ISO 26262 and AUTOSAR. Requirements and Solutions for Safety Related Software.

Simon Fürst.
5th Vector Congress.
1st and 2nd December 2010, Stuttgart.

BMW Group
ISO 26262 and AUTOSAR.

Preamble.

- ISO 26262 address the development of safety-related systems
  - AUTOSAR is “only” the infrastructure part of the software of a system

- ISO 26262 contains two different kind of requirements
  - Process related requirements: “how to develop the system”
  - Technical requirements: “what system to develop”
  - The AUTOSAR development partnership as well as the implementers of AUTOSAR have to respect both

- AUTOSAR only provides specifications
  - Only a subset of the requirements of ISO 26262 are applicable
  - For the implementers of AUTOSAR an overlapping subset of the requirements of ISO 26262 are applicable
What has AUTOSAR done so far?
ISO 26262 and AUTOSAR.

AUTOSAR’s basic approach.

Virtual Integration
Independent of hardware
Virtual Functional Bus

Introduction of HW Attributes
Holistic view of the entire system, both software and hardware

ECU Configuration
Run-Time Environment
Separation of system into its ECU (plus common infrastructure)
ISO 26262 and AUTOSAR

AUTOSAR Methodology.

<table>
<thead>
<tr>
<th>AUTOSAR System Configuration Generator</th>
<th>Component API Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU Resource Description (HW only)</td>
<td>SW-Component Description</td>
</tr>
<tr>
<td>System - Constraint Description</td>
<td></td>
</tr>
<tr>
<td>Decisions (e.g. mapping)</td>
<td>Component API e.g. app.h</td>
</tr>
</tbody>
</table>

Components:
- API Generator
- ECU Configuration Generator
- System Configuration Generator
- SW-C Implementation Generator

Generation steps:
- Decisions (e.g. mapping)
- ECU extract of System Configuration
- System Configuration Description
- AUTOSAR ECU Configuration Generator
- AUTOSAR RTE Generator
- OS, COM, … Generator
- Other Basic SW Generator
- MCAL – Generator

Information / Database (no files)
- AUTOSAR RTE
- OS, COM, …
- Other Basic SW
- MCAL

System per ECU

- List of implementations of SW components
- RTE extract of ECU configuration
- OS extract of ECU configuration e.g. OIL
- Basic SW Module A extract of ECU configuration
- Basic SW

Decisions (e.g. scheduling, …)
ISO 26262 und AUTOSAR.
Approach of AUTOSAR w.r.t. Functional Safety.

Sources
- ISO WD 26262
- Requirements from WPs & WGs
- Requirements from Applications
- Requirements from Safety Concepts

Consolidated Safety Requirements
- Process Safety Requirements
  - AUTOSAR Safety Guidelines
- Technical Safety Requirements
  - Interface Class 1
  - Interface Class 3
- Methodology Safety Requirements
  - Tools
  - Generation

Structure and Allocation
- Development Process
  - BSW & RTE Requirements
    - SRS
    - SWS
  - Tools
- List of requirements on development processes
- List of safety requirements allocated to BSW & RTE

Assignment
- Tools and Generation Process
  - Tools
  - Generation
- Update of existing documents of WPs
- Requirements on tools and generation process
ISO 26262 and AUTOSAR.
Overview of available, built-in AUTOSAR safety mechanisms.

– Built-in self test mechanisms for detecting hardware faults (testing and monitoring)

– Run-time mechanisms for detecting software faults during the execution of software
  – Program flow monitoring

– Run-time mechanisms for preventing fault interference
  – Memory partitioning for SW-Cs
  – Time partitioning for applications

– Run-time mechanisms for protecting the communication
  – End-to-end (E2E) communication protection for SW-Cs

– Run-time mechanisms for error handling
ISO 26262 and AUTOSAR.
Built in safety mechanisms for detecting errors.

