The Automotive Industry
Standards and technologies

Real-Time Industrial Systems

Marcello Cinque

Automotive industry

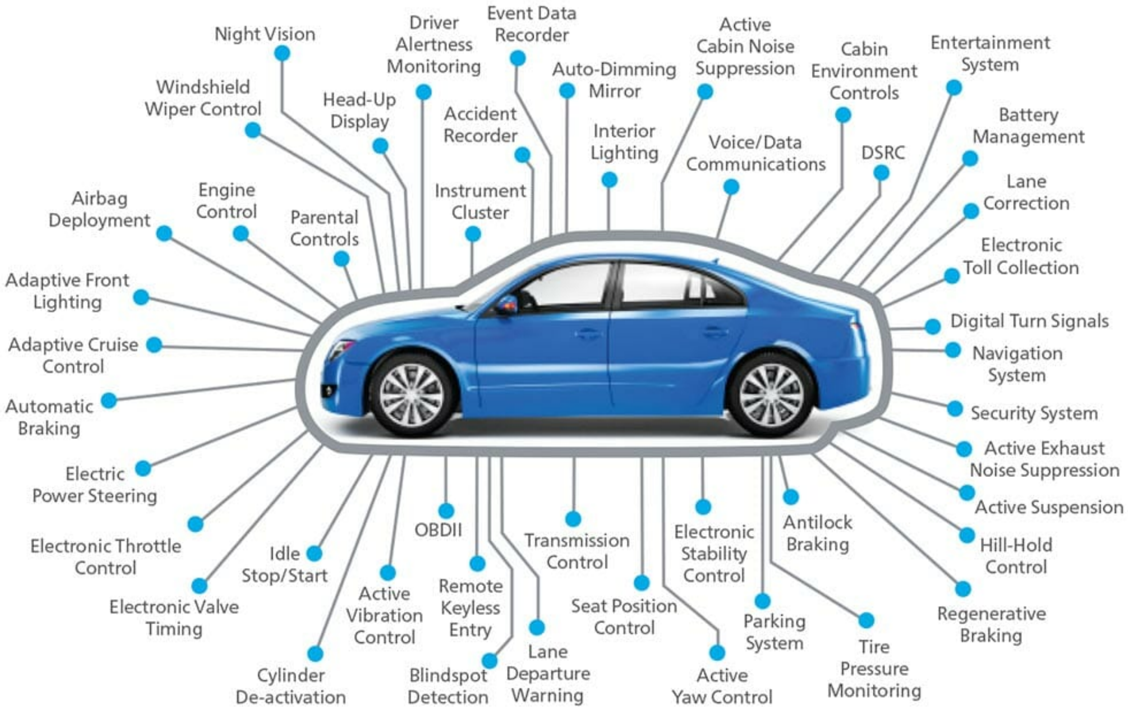
- All those companies and activities involved in the manufacture of motor vehicles, including most components, such as engines and bodies
- In the last 20-30 years the sector witnessed a shift from pure mechanical mass manufacturing to a blend of mechanics, electronics and computers
- Today we speak about “smart cars”, embedding 100+ ECUs, sensors, actuators, etc.

Examples of sensors

- *Rotation sensor*- designed to record the rate at which something spins.
- *Position sensor*- indicates the location of a moving part by varying the resistance in an electrical circuit.
- *Temperature sensor*- measures temperature of a gas or a liquid through variations in a circuit's resistance
- *Airflow sensor*- measures the air fed to the engine through the intake manifold
- *Knock sensor*: detects uneven fuel burning, that causes irregular vibration in the engine.
- *Oxygen sensor*- measures how much oxygen is in the exhaust, and thus how well the fuel is burning.

Example of computerized tasks

- ABS (Anti Breaking System)
- Traction control
- Electronic fuel injection
- Stability control
- In-car comfort (climate, seat, …)
- ADAS (Advanced Driver Assistance Systems)
    - Adaptive cruise control
    - Automatic emergency stop
    - Parking assistance
    - Automatic obstacle and pedestrian recognition
    - Driver’s attention recognition
    - Tires pressure monitoring
    - …
- … towards Autonomous Driving! (Google car & co.)



The importance of standards

- To regulate industry manufacturers, vendors, O&M
- Distinction between
  - Safety and Security standards
    - ISO 26262 for functional safety and ISO/SAE 21434 for cyber-security procedures, such as Thread Analysis and Risk Assessment – TARA
  - Platform standards
    - OSEK, AutoSAR

ISO 26262

ISO 26262
Road Vehicles - Fuctional Safety

- An engineering approach for safety in the automotive domain
- Adaption of IEC 61508 to comply with needs specific to road vehicles
  - IEC 61508: functional safety of electrical and/or electronic (E/E) systems
- The aim is to address possible hazards caused by malfunctions behavior of *E/E* safety-related systems in cars, and their interactions
  - So, it does not apply to hazards related to electrical shock, fire, smoke, heat, radiation, toxicity, flammability, reactivity, corrosion etc.

Automotive Safety Integrity Level (ASIL)

ASIL	Impact of Failure	Controllability	Exposure	In-Car Examples
A	Slight injury	Normally controllable	High probability	• Lag in display from rear-view camera
B	Severe injury	Normally controllable	High probability	• Failure of collision avoidance tone
C	Fatal/Survival uncertain	Difficult to control	Medium Probability	• Anti-Lock Braking system wheel lock-up • Out-of-control automatic transmission
D	Fatal / survival uncertain	Difficult to control	High Probability	• Steering-control lock-up • Airbag deployment while driving

How to determine the ASIL

Exposure E0 to E4 + Controllability C0 to C3 + Severity S0 to S3 = ASIL A to D

Exposure

Class				
E0	**E1**	**E2**	**E3**	**E4**
Incredible	Very low probability	Low probability	Medium probability	High probability

Controllability

Class			
C0	**C1**	**C2**	**C3**
Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Severity

