Widening the Schedulability Hierarchical Scheduling Systems

Boudjadar, Jalil; David, Alexandre; Kim, Jin Hyun; Larsen, Kim Guldstrand; Mikučionis, Marius; Nyman, Ulrik; Skou, Arne

Published in:
Formal Aspects of Component Software

DOI (link to publication from Publisher):
10.1007/978-3-319-15317-9_14

Publication date:
2015

Document Version
Accepted author manuscript, peer reviewed version

Widening the Schedulability of Hierarchical Scheduling Systems*

Abdeldjalil Boudjadar, Alexandre David, Jin Hyun Kim, Kim. G. Larsen, Marius Mikučionis, Ulrik Nyman, Arne Skou

Computer Science, Aalborg University, Denmark

Abstract. This paper presents a compositional approach for schedulability analysis of hierarchical systems, which enables to prove more systems schedulable by having richer and more detailed scheduling models. We use a lightweight method (statistical model checking) for design exploration, easily assuring high confidence in the correctness of the model. A satisfactory design can be proved schedulable using the computation costly method (symbolic model checking). In order to analyze a hierarchical scheduling system compositionally, we introduce the notion of a stochastic supplier modeling the supply of resources in each component. We specifically investigate two different techniques to widen the set of provably schedulable systems: 1) a new supplier model; 2) restricting the potential task offsets. We also provide a way to estimate the minimum resource supply (budget) that a component is required to provide.

1 Introduction

The use of hierarchical scheduling systems is a new trend in software architecture that integrates a number of individual components into a single system running on one execution platform. Hierarchical scheduling systems have reached a maturity where they are used in real automotive and space systems [14,9]. A class of analytical methods has been developed for hierarchical scheduling systems [19,18]. Due to their rigorous nature, analytical methods are easy to apply once proven correct, but very hard to prove correct. They also suffer from the abstractness of the models; they do not deal with any detail of the system behavior and thus grossly overestimate the amount of needed resources. Model-based methodologies for schedulability analysis [7,5,9] allow modeling more detailed and complicated behavior of individual tasks, relative to analytical methods while powerful analysis tools can be applied. Profiting from the technological advances in model checking, we provide a model based methodology for the schedulability analysis of hierarchical scheduling systems. We model tasks, resources, schedulers and suppliers as Parameterized Stopwatch Automata (PSA) [8]. The models can be quickly analyzed using statistical methods (Uppaal

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

SMC), which provide guarantees with a selected statistical margin. Once a satisfying model design has been found, the model can be analyzed using symbolic model checking (UPPAAL). Our approach aims at increasing resource utilization by 1) adjusting task offsets relative to the component period; 2) providing a new supplier model where the supply of resources is delayed as much as possible according to task requests. Our methodology also has the advantage that it is possible for system engineers to update the models in order to have a more realistic analysis of the system. In this way, they can utilize detailed knowledge of the system that they are working with; something that cannot be achieved with a classical analytical approach.

An example of a hierarchical scheduling system is depicted in Fig. 1. It includes two top level components Controls and Display and Nav. Ctrl scheduled according to the Earliest Deadline First (EDF) policy. Each component is characterized by timing requirements consisting of period and execution time (e.g. $(10, 6)$ for Nav. Ctrl). The attributes of tasks are similar to the ones of components. Task deadlines are the same as the task periods.

According to the CARTS tool [18], the hierarchical scheduling system of Fig. 1 is not schedulable. However using the specific approach shown in this paper, this system can be shown to be schedulable using different offset parameters and/or a new supplier model.

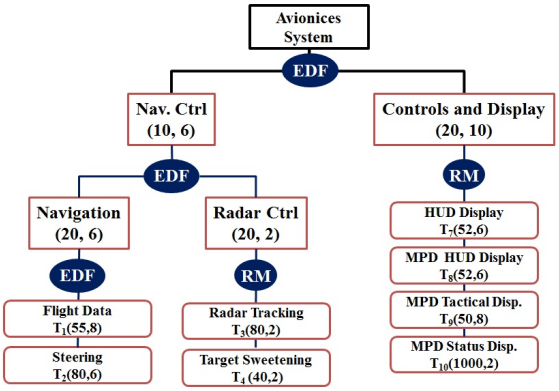


Fig. 1. A hierarchical scheduling systems.

Symbolic model checking offers absolute certainty that the verified properties are correct. However, it suffers from state space explosion and undecidability, thus some models might not be feasible to check and others will take a long time to verify. Statistical model checking provides high confidence in the results that one obtains in contrast to symbolic model checking.

