# Decoupling Criticality and Importance in Mixed-Criticality Scheduling

Konstantinos Bletsas*, Muhammad Ali Awan*, Pedro F. Souto†, Benny Akesson‡, Alan Burns§, Eduardo Tovar*
*CISTER Research Centre and ISEP/IPP, Porto, Portugal
†University of Porto, FEUP-Faculty of Engineering and CISTER Research Centre, Porto, Portugal
‡ESI (TNO), Eindhoven, the Netherlands
§Department of Computer Science, University of York, UK

***Abstract*—Research on mixed-criticality scheduling has flourished since Vestal's seminal 2007 paper, but more efforts are needed in order to make these results more suitable for industrial adoption and robust and versatile enough to influence the evolution of future certification standards in keeping up with the times. With this in mind, we introduce a more refined task model, in line with the fundamental principles of Vestal's mode-based adaptive mixed-criticality model, which allows a task's criticality and its importance to be specified independently from each other. A task's importance is the criterion that determines its presence in different system modes. Meanwhile, the task's criticality (reflected in its Safety Integrity Level (SIL) and defining the rules for its software development process), prescribes the degree of conservativeness for the task's estimated WCET during schedulability testing. We indicate how such a task model can help resolve some of the perceived weaknesses of the Vestal model, in terms of how it is interpreted, and demonstrate how the existing scheduling tests for the classic variant's of Vestal's model can be mapped to the new task model essentially without changes.**

## I. Background

### A. Introduction

Mixed-criticality systems are an important niche of real-time embedded systems, their defining characteristic being the fact that computing tasks of different criticalities execute on the same hardware and share system resources[1]. The criticality of a task is a measure of the severity of the consequences of a task failing (which, in the context of real-time scheduling means missing its deadline). Indeed, some tasks missing their deadline can have catastrophic consequences, whereas other tasks occasionally missing their deadlines might only have a minor effect. For this reason, the higher a task's criticality, the more conservative (and costlier, in terms of effort, time and money) the approach employed to upper-bound that task's worst-case execution time[2].

The sharing of system resources among tasks of different criticalities, unless carefully managed, can give rise to undesirable interactions that compromise safety. For this reason, and before getting to specific scheduling arrangements, the various certification authorities generally prescribe that (i) applications of lower criticality should not be able to cause the failure of tasks of higher criticality, and (ii) when tasks of different criticality share the same resources, they must all be engineered to the same strict standard of safety as the highest-criticality task thereamong. Clearly, this is inefficient. For this reason the certification authorities, in their guidelines [4], do not insist in zero interference among mixed-criticality applications, but instead expect such interference to be carefully accounted for and adequately mitigated[3]. In the context of the WCET problem, also analysing low-criticality tasks using highly conservative and pessimistic static WCET analysis techniques, as in the case of highly-critical components, would be wasteful of processing capacity. In fact, it would defeat the purpose behind the strong industrial shift to mixed-criticality scheduling, which is to efficiently utilise today's powerful multicores and reduce costs, size, weight and power.

### B. Vestal's model and its evolution

This reality motivated Vestal to propose the use of different WCET estimates with different corresponding degrees of assurance, for the same task, in different scenarios, in order to ensure *a priori* the correct temporal behavior of the system at run-time [3]. To illustrate the principle, consider a fixed-priority-scheduled system and assume that each task has both a (i) "reasonable" but, not conclusively safe, WCET estimate and (ii) a highly pessimistic, but demonstrably safe, WCET estimate. Then, when testing the schedulability of a low-criticality task, one would only need to use the "reasonable" estimates, for all higher-priority tasks, irrespectively of their criticality, as inputs to the familiar Worst-Case Response Time (WCRT) analysis [5] for the task under consideration. Conversely, testing the schedulability of a high-criticality task, one would use the respective pessimistic estimates. This initial model, coupled with a fixed task priority scheduling policy, was termed Static Mixed Criticality (SMC).

Baruah and Burns extended Vestal's initial model [6] by adding *modes* and the notion of run-time *system criticality lev-*

Work partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) within the CISTER Research Unit (CEC/04234).

[1] When tasks of different criticalities exist but are completely isolated, such systems are *multiple-criticality*, as opposed to mixed-criticality, and they constitute a different class of systems. See Footnote 1 in [1] and in [2].

