Quantitative Schedulability Analysis of Continuous Probability Tasks in a Hierarchical Context

Kim, Jin Hyun; Boudjadar, Jalil; Nyman, Ulrik; Mikucionis, Marius; Larsen, Kim Guldstrand; Skou, Arne; Lee, Insup; Thi Xuan Phan, Linh

Published in:
CBSE'15, Proceedings of the 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering

DOI (link to publication from Publisher):
10.1145/2737166.2737170

Publication date:
2015

Document Version
Early version, also known as pre-print

Quantitative Schedulability Analysis of Continuous Probability Tasks in a Hierarchical Context*

Jalil Boudjadar,
Alexandre David,
Jin Hyun Kim
Aalborg University
{jalil,adavid,jin}@cs.aau.dk

Kim G. Larsen,
Marius Mikučionis,
Ulrik Nyman,
Arne Skou
Aalborg University
{kgl,marius,ulrik,ask}@cs.aau.dk

Insup Lee,
Linh Thi Xuan Phan
University of Pennsylvania
{lee,linhphan}@cis.upenn.edu

ABSTRACT

We define the concept of degree of schedulability to characterize the schedulability and performance of soft real-time systems. The degree of schedulability of a system is given in terms of the two factors 1) Percentage of Missed Deadlines (PoMD); and 2) Degradation of the Quality of Service (DoQoS). Our work is set as a model-based framework for hierarchical scheduling systems where we introduce probability based sporadic tasks. The novel aspect is that we consider task arrival patterns that follow user-defined continuous probability distributions. The separately modeled task triggering events represent the system environment. We determine the degree of schedulability of a single scheduling component which can contain both periodic and sporadic tasks using statistical model checking in the form of UPPAAL SMC. Finally, we show the applicability of our framework by analyzing an avionics case study.

1. INTRODUCTION

In the areas of avionics and automotive, embedded systems are increasingly constructed as hierarchical scheduling systems, where a set of components share different resources. Some of these components are hard real-time (critical) while others may be soft real-time components, such that the hierarchical scheduling system itself is a mixed-criticality system. Due to the complexity and size of the systems, it is not feasible to analyze the complete system in one model.

We present a model-based analysis method that fits in a compositional approach [4, 26] for modeling and analyzing single-core hierarchical scheduling systems. In this paper we focus on the quantitative analysis of *soft real-time components*. We propose metrics for analyzing the quality of service of scheduling systems where some deadline misses can be tolerated. One novel aspect is that we handle sporadic tasks with arrival patterns modeled as continuous probabilistic functions.

*The research presented in this paper has been partially supported by EU Artemis Projects CRAFTERS and MBAT.

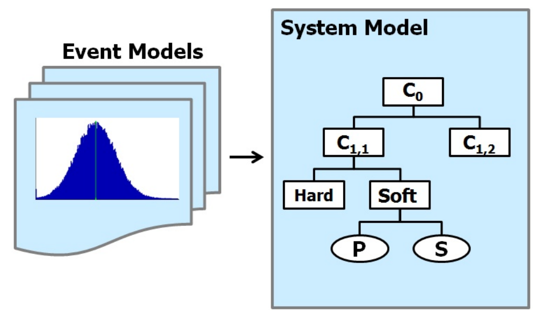


Figure 1: Overview of framework setup.

Supplying a system with less resources than it requires may lead to a degradation of the quality of service. Theoretical interest as well as practical considerations have motivated additional metrics, such as deadline miss ratio [20] and deadline miss probability [12], to quantify the degradation of the quality of service. For the estimation of the quality of service, in the case of hierarchical systems, we introduce the *degree of schedulability* (Sched°) in terms of the *Percentage of Missed Deadlines* (PoMD); and average delay per missed deadline, called *Degradation of Quality of Service* (DoQoS). The PoMD and DoQoS can be computed compositionally for any level of the system using only the interfaces of the immediately lower level components or tasks.

Our analysis technique relies on a model based setting that uses statistical and symbolic model checking. Symbolic model checking is used to ensure that the hard real-time tasks contained in a component never miss a deadline. The same models can be analyzed using statistical methods in order to obtain quality of service measures. Methods based on statistical model checking scale logarithmically in the size of the analyzed models, moreover they are trivially parallelizable and still scale sub-linearly [16], thus easily scaling to industrial size systems. Our framework gives a great degree of flexibility where all models are parameterized and can be instantiated for any kind of scheduling system.

Our method is intended to be used by system engineers during the design space exploration of an embedded system. The engineers can estimate the performance of the system under different environment assumptions and system configurations.

As illustrated in Fig. 1, we make the unusual choice of modeling the triggering events for the sporadic tasks separately from the system itself, leading to a clearer separation of concerns. A major motivation for this separation is that it more easily allows changing the environment model of the system without changing the system implementation. If one uses a static minimum inter-arrival time for sporadic tasks, the analysis can potentially be very pessimistic. The separation of concerns also allows for modeling different environments that represent different operating contexts or modes of the system. Our model fits well with sporadic tasks that are triggered by hardware interrupts, based on sensors and other embedded systems. An application field of this framework is automotive and avionics systems. The system consists of a set of hierarchical components. For each component, the timing requirements are represented by the interface consisting of period and budget. A component consists in a set of tasks sharing a single CPU according to a scheduling policy. We consider static (Fixed Priority) and dynamic priority (Earliest Deadline First) scheduling policies, together with preemptive execution of tasks. So that our system model can be analyzed under any kind of scheduling policy in the same way. When a soft real-time task misses its deadline, it continues to execute. Soft real-time tasks are triggered by external events that have continuous probabilistic arrival patterns such as Gaussian, uniform, exponential and user defined. We only consider cases where hard real-time tasks never miss a deadline.

Our main contributions are:

- We study the *degree of schedulability* (Sched°) of hierarchical scheduling systems where sporadic tasks have continuous probability arrival patterns.
- We show how to compute and estimate the two metrics *Percentage of Missed Deadlines* (PoMD); and average delay per missed deadline, called *Degradation of Quality of Service* (DoQoS).
- We provide detailed implementation models of the framework including explicit environment models as well a an avionics case-study.

The rest of the paper is structured as follows: Section 2 examines relevant related work, Section 3 introduces the compositional analysis framework. In Sections 4, 5 and 6 we introduce respectively continuous sporadic tasks, the models used to analyze them and the actual analysis. Finally, we demonstrate the applicability of our method on an avionics case study in Section 7, and conclude in Section 8.