This paper presents a methodology for performing compositional schedulability analysis of hierarchical scheduling systems. The general methodology consists of using a light weight statistical method and a costly but absolute certain symbolic method that operates on identical models. Design space exploration can be carried out at low cost using the statistical model checking in order to determine optimal system parameters that could be impossible to find using classical analytical methods. The use of automata and statistical model checking enables a larger class of tasks and resource supply models to be analyzed compared to the conventional real-time analytical method while still being efficient. Allowing designing systems based on a confidence level can be highly beneficial for soft real-time systems. In order to verify the schedulability of the system found us-

ing statistical methods we use the symbolic method on the final system design. The end goal of the methodology is to widen the set of concrete systems that can be proved schedulable. We instantiate our methodology using PSA models, Uppaal SMC and Uppaal. Using the general methodology and our specific tools, we investigate two concrete techniques that affect the resource utilization. The first technique decreasing the needed resources relies on an update of the stochastic supplier model, such that it defers resource consumption until a task is ready. Secondly, we describe a way of giving a potential decrease in the needed resources of a component by ensuring more synchronicity between the initial offset of tasks and the starting point of the parent component's period. In order to enable compositional verification, we introduce the concept of a stochastic supplier model. We evaluate our methodology by comparing our results to the ones obtained using the state of the art tool CARTS [18]. Our verification results are consistent with the results obtained from CARTS. In one particular case, we have uncovered a significant difference between CARTS and our methodology. After further investigation, this turned out to be an error in the implementation of CARTS and not the underlying theory. This has been confirmed by the developers of CARTS. When checking the schedulability of a system, our tools can prove the non-schedulability by means of a counterexample. Our methodology, which builds on previous work in [5], is very scalable because both design and analysis are compositional.

The rest of this paper is organized as follows: Section 2 describes our methodology. Section 3 presents our compositional modeling and analysis of hierarchical scheduling systems using Uppaal and Uppaal SMC. Section 4 describes two techniques to improve resource utilization. Section 5 compares our results with a state of the art tool. Section 6 describes related work. Section 7 is a conclusion.

2 Methodology and Preliminaries

This paper presents a general methodology, a specific approach and investigates two concrete techniques. The methodology could be instantiated using any modeling formalism supporting both a lightweight statistical analysis and a more costly formal verification. In this paper the methodology is instantiated as a specific approach using Parameterized Stopwatch Automata (PSA) together with the verification suite Uppaal SMC and Uppaal.

The two concrete techniques for enhancing the resource utilization are described in Section 4. Fig. 2 shows a graphical conceptual representation of different sets of systems that different methods can show to be schedulable. Systems that are easily proven schedulable using classical analytical approaches can also be proven correct using symbolic model checking. Systems that can be shown, with a high degree of certainty, to be correct using statistical model checking (SMC) cannot always be proven to be correct using symbolic model checking due to state space explosion. In the same way some complex systems that are analyzable using model checking cannot be proved correct using analytical approaches

[7]. Our methodology consists of exploring system models with different sets of parameters ($S(P_i)$) searching for a realistic configuration that optimally satisfies the requirements. These experiments are performed using SMC with a high confidence level. Using SMC one can easily and interactively obtain either a high degree of confidence that the model is correct or a counterexample showing an error trace. When a satisfying final configuration has been found the system can be proven to be schedulable using symbolic model checking. In very rare cases an error could be found at this stage, but this is highly unlikely due to the confidence levels obtained using SMC.

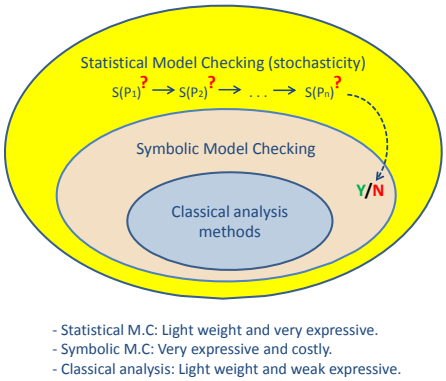


Fig. 2. Classes of systems that different methods can prove schedulable.

2.1 Statistical Model Checking

We use both SMC and classical symbolic model checking techniques to analyze the schedulability of hierarchical scheduling systems. The UPPAAL verification suite provides both symbolic and SMC. The models which in practice can be analyzed statistically, using the UPPAAL SMC verification engine, are larger and can contain more features.

