# Avionics, Railways, and the Industrial Internet of Things

## Standards and technologies

Real-Time Industrial Systems

Marcello Cinque

# Avionics industry

# Standards in the avionics industry

- Safety standards
  - DO-168B and DO-168C
- Platform standards
  - ARINC 653

# DO-178B

- «Software Considerations in Airborne Systems and Equipment Certification»
- Defined by the safety-critical working group RTCA SC-167 of the Radio Technical Commission for Aeronautics (RTCA) and WG-12 of the European Organisation for Civil Aviation Equipment (EUROCAE)
- Adopted by the Federal Aviation Administration (FAA)
- is a guideline dealing with the safety of safety-critical software used in airborne systems
- it standardizes design and production processes

# DO-178B: processes

- DO-178B defines three software lifecycle processes
  - Software Planning process
  - Software Development process
    - Requirements
    - Design
    - Coding
    - Integration/Test
  - Integral processes
    - Software Verification
    - Software Configuration Management
    - Software Quality Assurance
    - Certification Liason

# DO-178B: contents

- Per each of di such processes the standard defines:
  - The objectives to reach, also in terms of outputs to be produced;
  - The activities necessary to reach such objectives
  - The description of evidences needed to demonstrate that objectives have been reached.

# DO-178B: software levels and failrues

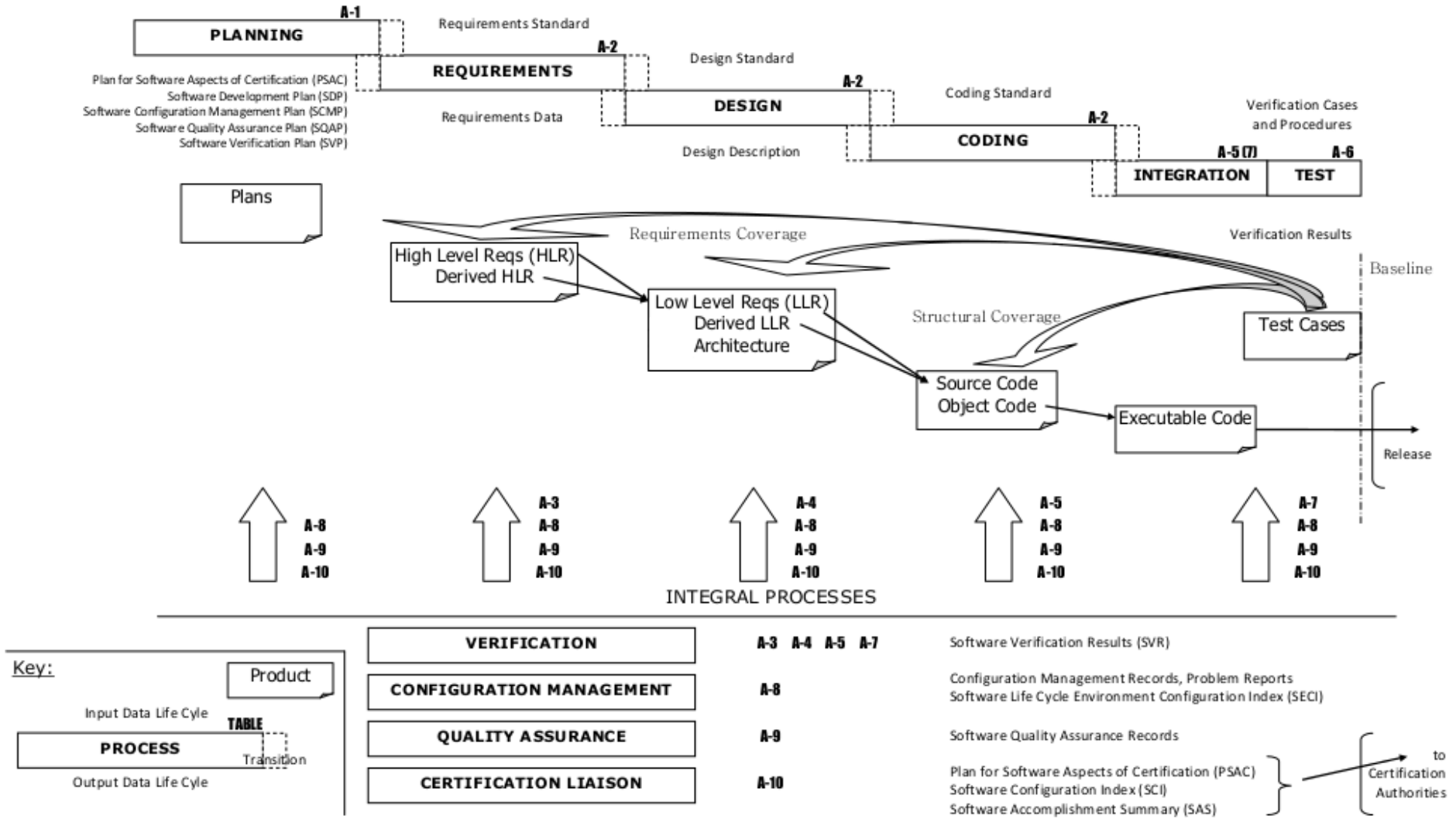
- Five software levels
  - **Level A**: Software at this level can lead to a catastrophic failure, e.g. loss of aircraft.
  - **Level B**: Software at this level can lead to hazardous failure, e.g., ample reduction of aircraft functionality; injuries to passengers and the crew.
  - **Level C**: Software at this level can lead to a major failure, e.g., reduction of the functionality of the aircraft.
  - **Level D**: Software at this level can lead to a minor failure, ., reduction of the functionality of the aircraft, change of the flight plan.
  - **Level E**: Software at this level does not lead to failure conditions (no effect).

# DO-178B: software planning

**Table A-1 Software Planning Process**

| | Objective | | Activity | Applicability by Software Level | | | | Output | |
|---|---|---|---|---|---|---|---|---|---|
| | Description | Ref | Ref | A | B | C | D | Data Item | Ref |
| 1 | The activities of the software life cycle processes are defined. | 4.1.a | 4.2.a<br>4.2.c<br>4.2.d<br>4.2.e<br>4.2.g<br>4.2.i<br>4.2.l<br>4.3.c | ○ | ○ | ○ | ○ | PSAC<br>SDP<br>SVP<br>SCM Plan<br>SQA Plan | 11.1<br>11.2<br>11.3<br>11.4<br>11.5 |
| 2 | The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined. | 4.1.b | 4.2i<br>4.3.b | ○ | ○ | ○ | | PSAC<br>SDP<br>SVP<br>SCM Plan<br>SQA Plan | 11.1<br>11.2<br>11.3<br>11.4<br>11.5 |

# DO-178B: Development and test process



# DO-178B: Certification process

- Submission of the “Plan for Software Aspects of Certification” (PSAC), containing the description of the system, the software, the adopted development process, …
- Resolution of PSAC’s problems identified by the Certification Authority
- Achieve the PSAC acceptance and submission of the Software Accomplishment Summary (SAS) and of the Software Configuration Index (SCI)
  - SAS: System functions and their allocation to the hardware and software; Architecture; Processor(s); Hardware and software interfaces; Safety features.
    - The primary purpose of SAS is to show compliance to the plans and processes set forth in the PSAC. The SAS demonstrates to the Certification Authority that the objectives set forth in the planning documents have been achieved. Any deviation from the plans that has not been approved by the Certification Authority needs to be clearly described in the SAS.
  - SCI: identifies the configuration of the software and should identify the following: Software product; Executable object code; Source code components; Previously developed software; Software life cycle data; etc.
- The Certificaiton entity verifies if the system to be certified conforms to what defined in the standard, analyzing SAS and SCI, the development process and its outputs.

