EIS (Embedded Intelligent Solutions) by Semcon
  - Mecel (making vehicles communicate)
  - Know IT Technology Management
  - SAAB (from military defence to civil security)
- Universities:
  - KTH Royal Institute of Technology
  - Chalmers University of Technology
DFEA 2020 – FUSS (FUnctional Systems Safety)

• FUSS is a subproject of DFEA2020, which focuses on
  – the system property of functional safety (system safety)
  – research on how to build safety related systems in compliance to the upcoming safety standard ISO 26262
  – safety demonstrator developed in compliance with ISO 26262
  – development of guidelines for efficient application of the standard

• and includes
  – study of an automotive fault model
  – automotive safety design pattern cookbook
  – automotive safety cases
  – guidance of safety assessment and audits
ISO 26262 Tool Classification (1)

• Quality of software tools and awareness of possible safety implications are important.
• Automotive experience: a “bug” in a software tool can lead to direct consequences.
• Examples:
  – A “bug” in a compilator may corrupt item’s machine code and affect software execution.
  – A “bug” in a hardware design tool can lead to a permanent design fault in the hardware.
  – A “bug” in a test tool can lead to escape of critical design error that may compromise item’s safety, cause a costly “re-call”, liability claims, etc.
ISO 26262 Tool Classification (2)

• Classification
  – Tool Impact (TI)
  – Tool error Detection (TD)
  – Tool Confidence Level (TCL)

• Qualification methods
  – Depending on the ASIL (Automotive Safety Integrity Level)
  – Confidence diagnostics
ISO 26262 Tool Classification (3)

• Tool Impact (TI)
  – TI1: No possibility of introducing errors
  – TI2: A possibility exists

• Tool error Detection (TD)
  – TD1: High degree of confidence
  – TD2: Medium degree of confidence
  – TD3: Low degree of confidence

• Tool Confidence Level (TCL)
  – TCL1: TI1 v (TI1 ^ TD1)
  – TCL2: TI2 ^ TD2
  – TCL3: TI2 ^ TD3
Example: A Test System for an ASIL D “Steer-By-Wire” Component

Example of ASIL D safety goal: The force feedback shall not cause any unintended steering so that the driver can lose control.

Example of one safety requirements for this safety goal: The steering feedback torque shall not exceed 5 Nm for more than 100ms. **The requirement shall be tested!**

Let’s try with a test system originally designed for functional testing of non-critical applications.
Test System Classification (1)

- **Test System for functional testing of non-critical applications:**
  - Constantly changed and lack of timing for “bug free” debugging.
  - Software documentation is often poor (focus on the test cases).
  - Increasingly high time pressure & high testing cost.
  - Test software quality is often not the highest priority (but the overall quality of the test result is of the very high priority).
- **TD3:** Low degree of confidence with respect to software.
- Undetected faults due to unsatisfactory test cases, potential “bugs” in the test system and wrong parameter configurations of the test system will inevitably lead to malfunctioning, safety issues and recalls.
- **TI2:** Possibility to affect safety definitely exists.
- **TI2 ^ TD3 = TCL3!?**
Test System Classification (2)

- What if we take the test system which has been used for testing of a safety-critical application and adapt to a new product?
  - A lot of time and money invested with respect to safety, but
  - Accidents in the past have shown that we shall not trust the software “by default” just because it was shown to be “bug free” on another safety-critical application (for example, the ARIANE 5 accident due to “re-using” of “proven” ARIANE 4 parts).
  - However, we can definitely trust this test system more than the system in the previous example!

- TD2: Medium degree of confidence can be assumed.
- TI2 \( \wedge \) TD2 = TCL2.
- And what about ASIL D?
Next Step: ASIL D Qualification Methods

• Confidence diagnostics for TCL2 and TCL3 in case of ASIL D (derived from Table 4 and Table 5 in Part 8)
  – 1a Increased confidence from use – Recommended
  – 1b Evaluation of the development process – Recommended
  – 1c Validation of the software tool – Highly Recommended
  – 1d Development in compliance with a safety standard – Highly Recommended

• TCL2 and TCL3 shall be addressed in a similar manner according to ISO 26262 for the ASIL D items!

• However, the ISO 26262 guidelines on the tool qualification end here... and we need to find the way out!

• We should consider the software tool as an Item and re-apply the respective parts of ISO 26262?
Existing Qualification Methods

• MathWorks
  – Embedded Coder, PolySpace
  – Reference process and predefined workflow
  – Joint work with TÜV SÜD

• but
  – Responsibility on the product developer.
  – No control for the tool usage.
  – Matching of each development process to the reference flow.
  – Unchanged software tool is assumed during development process.

• Becomes impractical for the test system in ASIL D level:
  – A single change in the system software may require re-qualification.
  – Re-qualification is required for every new product since, by its nature, the test system is often changed for testing the product.
EIS By Semcon Approach

- Enhancement of the test system with
  - Development guidelines
  - Development and safety process follow
  - Semi-automatic validation concept based on fault injection and self-monitoring
- and, thus,
  - Responsibility lies on the tool developer.
  - Increased control for the tool usage.
  - No need for re-qualification for every software change.
- Becomes feasible for the test system on ASIL D level.
Safety Arguments

• Software-induced fault injection (SWIFI) into the test system.
• ISO 26262 decomposition concept.
• Following ISO 26262 guidelines on software development for safety-critical test system parts.
• Self-monitoring functionality: Observing test system behavior and taking decision if we can trust results of the system.

