Published at IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 41(4): 1116-1129 (2022)

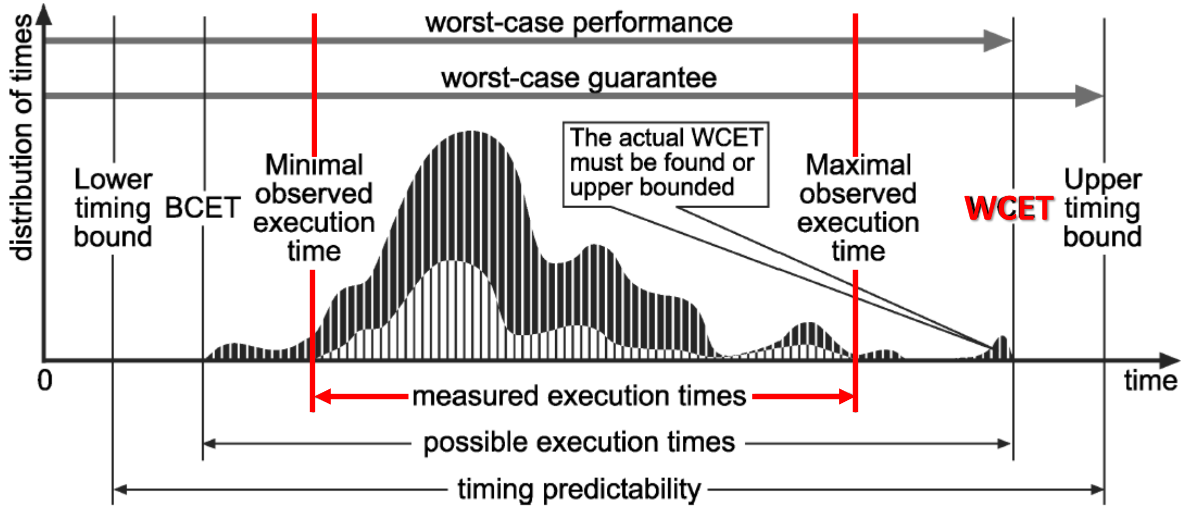
Bridging the Pragmatic Gaps for Mixed-Criticality Systems in the Automotive Industry

Zhe Jiang, Shuai Zhao, Ran Wei, Dawei Yang, Richard Paterson, **Nan Guan**, Yan Zhuang, Neil Audsley

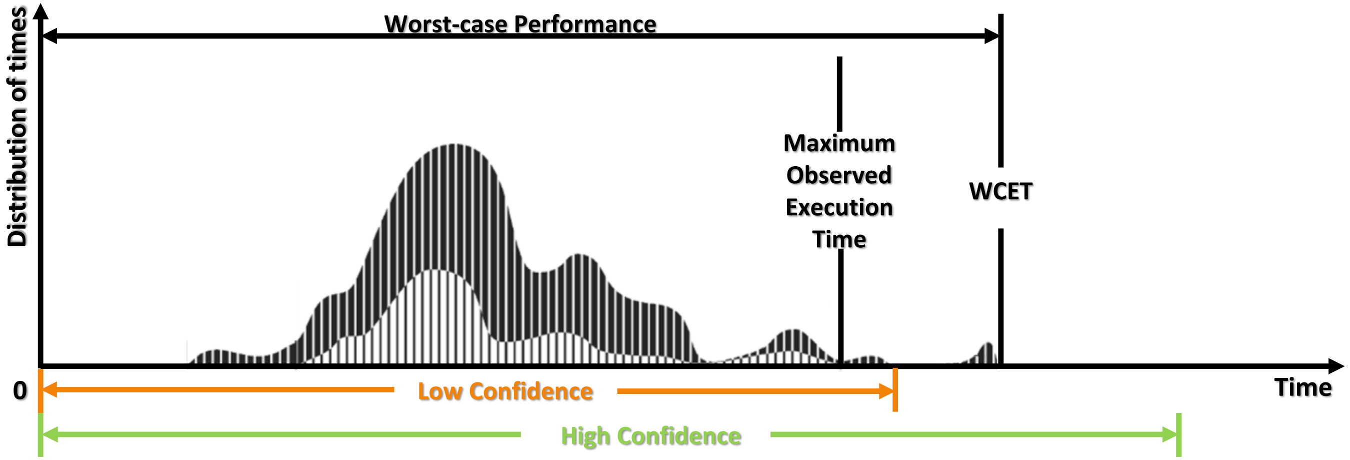
ARM Ltd, UK
University of York
Dalian University of Technology
City University of Hong Kong

Academia Research on MCS

- To meet the real-time requirement, acquiring the Worst-Case Execution Time (WCET) of each task is the first step.
  - However, it is unlikely to achieve the WCET of a tasks via measurement.