# DO-178B/C: examples of certified OS

- VxWorks by WindRiver
- Integrity-178 OS by Green Hills Software
- LynxOS-178
- PikeOS by SysGo

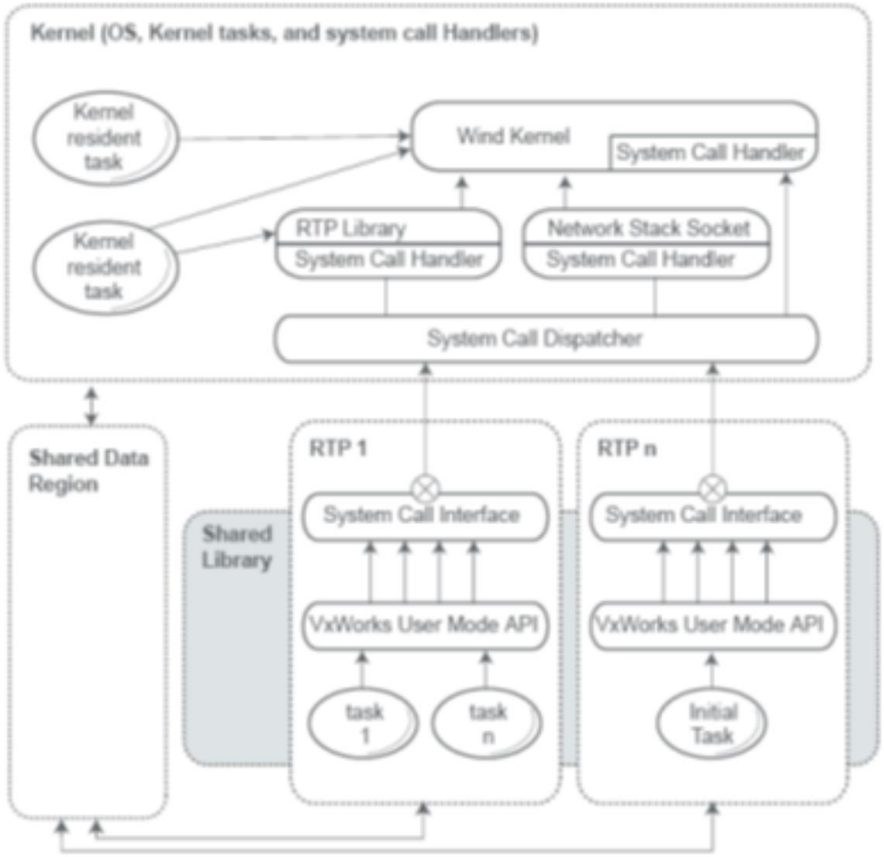
# VxWorks by WIND RIVER

- Main features:
  - Flat memory model
  - Priority preemptive scheduler
  - Multitasking kernel
  - User level real-time processes (RTPs)
  - Mutual exclusion with locks and semaphores
  - Inter task communication with queues
  - Watchdog timers
  - User interface shell

```
## Starting application at 0x4010100000 ...

 _____                  _____
 \77777777\            /77777777/
  \77777777\          /77777777/
   \7777777\        /7777777/
    \77777777\    /77777777/
     \77777777\   \7777777/
      \7777777\   \77777/            VxWorks 7 SMP 64-bit
       \77777777\  \777/
        \77777777\  \7/     Core Kernel version: 1.0.0.0
         \7777777\   -      Build date: May 30 2014 10:51:05
          \77777777\
           \7777777/        Copyright Wind River Systems, Inc.
            \77777/   -               1984-2014
             \777/   /7\
              \7/   /777\
               -   -------

           Board: Wind River Dev Kit MP8
       CPU Count: 8
  OS Memory Size: 1899MB
ED&R Policy Mode: Deployed

Adding 5290 symbols for standalone.

[vxWorks]# i
  NAME            TID       PRI   STATUS        PC           ERRNO  CPU #
---------- ---------------- --- ---------- ---------------- ------- -----
tJobTask    40104cdbc0         0 PEND       401020c83c             0     -
tExcTask    40102a073c         0 PEND       401020c83c             0     -
tLogTask    40104d01d8         0 PEND       401020b0f0             0     -
tShell0     40105c1d30         1 READY      4010215e08             0     0
ipcom_tick> 401057a990        20 PEND       401020c83c             0     -
tVxdbgTask  401057dc20        25 PEND       401020c83c             0     -
tNet0       40104d3b78        50 PEND       401020c2b4             0     -
ipcom_sysl> 40104c9810        50 PEND       401020d3d4             0     -
tNetConf    40105a6e40        50 PEND       401020c83c             0     -
miiBusMoni> 40104d5e08       252 DELAY      4010215640             0     -
ipcom_egd   4010583c20       255 DELAY      4010215640             0     -
tIdleTask0  40102a2fb0       287 READY      401020c004             0     -
tIdleTask1  40102a7220       287 READY      401020c00c             0     1
tIdleTask2  40102ab490       287 READY      401020c004             0     2
tIdleTask3  40102afb20       287 READY      401020c004             0     3
tIdleTask4  40102b1700       287 READY      401020c004             0     4
tIdleTask5  40102b2440       287 READY      401020c004             0     5
tIdleTask6  40102a4620       287 READY      401020c004             0     6
tIdleTask7  40102a4860       287 READY      401020c004             0     7
[vxWorks]#
```

# VxWorks Real-Time Process (RTP)

- Traditionally, VxWorks supported a lightweight kernel-level threading model, with flat shared memory model
- RTP introduced to have a protected user mode area where applications can execute
- RTPs are isolated from the kernel and between each other, but can share libraries and data regions
- RTPs are not schedulable entities but may contain one or more tasks

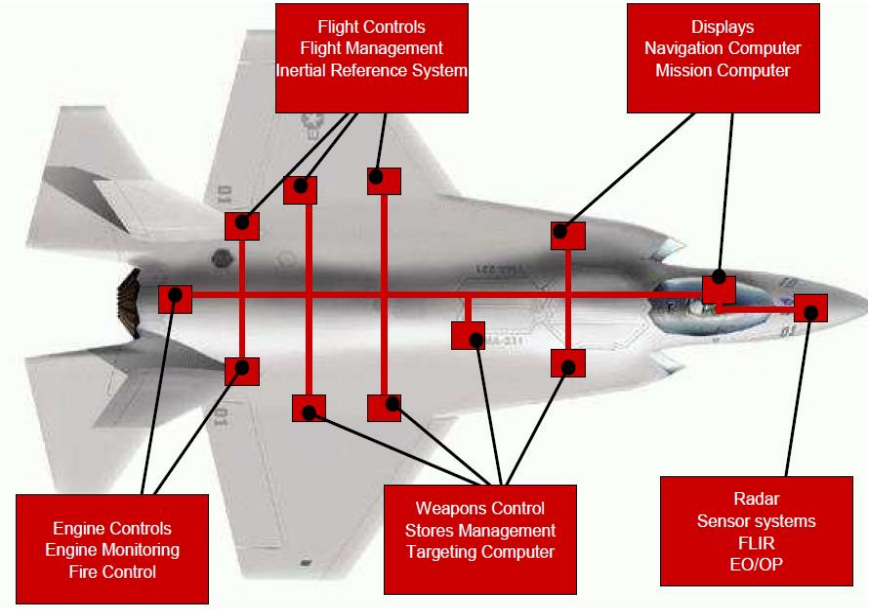


# ARINC-653

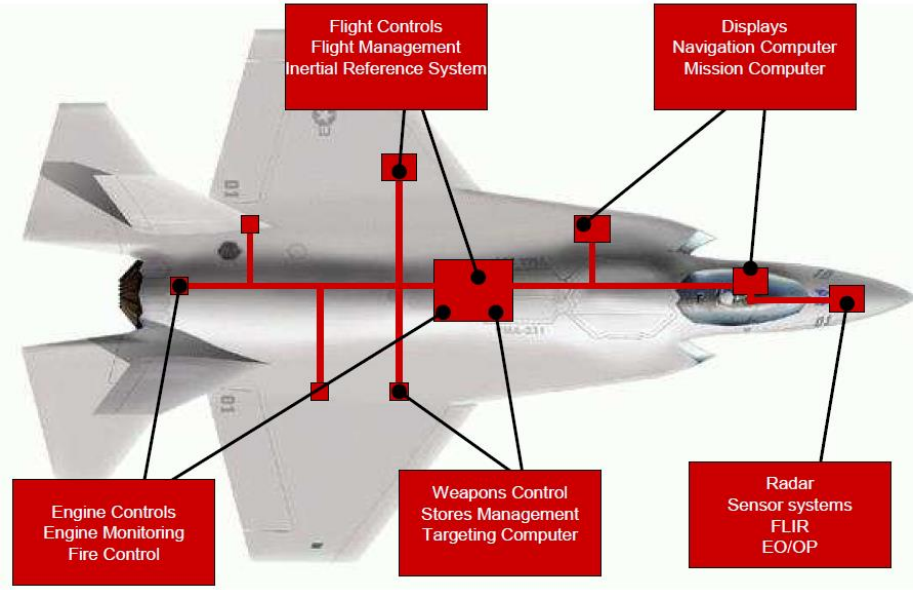
- Similarly to the automotive industry, the avionic industry is moving from a federated approach to an integration of multiple software systems on the same processing unit.
- In order to have more functionalities, more connectivity,..., in less space, weight, and power (SWaP)
  - Harware consolidation: multiple applications on fewer processors
  - Software pressure: larger volume of software on fewer processors
  - With new challenges to safety and security!
- ARINC 653 OS and applications are typically certified for DO-178B

# Federated vs IMA

**Federated Architecture**




**Integrated Modular Architecture (IMA)**



# ARINC 653 and DO-178B

- To meet software certification requirement of DO- 178B, 3 main needs have been identified
  - Safety-critical – according to a law
  - Real-Time – response times must be within a predetermined time period
  - Deterministic – results of the execution must be predictable and repeatable
- ARINC 653's RTOS guarantee an interface boundary for avionics software development, thus allowing independence of the avionics software applications.

