Degree of Schedulability of Mixed-Criticality Real-time Systems with Probabilistic Sporadic Tasks

A.Jalil Boudjadar, Alexandre David, Jin Hyun Kim, Kim G. Larsen, Marius Mikučionis, Ulrik Nyman, Arne Skou
Institute of Computer Science, Aalborg University, Denmark

***Abstract*—We define the concept of degree of schedulability to characterize the schedulability and performance of mixed-criticality scheduling 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 hierarchical scheduling framework 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 triggering of the sporadic tasks is modeled separately from the scheduling system in order to achieve separation of concerns. The 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. We support uniform, exponential, Gaussian and any user-defined probability distribution. Finally, we show the applicability of our framework by analyzing an avionics case study.**

I. INTRODUCTION

Limited resources are a strong factor in the system setting in some embedded software application fields. Engineers could be interested, not only, in whether or not the system always meets its requirements, but also how it behaves with insufficient resources. Supplying a system with less resources than it requires may lead to a degradation of the quality of service. A certain level of degradation may be acceptable in a given setting and we thus consider it important to answer questions regarding schedulability with estimates of the quality instead of just providing a yes/no answer.

This paper presents a methodology to measure the degradation of the quality of service of a given system according to the resources it has been provided with. We introduce the degree of schedulability as a technique to analyze the degradation of the quality of service in the schedulability analysis of mixed-criticality hierarchical scheduling systems. The methodology is intended as an engineering technique used to compare system configurations regarding resources and task attributes.

A hierarchical scheduling system [10] is a component-based system encompassing global resources shared between the system components. The system workload consists of a set of tasks declared with a set of timing attributes such as period, deadline and execution time. Task dependencies and the partitioning of hierarchical resources make the schedulability analysis of hierarchical scheduling systems difficult. One way to deal with such a situation is to consider much simpler system specifications where only interfaces are considered during the analysis instead of the concrete workload behavior. One difficulty can be the choice of interfaces that sufficiently express components and tasks. Another aspect is how to compute the composition of interfaces. In recent years, increasing attention has been given to compositional analysis techniques, where system components can be analyzed separately.

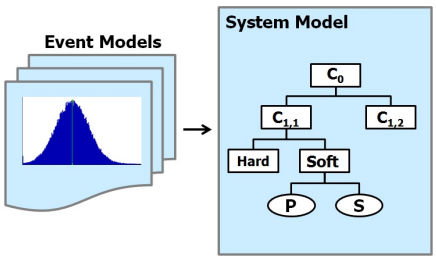


Fig. 1. Overview of framework setup.

The sporadic task model [3] [17] has received research attention [24] [23] over the years because of its usefulness in modeling recurring processes for hard-real-time systems. It has been originally introduced to model external interrupts of systems. Sporadic tasks can be used in the same way as periodic tasks to model periodic systems if we consider regular arrival times for tasks. However, the periodic task model cannot express the instantaneous and non-regular arrivals of sporadic tasks. It is obvious that it is not possible to guarantee the schedulability of real-time systems where sporadic tasks can occur arbitrarily frequently [3]. To alleviate this the sporadic task model operates with a minimal inter-arrival time. In many mixed-critical systems like tracking systems and automotive info-tainment systems, arrival times are adequately described by specific distribution functions. In this paper, we model real world events separately from the tasks. Events are modeled according to stochastic arrival patterns with an inter-arrival time that could be zero. Thus, we can determine the degree of schedulability of systems that are non-schedulable in the classical sense. Some classical approaches of the schedulability analysis of sporadic tasks treat the minimum inter-arrival time as the tasks period. This can seriously lead to a pessimistic over-estimation of the resource demand, and by that to a non-optimal resource utilization.

We consider a framework where the triggering mechanism of sporadic tasks is located outside the hierarchical scheduling system as illustrated in Fig. 1. Each sporadic task has its individual arrival pattern, which is characterized by a continuous probability distribution. In this paper, we are focusing on modeling and analyzing a single component at a time, with both sporadic and periodic real-time tasks. This analysis method fits within a larger compositional analysis framework [4], which analyzes a complete hierarchical scheduling system.