Academia Research on MCS

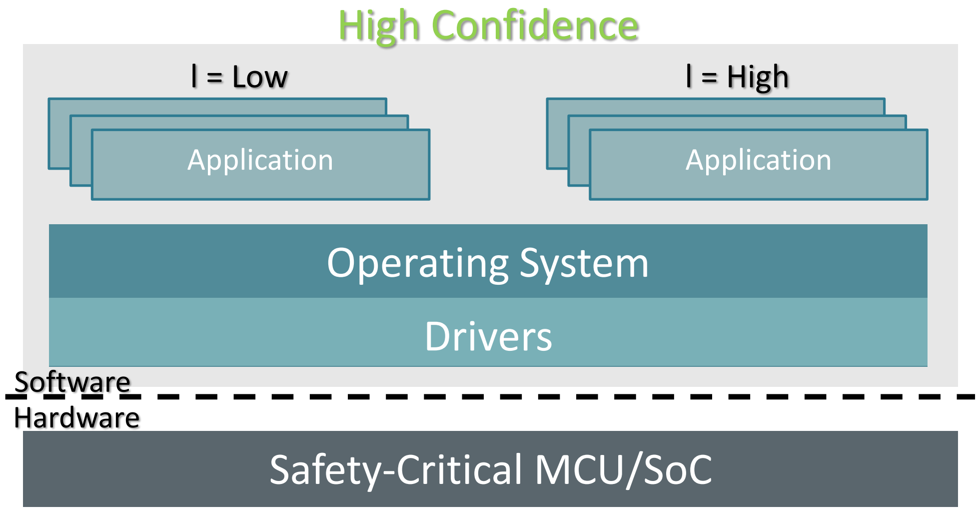


- In MCS, WECT is normally estimated with different levels of confidence*:
- Low confidence: optimistic, saving system resource, but risky.
- High confidence:, pessimistic, safe, but wasting system resource.
- **How to allocate the shared resources effectively and keep the system safe if the key question in Academia MCS research.**

*More levels of confidence can be considered.

