Cambridge Large One

www.cambridge.org

Article-type

Multi-Model Workload Specifications and their application to Cyber-Physical Systems

Alan Burns*,1 and Sanjoy Baruah*,2

[1]The University of York, UK
[2]Washington University in St. Louis, USA

*Author for correspondence. Email: alan.burns@york.ac.uk; baruah@wustl.edu

Abstract

To address the question of how to deliver time-sensitive software for Cyber-Physical Systems (CPS) requires a range of modelling and analysis techniques to be developed and integrated. A number of these required techniques are unique to time-sensitive software where timeliness is a correctness property rather than a performance attribute. This paper focuses on how to obtain worst-case estimates of the software's execution time; in particular, it considers how workload models are derived from assumptions about the system's run-time behaviour. The specific contribution of this paper is the exploration of the notion that a system can be subject to more than one workload model. Examples illustrate how such multi-models can lead to improved schedulability, and hence more efficient CPS. An important property of the approach is that the derived analysis exhibits model-bounded behaviour. This ensures that the maximum load on the system is never higher than that implied by the individual models.

1. Introduction

The safety properties of most safety-critical Cyber-Physical Systems (CPS) must be verified before they may be deployed in the field. Since such verification occurs prior to run-time, it is typically performed upon carefully-constructed *models* of the run-time behaviour that the system is expected to exhibit. Such models are designed to emphasise the salient features of interest from the perspective of verification. In particular the verification of timing correctness properties (e.g., that deadlines are met) is usually done by the application of results from *real-time scheduling theory*. The models used in this theory make assumptions regarding the form of the workload that will need to be accommodated and the characteristics of the platform upon which such executions will occur.

The *Sporadic Task Model* is one that is commonly applied in real-time CPS. The workload is defined by a number of concurrently executing *tasks*, each of which gives rise to an unbounded sequence of *jobs*. A minimum interval of time (called the task *period*) must elapse between consecutive jobs from the same task. Every job must complete by a defined *deadline*. Once the platform is identified, the resource requirements of each task can be obtained. For example, with a single processor preemptive platform the key modelling assumption is that each job has a known *worst-case execution time* (or *WCET*) that upper-bounds it actual execution time.

During the planning and verification phases of the system's development a run-time scheduling protocol is chosen, such as Fixed-Priority (FP) scheduling. All such protocols are coupled to some form of analysis, such as Response-Time Analysis (RTA) for FP scheduling (Joseph and Pandya 1986; Audsley et al. 1993), that ensures that if all the assumptions about the application, the platform and the scheduling protocol are correct then every job of each task will complete its execution prior to its deadline.

The validity of the verification depends upon the actual workload, platform and run-time scheduler being compliant with the model assumptions. But to deliver a well engineered and energy efficient system also requires the model assumptions to be realistic, i.e. to not be excessively pessimistic. Pragmatically we also require that the associated analysis be comprehensible, and hence potentially part of a safety-case for the CPS.

The recent rapid development of Machine Learning techniques has led to the widespread use of Deep Neural Nets (DNNs) within autonomous resource-constrained CPS. One of their primary applications is to undertake classification exercises. Here a complex chain of DNNs is used to 'understand' the dynamic environment within which the CPS is operating (Razavi et al. 2022).

10.1017/cbp.2024.2

CAMBRIDGE UNIVERSITY PRESS

Many of these CPS are employed (or are been considered for employment) in safety-critical applications and require accurate predictions to be delivered in real time using limited computing resources (this is sometimes called "edge AI" where the efficient execution of machine intelligence algorithms on embedded edge devices is required (Chen and Ran 2019; Yao et al. 2017).

In this paper we aim to deliver more efficient Cyber-Physical Systems of this kind by exploring the use of *Multi-Models*: defining the behaviour required of the system by not one but a collection of integrated models. Distinct models may reflect different modes of operation of the system, different states of the environment, or different users/stakeholders of the system. The notion of integrated multi-models is a generalisation of the hierarchical models used to define *Mixed-Criticality* systems. We therefore next review this material before returning to defining multi-models and to giving examples of the benefits of their use.

2. Background

Mixed-criticality systems (MCS), widely studied in the real-time scheduling literature, provide an illustrative example of the use of multi-models for representing complex components. Each task in the task model proposed by Vestal (Vestal 2007) is characterised by multiple WCET parameter values representing different estimates, that may be trusted to different levels of assurance. Each task is also assigned a *criticality* level, which is, informally speaking, an indicator of the importance of that task to overall system correctness.

From an analysis standpoint the important property of the Vestal model is not so much the notion of criticality but the fact that the task-set under inspection has *more than one workload model* (Burns 2019; Burns and Baruah 2023). Vestal suggests that different stakeholders would want to assign different values to one of the parameters (the WCET) characterising each task: in effect there is not one but a collection of models that are being applied to the task-set, each modelling the system from a somewhat different perspective.

Since the 2007 publication of Vestal's paper there have been over 500 papers produced that have extended and utilised this notion of MCS (see Burns and Davis 2022, 2017). There have also been a number of papers that have criticised the Vestal approach (Graydon and Bate 2013; Esper et al. 2015; Paulitsch et al. 2015; Ernst and Natale 2016; Esper et al. 2018). Much of this criticism is based on different views as to the meaning of 'criticality.' However, the rich body of results that have appeared under the umbrella of MCS do not require or assign any particular meaning to the term 'criticality'; what they utilise and exploit is the idea that there is more than one interpretation of the temporal properties (i.e. model parameters) of the tasks under consideration.