- **Memory:**
  - RAM Test
  - Flash Test
  - Support for ECC memory

- **Core:**
  - Core Test

- **Logical and temporal program flow monitoring**
– Detected errors in the basic software:
  – Are reported through DEM to SW-Cs. SW-Cs then executes application-specific actions
  – Are reported to FIM, which permits to disable some functions of SW-C

– Detected hardware errors:
  – Arithmetic exceptions (e.g. division by 0): handled by OS callouts (small error handling routines in the context of basic software). Typical reaction – ECU reset
  – HW errors detected by HW testing: handled by callouts. Typical reaction – ECU reset
  – Errors detected by MMU/MPU (memory and time partitioning). It will shut down or restart the faulty SW-C partition
ISO 26262 and AUTOSAR. Memory partitioning for SW-Cs.

- Enables create protection boundaries around groups of SW-Cs
- This is realized by user-mode/non-trusted memory partitions (for groups of SW-Cs)
- This protects from interference: (1) basic software and (2) SW-Cs in other partitions
- Basic software is not partitioned. It runs with in CPU supervisor mode with full access to memory, CPU and all other hardware resources
ISO 26262 and AUTOSAR.

End-to-End communication protection (1/2).

- E2E protection detects faults in data caused by both hardware and in software

Typical sources of interferences, causing errors detected by E2E protection:

SW-related sources:
S1. Error in mostly generated RTE, S2. Error in partially generated and partially hand-coded COM, S3. Error in network stack, S4. Error in generated IOC or OS

HW-related sources:
ISO 26262 and AUTOSAR.
End-to-End communication protection (2/2).

- Protection of data exchanged over communication channels like FlexRay and CAN
- Failure modes addressed as defined by ISO DIS 26262 for communication (repetition, deletion, insertion, incorrect sequence, corruption, timing faults, addressing faults, inconsistency, masquerading)
- Three different protection mechanisms for data are used
  - CRC, counter, Data ID, timeout detection
  - Data ID included in to calculated CRC, but not sent

CRC := CRC8 over (1) Data Id, (2) all serialized signal (including empty areas, excluding CRC byte itself)
ISO 26262 and AUTOSAR.  
Safety mechanisms supported by AUTOSAR.

- Implementation of typical safety concepts in the automotive domain
  - Intelligent HW watchdog (ASIC) / 3-level safety concept
  - Monitored channel (2 µCs, the second is a simple µC monitoring the first µC)
  - Dual channel (2 AUTOSAR µCs)

- Application redundancy (on the same or different µCs)

- Basic Software redundancy inside one ECU
ISO 26262 and AUTOSAR.
Application redundancy.

- Assuming integrity of HW/ECU and AUTOSAR Basic Software implementation, SW redundancy with ASIL decomposition can be used within the same ECU.
- Distribution of SW channels across ECUs is also possible.
Basic Software redundancy inside one ECU.

- Redundancy inside AUTOSAR e.g. double input/output data paths through
  - Redundant IO hardware abstraction and IO drivers
  - Redundant and diverse (e.g. ADC + DIO, internal ADC + external ADC)
- Redundancy through integration of complex drivers running on the same µC offering a redundant data path
What is ISO 26262 saying?
ISO 26262 and AUTOSAR.
Safety Element out of Context.

Idea
- A SEooC is a pre-qualified safety element that is developed independently from an item development
- A SEooC must be usable in an item development while taking benefit from the work done during the pre-qualification

Definition
A Safety Element out of Context (SEooC) is a safety element for which an item does not exist at the time of the development. A SEooC can either be a subsystem, a software component, or a hardware component.
- A SEooC is never an item.
- A SEooC can either be a subsystem, a hardware component, or a software component.
- Typically, requirements at higher levels remain in the status "assumed" (see ISO°26262-8, Clause°5) and will be confirmed when the SEooC is used in an item development.
- The correct implementation of the assumed requirements will be verified during the SEooC development, but the validation takes place during the item development. The development of a SEooC starts at a certain level of requirements and design, and all information on requirements or design prerequisites, are pre-determined in the status "assumed".
- Non-functional requirements might be in the status "assumed" at the same level of requirements where functional ones are in the status "accepted".
ISO 26262 and AUTOSAR.
SEooC and configurable software.