Meanwhile, SMC provides much faster responses. The speed of such responses depends entirely on the degree of certainty that one wants to obtain. The reason is that SMC consists in running a sufficiently high number of simulations of the system under analysis. The advantage of SMC resides in: 1) SMC provides a quick response in terms of less than a minute. This is also true in the case of non-schedulability were SMC produces counter-example witnesses; 2) SMC enables quantitative performance measurements instead of the Boolean (true, false) evaluation that symbolic model checking techniques provide. We can summarize the features of UPPAAL SMC that we use 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 preemptable tasks.
- Simulation and estimation of the value of expressions, `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 for a given simulation time and/or number of runs specified by `bound`.

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

2.2 Classical compositional framework

In this section, we provide the formal basis of our model-based compositional analysis approach. In fact, our theory conforms with the formal basis given in the compositional framework [20] for hierarchical scheduling systems.

A scheduling unit C is defined as a tuple (W, A) where W is a workload, consisting of a set of tasks $T_i = (p_i, e_i)$, and a scheduling policy A. Each task $T_i = (p_i, e_i)$ has timing attributes in the form of a period p_i and an execution time e_i. Task deadlines are the same as periods. The scheduling unit C (Fig. 3) is given a collective timing requirement $\Gamma = (\Pi, \Theta)$ called interface, where Π is a period and Θ is a budget for the component. The collective timing requirement Γ is a representative of all timing requirements of tasks constituting the workload W. In the compositional framework [20], the schedulability of the system is checked by the following relation: $\mathbf{dbf}(W, A, t) \leq \mathbf{sbf}_{\Gamma}(R, t)$ (1) where t is a time interval. In this relation, the demand bound function $\mathbf{dbf}(W, A, t)$ computes the maximum amount of resource required by W under scheduling algorithm A during t time units. The supply bound function $\mathbf{sbf}_{\Gamma}(R, t)$ returns the minimum amount of resource that the resource model R allocates during a time interval t according to the resource requirement Γ. The system is said to be schedulable under the EDF policy if and only if it satisfies relation (1) for any value t.

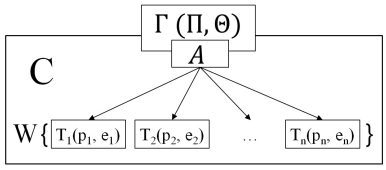


Fig. 3. Component and sub-tasks in a compositional framework

2.3 Conceptual Models of our Approach

In our model-based approach, we realize the compositional framework in the form of PSA models. We implemented the **dbf** as a set of tasks together with a scheduling algorithm, while the **sbf** is implemented by the supplier model. Such a supplier model (R_{PSA}) represents the classical resource model R in accordance with the contract Γ. The time when the workload can use resources follows from the scheduling algorithm, and is also constrained by the resource model R_{PSA}.

PSA supports stopwatches, which are clocks that can be stopped and resumed without a reset. The modeling formalism allows for having different rates of progression for stopwatches, but we only utilize the values 1 (running) and 0 (stopped). In our PSA models, the stopwatch is used to express the preemption of a task's execution. The execution of a task is preempted, i.e. the associated clock stops, in two cases: when it is preempted by a higher priority task or when any of the needed resources is not provided by the supplier.

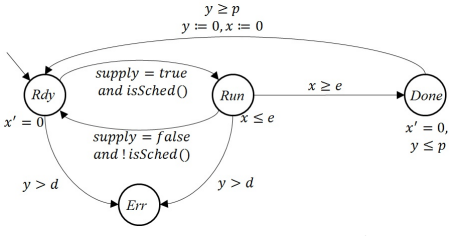


Fig. 4. Task model in PSA

Fig. 4 is a conceptual model of a task, which we will realize using PSA in Section 3. The clock x stops progressing in the locations where its derivative x' is set to 0. The clock x keeps progressing at other locations. The task starts at the initial location *Rdy* and moves to *Run* when the two following conditions hold: the task is scheduled to use a resource *pid*, ($isSched(pid)$) and there is a supply of necessary resources ($supply = true$). The clock x measures the execution time of the task while it is in the location *Run*. If either of the two conditions is false at the location *Run*, the task moves back to the location *Rdy*. The task stays at location *Run* until the stopwatch x reaches the execution time e, and then jumps to location *Done* delaying until the next period. A task joins the error location *Err* when its deadline d is missed ($y > d$). Throughout this paper we keep the assumption that $e \leq d \leq p$.