Recent work has illustrated (Burns et al. 2019; Jones and Burns 2020; Burns and Jones 2022) how the run-time behaviour of a simple MCS may be specified by using Rely Conditions (Assumptions) and Guarantee Conditions (Obligations). In the Mixed-Criticality framework there is a "degraded" mode

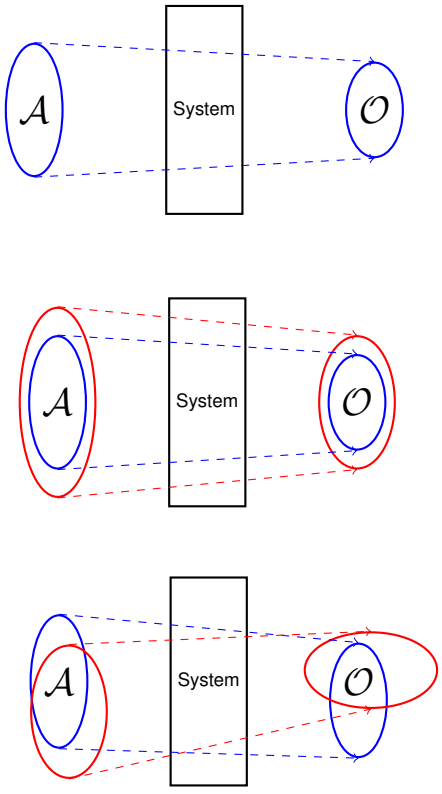

Figure 1. The top diagram depicts system execution as a mapping from a set $\mathcal{A}$ of assumed behaviors of its environment to a set $\mathcal{O}$ of system behaviors that fulfils its obligations. The middle diagram depicts a *mixed-criticality* system in which the sets of assumptions and obligations satisfy a subset/ super-set relationship. And the bottom diagram depicts the execution of multi-model systems with overlapping integrated assumptions and obligations.

with weaker Assumptions and weaker Obligations into which the system will transition following an 'over-run' fault (i.e. a task executing for more than its assumed WCET). In this degraded mode only the higher-criticality jobs are guaranteed to meet their deadlines. *This is an example of a hierarchical multi-model.*

It is sometimes convenient to interpret assumption-obligation specifications in terms of *mappings*. Under such an interpretation, the assumptions specify the set of all behaviors of the environment for which the system is expected to behave correctly; the obligations specify the corresponding correct system behaviours. Then *correct system execution maps each assumed behaviour of the environment to some correct system behaviour* – see the top diagram of Figure 1. The middle diagram in this Figure depicts a MCS with a hierarchical relationship between the assumptions and obligations. The bottom diagram generalises this relationship; there are overlapping sets of assumptions leading to overlapping obligations. In both of these situations, correct behaviour of the system requires at least one of the sets of assumptions to remain true.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

2.1 Analysis of MCS Multi-Models

Vestal's Mixed-Critically (MC) model, and much of the subsequent literature on mixed criticality, is based on the Sporadic Task Model. With this model each task, τ_i is defined by 4 parameters: C_i – WCET, T_i – Period (the minimum time between successive job releases from the same task), D_i – Deadline (the relative deadline of the task, any job released at time t must compete before $t + D_i$) and L_i the criticality level of the task.

As discussed above, each task has a WCET parameter for each criticality level and these values increase as one moves up the criticality ladder. Each task and the complete CPS itself is defined over a sequence of criticality modes. If at some time t the system is in mode L then only tasks of criticality level L or higher need to be guaranteed to execute correctly. And these guarantees must be made based on the assumption that all tasks can execute up to, but not beyond, the WCET value associated with L.

Priority assignment is crucially important in fixed-priority scheduling. For task systems that have relative deadline equal to period the rate monotonic priority assignment (RMPA) scheme is optimal (i.e. will lead to a schedulable task set if there exists any feasible priority ordering).[1] With RMPA priorities are ordered according to period (T) – the shorter the period the higher the priority. When relative deadlines (D) are shorter than period then the deadline monotonic assignment (DMPA) scheme is optimal. Here relative deadlines are used to order the priorities – the shorter the relative deadline the higher the priority. The final case to cover is when some of the tasks have relative deadlines that are larger than their periods. These tasks will on average finish before they are re-released, but in the worst case jobs from the same task can overlap. With these arbitrary deadlines neither RMPA or DMPA is optimal, but fortunately an optimal assignment algorithm does exist - this is known as Audsley's Algorithm (Audsley 1990).

The standard way of analysing a system to determine that all deadlines will be met is to apply Response-Time Analysis (RTA) (Joseph and Pandya 1986; Audsley et al. 1993). As the name implies, this involves computing (see eq(1)) the longest completion time (response time, R_i) for each task and checking that this is not greater than the task's relative deadline (i.e ensuring that $R_i \leq D_i, \forall i$).

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

where $\mathbf{hp}(i)$ is the set of tasks that have a higher priority than τ_i. This equation is solved using the standard techniques for solving recurrence relations (i.e. fixed point iteration).

With a MC model Audsley's optimal priority assignment algorithm can again be applied to maximise schedulability. In effect the algorithm delivers a priority ordering that is adequate (i.e. leads to a schedulable system) but also maximises the priorities of the higher criticality tasks. RTA can also be adapted to test for the schedulability of a MCS (Baruah, Burns, and Davis 2011). Unfortunately this adaptation is not straightforward. As a system transfers from one criticality level to the next a number of tasks will have their WCET budgets increased, whilst others will no longer be supported. This complex transition can temporarily increase the load on the system. As a result, analysis needs to be developed that can accommodate this criticality mode change. Many forms of such analysis have been published (see the following review papers for coverage of the many contributions in this area: Burns and Davis 2022, 2017; Guo and Baruah 2017; Arbaud, Juhász, and Jantsch 2018; Yoon et al. 2018; Althebeiti 2020; Cinque et al. 2022; Chai et al. 2019) a number of which have been later shown to contain errors. They differ in the assumptions made, the tightness of the analysis, and the level of degraded service allowed for those tasks that are no longer guaranteed.

One of the objectives of the approach developed in this paper is to remove the need for this complex mode change analysis.

3. Multi Sporadic Task Model

We consider a multi-model specification approach to be very general, and applicable to modelling a variety of different situations, with the different models accorded different interpretations. Here are some examples.

Different Environmental Conditions. A CPS that is intended to operate in several different environmental conditions may be expected to behave differently under these different conditions. Assumptions concerning different environmental conditions will give rise to different workload models even if the Obligations are the same for all valid assumptions.

Different Stakeholders. It may sometimes be the case that rather than developing individual bespoke systems for several different stakeholders, it is more efficient to develop a single CPS that is capable of meeting all their needs. Here the Assumptions may be the same but the Obligations differ.

Different Levels of Service. With a system that is defined to be resilient to faults and partial failures, different assumptions may relate to different fault models, with the corresponding obligations defining degraded (fault tolerant) levels of service.