2. RELATED WORK

In this section we present related work with a specific focus on sporadic tasks. The sporadic task model [3, 22], which is an extension of an earlier task model known as the Liu and Layland (LL) [17] task model has received immense research attention over the years. In [3], the authors propose an exact schedulability analysis by providing some necessary and sufficient conditions for a sporadic task system to be schedulable. In fact, the authors consider sporadic tasks with minimum inter-arrival time as periodic tasks, then define the set of legal requests that a task may perform. Based on such a function, they analyze the system schedulability regardless of the schedulability policy. However, considering sporadic tasks with known minimum inter-arrival times as periodic tasks may lead the schedulability analysis to be pessimistic and seriously overestimates the number of task arrivals. Our work differs by modeling probabilistic inter-arrival times and quantifying the system schedulability according to hard and soft real-time requirements.

In [29], the authors propose a framework for the schedulability analysis of real-time systems, where they define a generalized model for sporadic tasks to characterize more precisely the task arrival times. Each task is characterized by two constraints: *higher instantaneous arrival rate* which bounds the maximum number of task arrivals during some small time interval; *lower average arrival rate* which is used to specify the maximum number of arrivals over some longer time interval. In [9] the authors present a symmetric multicore framework where a flat scheduling system can be described in the Prelude language. The schedulability can be checked using generated UPPAAL models.

The work of [21] extends the work in [12] by making all the task attributes of a flat scheduling system probabilistic. However the methodology in [21] does not handle dynamic scheduling policies. In [28], the authors propose a method to control the preemptive behavior of real-time sporadic task systems by the use of CPU frequency scaling. They introduced a new sporadic task model in which the task arrival may deviate, according to a *discrete* time probability distribution, from the minimum inter-arrival time. Based on the probability of arrivals, the authors propose an on-line algorithm computing CPU frequencies that guarantee non-preemptiveness of task behavior while preserving system schedulability.

The work in [5] is an introduction to the concept of "degree of schedulability", without theory nor implementation. The current work is built on [5] by formally defining how to compute the two metrics: DoQoS and PoMD. Moreover, it also presents the UPPAAL models used for the implementation of the concepts as well as an avionics case-study.

To the best of our knowledge, there is no previous related work which uses continuous probabilities to characterize the arrival patterns of sporadic tasks. A concept similar to PoMD is given in the work by [20] which handles only flat soft real-time systems, whereas our framework can model and analyze hierarchical mixed criticality systems. Another difference is that [20] has a stochastically distributed execution time but with fixed periods. Our arrival patterns follow a probability distribution, whereas our tasks execution times are static.

The term "degree of schedulability" was first introduced in [23] to characterize the sum of response time delays from the individual task deadlines for static priority scheduling systems. The work in [23] is presented in the context of a distributed, but flat, real-time system with a common communication bus and only considers hard real-time systems.

We define the concept of DoQoS in a similar way, but focus on the total amount of time by which deadlines are missed. We define our notion of *degree of schedulability* (Sched°) by combining PoMD and DoQoS into one measure.

3. COMPOSITIONAL FRAMEWORK

A hierarchical scheduling system [1] consists of a set of concurrent real-time components sharing a set of resources according to a scheduling policy. Each component can be internally organized as a set of components, giving the system

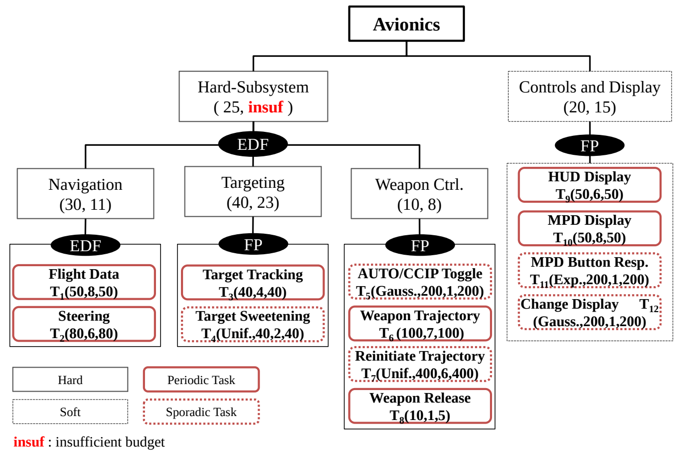


Figure 2: Mixed crit. hierarchical scheduling system

a tree like structure. The use of temporal partitioning [25] of components is motivated by the fact that it provides reduction of complexity, separation of concerns, confinement of failure modes, and temporal isolation among system applications. One obvious partitioning of the components in a mixed-criticality system is to group them according to their criticality [14]. Such a grouping enables easier certification of the safety critical components when they have minimal communication with the non safety critical parts [24].

In this paper we focus on the schedulability analysis of one component inside a hierarchical scheduling system. The hierarchical scheduling system can be as deeply nested as it is necessary for the given application, and is thus not restricted to two levels as shown in the avionics system of Fig. 2. The avionics system is based on a previously published case study [18, 13, 4]. Throughout this paper we use the **Targeting** component as a running example to illustrate our analysis method. The whole system will be analyzed in Section 7.

Formally, a hierarchical scheduling system $S = (C, R, A)$ is given by a set of hierarchical components C, a set of resources R and a scheduling algorithm A. A component, in turn, can be either a hierarchical unit $(\{C_1, .., C_n\}, A)$ of other components C_i, or a basic composition (W, A) of a workload W, together with a scheduling policy A. The workload W is a set of real-time tasks having time constraints like deadline, execution time and next arrival. The interface $\mathcal{I}$ [27] of a component is given in terms of a period and a budget, i.e. $\mathcal{I} = (p, b)$. The budget b specifies the resource amount that should be provided to the component workload in order for it to be schedulable. The interface $\mathcal{I}$ of a component $C(W, A)$ specifies the collective resource requirements that the workloads W performs under the scheduling policy A. In a compositional schedulability analysis framework [7, 4], a hierarchical system is said to be schedulable if each component is schedulable.

In Fig. 2, we specify for each task (in parenthesis) the period or arrival pattern (probability distribution together with the minimum inter-arrival time), followed by execution time and deadline. For each component we specify the period, minimum supply and the scheduling policy. One component has “insuf” as minimal supply because its resource requirement exceeds 100% of the system resource for one CPU. This is dealt with in Section 7 by distributing the components to different CPUs.

The analytical analysis approaches [15, 3, 19] compute whether or not a system is schedulable, according to EDF scheduling policy, by giving a firm response to the following question: is the demand bound function `dbf` of each component workload W, over a time interval t, lower or equal to the supply bound function `sbf` of a resource according to interface $\mathcal{I}$, over the same time interval, i.e. $\forall t > 0\ \mathtt{dbf}_A(W,t) \leq \mathtt{sbf}_{\mathcal{I}}(t)$. If such an equation is satisfied, the component is said to be schedulable. In the same way, in a model-based setting [28, 2, 10, 4] a system is said to be schedulable if the error locations, stating the deadline violation, are unreachable. Moreover, in our model-based framework, the condition $\mathtt{dbf}_A(W,t) \leq \mathtt{sbf}_{\mathcal{I}}(t)$ is applied not only to EDF but also for other scheduling policies, such as FP scheduling, so that the same task models are able to be used for different scheduling policies.

