

# How to apply Functional Safety to AUTOSAR ECUs

Joachim Kalmbach - Sr. Systems Engineer

Session ID: 0C041A

DEVCON 2015  
Accelerate. Innovate. Differentiate.

vector

# Joachim Kalmbach

---

Joachim Kalmbach is currently a Senior Systems Engineer for Vector CANtech in Novi, MI.

In his role, he supports Vector's North America customers in selecting AUTOSAR solutions.

Before joining Vector CANtech, he held for 8 years the position of Product Management Engineer specializing in AUTOSAR for Vector Informatik in Stuttgart, Germany.

His responsibilities included the development of Vector's Safety and Multi Core Solutions.

Kalmbach graduated in 2006 from the Reutlingen University of Applied Sciences as a Diplom Ingenieur (FH).







# How to apply Functional Safety to AUTOSAR ECUs

---

- You have received safety Requirements from your OEM?
  - You have to develop your ECU according ISO26262?
  - You have to integrate Software with different ASILs?
- Don't worry AUTOSAR provides features which support you!
- Vector provides a ready to use AUTOSAR solution for your ECUs
- Accepted by functional safety engineers at many OEMs
    - This minimizes effort for design of the functional safety concept
  - Design and development is certified by German TÜV
    - This minimizes effort for your qualification and certification

# How to apply Functional Safety to AUTOSAR ECUs

## Agenda:

- ISO26262 Road vehicles - Functional safety
- Functional Safety and AUTOSAR
- MICROSAR Safe – Vector's AUTOSAR Functional Safety Solution

## ISO26262 Road vehicles - Functional safety



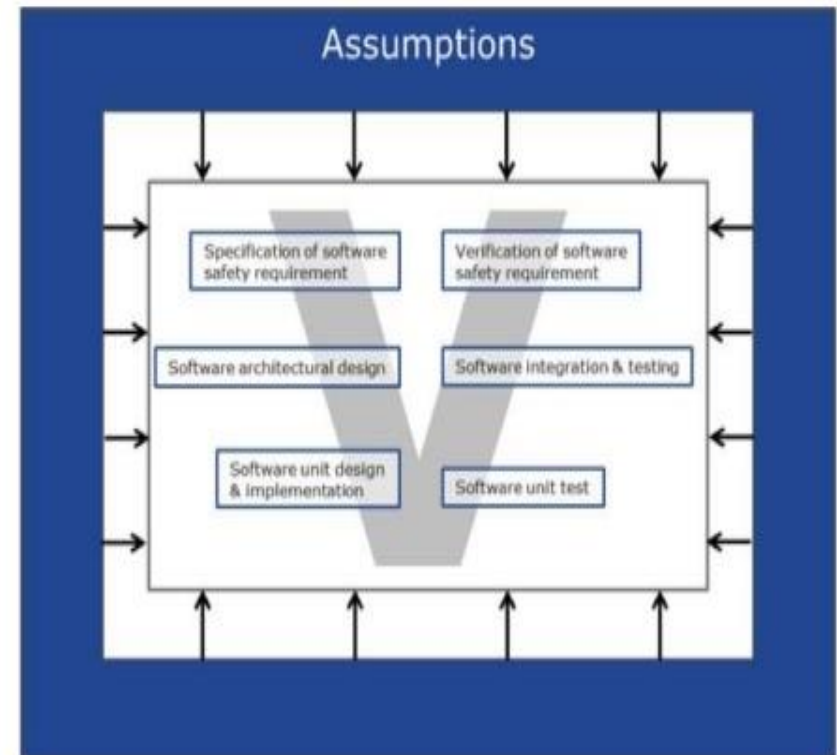
# Safety Element out of Context - SEooC

---

- How to develop generic software components (e.g. AUTOSAR Standard Software) for which:
  - Environment is unknown
  - Safety Goals are unknown
    - Safety Requirements can't be derived

# Safety Element out of Context - SEooC

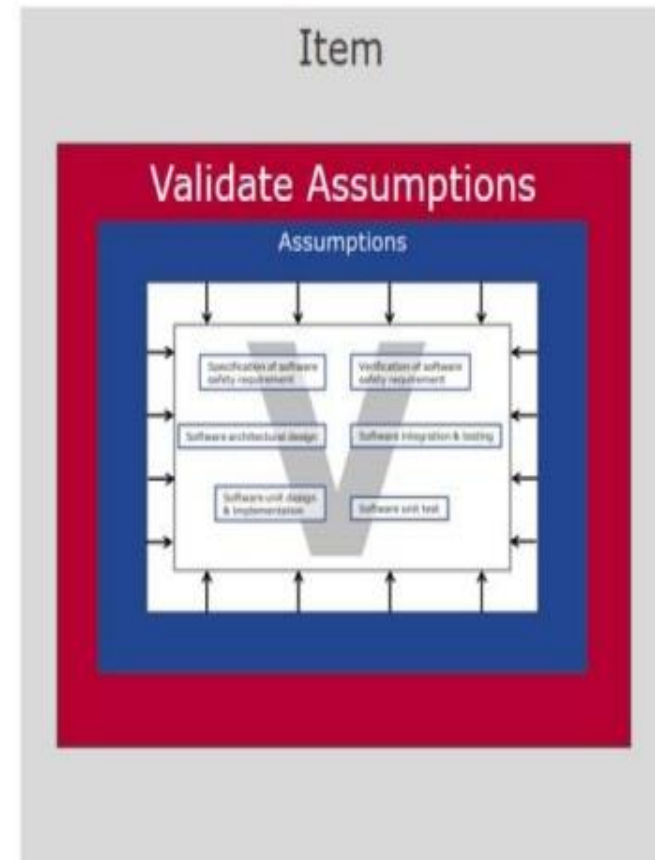
- The ISO 26262 defines SEooC – Safety Element out of Context
  - Generic element  
(e.g. subsystem, software component, hardware part)
  - No specific use case
  - Used within safety context
- Assumptions on Safety Requirements
- Development of the SEooC according to these Assumptions



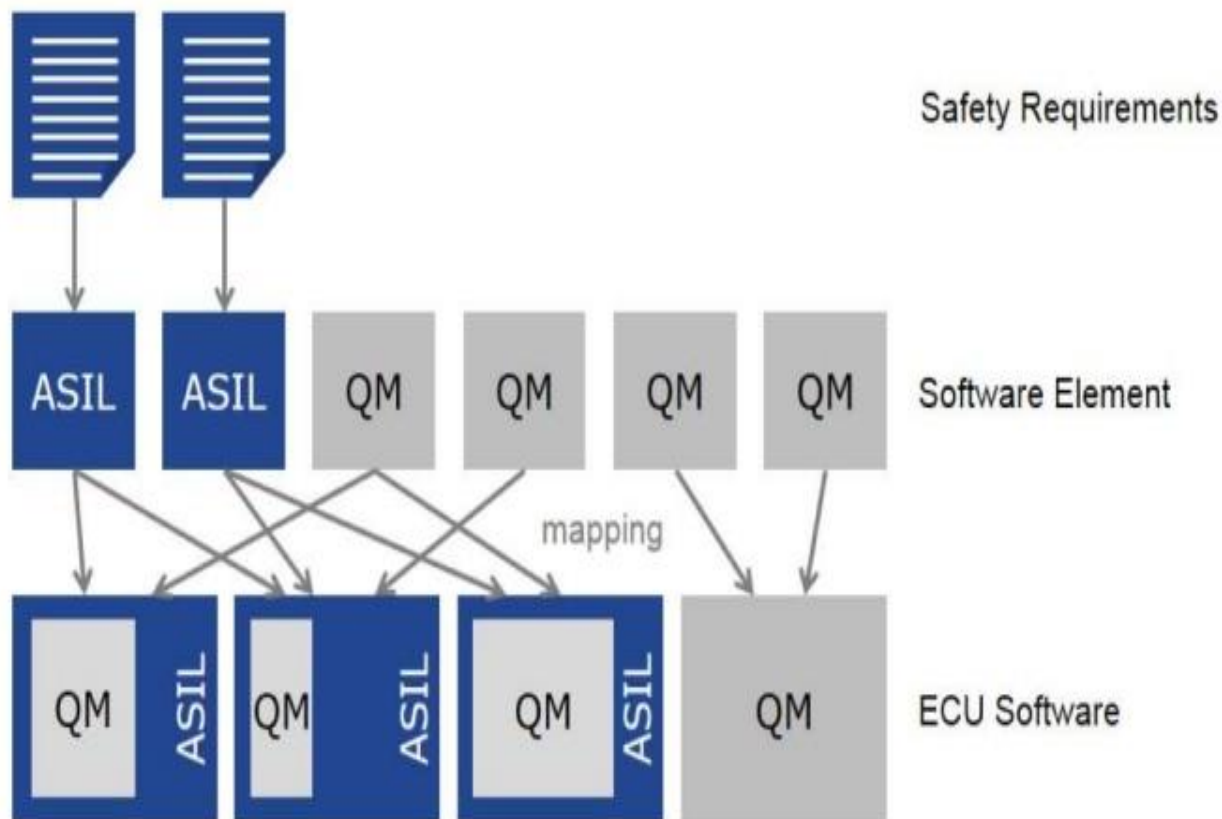


# Safety Element out of Context - SEooC

- Assumptions provided in Safety Manual
- Validation of the requirements when integrating SEooC into the item

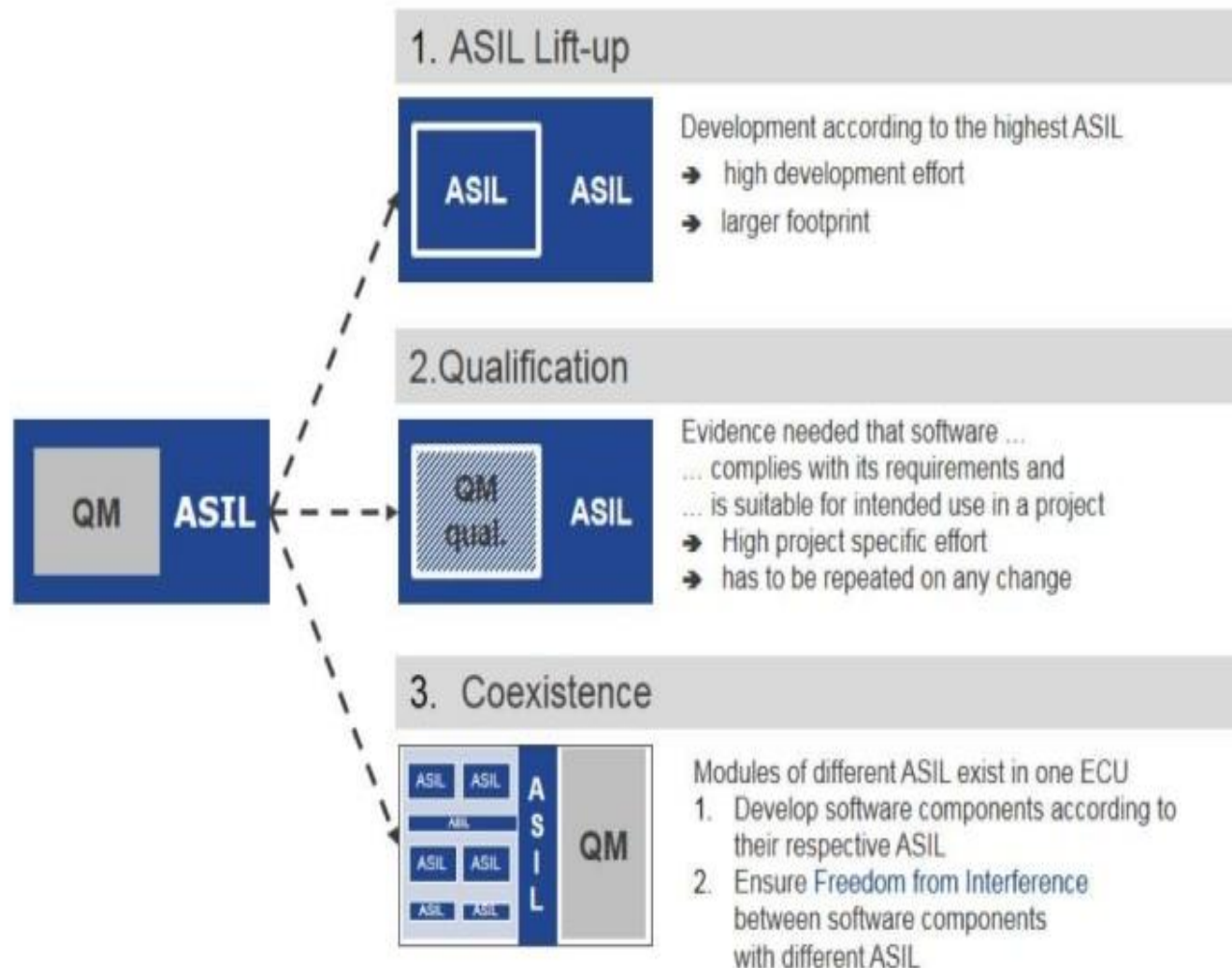


# Mapping of Safety Requirements

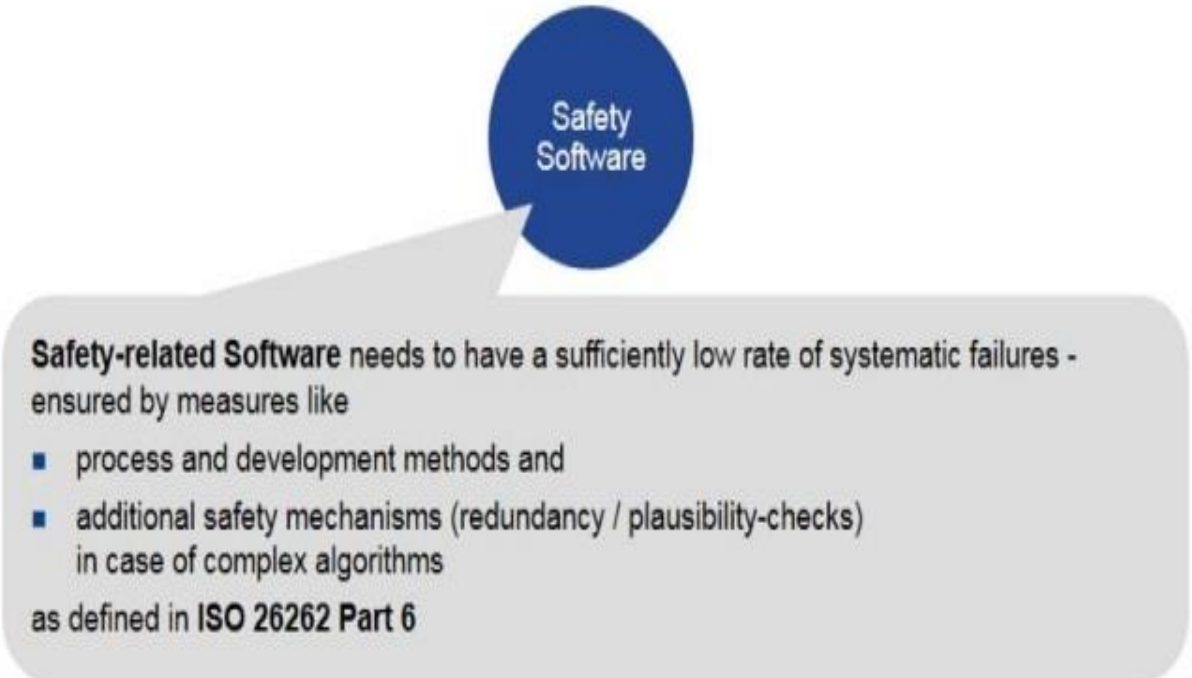


➔ **Mixed ASIL** approach  
is often the basis of the ECU Safety Concept

# Development Approaches for Mixed ASIL Systems



# Freedom from Interference



Safety  
Software

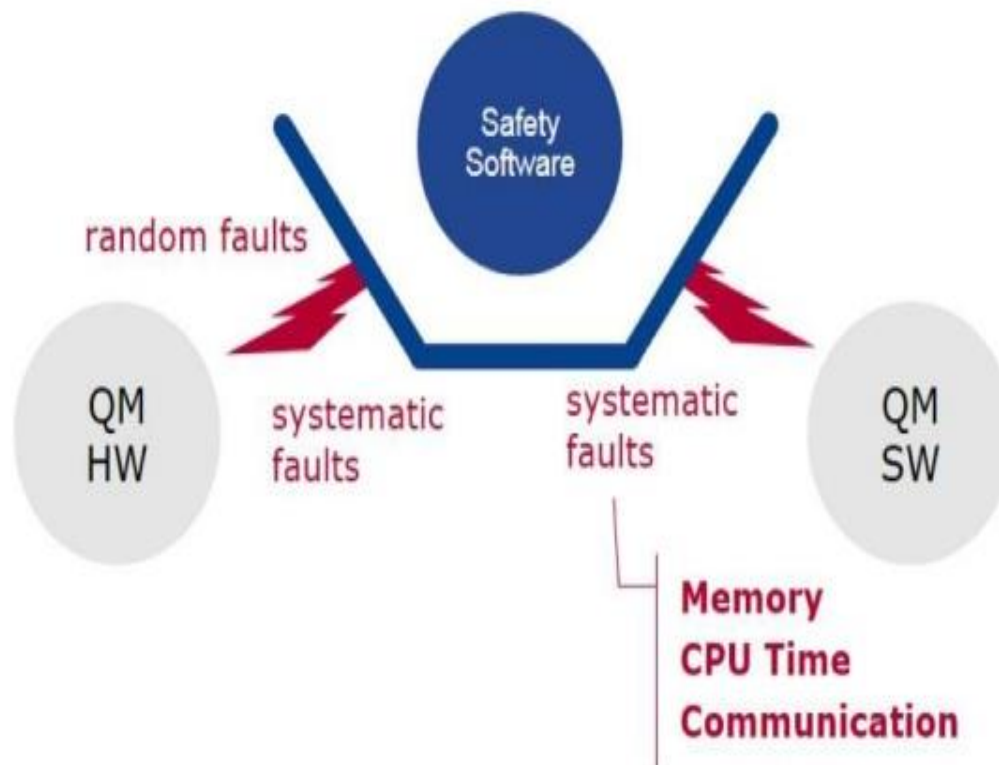
**Safety-related Software** needs to have a sufficiently low rate of systematic failures - ensured by measures like

