Introduction to ISO 26262

Even-André Karlsson
Introduction

- Even-André Karlsson – 30 years of Process and Quality improvement
  - Model based Improvement CMMI, A-SPICE, COBIT, ISDS
  - System engineering, Architecture, Tools, Requirements engineering
  - Agile, Lean, Team based organisation and Coaching
  - Automotive, Mechanical, Mobile, Telecom

- Company changes but with continued focus and services:
  - Process improvement
  - Software Quality
  - Software Safety
  - Supplier Management
  - Open Source Software

- SPICE/CMMI references
  - Accel, Atlas Copco, Autoliv, BorgWarner, Consat, GM, Mecel, Stoneridge, Volvo
  - ABB, Ericsson, FMC, IKEA, Kongsberg, QLIK, SAAB, Thales, Visma
Introduction – participants

- Name, role/background
- Experience in ISO 26262
- Expectations for the day
Agenda

- 1300-1315 Introduction
- 1315-1400 Functional Safety Background
- 1400-1530 ISO 26262
- 1530-1600 ISO 26262 and Automotive SPICE
- 1600-1630 Implementing ISO 26262
Principles

▪ Focus
  - Respect times
  - Email/phone
  - Active

▪ Communication
  - Listen
  - Respect
  - Seek understanding

▪ Parking lot
Functional Safety Background
Consequences of un-safe software

Unintended acceleration

Experts determined after 18 months review that the software was “substandard” and that Toyota had not followed “best practice”

Toyota has paid so far
• 1 Billion for deceased
• 1 Billion to US authorities for concealing information
• 1 Billion for reduced second hand value
Recalls

- 2004, Jaguar recalls 67,798 cars for transmission fix. A Software defect slams the car into reverse gear if there is a major oil pressure drop.

- 2015, Nissan recalls 23.00 ”Micras” due to a software defect that causes the car to suddenly accelerate unintentionally.

- 2016, GM recalls 4.3 million cars for airbag software defect. The bug, affecting all pickups and SUVs, can prevent the airbags from deploying in a crash.

- 2016, Volvo recalls 59.000 cars due to a software bug after some owners experienced that their engines stopping and restarting while they were driving.

Ref: Software Integrity
Automotive and the Standards

- 1985 ISO 9000 ➔ TS 16949
  - Product development
  - Product and Process Focus

- 1995 CMMI/SPICE ➔ Automotive SPICE
  - Software development
  - Software and Process Focus

- 2005 IEC 61508 ➔ ISO 26262
  - Safety critical development
  - Software, Hardware and Process Focus

- 201X – SECURITY ??
Safety

Software dependent systems is safe when:

▪ features ensure *predictable performance* under normal/abnormal conditions

▪ the *probability* of an undesirable event occurring is minimized

▪ an undesirable event does occur, the *consequences* are controlled

*absence of unreasonable risk*

*D. Herrman “Software Safety and Reliability”*
A problem has been detected and Windows has been shut down to prevent data loss.

MEMORY_MANAGEMENT

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If it is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical Information:

*** STOP: 0x00000001A

Beginning dump of physical memory
Physical memory dump complete.

Contact your system administrator or technical support group for further assistance.
Blue screens...

... are annoying in Windows computers...

... but could be safety concerns in embedded systems!
Safe systems

- How safe do our systems need to be?
- Safety requirements change over time
Safe system vs Safe usage....
Safety related automotive functionality

- Active systems
- Passive systems
- Information systems
- E/E enhanced mechanical systems
- Light control
- Powertrain
- Autonomous drive
Software based systems in cars
Safety related failure modes

- Absence of function when needed
  - Acceleration, break, turn, etc

- Unintended function
  - Acceleration, break, turn, engine stop, air bag, etc.
  - or...sudden power seat movement

- Safety of the intended functionality (SOTIF)
  - Artificial intelligence (AI) and machine learning play key roles in the development of autonomous vehicles ➔ increase complexity!
  - New standard
History of Functional Safety Standards

- The principles underpinning Functional Safety were developed in the military, nuclear and aerospace industries during the 1960-1970 ties

- **1995 IEC 1508**
  - New approach to functional safety – Risk based
  - Define safety requirements to reduce risk

- **1998-2000 IEC 61508**
  - New approach to functional safety – Risk based
  - Define safety requirements to reduce risk

- **IEC 61508 detailed in**
  - Medical IEC 62304
  - Machinery IEC 62061
  - Railway EN 5012X
  - Nuclear Process IEC 61513 …
  - Automotive IEC 26262
How do we “prove” that a system is safe?

