Safety Assurance Driven Problem Formulation for Mixed-Criticality Scheduling

Patrick Graydon, Mälardalens University
Iain Bate, University of York

First International Workshop on Mixed Criticality Systems

MÄLARDALEN UNIVERSITY
SWEDEN

And now for something completely different …

- Most MCS work is from a real time perspective
- So … what does a safety guy make of it?

Vestal's formulation

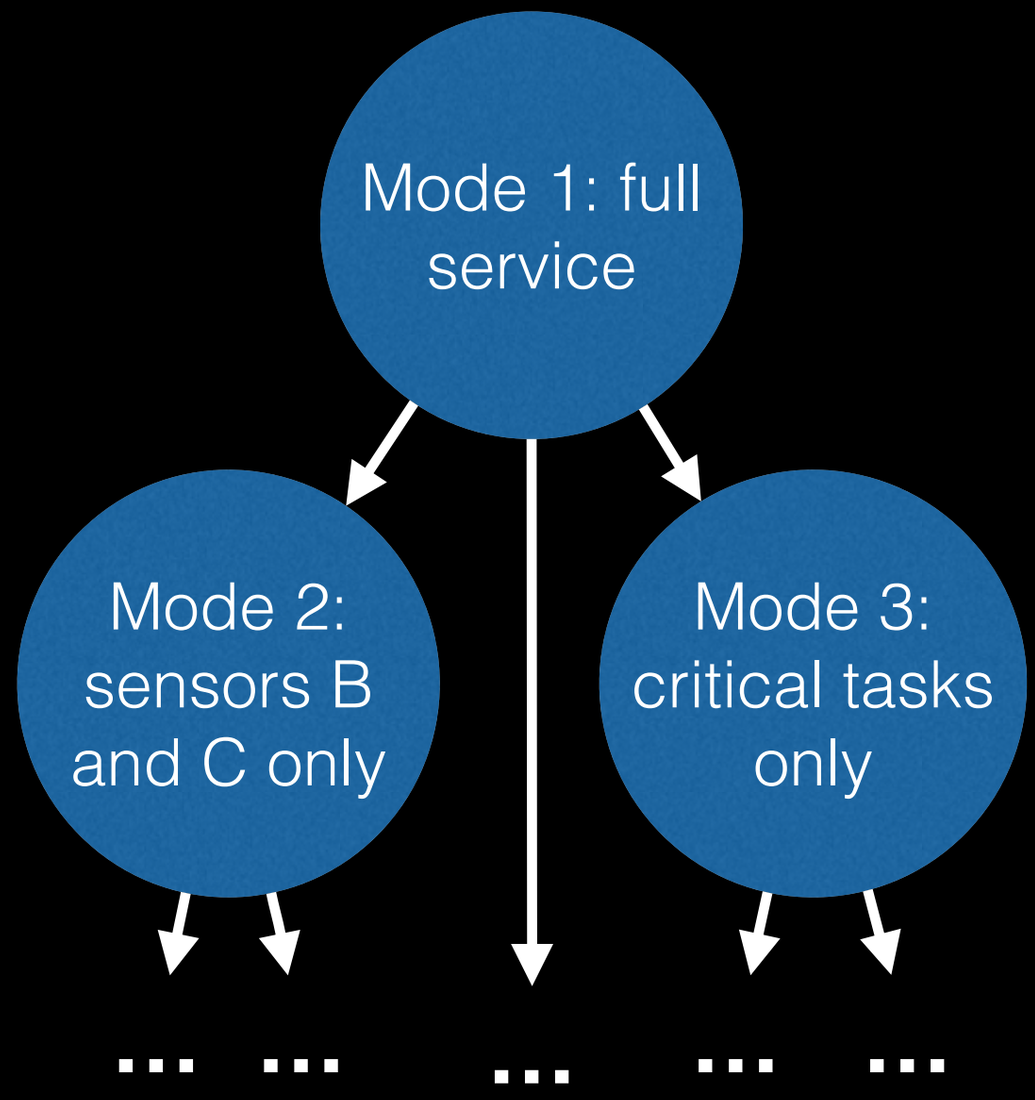
- Tasks $\tau_1 \ldots \tau_n$ with periods T_i and deadlines D_i
- 'An ordered set of design assurance levels' $\mathscr{L}$={A, B, C, D} with A being the highest
- $C_{i,j}$ gives the compute time for τ_i at level *just*
- $C_{i,A} \geq C_{i,B} \geq C_{i,C} \geq C_{i,D}$
- Goal: 'assure to level L_i' that each task τ_i 'never misses a deadline'

