Verlässliche Echtzeitsysteme – Können wir unseren Autos noch vertrauen?

Bernhard Sechser
Method Park Consulting GmbH, Erlangen
23.06.2015

Contents

- Who is Method Park?
- Why do we need Safety Standards?
- Process and Safety demands in Automotive
- Hazard Analysis and Risk Assessment
- Functional and Technical Development
- Software Process in detail
- Tool Qualification
- Summary

Method Park - Facts and Figures

Facts
- Founded in 2001
- Locations:
  Germany: Erlangen, Munich, Stuttgart
  USA: Detroit, Miami

Awards

Revenue & employees

Business unit revenue

- Method Park Software AG
- Method Park Consulting GmbH
- Method Park Engineering GmbH
Portfolio

Training
Wide range of seminars in the division systems and software engineering
Accredited by the following organizations:
SEI, ISTQB, ISQI, INTACS, IREB, iSQI, iNTACS, IREB, iSAQB, ECQA

Engineering
Areas:
• Project Coaching
• Software Development & Support
• On Site Support
• Off Site Projects
• Fixed Price Projects

Consulting/Coaching
Topics:
• CMMI®
• Automotive SPICE
• Project Management & Agile Development
• Process Improvement & Quality Management
• Functional Safety (ISO 26262)
• Variant & Complexity Management
• Product Line Management (PLM)
• Application Lifecycle Management (ALM)
• Requirements Management
• System & Software Architecture & Design
• AUTOSAR
• System & Software Testing

Our Customers

Defense
• Airbus Deutschland
• Diehl
• EADS
• Elbit
• Orbital
• Raytheon
• Anschütz
• KID

Our Customers
Healthcare
• Siemens
• Fresenius
• Ayla
• Ziehm Imaging
• NewTeC
• Innovations Software
• Technology

Further
• Bosch und Siemens Hausgeräte
• Deutsche Post
• GMC Software Technologies
• Kodak
• Landesbank Kiel
• Raabe Karcher
• Giesecke & Devrient
• Thales Rail Signaling

IT/Telecommunications
• GFT
• Intersoft
• Nash Technologies
• NEC
• Micronas
• Siemens
• Teleca

Government/Public
• Bundesagentur für Arbeit
• Kassenärztliche Vereinigung
Baden-Württemberg

Automotive
• Audi
• Automotive Lighting
• Blaufunk
• BMW
• Bosch
• Breu
• Continental
• Daimler
• Delphi
• ETAS
• HE System Elektronik
• Heilbro
• Helita
• IAV
• Johnson Controls
• Knorr-Brakes
• Kostal
• Magna
• Faurecia
• Pilkington
• Pensa
• Renesas
• Thales
• TRW
• Volkswagen
• Webasto
• Witten Automotive
• ZF
• Zollner

Example – Ariane 5 (July 4th, 1996)
Detonation shortly after takeoff because of an error in the control software
Root cause: Insufficient tests of a reused “proven in use” software component

Source: YouTube
Source: ESA

Contents

Who is Method Park?
Why do we need Safety Standards?
Process and Safety demands in Automotive
Hazard Analysis and Risk Assessment
Functional and Technical Development
Software Process in detail
Tool Qualification
Summary

Example – Ariane 5 (July 4th, 1996)
**Example – Therac-25**

Irradiation of patients with a lethal dose

Root cause: Insufficient safety functions

---

**Examples**

Application that can cause harm (a risk):
- Airbag exploding when infant is sitting in front seat

Need to assess the risk
- Infant getting injured – “not good at all”

Find a mitigation strategy, e.g. a safety function:
- Detecting infant in front seat and disabling airbag
  - a) sensor delivers signal to
  - b) software/hardware controlling an
  - c) actuator (disabler)

Functional Safety is then:
- An infant in front seat is not exposed to an unacceptable (unreasonable) risk

---

**Reasons for Failures**

63%

Root cause analysis of software failures in 90 healthcare companies

- 60%
- 50%
- 40%
- 30%
- 20%
- 10%

- Implementation
- Architecture Design
- Requirements
- Other

Source: Fraunhofer Institute for Experimental Software Engineering 2007
§ 823 Abs. 1 BGB:
„Anyone who injures intentionally or negligently the life, body, health, liberty, property or any other right of another person, is obliged to compensate for the resulting damages."

§ 1 Abs. 1 ProdhaftG:
„If someone is killed, his body or health injured or an item damaged by a defect in a product, the manufacturer of the product is obliged to replace the resulting damages."