- Follow standards with requirements & guidelines for safe systems
  - Exhaustive testing not possible

- Typically, the standards require
  - defined process that cover the whole life cycle

- activities to ensure that the defined way of working is followed and complies with the standard
- evidence of the safety related activities
Liability

- A manufacturer has to organize the company to ensure that design, development and documentation faults are eliminated or detected.
- The manufacturer has to prove that it is not responsible for a fault. ➔ By using state of the art science and technology.
- “State of the art” in automotive
  - IATF 16949
  - Automotive SPICE
  - ISO 26262
- If the malfunction could not have been detected by the technical state of the art, the liability is excluded.
Standards are always behind…
What is ISO 26262?

- A functional safety standard for E/E systems in road vehicles
- Addresses hazards caused by malfunctioning behavior of E/E systems
- Provides requirements on organization, processes, and methods
- Covers the product lifecycle from concept phase to decommissioning
- First edition was published November 2011.

- Second (and current) edition was published 2018
  - Inclusion of all road vehicles: busses, trucks and motorcycles
  - Safety of the Intended Functionality (SOTIF)
  - Cyber Security, Model Based Development and Agile SW development
  - Development of random hardware failure metrics
  - Ensure confidence in the use of software tools to include vendor validation
  - Semiconductors guide
Basic principles

▪ Perform risk analysis
▪ Define safety goals/requirements to reduce identified risks
▪ Avoid systematic failures by following defined processes and using recommended methods
▪ Control systematic and random hardware failures during operation
▪ Manage the safety activities (plan, follow-up, etc.)
▪ Evidence of the safety related activities - a safety case
▪ Traceability
▪ Perform functional safety assessment to judge the functional safety achieved
Management of functional safety

- Part 1: Vocabulary
- Part 2: Management of functional safety
- Part 3: Concept phase
- Part 4: Product development at the system level
- Part 5: Product development at the hardware level
- Part 6: Product development at the software level
- Part 7: Production & operation
- Part 8: Supporting processes
- Part 9: ASIL-oriented and safety-oriented analyses
- Part 10: Guideline on ISO 26262 (informative)
- Part 11: Guidelines on application of ISO 26262 to semiconductors
- Part 12: Adaptation of ISO 26262 for motorcycles
## ISO 26262 overview – clauses and lifecycle

<table>
<thead>
<tr>
<th>1. Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Management of functional safety</td>
</tr>
<tr>
<td>3. Concept phase</td>
</tr>
<tr>
<td>4. Product development at the system level</td>
</tr>
<tr>
<td>5. Product development at the hardware level</td>
</tr>
<tr>
<td>6. Product development at the software level</td>
</tr>
<tr>
<td>7. Production, operation, service and decommissioning</td>
</tr>
<tr>
<td>8. Supporting processes</td>
</tr>
<tr>
<td>9. Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses</td>
</tr>
<tr>
<td>10. Guidelines on ISO 26262</td>
</tr>
<tr>
<td>11. Guidelines on application of ISO 26262 to semiconductors</td>
</tr>
</tbody>
</table>

### 12. Adaptation of ISO 26262 for motorcycles
- 12-6 General topics for adaptation for motorcycles
- 12-7 Safety culture
- 12-8 Confirmation measures
- 12-9 Hazard analysis and risk assessment
- 12-10 Vehicle Integration and testing

### 2. Management of functional safety
- 2-5 Overall safety management
- 2-6 Project dependent safety management
- 2-7 Safety management regarding production, operation, service and decommissioning

### 3. Concept phase
- 3-5 Item definition
- 3-6 Hazard analysis and risk assessment
- 3-7 Functional safety concept

### 4. Product development at the system level
- 4-5 General topics for the product development at the system level
- 4-6 Technical safety concept
- 4-7 System and item integration and testing
- 4-8 Safety validation

### 5. Product development at the hardware level
- 5-5 General topics for the 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 verification

### 6. Product development at the software level
- 6-5 General topics for the 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 verification
- 6-10 Software integration and verification
- 6-11 Testing of the embedded software

### 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 management
- 8-11 Confidence in the use of software tools
- 8-12 Qualification of software components
- 8-13 Evaluation of hardware elements
- 8-14 Proven in use argument
- 8-15 Interfacing an application that is out of scope of ISO 26262
- 8-16 Integration of safety-related systems not developed according to ISO 26262

### 9. Automotive safety integrity level (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

### 11. Guidelines on application of ISO 26262 to semiconductors
Requirements for compliance

- Each clause contains requirements and recommendations
- Each requirement shall be complied with unless:
  a) tailoring shows that the requirement does not apply, or
  b) rationale for non-compliance has been assessed and accepted

Method tables
- ++ highly recommended
- + recommended
- o no recommendation (for or against)
- ASIL dependent
- Use “appropriate combination” for alternative entries and give rationale for selection

Table 1 — Topics to be covered by modelling and coding guidelines

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

*An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.*

*The objectives of method 1b are*
- Exclusion of ambiguously defined language constructs which may be interpreted differently by different modelers, programmers, code generators or compilers.
- Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
- Exclusion of language constructs which could result in unhandled run-time errors.
- Exclusion of language constructs which are not inherent in the language.
Management of functional safety