Illustrations of these applications of the multi-model idea are presented below.

3.1 Types of Multi-Model

Generalising from the representation of MCS as multi-models, three forms of relationship between the individual models within a Multi-Model framework have been defined (Burns and Baruah 2023):

1. Issues concerned with priority assignment and response-time analysis are covered in standard textbooks such as Buttazzo 2005; Burns and Wellings 2016.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

1. independent multi-models – all Assumptions hold at all times and all Obligations must always be satisfied.
2. integrated multi-models – at least one set of Assumptions-Obligations pairings must always be satisfied.
3. hierarchical multi-models – at least one set of Assumption-Obligations pairings must always be satisfied, and there is a hierarchical relationship between both the Assumptions and the corresponding Obligations.

Independent multi-models are straightforward as all Obligation must always be satisfied. Hierarchical Multi-models are well illustrated by the extensive literature on MCSs. The remainder of this paper concentrates on integrated multi-models. As the purpose of this contribution is to demonstrate the usefulness of adopting an integrated multi-model approach, we introduce the key notions of the multi-model idea via a running example that employs a generalisation of the Sporadic Task Model that was introduced in Section 1.

3.2 Integrated Multi-Models

Consider a CPS that performs some form of classification. There are various sensors that are polled by three tasks executing at three different rates/periods. A single processor is employed and the tasks are scheduled preemptively using the Fixed Priority (FP) scheme with priorities assigned optimally using DMPA.

The work to be undertaken by each of these three tasks is dependent upon the assumptions made about the conditions and constraints of the environment.

As a simple illustrative example consider a system that outputs the number and breed of a group of Cats and Dogs in a sensed environment.[2] A simple narrative is used to illustrate the main features of the approach and to clearly demonstrate the benefits that can be gained from its adoption.

To bound the work that needs to be undertaken there must be a bound on the number of pets (Cats and Dogs). Let us suppose that the use-case for this system additionally notes that there can be a few Cats and a lot of Dogs or a lot of Cats and few Dogs, but never a significant number of both. This leads to two distinct sets of assumptions ($\mathcal{A}1$ and $\mathcal{A}2$) about the worst-case run-time conditions of the environment. Each Assumption is a boolean predicate; so, for example:

$$\mathcal{A}1 \stackrel{\text{def}}{=} N_{Dogs} \leq 7 \wedge N_{Cats} \leq 2$$

$$\mathcal{A}2 \stackrel{\text{def}}{=} N_{Cats} \leq 6 \wedge N_{Dogs} \leq 1$$

The environment can be relied upon to behave according to these assumptions: they form part of the system's specification. At least one set of assumptions will be true at all times.

The first four columns in Table 1 defines the workload model for when $\mathcal{A}1$ applies:

Table 1. Workload model for when $\mathcal{A}1$ applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	2	3
τ_d	14	14	7	14

- The first task, τ_p, manages the sensors and undertakes other necessary computations. It has the shortest period and a fixed worst-case execution time across all assumptions about the load on the system.
- Task τ_c is primarily responsible for the classification of Cats and has a period[3] of 10, a deadline of 10 for each released job and a worst-case execution time, given $\mathcal{A}1$, of 2 (to reflect that there are at most 2 Cats).
- Task τ_d is concerned with the classification of Dogs. It has a longer period of 14 (to reflect some temporal difference between the sensed objects) and a worst-case execution time of 7 (as that is the maximum number of Dogs when $\mathcal{A}1$ applies).

  Both τ_c and τ_d are structured to first take in data about the environment, then process this data and finally to output the results of this processing.

The tasks in Table 1 (and all subsequent tables) are presented in priority order, so τ_p has the highest priority and τ_d the lowest (as determined by DMPA).

The final column in Table 1 is the result obtained by applying Response-Time Analysis (RTA) to this workload model. To be schedulable the worst case response time for each task must be no greater than the specified (relative) deadline. The response-times contained in Table 1 are obtain from the application of eq(1). For example, the first estimate of R_d, is 7. Putting 7 into the RHS of eq(1) delivers 11 (7+2+2). Continuing with 11 delivers 14 (7+3+4). Repeating with the input of 14 results in 14, and hence 14 is the worst-case response time for τ_d.

Clearly the system is schedulable (all job deadlines for all tasks will be satisfied) under the assumption that the system is executing within an environment defined by $\mathcal{A}1$. The fact that the deadline of τ_d is 14 and the worst-case response time is also 14 shows that the task set is on the cusp of being unschedulable.

A similar workload model can be derived for when $\mathcal{A}2$ applies – this is given in Table 2. Again the task set is deemed to be schedulable.

Table 2. Workload model for when $\mathcal{A}2$ applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	6	8
τ_d	14	14	1	9

2. This toy example is loosely based on an *Identify Friend or Foe* (IFF) (Bowden 1985) application system that uses DNN-based image processing (for example Gupta, Pooja, and Kakde 2023) to distinguish between friendly and hostile aircraft, and may further classify each kind.

3. As is usual in papers on scheduling, the time units for the example parameters are not given; all that is necessary for an example to be meaningful is that all the parameters are in the same units.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

We now have two models for the same system. Appropriate analysis for both models shows that each leads to an implementation that will meet all deadlines [4]. The assumptions for the two models must of course be valid, as must the predicate that asserts that at least one set of assumptions is always true.

If this multi-model approach was not adopted then one would need to employ the traditional method of using a single model that captures the worst-case behaviour of each task. The assumptions for the single model ($\mathcal{A}$) must capture all possible behaviours, so

$$\mathcal{A} \stackrel{\text{def}}{=} N_{Dogs} \leq 7 \wedge N_{Cats} \leq 6$$

This leads to the workload model in Table 3.

Table 3. Single Workload model for when $\mathcal{A}$ applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	6	8
τ_d	14	14	7	∞

The table shows that the worst-case response time of the lowest priority task is unbounded, and hence cannot be less than the finite deadline defined for the task – the task set is unschedulable. (This could also be observed by noting that the tasks' utilisation –WCET/Period– sum to greater than one (i.e. 1/5 + 6/10 + 7/14 = 1.3) – which for a single CPU platform is clearly infeasible.)