Safety
... is the absence of unacceptable (unreasonable) risks that can cause harm achieved through a planned strategy

Functional Safety
... is part of the overall safety that depends on a system or equipment operating correctly in response to its inputs.
... is achieved when every specified safety function is carried out and the level of performance required of each safety function is met
... is not to provide the perfect car, but a safe car.

Functional Safety Management
... is the management (plan, do, act, check) of all activities necessary to reach functional safety.
**Existing Standards**

- EN 62061: Manufacturing
- ISO 13849: Manufacturing
- EN 50126: Railway applications
- EN 50128: Railway applications
- EN 50129: Railway applications
- DO 178B: Aviation
- IEC 62304: Medical
- EN 50271: Gas Measuring
- EN 50402: Gas Measuring
- IEC 61511: Automation
- EN 62061: Automotive
- ISO 13849: Automotive
- ISO 26262: Automotive
- IEC 61513: Nuclear
- IEC 60880: Nuclear
- EN 50271: Nuclear
- EN 50402: Nuclear
- ISO 26262: Nuclear

**Scope of ISO 26262**

**Why not using IEC 61508?**

- Lessons learnt from application of IEC 61508 in automotive industry:
  - Not adapted to real-time and integrated embedded systems
  - Not adapted to automotive development and life cycles
  - No requirements for manufacturer / supplier relationship
  - No ‘consumer-goods’ orientation
  - ...

Companies had to solve these issues themselves until introduction of ISO 26262.

**Structure of ISO 26262**

Source: ISO 26262:2011

**ISO 15504 & Automotive SPICE®**

- Management Process Group (MAN)
  - MAN.1 Organizational alignment
  - MAN.2 Organizational management
  - MAN.3 Project management
  - MAN.4 Quality management
  - MAN.5 Risk management
  - MAN.6 Measurement
- Process Improvement Process Group (PIM)
  - PIM.1 Process establishment
  - PIM.2 Process assessment
  - PIM.3 Process improvement
- Resource & Infrastructure Process Group (RIN)
  - RIN.1 Human resource management
  - RIN.2 Training
  - RIN.3 Knowledge management
  - RIN.4 Infrastructure
- Reuse Process Group (REU)
  - REU.1 Asset management
  - REU.2 Reuse program management
  - REU.3 Domain engineering

**Primary Life Cycle Processes**

- Acquisition Process Group (ACQ)
  - ACQ.1 Acquisition preparation
  - ACQ.2 Supplier selection
  - ACQ.3 Contract agreement
  - ACQ.4 Supplier monitoring
  - ACQ.5 Customer acceptance
  - ACQ.6 Acquirer-related technical requirements
  - ACQ.7 Acquirer-related administrative requirements
  - ACQ.8 Project requirements
  - ACQ.9 Request for proposals
  - ACQ.10 Supplier qualification
- Supply Process Group (SPL)
  - SPL.1 Supplier tendering
  - SPL.2 Product release
  - SPL.3 Product acceptance support
- Engineering Process Group (ENG)
  - ENG.1 Requirements elicitation
  - ENG.2 System requirements analysis
  - ENG.3 Functional safety analysis
  - ENG.4 Software requirements analysis
  - ENG.5 Software design
  - ENG.6 Software construction
  - ENG.7 Software integration
  - ENG.8 Software testing
  - ENG.9 System integration
  - ENG.10 System testing
  - ENG.11 Software installation
  - ENG.12 Software and system maintenance
- Operation Process Group (OPE)
  - OPE.1 Operational use
  - OPE.2 Customer support

**Supporting Life Cycle Processes**

- Supplier Process Group (SUP)
  - SUP.1 Quality assurance
  - SUP.2 Supplier selection agreement
  - SUP.3 Validation
  - SUP.4 Supplier certification
  - SUP.5 Audit
  - SUP.6 Product evaluation
  - SUP.7 Documentation
  - SUP.8 Configuration management
  - SUP.9 Change request management

**Organizational Life Cycle Processes**

- Management Process Group (MAN)
  - MAN.1 Organizational alignment
  - MAN.2 Organizational management
  - MAN.3 Project management
  - MAN.4 Quality management
  - MAN.5 Risk management
  - MAN.6 Measurement
- Process Improvement Process Group (PIM)
  - PIM.1 Process establishment
  - PIM.2 Process assessment
  - PIM.3 Process improvement
- Resource & Infrastructure Process Group (RIN)
  - RIN.1 Human resource management
  - RIN.2 Training
  - RIN.3 Knowledge management
  - RIN.4 Infrastructure
- Reuse Process Group (REU)
  - REU.1 Asset management
  - REU.2 Reuse program management
  - REU.3 Domain engineering
**Structure of ISO 26262**