Baruah and Burns formulation

- Extends Vestal's model with:
  - Level-dependent periods T_i^l ($l > l' \Rightarrow T_i^l \leq T_i^{l'}$)
  - Level-dependent deadlines D_i^l ($l > l' \Rightarrow D_i^l \leq D_i^{l'}$)
  - A criterion for when an overrun is over and we can start executing less-critical tasks again (namely when the processor is next idle)

Ekberg and Yi formulation

- Support *reconfiguration* more generally
  - ‘The system designer [should] decide what it means … to be in any one criticality mode’
  - DAG *G* defines system *modes* and transitions
  - Task τ_1 is active in mode *m* iff $m \in \tau_i$



WCET confidence monotonicity assumption

- All three formulations explicitly assume *WCET confidence monotonicity*:
  - $\forall$ *i* : tasks, *a*, *b* : crit. levels • $a > b \Rightarrow C_{i,a} \geq C_{i,b}$
- Is this true?

Uncertainty in WCET

	Aleatoric	Epistemic
High Water Mark Testing	**Test coverage**	Tool correctness, configuration management, measurement method
Probabilistic Testing	Sample size, chosen sigma	Tool correctness, CM, measurement method, *distribution suitability*
Hybrid Approaches	Test coverage	Tool correctness, CM, measurement method, *analysis inputs*
Static Analysis	None	Tool correctness, CM, *tool inputs (e.g. loop bounds)*

Horseshoes, hand grenades, & WCET confidence monotonicity

- WCET confidence might not be monotonic
  - Not clear how hybrid and probabilistic approaches fit monotonicity assumption
- Maybe we don't need strict monotonicity …
  - Sometimes conservatism does buy confidence, e.g. *most* HWM testing vs. *most* static analysis
  - A little wording change might fix this problem