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase
Part 4: Product development at the system level
Part 5: Product development at the hardware level
Part 6: Product development at the software level

Part 7: Production & operation

Part 8: Supporting processes
Part 9: ASIL-oriented and safety-oriented analyses
Part 10: Guideline on ISO 26262 (informative)
Part 11: Guidelines on application of ISO 26262 to semiconductors

Part 12: Adaptation of ISO 26262 for motorcycles
Management of functional safety

Overall safety management
- Allocate safety responsibilities
- Create safety culture
- Training and qualification
- Quality management system

Project dependent safety management
- Appoint roles (PM & Safety Manager)
- Tailor safety activities
- Establish and follow up safety plan
- Develop safety case
- Confirmation measures

Safety management regarding production, operation, service and decommissioning
- Appoint roles
- Establish processes, e.g. field monitoring
Safety plan and safety case

- Safety plan
  - Plan to manage and guide the execution of the safety activities of a project including dates, milestones, tasks, deliverables, responsibilities and resources
  - Created and followed up by the (project) safety manager

- Safety case
  - Arguments that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development
  - Input to functional safety assessment
Confirmation measures

- **Confirmation review**
  Checks the compliance of work products to the ISO 26262 requirements

- **Functional safety audit**
  Evaluates the implementation of the processes required for functional safety

- **Functional safety assessment**
  Evaluates the functional safety achieved by the item. Shall consider:
  - work products in safety plan
  - processes required for safety
  - appropriateness and effectiveness of the implemented safety measures

Ref: TÜV training
Examples for evaluating a safety culture

Accountability not traceable
Cost/time highest priority
Reward system favors cost/time
Dependent assessor
Passive attitude (problem driven)
Resources not planned
“Group think”
No defined processes
No process improvement

Accountability traceable
Safety highest priority
Reward system favors safety
Independent assessor
Proactive attitude
Planned resources
Diversity encouraged
Defined processes are followed
Continuous process improvement

Poor safety culture

Good safety culture
Concept phase

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase

Part 4: Product development at the system level

Part 5: Product development at the hardware level

Part 6: Product development at the software level

Part 7: Production & operation

Part 8: Supporting processes

Part 9: ASIL-oriented and safety-oriented analyses

Part 10: Guideline on ISO 26262 (informative)

Part 11: Guidelines on application of ISO 26262 to semiconductors

Part 12: Adaptation of ISO 26262 for motorcycles
Concept phase

- Item definition
- Hazard analysis and risk assessment
- Functional safety concept
Item definition

- Functional and non-functional requirements
  - Operating modes and states
  - Operational and environmental constraints
  - Legal requirements and standards
  - Assumptions
  - Potential consequences of failures

- Item boundaries and interaction with other items or elements

- Determine if it is new development or modification of an existing item

- Impact Analysis
Hazard analysis and risk assessment

1. Situation analysis and hazard identification

2. Classification of hazardous events

Severity:

<table>
<thead>
<tr>
<th></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>

Probability of exposure:

<table>
<thead>
<tr>
<th></th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incredible</td>
<td>Very low probability</td>
<td>Low probability</td>
<td>Medium probability</td>
<td>High probability</td>
<td></td>
</tr>
</tbody>
</table>

Controllability:

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generally controllable</td>
<td>Simply controllable</td>
<td>Normally controllable</td>
<td>Difficult to control or uncontrollable</td>
<td></td>
</tr>
</tbody>
</table>

3. Determination of ASIL and safety goals
### Safety Goals and ASIL

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

Example 1: Airbag does not deploy during crash
- Severe injuries -> S3
- Very low exposure -> E1
- Not controllable -> C3

![ASIL A]

Safety goal: Airbag shall deploy during crash

Example 2: Unwanted airbag deployment
- Severe injuries -> S3
- High exposure -> E4
- Difficult to control -> C3

![ASIL D]

Safety goal: No unwanted airbag deployment
ASIL – Automotive Safety Integrity Level

- Represent how dangerous a hazardous event is
- Determines the required degree of safety measures to avoid unreasonable risk (which requirements in ISO 26262 that shall be applied)

\[ \text{ASIL} = \text{Severity} \times \text{Exposure} \times \text{Controllability} \]

- ASIL D is the most stringent level and ASIL A the least stringent level
- The ASIL is an attribute of a safety requirement