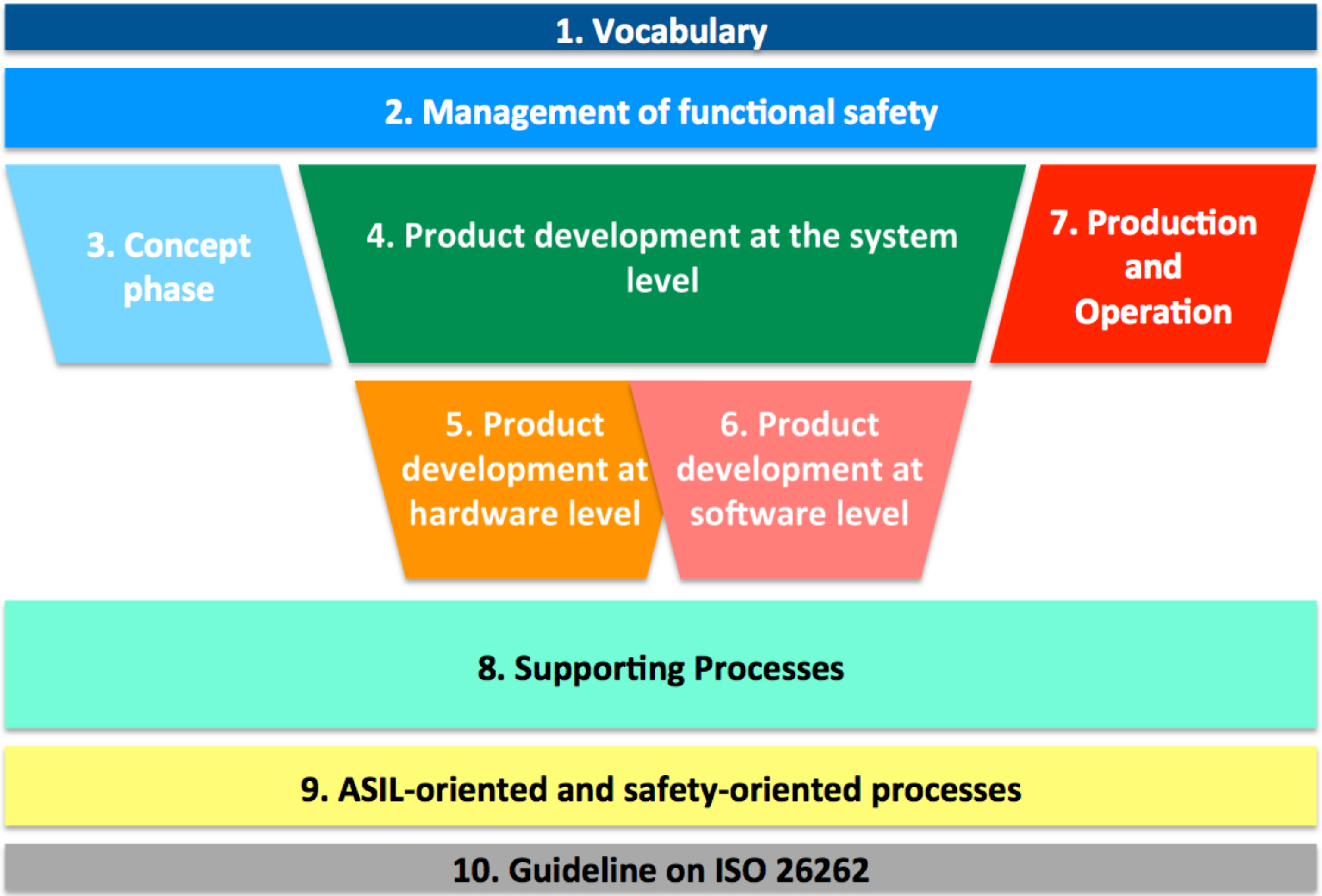
Class			
S0	**S1**	**S2**	**S3**
No injuries	Light and moderate injuries	Severe and life-threatining injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

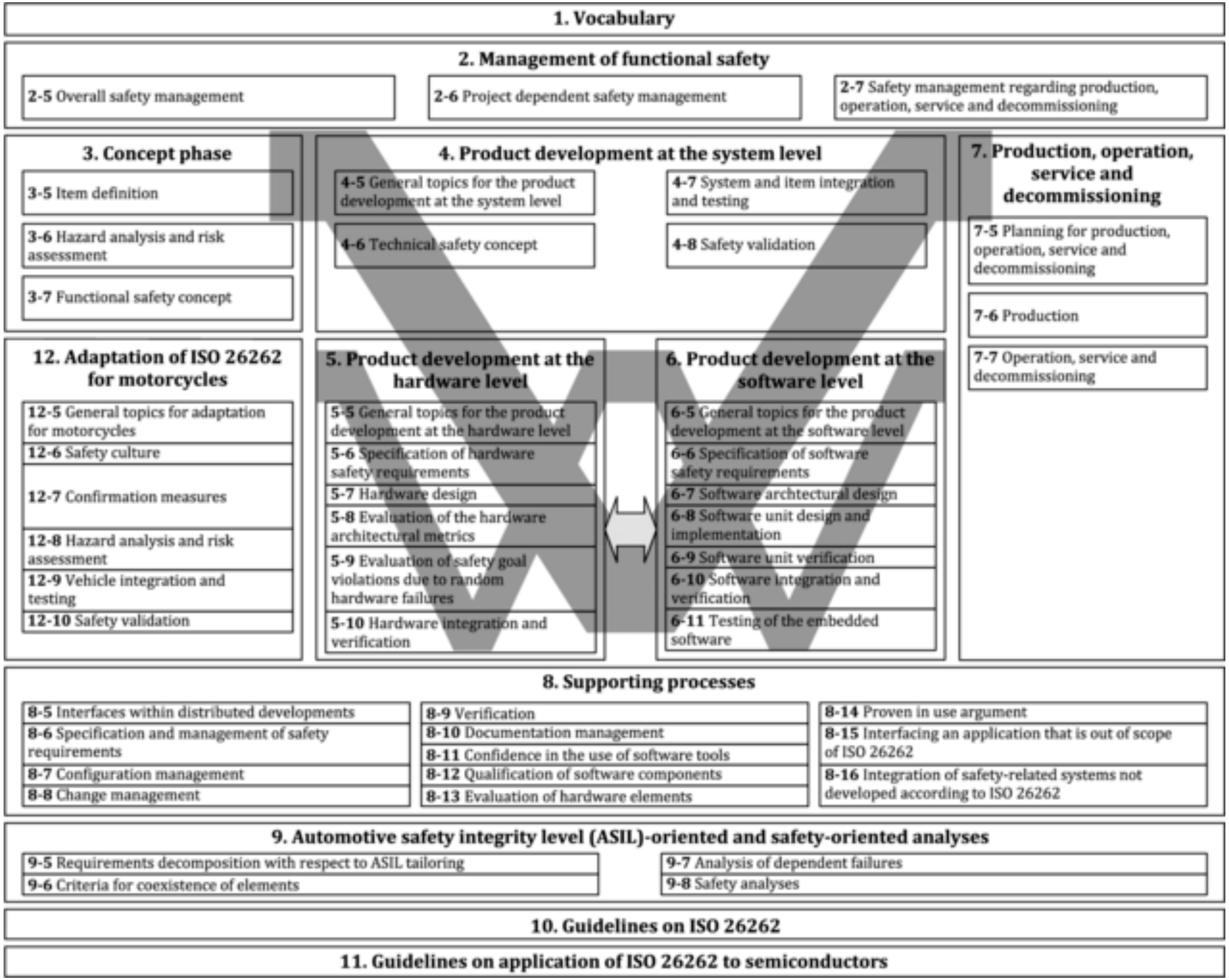
How to determine the ASIL

Severity class	Probability class	Controllability class C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	QM	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Note: QM denotes Quality Management with no need do comply to the standard