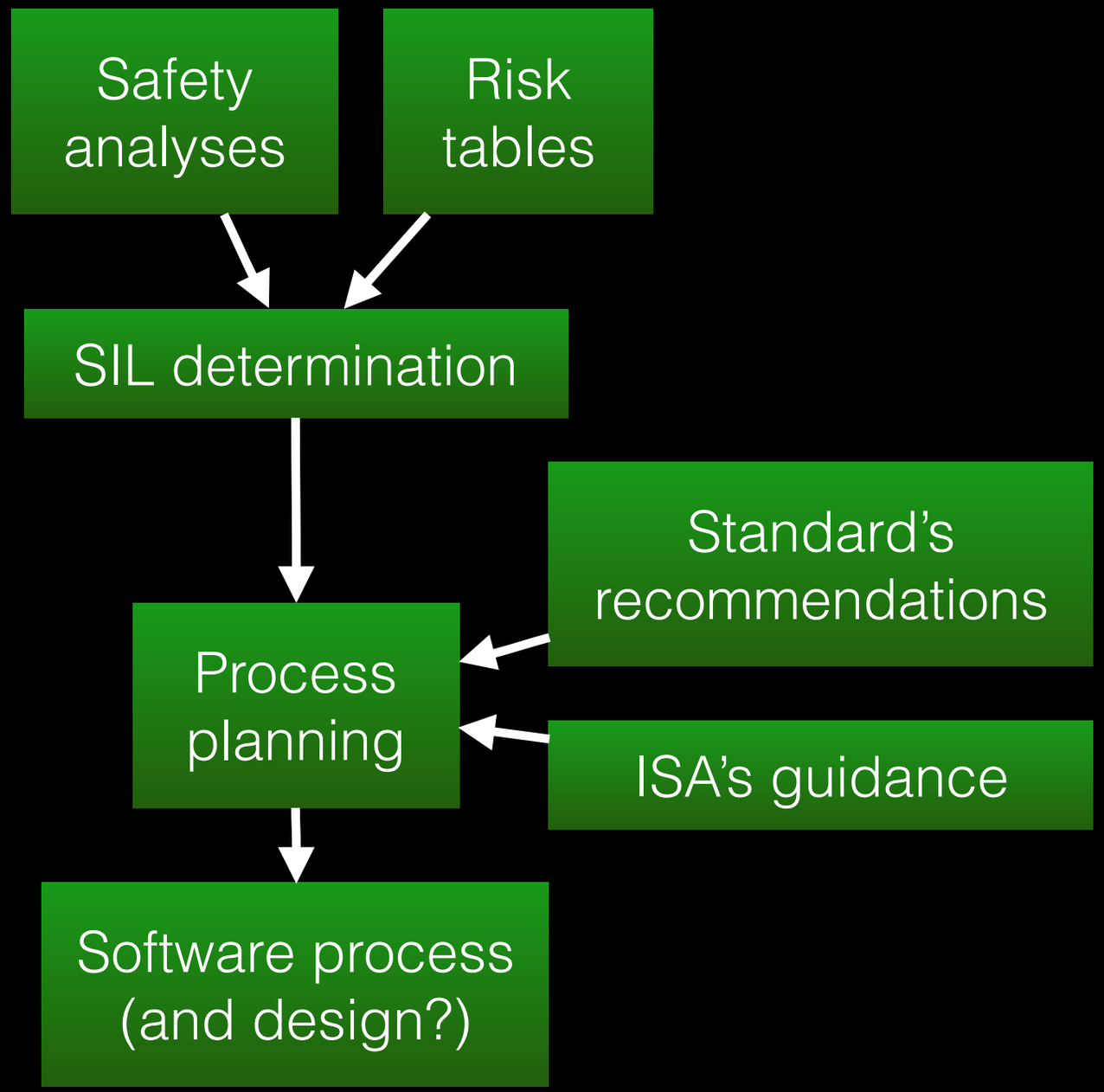
Ask not what safety can do for you …

- Safety standards vary
  - Must satisfy common safety claims … and the objectives of 61508, 50128, 178B/C, 26262, etc.

- But there are some common themes
  - Derived software safety requirements
  - SILs and process rigour
  - Partitioning and integrity
  - Survivability and graceful degradation