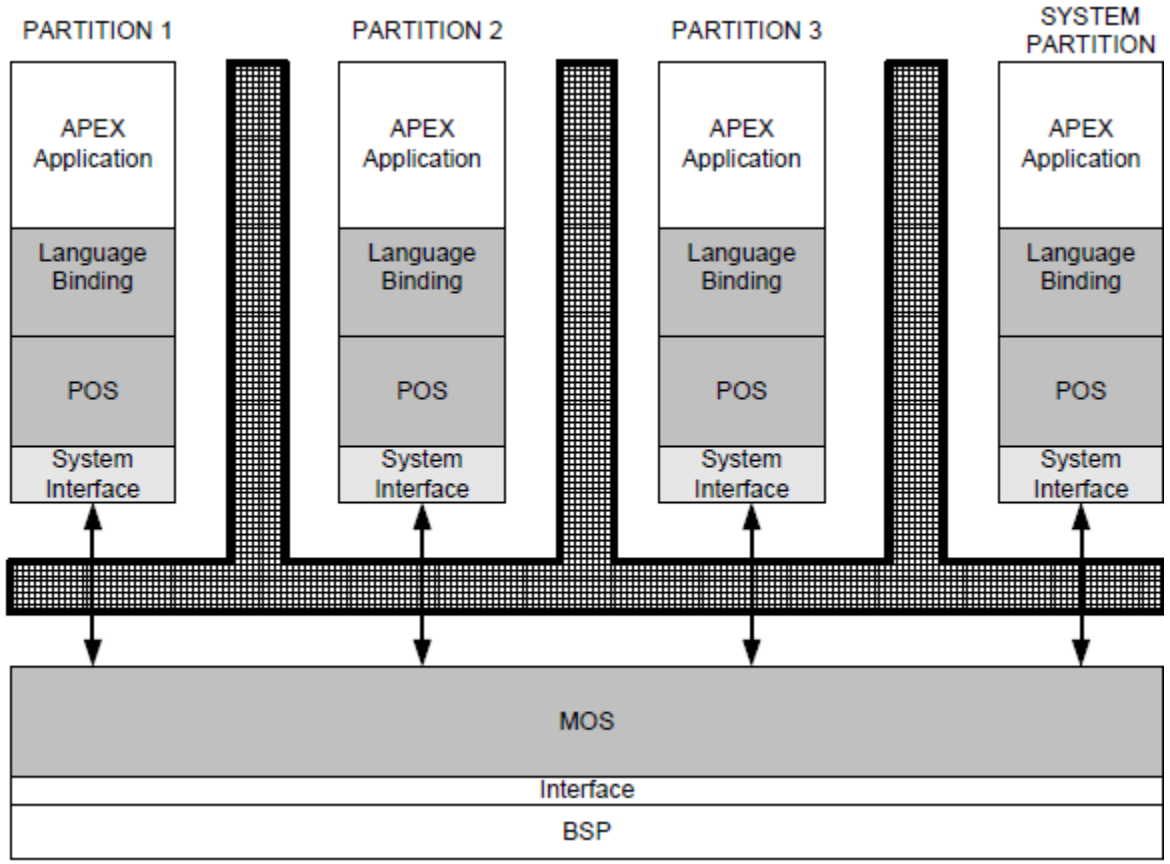
# APEX (APplications EXecutive)

- ARINC's programming interface to separate applications from the OS

- Routines for:
  - Time and space (memory) partitioning;
  - Health monitoring (error detection and reporting);
  - Communications via "ports".

- API available for C and Ada.

# ARINC: partitioning




- POS, Partition Operating System
- MOS, Module Operating System
- BSP, Board Support Package

**Architecture:**

- **Spatial isolation**: partiotions allocated to disjoint memory addresses
- **Process temporal isolation**: within each partition the POS schedules the processes
- **Partition temporal isolation**: the MOS cyclically schedules the partitions, according to an off-line algorithm

# ARINC 653 Services

- Partition management
  - Partitioning is the main concept of ARINC-653: execution environment with separate memory space and strictly protected in time;
  - All the resources used by a partition have to be defined at system configuration time, and created and defined in the initialization phase of the partition.
  - Example of services: get partition status, set partition mode, ...

Credits: A. Biondi – ARINC 653

# ARINC 653 Services

- Process management
  - A partition comprises one or more processes;
  - Typically the processes are scheduled according to Fixed- Priority preemptive (or limited preemptive) policy;
  - An ARINC 653 process can be in one of 4 available states
    - Dormant – ineligible for scheduling;
    - Waiting – not able to execute;
    - Ready – able to be executed;
    - Running – currently executing.
  - Typical operations: create process and collect process status or ID; start, stop, suspend or resume the process; change the process priority.

Credits: A. Biondi – ARINC 653

# ARINC 653 Services

- Time management
  - From the standard: “Time is unique and independent of partition execution within a core module. All values or capacities are related to this unique time and are not relative to any partition execution.”
  - GET_TIME to read the current system time;
  - Wait and time-out mechanism;
  - Budget management for hard real-time tasks (time capacity);
  - Periodicity specification.

Credits: A. Biondi – ARINC 653

# ARINC 653 Services

- Health Monitoring
  - Reporting and monitoring errors and exceptions;
  - The error handling is the highest priority process and it is invoked whenever a fault takes place;
  - Error handlers must be defined to manage an error, defining how a partition should respond.
  - Example of error handling:
    - Log the error;
    - Stop or restart the failed process;
    - Eventually stop or restart the entire partition;
    - Invoke the registered handler for the specific error code

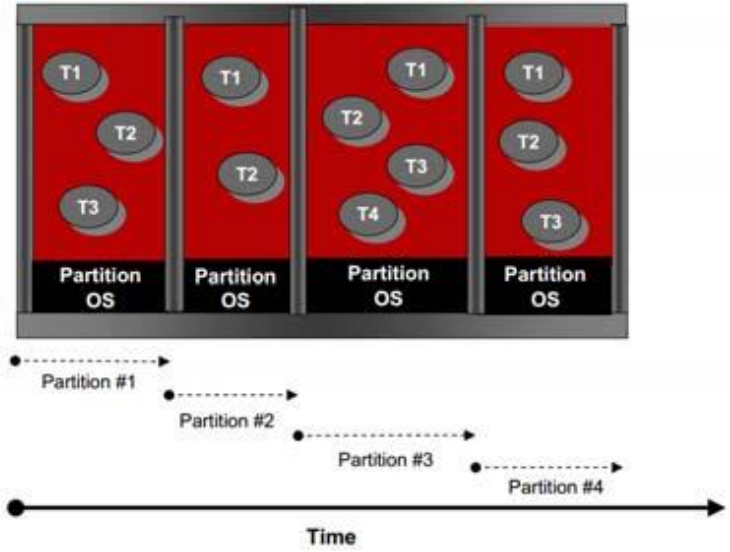
Credits: A. Biondi – ARINC 653

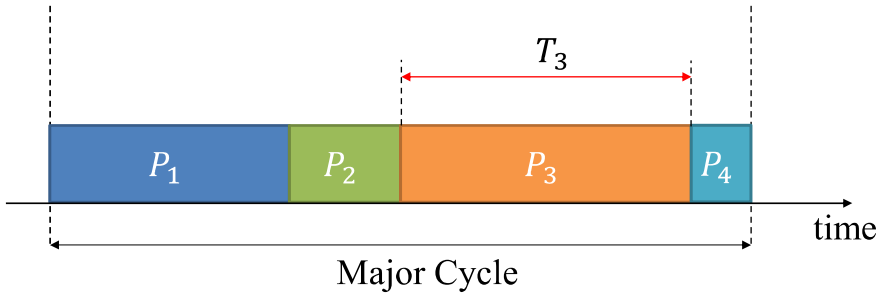
# ARINC 653 Services

- Communication:
  - **Intra-partition**. Processes in the same partition communicate with typical IPC mechanisms (buffers, semaphores); the communicaiton is managed by the POS
  - **Inter-partition**. Processes belonging to different parittions exchange messages trough logical ports or physical channels
    - Messages are of two types: *Sampling*, rewritable but not consumables, or *Queuing*, enqueued in a FIFO queue and consumed when read