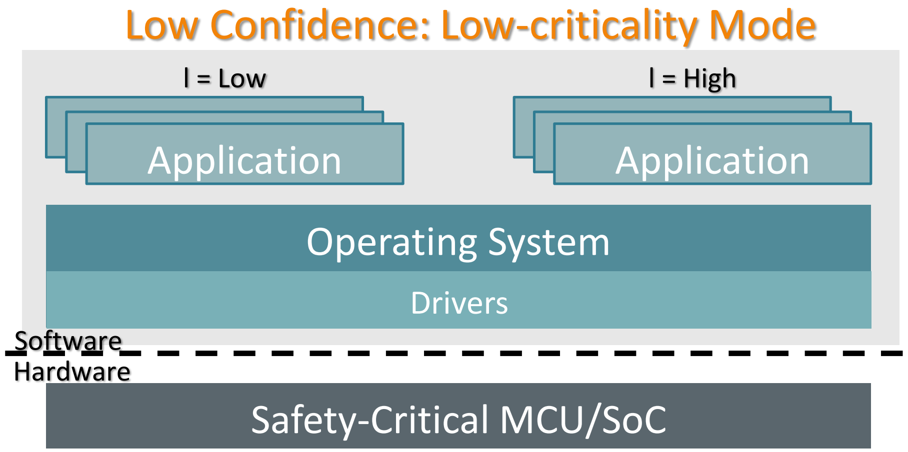
Academia Research on MCS

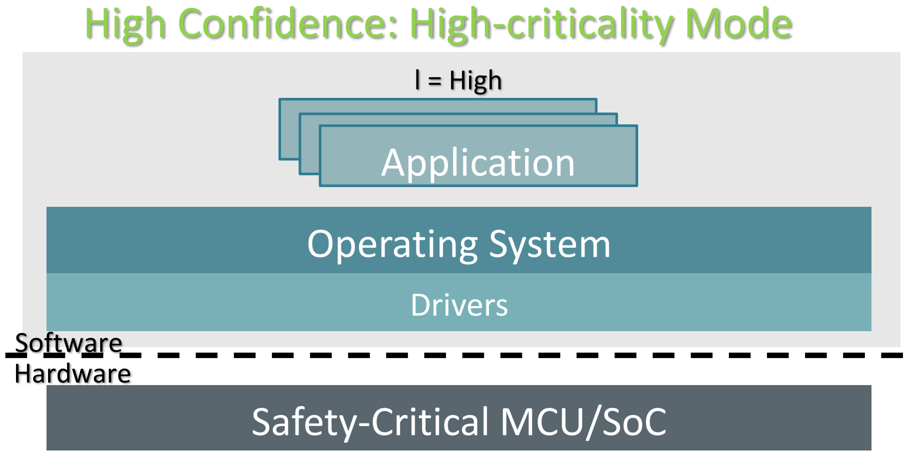
- In the earliest MCS model (i.e., SMC-no), all the tasks used the high confident estimation of WCET.
  - The system is safe
  - Utilization of the resources is low



Academia Research on MCS

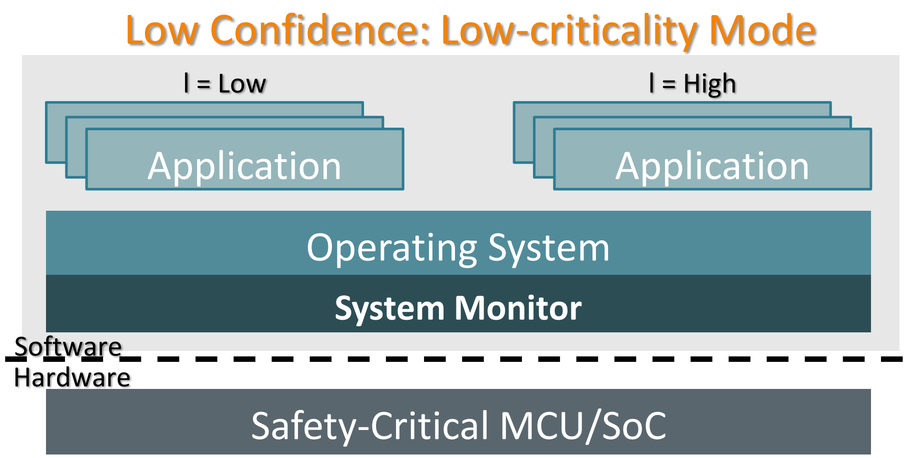
- Adaptive resource management (i.e. AMC) is an effective approach to address the issue, introducing different ***system mode***:
  - System first executes at the *low-criticality mode* (low confident WCET is used)
  - System goes to the *high-criticality mode* (high confident WCET is used), while meeting a predefined condition (e.g., over-run of a task)
  - In the high-criticality mode, low-criticality tasks are terminated.

