Verlässliche Echtzeitsysteme – Können wir unseren Autos noch vertrauen?

Bernhard Sechser
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

Method Park - Facts and Figures

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

Awards

Revenue & employees

Business unit revenue

33%

22%

45%

© 2012 Method Park Software AG / Bernhard Sechser / 30.04.2012 / Verlässliche Echtzeitsysteme
Portfolio

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

Consulting/Coaching
Topics:
- Software Process Improvement
- CMM®, 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
- Bausparkart
- BMW
- Bosch
- Brose
- Continental
- Daimler
- Delphi
- ETAS
- Helbako
- IAV
- Kien-Braes
- Manquardt
- Nokaer Acoustic
- Preh
- Thales
- TRW
- Volkswagen
- Webasto
- ZF
- Zollner

Engineering/Automation
- 7 layers
- ABB
- BDT
- Carl Schenck
- BMB Papst
- Heidelberg Druckmaschinen
- Indra
- Kratzer Automation
- Magnus
- Miflter Toledo
- Moeller Baur Group
- Rohn&Schwarz
- Siemens Industries
- Wago

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

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

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

Defense
- Airbus Deutschland
- Dahm
- RADS
- Raytheon Anschtz
- KFD

Further
- Bosch und Siemens Hausgeräte
- Deutsche Post
- GMC Software Technologies
- Kaldak
- 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?

Examples

Warning
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

63% Root cause analysis of software failures in 90 healthcare companies
50% Architecture Design
40% Requirements
30% Other

20% Implementation
16%
10%

Source: Fraunhofer Institute for Experimental Software Engineering 2007

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
Extract from German law

§ 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.”

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

EN 50271
EN 50402
Gas Measuring

IEC 61513
IEC 60880
Nuclear

DO 178B
Aviation

IEC 62304
Medical

ISO 26262
Automotive

IEC 61511
Automation

EN 50126
EN 50128
EN 50129
Rail

IEC 62304
Medical

© 2012 Method Park Software AG / Bernhard Sechser / 30.04.2012 / Verlässliche Echtzeitsysteme
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
Safety Lifecycle Overview

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

- **Development**
  - Item definition
  - Initiation of the safety lifecycle
  - Hazard analysis and risk assessment
  - Functional safety concept

- **Production**
  - Operation planning
  - Production planning
  - Safety validation and assessment
  - Production and Operation (Planning)

Product Development at the System Level

- Product development system level
  - System, Hardware and Software
  - Safety validation and assessment
  - Production and Operation (Planning)

Source: ISO/FDIS 26262-2 – BL18

Source: ISO/FDIS 26262-4 – BL18
Hazard Analysis and Risk Assessment

Risk reduction to an acceptable level

- Residual risk
- Tolerable risk
- EUC risk

Source: IEC 61508-5

- Necessary risk reduction
- Actual risk reduction
- Partial risk covered by other technology safety-related systems
- Partial risk covered by E/E/PE safety-related systems
- Partial risk covered by external risk reduction facilities

Risk reduction achieved by all safety-related systems and external risk reduction facilities

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

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>
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>Description</td>
<td>Incredible</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 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 or 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>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>

Combinations of Severity, Exposure and Controllability result in the applicable ASIL. The ASIL’s influence the development process of the items.

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>
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

After decomposition

Source: ISO/FDIS 26262-9 - BL18

Architectures

Example: Three channel structure 2003
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

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</th>
</tr>
</thead>
<tbody>
<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>0</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>

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/FDIS 26262-6:2011
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

Principles for software architectural design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range checks of input and output data</td>
<td>++</td>
</tr>
<tr>
<td>Plausibility check</td>
<td>+</td>
</tr>
<tr>
<td>Detection of data errors</td>
<td>+</td>
</tr>
<tr>
<td>External monitoring facility</td>
<td></td>
</tr>
<tr>
<td>Control flow monitoring</td>
<td>0</td>
</tr>
<tr>
<td>Diverse software design</td>
<td>0</td>
</tr>
<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>