[2] In Steve Vestal's words: " . . . the more confidence one needs in a task execution time bound (the less tolerant one is of missed deadlines), the larger and more conservative that bound tends to become in practice." [3]

[3] For example, the CAST-32A guidelines [4], clarify that "it is therefore important to identify the interference channels that could cause interference between the software applications hosted by their MCP platform, to mitigate the effects of each of those interference channels and to verify the selected means of mitigation".

*el*. We will henceforth refer to their model as the *Mode-based Vestal model*. In this variant model, each task has a (design) criticality level and a set of WCET estimates – one for every criticality level not exceeding its own and non-decreasing with respect to the latter. At startup, the "system criticality level" (in reality, an index of the system mode) is initialised to the lowest task criticality. If a task exceeds its WCET for the system's current criticality level, the system stops all tasks with criticality equal to that level and increments its criticality level. This constitutes a *mode change*, upon which, all tasks with criticalities lower than the current system criticality, are idled. Coupled with fixed-priority scheduling, the mode-based Vestal model is known as Adaptive Mixed Criticality (AMC), but the model itself is orthogonal to the scheduling policy. For example, it can be coupled with EDF [7], [8].

The execution time monitoring and dropping of lower-criticality tasks that exceed their WCET estimate for a given mode was a clever idea for the following reason: If a task cannot execute for more than a certain time (corresponding to that WCET estimate) without getting dropped, then that estimate (which could even be an underestimation of its true WCET), **becomes** a provably safe, unexceedable WCET estimate for the task – it will never execute for more than that. This solves the problem of unpredictable interference on higher-criticality tasks, assuming that it is acceptable to drop lower-criticality tasks, during the mode change, in the first place. This assumption in AMC may, however, not always hold, as we will discuss later, in Section I-D.

### C. Terminological issues

On a related observation, and to quell some long-standing terminology-related confusion, we note what were above referred to as "WCET estimates" (or often simply "WCETs") for a given task in different modes, are in reality execution time thresholds for triggering a switch to the next-higher mode. Except for the estimates used in the top-most mode (which need to provably upper-bound the true, but unknown WCETs), the estimates (i.e., thresholds) used in other modes are *reasonably expected*, but *not required*, to upper-bound the true WCET. However, too low a value increases the probability of a mode change (undesirable, due to the degradation of functionality entailed by dropping tasks) while a value that is too high wastes processing resources. In any case, the selection of these thresholds is up to the designer. Another terminology-related source of confusion is the use of the term "system criticality" to denote what is, essentially, an indicator of the mode. Strictly speaking, "criticality" characterises applications and their tasks. This is discussed more in Section IV.

### D. Other criticisms

Criticisms are sometimes voiced about the mode-based Vestal model [9], [10], [11], [12], [13] and its compatibility with the safety standards (e.g., IEC61508, ISO26262 or DO-178C) on which system certification is based. This is partially due to Vestal's use of the term "criticality" in a looser sense than the meaning it has in the standards (and the precedent that this set in the use of the term in the acadamic literature) and partially due to more legitimate concerns.

One of the more legitimate concerns, is that dropping tasks upon a mode change simply on the basis of their criticality, is not necessarily acceptable course of action in the general case. More generally, a task's criticality is not synonymous with its *importance*, which is a different attribute – and important tasks should not be discarded.

In response to such concerns, in this work **we introduce an extension of the mode-based Vestal model, whereby the importance of a task is decoupled from its criticality, and is specified separately**. The participation of a task in different modes follows from its importance, not its criticality. Meanwhile, its criticality, which determines the degree of conservativeness in its development process, also determines (in our model, just as in the standard mode-based Vestal), the degree of conservativeness in the estimation of its WCET in the different modes that it forms part of.

Note that this use of the term "importance" is different from that employed in [14] where it is simply used as a means of differentiating between tasks of the same criticality.

### E. Outline of this paper

The rest of this paper is structured as follows. In Section II, we describe this new model. In Section III, we describe how the schedulability analyses developed for the standard mode-based Vestal model can be mapped to the new decoupled model, **essentially without changes**. Subsequently, in Section IV, we discuss some of the existing criticisms to the standard mode-based Vestal model, and how our new task models addresses those. Section V offers concluding remarks.