Our main contributions are:

- We introduce continuous probability distributions to model sporadic events that trigger the execution of sporadic tasks.
- We study the system schedulability and determine 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).
- We provide a framework including explicit environment models, which allows us to model and analyze mixed-criticality hierarchical scheduling systems.

Compared to treating the minimal inter-arrival time as a period, our method aims at providing both more realistic and optimistic resource estimates for sporadic tasks. The rest of the paper is structured as follows: Section II examines relevant related work, Section III introduces the compositional analysis framework. In Sections IV, V and VI 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 VII, and conclude in Section VIII.

II. Related Work

In this section we present related work with a specific focus on sporadic tasks. To the best of our knowledge, there is no previous related work which uses continuous probabilities to characterize the arrival patterns of sporadic tasks. The sporadic task model [3], [17], which is an extension of an earlier task model known as the Liu and Layland (LL) [13] task model has received immense research attention over the years. In [24], 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. The authors characterize each task 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 [23], 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. In fact, a task arrival T may deviate with a delay t if the probability of T to occur at instant t is greater than a certain threshold. 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 schedulable. 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. A concept similar to PoMD as introduced in this paper is given in [16]. The term "degree of schedulability" was first introduced in [18] to characterize the sum of response time delays from the individual deadlines. We define the concept 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.

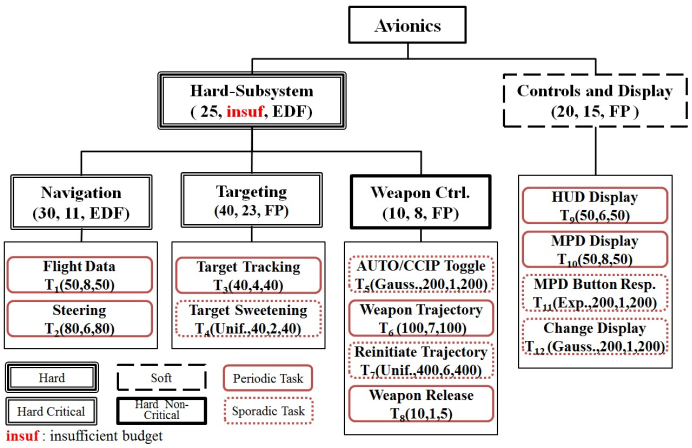


Fig. 2. Mixed criticality hierarchical scheduling system.

III. Compositional Analysis Framework for Sched°

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 again be internally organized as a set of components, giving the organization of the system a tree like structure. The use of temporal partitioning [20] between the 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. Such a grouping enables easier certification of the safety critical components when they have minimal communication with the non safety critical parts [19].

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 [14], [11], [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 VII.

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 real-time interface $\mathcal{I}$ [22] of a component $C(W, A)$ specifies the collective resource requirements that the workloads W performs under the scheduling policy A. $\mathcal{I}$ is simply given by a period p and a budget b in our framework.

In Fig. 2, we specify for each task (in parenthesis) the period or arrival pattern (probability distribution), 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 VII by distributing the components to different CPUs.

In a compositional schedulability analysis framework [6], [4], a hierarchical system is said to be schedulable if each component is schedulable. The analytical analysis approaches [12], [3], [15] compute whether or not a system is schedulable, according to a 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 [23], [2], [8], [4] a system is said to be schedulable if the error locations, stating the deadline violation, are unreachable.

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. We define a run π of a system $\mathcal{S}$ as 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; e_i indicate events (triggering, completing or preempting tasks) taking place with t_i time-units separating e_{i-1} and e_i resulting in a transition from state s_i to s_{i+1}. 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 upto time t;
- $\mathsf{Trig}_i^t(\pi) \in \mathbb{N}$ is the total number of triggerings of task i upto time t.

*Definition 3.1 (Percentage of Missed Deadlines (*PoMD*)):* The 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 upto 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. For the infinite set of infinite runs Runs, the PoMD can be formally defined using Lebesque integrals [9]. In practice a random variable which is defined using Lebesque integrals can be estimated using a finite sample of experiments. Thus, in our framework, the PoMD is defined over a finite set of runs Π of equal length by:

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

In fact, we compute the PoMD 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 [16].