In contrast to the mentioned techniques, we do not only consider if a system is schedulable or not, but we provide the *degree of schedulability* (Sched°) as a way to measure how schedulable a system is. We define the Sched° of an entity (system, component or task) by the two concepts: *Percentage of Missed Deadlines* (PoMD) and *Degradation of Quality of Service* (DoQoS). Each of these concepts can be computed for either a task, a component or a complete embedded system. They should be measured or simulated over a sufficiently large time bounded run and a sufficiently large number of runs in order to obtain usable values.

By $\mathcal{S}$ we designate the system comprising the probabilistic models of the event-triggering as well as the hierarchical scheduling of tasks as depicted in Fig. 1. A run π of a system $\mathcal{S}$ is an infinite sequence:

$$\pi = s_0(t_0, e_0)s_1(t_1, e_1)\dots s_n(t_n, e_n)\dots$$

where s_i is a global state giving information about the state of each task (e.g. idle, ready, running, blocked) and resource (e.g. idle, occupied) at stage i; s_0 is the initial state. Each e_i indicates an event (triggering, completing or preemption) signifying a transition from state s_i to s_{i+1}. Timestamp t_0 indicates the time from system initiation until event e_0. Every subsequent timestamp t_i (with $i \geq 1$) indicates the separation between events e_{i-1} and e_i.

We denote by Runs the set of runs of $\mathcal{S}$. For a run π and a time-bound $t \in \mathbb{R}_{\geq 0}$, we may define (in an obvious manner) the functions:

- $\mathsf{Miss}_i^t(\pi) \in \mathbb{N}$ is the total number of missed deadlines for task i up to time t;
- $\mathsf{Trig}_i^t(\pi) \in \mathbb{N}$ is the total number of triggerings of task i up to time t.

Definition 3.1. *The Percentage of Missed Deadlines (PoMD) of an entity X for a run π is given by:*

$$\mathsf{PoMD}^X(\pi) = (\limsup_{t\to\infty} \frac{\mathsf{Miss}_t(X,\pi)}{\mathsf{Trig}_t(X,\pi)}) \times 100$$

where $\mathsf{Miss}_t(X,\pi)$ is the total number of deadlines missed by X on run π up to time bound t, and $\mathsf{Trig}_t(X,\pi)$ is the total number of X executions triggered within the run π until time bound t. The entity X could be a task, a component or a system. In the case where X is a system, Miss_t and Trig_t are computed with the system components considered as tasks. Even if no top level component misses a deadline, a task inside one of the components could still miss its deadline. A healthy system engineering approach might be to ensure that all levels except the lowest levels have a PoMD of 0. Now, the probabilistic arrival patterns of tasks of S give rise to a unique probability measure $\mathbb{P}_{\mathcal{S}}$ over (Runs, $\mathcal{B}$)[1] as such PoMD^T and PoMD are random variables. In order to estimate the expected values of PoMD^X and ePoMD^X, we generate a set Π of random (according to the stochastic semantics of S) and independent runs and calculate the mean using the following formula:

$$\mathsf{ePoMD}^X(\Pi) = \frac{\sum_{\pi \in \Pi} \mathsf{PoMD}^X(\pi)}{|\Pi|}$$

In fact, we estimate the ePoMD at the system level by simulating the complete system and summing up all triggering events and deadline misses. Our concept of PoMD is similar to the concept Deadline Miss Ratio (DMR) from [20].

Definition 3.2. *We define the Degradation of Quality of Service (*DoQoS*) of a task T_i over a single run π by:*

$$\mathsf{DoQoS}^{T_i}(\pi) = \begin{cases} 0 & \textit{if } \limsup_{t\to\infty} \mathsf{Miss}_t(T_i,\pi) = 0 \\ \limsup_{t\to\infty} \frac{\mathsf{Overrun}_t(T_i,\pi)}{\mathsf{Miss}_t(T_i,\pi)} & \textit{Otherwise} \end{cases}$$

where $\mathsf{Overrun}_t(T_i,\pi)$ is the sum of the time amounts by which task T_i misses its deadline over run π. An example of an overrun for a specific triggering, $overrun_j$, is given in Fig. 3.

Similarly as done for PoMD*, we estimate the expected value of* DoQoS*, called* eDoQoS*, using a set of random and independent runs.*

$$\mathsf{eDoQoS}^{T_i}(\Pi) = \frac{\sum_{\pi \in \Pi} \mathsf{DoQoS}^{T_i}(\pi)}{|\Pi|}$$

The eDoQoS *of a component C over a set of time bounded runs Π is defined by the* eDoQoS *of its workload W as:*

$$\mathsf{eDoQoS}^C(\Pi) = \frac{\sum_{T_i \in W} \mathsf{eDoQoS}^{T_i}(\Pi)}{|W|}$$

Each item i in the workload W can either be a task or a component. The eDoQoS can be recursively calculated up to the system level. The eDoQoS of a task could be used to compare the same task embedded in different components with different configurations. For the eDoQoS of a component we chose to use a simple weighted average.

Definition 3.3. *We define the degree of schedulability (*Sched°*) of an entity in terms of two factors* $\mathsf{Sched}^\circ_\mathsf{P}$ *and* $\mathsf{Sched}^\circ_\mathsf{D}$ *to be given by:*

$$\mathsf{Sched}^\circ_\mathsf{P} = \left\{ \begin{array}{ll} \infty & \textit{if } \mathsf{ePoMD} = 0 \\ \frac{1}{\mathsf{ePoMD}} & \textit{Otherwise} \end{array} \right\}$$

$$\mathsf{Sched}^\circ_\mathsf{D} = \left\{ \begin{array}{ll} \infty & \textit{if } \mathsf{eDoQoS} = 0 \\ \frac{1}{\mathsf{eDoQoS}} & \textit{Otherwise} \end{array} \right\}$$

According to such a definition, an entity is absolutely schedulable if either $\mathsf{Sched}^\circ_\mathsf{P}$ or $\mathsf{Sched}^\circ_\mathsf{D}$ is equal to ∞. This corresponds to the classical notion of schedulability where no deadline is missed.

[1]Here $\mathcal{B}$ is the standard Σ-algebra over Runs generated from a standard cylinder construction. For more see e.g. [11].

To compare different system configurations in terms of the Sched°, we use the multi-objective Pareto frontier of $\mathsf{Sched}^\circ_\mathsf{P}$ and $\mathsf{Sched}^\circ_\mathsf{D}$. In this way, engineers could keep updating resources and requirements and compare the system Sched° from one configuration to another. Thus, this fact helps to define the best system configuration in terms of an equation including the amount of provided resources, the expected schedulability degree and the task requirements. But, Sched° is not intended as a measure to compare completely unrelated systems.

