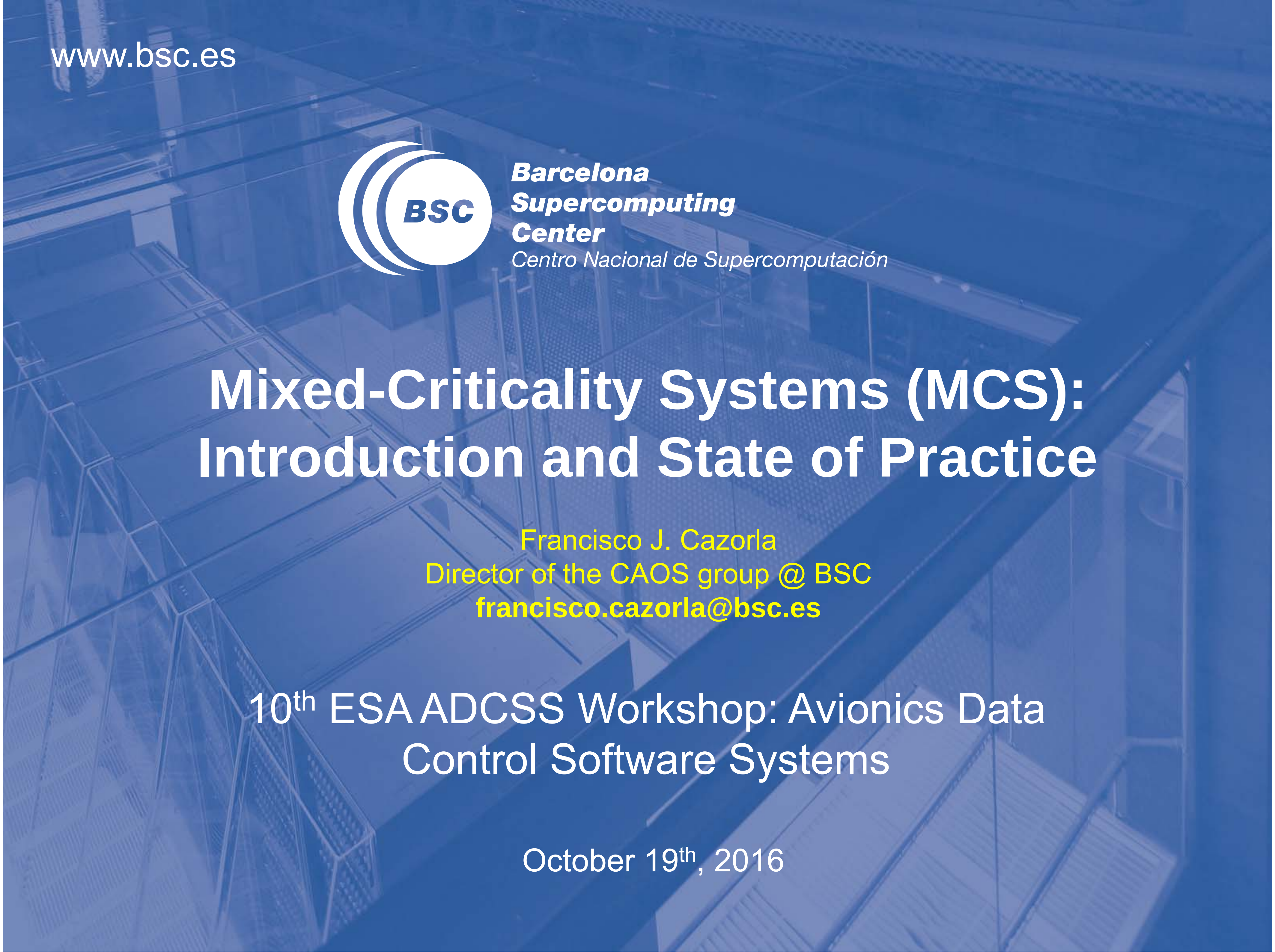


www.bsc.es

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Mixed-Criticality Systems (MCS): Introduction and State of Practice

Francisco J. Cazorla
Director of the CAOS group @ BSC
francisco.cazorla@bsc.es

10th ESA ADCSS Workshop: Avionics Data Control Software Systems

October 19th, 2016

Motivation

- Mixed-Criticality Systems
  - Motivation (need for) and definitions

- Criticality and Safety Standards
  - Introduction
  - Implications on HW/SW design

- MCS
  - Concepts (Pillars) for their realization of MCS in other domains
  - Automotive and Avionics

- (Some) Open challenges

www.bsc.es

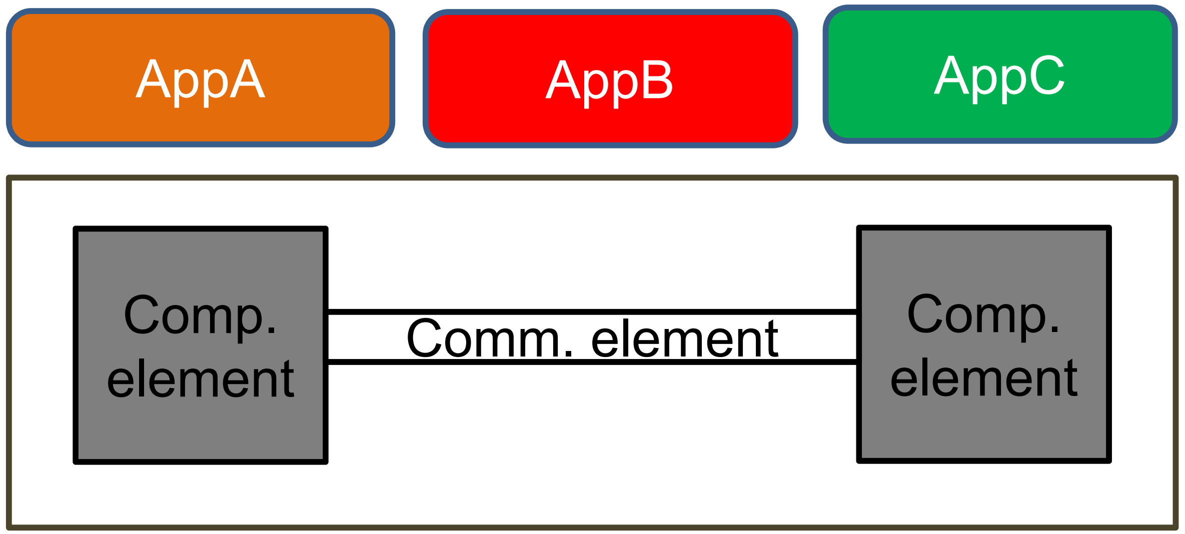
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