Flow of work products

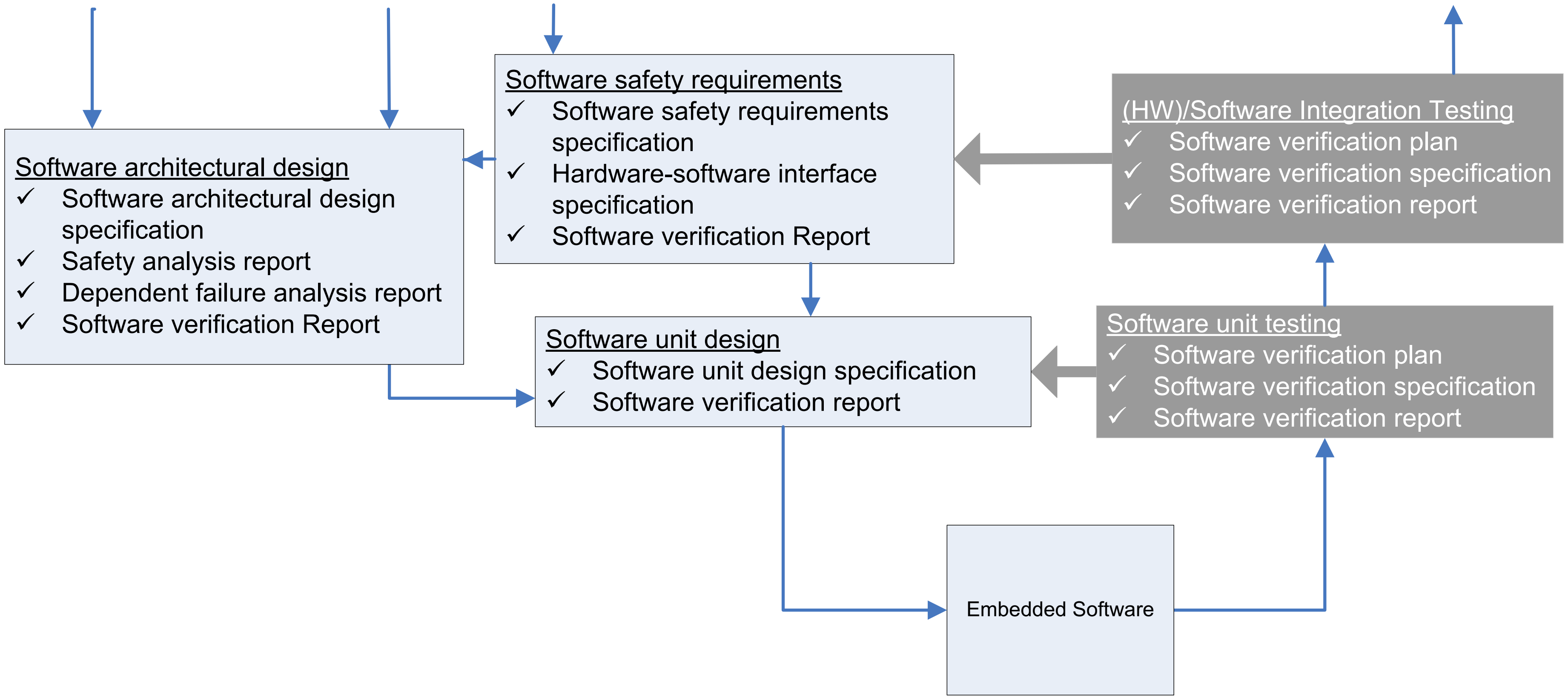



1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Project dependent safety management
2-7 Safety management regarding production, operation, service and decommissioning
3. Concept phase
3-5 Item definition
3-6 Hazard analysis and risk assessment
3-7 Functional safety 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
7. Production, operation, service and decommissioning
7-5 Planning for production, operation, service and decommissioning
7-6 Production
7-7 Operation, service and decommissioning
12. Adaptation of ISO 26262 for motorcycles
12-5 General topics for adaptation for motorcycles
12-6 Safety culture
12-7 Confirmation measures
12-8 Hazard analysis and risk assessment
12-9 Vehicle integration and testing
12-10 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 archtectural 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
10. Guidelines on ISO 26262
11. Guidelines on application of ISO 26262 to semiconductors

Work products: Concept phase

- ✓ Impact Analysis (Development of new Product or Modification of existing Product)
- ✓ Hazard analysis and risk assessment (HARA)
- ✓ Safety goals
- ✓ Functional safety concept (Requirements)
- ✓ Verification (Review) report

Work products: software level



Implementation recommendations

Methods	ASIL A	ASIL B	ASIL C	ASIL D
One entry and one exit point in subprograms and functions	++	++	++	++
No dynamic objects or variables, or else online test during their creation	+	++	++	++
Initialization of variables	++	++	++	++
No multiple use of variable names	+	++	++	++
Avoid global variables or else justify their usage	+	+	++	++
Limited use of pointers	o	+	+	++
No implicit type conversions	+	++	++	++
No hidden data flow or control flow	+	++	++	++
No unconditional jumps	++	++	++	++
No recursions	+	+	++	++

Coding rules: MISRA-C

- A set of C and C++ coding standards developed by the Motor Industry Software Reliability Association (MISRA)
- MISRA-C developed in multiple editions (1998, 2004, 2012, 2016)
- Adopted by ISO 26262 and AutoSAR
- Rules must be adopted in the code and can be verified by automated tools
  - Coverity, by Synopsis
  - Eclair, by Bugseng
  - Parasoft C/C++ Test, by Parasoft
  - SonarQube, by SonarSource
  - … and many others

MISRA-C rules examples

- Rule 59 MISRA-C:1998
  - The statement forming the body of an "if", "else if", "else", "while", "do ... while", or "for" statement shall always be enclosed in braces
- Rule 14.9 MISRA-C:2004
  - An if (expression) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement.
- Rule 18.1 MISRA-C:2012
  - A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand

Safety Element out-of-Context (SEooC)

- A safety-related element which is not developed in the context of a particular automotive system or vehicle.
- This means it has been developed for different (or wider) objective, yet it is useful to be adopted in an automotive system
- Examples of hardware SEooC:
  - Off-the-shelf sensors, microcontrollers, …
- Examples of software SEooC:
  - A library, the RTOS, the hypervisor, …

