Verlässliche Echtzeitsysteme – Können wir unseren Autos noch vertrauen?

Bernhard Secher
Method Park Software AG, Erlangen
30.04.2012
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
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
  - USA: Detroit, Miami

Awards

- 2009
- Handelsblatt

Revenue & employees

Business unit revenue

- Products: 33%
- Training & Consulting: 45%
- Engineering: 22%
Portfolio

**Product**
*stages*
Solution for integrated process management

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

**Consulting/Coaching**
Topics:
- Software Process Improvement
- CMMI®, SPICE, Automotive SPICE®
- AUTOSAR, Functional Safety
- Requirements Management
- Project and Quality Management
- Software Architecture & Design
- Software Testing

**Training**
Wide range of seminars in the division system and software development

Accredited by the following organizations:
SEI, ISTQB, ISQI, INTACS, IREP
Our Customers

Automotive
- Audi
- Automotive Lighting
- Blaupunkt
- BMW
- Bosch
- Brose
- Continental
- Daimler
- Delphi
- ETAS
- Helbako
- IAV
- Knorr-Braeske
- Marquardt
- Peiker Acustic
- Preh
- Thales
- TRW
- Volkswagen
- Webasto
- ZF
- Zollner

Engineering/Automation
- 7 layers
- ABB
- BDT
- Carl Schenk
- EBM Papst
- Heidelberg 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

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

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

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

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

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

Source: Fraunhofer Institute for Experimental Software Engineering 2007

Root cause analysis of software failures in 90 healthcare companies
Complexity

DAIMLER

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

- ISO 26262
  Automotive

- IEC 61511
  Automation

- EN 50126
- EN 50128
- EN 50129
  Rail

• • •
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

2.6 Overall safety management

2.8 Safety management during item development

2.7 Safety management after 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

4.6 Initiation of product development at the system level

4.8 Specification of the technical safety requirements

4.9 System design

4.11 Release for production

4.10 Functional safety assessment

4.12 Safety validation

4.13 Item integration and testing

## 5. Product development at the hardware level

5.6 Initiation of product development at the hardware level

5.8 Specification of hardware safety requirements

5.7 Hardware design

5.9 Hardware architectural metrics

5.10 Validation of the safety goal due to random HxV failures

5.11 Hardware integration and testing

## 6. Product development at the software level

6.5 Initiation of product development at the software level

6.6 Specification of software safety requirements

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 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 usage 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 dependent failures

9.8 Safety analyses

## 10. Guideline on ISO 26262

Source: ISO/FDIS 26262 - BL18
### 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.11 Technical requirements
- ACQ.12 Legal and administrative requirements
- ACQ.13 Project requirements
- ACQ.14 Request for proposals
- ACQ.15 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 System architectural design
- 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

**Support Process Group (SUP)**
- SUP.1 Quality assurance
- SUP.2 Verification
- SUP.3 Validation
- SUP.4 Joint review
- SUP.5 Audit
- SUP.6 Product evaluation
- SUP.7 Documentation
- SUP.8 Configuration management
- SUP.9 Problem resolution management
- SUP.10 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

ISO 15504 Process Groups

1. Vocabulary
2. Management of functional safety
3. Concept phase
4. Product development at the system level
5. Product development at the hardware level
6. Product development at the software level
7. Production and operation
8. Supporting processes
9. ASIL-oriented and safety-oriented analyses
10. Guideline on ISO 26262

Source: ISO/FDIS 26262 - BL18
Safety Lifecycle Overview

- **Concept**

- **Development**

- **Production**

Source: ISO/FDIS 26262-2 – BL18
Safety Lifecycle Overview

Concept Phase

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

Product Development

- System, Hardware and Software
- Safety validation and assessment
- Production and Operation (Planning)
Product Development at the System Level

Product Development

Source: ISO/FDIS 26262-2 – BL18

Source: ISO/FDIS 26262-4 – BL18
Product Development at the Hardware Level

ISO 26262-5: Product development at the hardware level

4.7 System Design

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 safety goal violations due to random hardware failures