Ref: TüV training
Hazard analysis is context dependent

<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Parameter</th>
<th>Guide-word</th>
<th>Deviation</th>
<th>Hazardous event</th>
<th>Operational situation</th>
<th>Conseq</th>
<th>S</th>
<th>E</th>
<th>C</th>
<th>ASIL</th>
<th>Causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Fuel level estimation (AE201)</td>
<td>Total fuel level</td>
<td>Supplied too high</td>
<td>Total fuel level supplied too high</td>
<td>Fuel gauge indicates higher fuel level than actual fuel level in the tank during driving</td>
<td>Free way&lt;br&gt;High way- heavy traffic&lt;br&gt;City driving, slippery road-high traffic&lt;br&gt;City driving- snow and ice-driving speed 50 km/h</td>
<td>Vehicle is driven until no more fuel could be collected from the tank. Resulting in engine stop suddenly. Thus crush by other cars coming from behind is expected.</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>A</td>
<td>1) Erroneous fuel estimation by Kalman filter&lt;br&gt;2) Bug in gauge function&lt;br&gt;3) Mechanical Fault in gauge</td>
</tr>
</tbody>
</table>

Typical Automotive Classifications

- **Rear Lights**
  - Both Side Failure,
  - ASIL-B

- **Rear View Camera**
  - No Valid Sensor Data,
  - ASIL-B

- **Brake Lights**
  - Both Side Failure,
  - ASIL-B

- **Anitlock Braking**
  - Unintended Full Power Braking,
  - ASIL-D

- **Active Suspension**
  - Suspension Oscillates,
  - ASIL-B to C

- **Instrument Cluster**
  - Loss of Critical Data,
  - ASIL-B

- **Airbag**
  - Inadvertent Deploy
  - ASIL-D

- **Engine Management**
  - Unwanted Acceleration,
  - ASIL-C to D

- **HeadLights**
  - Both Side Failure,
  - ASIL-B

- **Radar Cruise Control**
  - Inadvertent Braking,
  - ASIL-B

- **Vision ADAS**
  - Incorrect Sensor Feedback,
  - ASIL-B

- **Electric Power Steering**
  - Self-steering,
  - ASIL-D

Ref: Synopsys, Mentor
Impact of an ASIL?

- For all ASILs: Safety mechanisms to detect and handle the relevant failure modes at system level shall be introduced.

- For ASIL A and ASIL B
  - Emphasis on additional development activities and for quality assurance of introduced safety mechanisms. (e.g. Reviews and V&V activities)

- For ASIL C and ASIL D
  - Further emphasis on additional development activities and for quality assurance of introduced safety mechanisms.
  - Requirements on performance of safety mechanisms. (Typically require HW redundancy)
  - Independent confirmation measurements
Functional safety concept

- Functional safety requirement derived from the safety goals
- Functional safety requirements allocated to system architecture
- Input to the product development phase
Safety requirements hierarchy

- Safety Goals
  - Functional Safety Requirements (in Functional Safety Concept)
    - Technical Safety Requirements
      - System Architectural Design
        - Hardware Safety Requirements
        - Software Safety Requirements
  - System development (Part 4)
    - Concept (Part 3)

Hardware development (Part 5)
Software development (Part 6)
**ASIL Decomposition**

- Divide the architecture into **redundant** and **independent** parts

- Can be applied on all levels, and repeatedly

- But we need to ensure no common failures

---

**ASIL C**

- ASIL QM (C) + ASIL C

<table>
<thead>
<tr>
<th>Coupling factor class</th>
<th>ISO 26262-9, 7.4.4</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>system level</td>
<td>hardware level</td>
</tr>
<tr>
<td>Shared Resource</td>
<td>a) random hardware failures</td>
<td>— Power supply (see also Insufficient Environmental Immunity)</td>
</tr>
<tr>
<td></td>
<td>g) failures of common external resources</td>
<td>— Wiring harness</td>
</tr>
</tbody>
</table>

“Failure of shared resources” and “single physical root cause” in ISO 26262-11

© Addalot Consulting AB - All rights reserved
Product development: System level

- General topics for the product development at the system level
- Technical safety concept
- Product development: hardware level (part 5)
- Product development: software level (part 6)
- System and Item integration and testing
- Safety validation
Technical Safety Concept

- The **technical safety concept** is an aggregation of the technical safety requirements and the corresponding system architectural design that provides rationale as to why the system architectural design is suitable to fulfil safety requirements.

Requirements

- Specification of the technical safety requirements
- Safety mechanisms (detection, indication and control of faults)
- System architectural design specification
- Safety Analyses and avoidance of systematic failures
- Measures for control of random hardware failures
- Allocation to hardware and software
- Hardware-software interface (HSI) specification
- Verification methods
System and item integration and testing

- The integration of the item's elements is carried out in a systematic way starting from software-hardware integration and verification through system integration and verification to vehicle integration.

Requirements

- Specification of integration and test strategy
- Hardware-software integration and testing
- System integration and testing

Test methods, examples:

- Requirement based tests
- Fault injections tests
- Resource usage test
- Stress test
Safety Validation

- The purpose of safety validation is to provide evidence that the safety goals are achieved and that the safety concepts (FSC TSC) are appropriate.