MCS: Motivation, Definitions and Need for

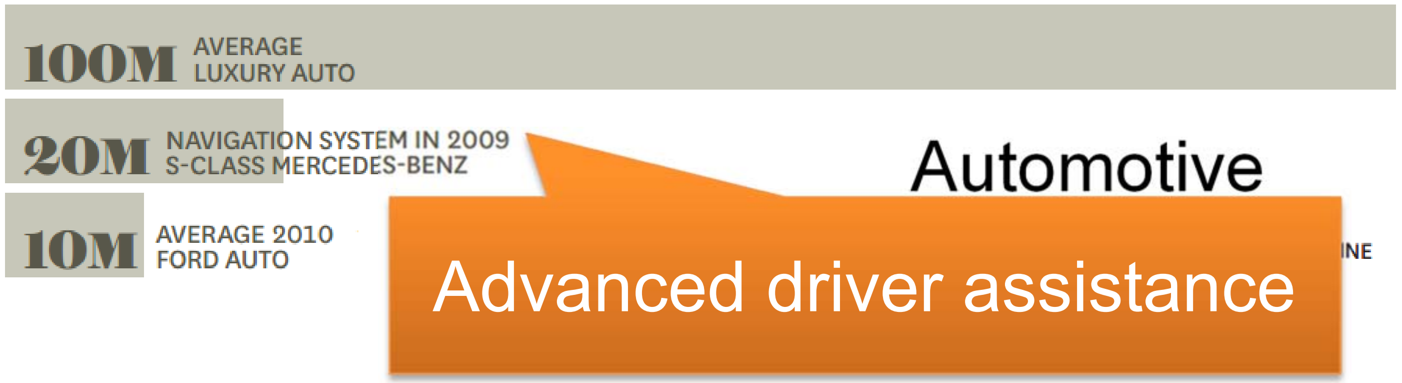
Francisco J. Cazorla

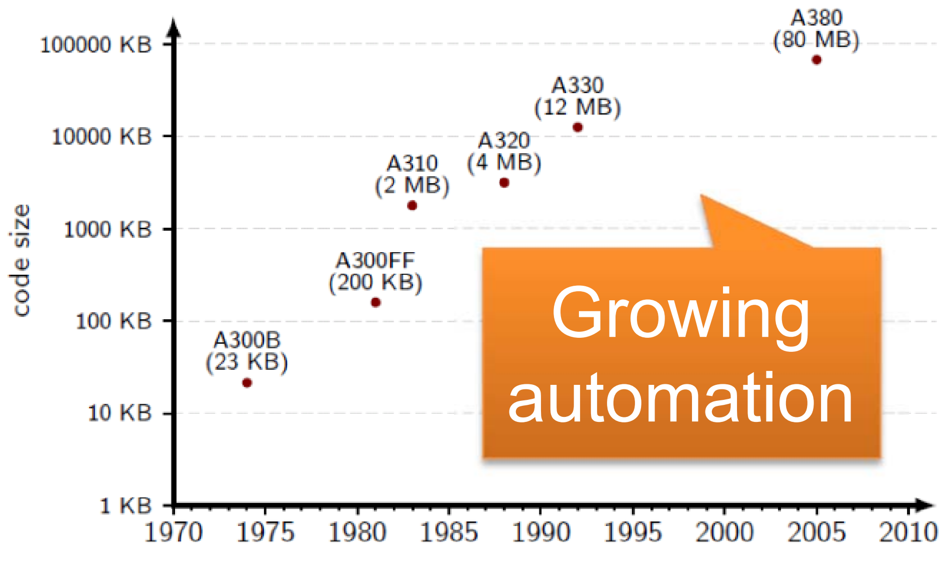
Mixed Criticality Systems (MCS)

- Embedded Computing System that encompasses the execution of several application functions that
  1. are subject to different criticalitites and
  2. share computation and (or) communication means

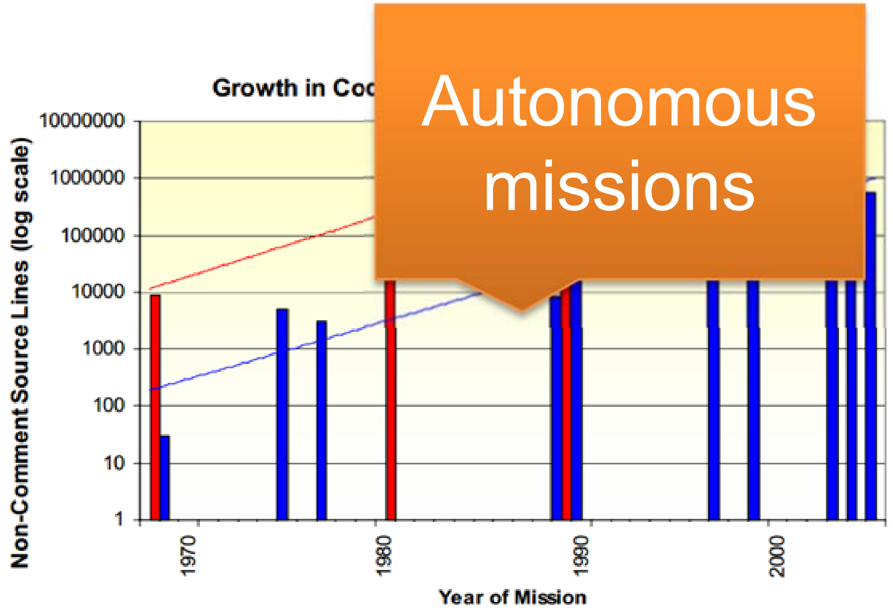


Trends is SW size (and complexity)





Avionics *source: Airbus



Space *source: Nasa

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation and Threat

- The ability to run on the same hardware applications with different criticalities helps reducing Space, Weight and Power (SWaP) costs
  - This aims at <u>improving the performance/cost</u> ratio of the system

- However, it must be factored in the potential increase in V&V and certification costs due to MCS execution

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Criticality and Safety Standards