Configuration of an AUTOSAR Basic Software Stack is heavily based on
- Configuration data and
- Calibration data
Configuration and calibration data is fully described in standardized XML templates
Excerpt from chapter 7 “Software architectural design”

7.4.3 The software architectural design shall exhibit the following properties by use of the principles listed in Table 4:

a) modularity;
b) encapsulation and;
c) minimum complexity.

Table 4 — 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>
### Table 5 — Mechanisms for error detection at 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 Plausibility check(^a)</td>
<td>++</td>
</tr>
<tr>
<td>1b Detection of data errors(^b)</td>
<td>+</td>
</tr>
<tr>
<td>1c External monitoring facility</td>
<td>0</td>
</tr>
<tr>
<td>1d Control flow monitoring</td>
<td>0</td>
</tr>
<tr>
<td>1e Diverse software design(^c)</td>
<td>0</td>
</tr>
</tbody>
</table>

\(^a\) Plausibility checks include assertion checks. Complex plausibility checks can be realised by using a reference model of the desired behaviour.

\(^b\) Types of methods that may be used to detect data errors include error detecting codes and multiple data storage.

\(^c\) Diverse software design is not intended to imply n-version programming.

### Table 6 — Mechanisms for error handling at 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 Static recovery mechanism(^a)</td>
<td>+</td>
</tr>
<tr>
<td>1b Graceful degradation(^b)</td>
<td>+</td>
</tr>
<tr>
<td>1c Independent parallel redundancy(^c)</td>
<td>0</td>
</tr>
<tr>
<td>1d Correcting codes for data</td>
<td>+</td>
</tr>
</tbody>
</table>

\(^a\) Static recovery mechanisms can be realised by recovery blocks, backward recovery, forward recovery and recovery through repetition.

\(^b\) Graceful degradation at the software level refers to prioritising functions to minimise the adverse effects of potential failures on functional safety.

\(^c\) For parallel redundancy to be independent there has to be dissimilar software in each parallel path.
# ISO 26262 and AUTOSAR

## Safety lifecycle for AUTOSAR as SEooC.

### 1. Vocabulary

### 2. Management of functional safety

| 2-5 Overall safety management | 2-6 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: system level

| 4-5 Initiation of product development at the system level | 4-6 Specification of the technical safety requirements | 4-7 System design | 4-8 Item integration and testing |

### 5. Product development: hardware level

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

### 6. Product development: 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 software safety requirements |

### 7. Production and operation

| 7-5 Production | 7-5 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 Qualification 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 (informative)
ISO 26262 and AUTOSAR.
Safety lifecycle for Implementers of AUTOSAR.

1. Vocabulary

2. Management of functional safety
   - 2-5 Overall safety management
   - 2-6 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: system level
   - 4-5 Initiation of product development at the system level
   - 4-6 Specification of the technical safety requirements
   - 4-7 System design
   - 4-8 Item integration and testing

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

6. Product development: 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 software safety requirements

7. Production and operation
   - 7-5 Production
   - 7-5 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 Qualification 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 (informative)
ISO 26262 and AUTOSAR.

**Conclusion.**

- AUTOSAR systematically derived safety mechanisms supported in R4.0
- AUTOSAR provides support for dedicated safety mechanisms with generic fault models
- Implementers have to tailor ISO 26262 according to their activities in the safety-lifecycle
- For all implemented safety mechanisms a safety manual is needed containing
  - The fault model according to which the safety mechanism was developed
  - The constraints that must be fulfilled when applying a safety mechanism
- During system and software design the safety manual is considered to appropriately use the safety mechanisms of an AUTOSAR implementation.
- **Safety related systems can by build with AUTOSAR compliant systems.**
ISO 26262 und AUTOSAR.
Thank you for your attention.
ISO 26262 and AUTOSAR.

Requirements, design and test phases.