# ARINC 653 OS Scheduling

- Two levels hierarchical scheduling:
  - Time division scheduling of a predermined set of parititions $P_1$, …, $P_n$
  - On top of each partition runs a specific OS scheduler (tipically Fixed- Priority)

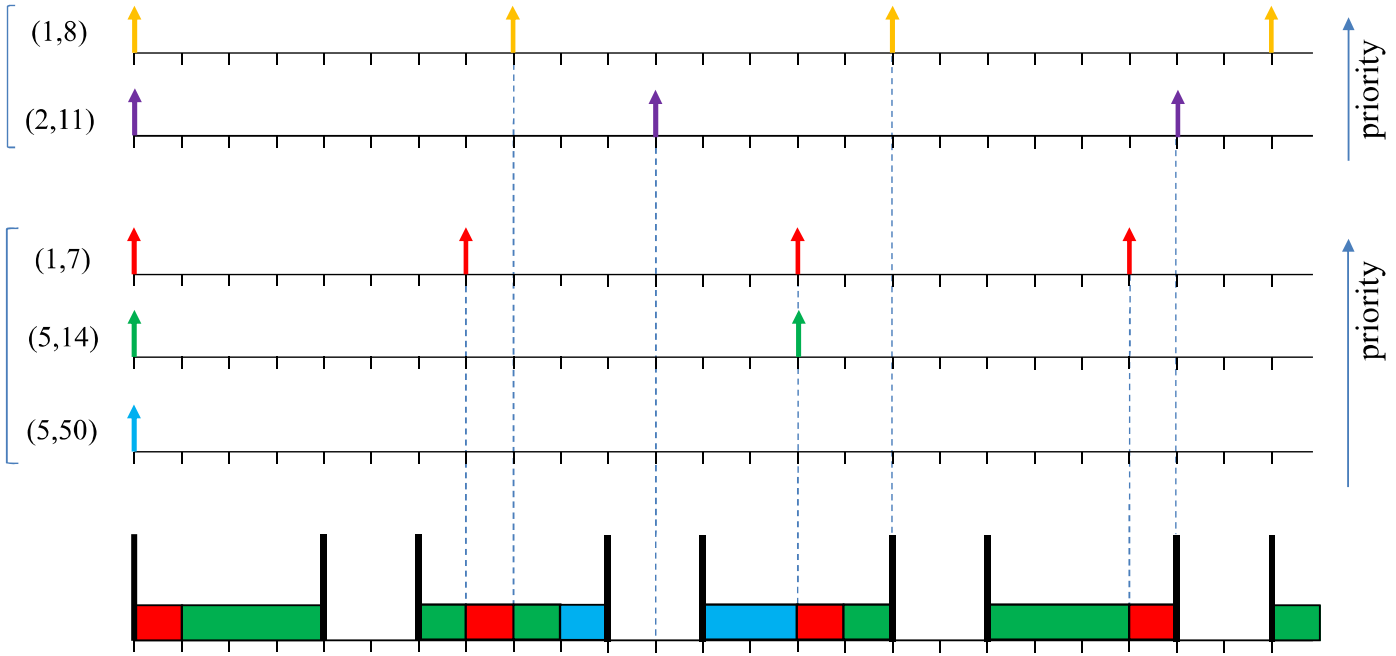



Credits: A. Biondi – ARINC 653

# ARINC 653 OS Scheduling

- Example: 2 partitions with periods 4 and 2




Credits: A. Biondi – ARINC 653

# Avionics Networking

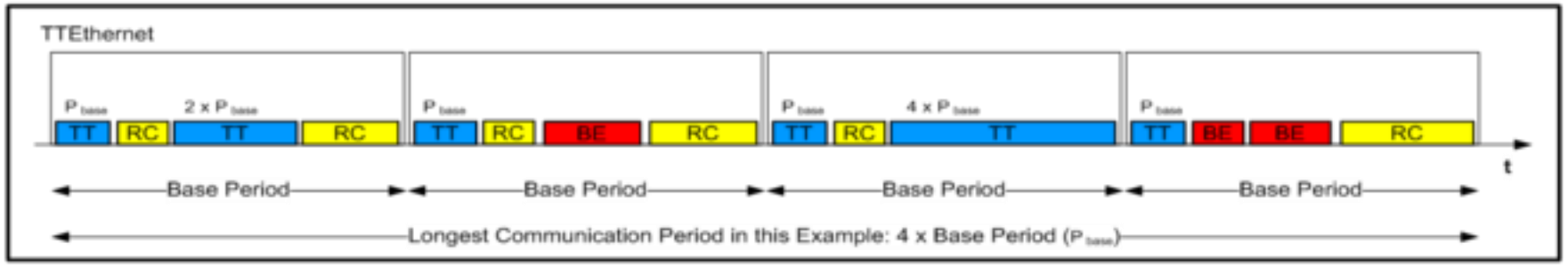
- The federated architecture was based on point-to-point communication with dedicated *avionics data bus* standardized according to ARINC 429
  - Unidirectional bus with 1 transmitter and up to 20 receivers
- The advent of the IMA architecture required deterministic full duplex communication on shared media

# AFDX (ARINC 664)

- Avionics Full DupleX switched ethernet
- Based on Ethernet (IEEE 802.3), to re-use cheap COTS hardware
- The MAC protocol is modified to become deterministic
- **Virtual Links (VLs)**: channel between one transmitter and several receivers with guaranteed bandwidth, limited latency and jitter, based on a static path
- **BAG (Bandwidth Allocation Gap)**: minimum delay between two consecutive frames (from 1ms to 128ms with increasing payload size, from 17 to 1471 octects)
- **Switched network** to avoid collision domains and as second line of defence:
  - End-Device send conformant traffic (Shaping)
  - Every switch enforces conformance (Policing)
  - No “healing” of ill-behaving streams, just drop

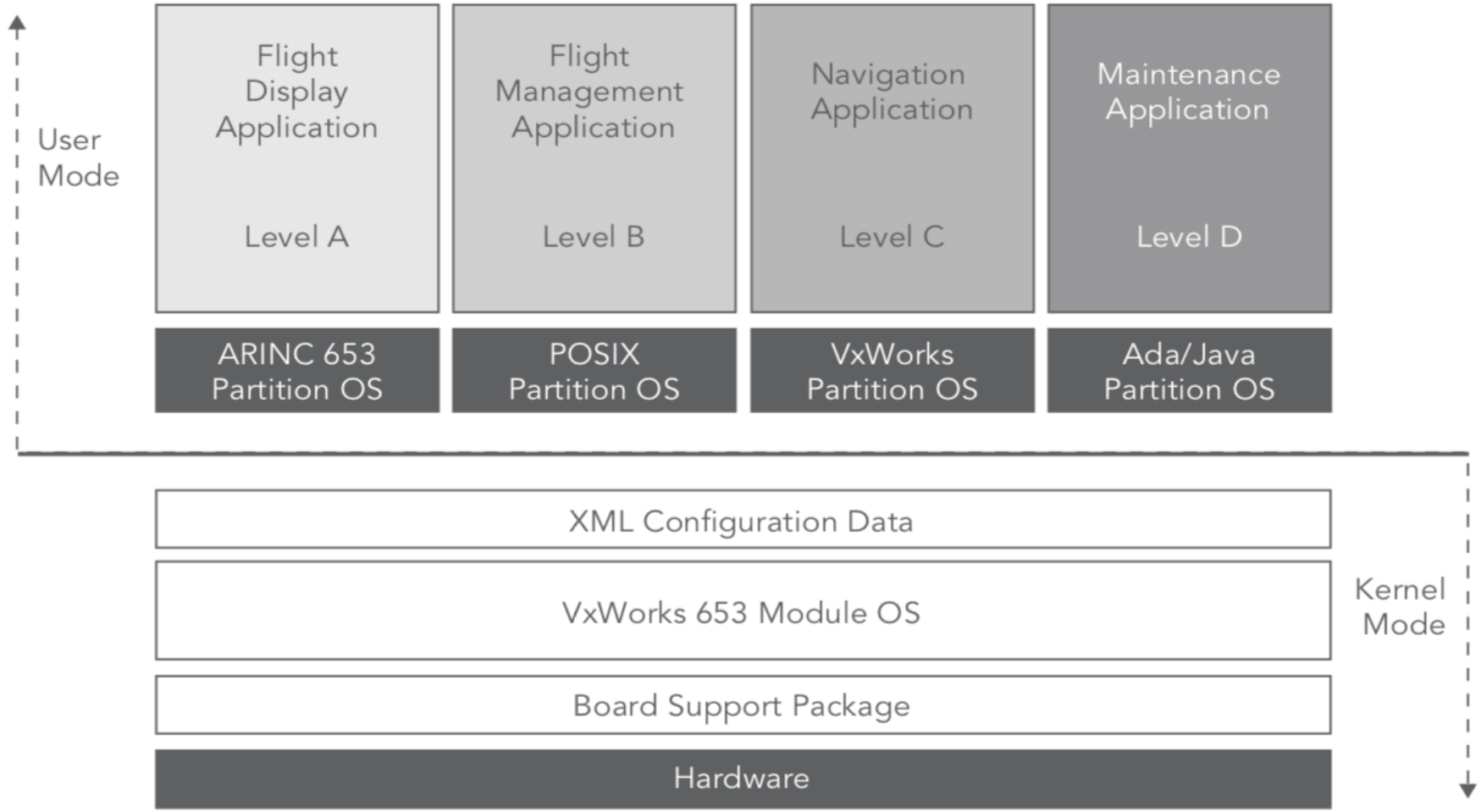
# TTEthernet


- Time-Triggered Ethernet by TTTech
- Re-defines the MAC with a time division multiple access
  - Time-triggered traffic: packets sent at predefined (scheduled) times
  - Rate-constrained traffic: to guarantee maximum latency and jitter
  - Best effort: the remaining traffic