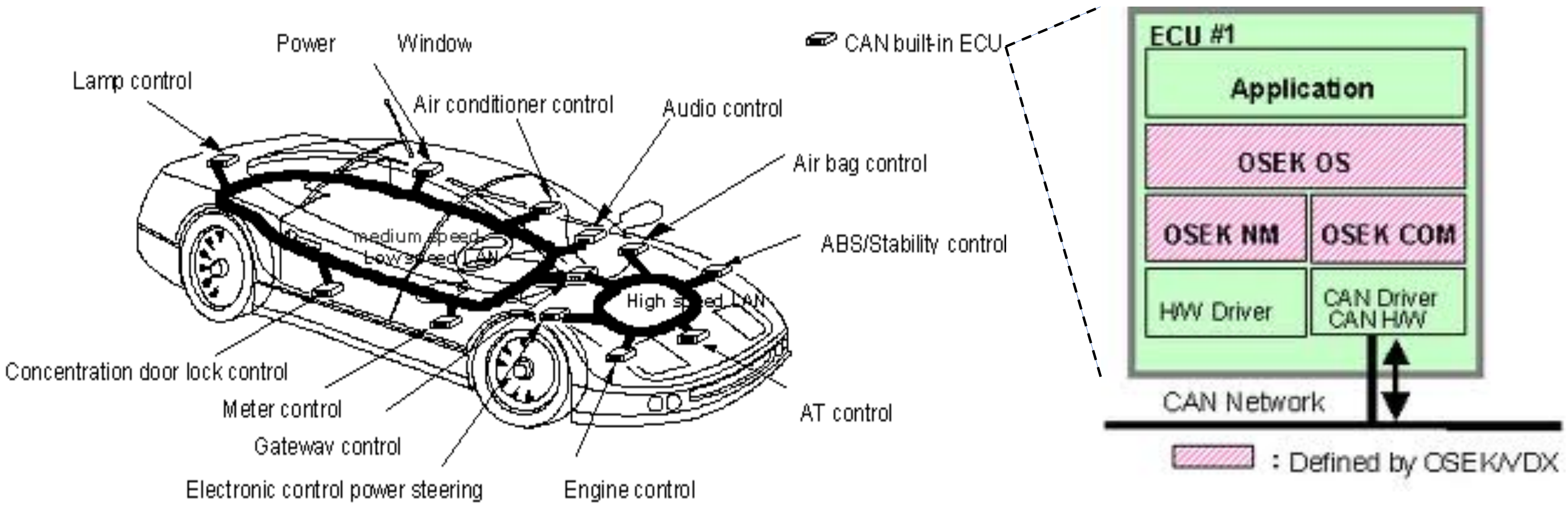
SEooC integration

- Because the element is developed out of context (i.e., not derived from the safety plan for the target project), additional measures must be applied
  - Such as, creation of a set of assumptions that the SEooC was designed to work within
  - These assumptions must be validated on the target platform during integration
- For instance, for a software to be integrated as SEooC, assumptions are needed on:
  - The software architecture of which the software will be a part of
  - The fact that Interference caused by the SEooC need to be handled by the system
  - Assumption of ASIL value of the software based on the data it handles
  - List of error conditions
  - Handling of instances of Data corruption
- During integration, it must be verified that all error conditions are properly handled (the SEooC can fail… in a controllable way)

OSEK

- Offene Systeme und derend Schnittstellen fur die Elektronik in Kraftfahrzeug (Open System and the corresponding interfaces for automotive electronic), OSEK/VDX is an initivative of german automotive companies (BMW, Bosch, etc.)
- VDX stands for Vehicle Distributed eXecutive and it describes the features of a distributed environment for the development of automotive applications
- The standard can be seen a set of RTOS APIs integrated in a network management system

OSEK Components



- OSEK OS: Operating System
- OSEK Time: time triggered operating system
- OSEK COM: communication services
- OSEK FTCOM: fault tolerant communication
- OSEK NM: Netowork Management
- OSEK OIL: Kernel configuration
- OSEK ORTI: Kernel awareness for debuggers

OSEK features

- **Scalability**: the OS can be used on several boards (from 8 bit microcontrollers to powerful processors)
- **Portability**: ANCI-C programming interface for tasks and ISRs (however, an interface with I/O is not specified since the OS can be ported on different hardware platforms)
- **Configurability**: a tool suite helps the designer to choose the proper services and the memory footprint. The OIL (OSEK Implementation Language) assists to define the system configuration (e.g., messages size)
- **Static allocation of components**: the code of kernel and tasks is statically allocated. The number of services is known at compile time.

OSEK tasks and scheduling

- Basic tasks:
  - A C function that never blocks (can only finish or be preempted by a higher priority task) and that can lock resources
  - It can share the stack with other tasks
- Extended tasks:
  - Can use events for synchronization and re-activation
  - Can have their own stack
- Scheduling:
  - Preemptive fixed-priority scheduling with immediate priority ceiling