- process and development methods and
- additional safety mechanisms (redundancy / plausibility-checks)  
in case of complex algorithms

as defined in **ISO 26262 Part 6**



# Freedom from Interference



# Freedom from Interference

---

## ■ Memory:

- memory corruption due to unintended writing to memory of another partition

## ■ CPU Time:

- blocking of partitions due to communication deadlocks
- wrong allocation of processor execution time

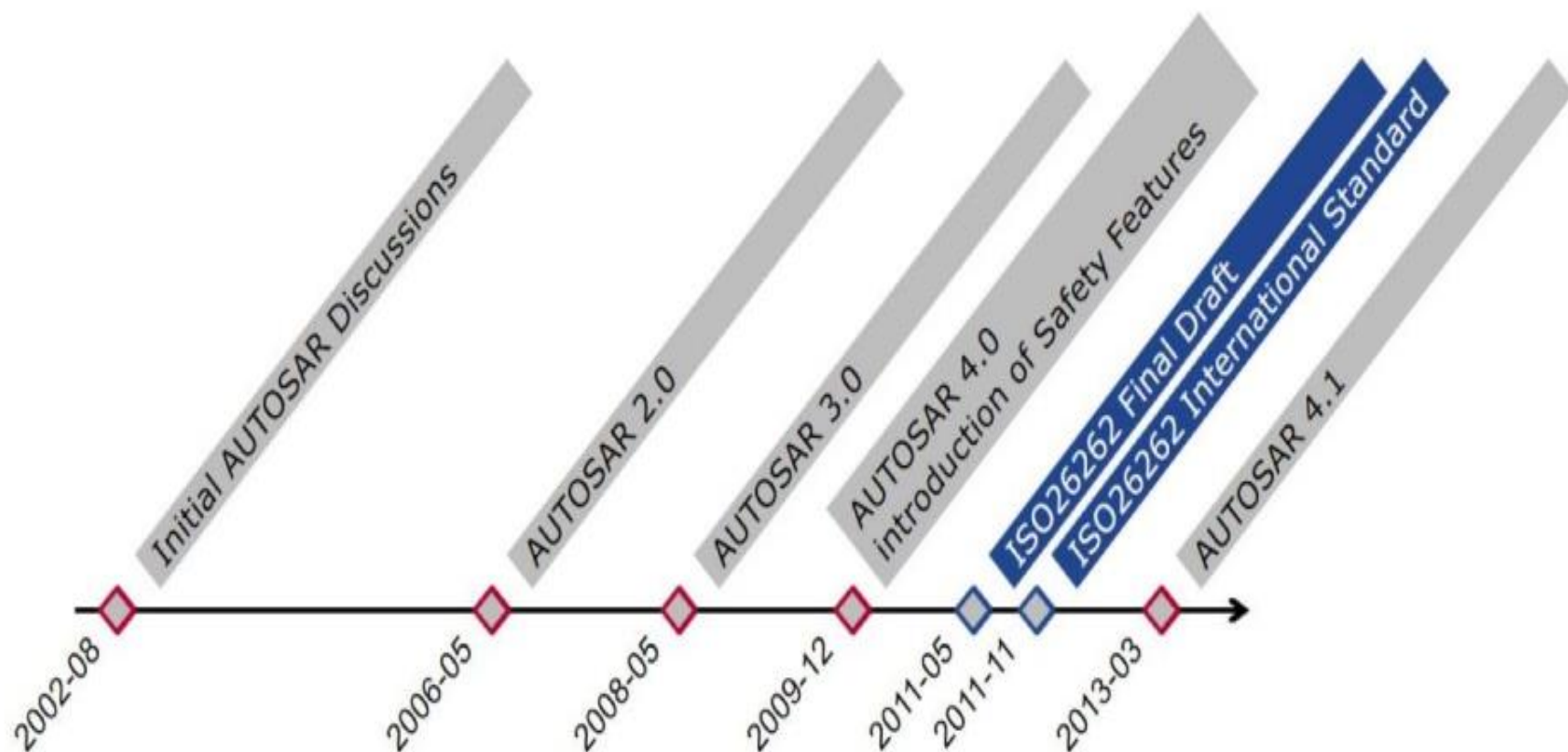
## ■ Communication:

- loss of communication
- unintended repetition
- insertion of messages
- re-sequencing
- message corruption
- message delay
- blocking access to data bus / "babbling idiot"

## Functional Safety and AUTOSAR



# Timeline AUTOSAR and ISO26262





# Timeline AUTOSAR and ISO26262

---

- AUTOSAR started long before the ISO26262 was released
  - The first versions of AUTOSAR were developed without taking any safety standard into account
- Defining Safety Requirements in AUTOSAR 4
- Introduction of Safety Features with AUTOSAR 4
- AUTOSAR specifies Features which can be used to ensure Freedom from Interference

# AUTOSAR Features which supports Freedom from Interference

---

## Memory

- memory corruption due to unintended writing to memory of another partition

## CPU Time

- blocking of partitions due to communication deadlocks
- wrong allocation of processor execution time

## Communication

- loss of communication
- unintended repetition
- insertion of messages
- re-sequencing
- message corruption
- message delay
- blocking access to data bus / "babbling idiot"

# AUTOSAR Features which supports Freedom from Interference

	Memory	CPU Time	Communication
Watchdog	Alive Supervision Program Flow Monitoring		
OS	Scalability Class 3 and 4 Memory Protection	Scalability Class 2 and 4 Timing Protection	
End 2 End Protection	Detect Faults in Communication Link		

# AUTOSAR Features which supports Freedom from Interference

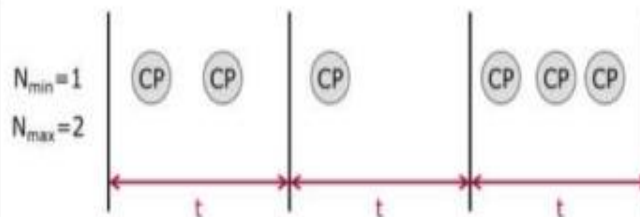
	Memory	CPU Time	Communication
Watchdog	Alive Supervision Program Flow Monitoring		
OS	Scalability Class 3 and 4 Memory Protection	Scalability Class 2 and 4 Timing Protection	
End 2 End Protection	Detect Faults in Communication Link		



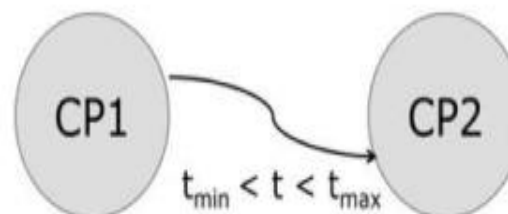
# AUTOSAR Watchdog Manager

- The Watchdog Manager provides three mechanisms:

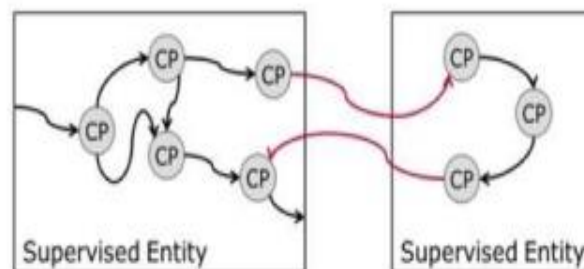
Alive supervision



Deadline monitoring

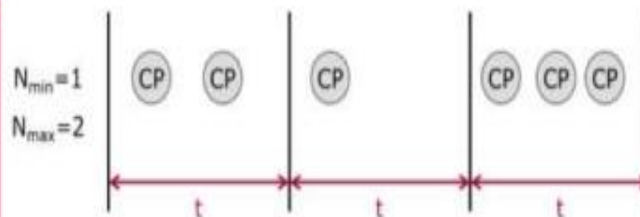


Logical monitoring

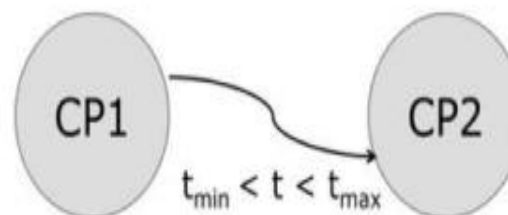


# AUTOSAR Watchdog Manager

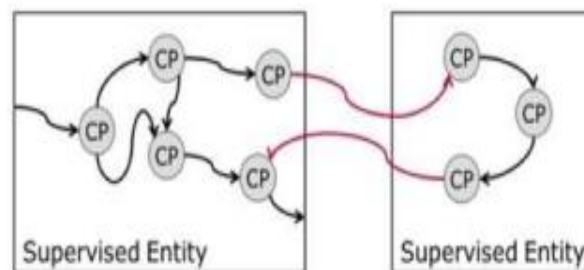
Alive supervision



Deadline monitoring

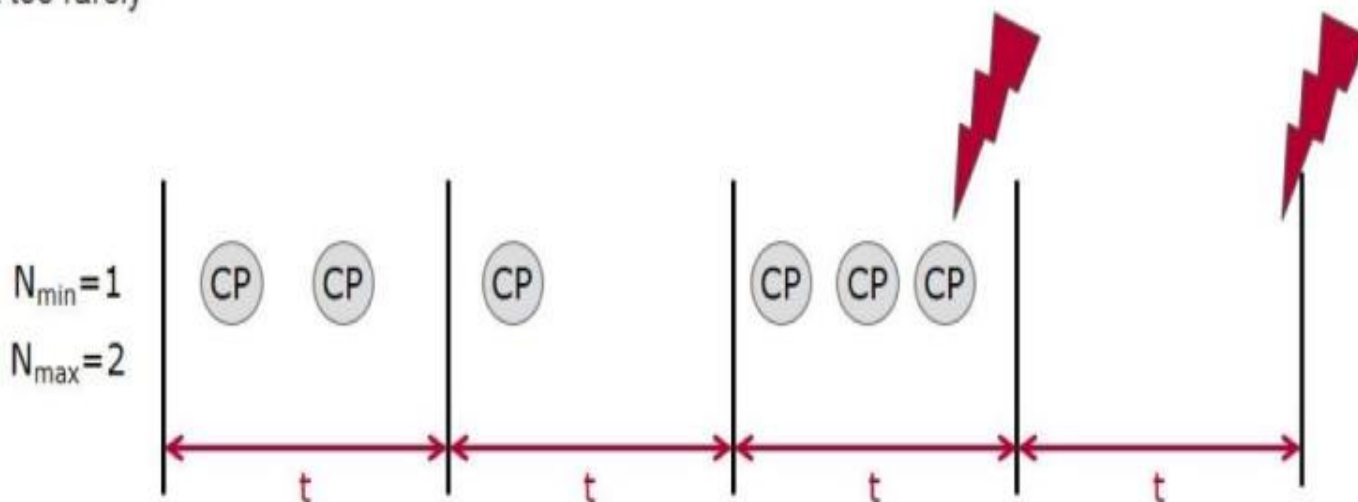


Logical monitoring



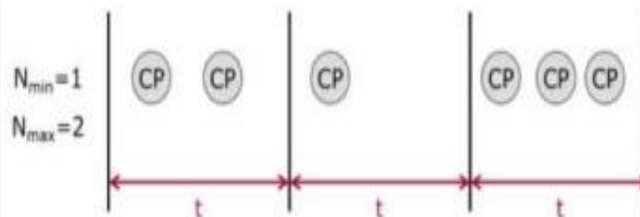
# Alive supervision

- Periodic Supervised Entities have constraints on:
  - the number of times they are executed within a given time span
- The Watchdog Manager checks periodically if the
  - Checkpoints of a Supervised Entity have been reached within the given limits
    - not too frequently
    - not too rarely

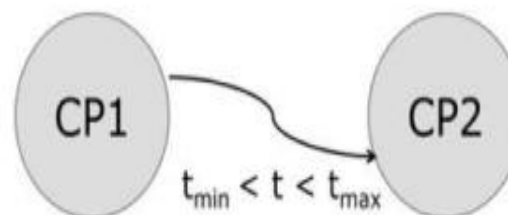


# AUTOSAR Watchdog Manager

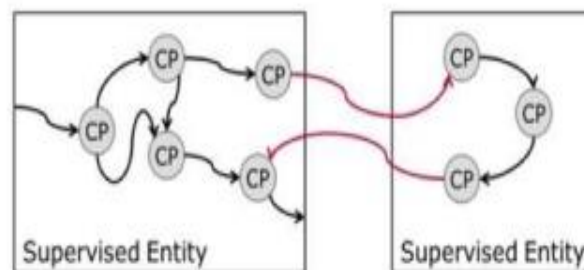
Alive supervision



Deadline monitoring



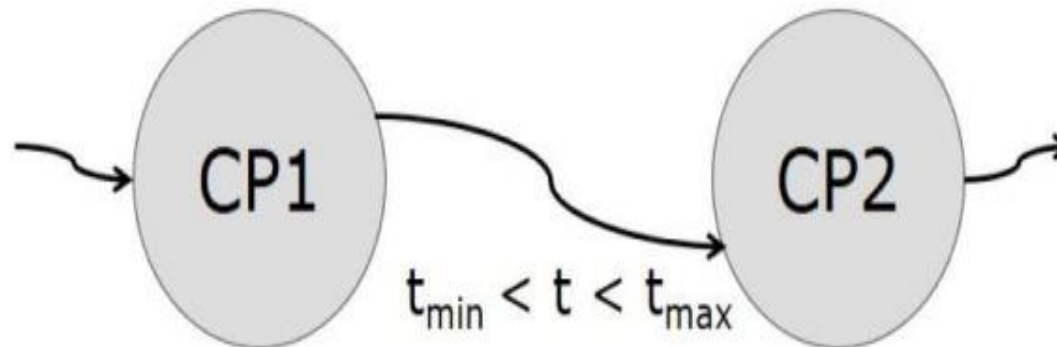
Logical monitoring





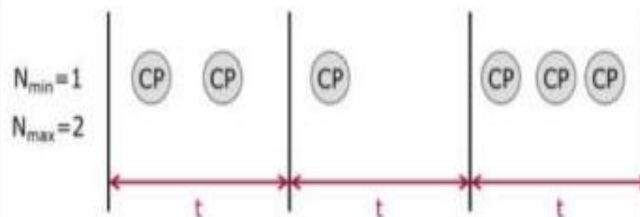
# Deadline monitoring

- Aperiodic or episodically Supervised Entities have individual constraints on:
  - the timing between two Checkpoints
- The Watchdog Manager checks if
  - the timing of transitions between two Checkpoints of a Supervised Entity is within the configured minimum and maximum

