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
- 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**
- 2004
- 2008
- 2011
- 2005

**Revenue & employees**
![Bar chart showing revenue and employees from 2001 to 2013]

**Business unit revenue**
- 24% Method Park Software AG
- 49% Method Park Consulting GmbH
- 26% Method Park Engineering GmbH

© 2015 Method Park Consulting GmbH / Bernhard Sechser / 23.06.2015 / Verlässliche Echtzeitsysteme
Portfolio

Product

Solution for integrated process management

Engineering

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

Consulting/Coaching

Topics:
- CMMI®, SPICE, 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

Training

Wide range of seminars in the division systems and software engineering

Accredited by the following organizations:
SEI, ISTQB, iSQI, iNTACS, IREB, iSAQB, ECQA
Our Customers

Automotive
- Audi
- Automotive Lighting
- Blaupunkt
- BMW
- Bosch
- Brose
- Continental
- Daimler
- Delphi
- ETAS
- HE System Elektronik
- Helbako
- Hella
- IAV
- Johnson Controls
- Knorr-Bra克斯
- Kostal
- Marquardt
- Peiker Acustic
- Preh
- Renesas
- Thales
- TRW
- Volkswagen
- Webasto
- Witte Automotive
- ZF
- Zollner

Engineering/Automation
- 7 layers
- ABB
- BDT
- Carl Schenk
- EBM Papst
- Heidelberger Druckmaschinen
- Insta
- Kratzer Automation
- Magirus
- Mettler Toledo
- Mühlbauer Group
- Rohde & Schwarz
- Siemens Industries
- Wago

Healthcare
- Carl Zeiss
- Siemens
- Fresenius
- Agfa
- Ziehm Imaging
- NewTec
- Innovations Software
- Technology

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

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

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

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: ESA

Source: YouTube
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

Question: How to measure and agree on the measures?
Your Brake Function is temporarily unavailable, please **STOP** the Car immediately!

- **OK**
- **CANCEL**

**Question:** Do we dare putting software in direct control of people’s life?
Reasons for Failures

- Implementation: 10%
- Architecture Design: 16%
- Requirements: 63%
- Other: 11%

Root cause analysis of software failures in 90 healthcare companies

Source: Fraunhofer Institute for Experimental Software Engineering 2007
**Current Situation**

**Trends in Automotive Electric/Electronics (E/E)**

- Increasing functionality and complexity of software-based car functions
- Increasing risks from systematic faults and random hardware faults
- Most of the new car functions are safety-related

Source: © Courtesy of Daimler; Presentation given at Automotive Electronics and Electrical Systems Forum 2008, May 6, 2008, Stuttgart, Germany
§ 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.“
Contents

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

**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

**IEC 61508**
Functional safety of electrical / electronic / programmable electronic safety-related systems

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

...
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

1. Vocabulary

2. Management of functional safety
   2-5 Overall safety management
   2-6 Safety management during the concept phase and the product development
   2-7 Safety management after the item’s release for production

3. Concept phase
   3-5 Item definition
   3-6 Initiation of the safety lifecycle
   3-7 Hazard analysis and risk assessment
   3-8 Functional safety concept

4. Product development at the system level
   4-5 Initiation of product development at the system level
   4-6 Specification of the technical safety requirements
   4-7 System design
   4-8 Item integration and testing
   4-11 Release for production
   4-10 Functional safety assessment
   4-9 Safety validation

5. Product development at the hardware level
   5-5 Initiation of product development at the hardware level
   5-6 Specification of hardware safety requirements
   5-7 Hardware design
   5-8 Evaluation of the hardware architectural metrics
   5-9 Evaluation of the safety goal violations due to random hardware failures
   5-10 Hardware integration and testing

6. Product development at the software level
   6-6 Initiation of product development at the software level
   6-7 Software architectural design
   6-8 Software unit design and implementation
   6-9 Software unit testing
   6-10 Software integration and testing
   6-11 Verification of software safety requirements