OSEK compliant OS examples

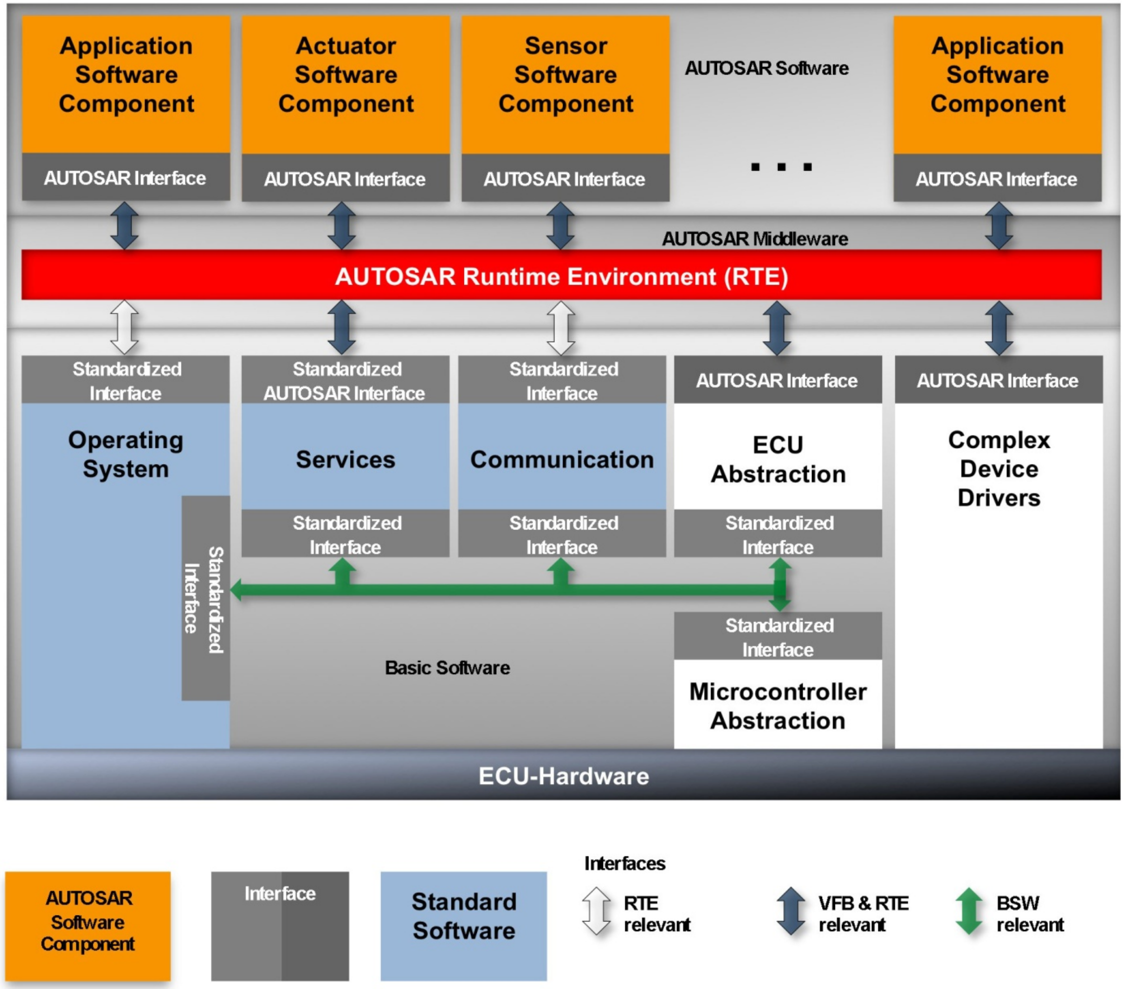
- EB tresos AutoCore OS, by Electrobit Automotive
- ERIKA, by ERIKA Enterprise
- RTA-OSEK, by the Bosch group
- Capital VStar OS, by Siemens
- SAFEOS, by TTTech
- MICROSAR.OS, by Vector Informatik

- Open source initiatives available, such as Trampoline or the Toppers project, etc.

AUTOSAR

- AUTomotive Open System Architecture, AUTOSAR, is a standard for automotive software architectures
- It standardizes:
  - The development process
  - The modelling of ECU and resources
  - The middleware architecture used on ECUs
- It promotes the adoption of model-driven approaches, with automatic code generation
- AUTOSAR OS is the part of the standard that focuses on the operating system
  - It extends OSEK and standardizes the aspects that were considered implementation dependent, such as memory protection

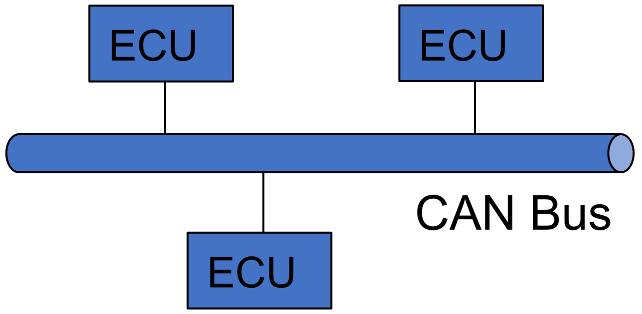
AUTOSAR Classic Platform



- The AUTOSAR software architecture must be:
  - Modular
  - Reusable
  - “Transferable”
  - Scalable

Networking: the CAN bus

- Controller Area Network: *de-facto* standard for ECU networking
  - supported by OSEK (OSEK COM) and AUTOSAR (Communication interface)



- Serial bus with constraints on bps and cable length
- This way, all nodes can "sense" the transmission even of a single bit!
- The channel is *wire-ANDed*: the bit on the channel is "0" if at least one node transmits a "0", otherwise is "1"

CAN Protocol

- Every CAN frame starts with an 11-bits ID, followed by a control field and 8 byte of data

ID	Control	DATA

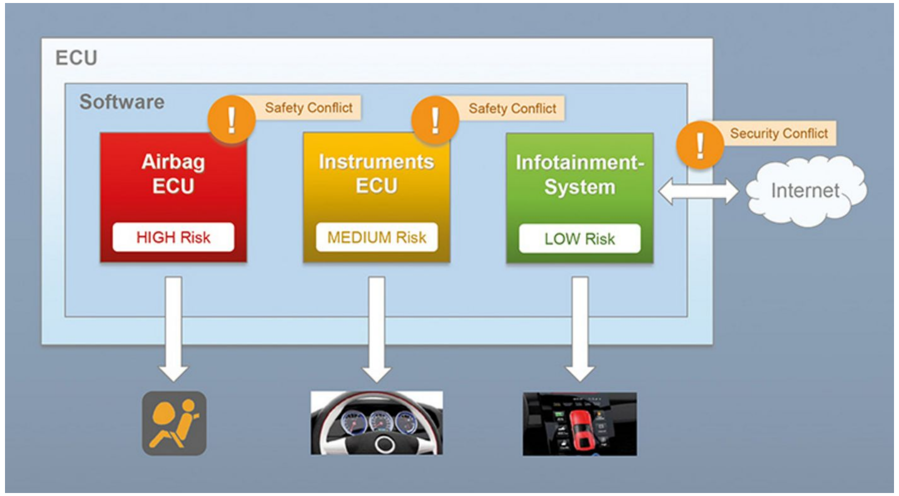
- Frames do not contain addresses
  - A node decides to receive a frame based on the ID
  - Hence, IDs are assigned to ECUs or to information types (e.g., right-front wheel speed)

- CAN Medium Access Control is a modified CSMA/CD
  - A node transmits a frame, starting from the ID, as soon as it sense the channel as free, while keeps listening (*carrier sense*)
  - If the node receives a “0” while it is sending a “1”, it stops the transmission (*collision detect*)
    - This means that another node is transmitting a frame with a lower ID
    - The collision is solved in favor of the lower ID frame
    - In other terms, frame IDs can be assigned according to their priorities