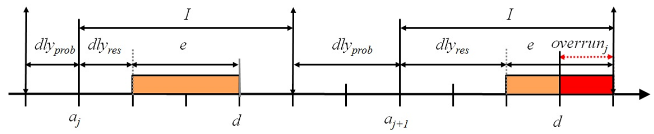


Figure 3: Execution of a sporadic task, $T^S(\mathcal{P}, 4, 2, 3)$

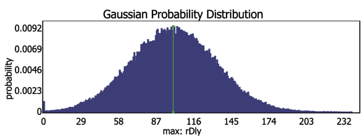
We reuse and adapt our previous work [4], for the schedulability analysis of hierarchical systems, now extended with probabilistic sporadic tasks. As mentioned earlier, a workload $W = \{T_1, .., T_m\}$ is a set of periodic and sporadic tasks. Periodic tasks [17] $T^p(p, e, d)$ are commonly given at least by a period p, an execution time e and a deadline d. Similarly, sporadic tasks [3] $T^s(I, e, d)$ are usually specified with a minimum inter-arrival time I, an execution time e and a relative deadline d. In order to characterize more precisely the arrival time of sporadic tasks and capture efficiently the deviation of their arrivals from the minimum inter-arrival time, we associate to each sporadic task a continuous probability distribution stating the probability of each possible delay dly_{prob}. Thus, our sporadic task model is given by $T^s(\mathcal{P}, I, e, d)$ where $\mathcal{P}$ is a probability distribution given by a density function $\mathcal{F}$. Depending on the density function $\mathcal{F}$, the probability distribution $\mathcal{P}$ could be uniform, exponential or Gaussian. Fig. 3 depicts an example of the execution of our probability based sporadic task model where we show how the probability distribution influences the task behavior, and thus affects the task schedulability. We use $a_{i,j}$ as the j^{th} arrival of the task with index i. The task arrival a_j delays for $dly_{prob} = 1$ time unit, according to the probability distribution, from the previous minimum inter-arrival time (expected at the starting point of the time axis). The task arrives at time a_j and becomes immediately ready to start its execution. Unfortunately, due to the resource availability the task waits $dly_{res} = 1$ time unit before acquiring resources and starting its execution. After being provided with resources, the task starts its execution e which achieves perfectly with the deadline d. After one minimum inter-arrival time $I = 4$ since the last task arrival a_j, the task may start a new execution. Always depending on the probability distribution, the new arrival a_{j+1} of the task delays for $dly_{prob} = 2$ time units from the last minimum inter-arrival time point. After being ready, the task delays again $dly_{res} = 1.5$ because of the resource availability. After acquiring resources, the task starts its execution $e = 2$ which leads task to miss its deadline d with an amount of time $dly_{miss} = 0.5$. One can remark that such an excess could be not critical and can be measured as Quality of Service (QoS) of the schedulability. Our probability-based sporadic task model is strictly more expressive than traditional real-time task models but could retain efficient demand computation for the analysis.

4. CONTINUOUS PROBABILITY TASKS

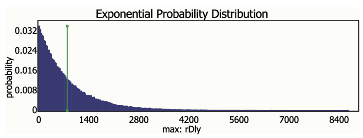
In this section, we introduce the characteristics of the probability-based sporadic tasks. Our framework models both a fixed inter-arrival time and a probability distribution. Obviously, a task cannot arrive before the inter-arrival time, and the inter-arrival time can potentially be set to zero. After the expiration of the inter-arrival time, the arrival of a given task delays with δ according to a continuous probability distribution, such as Gaussian $\mathcal{N} = (\mu, \sigma^2)$ with a mean value μ and a variance σ^2 (Fig. 4(a)). Fig. 4 shows the three specific probability distributions we consider in our setting: Gaussian, exponential and uniform. As the probability distribution is a parameter of the sporadic tasks in our framework, any user defined probability distribution can be used.

4.1 Probability Distributions

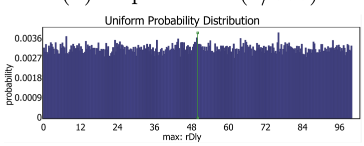
We have implemented the continuous probability distributions we consider via a set of Uppaal embedded functions over the time domain. An example of a Gaussian normalized curve, generated by Uppaal SMC, is depicted in Fig. 4(a) where the x axis represents continuous time from 0 to 240, μ=100 , and σ =100. Fig. 4(b) shows an exponential probability distribution, with the rate of exponential λ being $\frac{1}{800}$. The smaller λ is, the more spread out the distribution is. In contrast to the two previous probability distributions, the uniform distribution (Fig. 4(c)) has a equal probability for all time instances up to a maximum time where the probability drops to zero.



(a) Gaussian (100,100)



(b) Exponential (1/800)

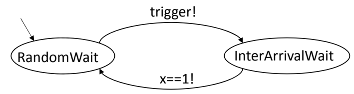


(c) Uniform (1,100)

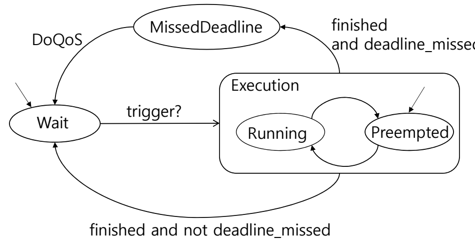
Figure 4: Probabilistic arrival patterns.

4.2 Conceptual Model of Sporadic Tasks

Our conceptual event model is shown in Fig. 5(a). When a delay has elapsed, the event triggers the corresponding task and moves to the location InterArrivalWait waiting for one inter-arrival time I before starting a new round. The conceptual task model (Fig. 5(b)) starts at location Wait waiting for the triggering event (trigger?) by which it moves to the composite state Ready. Depending on the scheduling, the task can alternate between the locations Run and Preempted, while the deadline is not missed. The task can leave this composite state when completing the execution. If the task ends up in location MissedDeadline, the overrun will be measured (used for estimating the DoQoS) before moving to the location Wait. The Uppaal implementations of our probability based sporadic task model, omitted because of the space limitation, are very close to the conceptual event triggering model given in Fig. 5(a) and can be found in the linked zip-file. In the Uppaal task model (Fig. 11), the composite state is modeled by a single location in which the preemption is captured by a stopwatch; which is active when the task is running and stopped when the task is preempted. Thus, the stopwatch will always contain the accumulated execution time of the task.



(a) Event model



(b) Task model

Figure 5: Conceptual model of a sporadic task and its triggering event.

5. ANALYSIS MODELS FOR THE DEGREE OF SCHEDULABILITY

For our compositional analysis framework, the hierarchical scheduling systems and their analysis elements consist of environment models, scheduling models, resource model, and task models.