7. Production and operation
   7-5 Production
   7-6 Operation, service (maintenance and repair), and decommissioning

8. Supporting processes
   8-5 Interfaces within distributed developments
   8-6 Specification and management of safety requirements
   8-7 Configuration management
   8-8 Change management
   8-9 Verification
   8-10 Documentation
   8-11 Confidence in the use of software tools
   8-12 Qualification of software components
   8-13 Qualification of hardware components
   8-14 Proven in use argument

9. ASIL-oriented and safety-oriented analyses
   9-5 Requirements decomposition with respect to ASIL tailoring
   9-6 Criteria for coexistence of elements
   9-7 Analysis of dependant failures
   9-8 Safety analyses

10. Guideline on ISO 26262

Source: ISO 26262:2011
Structure of ISO 26262

ISO 15504 Process Groups

Source: ISO 26262:2011
Safety Lifecycle Overview

- Concept
- Development
- Production

Source: ISO 26262-2:2011
Safety Lifecycle Overview

Concept Phase

- Focus on entire system
- Risks
- Safety Goals and Requirements
- Safety functions

Source: ISO 26262-2:2011
Safety Lifecycle Overview

Product Development

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

Source: ISO 26262-2:2011
Product Development at the System Level

Source: ISO 26262-2:2011

Source: ISO 26262-4:2011
Product Development

Product Development at the Hardware Level

Source: ISO 26262-2:2011

Source: ISO 26262-5:2011
Product Development

Product Development at the Software Level

Source: ISO 26262-2:2011

Source: ISO 26262-6:2011
Safety Lifecycle Overview

After Release for Production

- Production
- Installation
- Operation
- Maintenance and reparation
- Disassembly

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

Risk reduction to an acceptable level

- Residual risk
- Tolerable risk
- EUC risk

Necessary risk reduction

Actual risk reduction

- Partial risk covered by other risk reduction measures #2
- Partial risk covered by E/E/PE safety-related systems
- Partial risk covered by other risk reduction measures #1

Risk reduction achieved by all the safety-related systems and other risk reduction measures

Source: IEC 61508-5:2010
Situation analysis and hazard identification

- 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

→ Respect only the plain item (do not take risk-reducing measures into account!)

→ Involve persons with good knowledge and domain experience
Hazard Analysis and Risk Assessment

Associations of the central concepts

- **E** = Exposure
- **C** = Controllability
- **S** = Severity
- **ASIL** = Automotive Safety Integrity Level
## Hazard Analysis and Risk Assessment

### 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><strong>Description</strong></td>
<td>Incredible</td>
<td>Very low probability</td>
<td>Low probability</td>
<td>Medium probability</td>
<td>High probability</td>
</tr>
<tr>
<td><strong>Time</strong></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><strong>Event</strong></td>
<td>Situations that occur less often 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
**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><strong>Description</strong></td>
<td>Controllable in general</td>
<td>Simply controllable</td>
<td>Normally controllable</td>
<td>Difficult to control or uncontrollable</td>
</tr>
<tr>
<td><strong>Definition</strong></td>
<td>Controllable in general</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>
</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
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).

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
</tr>
<tr>
<td>S1</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E3</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E4</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>S2</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E1</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td>E3</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E4</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>S3</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E1</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E3</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>E4</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Source: ISO 26262-3:2011
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
## Example for Safety Goals: Park Brake System

<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</td>
<td>D</td>
<td>Brake released</td>
<td>50 ms</td>
</tr>
<tr>
<td></td>
<td>during drive and in all environmental conditions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>Guarantee the specified parking brake function in use case situation</td>
<td>A</td>
<td>Brake closed</td>
<td>500 ms</td>
</tr>
<tr>
<td></td>
<td>&quot;parking on slope&quot; in all environmental conditions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td>Avoidance of unintended release of the parking brake in use case situation</td>
<td>C</td>
<td>Brake closed</td>
<td>500 ms</td>
</tr>
<tr>
<td></td>
<td>&quot;parking on slope&quot; in all environmental conditions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Safety Goals and Functional Safety Requirements
ASIL Decomposition

Source: ISO 26262-9:2011
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
Topics to be covered by modeling and coding guidelines