*Definition 3.2 (Degradation of Quality of Service (*DoQoS*)):* The DoQoS of a task T_i over a finite set of runs Π is defined as:

$$\mathsf{DoQoS}^{T_i}(\Pi) = \begin{cases} 0 & \text{if } \lim_{t \to \infty} \sum_{\pi \in \Pi} \mathsf{Miss}_t(T_i, \pi) = 0 \\ \lim_{t \to \infty} \frac{\sum_{\pi \in \Pi} \mathsf{Overrun}_t(T_i, \pi)}{\sum_{\pi \in \Pi} \mathsf{Miss}_t(T_i, \pi)} & \text{Otherwise} \end{cases}$$

Similarly, the DoQoS of a component C over a set of time bounded runs Π is defined by the DoQoS of its workload W as:

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

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, i.e. upto time bound t each time the task T_i misses a deadline we accumulate the current delay by which the deadline is missed with the other delays made on the same run π. The value $overrun_j$ (Fig. 3) is the amount of time by which a specific deadline is missed. W is the component workload. Each item i in the workload can either be a task or a component. In this way, the DoQoS can be recursively calculated upto the system level. So for a simulation run, DoQoS is best explained as the average time that a deadline is exceeded by when it is missed.

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

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

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

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.

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.

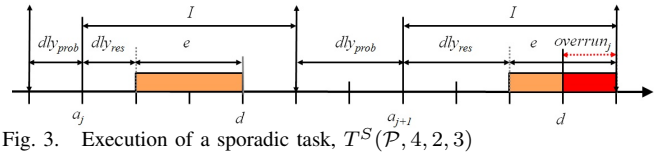


Fig. 3. Execution of a sporadic task, $T^S(\mathcal{P}, 4, 2, 3)$

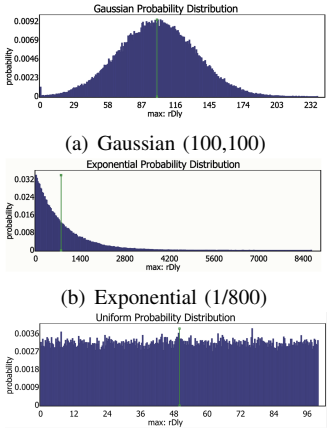
We reuse and adapt our previous work [4] for the schedulability analysis of hierarchical systems now extended with probabilistic sporadic tasks. As mentioned earlier, a workload $W = \{T_1, .., T_m\}$ is a set of periodic and sporadic tasks. Periodic tasks [13] $T^p(p, e, d)$ are commonly given at least 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 axe). 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.

IV. Continuous Probability Based Sporadic 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 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.

A. 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 100, μ=50 , and σ =50. 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 upto a maximum time where the probability drops to zero.



(a) Gaussian (100,100)

(b) Exponential (1/800)

(c) Uniform (1,100)

Fig. 4. Probabilistic arrival patterns.

B. Conceptual Sporadic Task Model

Our conceptual event model is shown in Fig. 5(a). When the 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(a)) starts in location Wait waiting for the triggering event (trigger?) and then moves to the location Run. Depending on whether or not the deadline of the task is missed, the task moves either directly to location Wait or to location MissedDeadline. If the task finishes its current execution after missing a deadline, the overrun will be measured (used for calculating the DoQoS) before moving to the location Wait. Uppaal implementations of our probability based sporadic task model are given in Section V, whereas the models of triggering events for different probability distributions are given in Section V-A.

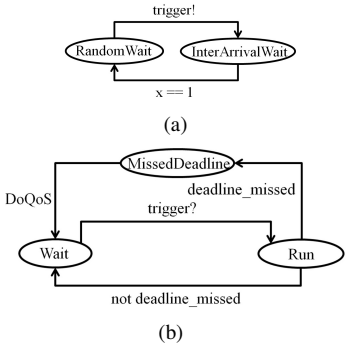


Fig. 5. Conceptual model of a sporadic task and its triggering event.