1. Viscosity  
2. Management of functional safety  
   1. Concept phase  
   2. Definition of functional safety  
   3. Requirement definition  
   4. Requirement specific safety measures  
   5. Safety analysis  
   6. Physical safety  
3. Production and operation  
4. Supporting processes  
5. ISO 15504 Process Groups  
   - Management  
   - Process Improvement  
   - Resource & Infrastructure  
   - Reuse  
   - Support  
   - Supply  
   - Acquisition  

**Safety Lifecycle Overview**

- **Concept Phase**
  - Focus on entire system  
  - Risks  
  - Safety Goals and Requirements  
  - Safety functions

- **Development**
  - System, Hardware and Software  
  - Safety validation and assessment  
  - Production and Operation (Planning)

- **Production**
Product Development

Product Development at the System Level

Source: ISO 26262-2:2011

Product Development at the Hardware Level

Source: ISO 26262-5:2011

Product Development at the Software Level

Source: ISO 26262-6:2011

Safety Lifecycle Overview

After Release for Production
- Production
- Installation
- Operation
- Maintenance and reparation
- Disassembly

Source: ISO 26262-2:2011
Hazard Analysis and Risk Assessment

List of driving and operating situations
Estimation of the probability of Exposure

Detailing failure modes leading to hazards in specific situations
Estimation of Controllability

Evaluating consequences of the hazards
Estimation of potential Severity

Risk reduction to an acceptable level

- Residual risk
- Tolerable risk
- Necessary risk reduction
- Actual risk reduction
- Risk reduction achieved by all the safety-related systems and other risk reduction measures

E = Exposure
C = Controllability
S = Severity
ASIL = Automotive Safety Integrity Level

Hazard Analysis and Risk Assessment

Associations of the central concepts
**Exposure**
State of being in an operational situation that can be hazardous if coincident with the failure mode under analysis

<table>
<thead>
<tr>
<th>Class</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Incr-e-dible</td>
<td>Very low probability</td>
<td>Low probability</td>
<td>Medium probability</td>
<td>High probability</td>
</tr>
<tr>
<td>Time</td>
<td>Not specified</td>
<td>Less than 1% of average operating time</td>
<td>1% - 10% of average operating time</td>
<td>&gt; 10% of average operating time</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Situations that occur less than once a year for the great majority of drivers</td>
<td>Situations that occur a few times a year for the great majority of drivers</td>
<td>Situations that occur once a month or more often for an average driver</td>
<td>All situations that occur during almost every drive on average</td>
<td></td>
</tr>
</tbody>
</table>

Source: ISO 26262-3:2011

---

**Severity**
Measure of the extent of harm to an individual in a specific situation

<table>
<thead>
<tr>
<th>Class</th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>No injuries</td>
<td>Light and moderate injuries</td>
<td>Severe and life-threatening injuries (survival probable)</td>
<td>Life-threatening injuries (survival uncertain), fatal injuries</td>
</tr>
</tbody>
</table>

Source: ISO 26262-3:2011

---

**Controllability**
Avoidance of the specified harm or damage through the timely reactions of the persons involved

<table>
<thead>
<tr>
<th>Class</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controllable in general</td>
<td>Simply controllable</td>
<td>Normally controllable</td>
<td>Difficult to control or uncontrollable</td>
</tr>
<tr>
<td>Definition</td>
<td>99% or more of all drivers or other traffic participants are usually able to avoid a specific harm.</td>
<td>90% or more of all drivers or other traffic participants are usually able to avoid a specific harm.</td>
<td>Less than 90% of all drivers or other traffic participants are usually able, or barely able, to avoid a specific harm.</td>
<td></td>
</tr>
</tbody>
</table>

Source: ISO 26262-3:2011

---

Combinations of Severity, Exposure and Controllability result in the applicable ASIL.

The ASIL’s influence the development process of the items.