We are using Uppaal SMC to perform a formalized statistical simulation of our models, known as Statistical Model Checking (SMC). SMC enables quantitative performance measurements instead of the Boolean (true, false) evaluation that symbolic model checking techniques provide. We can summarize the main features of Uppaal SMC in the following:

- Stopwatches [8] are clocks that can be stopped and resumed without a reset. They are very practical to measure the execution time of preemptive tasks.
- Simulation and estimation of the expected minimum or maximum value of expressions over a set of runs, `E[bound](min:expr)` and `E[bound](max:expr)`, for a given simulation time and/or number of runs specified by `bound`.
- Probability evaluation `Pr[bound] (P)` for a property `P` to be satisfied within a given simulation time and/or number of runs specified by `bound`. `P` is specified using either LTL or tMITL logic.

The disadvantage of using statistical model checking is that it will not provide complete certainty that a property is satisfied, but only verify it up to a specific confidence level [6], given as an analysis parameter.

5.1 Periodic Resource Model

The resource model that this paper considers is the Periodic Resource Model (PRM), which provides a specific amount of resources to a set of tasks or components every period [26]. The PRM represents the interface requirement between a set of tasks and their (higher level) scheduler. The high level scheduler is referred to by Supplier, which satisfies the interface requirement given by the periodic resource model. To represent the behavior of the resource supply, based on the interface requirement, we use the PRM in the form of a PSA model.

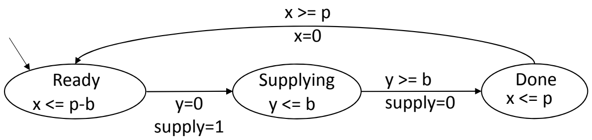


Figure 6: Conceptual model of PRM

In order to model the PRM for any type of scheduling policy, we provide a stochastic resource model as shown in Fig. 6. This resource model guarantees the specific amount of resource allocation for a specific period, but the beginning of the supply is non-deterministic such that it implements an asynchronous supply of resources of the PRM [26]. In this model, the variable supply represents the resource allocation, which is a variable shared with the task model. Thus, the supply is only enabled for b time units (budget) within the period p. At location Ready, the supply of resource can be delayed for at most $p - b$. The contracted amount of resource in the interface of a component is fully fed to a set of tasks in that component at location Supplying, and then the remaining time of a period is spent at location Done.

One can remark that our resource model supplies the whole budget non-preemptively in one chunk, but according to [26] if one considers only worst cases, both preemptive and non-preemptive resource models provide the same worst case analysis results. This is also true for our framework because we search for the traces with the highest PoMD or DoQoS. What we are analyzing is the DoQoS and PoMD of a component in any potential setting where this component could be used given that it is still supplied with its budget. We achieve this by analyzing all extreme cases of the supply of the budget.

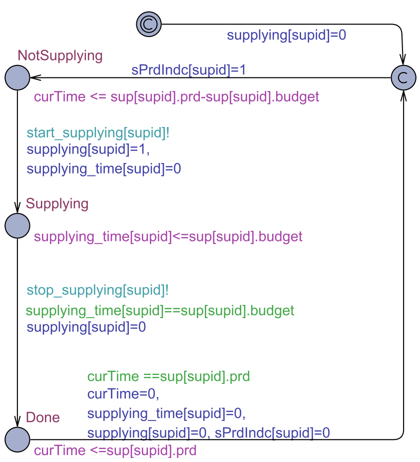
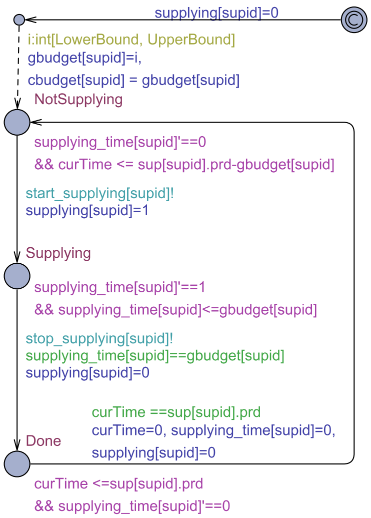


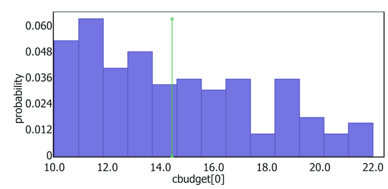
Figure 7: PSA template of Periodic Resource Model

Fig. 7 shows the supplier template. The initial location is committed in order to enforce the supplier to move instantaneously to the next location NotSupplying. Slack time (sup[supid].prd-sup[supid].budget) is the maximum amount of time that can elapse before the supplier starts supplying. Non-deterministically at some point between time zero and the slack time, the supplier moves to the location Supplying. At this location, the progress rate of the stopwatch supplying_time[supid] is set to 1. Once the budget is fully provided, the Supplier moves to the location Done.

Fig. 8 shows the supplier supply, and runs of tasks T_3^p and T_4^s (of Fig. 2). In this setting, T_4^s has priority over T_3^p, and executes sporadically over a uniform distribution. Thus, the execution period of T_4^s is irregular. The supplier at the bottom is supplying non-deterministically so the supply is also irregular within the period. Further explanation can be found in our previous paper [4].



(a) Budget estimation model



(b) Budgets causing deadline miss

Figure 9: Budget estimation.

In order to estimate the sufficient budget of a supplier (component) that makes the workload of a component schedulable, we present another stochastic supplier as shown in Fig. 9.(a). It starts supplying by selecting a random amount of budget using gbudget[supid] and cbudget[supid]. UPPAAL SMC checks whether any task misses deadline and generates a probability distribution of budgets leading to a deadline miss of a component. Fig. 9.(b) shows the estimated budget numbers that make the component of T_3^p and T_4^s non-schedulable, and it can be concluded that 23 is the minimum budget.

5.2 Scheduler

We have implemented different scheduling policies in our framework, but we only show the EDF scheduler here as an example. Fig. 10 shows the implementation of the EDF scheduler. At the initial location WaitSchedReq, it waits for a scheduling request. In the location SearchQPosition, the scheduler searches through the queue until it has found the right position. On the transition to location AckSchedReq, it

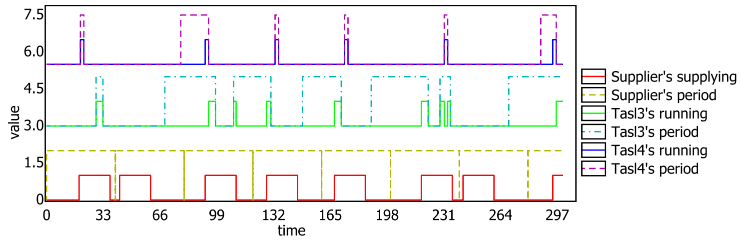


Figure 8: Supplier and Task Execution.

inserts the task at the correct place in the queue. Finally, on returning to location WaitSchedReq it communicates to the rest of the system the task id of the currently scheduled task.