• Example:
  – Test 43200: PASSED, Monitor: OK – PASSED
  – Test 45001: FAILED, Monitor: NOK – UNTRUSTED
  – Test 55998: FAILED, Monitor: OK – FAILED
Why Fault Injection?

- Extensively used by the certification authorities in judging the system functionality with respect to safety.
- Required verification method for ASIL D development.
- Simple and easy to manage and can be efficiently optimized.
- Does not depend so much on the system complexity as the other verification methods.
- Efficiency steadily increases with the number of faults injected.
- Non-disruptive if implemented in software.
ISO 26262 “Decomposition” and Self-Monitoring

- Decomposition of the requirements on the test system:
  - Monitor checks the guidelines and if the software development rules have been followed for the test system.
  - Monitor is much simpler than the test system and, hence, easier to completely verify and certify.
  - Monitor is stable and is not modified for every new release of the test system, i.e., no impact analysis on the Monitor is needed with the following re-certification process as a potential outcome.

![Diagram showing ASIL D, Monitor, Independence, Test System, and QM in a flowchart structure.](image-url)
Test System Validation Steps with Static Analysis and Runtime Validation

- Static Tool Validation Step A (by *tool developer*)
  
  ![Static Tool Validation Diagram]
  
  F = Fault Injection

- Static Tool Validation Step B (by *product developer*)
  
  ![Static Tool Validation Diagram]

- Dynamic Tool Validation Steps (by *product developer*)
  
  ![Dynamic Tool Validation Diagram]

“Fault Injection Run”

“Golden Run”
Lessons Learned

• We have done a practical evaluation on our in-house LabVIEW-based Universal Test Platform (UTP), originally not safety-critical, which we have managed to make suitable for testing of safety-critical applications.

• The monitor and fault injection concept does work.

• However, easier to begin new development of the test system for ASIL D with monitoring and fault injection integrated “by design” from the beginning than improve an existing test system.

• Tool development according to a chosen safety standard (in our case ISO 26262) from the early beginning can save a lot of effort compared to attempts of fixing safety aspects at the late development stages.
Acknowledgements

• Development of fault injection and self-monitoring concept:
  – Alexander Åström
  – Ola Örsmark

• “Proof-of-concept” on the Universal Test Platform (UTP):
  – Andreas Wallin
  – Qian “Linda” Wang
Summary

• ISO 26262 does not provide clear guidelines on test system qualification.
• Existing approaches (i.e., by MathWorks) shift tool qualification to the product developer and are not practical for test systems.

• Our alternative solution provides a self-qualified software tool:
  – Semi-automatic qualification with the monitor;
  – Based on software-induced fault injection; and
  – Composed of static and dynamic validation steps.
• Approach is evaluated on our LabVIEW-based test tool: the Universal Test Platform (UTP), originally developed for functional testing of non-safety critical applications.
• The concept is universal and can be applicable to different software tools.
Internal tool support for automating qualification process is necessary to reduce qualification effort and ensure completeness of the tool qualification!
Practical Evaluation on Our Universal Test Platform (UTP)

This test system has been originally designed for functional testing of non-critical applications.
Test Setup: Connecting ASIL D “Steer-By-Wire” Steery to UTP

- **Stimulus**: STEERING_WHEEL_ANGLE_MSG (CAN) and the analog angle sensor.
- **Response**: from the servo control (PWM) and FORCE_FEEDBACK_MSG (CAN).
- **ASIL D safety goal**: The force feedback shall not cause any unintended steering so that the driver can lose control.
- **A safety requirement for this safety goal**: The steering feedback torque shall not exceed 5 Nm for more than 100ms.
Practical Evaluation on the Universal Test Platform: Old Design

- No concept of time
- Only one value sent/fetched to/from hardware interface
- No support for generation of CAN messages as test stimuli
Practical Evaluation on the Universal Test Platform: New Design

- Buffers to prevent data loss
- μs accurate timestamps in hardware (Higher accuracy achievable, not needed)
- CAN bus can be used as stimuli channel
- Analog waveform generation added
Practical Evaluation on the Universal Test Platform: Monitoring and Fault Injection (1)

• Graphical fault injection into the LabVIEW code:
  – Insertion of fault injection primitives into the code for static analysis.
  – Qualification and semi-automatic static re-qualification.

• Run-time insertion of fault injection primitives into the code and communications (inputs/outputs) for dynamic analysis:
  – Accepting only those tests that we trust.
  – Signal FMEA (Fault Mode and Effect Analysis) to guarantee coverage and provide fault model.
  – Injecting faults into the code at the current execution path.

• Self-monitoring for data loses, timing issues, interface status and fault injection outcomes.
Practical Evaluation on the Universal Test Platform: Monitoring and Fault Injection (2)

“White-box” fault injection into LabVIEW code.

“Black-box” fault injection into signals and signal shapes.

- Monitor for data loss (buffer overflow).
- Monitor I/O between Test Procedure and Test Manager.