To cope with this inability to schedule the assumed worst-case workload the hardware platform would need to be upgraded. Either a faster processor would need to be used or a second processor added. Both would have cost and performance implications. The single set of assumptions ($\mathcal{A}$) could now be accommodated, but at run-time the extra capabilities would never be required: the environment would never produce the scenario of 7 Dogs and 6 Cats. The two-model specification allows the real worst-case behaviours to be represented with the result that the extra hardware is not required.

At run-time the system's initial behaviour will, typically, be compatible with both models. Hence both sets of assumptions are true, and there is again a single model with a basic set of assumptions, $\mathcal{A}0$:

$$\mathcal{A}0 \stackrel{\text{def}}{=} \mathcal{A}1 \wedge \mathcal{A}2$$

i.e.

$$\mathcal{A}0 \stackrel{\text{def}}{=} N_{Dogs} \leq 1 \wedge N_{Cats} \leq 2$$

Table 4 defines an upper bound on the workload model that sustains $\mathcal{A}0$. With these lower estimates for WCET the RTA computes worst-case response times that easily satisfy the deadlines. As long as the assumptions represented by this minimal workload are true we say that the system's environment is in state $S0$[5]. If $\mathcal{A}1$ applies, but not $\mathcal{A}0$ (and hence not $\mathcal{A}2$ either) then the system is in state $S1$. Similarly if $\mathcal{A}2$ applies, but not $\mathcal{A}0$ (or $\mathcal{A}1$) then the system is in state $S2$. If follows that the system must always be in exactly one of these three states (as depited in Figure 2).

Table 4. Workload model for when $\mathcal{A}0$ applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	2	3
τ_d	14	14	1	4

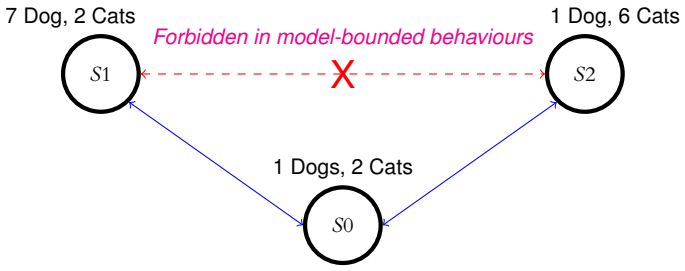

Figure 2. The three states of the example system, together with the maximum number of Dogs and Cats allowed in each state.

As long as each job completes before it has executed for the WCET bounds given in Table 4 (i.e. 1, 2 and 1 respectively for the three tasks) then the system will remain in state $S0$. This corresponds to the environment containing at most two Cats and one Dog. However, if a job from the third task executes for more than 1 (because, say, a second Dog is now present) then the system will move from state $S0$ to state $S1$. The assumptions, $\mathcal{A}1$, remain true, and the analysis for $S1$ will guarantee that all deadlines are met. Assumptions $\mathcal{A}2$ are no longer valid. Task τ_d although executing for more than 1 will not execute for more than 7 since $\mathcal{A}1$ applies.

Alternatively, whilst in $S0$ if a job from the second task executes for more than 2 time units then the system will move to $S2$. Assumptions $\mathcal{A}2$ hold, but $\mathcal{A}1$ do not. All deadlines are still met.

For either of the above transitions, we have moved from behaviour sanctioned by two sets of assumptions, to behaviour sanctioned by just one. With this simple example this is as far as the system can move. There are no further models, no fault tolerance and hence it must be assumed that while in $S1$ assumptions $\mathcal{A}1$ will require the three tasks to have execution times bounded by the values 1, 2 and 7. And similarly while in state $S2$ assumptions $\mathcal{A}2$ hold and bound the execution times to 1, 6 and 7.

In each state there is a maximum number of Cats and Dogs allowed. Once in, for example, state $S1$ the number of Dogs can vary between 2 and 7. If they ever drop to 1 then this

4. But note issues raised in Section 3.1.

5. The three states and the associated transitions (see Figure 2) are only introduced to clarify the behaviour of a multi-model system; they play no part in the run-time behaviour of the system.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

denotes a state change to $S0$. There can also be 0, 1 or 2 Cats in $S1$; changes to this number cannot cause a state change. Assumptions $\mathcal{A}1$ ensures that a third Cat cannot appear while in $S1$ (or $S0$).

With the above example we have a system defined by two models. These two models share a sub-model that defines when both models apply. This leads to the possibility, in a system with an extended life, that at different times different models will apply. For example, the environment could at some point consist of mainly Dogs, but later after a period with few, or no, animals a predominance of Cats could occur. In terms of applicable assumptions, the system moves from $\mathcal{A}0$ to $\mathcal{A}1$, back to $\mathcal{A}0$ and then to $\mathcal{A}2$. For reasons that are explained below we do not want to allow behaviour that is equivalent to a move directly from $S1$ to $S2$ or $S2$ to $S1$.

3.3 Model-Bounded Behaviour

To obtain the most schedulability benefit from the multi-model approach we require that it is sufficient for each workload model to be schedulable. We do not want a rapid transition from, say, $S2$ to $S1$, to place a load on the system that is higher than that induced by either model.

For example, if there was a rapid movement from an environment with 6 Cats and 1 Dog to one with 6 Dogs and 1 Cat then τ_d could suffer interference of 6 from τ_c and then have to execute for 6 itself. With τ_p executing for 1 in every 5, this means that τ_d may have a response time of over 14 and hence miss its deadline – see Figure 3.

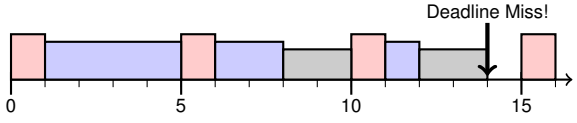

Figure 3. The three tasks τ_p (red), τ_c (blue) and τ_d (grey) are released at time 0. At time 1 task τ_c executes upon an input with 6 Cats. The environment causes a switch of models somewhere within the interval $[2, 8]$. At time 8 task τ_d executes upon an input with 6 Dogs. At time 14 its deadline elapses but it has only executed for 4 of its required 6 time units.

This rapid movement from one model to the other produces a scenario that has worse temporal behaviour than either of the individual models.