Francisco J. Cazorla

Criticality and Safety Standards

- Criticality mostly applies in the context of functional safety
  - It has been applied to other metrics
- Several safety-related standards in different domains
- IEC61508 (generic electrical and/or electronic and/or programmable electronic (E/E/PE))
  - ISO26262 (automotive domain)
  - EN-50126, EN-50128, EN-50129 (rail domain)
- ECSS-Q-ST-80C in Space
- DO-178C (SW aeronautics domain) - DO254 (HW aeronautics domain)

Development Cycle and Assurance Process

- Development life cycle of a critical system
  - Involve defining the list of requirements on the behaviour and characteristics that are expected from them
  - Functional requirements (what the system is expected to do),
  - Non-functional properties (safety, security and performance, including timing and energy constraints)

- In order to "**ensure**" the safety properties a system safety assessment process is followed

- The process varies across domains: e.g. ISO26262 for automotive

ISO26262

- Motivation (partial):
  - electronics, both hardware and software, can fail
  - evidence must be provided on the fact that risks have been minimized

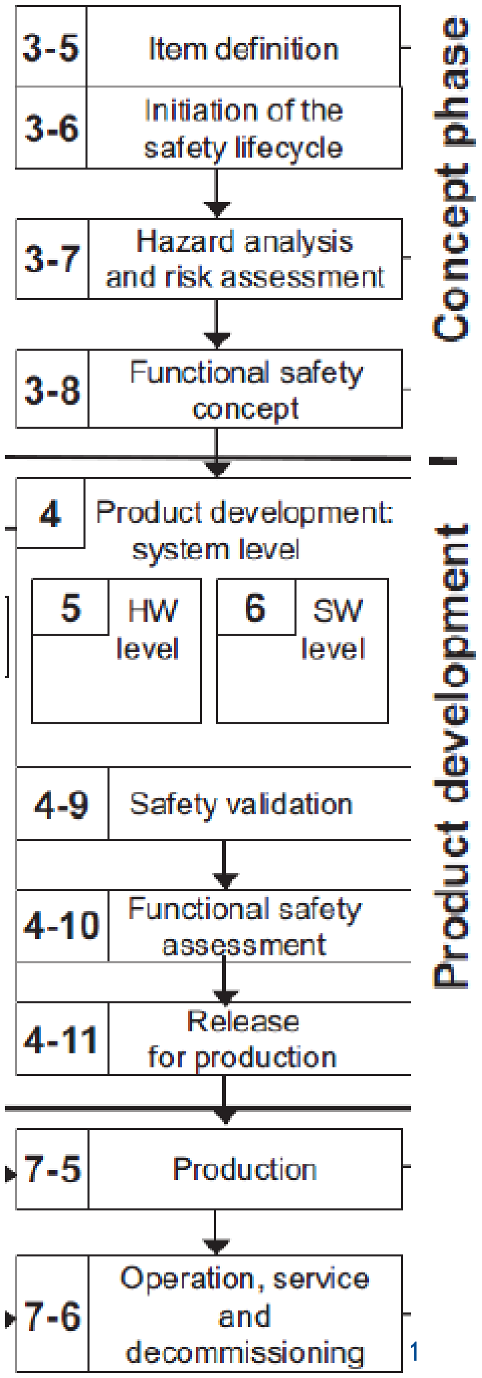
- ISO26262 aims at showing absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems

- It is necessary to assess the residual risk of having faults that may cause the violation of safety goals
  - Biased by hardware
  - 'The software is a different story'

General System Safety Assessment Process

1. Item definition
   - Define and describe the item to be dev. and cert.
   - Definition → functional requirements, dependences, environmental conditions

2. Hazard Analysis and Risk Assessment
   - Define Op. situations + Op. modes in which item malfunction results in hazardous events
   - Determine safety goals for the item such that *unreasonable risk* is avoided
     - The risk of a hazardous event is sufficiently low if the safety goal is achieved
   - Determine safety requirements associated with a hazardous event


3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
Concept phase
4 Product development: system level
5 HW level
6 SW level
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
Product development
7-5 Production
7-6 Operation, service and decommissioning

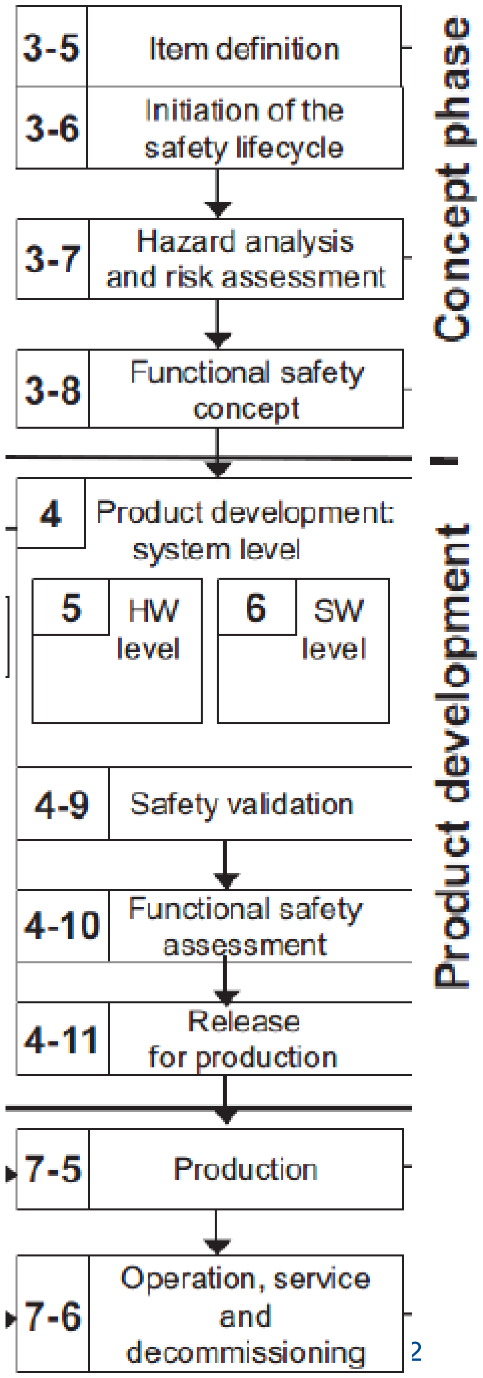
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Francisco J. Cazorla

General System Safety Assessment Process

3. Formulation of safety goals
   - Defined for each hazard and hazardous event
     - Systematic evaluation of hazards that may be caused by the item
   - ASIL associated to safety goal (severity, probability and controllability)