In the following, we relate the analytical view of the supply bound function and the resource model with the way they are implemented as a supplier model in our approach. We use the Periodic Resource Model (PRM) [20] as an example.

Fig. 5 shows an example of the resource allocations of the PRM which guarantees the resource requirement $\Gamma = (\Pi, \Theta)$ where Π is 5 and Θ is 2. Σ represents a delay until the next period and Λ is the delay located between the beginning of a period and the start of supply (slack time). The resource allocation in PRM does not need to be synchronized with the execution of tasks, thus Λ is deviated between 0 and $\Pi - \Theta$. The consecutive delay of Σ and Λ is denoted by Ψ, where no resource is allocated at all.

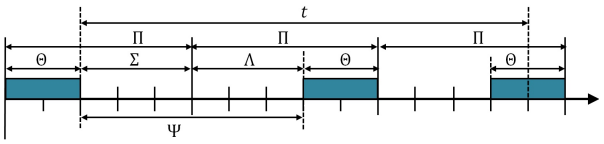


Fig. 5. Resource allocations of Periodic Resource Model

Property 1. The interval Ψ varies between 0 and the maximum consecutive delays of Σ and Λ, i.e. $0 \leq \Psi \leq 2(\Pi - \Theta)$.

The delay Λ varies between 0 and $\Pi - \Theta$ by the definition of the PRM. After a supply of Θ time units, the delay Σ is deviated between 0 and $\Pi - \Theta$. Consequently, $0 \leq \Lambda + \Sigma \leq 2(\Pi - \Theta)$.

The supply bound function $\mathbf{sbf}_\Gamma(R, t)$ based on the PRM is formulated as:

$$\mathbf{sbf}_\Gamma(PRM, t) = \left\lfloor \frac{t - (\Pi - \Theta)}{\Pi} \right\rfloor \cdot \Theta + \epsilon_s \tag{2}$$

$$\epsilon_s = \max\left(t - 2(\Pi - \Theta) - \Pi \left\lfloor \frac{t - (\Pi - \Theta)}{\Pi} \right\rfloor, 0\right) \tag{3}$$

Our PSA model for the PRM (R_{PSA}) is designed to generate all possible allocations of resources in compliance with $\Gamma = (\Pi, \Theta)$.

One can remark that our resource model supplies the whole budget non-preemptively in one chunk, however according to [20] if one considers only worst cases, both preemptive and non-preemptive resource models provide the same analysis results. Thus we will use a non-preemptive supplier model (Fig. 6) both in this conceptual description as well in the computation models. Fig. 6 shows

the conceptual model of PSA resource model. In this model, the variable *supply* represents the resource allocation, which is a shared variable with the task model. Thus the supply is only enabled for Θ time units within the period Π. The location *Rdy* of R_{PSA} corresponds to the delay Λ in PRM of Fig. 5. This represents a situation where a new period started but the resource allocation has not been started. The location *Sup* corresponds to Θ where the resource is allocated, and *Done* corresponds to Σ where the resource model waits for the next period.

In order to realize a compositional approach, our resource model R_{PSA} does not synchronize with the execution of tasks similarly to the resource allocation of PRM. Thus the resource model can stay at the location *Rdy* up to $\Pi - \Theta$ or immediately move to the location *Sup*. This resource model is designed to generate all possible resource allocations including the maximum duration of no resource allocation Ψ.

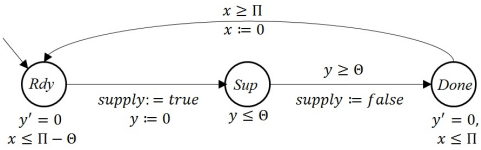


Fig. 6. Conceptual PRM model in PSA notation

3 Compositional Analysis Approach

In this section we present concrete PSA models based on the conceptual models presented in the previous section. The implementation contains a resource model (supplier template) and a task template. The scheduling policy is modeled as a separate PSA template, which is represented as a parameter when instantiating the concrete system. This increases the reconfigurability of our approach. We have modeled three different scheduling policies EDF, RM and FIFO but we only use EDF and RM in the experiments. Moreover, all tasks in the system are instances of the same Task template but with different parameters.