In Section 2.1 it was noted that mode changes in Mixed Criticality Systems suffer from this problem. A system can be schedulable in both the LO-criticality and HI-criticality modes but not during the transition between modes. More general analysis of mode change protocols (Sha et al. 1989; Burns and Quiggle 1990; Tindell, Burns, and Wellings 1992; Pedro and Burns 1998; Tindell and Alonso 1996; Emberson and Bate 2007; Real and Crespo 2004; Azim and Fischmeister 2016; Henzinger, Horowitz, and Kirsch 2001) also have to develop analysis that can address the worst-case mode change behaviour. However, in this paper we are addressing multi-model specifications, not multi-modal behaviours. In the latter it is reasonable for a system to transition from one mode to another (for example when a fault causes a task to overrun), but individual models focus on allowable extreme behaviours; for instance in the above example either an extreme number of Dogs, or of Cats, but not both. It is reasonable to constrain the multi-model framework so that transitions between when one model applies to when another model takes over do not result in temporal behaviour that is more extreme than that defined by the individual models. We refer to this as *model-bounded behaviour* (**MBB**).

As well as delivering an improvement in schedulability, model-bounded behaviour also has the advantage that the analysis of a CPS defined by a multi-model is no more complicated than when only one model is applied. The single model analysis is simply applied separately to each of the models in the multi-model specification. Complex scheduling theory and analysis, such as that employed in the mode change papers referenced earlier, presents a barrier to deployment in CPS that have a safety dimension: there is no evidence that safety cases have included arguments for timing correctness of more sophistication than that implied by the simple Sporadic Task Model. With model-bounded behaviour the required safety arguments are essentially the same.

3.4 Sufficient Analysis for Model-Bounded Behaviour

There are a number of ways of ensuring that a specific application of a multi-model specification has model-bounded behaviour (MBB). We could, for example, require that the environment returns to the conditions in which the basic assumptions apply between any transition amongst other specific models. So a transition between there being 6 Cats to 6 or even 7 Dogs must go via a phase during which there are at most 2 Cats and 1 Dog. In this section we develop sufficient analysis for MBB. More exact analysis may be possible; here we wish to demonstrate that there is a straightforward method for checking that an assumption of MBB is valid. To explain the form that this analysis takes we again structure the narrative around the simple example introduced above.

First we need to capture *the maximum rate of change* that the environment can impose on the CPS. This will be represented by the parameter T_E – the minimum time between changes to either the number of Cats or the number of Dogs. So if $T_E = 3$ then it will take at least 18 time units to move from being in state $S1$ with 7 Dogs to transitioning to $S0$ with just one Dog. (Here for ease of presentation we will use a single 'rate of charge' parameter; an extension to a model in which Dogs and Cats each have their own T_E parameter is straightforward.)

To prevent interference between states $S1$ and $S2$ (i.e. to prevent the analysis of the tasks in state $S2$ being influenced by the behaviour of the tasks in $S1$, and visa versa) it is sufficient to prove that any round of computation started in $S1$ ($S2$, respectively) must complete before the environment could transition the system to $S2$ ($S1$, resp.). (By 'round of computation' we mean the execution of a job from τ_d and a job from τ_c, along with the necessary executions of τ_p.) This is also known as the system's maximum *busy period*, with the end of the busy period being called an *idle instant*.

With fixed-priority scheduling the system's busy period

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

is exactly the response-time of the lowest priority task; in our example this is task τ_d. And it is a well-known property of preemptively-scheduled sporadic task systems that no behaviour prior to an idle instant can impact on, or influence in any way, the subsequent behaviour of the system.

So, for example, if the system is in $S1$ with 7 Dogs and 2 Cats then $\mathcal{A}1$ applies and Table 1 shows that the worst-case response-time of task τ_d is 14. Hence if $T_E = 4$ then in 14 time units the environment could have altered by a maximum of $\lceil 14/4 \rceil = 4$ changes; i.e. a reduction from 7 to 3 Dogs. This implies that the system will remain in $S1$; the analysis of $S2$ will not be effected by the behaviour of the tasks in $S1$. We potentially have model-bounded behaviour.

However, showing that we have the desirable behaviour when there is the maximum load on the system is not necessarily sufficient. If there are fewer Dogs then there is less work to do, but fewer steps are needed to move from $S1$ to $S0$ and then $S2$. In general it is, unfortunately, not possible to predetermine the worst-case starting condition of the number of Dogs, and hence all numbers from 2 to 7 must be examined.

To have model-bounded behaviour we require that regardless of the starting condition within $S1$ there must be an idle instant before the environment could have moved into state $S2$. Let $R_d(n)$ denote the response-time of task τ_d when the system is in state $S1$ and there are n Dogs. This value is easily obtained from eq(1) by using n rather than 7 as computation load C_d and assuming that there are the maximum number of Cats (i.e. $C_c = 2$). If the number of steps is equal to (or greater than) n then the penultimate step would have reduced the number of Dogs to 1 and hence a state transition from $S1$ to $S0$; then the final step could be an increase in the number of Cats (from 2 to 3) and hence a transition to $S2$. This breaks our rule that there must be an idle instant before $S2$ can be reached. To deliver model-bounded behaviour we require that the idle instant occurs no later then the time at which the transition to $S2$ occurs. I.e., we require that the number of steps necessary to transition to state $S2$ is greater than the maximum number of steps that the environment can accomplish in the time it takes to process n dogs; that is:

$$\forall n \in 2..7 : \left(n > \left\lceil \frac{R_d(n)}{T_E} \right\rceil \right) \tag{2}$$

Table 5. Example values from eq(2) for $T_E = 4$ and $T_E = 5$.

n	$R_d(n)$	$\lceil \frac{R_d(n)}{4} \rceil$	*Passed*	$\lceil \frac{R_d(n)}{5} \rceil$	*Passed*
7	14	4	Y	3	*Y*
6	10	3	Y	2	*Y*
5	9	3	Y	2	*Y*
4	8	2	Y	2	*Y*
3	7	2	Y	2	*Y*
2	5	2	N	1	*Y*

Table 5 contains the results from applying eq(2) for two values of T_E, namely 4 and 5. For $T_E = 5$, all values of n are acceptable. The most critical point is when $n = 2$, the minimum number of Dogs allowed in $S1$. Within the response time of 5 the environment could reduce the number of Dogs to 1 and hence the state would change to $S0$. But a further state change (if sanctioned) would move the environment to $S2$. To ensure that there is an idle instant before this step it is necessary that $T_E \geq 5$. If this is the case then we can assert model-bounded behaviour; if not then it is undecided as we have only derived a sufficient test.