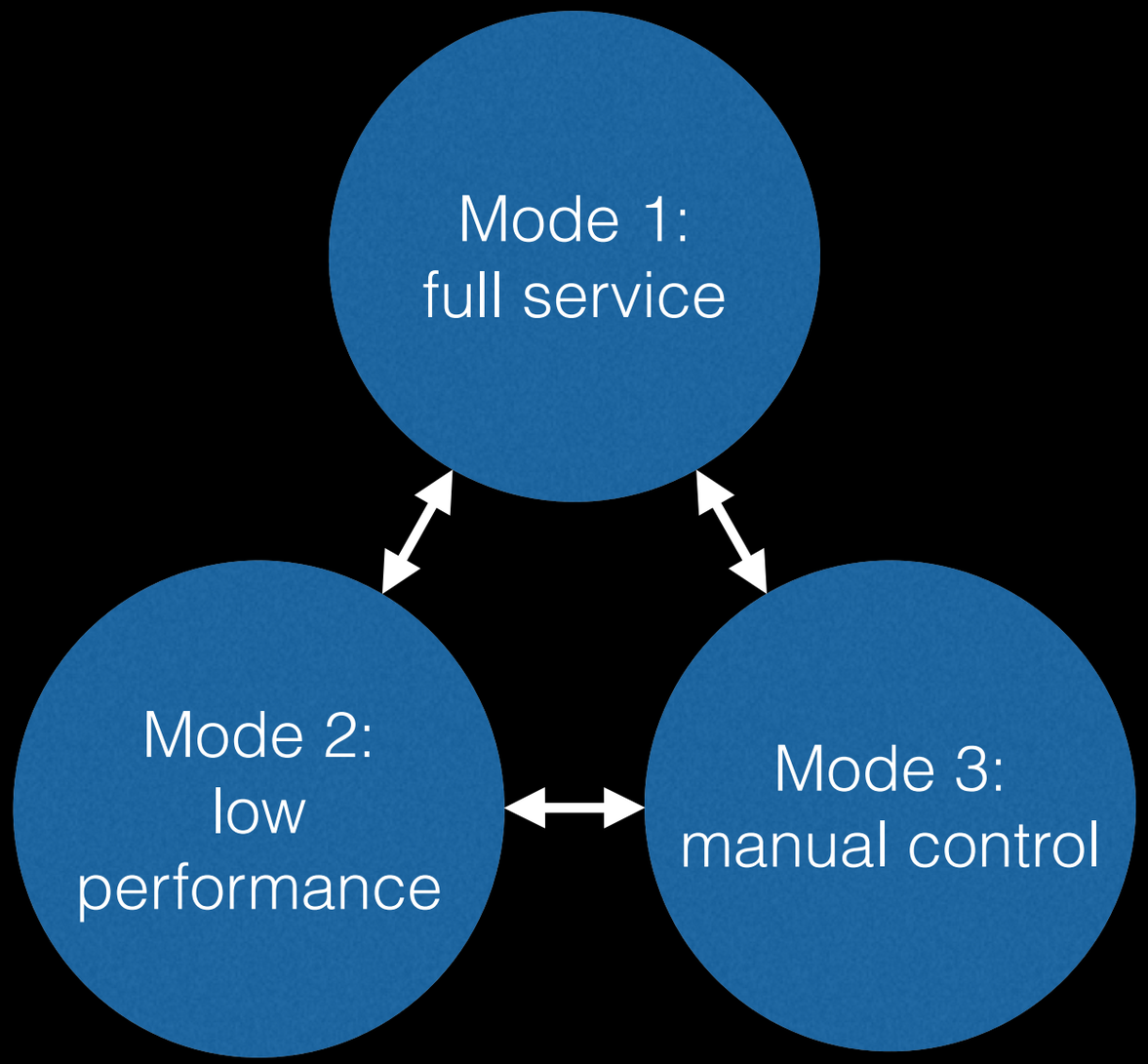
Meaning of ‘critical’ is critical

- Criticality is not deadline, period, or priority (directly)
- In Vestal’s formulation, criticality level is SIL
- SILs are complex and frequently misunderstood
- SIL is related to *importance* and to *confidence*
  - … but it is neither!



Survivability

- Provide *essential services* in the event of attack or failure
- Might mean avoiding designs that ‘go nonlinear’
- Might *also* mean reconfiguration for a different ‘acceptable form of service’
  - Ekberg and Yi call these ‘criticality modes’



Untangling ‘criticality’

- We humbly suggest one term per concept:
  - *Importance*: the consequence of a task overrunning its deadline (in a service mode)
  - *Confidence*: the confidence (absence of uncertainty) in a WCET limit or WCRT figure
  - *Service mode*: the ‘acceptable form of service’ the system is to provide
  - *Mode of operation*: how the operators are using the system at a given time

There are modes, and then there are *modes*

- Survivability and tolerating overruns share similarities *but there are important differences*
- Reconfiguration to tolerate failures might:
  - … involve loading new tasks into memory (e.g. onto a surviving IMA node)
  - … involve blending output from new and old (e.g. when changing aircraft control laws)
  - … be on a different time scale (secs, mins)

Safety assurance

- Reconfiguration for survivability and tolerating overruns have different assurance goals
  - The former shows ‘graceful degradation’
  - The latter shows ‘partitioning integrity’
- Mixing the two might make V&V harder
  - We have to test each mode transition …
    … and each transition trigger …
- Suggestion: keep them separate

To kill or not to kill?

- The path to recovery is not always clear
  - E.g. Ekberg and Yi formulation specifies a *DAG*
- To *never* restart low-importance tasks following a transient overload is … extreme
  - Could be a catastrophe if important tasks depend on ‘at least m-of-n service’ from less-important tasks
- Suggestion: explicit recovery with guarantees

Conclusions

- We love MCS: we can have our cake and eat it, too
- Existing formulations could be improved (from a safety assurance perspective)
  - Relax WCET confidence monotonicity assumption
  - Untangle the multiple meanings of ‘criticality’
  - Separate ‘partitioning’ and ‘survivability’ mechanisms
  - More rigorous treatment of recovery
- Next step: model safety argument surrounding MCS