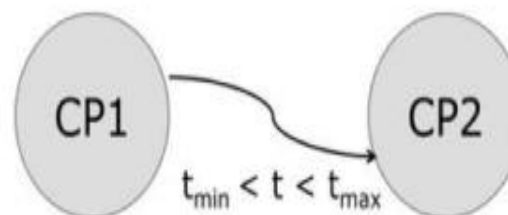


# AUTOSAR Watchdog Manager

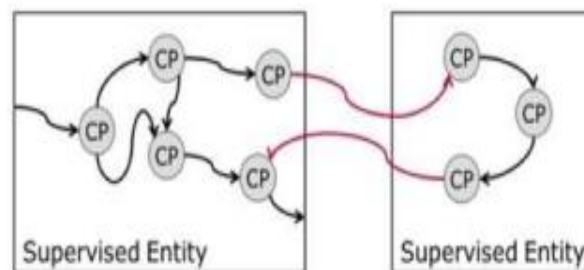
Alive supervision



Deadline monitoring

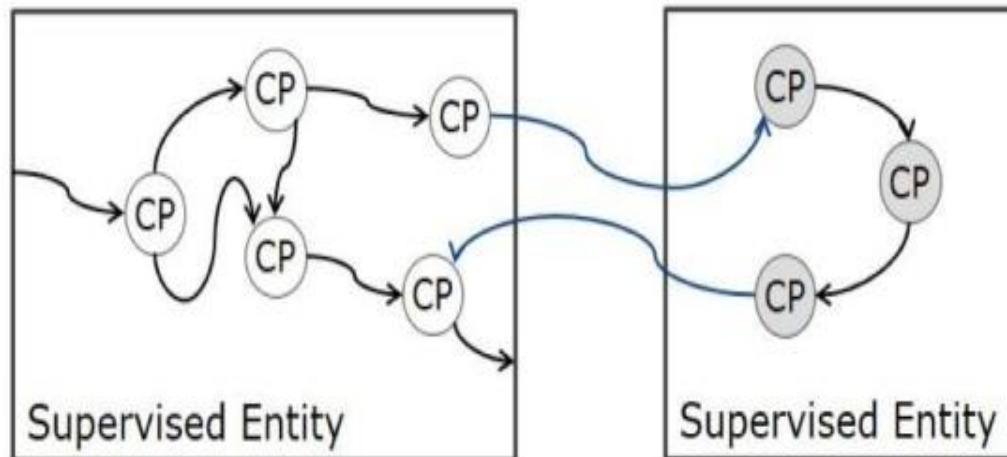


Logical monitoring



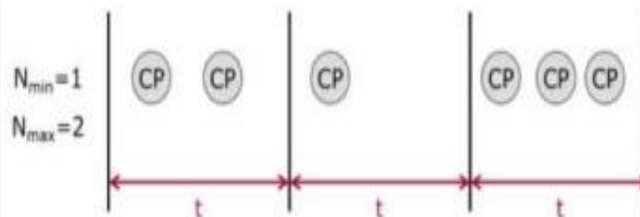
# Logical monitoring

- Logical supervision focuses on control flow errors, which cause a divergence from the valid program sequence during the error-free execution of the application.  
An incorrect control flow occurs if one or more program instructions are processed either in the incorrect sequence or are not even processed at all.
- The Watchdog Manager checks if
  - the transition from the last reported checkpoint to the reported checkpoint is
    - Allowed
    - Not Allowed

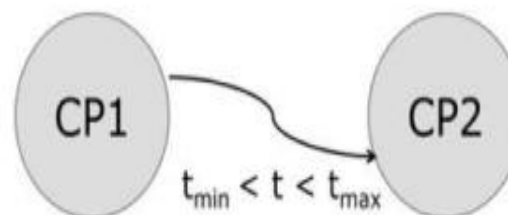


# AUTOSAR Watchdog Manager

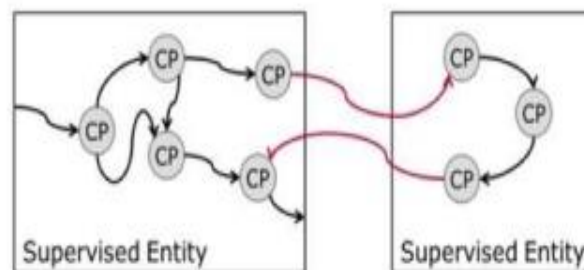
Alive supervision



Deadline monitoring

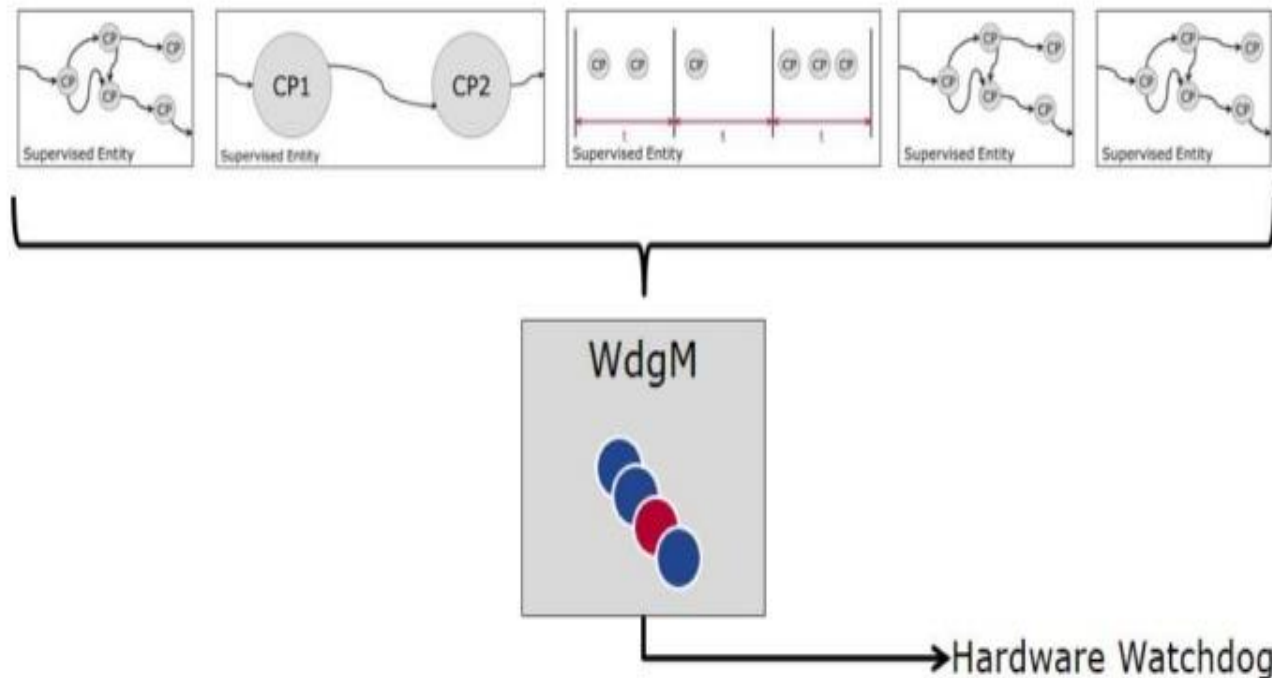


Logical monitoring



# Watchdog Manager

- Watchdog Manager combines all Supervised Entity states to a system state
- Depending on the system State the Watchdog Manager triggers the Watchdog Driver



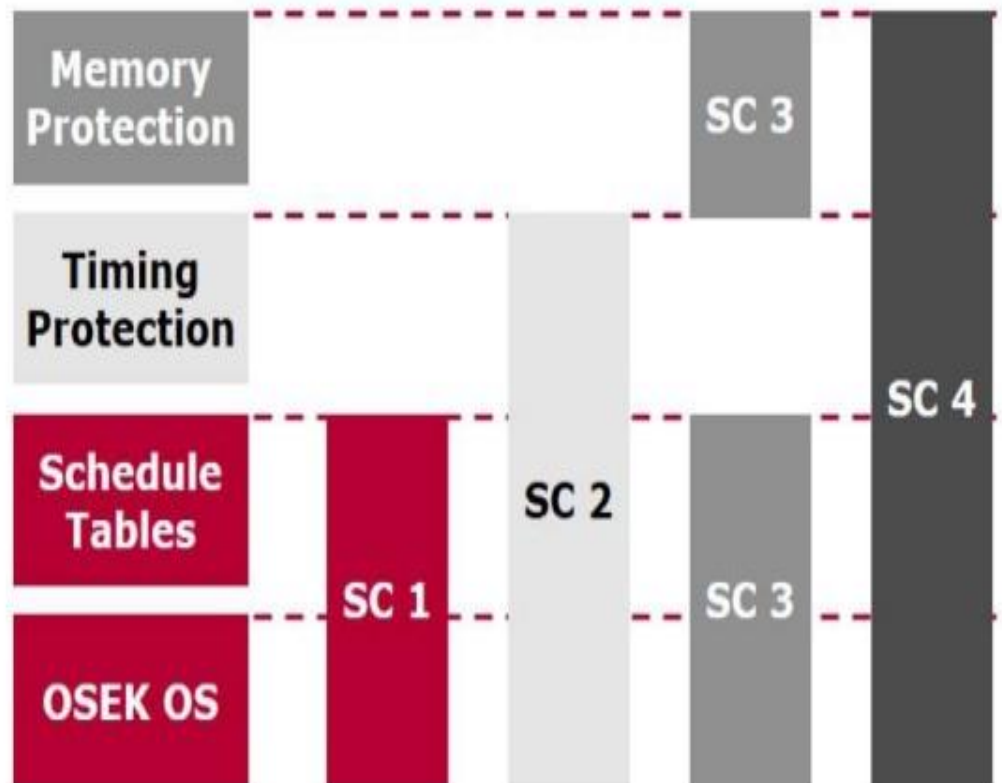


# AUTOSAR Features which supports Freedom from Interference

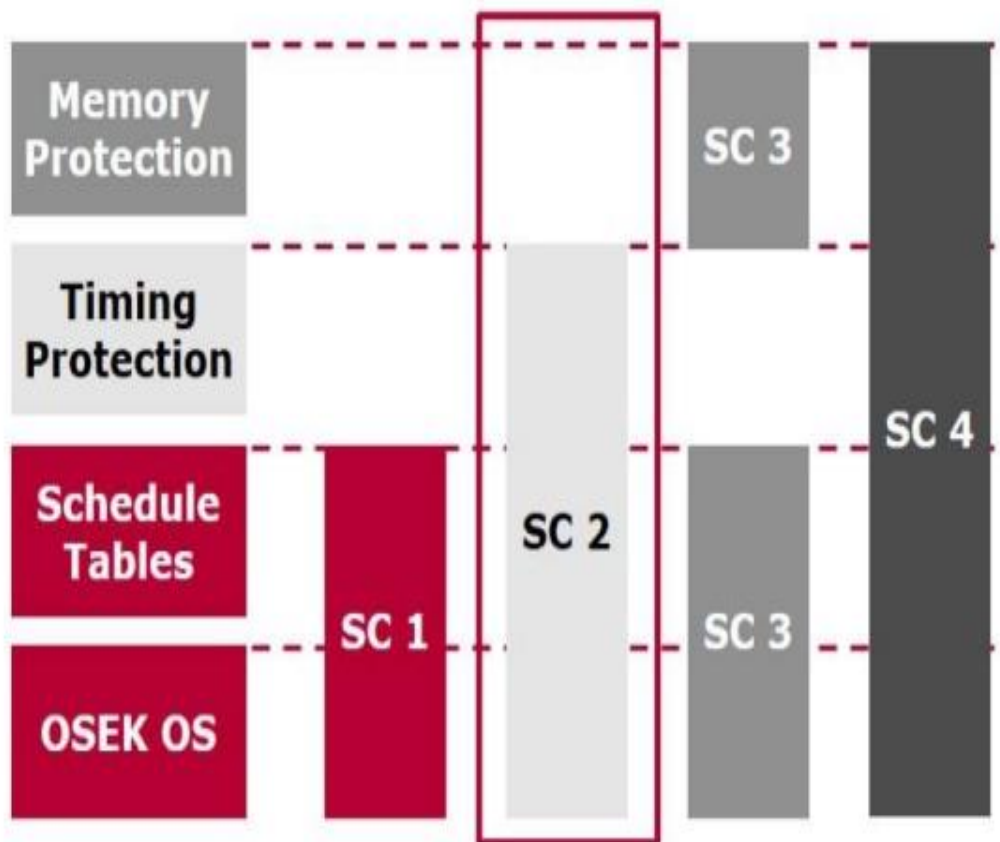
	Memory	CPU Time	Communication
Watchdog	Alive Supervision Program Flow Monitoring		
OS	Scalability Class 3 and 4 Memory Protection	Scalability Class 2 and 4 Timing Protection	
End 2 End Protection	Detect Faults in Communication Link		

# AUTOSAR OS

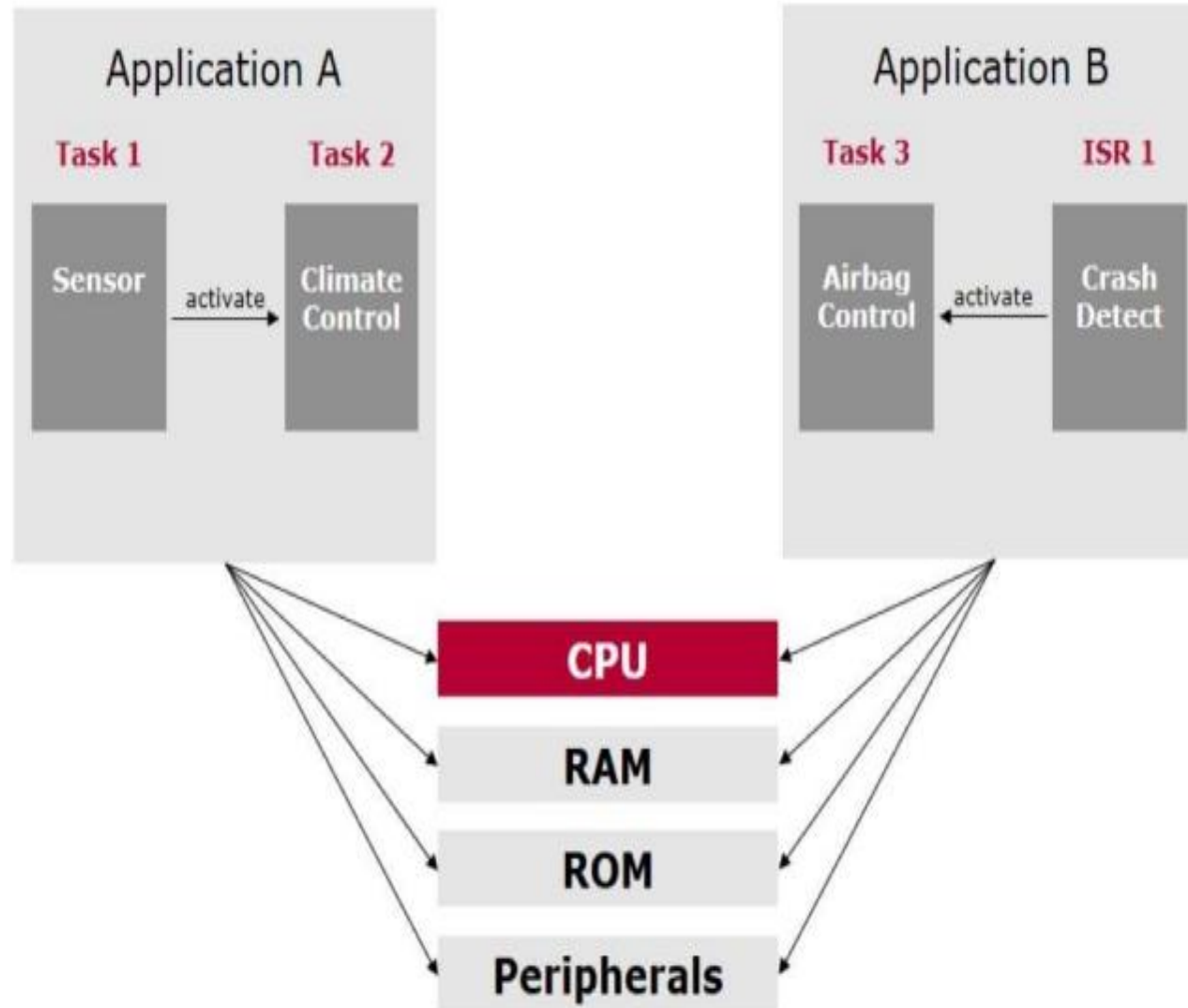
- AUTOSAR extends the OSEK/VDX Operating System specification
- The AUTOSAR extensions are grouped in Scalability classes (SC)