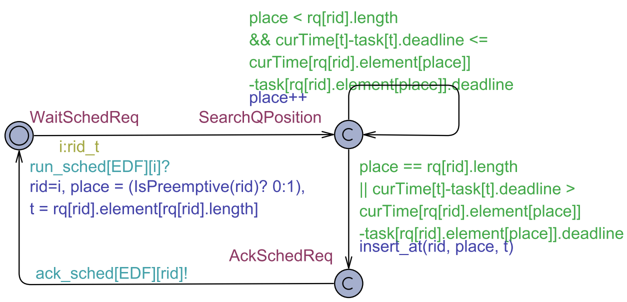


Figure 10: PSA template of EDF Scheduler

5.3 Task Models

In the rest of this paper, we use PoMD instead of ePoMD, and similarly DoQoS for eDoQoS.

In our framework, we provide 4 different task templates: hard real-time and soft real-time templates for periodic and sporadic tasks. The hard real-time task stops running immediately when it misses a deadline. Meanwhile, the soft real-time task continues running until the end of simulation time while measuring PoMD and DoQoS. Fig. 11 shows the soft real-time sporadic task template. It is triggered by the event startTask[tid]? from the environment model following a probability distribution. The clocks curTime[tid] and exeTime[tid] are used to measure the current time and the execution time respectively. The clock twcrt[tid] measures the worst-case response time. Similarly to the conceptual model, location Execution models both the running and preempted states of the task. The stopwatch exeTime[tid] measuring the execution time keeps increasing while the task is scheduled (exeTime[tid]' == isTaskSched()). The function isTaskSched() returns zero when the task is preempted and one when the task is scheduled. When the task is preempted it

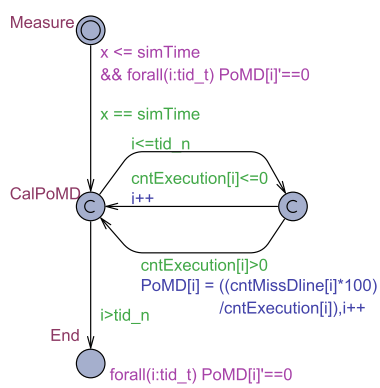


Figure 12: PoMD calculator

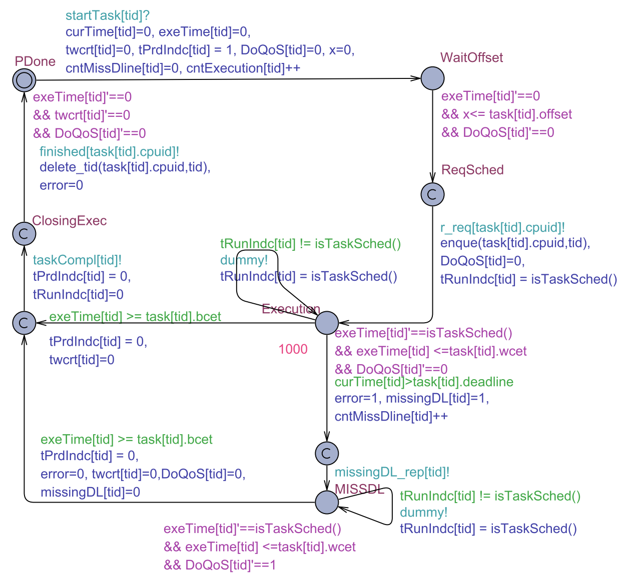


Figure 11: Soft real-time sporadic task

stays at location Execution, but the stopwatch exeTime[tid] is stopped by setting the rate of progression for it (exeTime[tid]') to zero. The task cannot stay active in the location Execution longer that the WCET (exeTime[tid] $\leq$ taskTid[wcet]). Once the task has actually been scheduled for more than the BCET, it can non-deterministically choose to go to the location ClosingExec, issuing a taskCompl[tid]! event stating that the task execution is done. When the deadline is missed, the task will be forced to change location to MISSDL. In that location, even though the deadline is missed the task keeps running, whereas the stopwatch exeTime[tid] keeps measuring the accumulated execution time.

The variables cntExecution[tid], cntMissDline[tid], and DoQoS[tid] are used to calculate PoMD and DoQoS by the PoMD calculator of Fig. 12 and the following queries:

E[gClock<=simTime;simNum] (max: PoMD[1])

E[gClock<=simTime;simNum] (max: DoQos[1])

For every simulation (simNum) of which time is up to simTime, one PoMD[1] is obtained by calculating the percentage of the accumulated number of missed deadlines of cntMissDline[tid], and the count of task executions of cntExecution[tid]. Finally, PoMD is calculated from the average of PoMD[1]s of all traces. DoQoS[tid] measures the delay after a task misses a deadline, and the maximum DoQoS[tid] is selected from one trace. Finally, DoQoS is determined by calculating the average of the maximum DoQoS[tid]s of each individual trace.

6. ANALYSIS OF THE DEGREE OF SCHEDULABILITY

We use as a running example in this section the Target component from Fig. 2. The workload is characterized by a periodic task $T_3^p(40, 4, 40)$ and a sporadic task $T_4^s(Unif., 40, 2, 40)$. In our setting, T_4^s has priority over T_3^p. Both tasks are scheduled according to the fixed priority scheduling (FP) policy. The sporadic task T_4^s follows the uniform probability distribution between 0 and 20 time units. The analysis is performed in the following steps:

1. Estimate a component budget as described in Section 5.

2. Analyze the Sched° for the estimated and lower budgets.

To estimate the budget of a component, we use the budget estimator shown in Fig. 9(a) and the following query:

$$Pr[cbudget[rid] <= randomBudget]$$
$$(<> gClock >= simTimeanderror)$$

As a result, we found that 23 time units every 40 time units is a good candidate as a sufficient budget for both tasks. In order to have valid results, in the next analysis section we perform experiments where we analyze the same system with a varying amount of traces and simulation time. When reaching more than 1000 traces and a simulation time of more than 100,000 time units, we see that the results stabilize.

6.1 Analysis of Mixed-Criticality Components

Because our analysis framework is compositional, we can analyze different components with different methods based on the criticality of each component. If a component contains hard real-time tasks, it should be analyzed using rigorous methods [4, 27]. Components containing both soft and hard real-time tasks should be analyzed both for the degree of schedulability and with rigorous methods to ensure that no hard task ever misses a deadline. So that, statistical model checking is used to analyze that the DoQoS and PoMD of the hard real-time tasks are 0. Secondly, in order to obtain 100% confidence, we use the firm schedulability analysis technique presented in [4]. In this way two different analysis techniques are combined to analyze a mixed-criticality hierarchical scheduling system in an efficient way, while ensuring confidence in the critical parts of the system.

6.2 Analysis Results

In Table 1, we show that T_3 and T_4 are schedulable under the interface (40, 23) even if T_4 is treated as a periodic task with a period equal to the minimal inter-arrival time. This is the classical worst-case budget estimation, and our analysis also confirms that tasks miss exactly 0 deadlines and have a DoQoS of 0. Throughout the running example, we use FP scheduling but our framework supports other scheduling policies.