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informal notations for requirements specification</td>
<td>+</td>
</tr>
<tr>
<td>Semi-formal notations for requirements specification</td>
<td>++</td>
</tr>
<tr>
<td>Formal notations for requirements specification</td>
<td>++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-8:2011
## Methods for the verification of Safety Requirements

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1a</strong> Verification by walk-through</td>
<td>++</td>
</tr>
<tr>
<td><strong>1b</strong> Verification by inspection</td>
<td>+</td>
</tr>
<tr>
<td><strong>1c</strong> Semi-formal verification (e.g. executable models)</td>
<td>+</td>
</tr>
<tr>
<td><strong>1d</strong> Formal verification</td>
<td>o</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

Source: ISO 26262-6:2011
# Principles for software architectural design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Hierarchical structure of software components</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1b Restricted size of software components</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1c Restricted size of interfaces</td>
<td>+ + + +</td>
</tr>
<tr>
<td>1d High cohesion within each software component</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td>1e Restricted coupling between software components</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td>1f Appropriate scheduling properties</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1g 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.

Error detection

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

Error handling

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

Source: ISO 26262-6:2011
## Methods for the verification of the software architectural design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1a Walk-through of the design</td>
<td>++</td>
</tr>
<tr>
<td>1b Inspection of the design</td>
<td>+</td>
</tr>
<tr>
<td>1c Simulation of dynamic parts of the design</td>
<td>+</td>
</tr>
<tr>
<td>1d Prototype generation</td>
<td>o</td>
</tr>
<tr>
<td>1e Formal verification</td>
<td>o</td>
</tr>
<tr>
<td>1f Control flow analysis</td>
<td>+</td>
</tr>
<tr>
<td>1g Data flow analysis</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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1a</strong> One entry and one exit point in subprograms and functions</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1b</strong> No dynamic objects or variables, or else online test during their creation</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1c</strong> Initialization of variables</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1d</strong> No multiple use of variable names</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1e</strong> Avoid global variables or else justify their usage</td>
<td>+ + ++ ++</td>
</tr>
<tr>
<td><strong>1f</strong> Limited use of pointers</td>
<td>o + + ++</td>
</tr>
<tr>
<td><strong>1g</strong> No implicit type conversions</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1h</strong> No hidden data flow or control flow</td>
<td>+ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1i</strong> No unconditional jumps</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td><strong>1j</strong> No recursions</td>
<td>+ + ++ ++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011
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

Source: ISO 26262-6:2011
Software Unit Testing

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

<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
## Methods for deriving test cases for software unit testing

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td><strong>1a</strong> Analysis of requirements</td>
<td>++</td>
</tr>
<tr>
<td><strong>1b</strong> Generation and analysis of equivalence classes</td>
<td>+</td>
</tr>
<tr>
<td><strong>1c</strong> Analysis of boundary values</td>
<td>+</td>
</tr>
<tr>
<td><strong>1d</strong> Error guessing</td>
<td>+</td>
</tr>
</tbody>
</table>

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></td>
<td>A</td>
</tr>
<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 26262-6:2011
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></td>
<td>A</td>
</tr>
<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 fulfills 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></td>
<td>A</td>
</tr>
<tr>
<td><strong>1a</strong> Hardware-in-the-loop</td>
<td>+</td>
</tr>
<tr>
<td><strong>1b</strong> Electronic control unit networks</td>
<td>++</td>
</tr>
<tr>
<td><strong>1c</strong> Vehicles</td>
<td>++</td>
</tr>
</tbody>
</table>

Source: ISO 26262-6:2011
What shall be provided to support the Safety Case?

- Product Definition
- Hazard Analysis
- Design Function
- Design Architecture
- Design System
- Design Component
- Create Hardware & Software
- Safety Analysis
- Safety Verification
- Verify Function
- Verify Architecture
- Verify System
- Verify Component
- Create System Design
- Verify System Design
- Safety Case - Arguments

Safety Case - Arguments
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:

#### Qualification methods of software tools classified TCL3

<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>

#### Qualification methods of software tools classified TCL2

<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
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