3.1 Stochastic Periodic Resource Model

In [21] the resource allocation by the supplier does not necessarily synchronize with tasks periods. That is, if the workloads of tasks start at time t_w and the resource allocation begins at time t_r then [21] assumes that t_w is not necessarily equal to t_r. This assumption leads to a stochastic periodic supplier model, where the supply of resources follows a uniform probability distribution within the supplier period. Thus, we impose no obligation on the scheduler at the parent level of providing resources at a certain point in time. We only consider

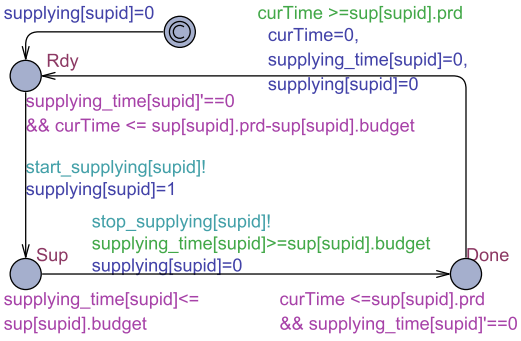


Fig. 7. Stochastic periodic resource model

that the whole budget should be provided before the end of the supplier period. Fig. 7 shows the supplier template, which communicates with the other templates through two output broadcast channels (start_supplying and stop_supplying) and a shared variable (supplying). These channels are used in the template Task to keep track of the resource supply. The initial location of the Supplier template is marked with double circles. Such a location is also marked with a "c" which indicates that it is a committed location, this leads the supplier to move instantaneously to the next location Rdy. Slack time is the maximum amount of time that can elapse before the supplier starts to supply resources. It is used in several places of Supplier and written as $sup[supid].prd - sup[supid].budget$. The location Rdy has an invariant consisting of a conjunction of two parts. The first part $supplying\_time[supid]' == 0$ means that the clock representing the supplied amount of resources does not progress while the template resides in the location Rdy. The second part $curTime <= sup[supid].prd - sup[supid].budget$ ensures that once $curTime$ has reached the end of the slack time the template leaves the location Rdy.

At some point in time between time zero and the slack time, the supplier moves to the location Sup. In this location, the progress rate of the clock supplying_time[supid] is set to 1, signifying that the supplier keeps supplying resources. While the likelihood of delays happening at location Rdy would be the same, we treat such a non-deterministic wait via a uniform probability distribution when performing statistical analysis. One can notice that non-determinism motivates the use of statistical model checking. The supplier can not provide more resources than budgeted and will move to the location Done when it has provided the needed resources. At the start of the next period, the supplier moves to the location Rdy.

3.2 Task Model

The task model in this paper has various execution attributes, such as the worst case execution time, deadline, initial offset and regular offset. Thus, our framework can easily be used to describe complicated hierarchical scheduling systems. Formally, a task within the workload $W = \{T_1, T_2, ..., T_n\}$ is defined by

- pri: Task priority.
- initial_offset: The offset of the initial period of the task.
- offset: The offset from the beginning of each period until task release.
- bcet: Best-case execution time.
- wcet: Worst-case execution time.
- preemptable: Whether a task is preemptable.
- tid: Task identifier.

Fig 8 shows the PSA task template. It begins its execution by waiting, non-deterministically, for an amount of time up to the initial offset (initial_offset). Using this parameter, we can adjust the synchronicity of the task execution with the supplier. This will be further explained in section 4. The stopwatch

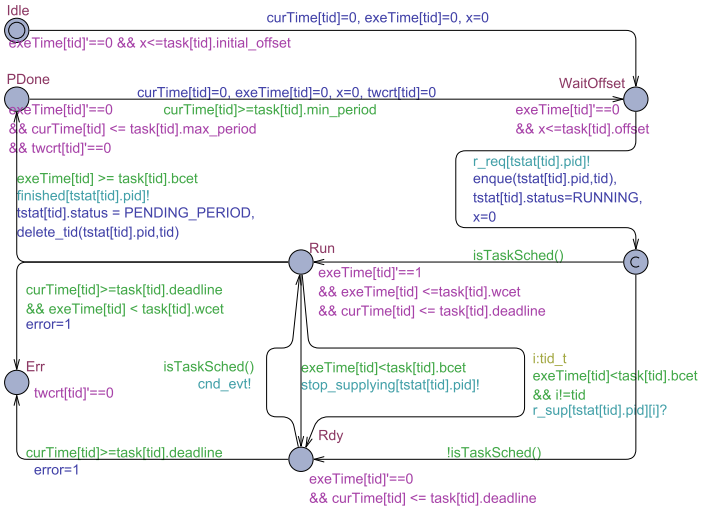


Fig. 8. Task model

twcrt is used to measure the worst-case response time of the task. The behavior of the task model consists mainly in checking whether a resource is available or not, by checking the supplier status $supplying[tstat[tid].pid]$, which is done inside the function isTaskSched().