Suppose that the resource amount provided to the component is reduced to 18. In order to have a baseline to compare with, in the next analysis steps, we perform an artificial experiment presented in Table 2. We analyze task T_4 using the sporadic template, but with a completely fixed periodic arrival pattern. Note that the sporadic task T_4^s never misses its deadline, because it has the highest priority. Table 2 shows the average value of the PoMD and DoQoS for 1000 traces as well as the variance of each one. In the following, we fix the set of tasks and vary the arrival pattern of the sporadic task. This is done in order to show the versatility of our method. In an engineering setting, the arrival patterns will usually be fixed while the workload and budgets vary. For the same deficit budget (40, 18), Table. 3 shows the degree of schedulability when the sporadic task T_4^s is assumed to follow an exponential probability distribution with different rates of exponential.

Table 4 shows the Sched° for the two tasks given a uniform probability distribution for triggering the sporadic task T_4^s. Table 5 shows the results of our analysis when using different Gaussian distributions, all with a mean value μ of 10 and different deviations σ.

To sum up, we have provided a highly configurable analysis framework where the workload, task types, arrival-patterns, priorities and scheduling mechanisms can be varied and a given system configuration can be easily analyzed. All UPPAAL models used in this paper are available at: `http://people.cs.aau.dk/~ulrik/submissions/872346/CBSE.zip`

Table 1: The degree of schedulability of tasks under periodic events

Component ((40, 23), FP)	PoMD	DoQoS
$T_3^p(40, 4)$,	0	0
$T_4^s(40, 2)$,	0	0

Table 2: The degree of schedulability of tasks under lack of budget

Component ((40, 18), FP)	PoMD	DoQoS
$T_3^P(40, 4)$,	0.032 ± 0.033	1.610 ± 0.616
T_4^s(Periodic, 40, 2, 40)	0	0

7. CASE STUDY

As a case study to show the applicability of our analysis framework, we analyze the schedulability of an avionics system [18, 13, 4]. We use the same timing specification as [13], whereas the system structure depicted in Fig. 2 follows the description given in [18]. In our analysis we include information about the criticality of the individual tasks, something which has not been included in any of the previous treatments of that case study. Table 6 summarizes both architecture and timing attributes of the avionics system.

The avionics system is a mixed-criticality application, where

Table 3: Degree of schedulability of tasks under Exponential distribution.

Component ((40, 18), FP)	Rate of Exp. of T_4^s	PoMD	DoQoS
$T_3^p(40, 4)$	1/100,000	0.040 ± 0.040	0.489 ± 0.518
	1/1000	0.043 ± 0.041	0.581 ± 0.630
	1/10	0.035 ± 0.032	0.705 ± 0.634
T_4^s(Exp.,40,2,40)	1/100,000	0.0± 0.0	0.003 ± 0.008
	1/1000	0.182 ± 0.168	0.259 ± 0.353
	1/10	0.223 ± 0.039	1.792 ± 0.319

Table 6: Generic Avionics Components and Tasks

Component	Criticality	T_i	e_i	p_i	d_i	Importance
Navigation	Hard	Aircraft flight data(T_1^p)	8	50(55)		critical
		Steering(T_2^p)	6	80		critical
Targeting	Hard	Target tracking(T_3^p)	4	40		critical
		Target sweetening(T_4^s)	2		40	critical
Weapon Control	Hard	AUTO/CCIP toggle(T_5^s)	1		200	critical
		Weapon trajectory(T_6^p)	7	100		critical
		Reinitiate trajectory(T_7^s)	6		400	essential
		Weapon release(T_8^p)	1	10	5	critical
Controls & Displays	Soft	HUD display(T_9^p)	6	55(52)		essential
		MPD tactical display(T_{10}^p)	8	50(52)		essential
		MPD button response (T_{11}^s)	1		200	background
		Change display mode (T_{12}^s)	1		200	background

Table 4: Degree of schedulability of tasks under Uniform distribution.

Component ((40, 18), FP)	PoMD	DoQoS
$T_3^p(40, 4)$,	0.057 ± 0.040	0.008 ± 0.008
T_4^s(Unif., 40, 2, 40)	0.188 ± 0.037	1.941 ± 0.274

Table 5: Degree of schedulability under Gaussian distribution

Component ((40, 18), FP)	(μ, σ) of T_4^s	PoMD	DoQoS
$T_3^p(40, 4)$	(10,10)	0.068 ± 0.047	0.903 ± 0.648
	(10, 8)	0.046 ± 0.045	0.903 ± 0.777
	(10, 5)	0.033 ± 0.036	0.810 ± 0.686
	(10, 1)	0.057 ± 0.471	0.875 ± 0.759
T_4^s(Gauss., 40, 2, 40)	(10,10)	0.184 ± 0.038	1.844 ± 0.287
	(10, 8)	0.173 ± 0.040	1.908 ± 0.239
	(10, 5)	0.175 ± 0.038	1.801 ± 0.277
	(10, 1)	0.185 ± 0.340	1.823 ± 0.308

we mainly considered 7 periodic tasks and 5 sporadic tasks all grouped in 4 components. In the case of critical sporadic tasks, we have introduced a periodic event to trigger each task where the event period is equal to the minimum inter-arrival time of those tasks. We characterize the arrival times of non-critical sporadic tasks by different probability distributions where the delays generated by such distributions are relatively proportional to the minimum inter-arrival times of the corresponding tasks. This was chosen as the original case-study did not contain any information on the probability distributions. Because of space limitations, we only provide the analysis results of one component (**Controls & Displays**). Table 7 states the degree of schedulability of the tasks in the component **Control & Displays**. One can remark that component **Control & Displays** cannot be scheduled with a budget less than 20 for a period equal to 30 because, at least, one of the tasks misses its deadline. In particular, tasks **HUD Display** and **MPD Display** miss their deadlines because they are "background tasks", i.e. having lower priorities. While keeping budget increasing, DoQoS and PoMD are decreasing until reaching 0, meaning that the corresponding budget (20) is the minimal sufficient budget making the component workload schedulable. Other budget values (14, 17, 19) can be acceptable as sufficient, depending on the quality of service required by the system. We have quantified the components schedulability according to the budgets, hard and soft real-time requirements.

8. CONCLUSIONS

We have presented a compositional method for analyzing the degree of schedulability of hierarchical real-time systems. The system is modeled in terms of components containing periodic and sporadic tasks. In order to characterize more accurately the arrival time of sporadic tasks, we introduced continuous probability distributions. Given hard and soft real-time requirements, our approach provides probabilistic guarantees on the system schedulability. The Degree of Schedulability (Sched°) is defined by the two factors: 1) Percentage of Missed Deadlines (PoMD) and 2) Degradation of Quality of Service (DoQoS). These concepts are helpful when analyzing systems or components with insufficient budgets to meet all deadlines. UPPAAL SMC is used to perform statistical model checking, in order to compute the DoQoS and PoMD. Finally, we have demonstrated the applicability of our approach by analyzing the degree of schedulability of an avionics case study which was previously shown to be non-schedulable [18, 13, 4].