Requirements

- Safety validation environment
- Specification of safety validation
- Execution of safety validation
- Evaluation

Methods to be used for validation

- Analysis (e.g. FMEA, FTA, simulation)
- Long term tests
- User test
- Reviews
Management of functional safety

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase

Part 4: Product development at the system level

Part 5: Product development at the hardware level

Part 6: Product development at the software level

Part 7: Production & operation

Part 8: Supporting processes

Part 9: ASIL-oriented and safety-oriented analyses

Part 10: Guideline on ISO 26262 (informative)

Part 11: Guidelines on application of ISO 26262 to semiconductors

Part 12: Adaptation of ISO 26262 for motorcycles
Product development: hardware level

- General topics for the product development at the hardware level
- Specification of hardware safety requirements
- Hardware design
  - Evaluation of the hardware architectural metrics
  - Evaluation of safety goal violation due to random hardware failure
- Hardware integration and testing
Hardware architecture metrics

Single-point fault metrics = $1 - \frac{\sum(\lambda_{SPF} + \lambda_{RF})}{\sum \lambda}$

Latent-fault metrics = $1 - \frac{\sum \lambda_{MPF, Latent}}{\sum(\lambda - \lambda_{SPF} - \lambda_{RF})}$

- $\lambda$: Fault frequency
- SPF: Single-Point Fault
- MPF: Multiple-Point Fault
- RF: Residual Fault
Random HW failure goals

<table>
<thead>
<tr>
<th>ASIL</th>
<th>Failure rate / h⁻¹</th>
<th>FIT / 10⁻⁹ h⁻¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>&lt; 1 × 10⁻⁸</td>
<td>&lt; 10</td>
</tr>
<tr>
<td>C</td>
<td>&lt; 1 × 10⁻⁷</td>
<td>&lt; 100</td>
</tr>
<tr>
<td>B</td>
<td>&lt; 1 × 10⁻⁷</td>
<td>&lt; 100</td>
</tr>
</tbody>
</table>

What does this mean?

- 10 FIT = 10 Error per 10⁹ hours = 10 Errors per 114,155 years
- But with 2,000,000 cars on the road, it means that 175 cars will experience this fault every year...
- Now it is not so bad, as the cars don’t run 24/7, but assume they run an hour a day, we still have 7 exploding airbags every year...
- Calculations are mainly to show that you have done an analysis.
Management of functional safety

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase

Part 4: Product development at the system level

Part 5: Product development at the hardware level

Part 6: Product development at the software level

Part 7: Production & operation

Part 8: Supporting processes

Part 9: ASIL-oriented and safety-oriented analyses

Part 10: Guideline on ISO 26262 (informative)

Part 11: Guidelines on application of ISO 26262 to semiconductors

Part 12: Adaptation of ISO 26262 for motorcycles
Product development at the software level

- General topics for the product development at the hardware level
- Specification of the software safety requirements
- Software architectural design
- Software unit design and implementation
- Software integration and verification
- Testing of the embedded software
Overview
General topics for the product development at the software level

Objective

▪ to ensure a suitable and consistent software development process; and
▪ to ensure a suitable software development environment

Requirements

▪ Software development processes and software development environments
  - suitable for developing safety-related embedded software
  - support consistency across the sub-phases of the software development lifecycle
  - are compatible with the system and hardware development phases
▪ Criteria for selecting a design, modelling or programming language
Specification of the software safety req’s

Objectives

▪ Specify software safety requirements derived from the technical safety concept and the system design specification

▪ Detail the hardware-software interface requirements

▪ Verify that the software safety requirements and the hw-sw interface req’s are consistent with the technical safety concept and the system design spec.

Requirements

▪ Scope of software safety requirements

▪ Derivation of software safety requirements

▪ ASIL decomposition

▪ HW/SW interface specification

▪ Non safety related functions

▪ Verification of software safety requirements
Software architectural design

Objectives

1. Develop a software architectural design that realizes the software safety requirements
2. Verify the software architectural design

General

- The software architectural design represents all software components and their interactions in a hierarchical structure.
  - Static aspects, such as interfaces and data paths between all software components
  - Dynamic aspects, such as process sequences and timing behavior are described

- In order to develop a software architectural design both software safety requirements as well as all non-safety-related requirements are implemented.

- The software architectural design provides the means to implement the software safety requirements and to manage the complexity of the software development.
Architecture and SW Safety analysis

- Use well known architecture
- Keep it simple
- Basis for SW Safety-oriented analysis
- SW Safety-oriented analysis can be very cumbersome if at too detailed level.
- Whole Appendix E discuss this
Req’s and recommendations

- Use of appropriate notation
- Design considerations
- Modular design
- Identification of sw units
- Design aspects
- Component categorization
- New/modified components
- Re-used components
- Allocation of Safety req’s
- ASIL of combined components
- Software partitioning
- Dependent failure analysis
- Safety analysis
- Error detection
- Error handling
- New hazards
- Resource usage
- Architectural design verification
SW unit design and implementation

Objectives

▪ Develop a software unit design in accordance with the software architectural design
▪ Implement the software units as specified.