The execution of a task can always be suspended whenever the supplier stops providing the task's requested resource. A task may travel several times between location Ready and location Run due to preemption by other tasks in the same component. This preemption is implemented at the level of the scheduling policy. Once the execution of a task is achieved within its deadline, the task moves to the location PDone, before starting the next period. If a task misses its deadline it moves to the location Err where it assigns 1 to the global variable error. This variable is used when analyzing the schedulability. The models used are available at `http://people.cs.aau.dk/~ulrik/submissions/721641/models.zip`. The top level system is formed by a parallel composition of component suppliers together with a scheduling policy. The schedulability of the top level system is performed according to [5]. The PSA models of scheduling algorithms are not included in the paper because their behavior is trivial, but they are provided in the above link.

3.3 Automated computation of the supplier budget

We have automated a technique for directly estimating the supplier budget. Such an automation is realized by adding a helper template to the system and exploiting the expressiveness of the UPPAAL SMC query language. Fig. 9 shows the modified initial states

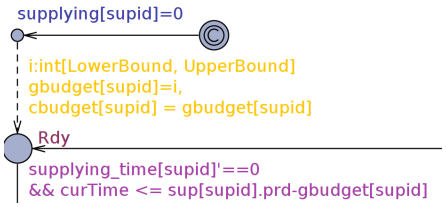


Fig. 9. Modified resource model

of the Supplier template. We do not show the helper because of its simple behavior consisting of one transition, storing a value at the end of the simulation time. Between the initial location and the Rdy location of the modified supplier template, the budget is assigned a uniformly distributed random value between 0 and the period of the supplier, given in the template as the two constants LowerBound and UpperBound. The minimal budget can be found by searching for every budget value which makes the system non schedulable. To this end, we use the following query:

$$\text{Pr[cbudget[1]} <= \text{rbudget]} (<> \text{globalTime} >= \text{simTime and error}) \quad (4)$$

where cbudget[1] is the budget candidate for the supplier in a given run, rbudget is a constant value that is larger than any of the potential budgets, and globalTime is the current simulation time (clock). In the helper template, cbudget[1] is assigned a value larger than rbudget when the simulation has executed for simTime time units. Thus, this query finds every number between 0 and the supplier's budget for which UPPAAL SMC finds a run where a task misses its deadline before the expiry of simTime, i.e. globalTime>=simTime.

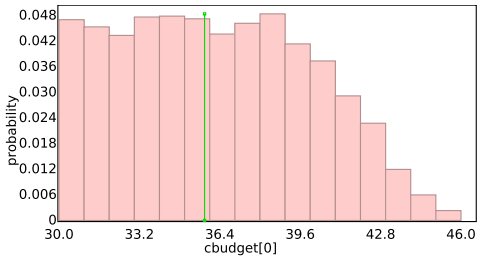


Fig. 10. Probability distribution of supplier's budgets that make component S_2 of Table 2 non schedulable under EDF

Fig. 10 and 11 show the probability distributions of budgets that UPPAAL SMC produces after checking the system using query (4). Fig. 10 shows that for every potential budget between 0 and 45 a run where a deadline has been missed was found. In other words, a budget greater than 45 *can* make the system schedulable.

Given a budget, the schedulability of a component and its workload can be checked using the following query:

$$\text{Pr}[<= \text{simTime}](<> \text{error}) \quad (5)$$

Such a query computes the probability of a component to finally (<>) reach an error, where simTime is a simulation time and error is a global variable indicating whether a task has missed its deadline or not.

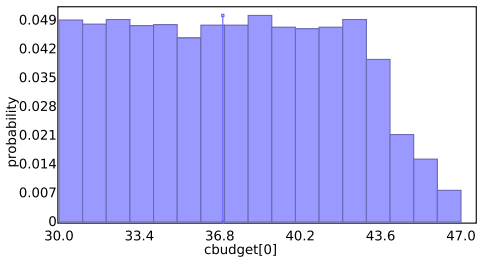


Fig. 11. Probability distribution of supplier's budgets that make component S_2 of Table 2 non schedulable under RM

By using the budget found via query (4) as a parameter value for component S_2 of Table 2 when checking query (5), we can see that this indeed makes component S_2 schedulable under EDF. Fig. 11 is the estimation results of the

same component under RM, showing that a supplier budget greater than 47 can make component S_2 schedulable. By using query (4), we can obtain a very good estimate for the minimal budget. In practice one might still need to check two or three values using query (5) after having applied query (4). Once a candidate budget is strongly determined, we apply symbolic model checking to be absolutely certain.