To complete the analysis we also need to consider $S2$. Let m stand for the number of Cats. It can take on the values 3, 4, 5 and 6. As $S0$ contains two Cats eq(2) becomes:

$$\forall m \in 3..6 : \left(m > \left\lceil \frac{R_c(m)}{T_E} \right\rceil + 1 \right) \tag{3}$$

Table 6 has the results for $T_E = 5$, showing that this remains an acceptable assumption when initially at state $S2$:

Table 6. Example values for analysis of $S2$.

m	$R_c(m)$	$\lceil \frac{R_c(m)}{5} \rceil + 1$	*Passed*
6	8	3	*Y*
5	7	3	*Y*
4	5	2	*Y*
3	4	2	*Y*

An examination of the predicate that is represented by eq(2) shows that if a value of T_E leads to model-bounded behaviour then all larger values will also lead to this behaviour. This is an example of *sustainable* scheduling analysis (Baruah and Burns 2006). It also follows that if the system is schedulable and all tasks have deadlines no greater than their periods then if the environment is changing at a slower rate than the slowest task then the longest task response time will be shorter than T_E and hence at most a single change to the environment could have occurred before the important idle instant. To move from $S1$ to $S2$ requires a minimum of two steps and hence model-bounded behaviour. Even if the example was altered so that $S0$ was a maximum of zero Cats and zero Dogs it would still take two steps to move from the domain sanctioned by one model to the domain sanctioned by the other model. And with the forced idle instant within these two steps we have model-bounded behaviour. It follows that for any example, and any number of models, there is *a simple sufficient test for MBB*: $T_E > \max(T_i)$.

To summarise the results presented in this section; we have used an example with two workload models to show how model-bounded behaviour can be checked for. This behaviour allows the schedulability of the whole system to follow directly from the schedulability of each model. To assert that the whole system has model-bounded behaviour it is necessary to match the rate of change of the environment with the periods of the tasks that make up the CPS. In the usual case where the environment is changing at a much slower rate than the system's tasks then the system always has model-bounded behaviour. In other cases the sufficient test developed above allows an effective and straightforward check to

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

be made.

Future work will consider if a tractable more exact test can be developed. We will also allow different aspects of the environment to have different maximum rates of change (for example Dog numbers changing quicker than Cat numbers). The latter should be an easy extension to incorporate.

3.5 Multi Stakeholder Example

In the above example the existence of two models follows directly from there being two sets of assumptions. The behaviour of the environment allows these two models to exist and ensures that, in the absence of faults, at least one of them defines the current workload on the system. There are however other scenarios that can benefit from multi-model specifications. One of which is when there is only a single set of assumptions but more than one stakeholder - with each stakeholder requiring different aspects of the computation to be undertaken, and hence give rise to different estimates of worst-case execution times. Here the obligations are distinct although they may overlap.

We adapt the above example so that there is just one set of assumptions, $\mathcal{A}$, but two stakeholders, *SKD* and *SKC*. The first stakeholder is only interested in the breeds of the Dogs; the second is only interested in the breeds of the Cats.

$$\mathcal{A} \stackrel{\text{def}}{=} N_{Cats} \leq 5 \wedge N_{Dogs} \leq 5$$

Initially we assume that there is just one stakeholder, *SKP* that is interested in the breeds of both sets of Pets. Table 7 gives the WCET and Response Times for this situation. Again the lowest priority task is unschedulable.

Table 7. Workload model for when $\mathcal{A}$ and *SKP* applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	5	7
τ_d	14	14	5	∞

The individual stakeholders however require less work to be done. For example task τ_c has less work to do when *SKD* applies. Tables 8 and 9 give the workload levels applicable for the two stakeholders. Both models are schedulable.

Table 8. Workload model for when $\mathcal{A}$ and *SKD* applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	1	2
τ_d	14	14	5	8

When the multi-model specification comes from distinct stakeholders then it is obviously necessary that only one stakeholder can be active at any specific time[6]. Again, a long running system may change stakeholders but it must be the case that the change occurs after a period of dormancy in which there are either no Dogs or Cats to classify or the classification action is inhibited. The latter being equivalent to there being no stakeholder for a short period of time. They could be represented by the workload model in Table 10.

Table 9. Workload model for when $\mathcal{A}$ and *SKC* applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	5	7
τ_d	14	14	1	8

Table 10. Workload model for when $\mathcal{A}$ applies with no Stakeholder.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	1	2
τ_d	14	14	1	3

Multiple stakeholders can of course be combined with multiple environmental conditions. For example, Table 11 considers the scenario that the original set of assumptions $\mathcal{A}1$ and stakeholder *SKD* apply. Note the introduction of a more constraining stakeholder leads to the response-time of τ_d being reduced from 14 to 10. If the deadline of τ_d was, say, 12 rather than 14 then this difference in response time would make the difference between being schedulable or not.

Table 11. Workload model for when $\mathcal{A}1$ and *SKD* applies.

Task	*Period*	*Deadline*	WCET	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	1	3
τ_d	14	14	7	10

3.6 Mixed-Criticality Revisited

As discussed in Section 2 one of the motivations for defining the multi-model approach is to generalise the modelling work that has been applied in the Mixed Criticality (MC) domain. With a MC specification there is usually a single set of assumptions and a single stakeholder, but more than one way of characterising the single model. As the characterisations have the same effects on all the relevant parameters (e.g. all WCET values increase when one moves from a LO-criticality model to a HI-criticality model) then the multi-model is hierarchical. It is also clear that to compensate for the scheduling parameter getting universally worse, the workload must be reduced in some other way – this is typically done by removing some tasks from the set that must be declared to be schedulable (Vestal 2007).