QM = Quality Management
No specific ISO 26262 requirement has to be observed

If S0 or E0 or C0 is set, no ASIL is required (QM).
Safety Goals
- top-level safety requirements as a result of the hazard analysis and risk assessment
- assigned to each identified hazard rated with an ASIL A-D
- lead to item characteristics needed to avert hazards or to reduce risks associated with the hazards to an acceptable level
- are assigned to a safe state that must be reached in case of appearance
- indicate the maximum fault tolerance time within the safe state must be reached

fault tolerance time = fault recognition time + fault reaction time

Safe State – Operating mode of an item without an unreasonable level of risk
- Example: intended operating mode, degraded operating mode or switched-off mode

<table>
<thead>
<tr>
<th>ID</th>
<th>Safety Goal</th>
<th>ASIL</th>
<th>Safe State</th>
<th>FTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Avoidance of unintended maximum brake force build up at one or several wheels during drive and in all environmental conditions</td>
<td>D</td>
<td>Brake released</td>
<td>50 ms</td>
</tr>
<tr>
<td>G2</td>
<td>Guarantee the specified parking brake function in use case situation &quot;parking on slope&quot; in all environmental conditions</td>
<td>A</td>
<td>Brake closed</td>
<td>500 ms</td>
</tr>
<tr>
<td>G3</td>
<td>Avoidance of unintended release of the parking brake in use case situation &quot;parking on slope&quot; in all environmental conditions</td>
<td>C</td>
<td>Brake closed</td>
<td>500 ms</td>
</tr>
</tbody>
</table>
Functional Safety Concept

Safety Goals and Functional Safety Requirements

ASIL Decomposition

Source: ISO 26262-9:2011

Architectures

Example: Three channel structure 2oo3

Contents

- Who is Method Park?
- Why do we need Safety Standards?
- Process and Safety demands in Automotive
- Hazard Analysis and Risk Assessment
- Functional and Technical Development
- Software Process in detail
- Tool Qualification
- Summary
**Product Development at Hardware & Software Level**

- **Important part:**
  - Hardware-Software Interface Specification (HSI)

  Source: ISO 26262-4:2011

---

**How to understand the standard tables**

For each method, the degree of recommendation to use corresponding methods depends on the ASIL and is categorized as follows:

- **"++"** The method is highly recommended for this ASIL
- **"+"** The method is recommended for this ASIL
- **"o"** The method has no recommendation for or against its usage for this ASIL

---

**Initiation of Product Development at the Software Level**

- **Topics to be covered by modeling and coding guidelines**

<table>
<thead>
<tr>
<th>Topics</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Enforcement of low complexity</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b Use of language subsets</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1c Enforcement of strong typing</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1d Use of defensive implementation techniques</td>
<td>0</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1e Use of established design principles</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1f Use of unambiguous graphical representation</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1g Use of style guides</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1h Use of naming conventions</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

  Source: ISO 26262-6:2011

---

**Specification of Software Safety Requirements**

- **Goals**

  - Derive Software Safety Requirements from and ensure consistency with
    - System Design
    - Technical Safety Concept
  - Detail the hardware-software interface requirements

  Source: ISO 26262-6:2011
### Specification of Software Safety Requirements

**Methods for specifying Safety Requirements**
- Safety requirements shall be specified by an appropriate combination of natural language and methods listed in the table.
- For higher level safety requirements (e.g., functional and technical safety requirements) natural language is more appropriate while for lower level safety requirements (e.g., software and hardware safety requirements) notations listed in the table are more appropriate.

**Methods**

<table>
<thead>
<tr>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1c</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1d</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Source: ISO 26262-8:2011

### Specification of Software Safety Requirements

**Methods for the verification of Safety Requirements**

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification by walk-through</td>
<td>++</td>
</tr>
<tr>
<td>Verification by inspection</td>
<td>+</td>
</tr>
<tr>
<td>Semi-formal verification (e.g., executable models)</td>
<td>++</td>
</tr>
<tr>
<td>Formal verification</td>
<td>+</td>
</tr>
</tbody>
</table>

Source: ISO 26262-8:2011

### Software Architectural Design

**Goals**
- Develop an Architecture that implements the Software Safety Requirements
  - Static and dynamic interfaces
  - Safety-related and non safety related requirements
- Verify the Software Architecture
  - Compliance with the requirements
  - Compatibility with hardware
  - Respect of design principles and standards