## II. The Decoupled Task Model

Consider a set $\tau \overset{\text{def}}{=} \{\tau_1, \dots \tau_n\}$ of $n$ mixed-criticality sporadic tasks. Each task $\tau_i$ has a minimum inter-arrival time $T_i$ and a relative deadline $D_i \leq T_i$. It also has a *worst-case execution time* (WCET), whose exact value is in practice unknowable, and can only be estimated. Different estimates can be obtained for the same task by the use of different techniques:

1) **Provably safe** WCET estimates, which are obtained by formal analysis and/or static path analysis, with rigorous pessimistic assumptions. They tend to be excessively pessimistic and costly to derive, in terms of time, effort and money.
2) **Potentially unsafe** WCET estimates, i.e., *probably* but not *provably* safe. These may be derived by simplistic path analysis, or via measurements and perhaps probabilistic techniques.

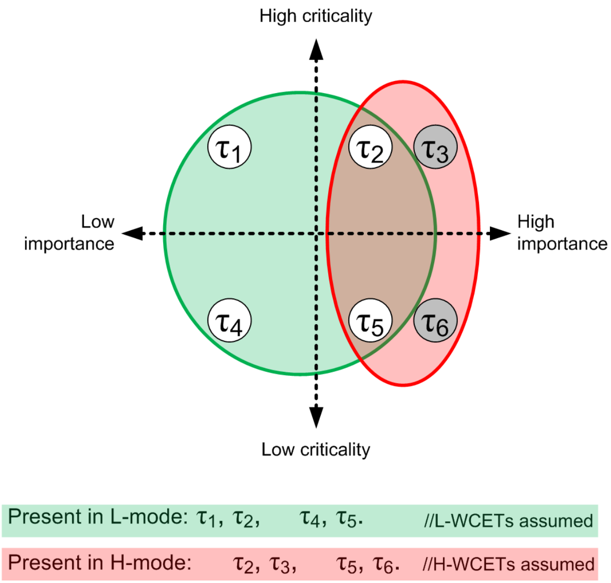
Although Vestal's model has been generalised to an arbitrary number of criticality levels (denoted numerically), in this work, for simplicity we assume just two criticality levels, high (H) and low (L) – which suffice in order to illustrate the principle. A task's criticality is denoted by $\kappa_i$.

Under the variant of Vestal's model introduced with AMC, there would be two modes (L and H), with all tasks executing in the L-mode and only the H-tasks executing in the H-mode. Instead, in our model, in addition to its criticality $\kappa_i$, each task is also characterised by (what we conventionally call) its *importance* $\lambda_i$, which in the general case is indicated numerically. In this work, however, we assume that it can be high (H) or low (L), for simplicity. This attribute, input by the designer, reflects the design and application requirements and the implication is that, if needed, a low-importance task can be "dropped" (i.e., idled, at mode change) whereas a high-importance task cannot. Therefore, under our model, which tasks execute in which mode is determined on the basis of their importance, not their criticality[4]. This is a more general model than that by Baruah and Burns [6], which can be described as a special case (namely, $\lambda_i = \kappa_i,\ \forall i$).

In accordance with the above principles, there exist 4 possible classes of tasks, corresponding to the possible combinations of criticality ($\kappa_i \in \{L,\ H\}$) and importance ($\lambda_i \in \{L,\ H\}$).

Tasks of high importance (irrespective of their criticality) can also sub-categorised into (i) tasks that are present in the system already at start-up and (ii) tasks that are only introduced after the mode change (e.g., as fail-safes for one or more tasks that were dropped). Figure 1 illustrates this model, with one task for each kind. Initially (i.e., in L-mode), there exist 4 tasks in the system ($\tau_1$, $\tau_2$, $\tau_4$ and $\tau_5$, encircled in green). All of those have different combinations of criticality and importance. In L-mode, the system must be provably schedulable as long as no task executes for more than its corresponding WCET estimate for that mode ($C_i^L$). However, if any of those 4 tasks exceeds its $C_i^L$, then a mode change is triggered. The low-importance tasks $\tau_1$ and $\tau_4$ are then immediately dispensed with; the high importance tasks $\tau_2$ and $\tau_5$ persist in the new mode. Additionally, high-importance tasks $\tau_3$ and $\tau_6$ are added to the system, possibly to compensate for the functionality of the dropped tasks. In the H-mode, it has to be offline-provable that no task among those present ($\tau_2$, $\tau_3$, $\tau_5$ and $\tau_6$, encircled in red), can miss a deadline, assuming that these tasks execute for up to their corresponding WCET estimate for the new mode ($C_i$). This requirement also applies to jobs caught in the mode transition (i.e., released before the mode change, but completing after the mode change).