Automotive-grade hypervisors

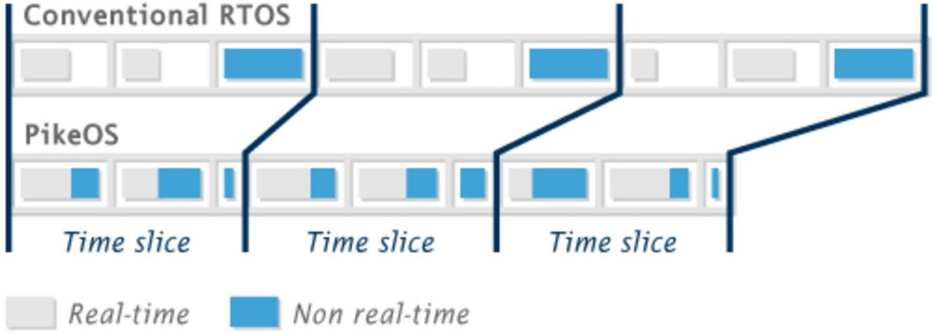
- Many initiatives, mostly commercial, to implement automotive mixed-criticality systems using hypervisor
- Pushed by
  - the adoption of multi-core (and/or AMP) platforms (ECU consolidation)
  - the challenging needs of ADAS, requiring control applications to be integrated with advanced signal/video processing, AI, …
- ISO 26262 compliance (as SEooC or facilitated by integration with AUTOSAR)

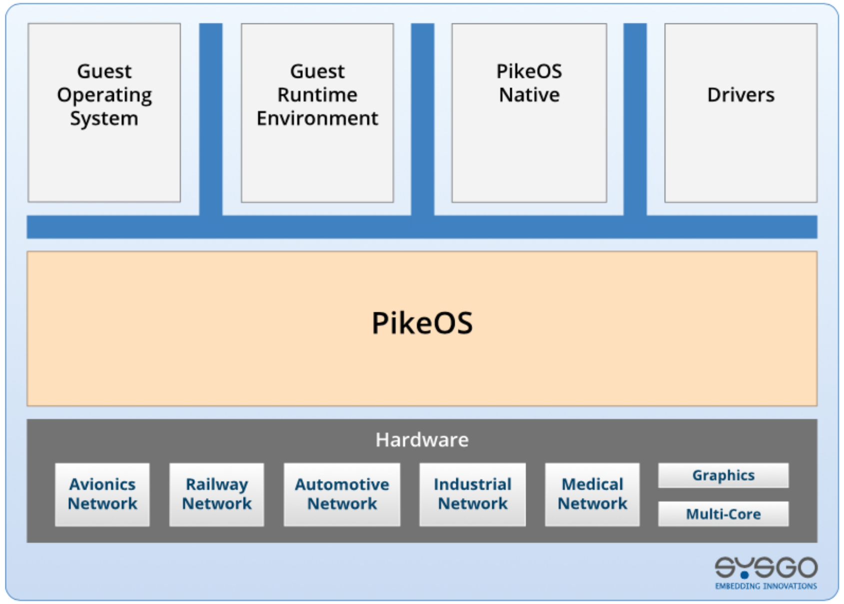
“ECU consolidation” concept



PikeOS

- An RTOS and hypervisor developed by SysGO
- It combines fixed-slice (partitioning) and priority-driven scheduling
- Each crtical VM is assigned a static timeslice and, if a critical (high-prioity) VM currently has no workload, the excess capacity is assigned to a non-critical (low-priority) VM





PikeOS and AUTOSAR

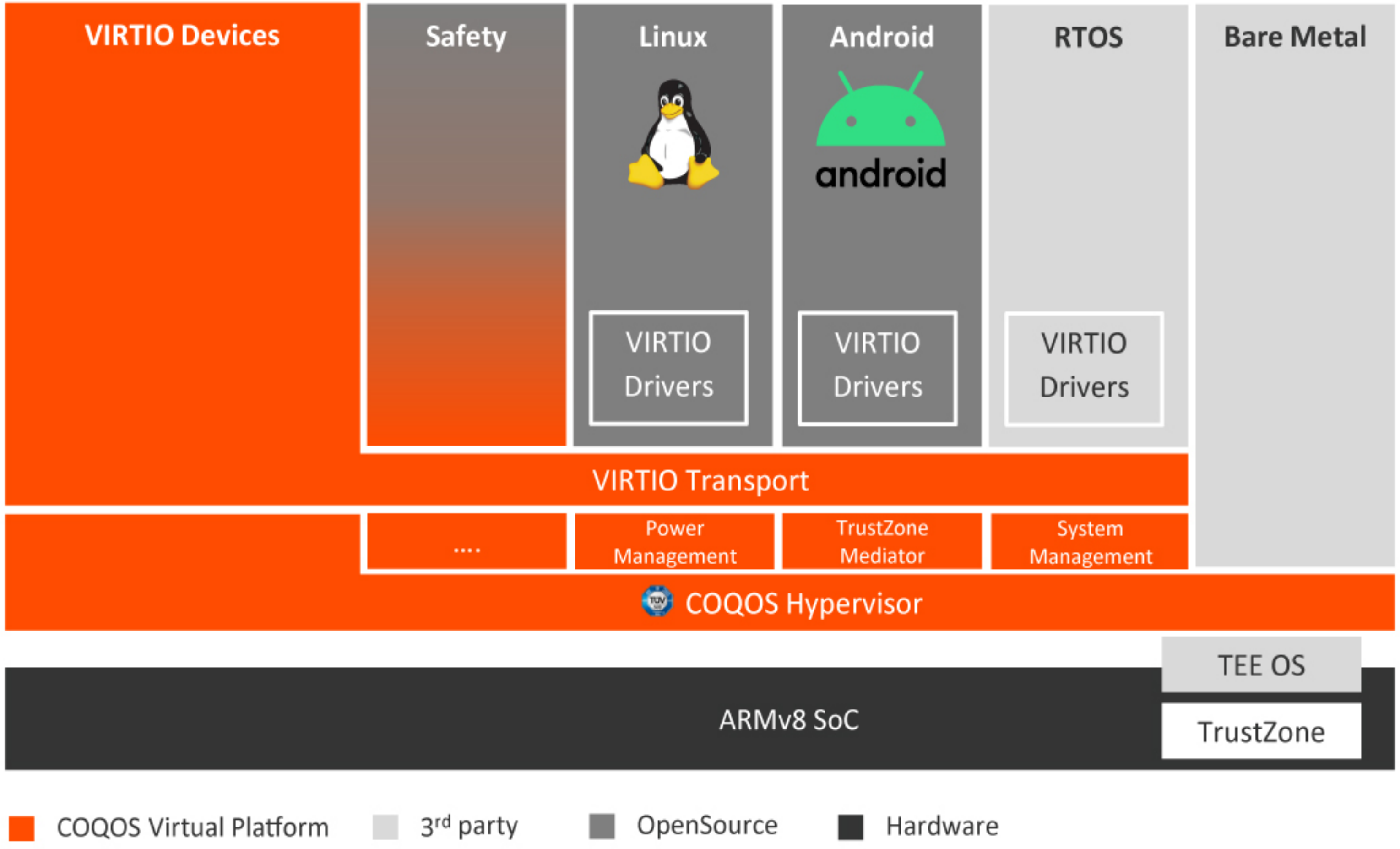
- PikeOS can support AUTOSAR classic platform, running in one of the PikeOS partitions
- MICROSAR: Compliance with AUTOSAR adaptive platfrom available via the SYSGO/Vector Joint Venture, with certification ISO 26262 up to ASIL D. It runs on PikeOS with POSIX partitions.