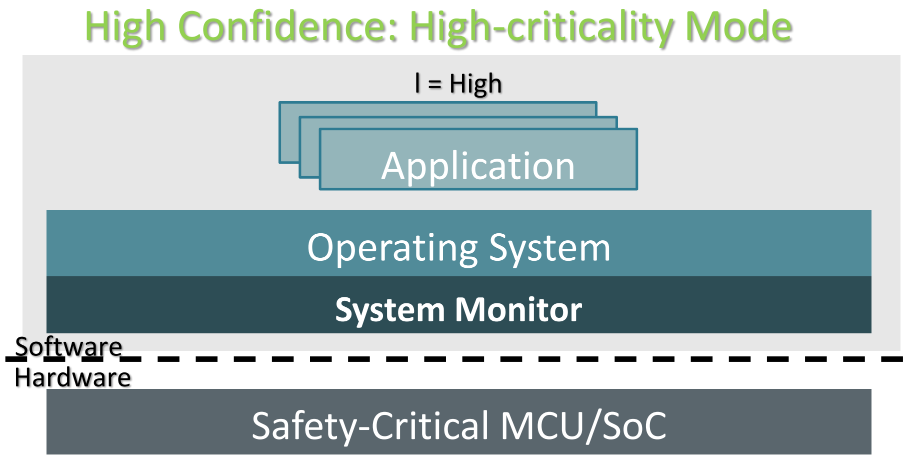




Academia Research on MCS

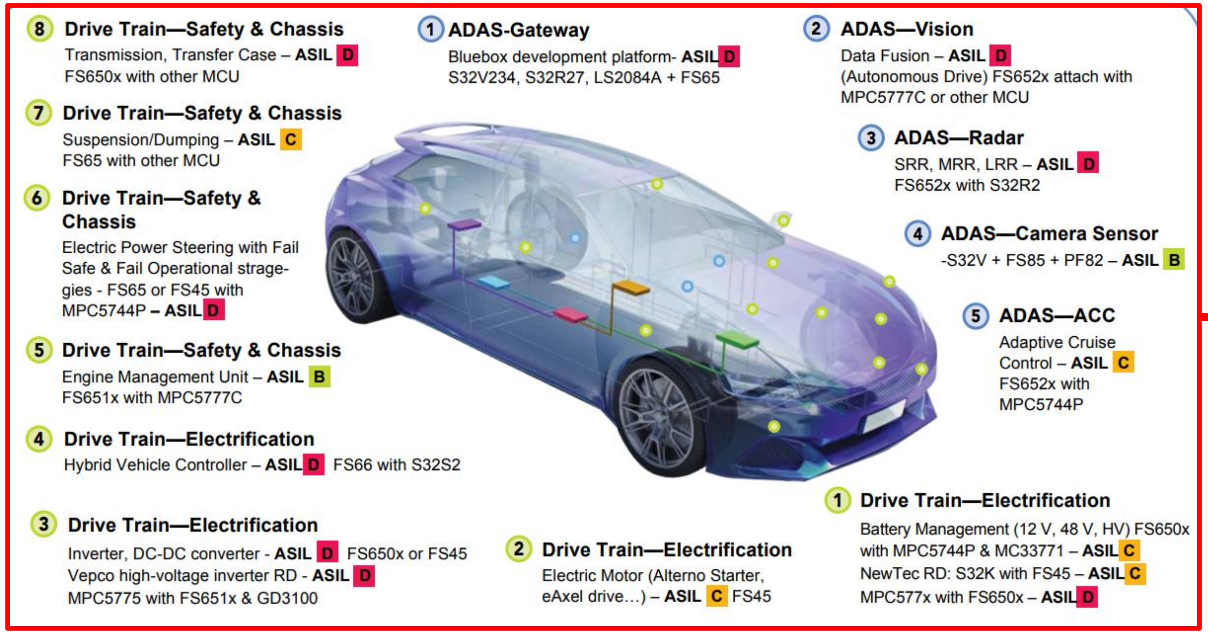
- Adaptive resource management (i.e. AMC) is an effective approach to address the issue, introducing different ***system mode***:
  - System first executes at the *low-criticality mode* (low confident WCET is used)
  - System goes to the *high-criticality mode* (high confident WCET is used), while meeting a predefined condition (e.g., over-run of a task)
  - In the high-criticality mode, low-criticality tasks are terminated.





Mixed-Criticality System in Automotive

- MCS is attractive to Automotive industry
  - With the diverse functionalities required by modern safety-critical systems and the rapid evolution of executed platforms.



Mismatches between
Academia Research and Automotive Industry Practice

ASIL in ISO 26262

- Automotive Safety Integrity Level (ASIL): the degree of rigor that should be applied in development, implementation, and verification of a requirement in order to avoid unreasonable risk in the product
- ASIL-A, ASIL-B, ASIL-C, ASIL-D
  - ASIL-A is the least stringent level
  - ASIL-D the most stringent one
- ASIL assigned via safety analysis, considering
  - Severity: the degree of harm, S1 (no harm) to S3
  - Exposure: probability of occurrence, E1 (very low probability) to E4
  - Controllability: controllability of hazard of the failure C1 (controllable) to C3

ASIL in ISO 26262

Definition of ASIL

Severity	Exposure	Controllability		
		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	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

QM stands for Quality Management, where the assigned requirement can be developed using ordinary QM approaches.