# AUTOSAR OS - Scalability Class 2



# AUTOSAR OS - Scalability Class 2



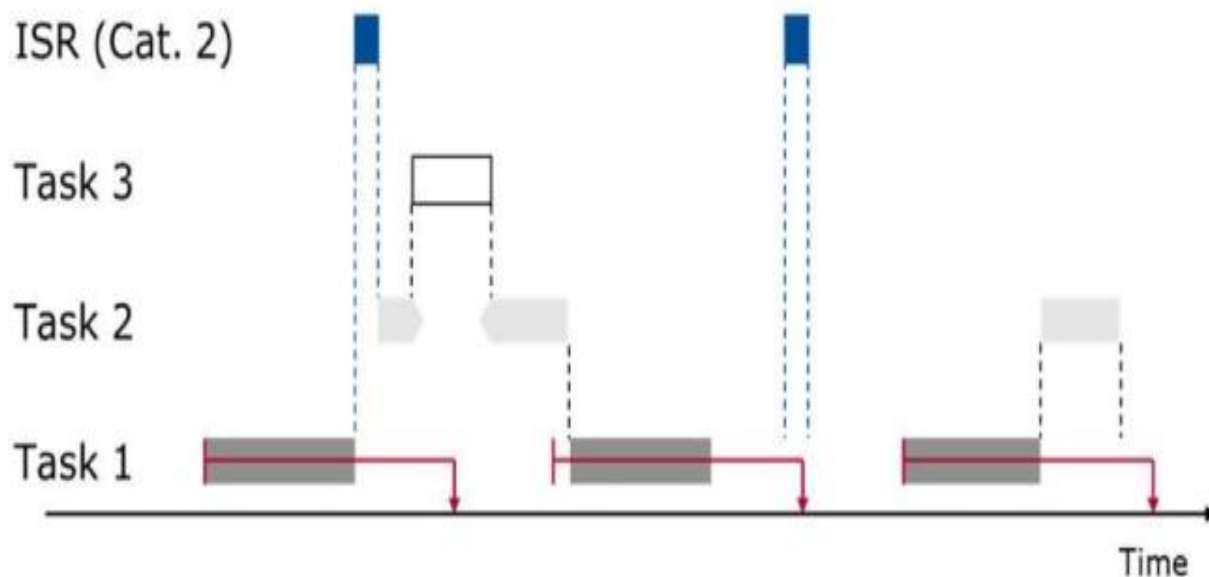
## AUTOSAR OS - Scalability Class 2

---

- The application shall be able to finalize data evaluation in due time, i.e.
  - Sufficient processing time is provided
  - Processing time is provided in due time
  - The application is not interrupted too long
  
- The supervision shall detect the cause rather than the violation

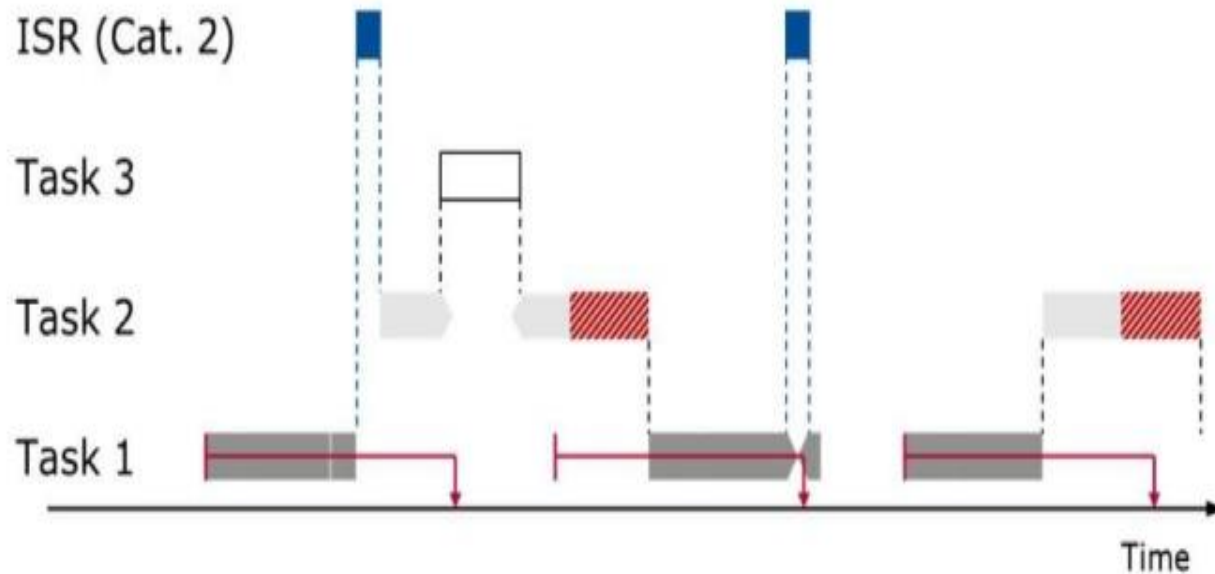


# Interference Scenarios



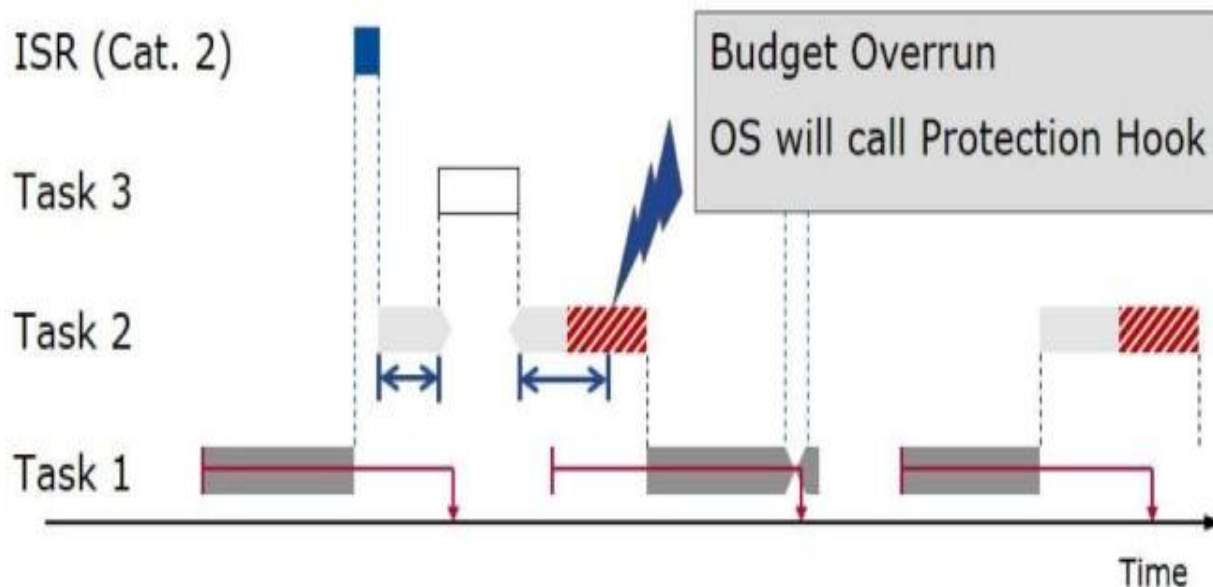
- Task 1 gets enough runtime to finish task execution within the deadline
- Even if the start is delayed by Task 2

# Interference Scenarios



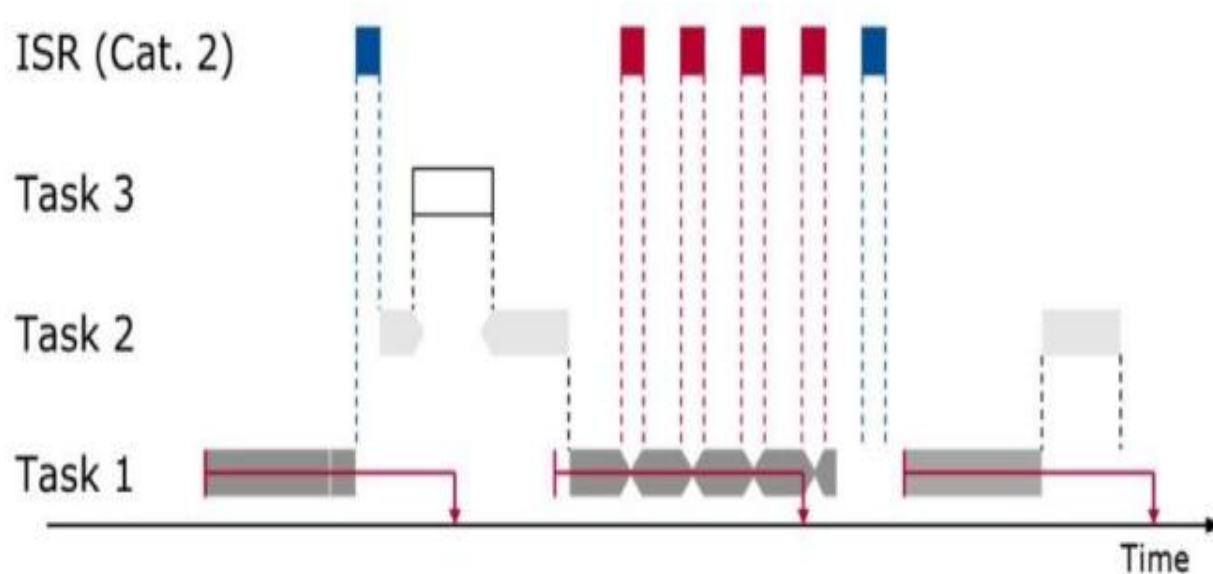
- Due to a defective prolongation of the Task 2's execution time, Task1 exceeds the given deadline

# Interference Scenarios



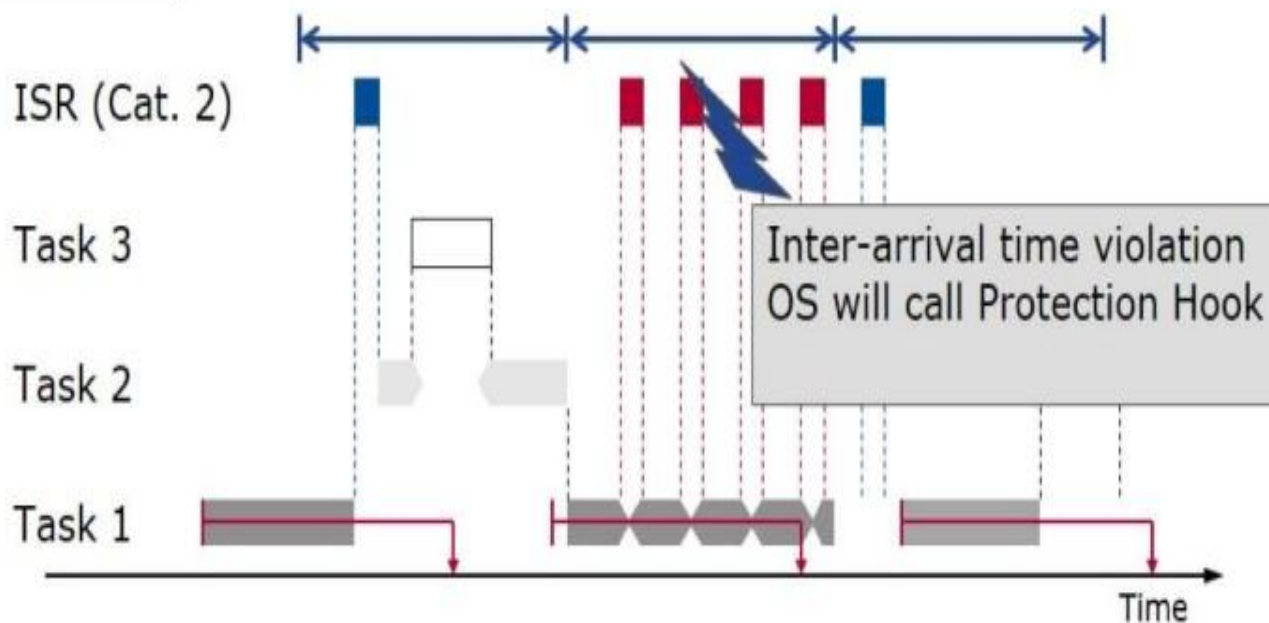
- Due to a defective prolongation of the Task 2's execution time, Task1 exceeds the given deadline
- If the Timing protection is configured in a proper way the OS can detect the Budget overrun of Task 2 and will call the Protection Hook

# Interference Scenarios



- The interrupt occurs too often. Due to these interrupts, Task 1 exceeds its deadline.

# Interference Scenarios



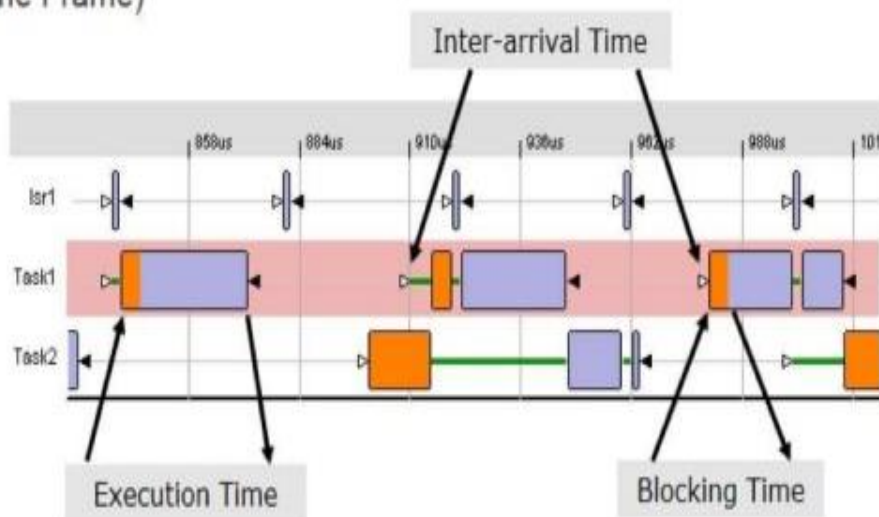
- The interrupt occurs too often. Due to these interrupts, Task 1 exceeds its deadline.
- If the Timing Protection is configured in a proper way, the OS will supervise the Inter-arrival time of the interrupts and detect a violation.



## OS SC2 - Supervised Characteristics

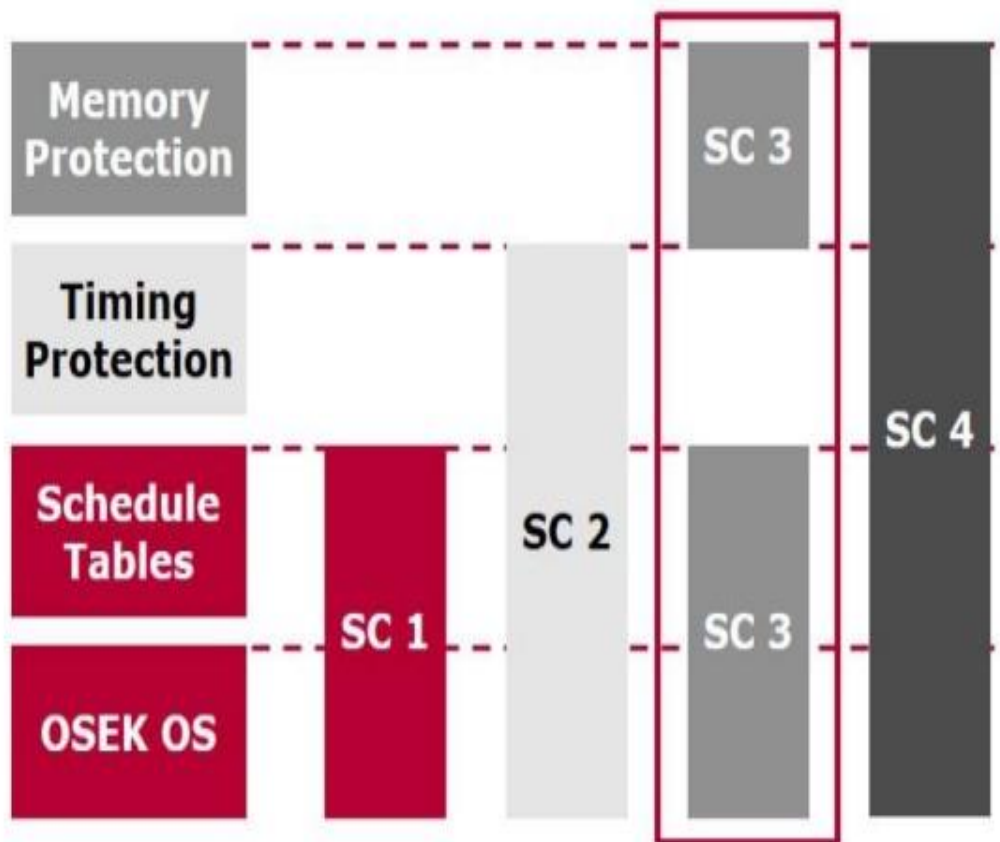
■ The following three characteristics of a Task or ISR (only category 2) are supervised:

- Execution time (Execution Budget)
- Blocking time (Lock Budget)
- Inter-arrival time (Time Frame)

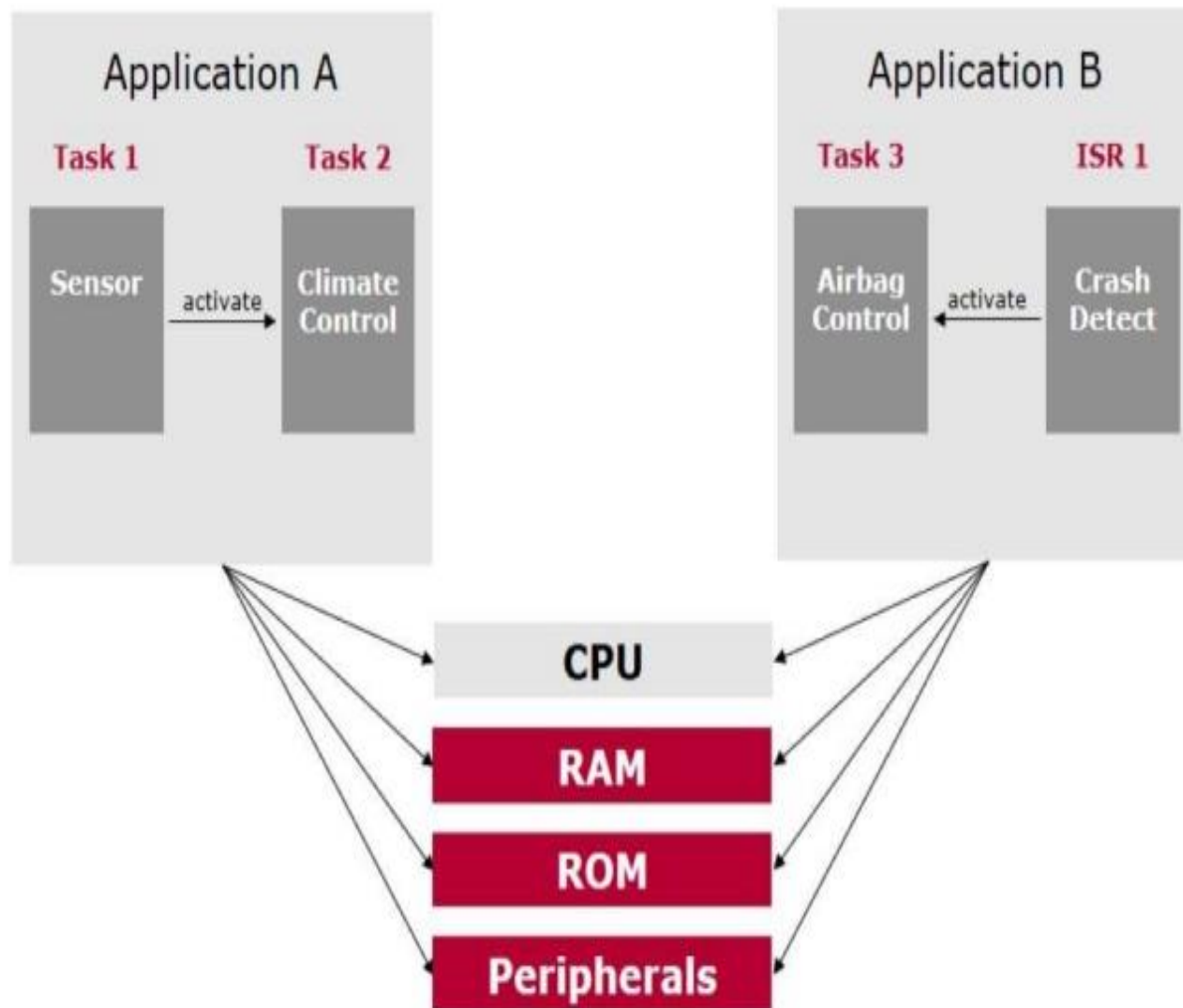


→ Timing Protection in the OS is a Budget Monitoring

# AUTOSAR OS - Scalability Class 2



# AUTOSAR OS Scalability Class 3

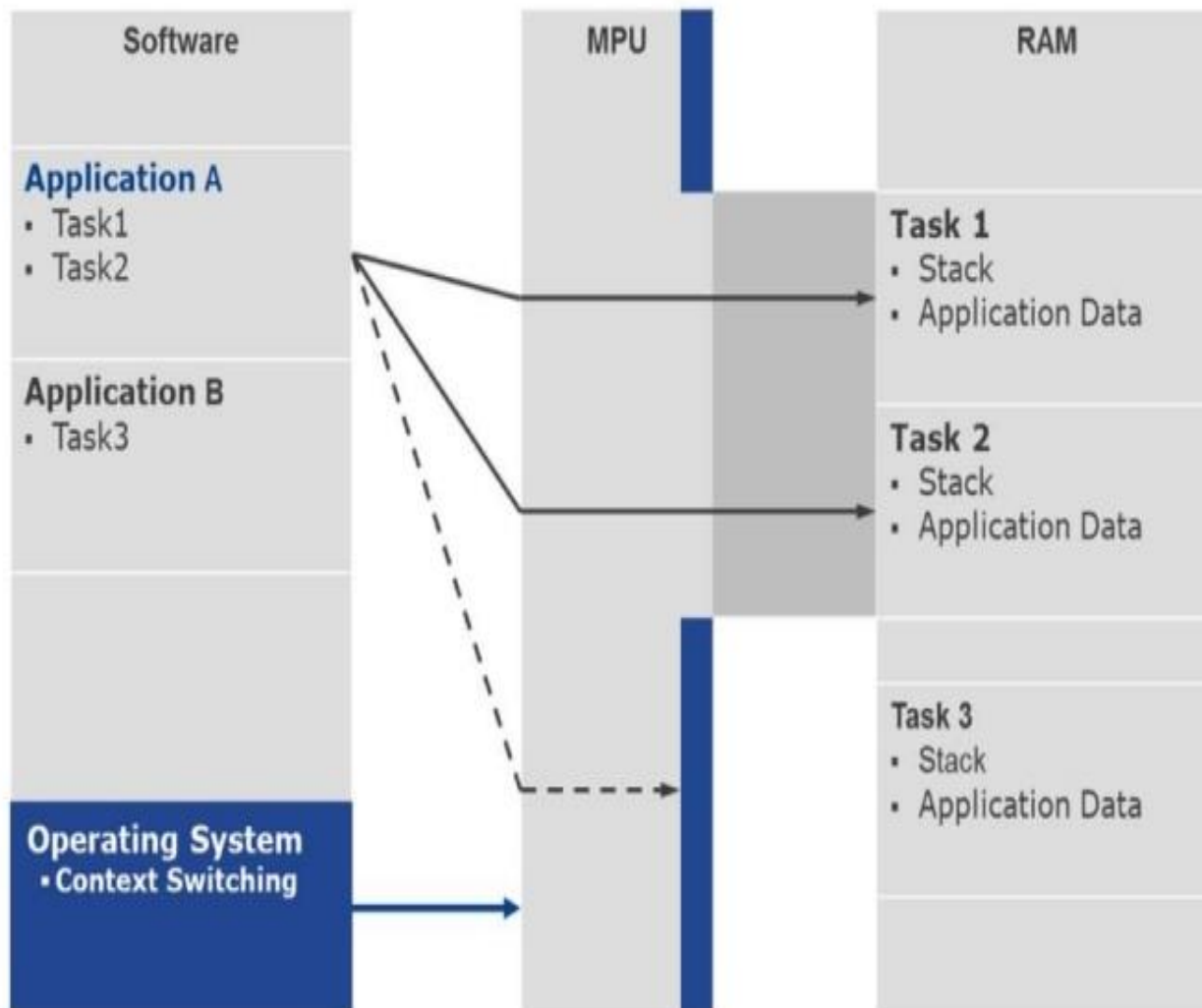


## Goal OS Scalability Class 3

---

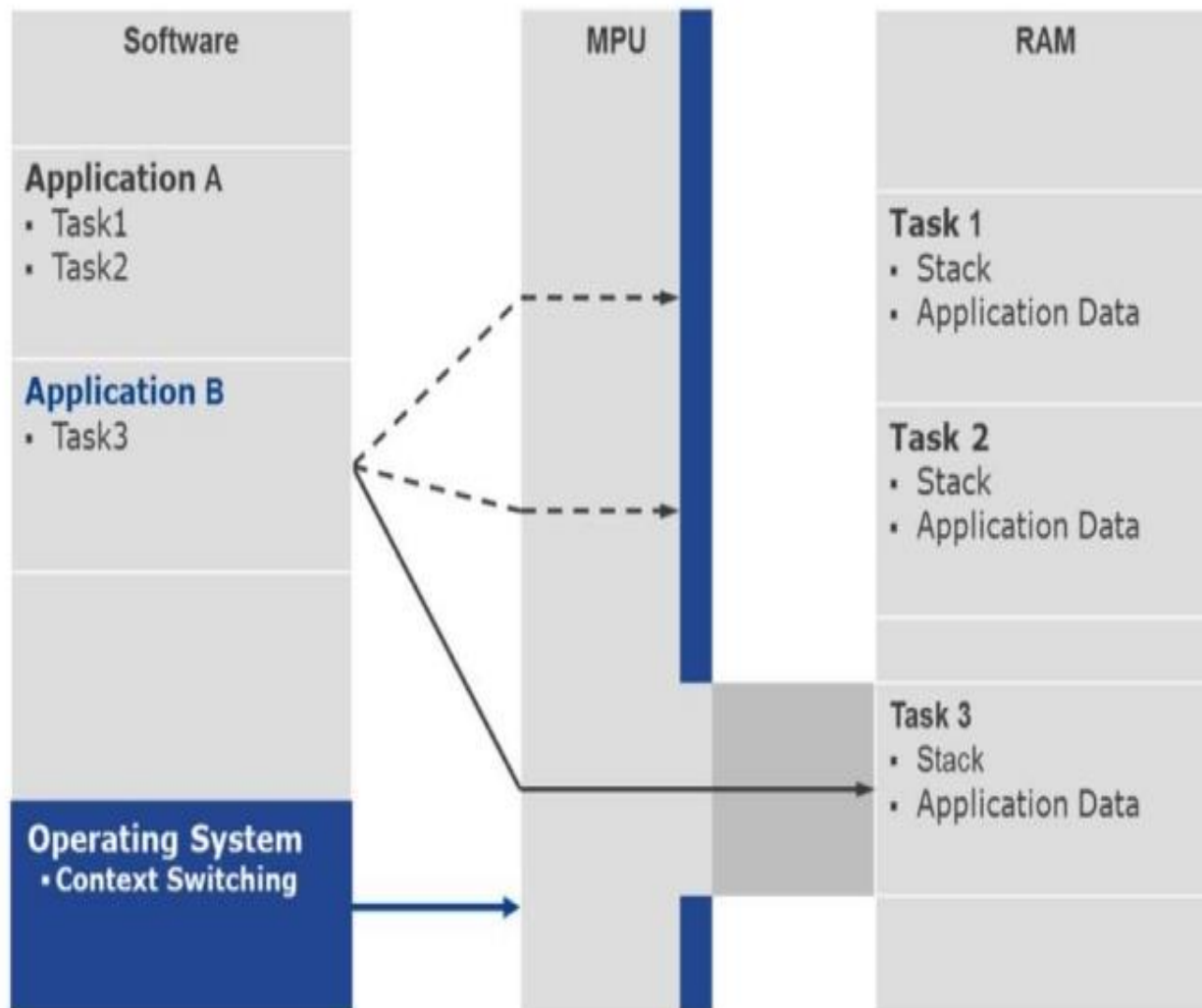
- The Application shall only be able to access Memory for which a Memory Access is configured.
- The OS shall prevent the access to Memory which is not configured for the Application.
- The OS shall provide Service Protection The OS shall prevent Application from accessing services for which no Access is configured.
- Hardware support is needed:
  - MPU (Memory Protection Unit)

# Memory Protection





# Memory Protection

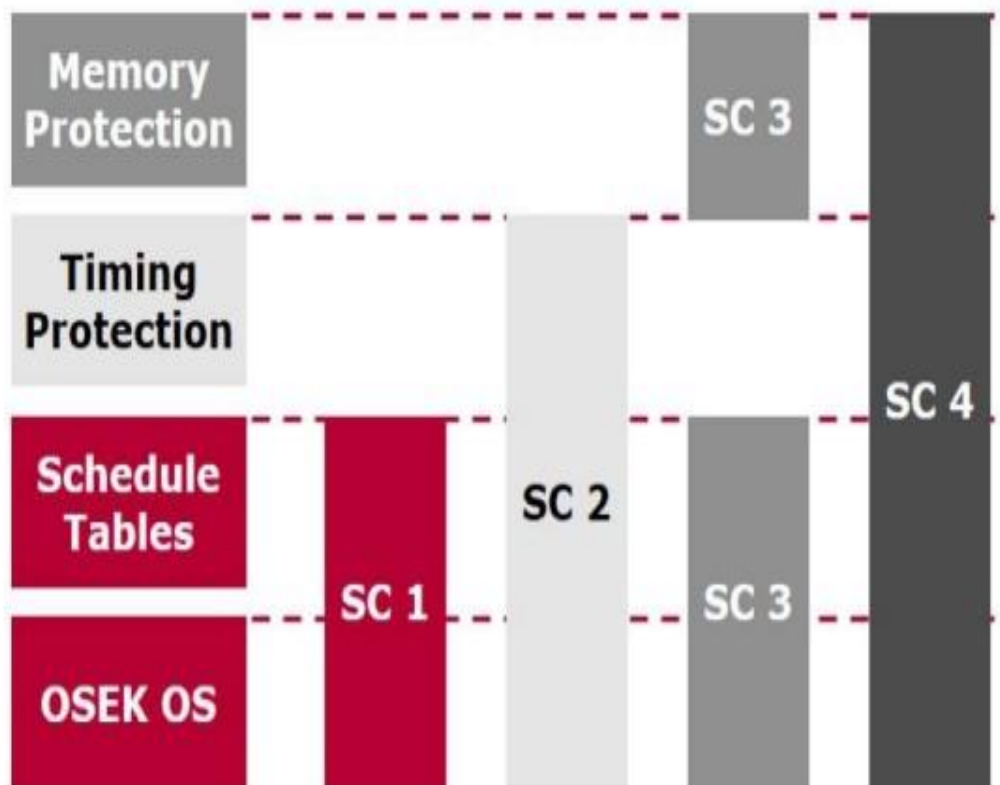


## Service Protection (SC2 and SC3)

---

- Service Protection is part of SC3 and SC4
- The following OS elements are protected
  - Task
  - Counter
  - Alarm
  - Resource & Spinlock
  - ScheduleTable

# AUTOSAR OS - Scalability Class 2



# Error Reaction

---

- For each violation of Memory Protection or Timing Protection the OS calls the Protection Hook:
  - Within the Protection Hook the Application can decide weather the OS shall:
    - forcibly terminate the Task/Category 2 ISR which causes the problem
    - forcibly terminate the OS-Application the Task/Category 2 ISR belong
      - optional with restart
      - shutdown the system
      - do nothing

# AUTOSAR Features which supports Freedom from Interference

	Memory	CPU Time	Communication
Watchdog	Alive Supervision Program Flow Monitoring		
OS	Scalability Class 3 and 4 Memory Protection	Scalability Class 2 and 4 Timing Protection	
End 2 End Protection	Detect Faults in Communication Link		