V. Analysis Models for the Degree of Schedulability

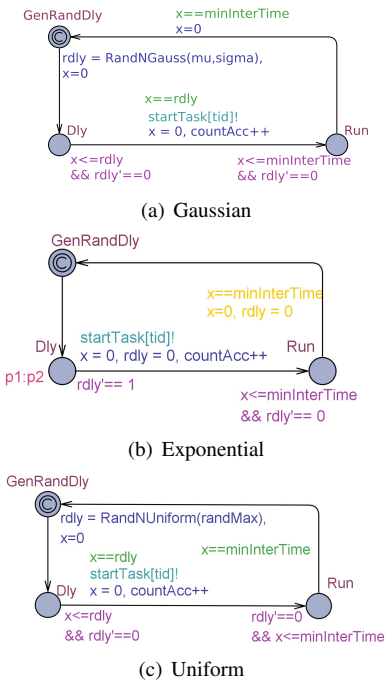
For our compositional analysis framework, the hierarchical scheduling systems and their analysis elements consist of environment models, scheduling models, resource model, and task models.

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 [7] 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 upto a specific confidence level, given as an analysis parameter [5].

A. Environment Model



(a) Gaussian

(b) Exponential

(c) Uniform

Fig. 6. Sporadic event models.

Fig. 6 shows the actual Uppaal models that generate the probability distributions shown in Fig. 4. These also correspond to the conceptual triggering event given in Fig. 5(a). The models in Fig. 6 all follow the same structure with a committed initial location GenRandDly in the upper left corner. For the Gaussian and Uniform distributions, the templates start by determining a random delay based on the probability distribution specified in terms of a density function together with other parameters. After selecting a random delay and moving from location GenRandDly to Dly, the template waits until the selected amount of delay has elapsed. The amount of time spent in Dly corresponds to dly_{prob} in Fig. 3. Upon moving to location Run, the event template triggers the corresponding task by a communication on a broadcast channel (e.g. startTask[tid]!). After waiting in location InterArrival for the duration of one inter-arrival time (I in Fig. 3), the event model joins the initial location. Functions *RandNGauss* and *RandNUniform*, of Fig. 6(a) and Fig. 6(c) respectively, return floating point numbers because Uppaal has recently been extended with the data type **double**.

B. Resource model

The resource model of this paper is a periodic one, which provides a specific amount of resources to a set of tasks or components [21]. The resource model is given by a stochastic supplier, which supplies a resource allocation non-deterministically over supplier's period.

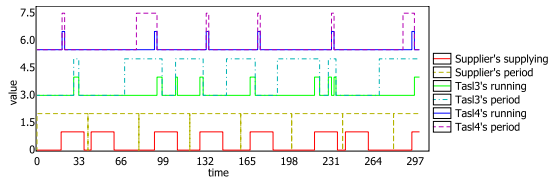
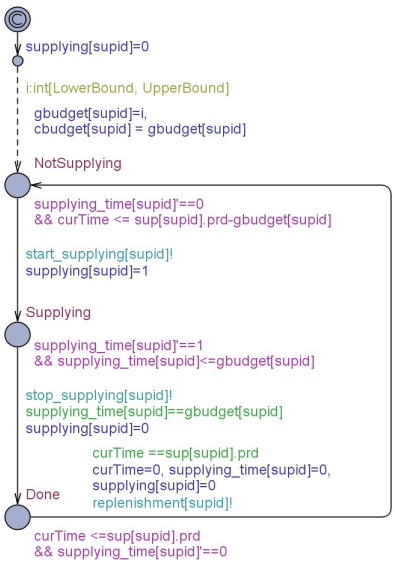


Fig. 7. Supplier and Task Execution.