ASIL in ISO 26262

- ASIL Decomposition:
  - a safety requirement can be decomposed into two subsequent safety requirements, where their ASILs can be tailored

- An ASIL-D requirement can be decomposed as:
  - one ASIL-C + one ASIL-A; one ASIL-B + One ASIL-B; one ASIL-D + one QM
- An ASIL-C requirement can be decomposed as :
  - one ASIL-B + one ASIL-A; one ASIL-C + one QM
- An ASIL-B requirement can be decomposed as:
  - one ASIL-A + one ASIL-A; one ASIL-B + one QM;
- An ASIL-A requirement can be decomposed as:
  - one ASIL-A + one QM;

Mismatch

- Suppose we directly use ASIL as the criticality level in AMC
  - Mode switch to high-criticality level: suspend the low-criticality tasks

- Example: when system switches into criticality level ASIL-C
  - ASIL-C and ASIL-D tasks execute
  - ASIL-A and ASIL-B tasks suspended

  - Consider a ASIL-B task
    - Suspend it will increase it exposure from E3 to E4
    - Because its function will always fail (suspended)
    - Suspension effectively makes “it into ASIL-C”
    - We should not suspend it

Severity	Exposure	Controllability		
		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	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

ASILs are statically allocated and cannot be raised at run-time, even though the suspension effectively raises its safety requirement

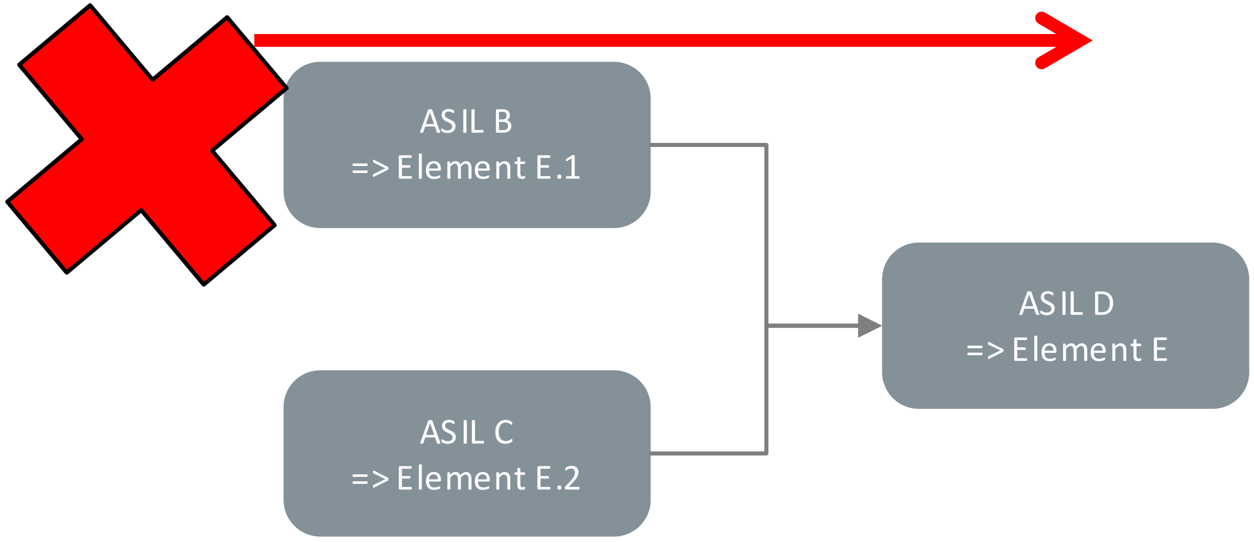
Mismatch

- Another problem is caused by ASIL Decomposition

- For example
  - Safety requirement R1 of ASIL-D decomposed into R1:1 (ASIL-C) and R2:2 (ASIL-A)
  - task #i (with criticality level C) and task #j (with criticality level = A) are tasks fulfilling these two safety requirements.
  - When the AMC model switches from Mode
- A (L = A) to Mode B (L = B), task #j is suspended, which poses threats to fulfilment of R1