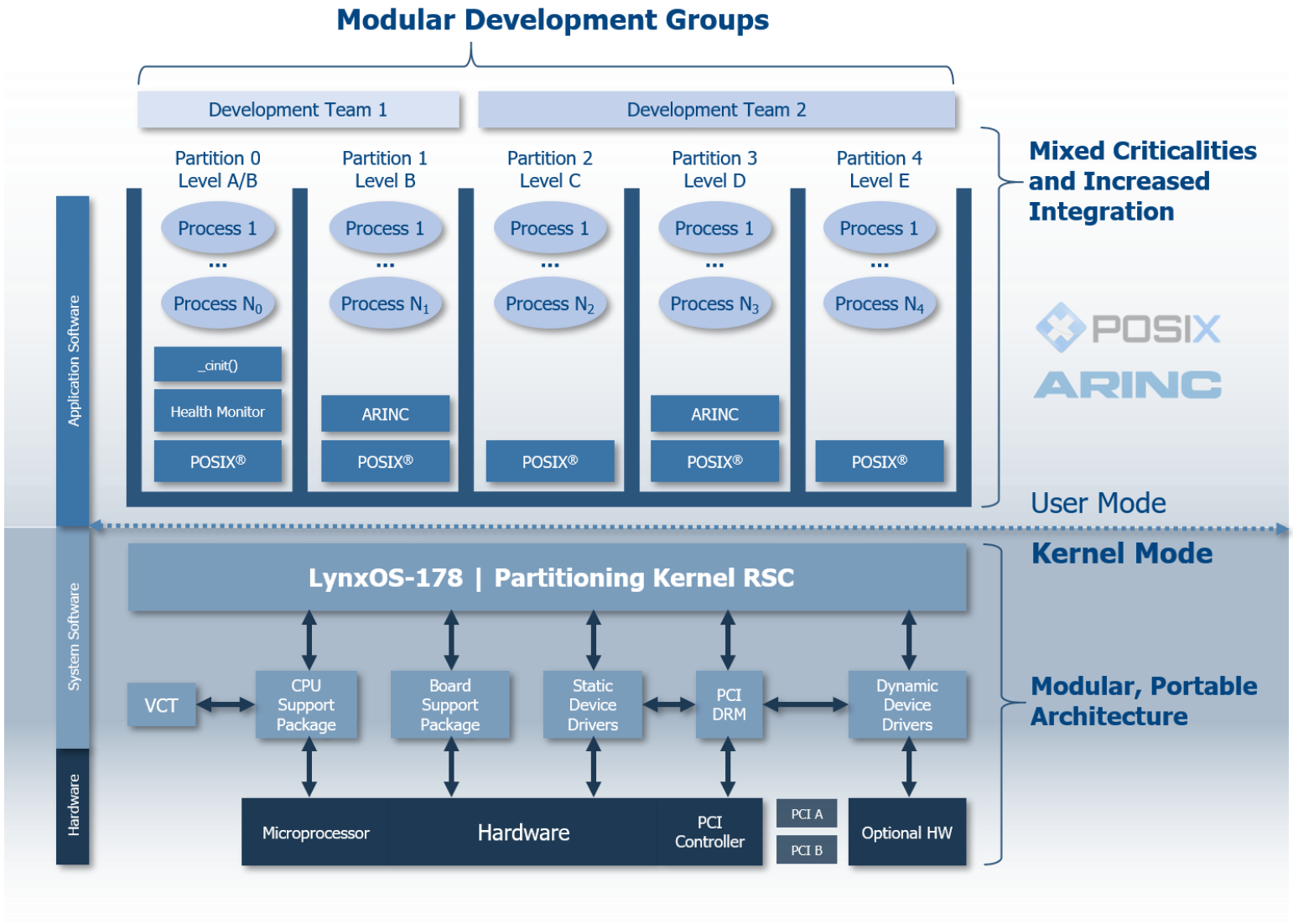


# Examples of ARINC 653 Platforms

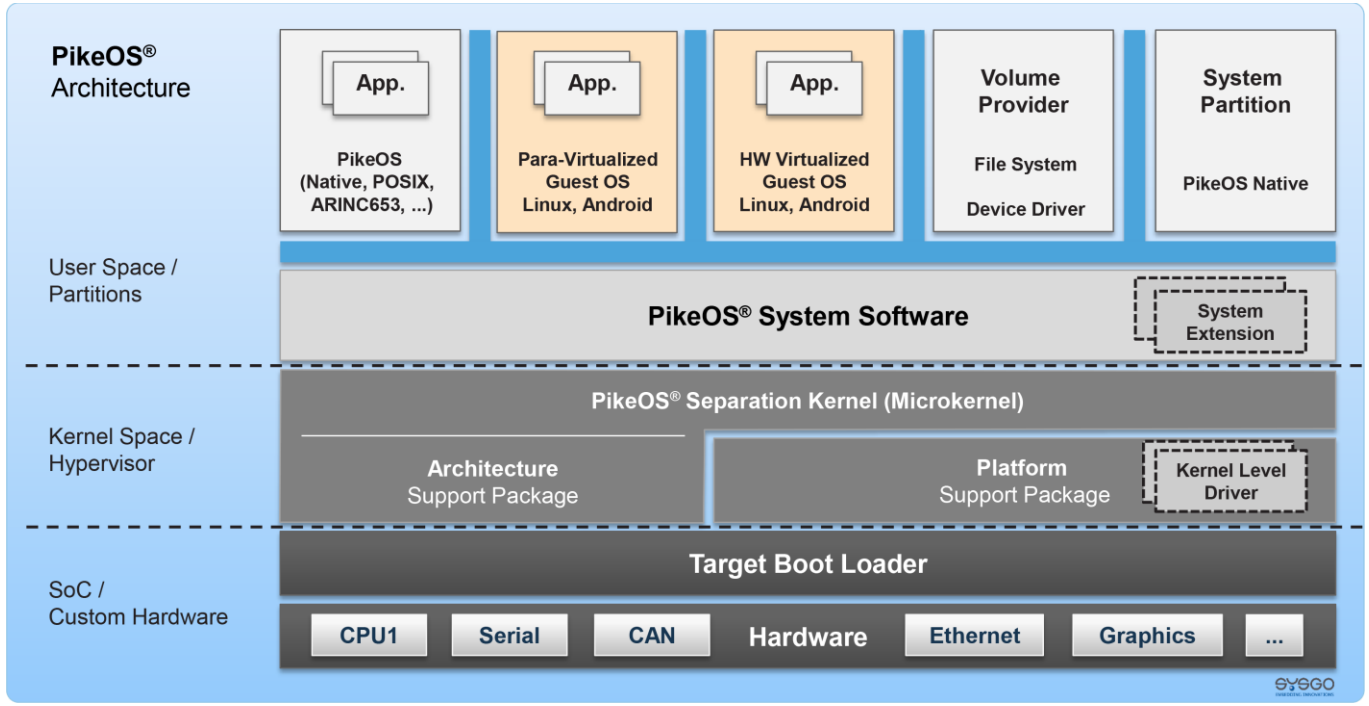
# VxWorks 653 Platform

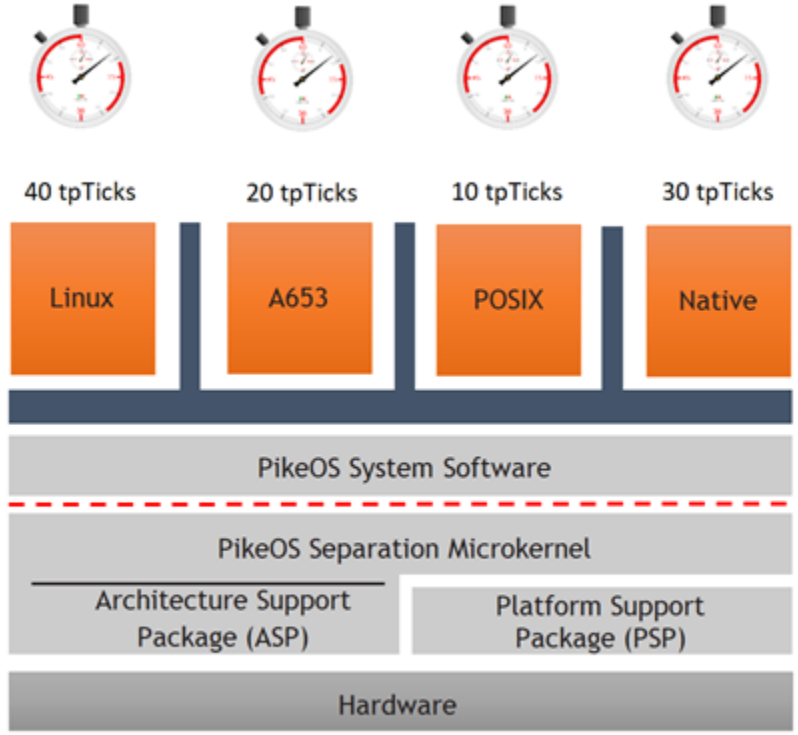


# LynxOS-178


Modular Development Groups
Development Team 1
Development Team 2
Partition 0 Level A/B
Partition 1 Level B
Partition 2 Level C
Partition 3 Level D
Partition 4 Level E
Process 1
Process N0
Process N1
Process N2
Process N3
Process N4
_cinit()
Health Monitor
ARINC
POSIX®
Application Software
Mixed Criticalities and Increased Integration
POSIX
ARINC
User Mode
Kernel Mode
LynxOS-178 | Partitioning Kernel RSC
System Software
VCT
CPU Support Package
Board Support Package
Static Device Drivers
PCI DRM
Dynamic Device Drivers
Modular, Portable Architecture
Hardware
Microprocessor
Hardware
PCI Controller
PCI A
PCI B
Optional HW

# PikeOS with time partitioning

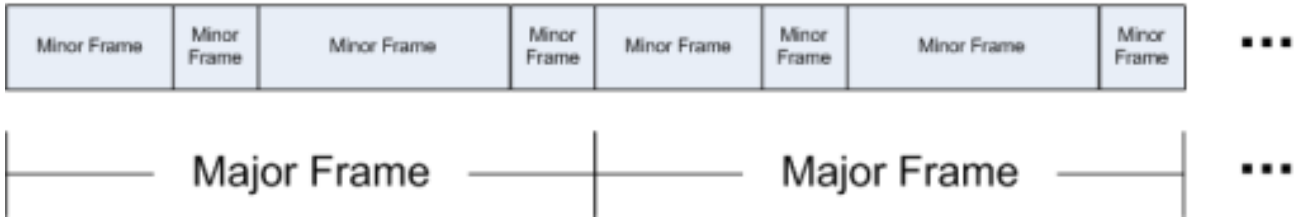



Example for Time Partitioning

# Xen's ARINC653 Scheduler

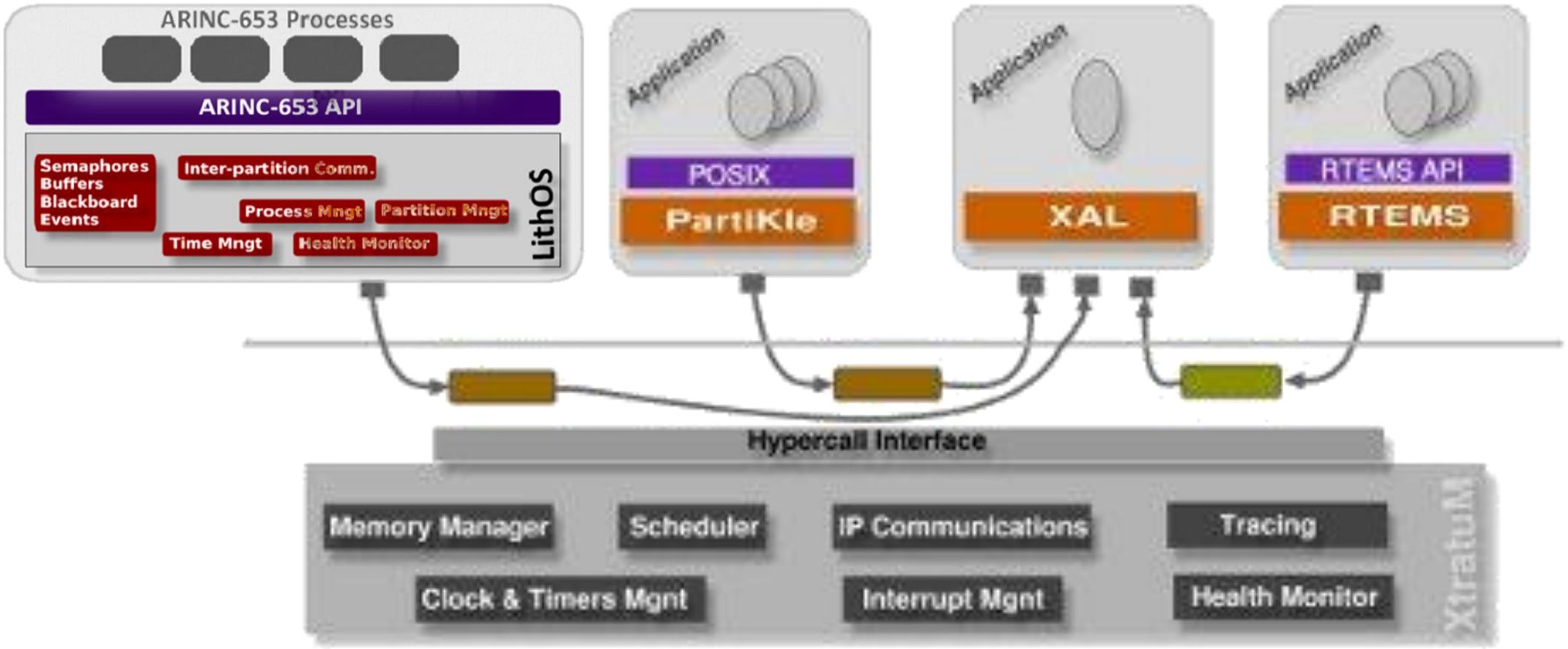
- Cyclic executive of VCPUs
- The schedule configuration sets the major frame duration, as well as the sequence of minor frames, specified by a domain/timeslice pair.
- At the beginning of the major frame, the scheduler runs the VCPU corresponding with the domain of the first minor frame within the schedule. When it's timeslice has completed the scheduler runs the VCPU corresponding with the domain of the next minor frame within the schedule until there are no more minor frames remaining or until the major frame has expired. When the major frame expires the process is restarted.



# XtratuM

- Real-time paravirtualized hypervisor designed following ARINC 653
- Strong temporal isolation: cyclic executive scheduling of partitions
- Strong spatial isolation: only the hypervisor runs in supervisor mode, while the partitions have access to independent memory regions
- Communication implemented according to the ARINC's port model
- All interrupts (including fault traps) managed by the hypervisor
- Non-preemptable: monolithic hypervisor that cannot be interrupted (no internal race conditions)
- Small size to simplify certification processes
- Fine grain allocation of hardware resources to partitions
- Deterministic hypercalls