4 Enhancement of Resource Utilization

This section presents two techniques for enhancing the utilization of a resource: 1) introducing a new supplier model; 2) making tasks more synchronous with their suppliers by adjusting tasks initial offset.

4.1 Synchronous Periodic Resource Model

In order to increase the resource utilization by trying to avoid supplying resource when it is not needed, i.e. no waiting task, we introduce a new supplier model. The new supplier relies on delaying the resource supply, while no task is requesting resource, until a task request is received. Such a delay is up to the component slack time (period-budget).

Fig 12 depicts the PSA template that implements our new supplier model. Once started, the supplier joins location Rdy and keeps waiting while the slack time is not expired. Such a constraint is implemented by the location invariant curTime $\leq$ sup[supid].prd-sup[supid].budget, where prd and budget are respectively the period and budget of the supplier. One can remark that, at location Rdy, the stopwatch measuring the resource supply is not progressing (supplying_time[supid]'==0). Non deterministically, the supplier moves from location Rdy to the location Sup by either receiving a task request (guard isReq() over the crooked edge), or once the slack time is expired (guard curTime$\geq$sup[supid].prd-sup[supid].budget over the vertical edge).

At location Sup, the supplier keeps supplying resource before moving to the location Done. Such a location can be reached once the whole budget is supplied. From location Done and once the period is expired, the supplier joins location Rdy to start new period and resets its clocks.

Table 1 shows the gain in resource utilization obtained when applying the new supplier model. At the first stage, using the periodic resource model PRM, we compute the component budgets of the avionics

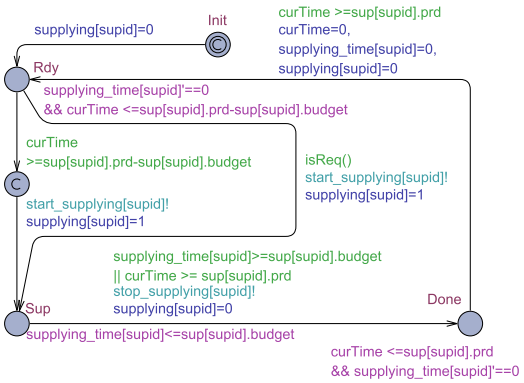
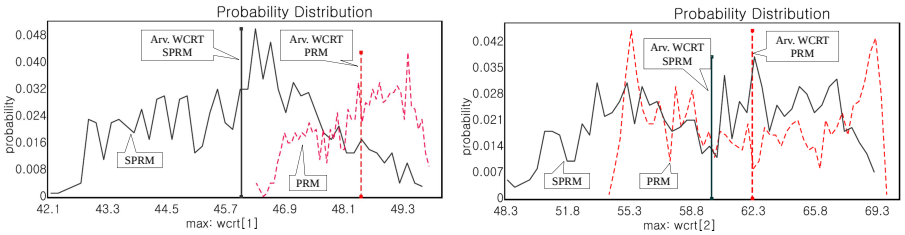


Fig. 12. Synchronous Periodic Resource Model in PSA

system we mentioned earlier. The component budgets obtained via CARTS (2nd column) and UPPAAL SMC (3rd column) are identical; (10,6) (20,6) (20,2) (20,10). By replacing PRM with our new supplier model, we recompute the minimum budgets making the avionics components schedulable using UPPAAL SMC (4th column). For components Nav.Radar Ctrl and Navigation, the budgets are decreased to 5 in each case, with a gain of 17% thanks to our new supplier model. We have checked and confirmed such new budgets using the UPPAAL symbolic model checking (MC).



(a) Flight data (b) Steering

Fig. 13. Probability distributions of WCRT of flight data and steering tasks. The queries $E[<= 100000; 1000](max : wcrt[1])$ and $E[<= 100000; 1000](max : wcrt[2])$ are used to generate the probability distributions using UPPAAL SMC.