9. REFERENCES

[1] M. Anand, S. Fischmeister, and I. Lee. A comparison of compositional schedulability analysis techniques for hierarchical real-time systems. *ACM Trans. Embed. Comput. Syst.*, 13(1):2:1–2:37, Sept. 2013.

[2] M. Åsberg, T. Nolte, and P. Pettersson. Prototyping and code synthesis of hierarchically scheduled systems using times. *Journal of Convergence (Consumer Electronics)*, 1(1):77–86, December 2010.

[3] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *RTSS*, pages 182–190. IEEE Computer Society Press, 1990.

[4] A. Boudjadar, A. David, J. H. Kim, K. G. Larsen, M. Mikučionis, U. Nyman, and A. Skou. Hierarchical scheduling framework based on compositional analysis using uppaal. In *Proceedings of FACS 2013*, lncs. Springer, 2013. To appear.

[5] A. Boudjadar, A. David, J. H. Kim, K. G. Larsen, M. Mikucionis, U. Nyman, and A. Skou. Degree of schedulability of mixed-criticality real-time systems with probabilistic sporadic tasks. In *Theoretical Aspects of Software Engineering Conference (TASE), 2014*, pages 126–130, Sept 2014.

Table 7: Schedulability degree of component Controls & Displays

Task	Sched°	Budget=14	Budget=17	Budget=19	Budget=20
HUD Display(T_9)	DoQoS	0.004±0.003	0	0	0
	PoMD	0.004±0.004	0	0	0
MPD Display(T_{10})	DoQoS	3.068±0.151	0.343±0.052	0.003±0.003	0
	PoMD	0.231±0.018	0.002±0.002	0.0005±0	0
MPD Button(T_{11})	DoQoS	0	0	0	0
	PoMD	0	0	0	0
Change Mode(T_{12})	DoQoS	0	0	0	0
	PoMD	0	0	0	0

[6] P. E. Bulychev, A. David, K. G. Larsen, M. Mikucionis, D. B. Poulsen, A. Legay, and Z. Wang. Uppaal-smc: Statistical model checking for priced timed automata. In H. Wiklicky and M. Massink, editors, *QAPL*, volume 85 of *EPTCS*, pages 1–16, 2012.

[7] L. Carnevali, A. Pinzuti, and E. Vicario. Compositional verification for hierarchical scheduling of real-time systems. *IEEE Transactions on Software Engineering*, 39(5):638–657, 2013.

[8] F. Cassez and K. G. Larsen. The impressive power of stopwatches. In C. Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2000.

[9] M. Cordovilla, F. Boniol, J. Forget, E. Noulard, and C. Pagetti. Developing critical embedded systems on multicore architectures: the PRELUDE-SCHEDMCORE toolset. In S. Faucou, A. Burns, and L. George, editors, *RTNS 2011*, pages 107–116, 2011.

[10] A. David, K. G. Larsen, A. Legay, and M. Mikučionis. Schedulability of herschel-planck revisited using statistical model checking. In *ISoLA (2)*, volume 7610 of *LNCS*, pages 293–307. Springer, 2012.

[11] A. David, K. G. Larsen, A. Legay, M. Mikucionis, D. B. Poulsen, J. van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In U. Fahrenberg and S. Tripakis, editors, *FORMATS*, volume 6919 of *LNCS*, pages 80–96. Springer, 2011.

[12] J. Diaz, D. Garcia, K. Kim, C.-G. Lee, L. Lo Bello, J. Lopez, S.-L. Min, and O. Mirabella. Stochastic analysis of periodic real-time systems. In *RTSS 2002*, pages 289–300, 2002.

[13] R. Dodd. Coloured petri net modelling of a generic avionics missions computer. Technical report, Department of Defence, Australia, Air Operations Division, 2006.

[14] G. Horvath, S. Chung, D. Dvorak, and D. Hecox. Safety-critical partitioned software architecture: A partitioned software architecture for robotic spacecraft. In *Infotech@Aerospace Conference 2011*. American Institute of Aeronautics and Astronautics, 2011.

[15] J. Lee, L. T. X. Phan, S. Chen, O. Sokolsky, and I. Lee. Improving resource utilization for compositional scheduling using dprm interfaces. *SIGBED Rev.*, 8(1):38–45, Mar. 2011.

[16] A. Legay and B. Delahaye. Statistical model checking : An overview. *CoRR*, abs/1005.1327, 2010.

[17] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, Jan. 1973.

[18] C. D. Locke, D. R. Vogel, L. Lucas, and J. B. Goodenough. Generic avionics software specification. Technical report, DTIC Document, 1990.

[19] J. Lorente, G. Lipari, and E. Bini. A hierarchical scheduling model for component-based real-time systems. In *IPDPS 2006*, pages 8 pp.–, 2006.

[20] S. Manolache, P. Eles, and Z. Peng. Optimization of soft real-time systems with deadline miss ratio constraints. In *Proceedings of RTAS 2004*, pages 562–570, 2004.

[21] D. Maxim and L. Cucu-Grosjean. Response Time Analysis for Fixed-Priority Tasks with Multiple Probabilistic Parameters. In *RTSS 2013*, Vancouver, Canada, 2013.

[22] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Cambridge, USA, 1983.

[23] P. Pop, P. Eles, and Z. Peng. Schedulability-driven communication synthesis for time triggered embedded systems. *Real-Time Systems*, 26(3):297–325, 2004.

[24] P. Pop, L. Tsiopoulos, S. Voss, O. Slotosch, C. Ficek, U. Nyman, and A. Lopez. Methods and tools for reducing certification costs of mixed-criticality applications on multi-core platforms: the recomp approach. In *WICERT 2013 proceedings*, 2013.

[25] K. Purna and D. Bhatia. Temporal partitioning and scheduling data flow graphs for reconfigurable computers. *Computers, IEEE Transactions on*, 48(6):579–590, Jun 1999.

[26] I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. In *ECRTS*, pages 181–190. IEEE Computer Society, 2008.

[27] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.

[28] A. Thekkilakattil, R. Dobrin, and S. Punnekkat. Probabilistic preemption control using frequency scaling for sporadic real-time tasks. In *The 7th IEEE International Symposium on Industrial Embedded Systems*, 2012.

[29] Y. Zhang, D. K. Krecker, C. Gill, C. Lu, and G. H. Thaker. Practical schedulability analysis for generalized sporadic tasks in distributed real-time systems. In *Proceedings of ECRTS '08*, pages 223–232, Washington, DC, USA, 2008. IEEE Computer Society.