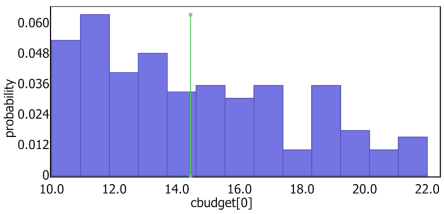
Fig. 7 shows supplier's supplying and tasks' running of T_3^p and T_4^s. In this setting, T_4^s has priority over T_3^p, and executes sporadically over a uniform distribution. Thus, the execution period of T_4^s is irregular. The supplier at the bottom is supplying non-deterministically so the supplying is also irregular within the period. The detailed explanation can be found in our previous paper [4]. In order to estimate the sufficient budget of a supplier that makes the workload of a component schedulable, we present another stochastic supplier as shown in Fig. 8.(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. 8.(b) shows the estimated budget numbers that makes the component of T_3^p and T_4^s non-schedulable, and it can be concluded that 23 is the minimum budget for the component.

C. Task Models

For 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 to run until the end of simulation time while measuring PoMD and DoQoS. Fig. 9 shows the soft



(a) Budget estimation model



(b) Budgets causing deadline miss

Fig. 8. Budget estimation

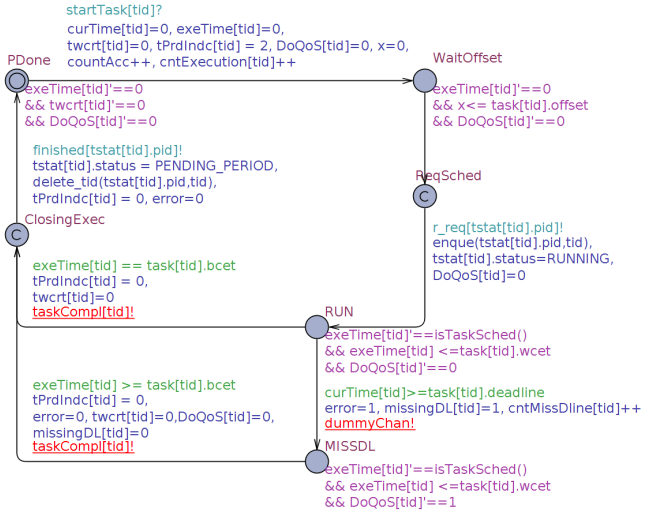


Fig. 9. Soft real-time sporadic task

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 (resource usage) respectively. The clock twcrt[tid] measures the worst-case response time. The variables cntExecution[tid], cntMissDline[tid], and DoQoS[tid] are used to calculate PoMD and DoQoS with PoMD calculator in Fig. 10 and the following queries:

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

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

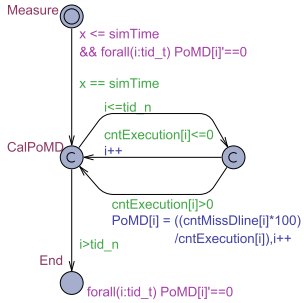


Fig. 10. PoMD calculator

For every simulation (simNum) of which time is upto 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 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 simulation trace.

VI. 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 (FPS) 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 budget for the component as described in Section V.
2) Analyze the Sched° for the estimated and lower budgets.

To estimate the budget of a component, we use the budget estimation template shown in Fig. 8(a) and the following query:

Pr[cbudget[rid]<=randomBudget](<> gClock>=simTime and error)

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 1,000 traces and a simulation time of more than 100,000 time units we see that the results stabilize.

A. Analysis Results

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

TABLE I
THE DEGREE OF SCHEDULABILITY OF TASKS UNDER PERIODIC EVENTS

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

TABLE II
THE DEGREE OF SCHEDULABILITY OF TASKS UNDER LACK OF BUDGET

Component ((40, 18), FP)	PoMD	DoQoS
T_3^P(40, 4), T_4^s(Periodic, 40, 2, 40)	1.532 ± 0.191 0	3.034 ± 0.635 0

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 II. 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 II 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 arrival patters 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. III 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 IV shows the Sched$^\circ$ for the two tasks given a uniform probability distribution for triggering the sporadic task T_4^s. Table V shows the results of our analysis when using different Gaussian distributions, all with a mean value μ of 10 and different deviations σ.

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.