Note that our model imposes no constraint between the number of (low-importance) tasks dropped at mode change and the number of new (high-importance) tasks added to the system after the mode change (e.g., to compensate for their functionality). Neither is there any constraint on how the attributes of those tasks ($D_i$, $T_i$, $C_i$) can be related. For convenience, a task present only in the H-mode can be equivalently modelled, for schedulability analysis purposes, as a task present in both modes, with $C_i^L = 0$. Meanwhile, for low-criticality but high-importance tasks (i.e., for which a WCET estimate $C_i^H$ for H-mode execution needs to be defined), we think that it is reasonable to use $C_i^H = C_i^L$. The reason is that rare jobs that exceed this execution time can be dropped, even in the H-mode (because the job is not critical) but the task overall cannot be dropped (because it is important) and its next job will arrive and be executed as normal. Still, there is nothing in our model that prevents the designer from specifying some other $C_i^H > C_i^L$ for a low-criticality, but high-importance, task $\tau_i$. As for important tasks ($\lambda_i = H$) that cannot tolerate even a single dropped job, this implies that they are in fact high-criticality and need to be specified as such by the designer (i.e., $\kappa_i = H$); their WCET estimates for the H-mode would accordingly also need to be provably safe.



Present in L-mode: τ1, τ2, τ4, τ5. //L-WCETs assumed

Present in H-mode: τ2, τ3, τ5, τ6. //H-WCETs assumed

Fig. 1: A Venn diagram illustrating the different types of tasks in our task model, the modes that they can be part of and the WCET estimates used for schedulability-testing purposes.

Ultimately, $C_i^H \geq C_i^L$, for every task that is part of the H-mode.

## III. Uniprocessor Analysis

Having introduced the task model, we proceed with showing how schedulability analysis formulated for the standard mode-based Vestal model can be mapped to it. The only change to the equations is that the task selector for the different modes is now the task importance. For illustration purposes, we assume a uniprocessor system and a fixed priority scheduling policy. For this case, and for the standard mode-based Vestal model, the literature offers the well-known AMC-rtb and AMC-max tests (both formulated in [15]). For schedulability testing in L-mode, for both AMC-rtb and AMC-max, a task's worst-case response time (WCRT) is upper-bounded by

$$R_i^L = C_i^L + \sum_{\tau_j \in hp(i)} \left\lceil \frac{R_i^L}{T_j} \right\rceil C_j^L \qquad (1)$$

[4] In [11], Esper et al. discuss examples of abstract systems where a task's criticality does not reflect its importance.

where $hp(i)$ is the set of higher-priority tasks and $C_j^L$ is the WCET estimate for $\tau_j$ in L-mode.

For the schedulability testing in H-mode, the WCRT equations, for AMC-rtb and AMC-max, respectively, are

$$R_i^H = C_i^H + \sum_{\substack{\tau_j \in hp(i) \\ \kappa_j = H}} \left\lceil \frac{R_i^H}{T_j} \right\rceil C_j^H + \sum_{\substack{\tau_\ell \in hp(i) \\ \kappa_\ell = L}} \left\lceil \frac{R_i^L}{T_\ell} \right\rceil C_\ell^L \tag{2}$$

and

$$R_i^H = \max(R_i^s), \ \forall s \in \{0, R_i^L\} \tag{3}$$

where

$$R_i^s = C_i^H + \sum_{\substack{\tau_\ell \in hp(i) \\ \kappa_\ell = L}} \left( \left\lfloor \frac{s}{T_\ell} \right\rfloor + 1 \right) C_\ell^L + \sum_{\substack{\tau_j \in hp(i) \\ \kappa_j = H}} \left\{ M(j,s,R_i^s) C_j^H + \left( \left\lceil \frac{t}{T_j} \right\rceil - M(j,s,R_i^s) \right) C_j^L \right\} \tag{4}$$

where

$$M(j,s,t) = \min \left\{ \left\lceil \frac{t - s - (T_j - D_j)}{T_j} \right\rceil + 1, \left\lceil \frac{t}{T_j} \right\rceil \right\} \tag{5}$$