Error detection

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range checks of input and output data</td>
<td>++</td>
</tr>
<tr>
<td>Plausibility check</td>
<td>+</td>
</tr>
<tr>
<td>Detection of data errors</td>
<td>+</td>
</tr>
<tr>
<td>External monitoring facility</td>
<td></td>
</tr>
<tr>
<td>Control flow monitoring</td>
<td>0</td>
</tr>
<tr>
<td>Diverse software design</td>
<td>0</td>
</tr>
<tr>
<td>Static recovery mechanism</td>
<td>+</td>
</tr>
<tr>
<td>Graceful degradation</td>
<td>+</td>
</tr>
<tr>
<td>Independent parallel redundancy</td>
<td>0</td>
</tr>
<tr>
<td>Correcting codes for data</td>
<td>+</td>
</tr>
<tr>
<td>Walk-through of the design</td>
<td>++</td>
</tr>
<tr>
<td>Inspection of the design</td>
<td>+</td>
</tr>
<tr>
<td>Simulation of dynamic parts of the design</td>
<td>+</td>
</tr>
<tr>
<td>Prototype generation</td>
<td>0</td>
</tr>
<tr>
<td>Formal verification</td>
<td>0</td>
</tr>
<tr>
<td>Control flow analysis</td>
<td>+</td>
</tr>
<tr>
<td>Data flow analysis</td>
<td>+</td>
</tr>
</tbody>
</table>

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>
</tr>
</thead>
<tbody>
<tr>
<td>Range checks of input and output data</td>
<td>++</td>
</tr>
<tr>
<td>Plausibility check</td>
<td>+</td>
</tr>
<tr>
<td>Detection of data errors</td>
<td>+</td>
</tr>
<tr>
<td>External monitoring facility</td>
<td></td>
</tr>
<tr>
<td>Control flow monitoring</td>
<td>0</td>
</tr>
<tr>
<td>Diverse software design</td>
<td>0</td>
</tr>
<tr>
<td>Static recovery mechanism</td>
<td>+</td>
</tr>
<tr>
<td>Graceful degradation</td>
<td>+</td>
</tr>
<tr>
<td>Independent parallel redundancy</td>
<td>0</td>
</tr>
<tr>
<td>Correcting codes for data</td>
<td>+</td>
</tr>
<tr>
<td>Walk-through of the design</td>
<td>++</td>
</tr>
<tr>
<td>Inspection of the design</td>
<td>+</td>
</tr>
<tr>
<td>Simulation of dynamic parts of the design</td>
<td>+</td>
</tr>
<tr>
<td>Prototype generation</td>
<td>0</td>
</tr>
<tr>
<td>Formal verification</td>
<td>0</td>
</tr>
<tr>
<td>Control flow analysis</td>
<td>+</td>
</tr>
<tr>
<td>Data flow analysis</td>
<td>+</td>
</tr>
</tbody>
</table>

Methods for the verification of the software architectural design

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range checks of input and output data</td>
<td>++</td>
</tr>
<tr>
<td>Plausibility check</td>
<td>+</td>
</tr>
<tr>
<td>Detection of data errors</td>
<td>+</td>
</tr>
<tr>
<td>External monitoring facility</td>
<td></td>
</tr>
<tr>
<td>Control flow monitoring</td>
<td>0</td>
</tr>
<tr>
<td>Diverse software design</td>
<td>0</td>
</tr>
<tr>
<td>Static recovery mechanism</td>
<td>+</td>
</tr>
<tr>
<td>Graceful degradation</td>
<td>+</td>
</tr>
<tr>
<td>Independent parallel redundancy</td>
<td>0</td>
</tr>
<tr>
<td>Correcting codes for data</td>
<td>+</td>
</tr>
<tr>
<td>Walk-through of the design</td>
<td>++</td>
</tr>
<tr>
<td>Inspection of the design</td>
<td>+</td>
</tr>
<tr>
<td>Simulation of dynamic parts of the design</td>
<td>+</td>
</tr>
<tr>
<td>Prototype generation</td>
<td>0</td>
</tr>
<tr>
<td>Formal verification</td>
<td>0</td>
</tr>
<tr>
<td>Control flow analysis</td>
<td>+</td>
</tr>
<tr>
<td>Data flow analysis</td>
<td>+</td>
</tr>
</tbody>
</table>
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

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

Design principles for software unit design and implementation

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>A</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
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</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>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

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/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

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Requirements-based test</td>
<td>A++</td>
</tr>
<tr>
<td>1b Interface test</td>
<td>A++</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

Verification of Software Safety Requirements

Goals

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

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

Source: ISO/FDIS 26262-6:2011
Functional Safety Assessment

What shall be provided to support the Safety Case?

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

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)

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

**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
Qualification of Software Tools

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

<table>
<thead>
<tr>
<th>TI1</th>
<th>TD1</th>
<th>TD2</th>
<th>TD3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCL1</td>
<td>TCL1</td>
<td>TCL1</td>
<td>TCL1</td>
</tr>
</tbody>
</table>

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

Source: ISO/FDIS 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></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