COQOS hypervisor

- Developed by OpenSynergy
- ISO 26262 certified up to level ASIL B (QM, A, B) as a SEooC

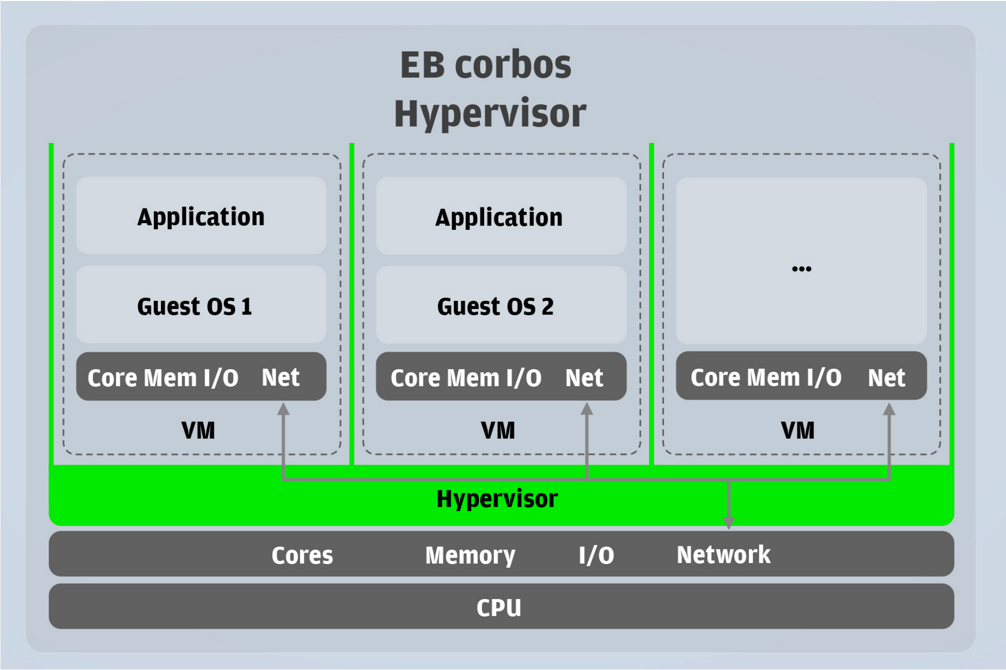
- Distinguishing features:
  - Introduces solutions to share the display and the GPUs among VMs
  - Adopts VirtIO for the virtualization of peripherals

COQOS architecture



CORBOS

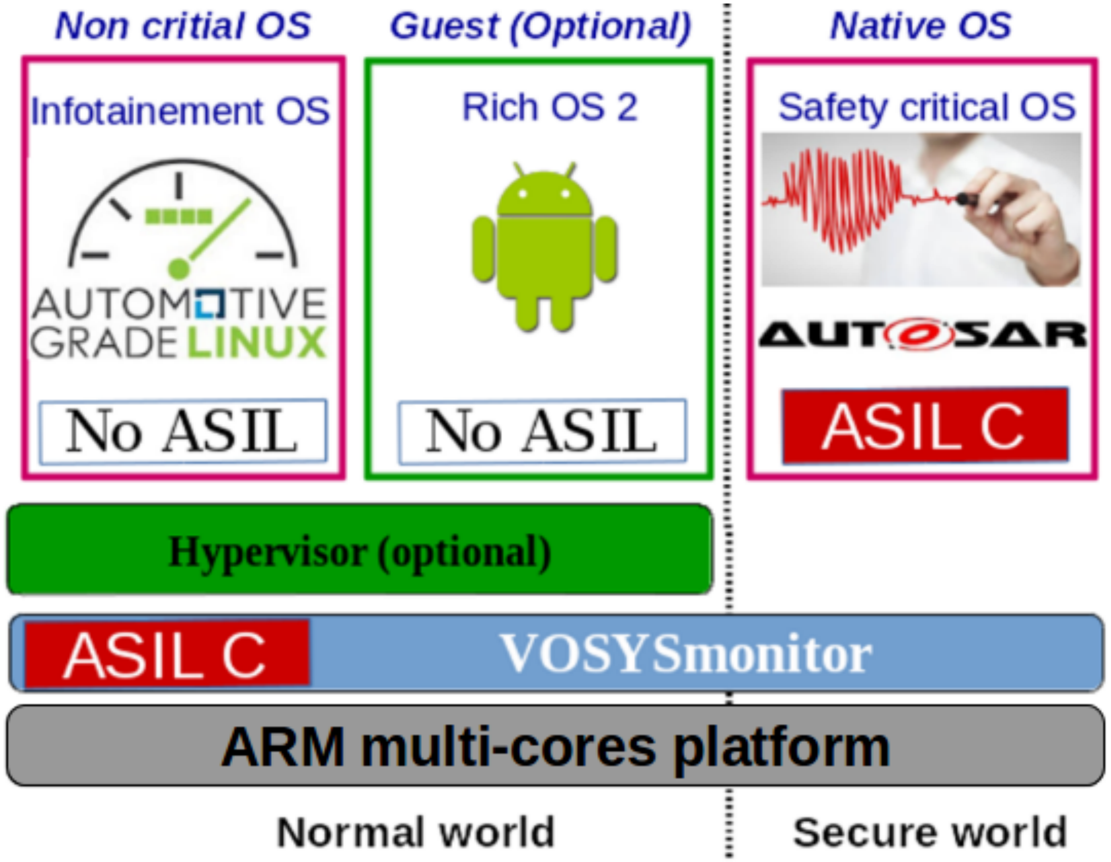
- Developed by Electrobit following the ISO 26262 SEooC dev. process
  - To falicititate integration in safety crticial environments
- Microkernel hypervisor with hw virtualization for VM isolation
- Network-based inter-VM communication