This sub-phase safety-related and non-safety-related requirements are handled within one development process.

Requirements

▪ Suitable and consistent unit design
▪ Unit design notation (natural, informal, semi-formal, formal)
▪ Specification of the software units
▪ Design principles for software unit design
Software unit verification

Objective

▪ Provide evidence that the software unit design satisfies the allocated software requirements and is suitable for the implementation

Requirements

▪ The software unit testing methods
▪ Methods for deriving software unit test cases
▪ Code coverage
▪ The test environment for software unit
Software integration and testing

Objectives

- Integrate the software
- Provide evidence that the integrated software units and SW components fulfil their requirements according to the software architectural design

Requirements

- The software integration approach
- Software integration test methods
- Methods for deriving software integration test cases
- Coverage of requirements
- Methods for structural coverage
- The test environment for software integration testing
Testing of the embedded software

Objective
- Fulfils the safety-related requirements when executed in the target environment

Requirements
- Test environments (Hardware-in-the-loop, ECU/Bench, Vehicle)
- Methods for tests
- Methods for deriving test cases
- Evaluation of test result
Production & operation

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase

Part 4: Product development at the system level

Part 5: Product development at the hardware level

Part 6: Product development at the software level

Part 7: Production & operation

Part 8: Supporting processes

Part 9: ASIL-oriented and safety-oriented analyses

Part 10: Guideline on ISO 26262 (informative)

Part 11: Guidelines on application of ISO 26262 to semiconductors

Part 12: Adaptation of ISO 26262 for motorcycles
Production, operation, service and decommissioning

Planning for production, operation, service and decommissioning

Production

Operation, service and decommissioning
Supporting processes

Part 1: Vocabulary
Part 2: Management of functional safety
Part 3: Concept phase
Part 4: Product development at the system level
Part 5: Product development at the hardware level
Part 6: Product development at the software level
Part 7: Production & operation
Part 8: Supporting processes
Part 9: ASIL-oriented and safety-oriented analyses
Part 10: Guideline on ISO 26262 (informative)
Part 11: Guidelines on application of ISO 26262 to semiconductors
Part 12: Adaptation of ISO 26262 for motorcycles
Supporting processes

- Interfaces within distributed developments
- Specification and management of safety requirements
- Configuration management
- Change management
- Verification
- Documentation management
- Confidence in the use of software tools
- Qualification of software components
- Evaluation of hardware elements
- Proven in use argument
- Interfacing an application that is out of scope of ISO 26262
- Integration of safety-related systems not developed according to ISO 26262
ASIL-oriented and safety-oriented analyses

Part 1: Vocabulary

Part 2: Management of functional safety

Part 3: Concept phase

Part 4: Product development at the system level

Part 7: Production & operation

Part 8: Supporting processes

Part 9: ASIL-oriented and safety-oriented analyses

Part 10: Guideline on ISO 26262 (informative)

Part 11: Guidelines on application of ISO 26262 to semiconductors
ASIL-oriented and safety-oriented analyses

1. Requirements decomposition with respect to ASIL tailoring
2. Criteria for coexistence of elements
3. Analysis of dependent failures
4. Safety analysis
Interpretation of tables (1/2)

Table 2 — Notations for software architectural design Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Informal notations</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1b Semi-formal notations</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1c Formal notations</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Alternative methods, choose *appropriate combination*

ASIL level

- o = No recommendation
- +  = Recommended
- ++ = Highly recommended

Here it is natural to choose one, i.e. Informal notation for ASIL A and B and Semi-formal notations for ASIL B, C and D, thus for ASIL B we can chose. If we want to use Informal notation for ASIL C or D we have to document a rationale.
Interpretation of tables (2/2)

Table 3 — Principles for software architectural design

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

Here most should be used, all the time, or we need to argue why not
ISO 26262 and Automotive SPICE
Automotive SPICE in a nutshell

Automotive SPICE is an adaptation of ISO 33001 for automotive domain with

...is a model / framework good practices being used throughout automotive industry. It describes “What” should be done” not “how”.

... ...is a collection of process areas of the whole product life cycle: Acquisition & Supply, Systems & Software Engineering, Support & Organization, and Project & Process Management

...is a capability model for rating and improving process capability

...provides guidance for improving the organization’s processes
Automotive SPICE in a nutshell (cont’d)

- A set of processes and process groups

- A framework to determine process capability
Automotive SPICE (Version 3.1)
## Capability Levels and Process Attributes