4. Create a Safety Concept
   - Derive functional safety requirements from safety goals
   - Safety requirements are allocated to HW and SW components according to their architectural design
   - Define the mechanisms required to
     - reduce the risk of the faults or
     - to mitigate their effects and
     - avoid the propagation of failures


3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
Concept phase
4 Product development: system level
5 HW level
6 SW level
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
Product development
7-5 Production
7-6 Operation, service and decommissioning

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Criticality Assignment

- Standard (domain) specific
- Result of an analysis in which it is assessed
  - the severity of a malfunction,
  - the probability of exposing a malfunction
  - The controllability of the situation in case of a malfunction occurring

Controllability	Exposure	Severity			
		S0	S1	S2	S3
C1	E1	QM	QM	QM	QM
	E2	QM	QM	QM	QM
	E3	QM	QM	QM	A
	E4	QM	QM	A	B
C2	E1	QM	QM	QM	QM
	E2	QM	QM	QM	A
	E3	QM	QM	A	B
	E4	QM	A	B	C
C3	E1	QM	QM	QM	A
	E2	QM	QM	A	B
	E3	QM	A	B	C
	E4	QM	B	C	D

- ASIL:
  - Quality mgmt (no safety measure required),
  - ASILA,
  - …,
  - ASILD (highest)
  - Factors
    - Sj > Si → more severe
    - Cj > Ci → more difficult to control
    - Ej > Ei → more frequent
- Bear in mind:
  - Severity is not <u>the</u> metric
  - Dropping low-critical tasks

www.bsc.es

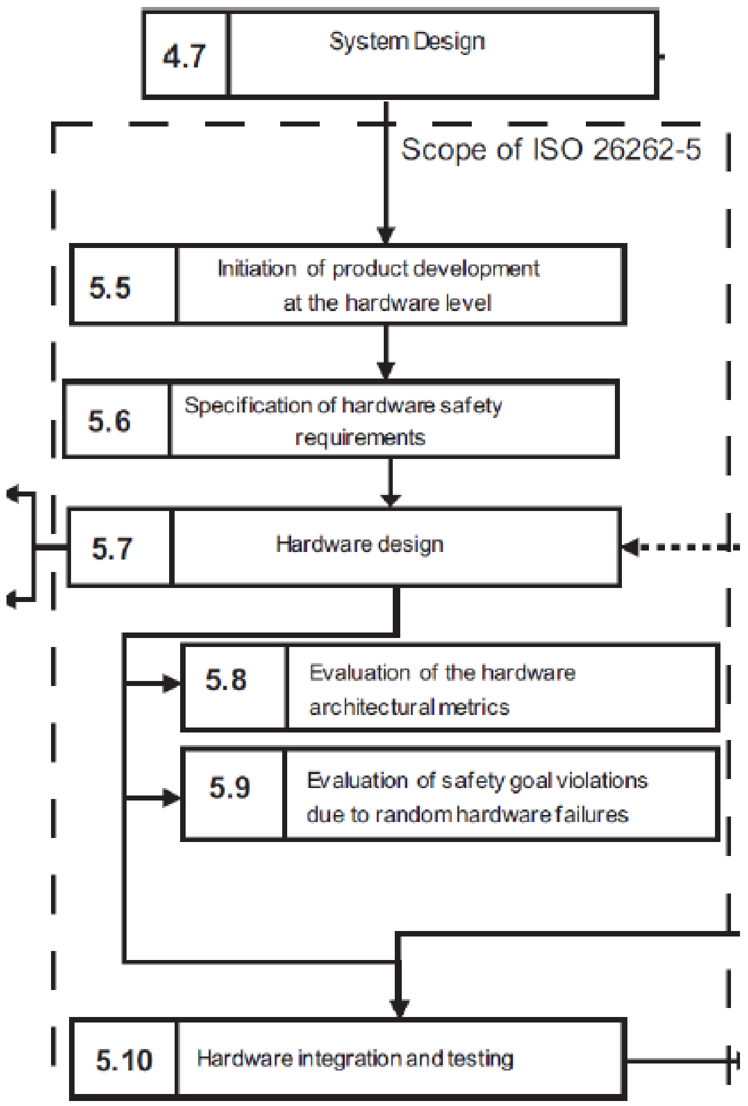
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Impact on Hardware and Software Components

Francisco J. Cazorla

ISO26262: Hardware /1

- Hardware development
  - Use safety concept and system design to derive <u>hardware safety requirements</u>
  - Requirements and attributes of HW safety mechanisms
- Hardware evaluation:
  - After HW design → design should be analyzed regarding whether it meets the requirements from its ASIL
  - Obtain failure modes, failure rates and diagnostic coverage
  - These metrics are used for
    - evaluation of HW architectural metrics and
    - evaluation of safety goals


4.7 System Design
Scope of ISO 26262-5
5.5 Initiation of 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 testing

ISO26262: Hardware /2

- Recall criticality applies at system function level (safety goal in ISO26262)
- So the question is: ***What HW faults have the potential to contribute to the violation of the safety goal?***
- Good news:
  - Several reliability models exist to quantify the compliance of hardware components to reliability requirements
  - For instance in ISO26262 covers
    - Systematic: produced by human error during system dev. and operation
    - Random: due to physical causes (wear-our, thermal stress, …)

ISO26262: Hardware /3

- Systematic faults: failure rate
  - Based on a number of failure modes, each defining the failure rate, λ
  - Types:
    - Safely ignorable fault (occurrence will not significantly increase the probability of violation of a safety go),
    - Single point fault, in an element not covered by the Safety mechanism that leads directly to safety goal violation
    - Residual fault in an element covered by a Safety mechanism not covered by element's safety mechanism
    - Multiple point fault (perceived, detected, latent)

- Systematic faults: Diagnosis
  - The diagnostic coverage of each safety mechanism must be evaluated to find the faults that can violate the safety goal

ISO26262: Hardware /4

- ***What HW faults have the potential to contribute to the violation of the safety goal?***
  - Safely ignorable and multiple point faults that are detected or perceived, are typically regarded as irrelevant.
    - “visible” before they can produce any harm, or they simply cannot produce any harm
  - Instead, single-point faults; residual faults and latent multiple point faults are critical since they may lead to the violation of the safety goal with a single fault

- The HW is assessed w.r.t its ability to *remove* failures
  - A **fault metric** is defined for those HW elements whose failures have the potential to contribute to the violation of a safety goal