In our running example, assume there are two estimates of WCET for each task, one derived from measurements (WCET-M) and applicable to mission-critical systems, the other de-

6. If both stakeholders must be accommodated then this would be an example of an independent multi-model; as both models must be satisfied then this is equivalent to there being the single stakeholder *SKP* – and in this example the workload cannot be scheduled.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

rived from static analysis (WCET-A) and applicable to safety-critical systems. To exploit the same example, assume that we have the following definition of the system's assumptions (one set):

$$\mathcal{A} \stackrel{\text{def}}{=} N_{Cats} \leq 4 \wedge N_{Dogs} \leq 4$$

and that the classification of Cats is safety-critical and Dogs is mission-critical. The two models now apply to mission critical and safety-critical behaviour – see Tables 12 and 13. Both models deliver run-time schedulability.

Table 12. Workload model for mission-critical behaviour.

Task	*Period*	*Deadline*	WCET-M	*ResponseTime*
τ_p	5	3	1	1
τ_c	10	10	4	5
τ_d	14	14	4	10

Table 13. Workload model for safety-critical behaviour.

Task	*Period*	*Deadline*	WCET-A	*ResponseTime*
τ_p	5	3	1.5	1.5
τ_c	10	10	7	10

When the safety-critical model applies, then task τ_d no longer generates work that needs to be scheduled and it could therefore be aborted (or execute in the background). It does not interfere with the crucial tasks and its performance, if it does execute, then it could be assigned a quality of service, rather than a correctness, attribute.

As the transition from mission-critical to safety-critical behaviour occurs during the application's execution; e.g. when τ_c executes for more than 4 without completing, it is not possible to guarantee model-bounded behaviour. Much of the analysis associated with Mixed-Criticality systems is concerned with proving that deadlines will continue to be met during these transitions. By requiring model-bounded behaviour for the general integrated multi-model framework we have removed the need to derive this analysis.

3.7 Summary

The toy example used in this narrative has illustrated how a CPS that is unschedulable if a single model is employed can correctly be identified as schedulable if a two-model specification is used. The efficacy of this two-model specification comes from the existence of two different set of assumptions about the workload of the system. Examples have also been given of different models reflecting different stakeholders.

Although only two-model examples were used, the approach naturally extends to multiple models. It is also possible to combine integrated multi-models and hierarchical ones. To illustrate this consider the assumptions represented by $\mathcal{A}1$. i.e $N_{Dogs} \leq 7 \wedge N_{Cats} \leq 2$. The associated obligations are that the breeds of these Dogs and Cats must be classified. We could add a degraded model, $\mathcal{A}1^*$, that allows up to, say, 10 Dogs (and 2 Cats) but after 7 the number of Dogs is counted but the breed is not computed. This is an example of a hierarchical relationship as clearly $\mathcal{A}1 \Rightarrow \mathcal{A}1^*$.

The integrated multi-model approach defined in this paper incorporates a non-hierarchical relationship between the models. It also requires that the defined models, the dynamics of the environment, the run-time scheduler and the associated analysis collectively deliver model-bounded behaviour. This ensures that if each model is deemed to be schedulable then so is the complete system and all its allowable run-time state changes.

As CPS become more general purpose and the environments in which they operate become more complicated and multi-faceted there is an increasing need to be more precise about the assumptions and obligations defining the system's workload. The use of more than one workload model is a natural way of providing this precision.

4. Conclusion

The specification of all CPS software must capture the assumptions made about the environment in which the CPS will operate. For time-sensitive software this must include estimates of the maximum workload that the CPS can experience. During system development the validity of these assumptions can be relied upon. However, a well engineered CPS (e.g. small footprint, low power consumption etc.) will require that assumptions about workload are realistic as well as correct. The assumptions must not include infeasible scenarios that, if catered for, will lead to over engineered inefficient solutions. Moreover, the analysis of the temporal properties of CPS control software must be proved correct using techniques in which high confidence can be placed. The complex models and analysis often found in published research papers do not readily translate into industrial practice.

In this paper we have developed the notion of multi-model specifications to allow workload assumptions to more precisely represent the conditions under which the deployed CPS must operate. Different environmental conditions, stakeholders and levels of service give rise to different sets of assumptions and obligations, and therefore different workload models. To collapse all of these diverse situations into a single model is unnecessary and leads to over-specified systems. By directly representing different set of assumptions and obligations as distinct models, we have enabled the efficient implementation of time-sensitive software for CPS. As part of this argument in favour of a multi-model approach the paper has shown how the commonly used Sporadic Task Model, preemptive fixed-priority scheduling, and response-time analysis can be applied directly to integrated multi-model specifications.

Although this paper has focused on timing issues and has applied the multi-model idea to workload models of time-sensitive software, the approach can clearly be generalised to apply to other properties of CPS. For example, power consumption analysis could be developed using multi-models. For fault-tolerant systems, different assumptions about the hostility of the environment could lead to multiple fault models. Al-

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press

though an initial approach here is likely to be hierarchical (following the mixed-criticality schemes) a more comprehensive approach with many sources of disturbance is likely to benefit from the more general integrated multi-model framework. Whether different levels of security and sources of intrusion can benefit from multi-modeling is an open issue.

Funding Statement This research was funded in part by Innovate UK HICLASS project (113213), and the US National Science Foundation (Grants CNS-2141256 and CPS-2229290).

Competing Interests None.

Connections Reference Lee EA, Woodcock J. Time- Sensitive Software. Research Directions: Cyber-Physical Systems. 2023;1:e1. doi:10.1017/cbp.2023.1

References

Althebeiti, H. 2020. *Research review on mixed-criticality scheduling.* Technical Report. University of Central Florida, STARS.

Arbaud, R., D. Juhász, and A. Jantsch. 2018. *Resource management for mixed-criticality systems on multi-core platforms with focus on communication.* Technical report. ResearchGate.

Audsley, N.C. 1990. *Optimal priority assignment and feasibility of static priority tasks with arbitrary start times.* Technical report YCS164. Department of Computer Science, University of York.

Audsley, N.C., A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. 1993. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal* 8 (5): 284–292.

Azim, A., and S. Fischmeister. 2016. Efficient mode changes in multi-mode systems. In *Proc. computer design (iccd),* 592–599. IEEE.

Baruah, S.K., and A. Burns. 2006. Sustainable schedulability analysis. In *Proc. of ieee real-time systems symposium (rtss),* 159–168.

Baruah, S.K., A. Burns, and R.I. Davis. 2011. Response-time analysis for mixed criticality systems. In *Proc. ieee real-time systems symposium (rtss),* 34–43.