<table>
<thead>
<tr>
<th>Capability Level</th>
<th>Process Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 5 - Innovated</strong></td>
<td>Continuous Improvement of the Defined Process</td>
</tr>
<tr>
<td></td>
<td>PA - 5.2 Process Optimization</td>
</tr>
<tr>
<td></td>
<td>PA - 5.1 Process Innovation</td>
</tr>
<tr>
<td><strong>Level 4 - Predicted</strong></td>
<td>Predictable performance of the Defined Process</td>
</tr>
<tr>
<td></td>
<td>PA - 4.2 Quantitative Control</td>
</tr>
<tr>
<td></td>
<td>PA - 4.1 Quantitative Analysis</td>
</tr>
<tr>
<td><strong>Level 3 - Established</strong></td>
<td>Established a Defined Process tailored from a Standard Process</td>
</tr>
<tr>
<td></td>
<td>PA - 3.2 Process Deployment</td>
</tr>
<tr>
<td></td>
<td>PA - 3.1 Process Definition</td>
</tr>
<tr>
<td><strong>Level 2 - Managed</strong></td>
<td>Manage that the base practices are performed</td>
</tr>
<tr>
<td></td>
<td>PA - 2.2 Work Product Management</td>
</tr>
<tr>
<td></td>
<td>PA - 2.1 Performance Management</td>
</tr>
<tr>
<td><strong>Level 1 - Performed</strong></td>
<td>Perform all base practices</td>
</tr>
<tr>
<td></td>
<td>PA - 1.1 Process Performance</td>
</tr>
<tr>
<td><strong>Level 0 - Incomplete</strong></td>
<td></td>
</tr>
</tbody>
</table>
Purpose and scope

Maturity models

- **Purpose**
  - Improve processes based on business goals
  - Assess process capability/maturity
    - Provide assessment results that are repeatable, objective and comparable

- **Scope/Coverage**
  - Development (ASPICE, CMMI-DEV)
  - Development & oper. (SPICE, CMMI-SVC)
  - Products and services (CMMI)
  - Systems and software (SPICE)
  - Process capability/maturity
  - Process assessment (incl. method)

Functional safety standards

- **Purpose**
  - Develop safe products
  - Assess functional safety

- **Scope/Coverage**
  - Development, production, and operation
  - Safety critical E/E systems
  - Processes, methods and technical/product aspects
  - Safety integrity levels
  - Safety culture
  - Functional safety assessment
Coverage of A-SPICE and ISO 26262

A-SPICE and ISO 26262

ISO 26262 method and document requirements on processes covered by A-SPICE
E.G.: Boundary value testing

Additional process, method, and document requirements in ISO 26262
E.G.: HW process, Safety Analysis

Other ISO 26262 requirements (not process related)
E.G.: HW target values, Functional Safety Assessment
## A-SPICE support for ISO 26262

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall safety management</td>
<td>Safety management during the concept phase and the product development</td>
<td>Item definition</td>
<td>Specification of the technical safety requirements</td>
<td>Production</td>
</tr>
<tr>
<td>2-6 Safety management during the concept phase and the product development</td>
<td>2-7 Safety management after the item’s release for production</td>
<td>Initiation of the safety lifecycle</td>
<td>4-6 Functional safety assessment</td>
<td>Operation, service (maintenance and repair), and decommissioning</td>
</tr>
<tr>
<td>3-7 Hazard analysis and risk assessment</td>
<td>4-7 System design</td>
<td>3-8 Functional safety concept</td>
<td>4-8 Item integration and testing</td>
<td></td>
</tr>
<tr>
<td>3-5 Initiation of product development at the system level</td>
<td>4-9 Safety validation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-5 Initiation of product development at the hardware level</td>
<td>4-10 Functional safety assessment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-6 Specification of hardware safety requirements</td>
<td>4-11 Release for production</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-7 Hardware design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-8 Evaluation of the hardware architectural metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-9 Evaluation of the safety goal violations due to random hardware failures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-10 Hardware integration and testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-5 Initiation of product development at the software level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-6 Specification of software safety requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-7 Software architectural design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-8 Software unit design and implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-9 Software unit testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-10 Software integration and testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-11 Verification of software safety requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-5 Interfaces within distributed developments</td>
<td>8-10 Documentation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-6 Specification and management of safety requirements</td>
<td>8-11 Confidence in the use of software tools</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-7 Configuration management</td>
<td>8-12 Qualification of software components</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-8 Change management</td>
<td>8-13 Qualification of hardware components</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-9 Verification</td>
<td>8-14 Proven in use argument</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-5 Requirements decomposition with respect to ASIL tailoring</td>
<td>9-7 Analysis of dependent failures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-6 Criteria for coexistence of elements</td>
<td>9-8 Safety analyses</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Guideline on ISO 26262</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Support Levels:**
- **Strong support**
- **Medium support**
- **Weak support**
A-SPICE capability levels needed for functional safety

- ISO 26262 expects that organizational process exist that are tailored for the project => many processes have to be on capability level 3.