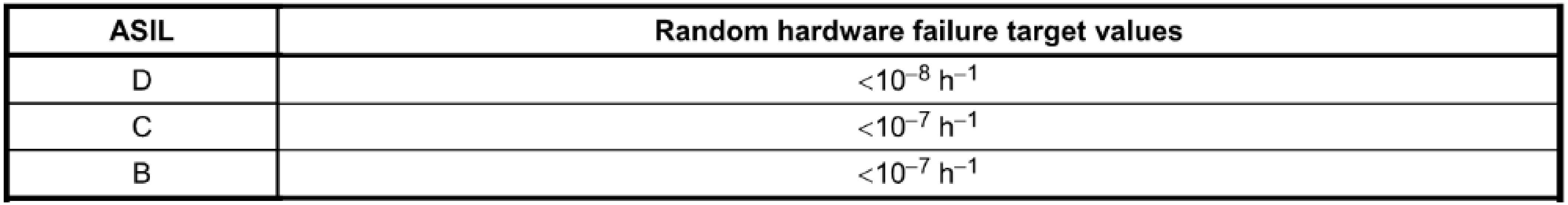
ISO26262: Hardware /5

- Two metrics are calculated to:
  - single-point fault metric (SPFM) and
  - latent fault metric (LFM).
- SPFM:
  - HW item's robustness to single-point and residual faults either by <u>design</u> or by <u>safety mechanism coverage</u>
- LFM:
  - HW item's robustness for latent faults either by design, by safety mechanism coverage or by the driver recognizing that the fault exists before the safety goal is violated
- Targets for the single-point fault metric are as follows:

	ASILB	ASILC	ASILD
SPFM	≥90%	≥97%	≥99%
LFM	≥60%	≥80%	≥90%

ISO26262: Hardware: /6

- Failure rate classes 1 to 5 are defined
  - Different target failure rates.
  - The degree of diagnostic coverage and the ASIL level determine the failure rate class for the hardware part.
  - An ASIL D safety goal requires proving the residual failure rate below 10^{-7} (failure rate class 4) if the diagnostic coverage is above 99.9%.
  - Lower failure rates are required if the diagnostic coverage is lower.
  - And higher failure rates are allowed if the ASIL is lower (e.g., C, B).
- Probabilistic metric for random hardware failures (PMHF)



ASIL	Random hardware failure target values
D	$<10^{-8}\ h^{-1}$
C	$<10^{-7}\ h^{-1}$
B	$<10^{-7}\ h^{-1}$

ISO26262: Software : Similarities to Hardware

- Similar to HW, for SW ISO26262 provides principles for
  - software architectural design,
  - mechanisms for error detection at software architectural level
  - mechanisms for error handling at software architectural level

- Each software component must be categorized and verification techniques applied to it accordingly

ISO26262: Software : Differences

- All software faults are systematic

- Safety assessment of SW is performed via a **qualitative process**
  - Confidence figure cannot be put on the reliability of software elements
  - The development of a SW element to a certain ASIL does not carries the assignment of a failure rate for it
  - Argue about the acceptability of software based on the suitability of the development processes followed as recommended by the standard

ISO26262: Software : Differences

- At the HW, the coverage of the fault-detection (and correction mechanisms) are evaluated as well as the residual risk of undetected potentially-impacting faults

- Standards are pretty 'poor' about admitting the reality of …
  - software having bugs
    - If you follow the development standards that → SW does not fail
  - software incurring timing violations
    - Timing bound: Maximum Observed Execution Time * 1.X
    - X such that timing violations do not occur (never)

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Requirements of MCS

Francisco J. Cazorla

Recap

- MCS definition:
  - Computing System in which applications subject to different (**functional safety**) criticalities share computation and (or) communication means

- MCS goals:
  - Preserving criticality (safety) characteristics of each individual application
  - Increase a high benefit/cost ratio
    - Carefully assess the cost of safety assessment in MCS

    and
    - put in place measures to reduce those costs

Goals and Pillars

“Demonstration of sufficient independence”
and
“mechanisms to reach this goal”

- Examples
  - ISO26262: “Freedom from interference”
  - IEC-61508-3: “Non-interference between software elements”

- Pillars to reach these goals:
  - Partitioning
  - Monitors

Goals and Pillars

“Failing to demonstrate this $\rightarrow$ the whole MCS is designed according to the highest level of criticality involved”

- Cots heavily increase
- Benefit/cost ration of MCS decreases

IEC61508: Partitioning

- Partition:
  - Allows the isolation of SW components
  - Reduce V&V costs
  - Allows containment of faults
- How:
  - Physical segregation: 1 SW element → 1 HW element (high cost)
  - Virtual segregation: Create provisions on the HW to allow SW components to share the platform
- Applications should not interfere with each other behaviour
  - Spatial isolation prevents inter-application data/code (functional) corruption
  - Temporal isolation prevents blocking use of resources or using CPU longer than expected
  - Evidence of the absence of interference required by analysing the system

IEC61508: Partitioning : (Some) Recommendations

- Spatial independence:
  - HW memory protection
  - Virtual memory space
  - Rigorous design + source (and object) code analysis

- Temporal independence:
  - Scheduling Policy:
    - deterministic (cyclic scheduling)
    - strict priority based scheduling (preventing priority inversion)
  - Time budgets
    - kills application trying to violate its budget
    - No process can clog CPU
  - Resource sharing protocol → prevent unwanted locking of resources

DO178C: Partitioning

1. Code/data of a SWc cannot be affected by another SWc
2. A partitioned SWc only allowed to use CPU during its budget
3. Each partition should be able to contain faults (not to propagate them to other partitions)
4. Partition software has the same or higher DAL than highest DAL to SWc in any of the partitions
5. Safety assessment to be done in HW so that a failure cause failure on SW partitions hence affecting system safety

IEC51508: Monitors

- Concept:
  - Monitors application behaivour (protection against faults)
  - Fault detected → trigger an event to active corrective actions
  - Prevent the propagation of failures (provision for)
  - External monitor running on independent hardware
- Benefits:
  - Demonstrate separation of concerns
  - Reduce criticality of monitored application
- Remarks:
  - The monitor is as critical as the application it monitors

DO178C: Monitors

- Inherits highest DAL of monitored functions
- Will detect the intended faults under all conditions (monitor cover all cases)
- Independence between the monitor and the monitored function required