5.10 Hardware integration and testing

5.11 Qualification of hardware components

4.8 Item integration and testing

7.5 Production

7.6 Operation, service (maintenance and repair), and decommissioning

Source: ISO/FDIS 26262-2 – BL18

Source: ISO/FDIS 26262-5 – BL18
Product Development

Product Development at the Software Level

Source: ISO/FDIS 26262-2 – BL18

Source: ISO/FDIS 26262-6 – BL18
Safety Lifecycle Overview

After Release for Production

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

Source: IEC 61508-5
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

- **Failure Mode**
- **HAZARD**
- **Situation**
- **ASIL**

- **Hazardous Event**
- **Control**
- **Avoidance of Harm**

- **ECS**
- **Harm**
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>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>
<td></td>
</tr>
</tbody>
</table>
### 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>
**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>
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

Example for safety goals: Park Brake System

<table>
<thead>
<tr>
<th>ID</th>
<th>Safety Goal</th>
<th>ASIL</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>
</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>
</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>
</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
### ASIL Decomposition

<table>
<thead>
<tr>
<th>ASIL D Decomposition</th>
<th>ASIL C Decomposition</th>
<th>ASIL B Decomposition</th>
<th>ASIL A Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>before decomposition</strong></td>
<td><strong>before decomposition</strong></td>
<td><strong>before decomposition</strong></td>
<td><strong>before decomposition</strong></td>
</tr>
<tr>
<td>ASIL D</td>
<td>ASIL C</td>
<td>ASIL B</td>
<td>ASIL A</td>
</tr>
<tr>
<td>requirements in 5.4.11</td>
<td>requirements in 5.4.11</td>
<td>requirements in 5.4.11</td>
<td>requirements in 5.4.11</td>
</tr>
<tr>
<td>ASIL D</td>
<td>ASIL B</td>
<td>ASIL A</td>
<td>ASIL A</td>
</tr>
<tr>
<td>requirements in 5.4.11 and 5.4.12</td>
<td>requirements in 5.4.11</td>
<td>requirements in 5.4.11</td>
<td>requirements in 5.4.11</td>
</tr>
<tr>
<td>ASIL D(D) + ASIL A(D)</td>
<td>ASIL C(C) + QM(C)</td>
<td>ASIL B(B) + QM(B)</td>
<td>ASIL A(A) + QM(A)</td>
</tr>
</tbody>
</table>

**Source:** ISO/FDIS 26262-9 – BL18
Example: Three channel structure 2003
Product Development at Hardware & Software Level

Important part: Hardware-Software Interface Specification (HSI)

Source: ISO/FDIS 26262-4 – BL18
### Topics to be covered by modeling and coding guidelines

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

Source: ISO/FDIS 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
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
## Principles for 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 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/FDIS 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></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Range checks of input and output data</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b Plausibility check</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>
</tr>
<tr>
<td>1d External monitoring facility</td>
<td>o</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1e Control flow monitoring</td>
<td>o</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1f Diverse software design</td>
<td>o</td>
<td>o</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>

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

Error detection

Error handling

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

Source: ISO/FDIS 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
### Design principles for software unit design and implementation

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

Source: ISO/FDIS 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
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></td>
<td>A</td>
</tr>
<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/FDIS 26262-6:2011
# Software Unit Testing

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

Source: ISO/FDIS 26262-6:2011
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/FDIS 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
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

### Methods

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

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

Goals

- Verify that the embedded software fulfills the Software Safety Requirements in the target environment
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></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 network environments</td>
<td>++</td>
</tr>
<tr>
<td><strong>1c</strong> Vehicles</td>
<td>++</td>
</tr>
</tbody>
</table>

Source: ISO/FDIS 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**
- **Verify**
  - System
  - Architecture
  - Function
  - Product
- **Validated Component**
  - Verify Component
  - Verify System
  - Verify Architecture
  - Verify Function
  - Validated Product
- **Safety Verification**
- **Safety Case - Arguments**

© 2012 Method Park Software AG / Bernhard Sechser / 30.04.2012 / Verlässliche Echtzeitsysteme
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
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).
Qualification of Software Tools

**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 ...
- TD3 – all other cases
**Tool Confidence Level (TCL)** – Based on the values determined for the classes of TI and TD

<table>
<thead>
<tr>
<th></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/FDIS 26262-8:2011
Qualification of Software Tools

Qualification methods:

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

Source: ISO/FDIS 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 Software AG  
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