Under the new model, what changes is that, in the degraded mode, the subset of tasks executing consists of all tasks with $\lambda_j = H$ (high importance) instead of $\kappa_j = H$ (high criticality). However, it still holds that $C_i^H \geq C_i^L$, for every task that is part of the H-mode. Correspondingly, Equation (1) need not be modified at all, whereas Equation (2) (AMC-rtb) is slightly modified to

$$R_i^H = C_i^H + \sum_{\substack{\tau_j \in hp(i) \\ \lambda_j = H}} \left\lceil \frac{R_i^H}{T_j} \right\rceil C_j^H + \sum_{\substack{\tau_\ell \in hp(i) \\ \lambda_\ell = L}} \left\lceil \frac{R_i^L}{T_\ell} \right\rceil C_\ell^L \tag{6}$$

and Equation (4) is changed to

$$R_i^s = C_i^H + \sum_{\substack{\tau_\ell \in hp(i) \\ \lambda_\ell = L}} \left( \left\lfloor \frac{s}{T_\ell} \right\rfloor + 1 \right) C_\ell^L + \sum_{\substack{\tau_j \in hp(i) \\ \lambda_j = H}} \left\{ M(j,s,R_i^s) C_j^H + \left( \left\lceil \frac{t}{T_j} \right\rceil - M(j,s,R_i^s) \right) C_j^L \right\} \tag{7}$$

with Equations (3) and (5) entirely unaffected. We typeset the modified Equations (6) and (7) with the affected terms in oversized red, in order to highlight how minimal and how straightforward the changes are. Note that the schedulability test for AMC-max is still safe even when it has to be guaranteed that jobs caught in the mode transition of tasks that are to be dropped shall not be terminated if they do not exceed their L-mode WCETs[5]. This property of AMC-max also holds under our model, if, due to design requirements, such semantics need to be enforced.

Adapting other schedulability tests for the standard mode-based Vestal model (e.g., for an EDF scheduling policy, with uniform [7] or per-task [8] deadline scaling), is analogous.

## IV. Major misconceptions about the Vestal model

Having introduced our task model and shown how its schedulability analysis is available "for free" from the literature on the standard mode-based Vestal model, we are going to briefly examine to what extent its adoption settles some criticisms voiced (e.g., in [9], [10], [11], [12], [13]) at the standard mode-based Vestal model that inspired it. We also use the opportunity to highlight why some other criticisms are misframed.

### A. *Conflating the software assurance level of a task with the notion of importance*

In systems with criticality concerns, tasks are part of one or more system components or functions, which in turn are assigned assurance levels. In the automotive domain, these are called *Safety Integrity Levels* (SILs) whereas in avionics, *Development Assurance Levels (DALs) are the equivalent concept*. Each task associated to a system component inherits the latter's SIL (DAL), with tasks belonging to multiple components (or components that are not partitioned) inheriting the highest SIL (DAL) thereof (see p. 10 in [16]). Tasks must be developed in accordance with the rules defined for their SIL/DAL.

As already mentioned, a major legitimate criticism (e.g., in [9]) on the mode-based Vestal model is that all tasks of a higher-criticality system component (i.e., higher SIL) are always given higher importance than the tasks of any lower-criticality system component (lower SIL). In other words, the importance of a task is treated as depending entirely on its criticality level. In reality though, as the critics correctly point out, some tasks of a higher-criticality component may be unimportant tasks that simply "inherited" their higher SIL due to their interaction/communication with other tasks, whose failure would be catastrophic. Conversely, there may be tasks in a lower-criticality component whose failure can be catastrophic.

As explained, our proposed model decouples the concept of criticality from that of importance. The system designer has the flexibility to specify the importance of different tasks, in accordance with the nature of the system, and decide on their placement into the different modes of operation. Hence we believe that this fully resolves the particular criticism.