Bowden, Lord. 1985. The story of IFF (identification friend or foe). *IEE Proceedings A (Physical Science, Measurement and Instrumentation, Management and Education, Reviews)* 132 (6): 435–437.

Burns, A. 2019. Multi-model systems – an MCS by any other name. In *Proc. 7th int. rtss workshop on mixed criticality systems (wmc),* 5–8.

Burns, A., and S. Baruah. 2023. Multi-model specifications and their application to classification systems, in *Proc. 31th international conference on real-time networks and systems.* RTNS 2023. ACM.

Burns, A., S. Baruah, C.B. Jones, and I. Bate. 2019. Reasoning about the relationship between the scheduler and mixed-criticality jobs. In *Proc. 7th int. rtss workshop on mixed criticality systems (wmc),* 17–22.

Burns, A., and R.I. Davis. 2017. A survey of research into mixed criticality systems. *ACM Computer Surveys* 50 (6): 1–37.

———. 2022. *Mixed criticality systems: a review (13th edition).* White Rose Repository: https://eprints.whiterose.ac.uk/183619/, MCC-1(13). Department of Computer Science, University of York.

Burns, A., and C.B. Jones. 2022. An approach to formally specifying the behaviour of mixed-criticality systems. In *Proc. 34th euromicro conference on real-time systems (ecrts),* edited by Martina Maggio, vol. 231, 14:1–14:23. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Burns, A., and T.J. Quiggle. 1990. Effective use of abort in programming mode changes. *Ada Letters.*

Burns, A., and A. J. Wellings. 2016. *Analysable real-time systems programmed in ada.* ISBN: 9781530265503.

Buttazzo, G.C. 2005. *Hard Real-Time Computing Systems.* Springer.

Chai, H., G. Zhang, J. Sun, A. Vajdi, J. Hua, and J. Zhou. 2019. A review of recent techniques in mixed-criticality systems. *Journal of Circuits, Systems and Computers* 28 (07): 1930007.

Chen, J., and X. Ran. 2019. Deep learning with edge computing: a review. *Proc. IEEE* 107 (8): 1655–1674.

Cinque, M., D. Cotroneo, L. De Simone, and S. Rosiello. 2022. Virtualizing mixed-criticality systems: a survey on industrial trends and issues. *Future Generation Computer Systems* 129:315–330. ISSN: 0167-739X.

Emberson, P., and I. Bate. 2007. Minimising task migrations and priority changes in mode transitions. In *Proc. of the 13th ieee real-time and embedded technology and applications symposium (rtas 07),* 158–167.

Ernst, R., and M. Di Natale. 2016. Mixed criticality systems?a history of misconceptions? *IEEE Design & Test* 33 (5): 65–74.

Esper, A., G. Neilissen, V. Neils, and E. Tovar. 2015. How realistic is the mixed-criticality real-time system model. In *23rd international conference on real-time networks and systems (rtns 2015),* 139–148.

Esper, A., G. Nelissen, V. Nélis, and E. Tovar. 2018. An industrial view on the common academic understanding of mixed-criticality systems. *Real-Time Systems* 54 (3): 745–795.

Graydon, P., and I. Bate. 2013. Safety assurance driven problem formulation for mixed-criticality scheduling. In *Proc. wmc, rtss,* 19–24.

Guo, Z., and S. Baruah. 2017. Mixed-criticality real-time systems. In *Cyber-physical systems: a reference,* edited by X. Wang. Springer, Berlin, Heidelberg.

Gupta, P., J. Pooja, and O.G. Kakde. 2023. Deep learning techniques in radar emitter identification. *Defence Science Journal* 73 (5): 551.

Henzinger, T.A., B. Horowitz, and C.M. Kirsch. 2001. Embedded control systems development with Giotto. In *Proc. of the acm sigplan workshop on languages, compilers and tools for embedded systems,* 64–72.

Jones, C.B., and A. Burns. 2020. *A rely-guarantee specification of mixed-criticality scheduling.* arXiv. 2012.01493.

Joseph, M., and P. Pandya. 1986. Finding response times in a real-time system. *BCS Computer Journal* 29 (5): 390–395.

Paulitsch, M., O.M. Duarte, H. Karray, K. Mueller, D. Muench, and J. Nowotsch. 2015. Mixed-criticality embedded systems–a balance ensuring partitioning and performance. In *Proc. euromicro conference on digital system design (dsd),* 453–461. IEEE.

Pedro, P., and A. Burns. 1998. Schedulability analysis for mode changes in flexible real-time systems. In *10th euromicro workshop on real-time systems,* 172–179. IEEE Computer Society.

Razavi, K., M. Luthra, B. Koldehofe, Max M. Muhlhauser, and L. Wang. 2022. FA2: Fast, accurate autoscaling for serving deep learning inference with SLA guarantees. In *Proc. ieee real-time and embedded technology and applications symposium (rtas).*

Real, J., and A. Crespo. 2004. Mode change protocols for real-time systems: A survey and a new protocol. *Journal of Real-Time Systems* 26 (2): 161–197.

Sha, L., R. Rajkumar, J. Lehoczky, and K. Ramamritham. 1989. Mode change protocols for priority-driven premptive scheduling. *Journal of Real-Time Systems* 1 (3): 244–264.

Tindell, K., and A Alonso. 1996. *A very simple protocol for mode changes in priority preemptive systems.* Technical report. Universidad Politecnica de Madrid.

Tindell, K., A. Burns, and A. J. Wellings. 1992. Mode changes in priority preemptive scheduled systems. In *Proc. real time systems symposium,* 100–109. Phoenix, Arizona.

Vestal, S. 2007. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. real-time systems symposium (rtss),* 239–243.

Yao, S., S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher. 2017. Deepsense: a unified deep learning framework for time-series mobile sensing data processing. In *Proc. of the 26th international conference on world wide web,* 351–360.

Yoon, M., Junho J. Park, Y. Kim, JeongHoon J. Yi, and B. Koo. 2018. Research trends of mixed-criticality system. *The Journal of the Korea Contents Association* 18 (9): 125–140.

https://doi.org/10.1017/cbp.2024.2 Published online by Cambridge University Press