Reference: ISO 26262 Essentials, KMC
In summary

- A-SPICE and ISO26262
  - Has large overlap
  - No contradiction
  - A-SPICE can be seen as a prerequisite for ISO 26262

- A-SPICE
  - Focus on System and SW development processes

- ISO 26262
  - Focus on safe product
Self Assessment Exercise
<table>
<thead>
<tr>
<th>ISO 26262 Requirement</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The organization shall create, foster, and sustain a safety culture</td>
<td></td>
</tr>
<tr>
<td>2. The organization shall establish a continuous improvement process</td>
<td></td>
</tr>
<tr>
<td>3. The organization shall have an operational quality management system</td>
<td></td>
</tr>
<tr>
<td>4. A safety case shall be developed in accordance with the safety plan</td>
<td></td>
</tr>
<tr>
<td>5. An ASIL shall be determined for each hazardous event.</td>
<td></td>
</tr>
<tr>
<td>6. A safety goal shall be determined for hazardous events with an ASIL.</td>
<td></td>
</tr>
<tr>
<td>7. The functional safety requirements shall be derived from the safety goals</td>
<td></td>
</tr>
<tr>
<td>8. The technical safety requirements shall specify necessary safety mechanisms</td>
<td></td>
</tr>
<tr>
<td>9. Safety analyses on the system design to identify causes of systematic failures</td>
<td></td>
</tr>
<tr>
<td>10. Diagnostic coverage of safety-related hardware elements shall be estimated</td>
<td></td>
</tr>
<tr>
<td>11. Software architectural design described with appropriate levels of abstraction</td>
<td></td>
</tr>
<tr>
<td>12. Every safety-related software component shall be categorized</td>
<td></td>
</tr>
</tbody>
</table>
Implementing ISO 26262
The survey was sent out to over 600 professionals in the automotive industry. 90% of companies surveyed have at least started to implement the ISO 26262 but only 1/5 of those have managed to fully implement the standard into their processes.

20% Fully Implemented
44% Mostly Implemented
36% Starting to be Implemented

What is highest ASIL category your company is dealing with?

0 % ASIL A
15.38% ASIL B
3.85% ASIL C
80.77% ASIL D

What Part of the ISO 26262 does your organization find most challenging?

32.14% Safety Management for the Organization
3.57% Concept Phase
32.14% System Development with Technical Safety Concepts
7.14% Product Development at the Hardware Level
10.71% Product Development at the Software Level
14.29% ASIL & Safety-Oriented Analysis (ex. FTA)

“Made us think more about human factors.”
Safety/ISO 26262 specific challenges

- Establish safety culture
- A-SPICE Level-3 capability needed for many processes
- Safety analysis techniques, e.g. HARA, FTA, FMEA
- Design for safety, i.e. design patterns, HW/SW design
- Test methods, e.g., fault injection, struct coverage, and equivalence classes
- Handling/qualification of legacy systems, SW&HW components, tools
- Development of safety case
- Functional safety assessment
- Many organizational parts involved
  - System, SW, HW, Test, Production, Legal, Sourcing
A typical improvement journey

Why?  What?  When?  How much?

Business goals
Improvement goals
Improvement metrics

Where are we today?

Appraisal results
Measurements

How?

Improvement approach
Improvement plan

Just do it…

Change management

© Addalot Consulting AB - All rights reserved
© Addalot Consulting AB - All rights reserved
Common Improvement Pitfalls

- Improvement goals are not aligned with business goals
- Management not committed to Improvement
  - Adequate resources not provided
  - Premature delegation of process improvement responsibilities
- Process Theory
  - Improvement run from an Process Group away from projects
  - Neglecting existing practices
  - Lots of diagrams but little content
- Overconfidence in or misinterpretation of models
  - There are no "silver bullets"
  - The check list syndrome
- Everything done at the same time - big bang strategy
- Neglecting the “human side” of the change
  - People change not organizations
Specific recommendations

- Takes time…. “1-step” in A-Spice takes 9-12 months at 5-10% of the engineering capacity

- Don’t separate A-SPICE/ISO implementation
  - Same Standard process and same people

- Take it in steps - what order? …It depends…
  - Establish and ensure usage of standard process
  - Initiate Safety culture/activities

- ISO 26262 require more top down

- Process deployment >> Process definition

- Drive introduction as project with clear goals and follow up

- Don’t neglect emotional aspects – “what’s in it for me”

- Communication
Prerequisites for change

- Management Commitment
- Awareness
- Motivation
- Time & Resources
- Responsibilities & Authorities
- Monitoring
- Knowledge
- Stakeholder Involvement
- Communication
- Adherence follow-up

Successful Change
Summary
Summary

- ISO 26262 is expected by the Automotive industry as “State of the art”
- Extensive standard covering several areas:
  - Required degree of safety measures:
  - Fit well together with Automotive SPICE
  - Challenge for Improvement to be successful

\[
\text{ASIL} = \text{Severity} \times \text{Exposure} \times \text{Controllability}
\]
“Excellent firms don't believe in excellence - only in constant improvement and change.”

In Search of Excellence - Tom Peters