[5] Quoting from the original AMC-max paper [15]: "For a possible alternative system model (in which all low criticality tasks, that have been released but not yet completed, are allowed to consume up to C(L) before being descheduled) this bound is tight."

### *B. Graceful degradation*

Graceful degradation of the system has been recommended in safety standards (e.g., IEC61508), whereby the system is allowed to enter in a degraded mode with limited services without compromising its safety. The conventional techniques for the mode-based Vestal model aim for graceful degradation by dropping the low-criticality tasks, under the implicit assumption that these are the less important tasks. However, as explained, importance is not just a function of the criticality level, as it may also depend on the mode of operation and application context. Hence, one valid criticism is that this is a flawed attempt at graceful degradation.

Our proposed model allows the system designer to define a degradation policy based on the importance of each task, irrespective of its criticality and mode of operation. The system designer can thereby specify which tasks are to be dropped, kept or added at each mode transition.

### *C. WCET estimates*

It has been noted (e.g., in [10], [17], [11] that nowhere do the safety standards foresee the existence of multiple WCET estimates for the same task, and that this would bring into question the compatibility of the mode-based Vestal model with these standards. To this observation, we counter that (as also noted in Section I) this is a terminological issue, even for the standard mode-based Vestal model [15]. What in the literature of the Vestal model are refered to as additional, non-provably-safe "WCET estimates", are in fact *execution time thresholds* whose excedance triggers the switch to the next mode. The consequence of setting such a threshold too high or too low (by using an execution time estimation technique that is, correspondingly, more/less conservative) is, respectively, inefficient platform utilisation vs. greater likelihood of triggering a mode change (which is also undesirable). It is a design tradeoff, and both in the standard mode-based Vestal model [15] and ours, nothing prevents the designer from specifying a provably safe WCET estimate for a task in all modes that it is part of. The fact that these estimates tend to be conventionally called (simply) "WCETs" by the people in the real-time scheduling community, obscures their true nature. However, this usage simply follows from the fact that they do behave like WCETs, when they are input into the schedulability tests that the researchers construct.

As Baruah already noted [18], even if the standards do not explicitly foresee the use of multiple execution time estimates per task, they offer no technical arguments precluding their use either, as part of a technique devised for proving the desired safety guarantees for the system. Which is why he surmises that objections to the use of multiple execution time estimates "seem in large part to be a social and cultural problem, rather than a technical one" [18].

### *D. Temporal and spatial isolation*

The mixed criticality applications hosted on the same multicore platform can (in the absence of mechanisms preventing this) interfere with each other on multiple shared channels, including CPU, caches, memory buses, memory controllers and I/O devices [19]. Many mitigation and prevention techniques are proposed in the literature to eliminate or predictably reduce the interference among applications of different criticality. Several works already exist [20], [21], [22], [23], [24], [25], [24], [26], [27], [28], [29], [30], [31], [32], [13], [33] on ensuring temporal and spatial isolation. Some of these works already explicitly assume the mode-based Vestal model while the others can still work in its context. For instance, server-based techniques can be used to ensure temporal isolation at the CPU level [23]. The Cache Lockdown approach [34], [35] proposed in the context of the SCE framework [22] allows spatial isolation at the cache level. Similarly, the use of MemGuard [20] (and memory access regulation in general) allows for upper-bounding memory-access-related stalls and integrating them into the scehedulablity analysis.

Recently, a concern has been raised [11] that a misbehaviour in the minimum interarrival time of a task can affect the temporal isolation of a mixed-criticality system. We believe that further study is required within the context of the mode-based Vestal model to address this challenge of variation in the minimum interarrival time of a task. One way to deal with this, entirely in accordance with the spirit of the mode-based Vestal model, would be for such an inter-arrival time violation to trigger a mode change, analogously as done with execution times. Certainly, the literature contains works which consider the change of interarrival time parameters in mixed-criticality systems in different modes (e.g, [36]). Baruah also shares our view that, rather than just execution times, the theory pertaining to the Vestal model "could be used to deal with any form of inherent uncertainty and nondeterminism with regard to the run-time behavior of systems" [18].