# XtratuM architecture

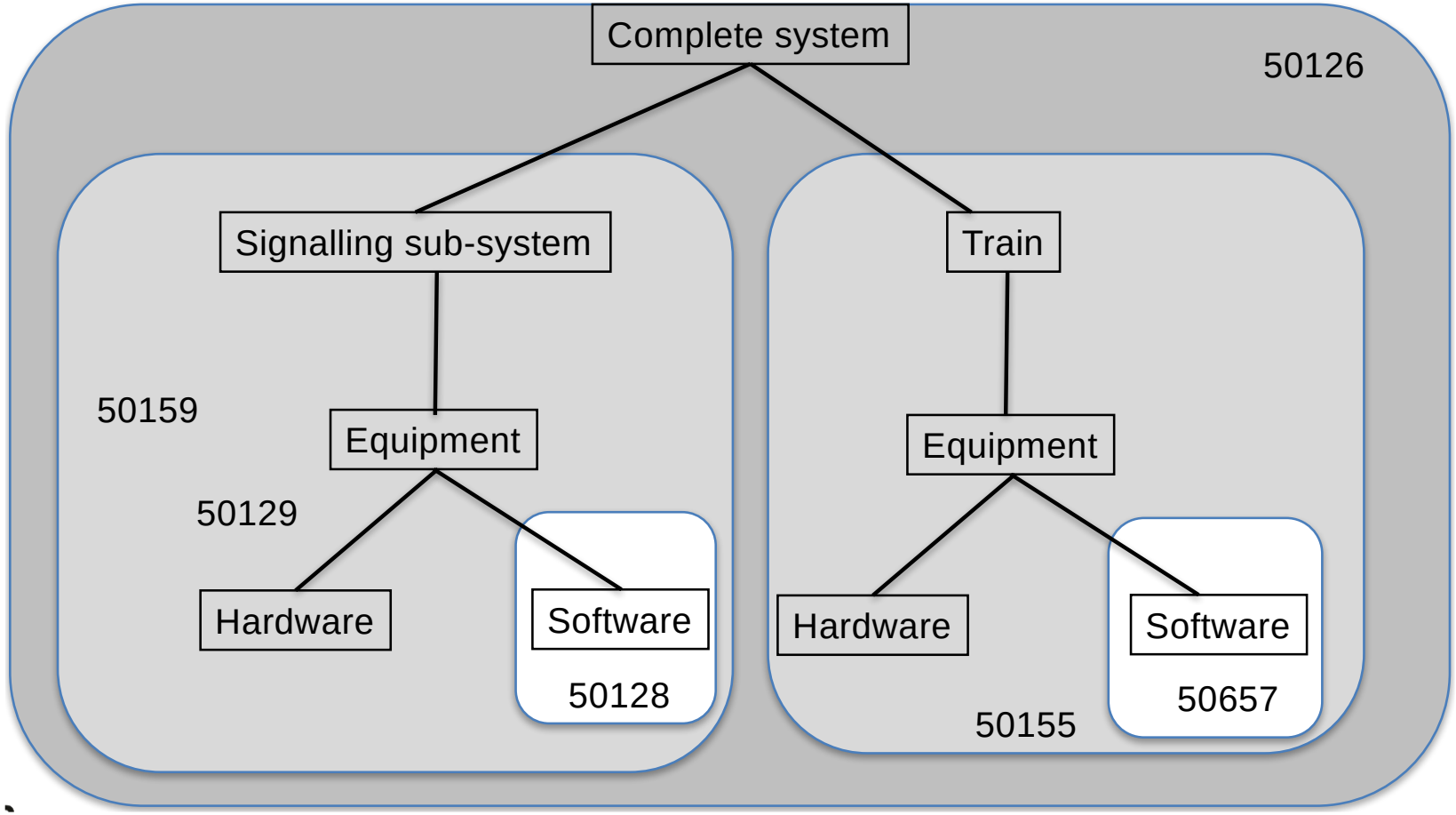


# Railways industry

# CENELEC EN 50128

- Specifies the process and technical requirements for the development of software for programmable electronic systems for use in railway control and protection applications
- not relevant for software that has been identified as having no impact on safety, i.e. software of which failures cannot affect any identified safety functions

# CENELEC


Complete system
50126
Signalling sub-system
Train
50159
Equipment
Equipment
50129
Hardware
Software
Hardware
Software
50128
50155
50657

# Software Safety Integrity Levels (SSIL)

| Software Safety Integrity Level | Description of software safety integrity |
|---|---|
| 4 | Very high |
| 3 | High |
| 2 | Medium |
| 1 | Low |
| 0 | Non-safety-related |

# SSIL and Risk

- CENELEC EN 50128 requires the following hazard consequences to be taken into account when determining SIL:
  - loss of human life or lives;
  - injuries to, or illness, of persons;
  - environmental pollution;
  - loss of, or damage, to property.
- Two Levles: In CENELEC EN 50128 the requirements for SIL 3 are the same as for SIL 4, and the requirements for SIL 1 are the same as for SIL 2.
- In practice there are only three levels:
  - SIL 0
  - SIL 1 / SIL 2 (if you have SIL1 you choose SIL0 or SIL2)
  - SIL 3 / SIL 4

# Software development life cycle

# Classes of Tools

- T2 and T3 tools adoption need to be justified in the process, in terms of the software requirement specification for the required SIL

| Tool Class | Description | Examples |
|---|---|---|
| **T1** | Tool output does not contribute to executable code | Text editor, VCS |
| **T2** | Tool tests / verifies design or executable code; cannot introduce defects into the executable code, but may fail to detect existing defects | Static analysis tool, Code coverage test tool |
| **T3** | Tool output contributes to executable code | Compiler, Linker |

# SRS

**Table A.2 – Software Requirements Specification (7.2)**

| TECHNIQUE/MEASURE | Ref | SIL 0 | SIL 1 | SIL 2 | SIL 3 | SIL 4 |
|---|---|---|---|---|---|---|
| 1. Formal Methods (based on a mathematical approach) | D.28 | - | R | R | HR | HR |
| 2. Modelling | Table A.17 | R | R | R | HR | HR |
| 3. Structured methodology | D.52 | R | R | R | HR | HR |
| 4. Decision Tables | D.13 | R | R | R | HR | HR |

Requirements:

1) The Software Requirements Specification shall include a description of the problem in natural language and any necessary formal or semiformal notation.
2) The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used.

**Table A.17 – Modelling**

| TECHNIQUE/MEASURE | Ref | SIL 0 | SIL 1 | SIL 2 | SIL 3 | SIL 4 |
|---|---|---|---|---|---|---|
| 1. Data Modelling | D.65 | R | R | R | HR | HR |
| 2. Data Flow Diagrams | D.11 | - | R | R | HR | HR |
| 3. Control Flow Diagrams | D.66 | R | R | R | HR | HR |
| 4. Finite State Machines or State Transition Diagrams | D.27 | - | HR | HR | HR | HR |
| 5. Time Petri Nets | D.55 | - | R | R | HR | HR |
| 6. Decision/Truth Tables | D.13 | R | R | R | HR | HR |
| 7. Formal Methods | D.28 | - | R | R | HR | HR |
| 8. Performance Modelling | D.39 | - | R | R | HR | HR |
| 9. Prototyping/Animation | D.43 | - | R | R | R | R |
| 10. Structure Diagrams | D.51 | - | R | R | HR | HR |
| 11. Sequence Diagrams | D.67 | R | HR | HR | HR | HR |

Requirements:

1) A modelling guideline shall be defined and used.
2) At least one of the HR techniques shall be chosen.

# Coding rules

**Table A.12 – Coding Standards**

| TECHNIQUE/MEASURE | Ref | SIL 0 | SIL 1 | SIL 2 | SIL 3 | SIL 4 |
|---|---|---|---|---|---|---|
| 1. Coding Standard | D.15 | HR | HR | HR | M | M |
| 2. Coding Style Guide | D.15 | HR | HR | HR | HR | HR |
| 3. No Dynamic Objects | D.15 | - | R | R | HR | HR |
| 4. No Dynamic Variables | D.15 | - | R | R | HR | HR |
| 5. Limited Use of Pointers | D.15 | - | R | R | R | R |
| 6. Limited Use of Recursion | D.15 | - | R | R | HR | HR |
| 7. No Unconditional Jumps | D.15 | - | HR | HR | HR | HR |
| 8. Limited size and complexity of Functions, Subroutines and Methods | D.38 | HR | HR | HR | HR | HR |
| 9. Entry/Exit Point strategy for Functions, Subroutines and Methods | D.38 | R | HR | HR | HR | HR |
| 9. Limited number of subroutine parameters | D.38 | R | R | R | R | R |
| 10. Limited use of Global Variables | D.38 | HR | HR | HR | M | M |
| Requirement:<br>1) It is accepted that techniques 3, 4 and 5 may be present as part of a validated compiler or translator. | | | | | | |

# Verification and Testing

**Table A.5 – Verification and Testing (6.2 and 7.3)**

| TECHNIQUE/MEASURE | Ref | SIL 0 | SIL 1 | SIL 2 | SIL 3 | SIL 4 |
|---|---|---|---|---|---|---|
| 1. Formal Proof | D.29 | - | R | R | HR | HR |
| 2. Static Analysis | Table A.19 | - | HR | HR | HR | HR |
| 3. Dynamic Analysis and Testing | Table A.13 | - | HR | HR | HR | HR |
| 4. Metrics | D.37 | - | R | R | R | R |
| 5. Traceability | D.58 | R | HR | HR | M | M |
| 6. Software Error Effect Analysis | D.25 | - | R | R | HR | HR |
| 7. Test Coverage for code | Table A.21 | R | HR | HR | HR | HR |
| 8. Functional/ Black-box Testing | Table A.14 | HR | HR | HR | M | M |
| 9. Performance Testing | Table A.18 | - | HR | HR | HR | HR |
| 10. Interface Testing | D.34 | HR | HR | HR | HR | HR |

Requirements:

1) For software Safety Integrity Levels 3 and 4, the approved combination of techniques is 3, 5, 7, 8 and one from 1, 2 or 6.