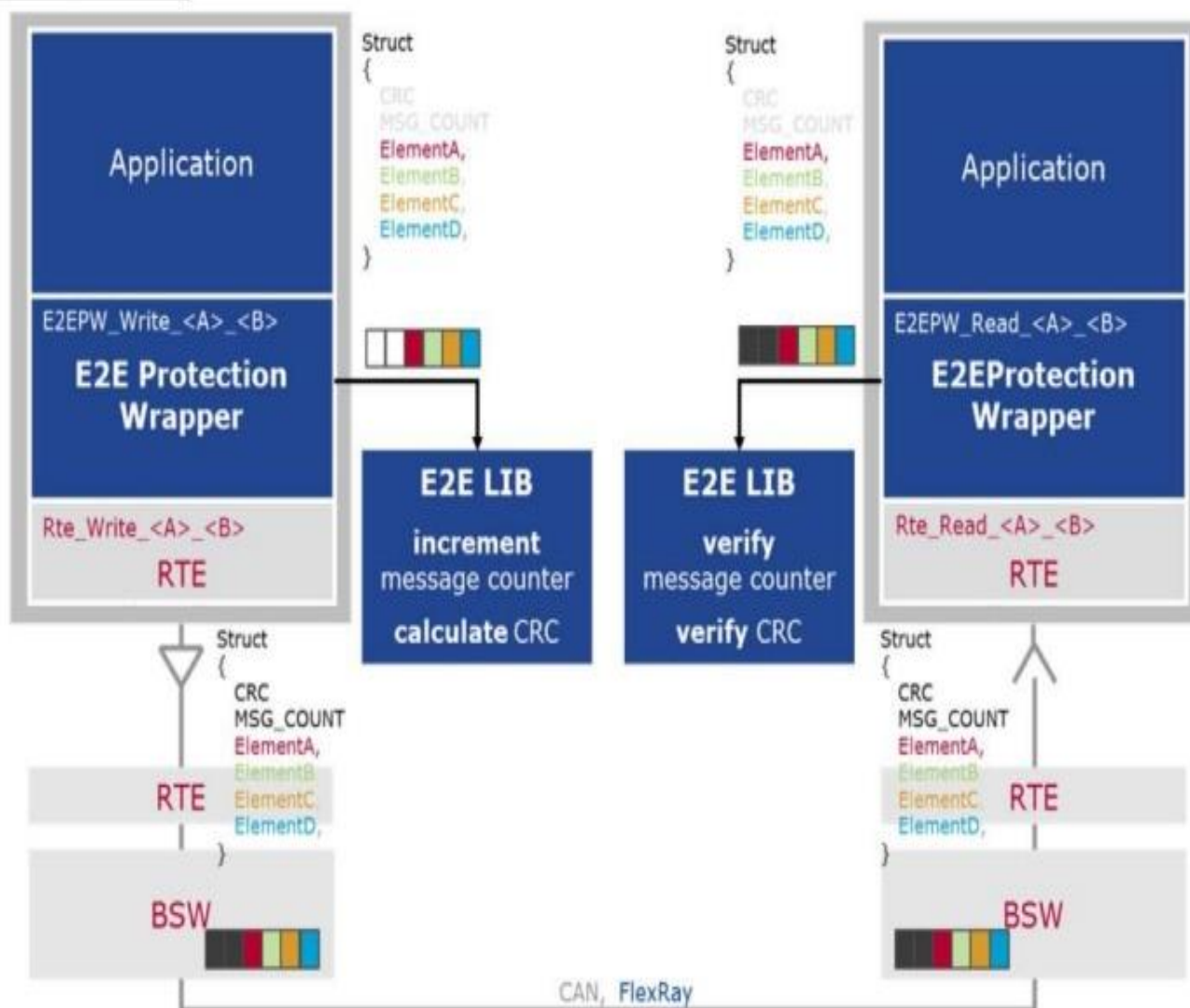
# End to End Protection (E2E)

---

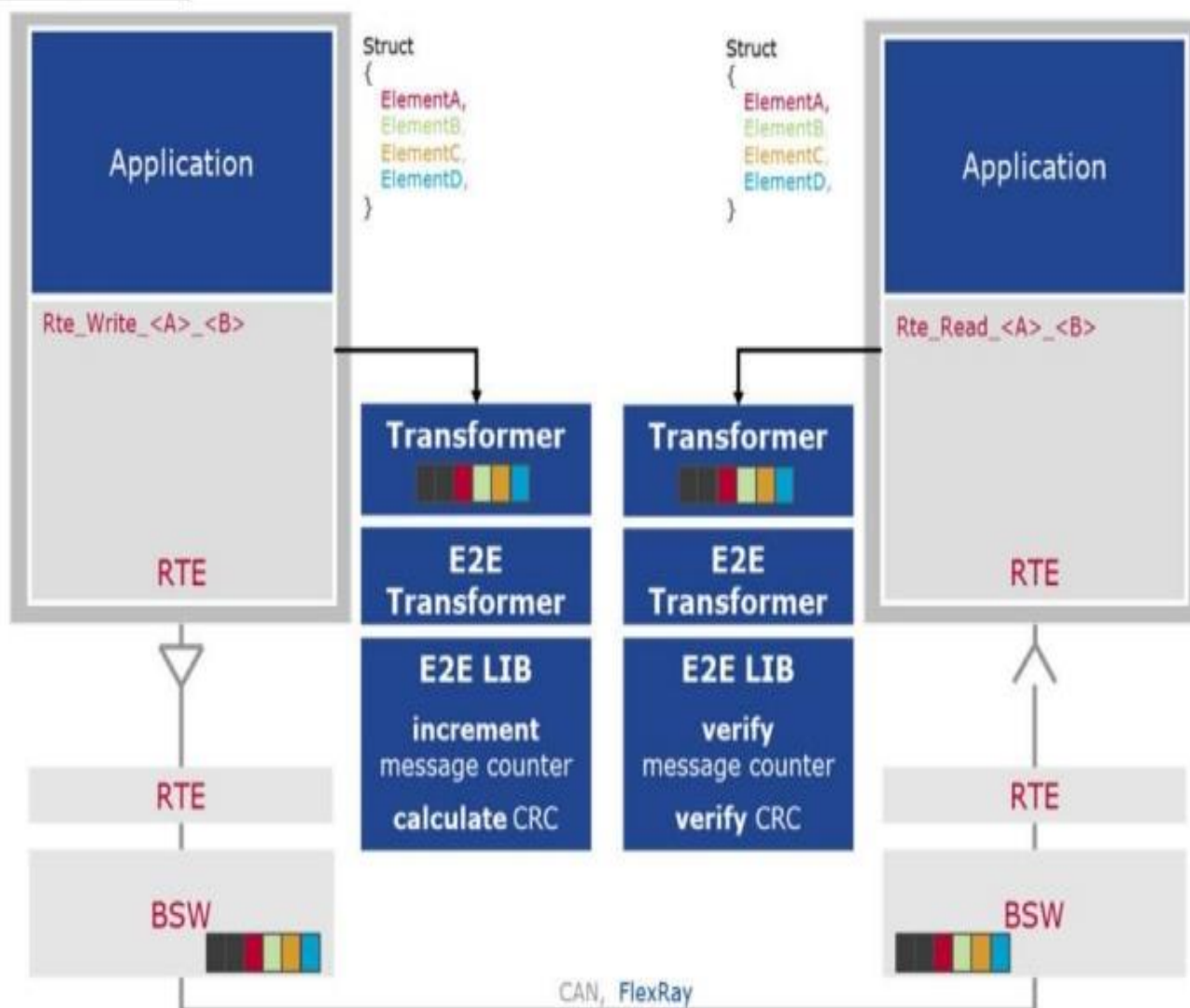
- End to End Protection detect faults in the communication link between two applications with help of:
  - Message Counter
    - Repetition
    - Loss
    - Insertion
    - Incorrect Sequence
    - Blocking
  - CRC
    - Corruption
  - Data ID
    - Masquerade and incorrect addressing
    - Insertion



# End to End Protection (E2E)



# End to End Protection (E2E) > AUTOSAR 4.2



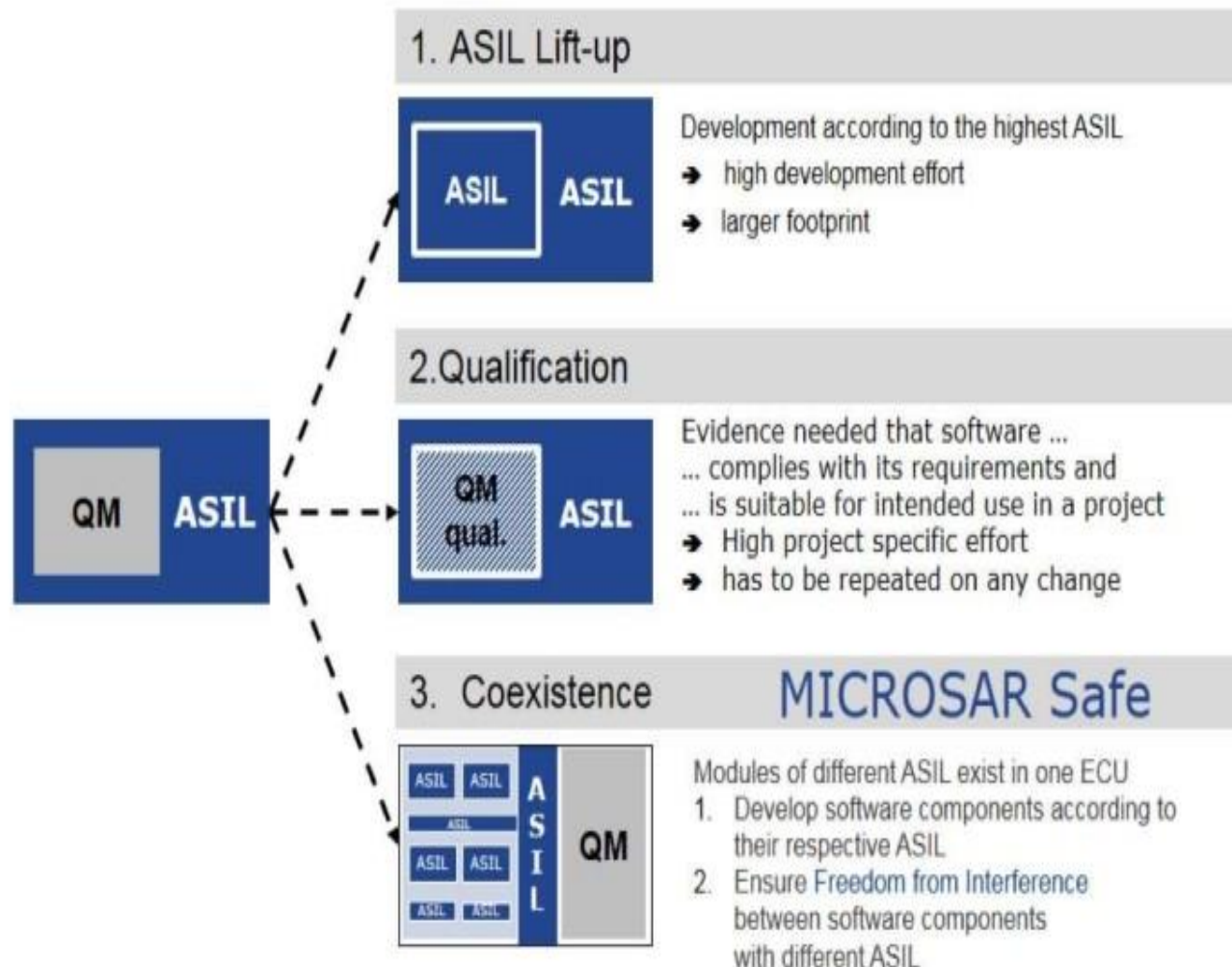
# AUTOSAR Features which supports Freedom from Interference

	Memory	CPU Time	Communication
Watchdog	Alive Supervision Program Flow Monitoring		
OS	Scalability Class 3 and 4 Memory Protection	Scalability Class 2 and 4 Timing Protection	
End 2 End Protection	Detect Faults in Communication Link		

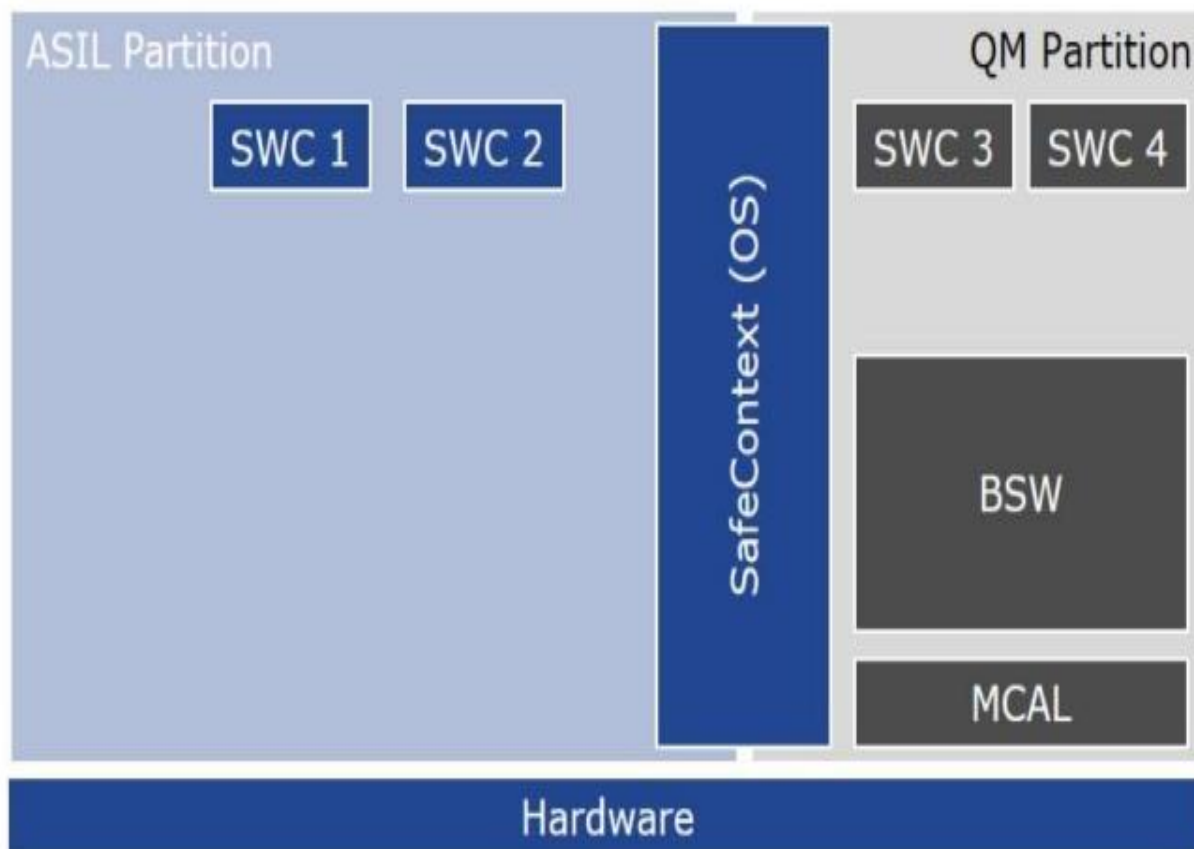
## MICROSAR Safe – Vector's AUTOSAR Functional Safety Solution



# MICROSAR Safe

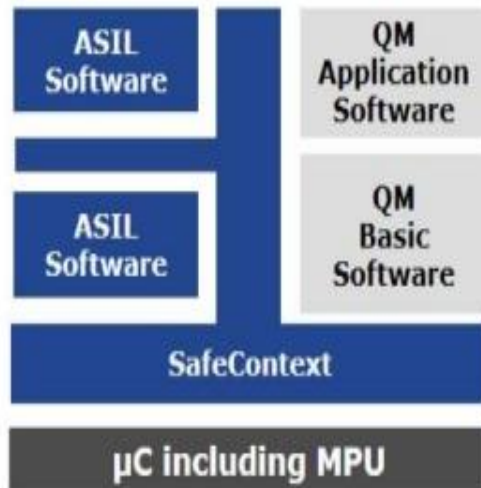


# MICROSAR Safe



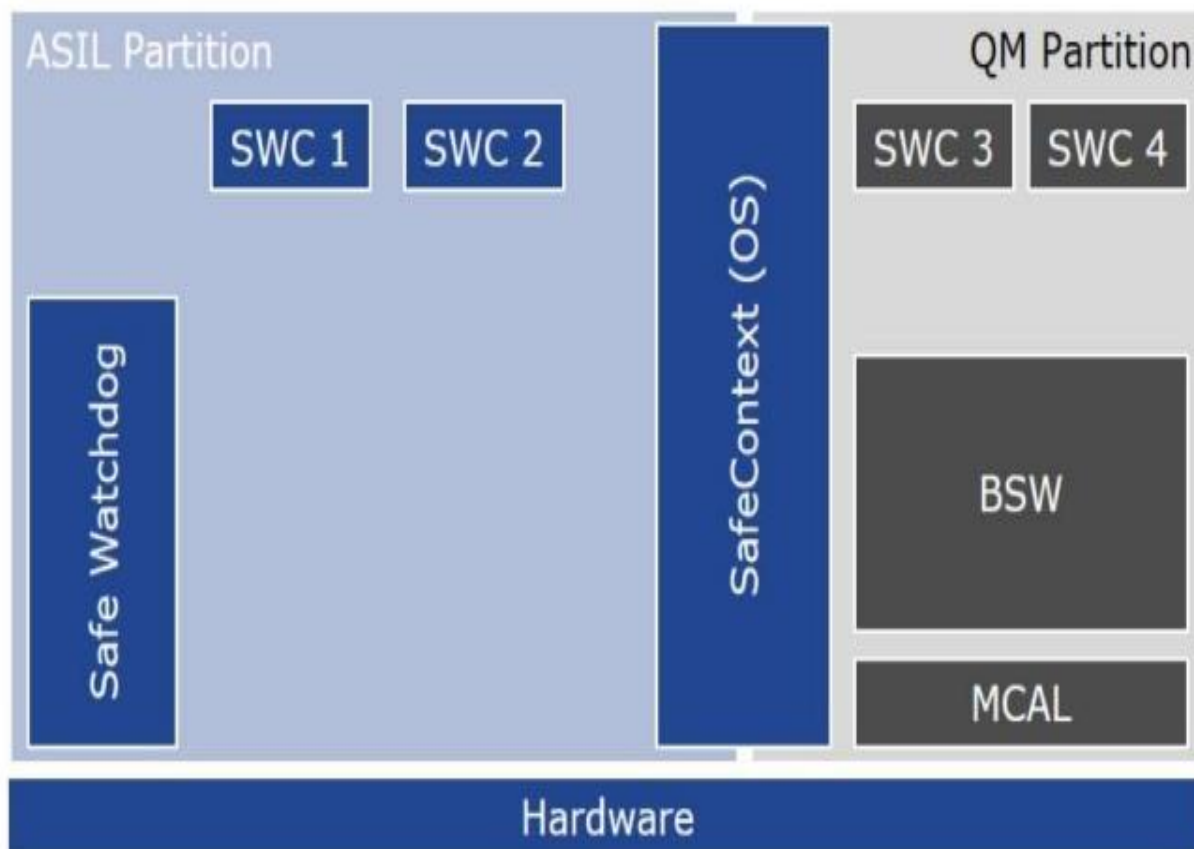


# SafeContext



- Supports memory separation by MPU
  - Provides safe context switch for each safety related task:
    - register settings
    - stack pointer and program counter
    - MPU settings
  - Available for single- and multi-core
- 
- Features **additional to AUTOSAR-OS SC3/SC4**:
    - Non-trusted function calls
    - Save access to peripheral registers even from user mode (to allow certain HW accesses from QM-code)
    - Usage of hardware specific features to protect registers
    - Optimized S/R communication across different contexts