VII. CASE STUDY

As a case study to show the applicability of our analysis framework, we analyze the schedulability of an avionics system [14], [11], [4]. We use the same timing specification as [11], whereas the system structure depicted in Fig. 2 follows

TABLE III
THE 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	0.350 ± 0.435	3.630 ± 0.672
	1/1000	0.363 ± 0.434	4.950 ± 0.840
	1/10	0.488 ± 0.049	8.404 ± 1.186
T_4^s(Exp., 40, 2, 40)	1/100,000	0.233 ± 0.284	0.022 ± 0.040
	1/1000	0.267 ± 0.035	2.187 ± 0.435
	1/10	0.072 ± 0.016	6.766 ± 0.636

TABLE IV
THE DEGREE OF SCHEDULABILITY OF TASKS UNDER UNIFORM DISTRIBUTION.

Component ((40, 18), FP)	PoMD	DoQoS
T_3^p(40, 4), T_4^s(Unif., 40, 2, 40)	0.497 ± 0.051 0.052 ± 0.015	7.564 ± 1.126 6.515 ± 0.688

TABLE V
THE DEGREE OF SCHEDULABILITY WITH GAUSSIAN DISTRIBUTION

Component ((40, 18), FP)	(μ, σ) of T_4^s	PoMD	DoQoS
T_3^p(40, 4)	(10,10)	2.000 ± 0.653	3.677 ± 2.411
	(10, 8)	1.440 ± 0.559	3.024 ± 1.914
	(10, 5)	1.640 ± 0.608	3.260 ± 1.966
	(10, 1)	0.400 ± 0.579	3.943 ± 2.611
T_4^s(Gauss., 40, 2, 40)	(10,10)	0.350 ± 0.255	1.687 ± 1.339
	(10, 8)	0.490 ± 0.326	1.135 ± 1.076
	(10, 5)	0.390 ± 0.249	0.551 ± 0.672
	(10, 1)	0.600 ± 0.324	0.704 ± 0.768

the description given in [14]. 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 the case study. Table VI summarizes both architecture and timing attributes of the components from the avionics system that we consider.

The avionics system is a mixed-criticality application, where we mainly considered 7 periodic and 5 sporadic tasks 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 that task. We characterize the arrival times of non-critical sporadic tasks by different probability distributions where the delay generated by such a distribution is relatively proportional to the minimum inter-arrival time of the corresponding task. 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 VII 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 a minimum sufficient budget depending on the quality of service required by the system. Over this case study, we have quantified the components schedulability according to the budgets, hard and soft real-time requirements.

VIII. 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$^\circ$) 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

TABLE VI
GENERIC AVIONICS COMPONENTS AND TASKS

Component	Criticality	T_i	e_i	p_i	d_i	Importance
Navigation	Hard critical	Aircraft flight data(T_1^p)	8	50(55)		critical
		Steering(T_2^p)	6	80		critical
Targeting	Hard critical	Target tracking(T_3^p)	4	40		critical
		Target sweetening(T_4^s)	2		40	critical
Weapon Control	Hard non-critical	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 VII
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

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 degree of schedulability of an avionics case study which was previously shown to be unschedulable [14], [11], [4].

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 *In Proceedings of the 11th Real-Time Systems Symposium*, 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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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 *Lecture Notes in Computer Science*, pages 80–96. Springer, 2011.

[10] Z. Deng and J. W. s. Liu. Scheduling real-time applications in an open environment. In *in Proceedings of the 18th IEEE Real-Time Systems Symposium, IEEE Computer*, pages 308–319. Society Press, 1997.

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

[12] 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.

[13] 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.

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

[15] J. Lorente, G. Lipari, and E. Bini. A hierarchical scheduling model for component-based real-time systems. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8 pp.–, 2006.

[16] S. Manolache, P. Eles, and Z. Peng. Optimization of soft real-time systems with deadline miss ratio constraints. In *Real-Time and Embedded Technology and Applications Symposium, Proceedings. RTAS 2004. 10th IEEE*, pages 562–570, 2004.

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

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

[19] 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.

[20] 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.

[21] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS*, pages 2–13. IEEE Computer Society, 2003.

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

[23] 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*, June 2012.

[24] 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 the 2008 Euromicro Conference on Real-Time Systems*, ECRTS '08, pages 223–232, Washington, DC, USA, 2008. IEEE Computer Society.