Mismatch

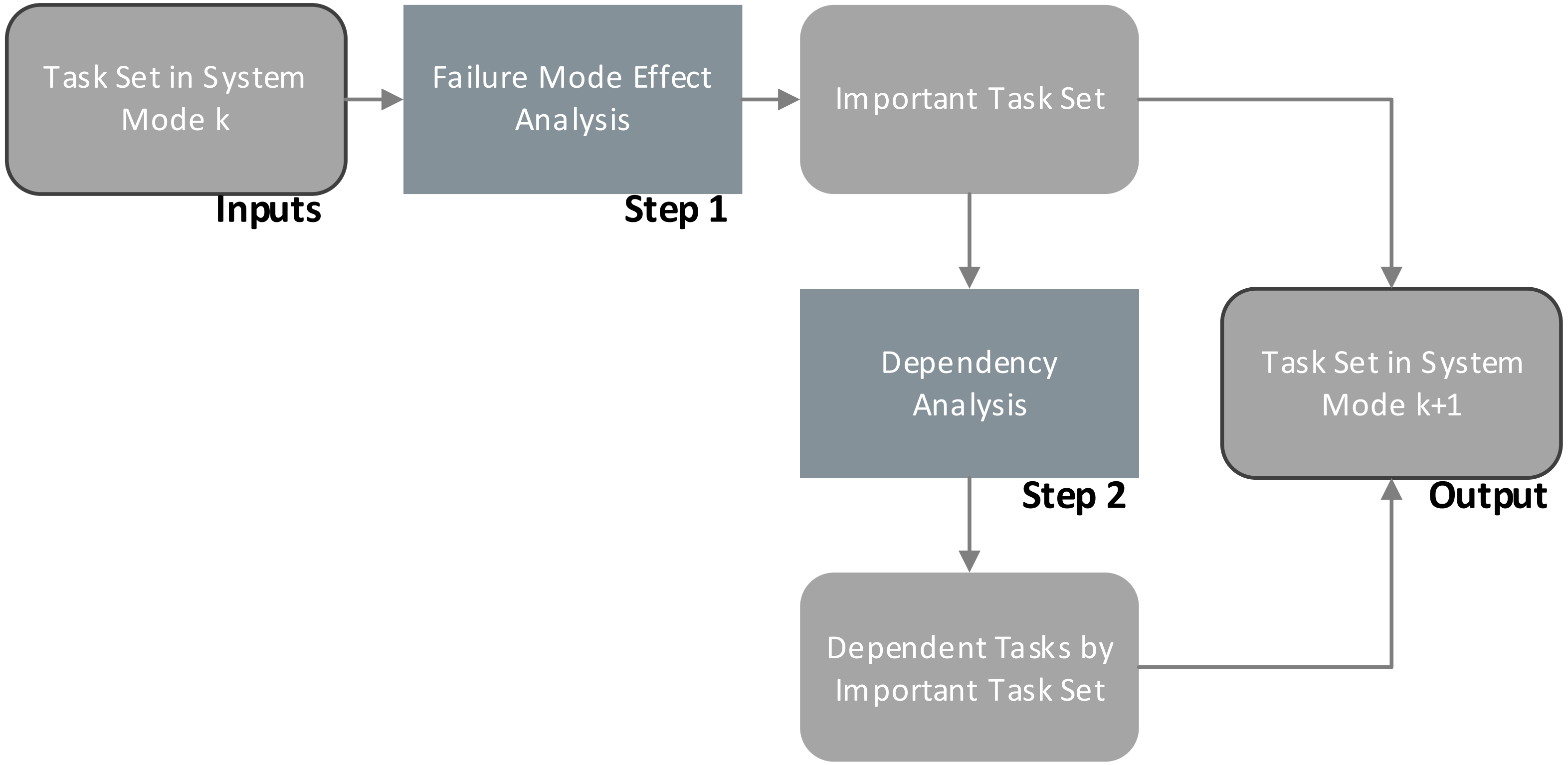
- If a high-criticality tasks is dependent on a low-criticality task, killing low-criticality tasks will cause the corruption of the high-criticality task