# MICROSAR Safe

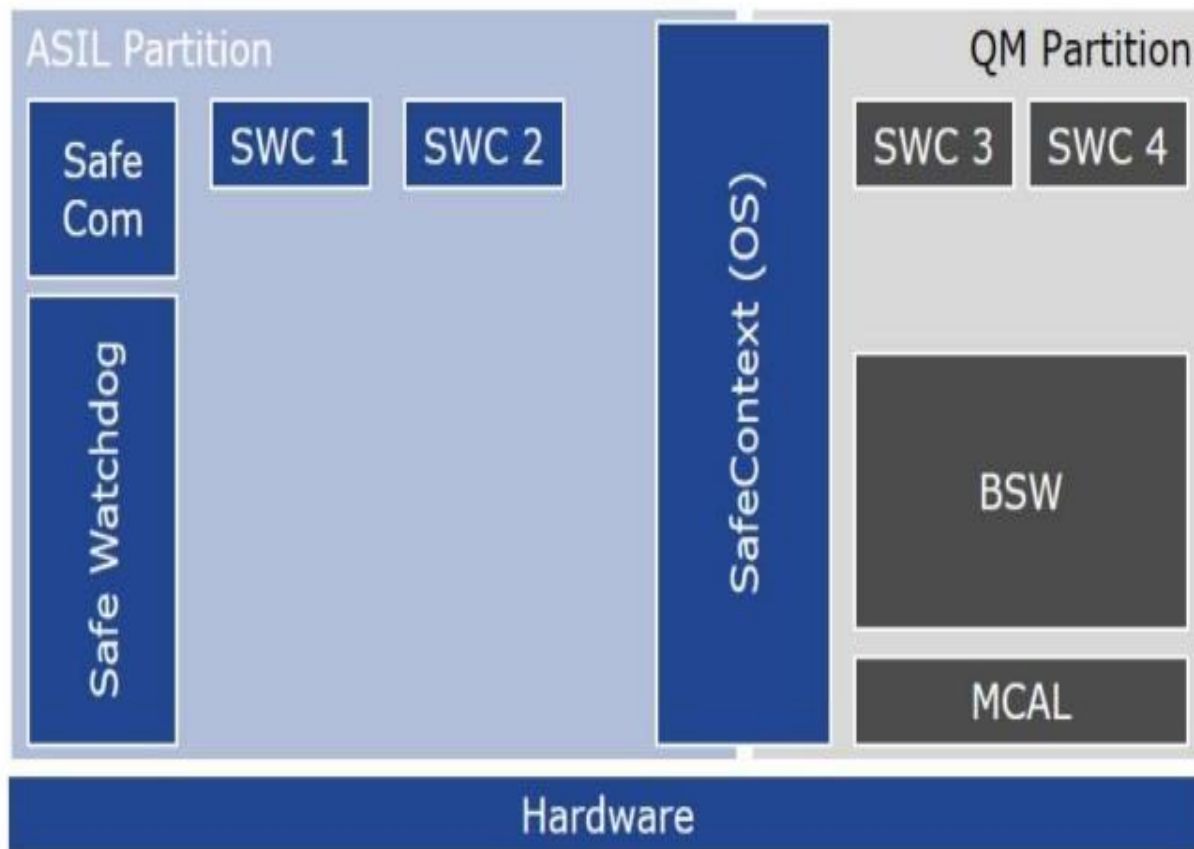


# SafeWatchdog

---

- Provides Freedom from Interference regarding Timing
- Monitoring options:
  - Alive Supervision
  - Program Flow Monitoring
  - Deadline Monitoring
- Supervised entities can be mapped to different memory partitions  
(e.g. for an ASIL B and an ASIL D partition)
- Calls to Checkpoint reached without context switch
- Possible responses to supervision violations
  - wait for watchdog expiration → ECU reset
  - immediate ECU reset  
(also allowing a secondary reset path via the MCU driver)
- Supports different trigger modes e.g. slow/fast
- Optimized solution for multi-core available

# MICROSAR Safe

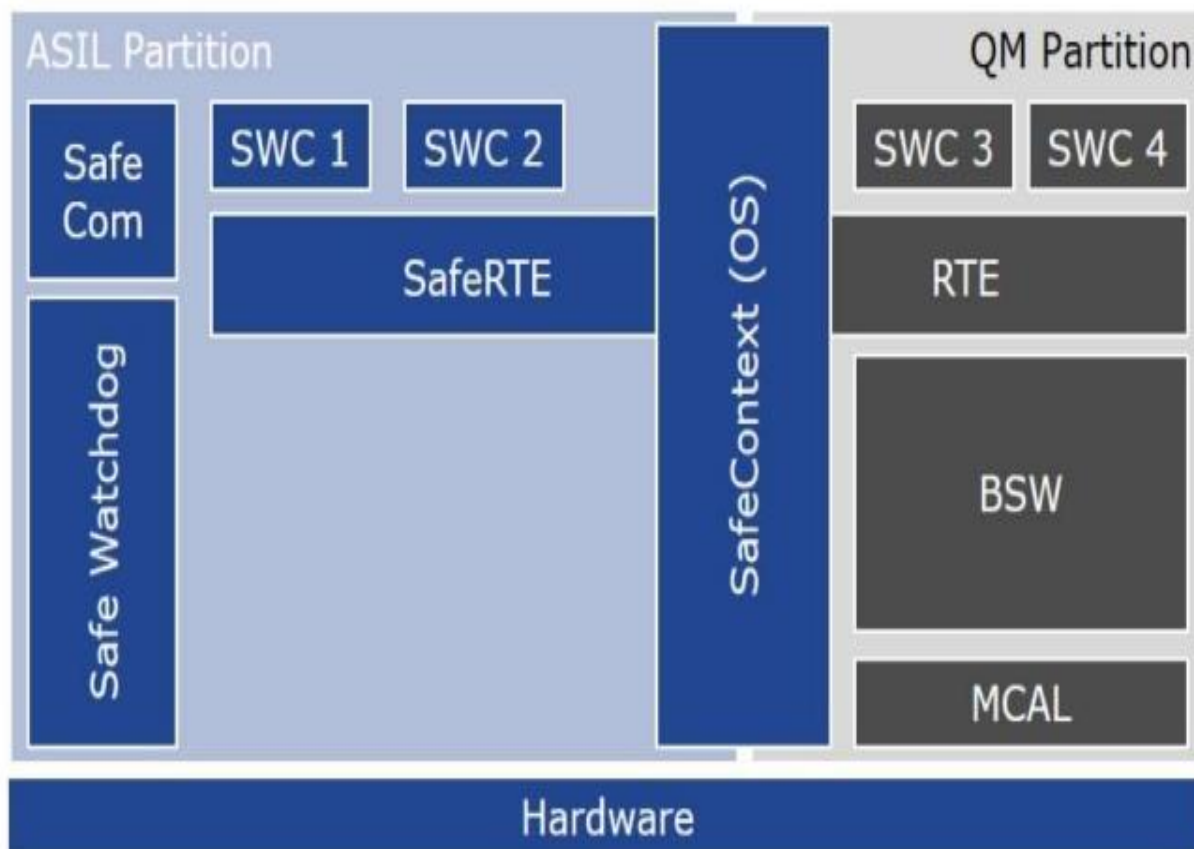


# SafeCom

---

- Provides Freedom from Interference regarding Communication
- Contains E2E Library and E2E Protection Wrapper
  - Supported Profiles
    - Profile 1
    - Profile 2
- Since AUTOSAR 4.2 also available as E2E Library and E2E Transformer in combination with SOMEIP and COM Transformer
  - Supported Profiles
    - Profile 4
    - Profile 5
    - Profile 6