### Software Architectural Design

**Principles for software architectural design**

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical structure of software components</td>
<td>++</td>
</tr>
<tr>
<td>Restricted size of software components</td>
<td>++</td>
</tr>
<tr>
<td>Restricted size of interfaces</td>
<td>+</td>
</tr>
<tr>
<td>High cohesion within each software component</td>
<td>+</td>
</tr>
<tr>
<td>Restricted coupling between software components</td>
<td>+</td>
</tr>
<tr>
<td>Appropriate scheduling properties</td>
<td>+</td>
</tr>
<tr>
<td>Restricted use of interrupts</td>
<td>+</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011
Based on the results of the safety analysis the mechanisms for error detection and error handling shall be applied.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Range checks of input and output data</td>
<td>++ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b Plausibility check</td>
<td>+ + ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c Detection of data errors</td>
<td>+ + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d External monitoring facility</td>
<td>o + + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e Control flow monitoring</td>
<td>o ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1f Diverse software design</td>
<td>o o + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error detection

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Static recovery mechanism</td>
<td>+ + + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b Graceful degradation</td>
<td>+ + ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c Independent parallel redundancy</td>
<td>o o + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d Correcting codes for data</td>
<td>+ + + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error handling

Source: ISO 26262-6:2011

Methods for the verification of the software architectural design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Walk-through of the design</td>
<td>++ + o o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b Inspection of the design</td>
<td>+ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c Simulation of dynamic parts of the design</td>
<td>+ + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d Prototype generation</td>
<td>o o + +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e Formal verification</td>
<td>o o +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1f Control flow analysis</td>
<td>+ + ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1g Data flow analysis</td>
<td>+ + ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011

Software Unit Design and Implementation

Goals

- Specify SW Units based on:
  - SW Architecture
  - SW Safety Requirements
- Implement the SW Units
- Verify SW Units
  - Code reviews / inspections

Source: ISO 26262-6:2011

Design principles for software unit design and implementation

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a One entry and one exit point in subprograms and functions</td>
<td>++ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b No dynamic objects or variables, or else online test during their creation</td>
<td>+ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c Initialization of variables</td>
<td>++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d No multiple use of variable names</td>
<td>+ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e Avoid global variables or else justify their usage</td>
<td>+ + ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1f Limited use of pointers</td>
<td>o + + ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1g No implicit type conversions</td>
<td>+ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1h No hidden data flow or control flow</td>
<td>+ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1i No unconditional jumps</td>
<td>++ ++ ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1j No recursions</td>
<td>+ + ++ ++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011
Software Unit Design and Implementation

Example: MISRA C

- Programming standard developed by Motor Industry Software Reliability Association
- Avoidance of runtime errors due to unsafe C constructs
- The respect of MISRA C shall be demonstrated → static code analysis

Infos: www.misra.org

Software Unit Testing

Goals

- Demonstrate that the software units fulfil the Software Unit Specifications
- Verify absence of undesired functionalities

The software unit testing methods shall be applied to demonstrate that the software units achieve:

- Compliance with the software unit design specification
- Compliance with the specification of the hardware-software interface
- Correct implementation of the functionality
- Absence of unintended functionality
- Robustness
- Sufficiency of the resources to support the functionality

Methods for deriving test cases for software unit testing

- Analysis of requirements
- Generation and analysis of equivalence classes
- Analysis of boundary values
- Error guessing

Source: ISO 26262-6:2011
Software Unit Testing

Structural coverage metrics at the software unit level

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Statement coverage</td>
<td>++ ++ + +</td>
</tr>
<tr>
<td>1b Branch coverage</td>
<td>+ ++ + +</td>
</tr>
<tr>
<td>1c MC/DC (Modified Condition/Decision Coverage)</td>
<td>+ + ++ ++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011

Software Integration and Testing

Goals

- Integrate SW components
- Integration sequence
- Testing of interfaces between components/units
- Verify correct implementation of the SW Architecture

Source: ISO 2626-6:2011

The software integration test methods shall be applied to demonstrate that both the software components and the embedded software achieve:

- Compliance with the software architectural design
- Compliance with the specification of the hardware-software interface
- Correct implementation of the functionality
- Robustness and sufficiency of the resources to support the functionality

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Requirements-based test</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1b Interface test</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1c Fault injection test</td>
<td>+ + ++ ++</td>
</tr>
<tr>
<td>1d Resource usage test</td>
<td>+ + + +</td>
</tr>
<tr>
<td>1e Back-to-back comparison test between model and code, if applicable</td>
<td>+ + ++ ++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011

Structural coverage metrics at the software architectural level

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Function coverage</td>
<td>+ + ++ ++</td>
</tr>
<tr>
<td>1b Call coverage</td>
<td>+ + ++ ++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011
Verification of Software Safety Requirements