### *E. Mode switch*

In an industrial context, the term "criticality" refers to the level of assurance (SIL, DAL or ASIL) applied in the software development process of the safety-critical application. In the standard mode-based Vestal model, the term is "overloaded" to represent two additional concepts: the mode of operation and importance of a task. The latter comes from the (contested) assumption, in that work [15] that a task's importance is solely determined by its criticality. With respect to the former "overloaded" meaning, when a system switches from mode $n$ to mode $n + 1$, this does not really mean that the tasks change their criticality from criticality $n$ to criticality $n + 1$; it just refers to the transition in mode of operation. Despite the confusing terminology, the criticality of any application or task is not changed due to the mode transition.

Nevertheless, we recommend to other researchers in academia to use that term "mode of operation" rather than, e.g., "low system criticality" when refering to the mode, to avoid any confusion. Moreover, the decoupling of criticality from importance in our proposed model eliminates the concerns about the misuse of the term "criticality" to indicate the importance. Finally, it is worth emphasising that, in any case, this misuse of the term has no impact on the validity of the existing analyses for the mode-based Vestal model.

## V. Conclusion

We introduced a variant of the mode-based Vestal task model for mixed-criticality systems, in order to address some criticisms about the model and bridge the gap with current industrial practice. Its main feature is the decoupling of task criticality from task importance. The new model retains the essence of its predecessor (run-time robustness and efficient processor utilisation via flexible mode transition), and it remains fully backwards compatible with all its existing analyses. We also pointed out how this new model settles the concerns voiced about Vestal's mode-based model and its terminology.

To conclude, although we have no intention of discouraging anyone from exploring other techniques and paradigms, we believe that any issues or points of concern pertaining to the Vestal model, can be resolved in accordance with its spirit, and do not necessitate a break from it. We intend to work with others in the community in order to further refine the Vestal model, to not only make it more nuanced with respect to the current incarnations of the safety standards, but ultimately also to influence the drafting of future standards. Remote as this may seem today, it would not be that dissimilar to what happened with fixed-priority scheduling, which took many years and enormous effort by many people until it became endorsed by the standards and established industrial practice.

## References

[1] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," Comput. Surveys, vol. 50, no. 6, pp. 82:1–82:37, Nov 2017.

[2] A. Burns and R. Davis, "Mixed criticality systems – a review (11th edition)," Department of Computer Science, University of York, UK, Tech. Rep., Aug. 2018.

[3] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in 28th RTSS, 2007.

[4] "Certification authorities software team (cast), position paper (cast-32a) multicore processors," Certification authorities in North and South America, Europe, and Asia, November 2016.

[5] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," The Comp. J., vol. 29, no. 5, pp. 390–395, 1986.

[6] S. Baruah and A. Burns, "Implementing mixed criticality systems in Ada," in 16th Ada-Europe Conference, 2011, pp. 174–188.

[7] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in 24th ECRTS, July 2012, pp. 145–154.

[8] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in 24th ECRTS, July 2012, pp. 135–144.

[9] A. Esper, G. Nelissen, V. Nélis, and E. Tovar, "How realistic is the mixed-criticality real-time system model?" in 23rd RTNS, ser. RTNS '15, 2015, pp. 139–148.

[10] R. Ernst and M. D. Natale, "Mixed criticality systemsa history of misconceptions?" IEEEDesign Test, vol. 33, no. 5, pp. 65–74, Oct 2016.

[11] A. Esper, G. Nelissen, V. Nélis, and E. Tovar, "An industrial view on the common academic understanding of mixed-criticality systems," J. Real–Time Syst., vol. 54, no. 3, pp. 745–795, Jul 2018.

[12] P. Graydon and I. Bate, "Safety assurance driven problem formulation for mixed-criticality scheduling," in WMC, 2013, pp. 19–24.

[13] M. Paulitsch, O. M. Duarte, H. Karray, K. Mueller, D. Muench, and J. Nowotsch, "Mixed-criticality embedded systems – a balance ensuring partitioning and performance," in 18th DSD, Aug 2015, pp. 453–461.

[14] T. Fleming and A. Burns, "Incorporating the notion of importance into mixed criticality systems," in WMC, L. Cucu-Grosjean and R. Davis, Eds., 2014, pp. 33–38.

[15] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in 32nd RTSS, 2011, pp. 34–43.

[16] RTCA, Inc., RTCA/DO-178C. U.S. Dept. of Transportation, Federal Aviation Administration, 2012.