2) For Software Safety Integrity Level 1 and 2, the approved combinations of techniques is 5 together with one from 2, 3 or 8.

NOTE 1 Techniques/measures 1, 2, 4, 5, 6 and 7 are for verification activities.

NOTE 2 Techniques/measures 3, 8, 9 and 10 are for testing activities.

# Pre-existing software

7.3.4.7 The use of pre-existing software shall be subject to the following restrictions.

a) For all software safety integrity levels the following information shall clearly be identified and documented:
   - the requirements that the pre-existing software is intended to fulfil;
   - the assumptions about the environment of the pre-existing software;
   - interfaces with other parts of the software.

b) For all software safety integrity levels the pre-existing software shall be included in the validation process of the whole software.

c) For software safety integrity levels SIL 3 or SIL 4, the following precautions shall be taken:
   - an analysis of possible failures of the pre-existing software and their consequences on the whole software shall be carried out;
   - a strategy shall be defined to detect failures of the pre-existing software and to protect the system from these failures;
   - the verification and validation process shall ensure
     1) that the pre-existing software fulfils the allocated requirements,
     2) that failures of the pre-existing software are detected and the system where the pre-existing software is integrated into is protected from these failures,
     3) that the assumptions about the environment of the pre-existing software are fulfilled.

d) The pre-existing software shall be accompanied by a sufficiently precise (e.g. limited to the used functions) and complete description (i.e. functions, constraints and evidence). The description shall include hardware and/or software constraints of which the integrator must be aware and take into consideration during application. In particular it forms the vehicle for informing the integrator of what the software was designed for, its properties, behaviour and characteristics.

# Industrial Internet of Things

# Industrial Internet of Things

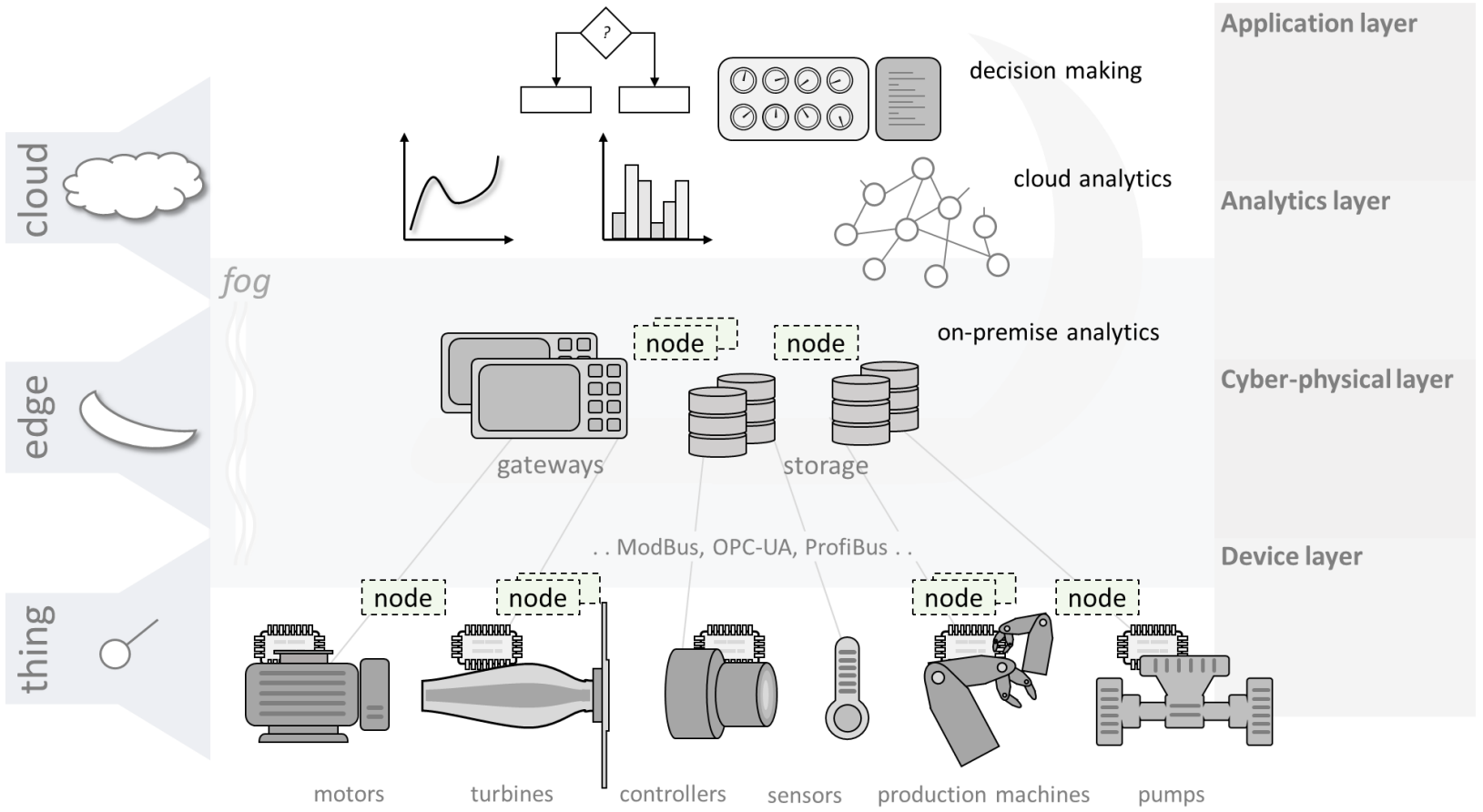
- The extension of the IoT to the digital transformation of industries, also known as the fourth industrial revolution (Industry 4.0)
- In the IIoT the “things” are (smart) sensors, actuators, controllers, instruments and other devices interconnected with plant control systems and supervision systems for manufacturing
- It is an extension of distributed control systems with the innovative technologies brought by cloud computing
- Involved industries:
  - Energy, Manufacturing, Healthcare, Transportation, Smart Cities, Retail, …

# Industrial Internet of Things

- Convergence of several technologies
  - **Cyber-physical systems** (CPS): the merging of physic processes with computing and communication (smart grid, robotics, automatic pilots, …)
  - **Cloud computing**: consumption of services from the Internet
  - **Edge computing**: paradigm to bring computing and storage resources closer to the location it is needed, at the “edge” of the network
  - **Big data analytics** and **Artificial Intelligence**: for the automated analysis of the massive amounts of data produced by the CPS
  - **Real-time computing**: for real-time control and supervision
  - **Time-sensitive (wireless) networking**: for deterministic and/or low latency (often over-the-air) communication

# IIoT architecture

- A multi-layered mixed-criticality system



# IIoT Standards …?

- Still under development…
- De facto utilized solutions:
  - REST microservices
  - MQTT publish/subscribe middleware
  - XMPP for communication and presence management
  - OPC (OLE for Process Control): bridge between Windows and process control
    - *What about sw platforms and hypervisors??*
- Industrial groups
  - Industry IoT Consortium: an OMG group (www.iiconsortium.org)
  - Open Industry 4.0 Alliance (www.openindustry4.com)

# Wireless communication protocols eligible for IIoT (non exhaustive list…)

- Bluetooth and Bluetooth LE
- Zigbee
- WirelessHART
- LoRa
- 5G cellular
- … many others … and proprietary solutions

# Bluetooth


- IEEE 802.15.1 standard for Personal Area Networks (PAN)
- 2.4 GHz ISM band with spread spectrum
- Bluetooth piconets are PANs with maximum 8 devices, 1 master, 7 slaves, with time-division multiplexing of the channel guided by the master

# ZigBee


- IEEE 802.15.4 standard for low power PANs
- Ultra low power chips usually integrated in microcontrollers
- Allows both CSMA/CA or beacon-oriented
  - In beacon-oriented networks, the chip is woken-up only when a beacon is sent by a master → ultra low power
  - Beacons can be sent on a fixed-timing schedule, if needed

# WirelessHART

- IEEE 802.15.4, same family of ZigBee, but specialized for industrial wireless sensor networks
- Wireless extension of the HART protocol used in industrial automation field networks
- TDMA synchronized, latency-controlled communications
- Additional features for reliability and security