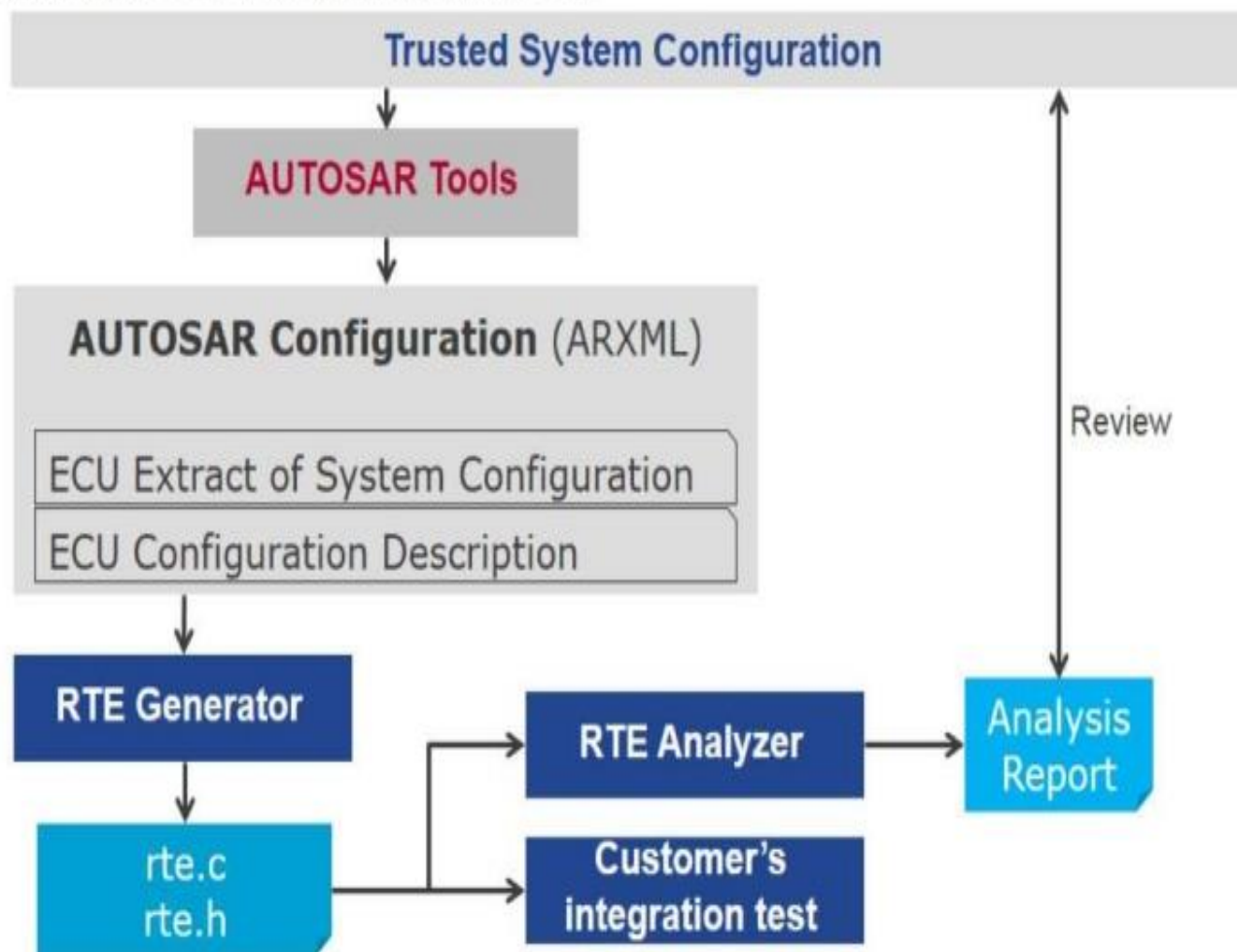
# MICROSAR Safe





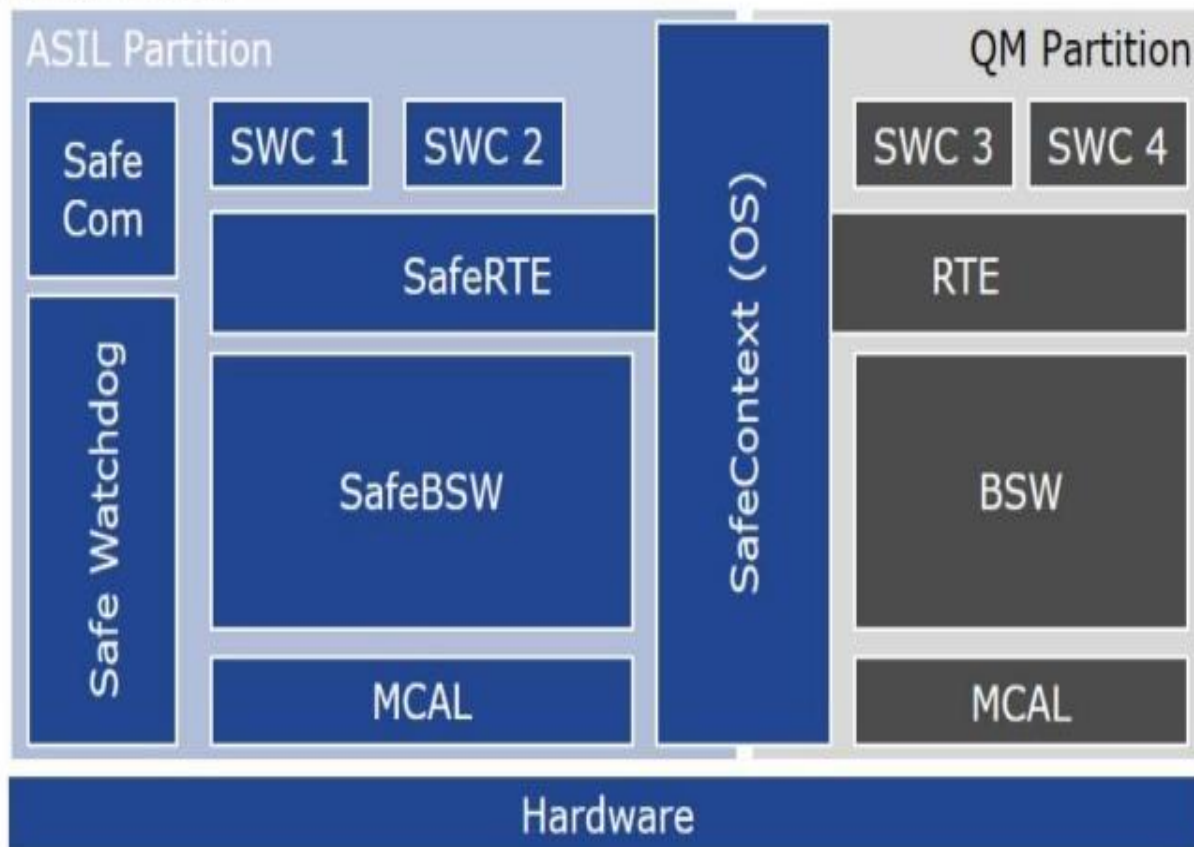
# SafeRTE

- Tool based verification process of the generated RTE code



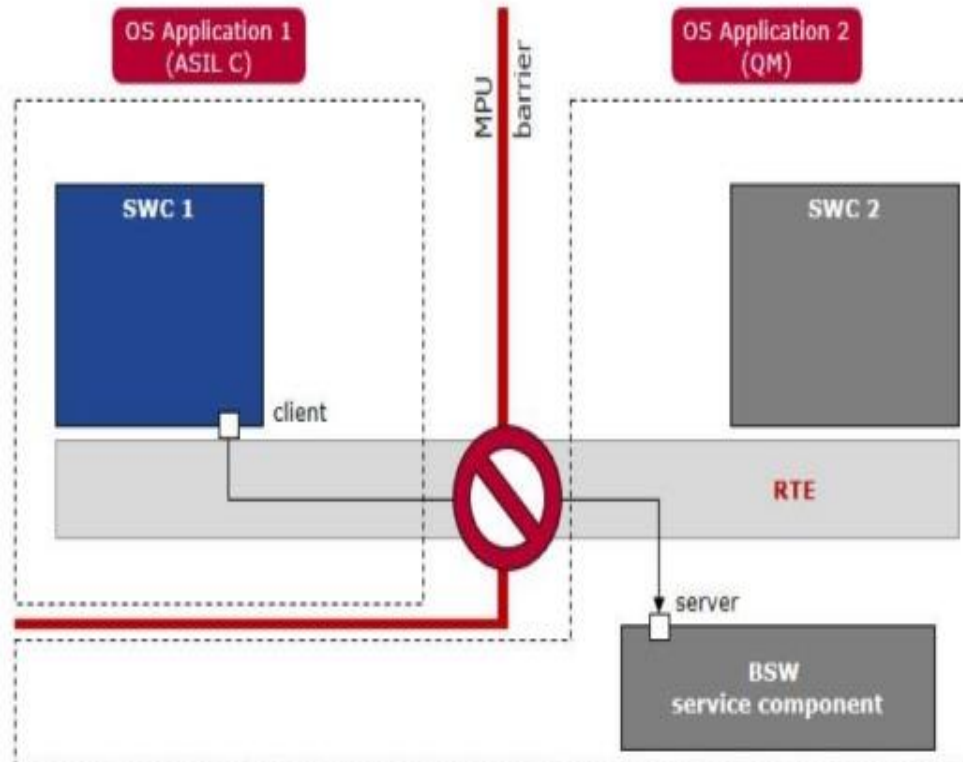
# MICROSAR Safe

## ■ MICROSAR Safe Outlook



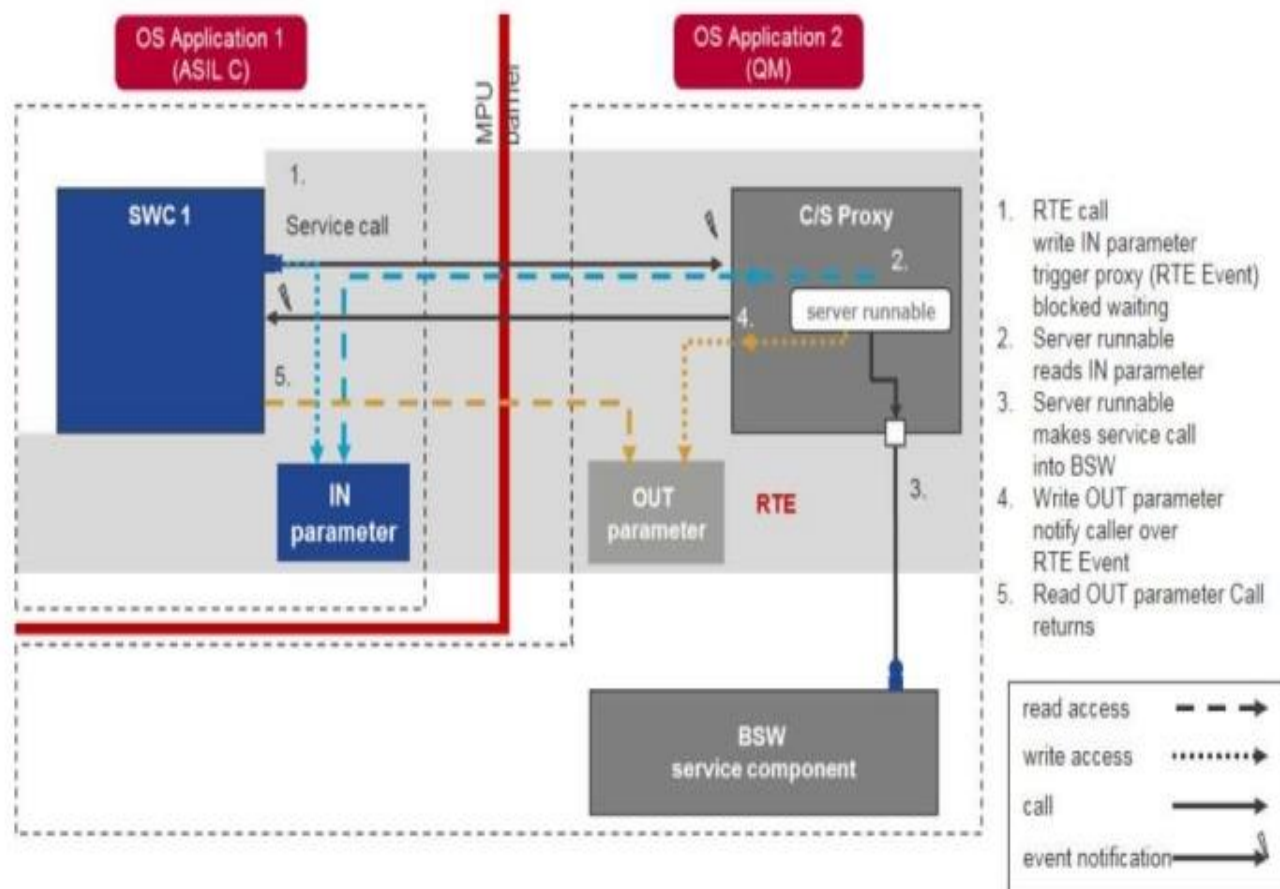
# Why SafeBSW

- No client/server communication between OS Application:
  - Invocation of lower level ASIL code in callers context is prohibited
  - Access rights for write parameters could not be given

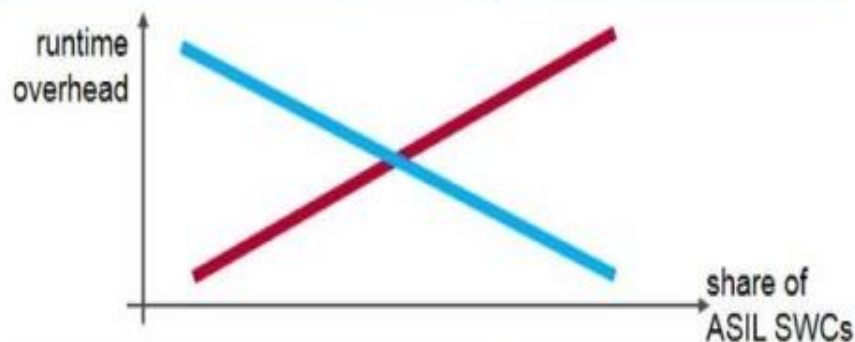
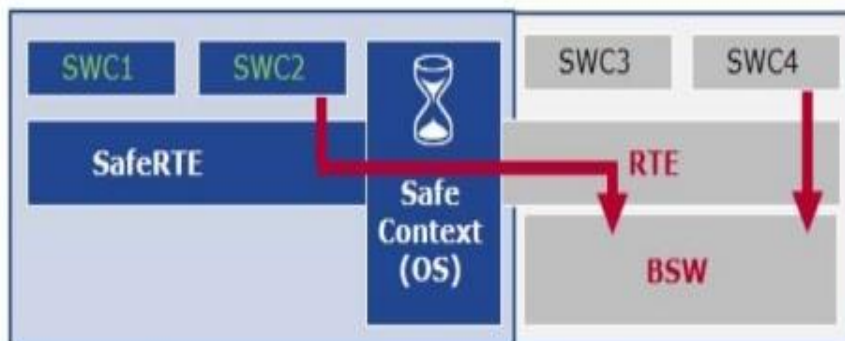


# Why SafeBSW

- Use case: ASIL SW performs a call into a BSW service



# Why SafeBSW



Drawback of partitioning:

- Context switches needs runtime

- Pay attention to the share of ASIL SWCs

➔ For high ASIL Share BSW should be part of ASIL partition

# SafeBSW

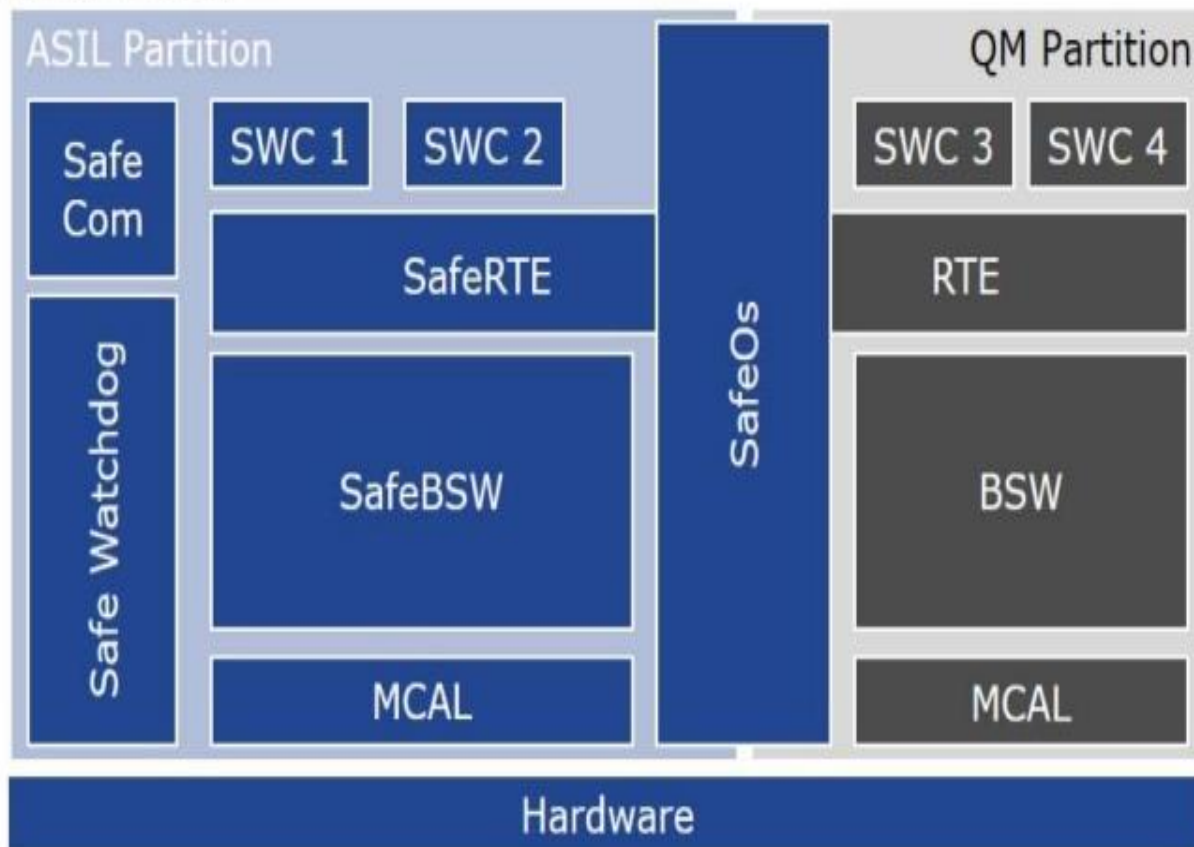
---

- Goal: BSW MICROSAR 4 acc. ASIL D
- Basis:
  - SafeContext, SafeWatchdog, SafeCom, SafeRTE
- Additional Safety Features derived from customer needs:
  - Safe ECU initialization
  - Safe ECU reset
  - Safe write/read of NVRAM data
  - Safe scheduling
  - ...
- Development Process
  - According ISO 26262 ASIL D
- Availability
  - 2016



# MICROSAR Safe

## ■ MICROSAR Safe Outlook

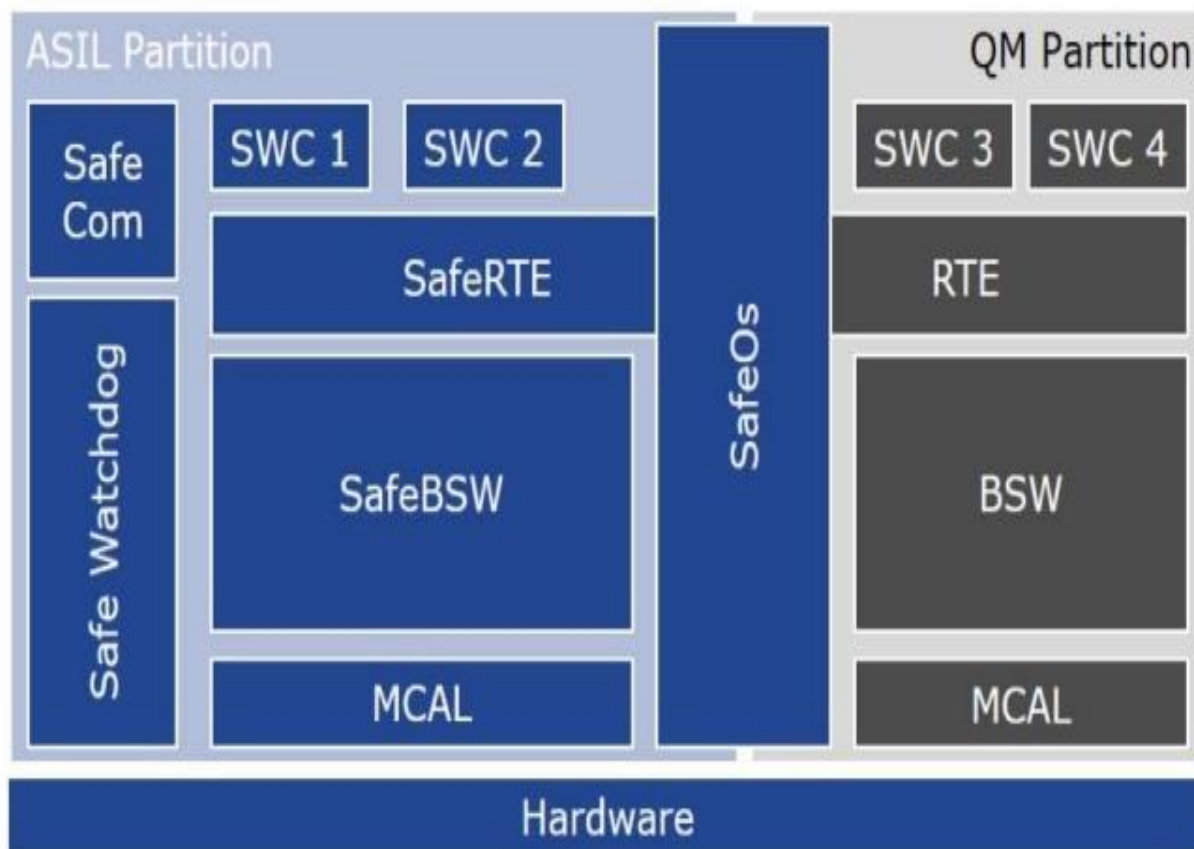


# SafeOs

---

- Completely developed according to ASIL D
- Provides Freedom from Interference regarding
  - Memory
  - Timing
- Safety Functionality
  - Initialization
  - Reset
  - Scheduling
  - Memory partitioning
  - Timing partitioning
  - Functional degradation
  - Inter OS Application Communication (IOC)
- Multicore OS supports isolation of cores
- Availability
  - 2016

# MICROSAR Safe



# Summary



# How to apply Functional Safety to AUTOSAR ECUs

---

- AUTOSAR provides features which helps you to build ECUs and to comply to ISO26262
  - Vector provides Building Blocks which makes it easier for you to build ISO 26262 compliant ECUs
    - Vector's solution uses functionalities already standardized by AUTOSAR.
      - This minimizes effort for introduction and for re-use in later projects.
    - Available
      - Vector's solution is available for many hardware platforms
- Vector reduces time-to-market and development costs

# Questions?





# Please Complete the Survey Form

Please take a moment to provide us with your feedback. Using your DevCon 2015 app, choose Surveys from the navigation bar.

The feedback we receive will help us with structuring lectures and labs at future DevCons. Thank you!