[17] B. Nikolić, "Mixed criticality systems – a history of misconceptions?" Invited talk. 5th Int. Workshop on Mixed Criticality Systems (WMC), Slides available at https://github.com/CPS-research-group/WMC2017/raw/master/presentations/nikolic.pptx, 2017.

[18] S. Baruah, "Mixed-criticality scheduling theory: Scope, promise, and limitations," IEEEDesign Test, vol. 35, no. 2, pp. 31–37, April 2018.

[19] D. Dasari, B. Akesson, V. Nlis, M. A. Awan, and S. M. Petters, "Identifying the sources of unpredictability in cots-based multicore systems," in 8th SIES, June 2013, pp. 39–48.

[20] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," in 19th RTAS, April 2013, pp. 55–64.

[21] G. Yao, H. Yun, Z. P. Wu, R. Pellizzoni, M. Caccamo, and L. Sha, "Schedulability analysis for memory bandwidth regulated multicore real-time systems," Trans. Computers, vol. 65, no. 2, pp. 601–614, Feb 2016.

[22] L. Sha, M. Caccamo, R. Mancuso, J.-E. Kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. Kegley, D. Perlman, G. Arundale, and R. Bradford, "Single core equivalent virtual machines for hard realtime computing on multicore processors," University of Illinois, Tech. Rep., 2014.

[23] M. A. Awan, K. Bletsas, P. F. Souto, and E. Tovar, "Semi-partitioned mixed-criticality scheduling," in 30thARCS, 2017, pp. 205–218.

[24] M. A. Awan, K. Bletsas, P. F. Souto, B. Akesson, and E. Tovar, "Mixed-Criticality Scheduling with Dynamic Redistribution of Shared Cache," in 29th ECRTS, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 76, 2017, pp. 18:1–18:21.

[25] M. A. Awan, P. Souto, K. Bletsas, B. Akesson, and E. Tovar, "Mixed-criticality scheduling with memory bandwidth regulation," in 55th DATE, March 2018.

[26] M. A. Awan, P. F. Souto, K. Bletsas, B. Akesson, and E. Tovar, "Worst-case stall analysis for multicore architectures with two memory controllers," in 30th ECRTS, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 106, 2018, pp. 2:1–2:22.

[27] M. A. Awan, K. Bletsas, P. F. Souto, B. Akesson, and E. Tovar, "Mixed-criticality scheduling with dynamic memory bandwidth regulation," in 24th RTCSA, 2018.

[28] R. Mancuso, R. Pellizzoni, N. Tokcan, and M. Caccamo, "WCET Derivation under Single Core Equivalence with Explicit Memory Budget Assignment," in 29th ECRTS, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 76. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 3:1–3:23.

[29] K. Lampka, G. Giannopoulou, R. Pellizzoni, Z. Wu, and N. Stoimenov, "A formal approach to the WCRT analysis of multicore systems with memory contention under phase-structured task sets," J. Real–Time Syst., vol. 50, no. 5, pp. 736–773, Nov 2014.

[30] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in COTS-based multi-core systems," in 20th RTAS, April 2014, pp. 145–154.

[31] M. Chisholm, B. C. Ward, N. Kim, and J. H. Anderson, "Cache sharing and isolation tradeoffs in multicore mixed-criticality systems," in 36rd RTSS, Dec 2015, pp. 305–316.

[32] M. Chisholm, N. Kim, S. Tang, N. Otterness, J. H. Anderson, F. D. Smith, and D. E. Porter, "Supporting mode changes while providing hardware isolation in mixed-criticality multicore systems," in 25th RTNS, ser. RTNS '17, 2017, pp. 58–67.

[33] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memory access control in multiprocessor for real-time systems with mixed criticality," in 24th ECRTS, 2012, pp. 299–308.

[34] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni, "Real-time cache management framework for multi-core architectures," in 19th RTAS, 2013, pp. 45–54.

[35] R. Mancuso, R. Pellizzoni, M. Caccamo, L. Sha, and H. Yun, "WCET(m) estimation in multi-core systems using single core equivalence," in 27th ECRTS, July 2015, pp. 174–183.

[36] M. Jan, L. Zaourar, and M. Pitel, "Maximizing the execution rate of low-criticality tasks in mixed criticality systems," in WMC, 2013.