**Goals**
- Verify that the embedded software fulfils the Software Safety Requirements in the target environment

Source: ISO 26262-6:2011

**Verification of Software Safety Requirements**

- Verify that the embedded software fulfils the software safety requirements
- Verification of the software safety requirements shall be executed on the target hardware
- The results of the verification of the software safety requirements shall be evaluated in accordance with:
  - Compliance with the expected results
  - Coverage of the software safety requirements
  - A pass or fail criteria

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Hardware-in-the-loop</td>
<td>A</td>
</tr>
<tr>
<td>1b Electronic control unit network environments</td>
<td>B</td>
</tr>
<tr>
<td>1c Vehicles</td>
<td>C</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011

**Functional Safety Assessment**

**Contents**

- Who is Method Park?
- Why do we need Safety Standards?
- Process and Safety demands in Automotive
- Hazard Analysis and Risk Assessment
- Functional and Technical Development
- Software Process in detail
- Tool Qualification
- Summary
Qualification of Software Tools

To determine the required level of confidence in a software tool, perform a use case analysis:

- Evaluate if a malfunctioning software tool and its erroneous output can lead to the violation of any safety requirement allocated to the safety-related item or element to be developed
- Establish probability of preventing or detecting such errors in its output
  - Considers measures internal to the software tool (e.g. monitoring)
  - Measures external to the software tool implemented in the development process for the safety-related item or element (e.g. guidelines, tests, reviews)

### Tool Impact (TI)
Possibility that a safety requirement, allocated to the safety-related item or element, is violated if the software tool is malfunctioning or producing erroneous output

- **TI1** – no such possibility
- **TI2** – all other cases

### Tool error Detection (TD)
Probability of preventing or detecting that the software tool is malfunctioning or producing erroneous output

- **TD1** – high degree of confidence for prevention or detection
- **TD2** – medium degree of confidence for prevention or detection
- **TD3** – all other cases

### Tool Confidence Level (TCL)
Based on the values determined for the classes of TI and TD

<table>
<thead>
<tr>
<th>TI</th>
<th>TD1</th>
<th>TD2</th>
<th>TD3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI1</td>
<td>TCL1</td>
<td>TCL1</td>
<td>TCL1</td>
</tr>
<tr>
<td>TI2</td>
<td>TCL1</td>
<td>TCL2</td>
<td>TCL3</td>
</tr>
</tbody>
</table>

Source: ISO 26262-8:2011

Qualification methods:

<table>
<thead>
<tr>
<th>Qualification methods of software tools classified TCL3</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1a</td>
<td></td>
</tr>
<tr>
<td>Increased confidence from use</td>
<td>++</td>
</tr>
<tr>
<td>1b</td>
<td></td>
</tr>
<tr>
<td>Evaluation of the tool development process</td>
<td>++</td>
</tr>
<tr>
<td>1c</td>
<td></td>
</tr>
<tr>
<td>Validation of the software tool</td>
<td>+</td>
</tr>
<tr>
<td>1d</td>
<td></td>
</tr>
<tr>
<td>Development in accordance with a safety standard</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qualification methods of software tools classified TCL2</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1a</td>
<td></td>
</tr>
<tr>
<td>Increased confidence from use</td>
<td>++</td>
</tr>
<tr>
<td>1b</td>
<td></td>
</tr>
<tr>
<td>Evaluation of the tool development process</td>
<td>++</td>
</tr>
<tr>
<td>1c</td>
<td></td>
</tr>
<tr>
<td>Validation of the software tool</td>
<td>+</td>
</tr>
<tr>
<td>1d</td>
<td></td>
</tr>
<tr>
<td>Development in accordance with a safety standard</td>
<td>+</td>
</tr>
</tbody>
</table>

Source: ISO 26262-8:2011
Contents

- Who is Method Park?
- Why do we need Safety Standards?
- Process and Safety demands in Automotive
- Hazard Analysis and Risk Assessment
- Functional and Technical Development
- Software Process in detail
- Tool Qualification
- Summary

Summary

- Today’s electronic systems are too complex to understand all potential hazards
- An approach for Functional Safety is needed to avoid severe injuries and damages in human lives and property
- A standardized way to show that your product is safe is needed – best practice yet not fully established – guidance needed

Thank you!

Bernhard Sechser
Principal Consultant SPICE & Safety
Method Park Consulting GmbH
Wetterkreuz 19a
91058 Erlangen
Germany

Phone: +49 9131 97206-427
Mobile: +49 173 3882055
Bernhard.Sechser@methodpark.com
http://www.xing.com/profile/Bernhard_Sechser
http://www.methodpark.com