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Challenges

Francisco J. Cazorla

MCS: Opportunities and Threats

- The ability to run on the same hardware applications with different functional safety criticalities helps reducing Space, Weight and Power (SWaP) costs

- The safety properties of each application are to be maintained

- However, it must be factored in the potential increase in V&V and certification costs due to MCS execution

- Goal: <u>improving the performance/cost </u>ratio of the system

Temporal Isolation: Hardware support

- Minimum computation power a task will enjoy in a multicore?
  - Hardware support proposed to derive bounds
  - e.g. carefully shared (among cores) requests queues
    - Prevent one task can clog the queue
    - A task should have the same available entries under all core counts

- ESA activities:
  - **Multicore OS benchmark**
  - NPI: Architectural solutions for the timing predictability of the Next Generation Multi-Processor (NGMP)
  - Barcelona Supercomputing Center – Cobham Gaisler

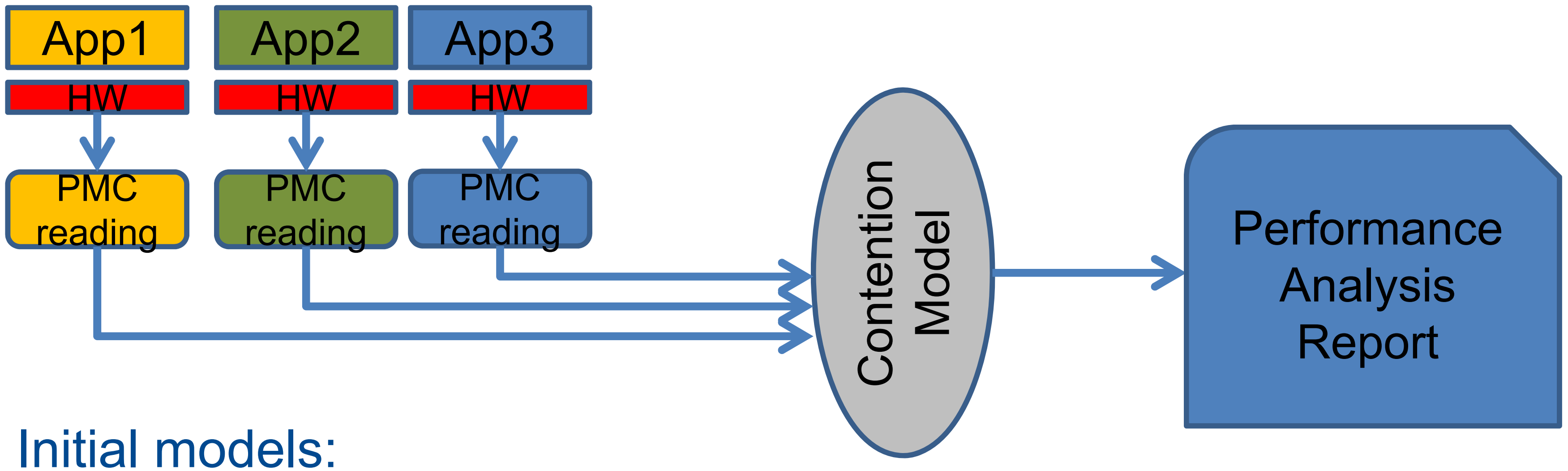
- Bear in mind resource reservation → **wasted CPU capacity**
  - **More research needed to reduce resource reservation needs**

Temporal Isolation: Software support

- Hardware support costly and not available in all COTS
- SW solutions required to bound contention interference
- Goal: "Tasks assigned time budgets and tasks preventing from using the CPU outside their budgets"
  - Works in single-core
  - Not enough in multicores
- With multicores CPU provides variable computing power
  - Corunner tasks determine the 'amount of' computation power one can use'

Temporal Isolation: Software support

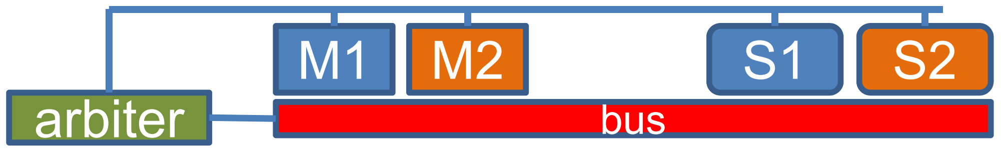
- Idea to deal with multicore contention
  - Characterize tasks in isolation via their PMCs and then derive a model of the slowdown the can suffer if run together



  - Initial models:
    - ESA activities (**Multi-core Architectures – Cache Structure optimization for better RT performance**) and
    - EU PROXIMA project