Isolation

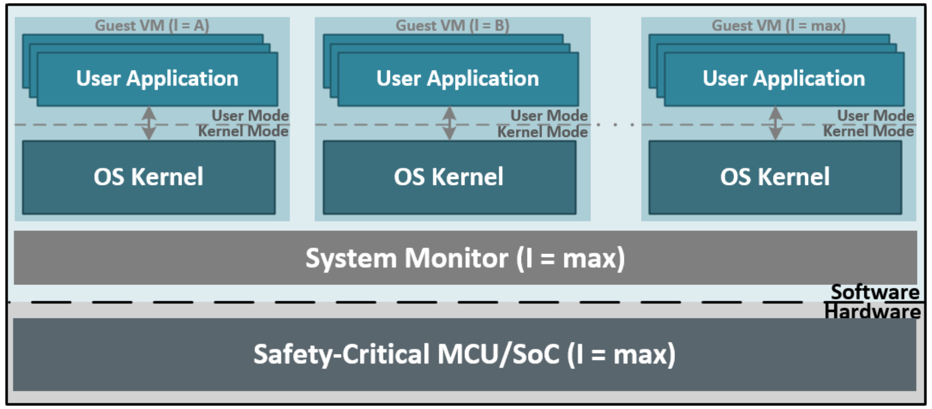
- Isolation between different criticality tasks is regulated by all the safety-related standards.
- This is always the essential requirements.
- ISO26262: *“If freedom from interference between elements implementing safety requirements cannot be argued in the preliminary architecture then the architectural elements shall be developed in accordance with the highest ASIL for those safety requirements”*
- Isolation includes: Timing isolation, space isolation, and fault isolation.

Solution: Run-time Safety Analysis

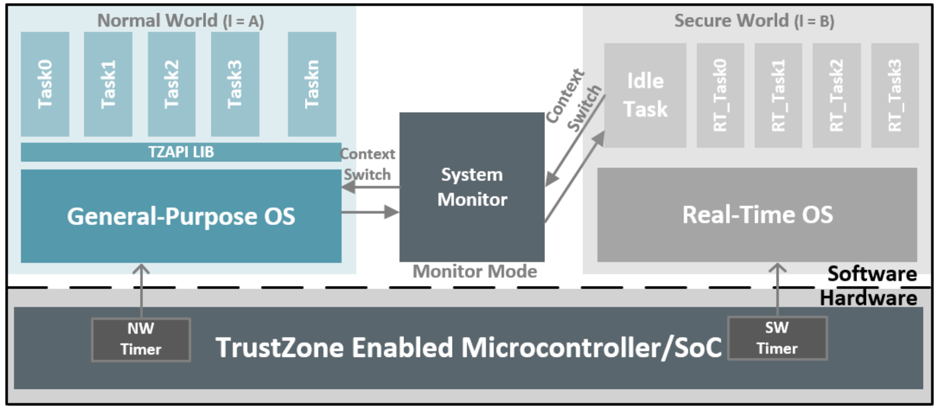


Solution: Three System Architectures

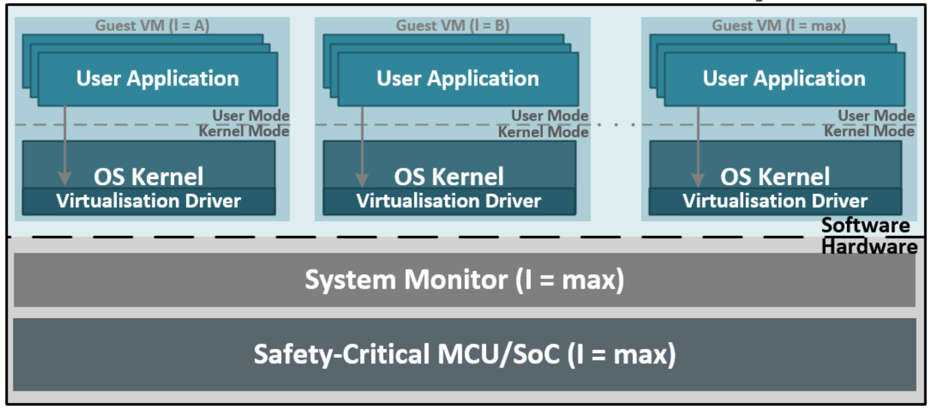
- Software Virtualized System



- ARM TrustZone System



- Hardware Virtualized System



Summary

- MCS is a key direction in safety-critical systems, it is well studied in academia, but still has gaps in industry.

- We propose run-time safety analysis and three system architectures to solve the gaps.

- The main intention of this paper is to encourage tighter connections between academia and industry.