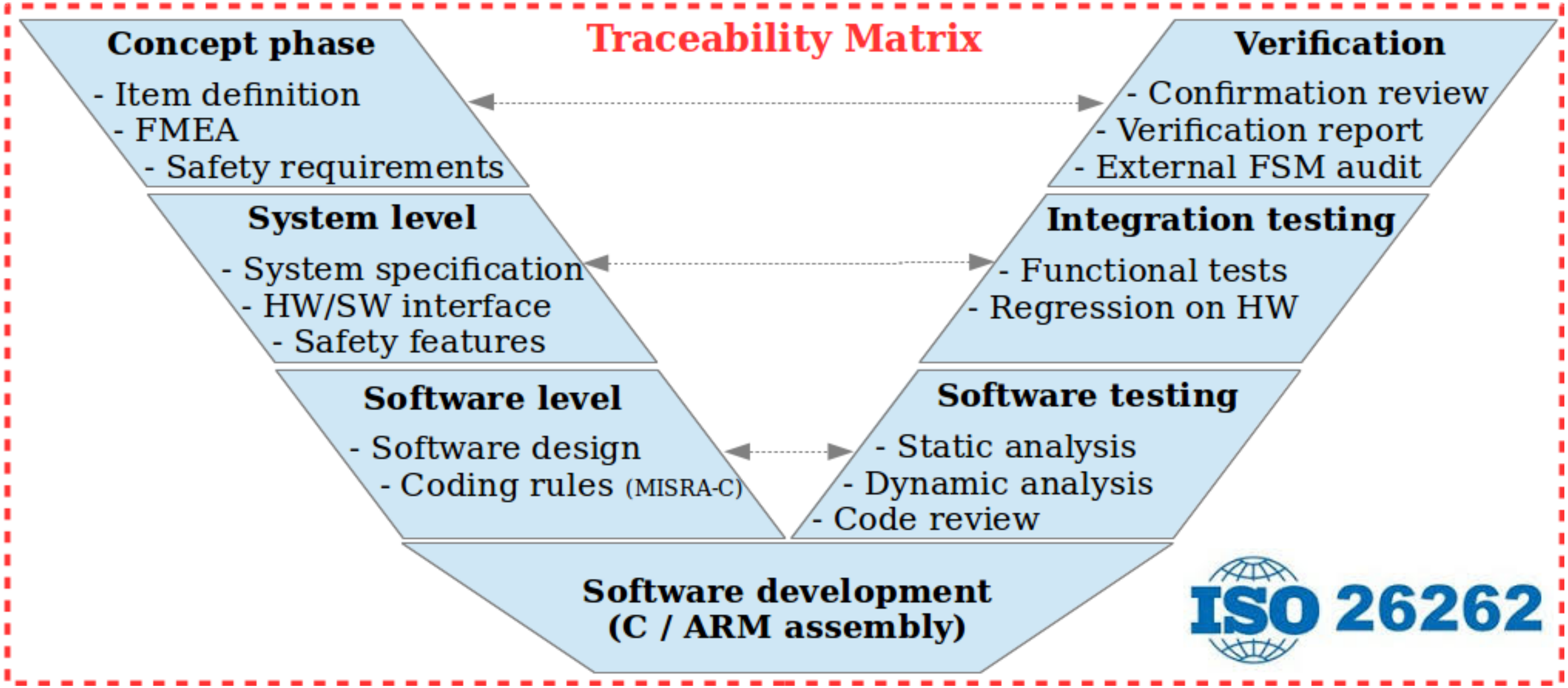
VOSYSMonitor

Virtual Open Systems

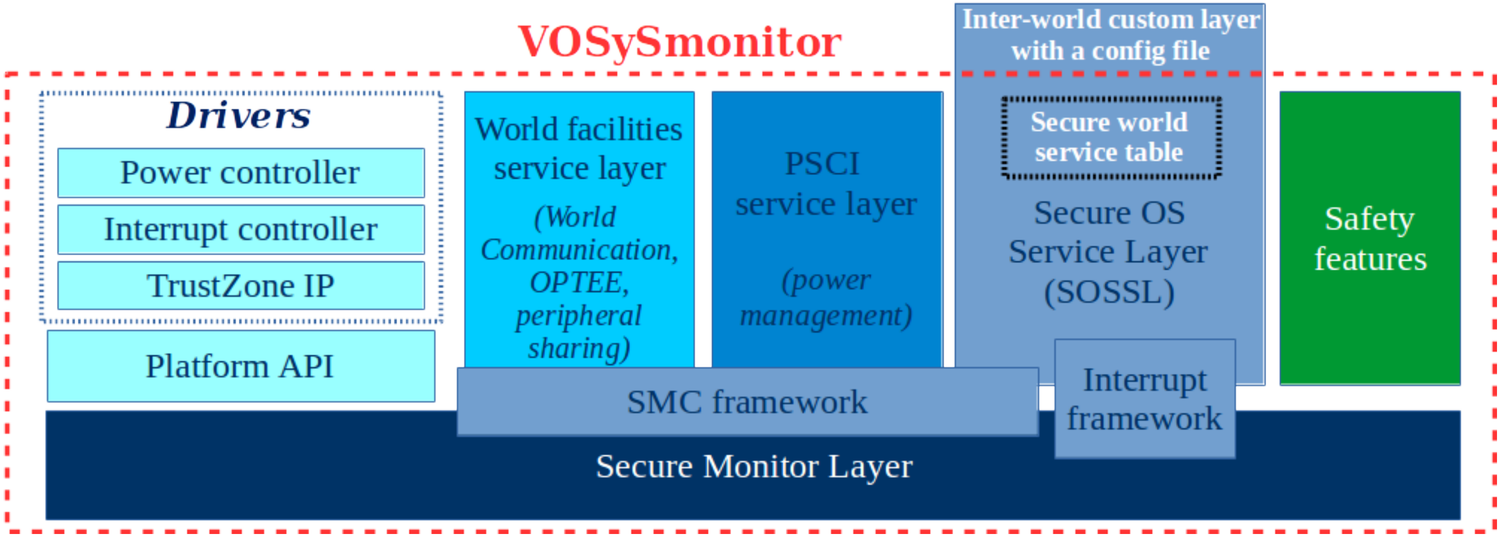
- Developed by Virtual Open Systems following ISO 26262 up to ASIL C
- Based on on the ARM TrustZone technology, which enforces CPU and interrupt isolation between the RTOS (running in the secure world) and the GPOS (running on the normal world)



VOSYSMonitor development process



VOSYSMonitor: hypervisor architecture



- **SOSSL:** Interface between the Secure OS and the monitor layer to dispatch SMC services as well as to handle interrupts forwarding.
- **PSCI:** Service layer for power management services (Power ON/OFF cores, etc).
- **World facilities Service Layer:** VOSYSmonitor custom services (e.g., inter-world communication channel) in order to provide advanced features to systems running in each world.
- **Safety features:** Countermeasures (e.g., safe state, self-tests, etc) to preserve the Secure OS in case of internal / hardware failures.