Monitoring: Prevent a task from blocking a shared resource

- Example of shared resource: AMBA AHB bus



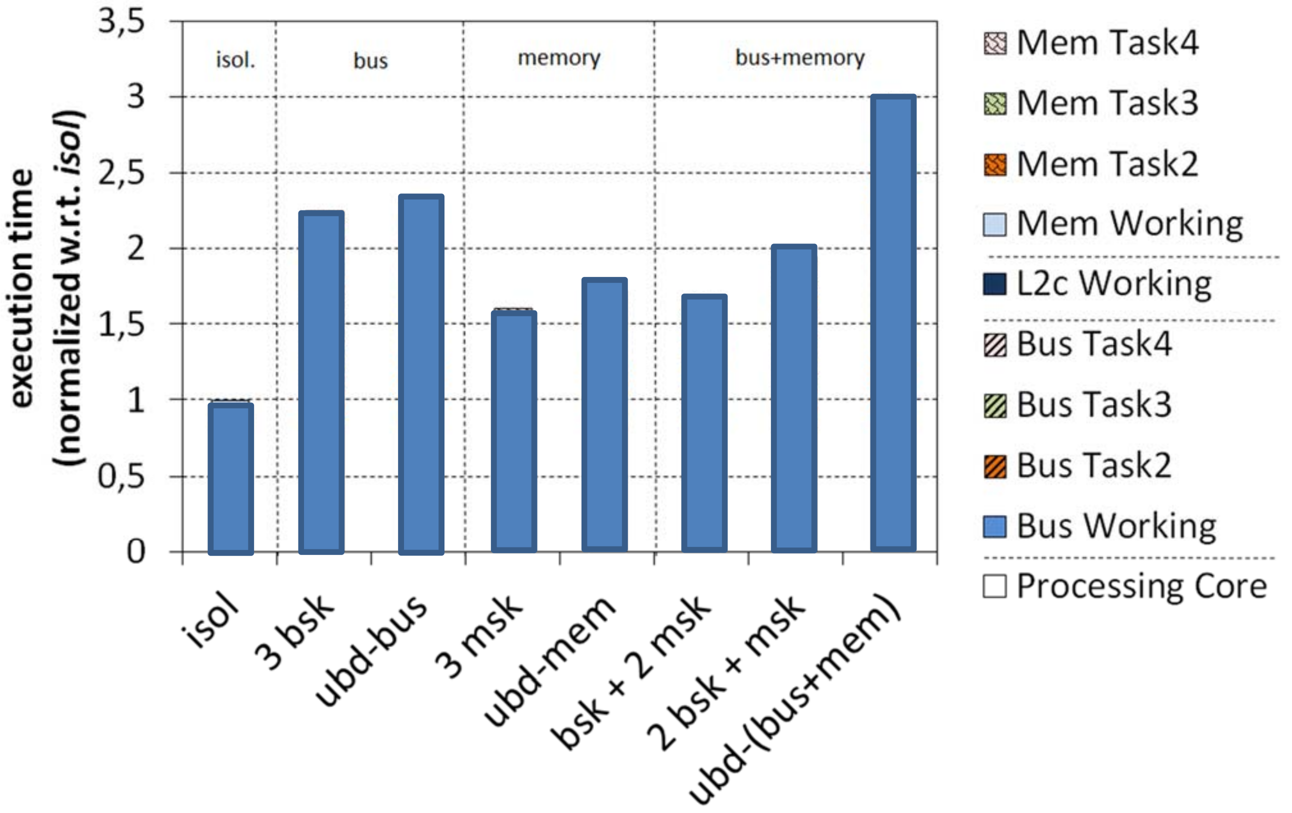
  – A HW AMBA compatible (master) can lock the bus infinite time and
  – the arbiter has not mean to relinquish the grant from the component
  – A <u>single request </u>is enough → <u>controlling request count is not enough</u>

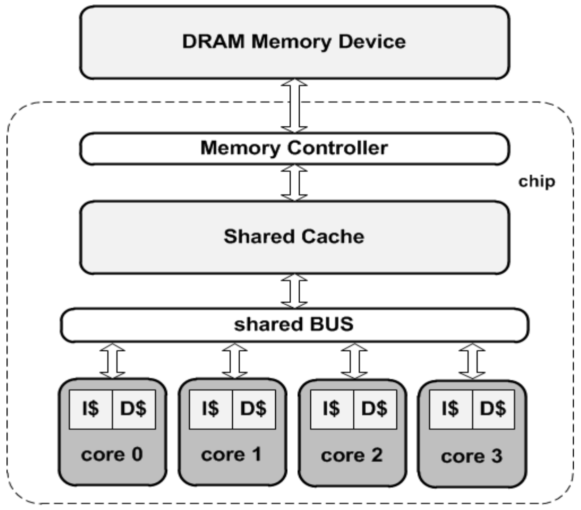
- Means need to be put in place for <u>controlling bus usage time</u>

- Small extension to performance monitoring counter support
  - ESA activities (**Multi-core Architectures – Cache Structure optimization for better RT performance**): BSC and Cobham Gaisler

Much more advanced monitors

- PMC support to provide further evidence on timing bounds
- Cycle contention stack

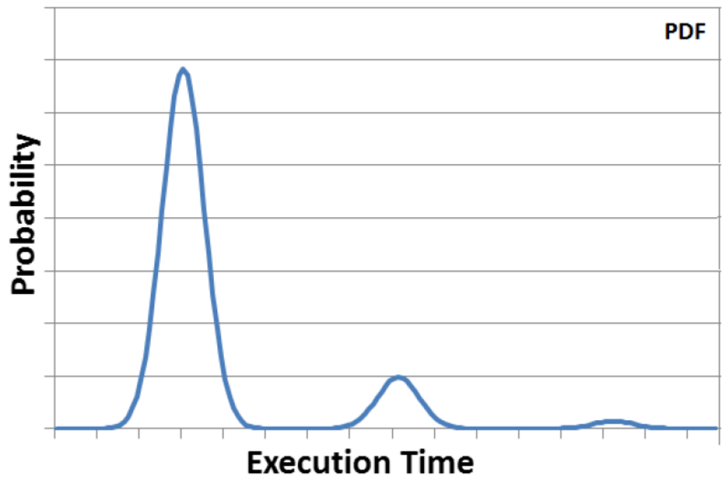


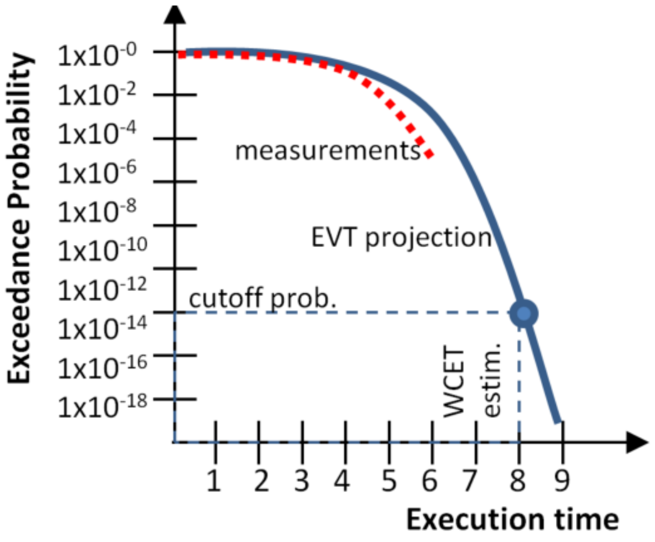


- ESA activities (**Multi-core Architectures – Cache Structure optimization for better RT performance**): BSC and Cobham Gaisler

Probabilistic WCET for software

- Idea: exceeding a time budget can be seen as a software fault if it may cause system-level consequences
  - Single (pessimistic) WCET estimate that factors all worst-events,
    - If something can happen, assume it happens (+20x slowdown observed)
  - Use a probabilistic WCET curve that factors in those events that can happen with a given probability



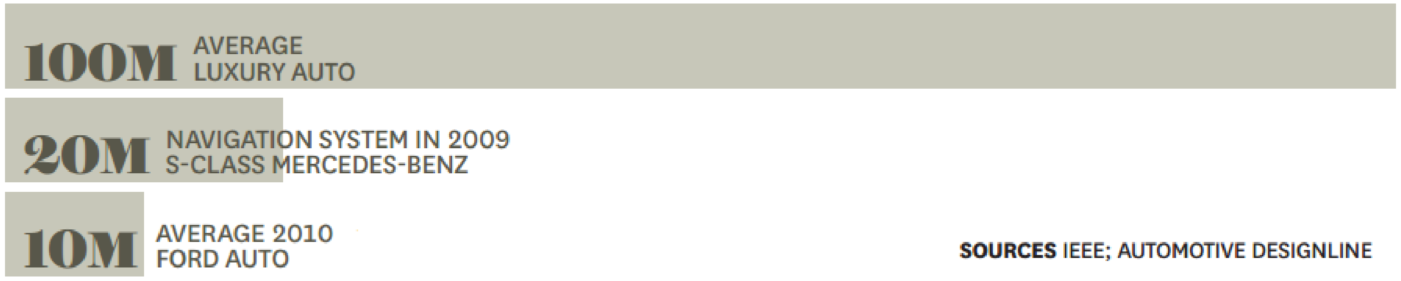


It is not uncommon that engineers apply a 50% margin → Margin based on more solid analysis → Eases providing evidence of correctness

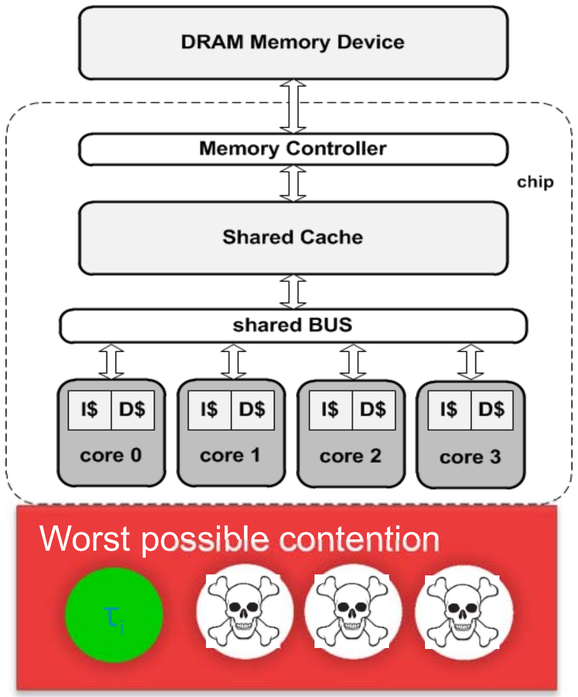
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Francisco J. Cazorla

Probabilistic WCET for software



- Incremental SW dev. and integration
  - SW from different SW Providers (SP)
  - In Early-Design Phases (**EDP**) each SP is provided a time budget
    - Task execution time depends on its corunners
    - Assuming the worst-contention → upto 20x slowdown observed
- Idea:
  - Quantify contenders usage of resources
  - Derive WCET for the task under analysis under that assumption
  - Deploy quota enforcing mechanisms
- ESA project: **Emulator of Future NGMP Multicore** (GMV, Rapita, BSC)

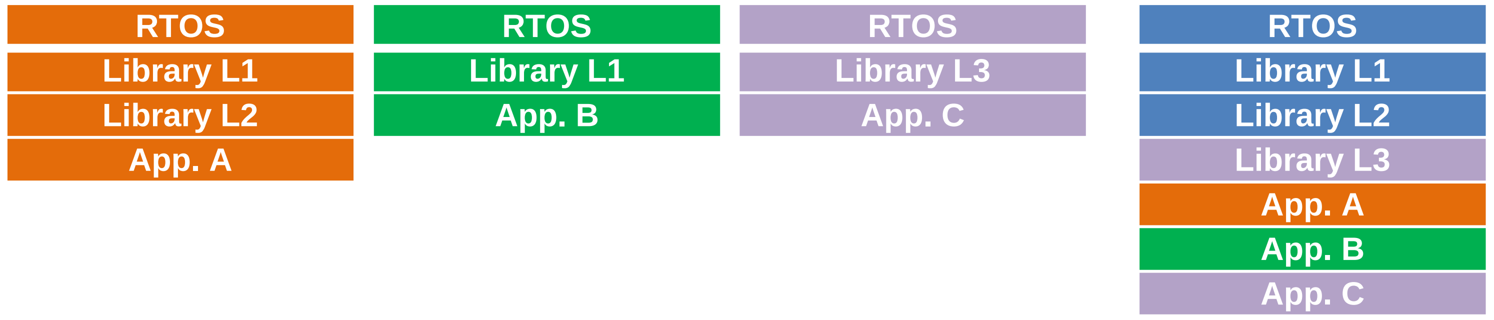


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Francisco J. Cazorla

Probabilistic WCET for software

- Complex SW → developed in increments
  - In each increment more modules are brought together
  - Memory mapping and <u>cache</u> alignment changes across integrations



  - WCET analysis can be performed when cache alignments are fixed
    - Push WCET towards late design phase (LDP)
    - Costly LDP changes
  - Obtain reliable early estimates of tasks <u>in isolation</u> execution time factoring in the impact of different cache alignments
    - The same applies to multicores

Probabilistic WCET for software

- ESA activities
  - PROARTIS for Space
  - Analysis of the suitability of implementing the eviction frequency limitation technique in the NGMP
- PROXIMA FP7 Project

Criticality

- It mostly applies in the context of functional safety
- Current systems have many other non-functional metrics
  - Dependability (e.g. availability, integrity)
  - Timing
  - Security
  - …
- Standards do not (clearly) cover criticality when applied to other metrics
  - As an example we have security
    - How to add security into 'residual risk‘?
    - How to quantify security?

Conclusion

- MCS are deployed in several domain already
  – Tools are in place for Space (e.g. XtratuM)
- The challenge is to deal with new (COTS) hardware features while increasing certification costs and increase overall system benefit/cost ratio
  – “Monitors are welcome”, but cannot be abused!
- Changes in hardware and in software required
- Changes in the safety standards too
  – Quantify software reliability
  – New metrics

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Mixed-Criticality Systems (MCS): Introduction and State of Practice

Francisco J. Cazorla
Director of the CAOS group @ BSC
francisco.cazorla@bsc.es

10th ESA ADCSS Workshop: Avionics Data Control Software Systems

October 19th, 2016