Table 1. Resource utilization comparison

Analysis Tool	CARTS	SMC	SMC & MC
Resource Model	PRM	PRM	Synchronous PRM
Nav. Radar Ctrl	(10, 6)	(10, 6)	**(10, 5)**
Navigation	(20, 6)	(20, 6)	**(20, 5)**
Radar Ctrl	(20, 2)	(20, 2)	(20, 2)
Control & Display	(20, 10)	(20, 10)	(20, 10)

In order to evaluate the effects of introducing a new supplier model, we have made a statistical experiment using the two tasks in the component Navigation. Fig. 13 shows the probability distributions of WCRT using two different resource models. This shows that the average WCRT is enhanced using the new supplier model SPRM. There is no significant difference in the actual WCRT. By using these plots we can see how much a hierarchical system can be improved by using different system settings. The fact that we can easily generate such plots also shows the versatility of a model and simulation based approach.

4.2 Offset Manipulation

Our second technique consists in limiting the initial offset for the arrival of all tasks. Explicitly including offsets in the schedulability analysis was initiated in [22]. By limiting this initial offset to a certain percentage of the supplier period, the given component can be schedulable with a lower budget. This is an assumption that we are making about the system. It is the responsibility of the system engineers to confirm that the offset that they chose actually conforms with the real system. Thus we are not computing optimal offsets making the system

schedulable, but investigating the impact of different offsets on the individual component resource requirements. As shown in Table 5.1, the smallest supplier budget can be obtained if all tasks arrive exactly synchronously with the start of the supplier period. This could be hard to achieve in practice. On the other hand, we think that it is indeed very possible to make the tasks synchronized with the supplier period such that all tasks arrive within either the first 20% or 50% of the supplier period. For these realistic values, we still obtain significant savings in the budget that a given component needs in order to be schedulable (See column 6 to 9 of Table 5.1). The percentage value is a parameter that can be easily changed in our setting when checking the schedulability. Similarly to Table 2, all statistical results in Table 5.1 are found using a confidence level of 0.95.

Another observation that we have made is that, the length of the period of the supplier can have a great impact on the budget that a component needs in order to be schedulable. This can be seen in Table 5.1 for component S_5. We have analyzed the same component with two different supplier periods. The first period is not a common divisor of the task periods (50000), while the second supplier period (10000) is a common divisor of the task periods. For the first experiment, the component can be schedulable with 30% of the complete system resources, while in the second case it can be schedulable using only 18.8% of the system resources (see column 4). In fact, this observation is an experimental result that can be found using both our approach and the CARTS tool (see Table 2 for component S_4).

4.3 Confirming Uppaal SMC results with model checking

In order to give absolute responses about the schedulability analysis performed using Uppaal SMC, we have verified some of the Uppaal SMC results by means of symbolic model checking. These are marked in Table 5.1 by a gray background color in the cells. The reason for only verifying some of our results, but not all, is that for some of the models the verification time is as much as a couple of days.

According to our experience, statistical model checking is a good way to deal with the undecidability challenge of symbolic model checking in schedulability analysis, but does not represent an alternative.

5 Evaluation and Comparison

In order to evaluate the correctness of our model-based approach, we compare the component budgets from our estimation to the budgets obtained by the CARTS tool [18] for the same hierarchical system configurations. All the results presented in Table 2 are obtained with a confidence 0.95. When Uppaal SMC returns a result where the estimated probability of missing a deadline is an interval from zero to some low value ϵ (e.g. [0,0.0973938]), this means that Uppaal SMC did not find any trace in which a deadline was missed, i.e. with 95% confidence a

Table 2. Comparison of the estimated budgets of CARTS and Uppaal SMC

Comp	Tasks	P, WCET	CARTS		SMC	
			EDF	RM	EDF	RM
S_1	T_1 T_2	500, 30 500, 100	100, 32.5	100, 32.5	100, 33	100, 33
S_2	T_1 T_2	170, 30 500, 100	100, 46.67	100, 47.5	100, 47	100, 48
S_3	T_1 T_2	250, 40 750, 50	**150, 42.5**	**150, 42.5**	**150, 45**	**150, 45**
S_4	T_1 T_2 T_3 T_4	80000, 6890 100000, 8192 200000, 2644 1000000, 5874	50000, 15082 10000, 1880	50000, 15082 10000, 2155.6	50000, 15082 10000, 1875	50000, 15082 10000, 2155

deadline will not be missed with the given budget and probability distribution. If a higher confidence is needed, the confidence value can be increased and the query can be rerun.

Table 2 shows the comparison we have done with the CARTS tool. Column 1 (Comp) contains 4 different components on which we have performed the experiment. The workload of each component is stated on the second column (Tasks). In fact, each of component S_1, S_2 and S_3 is a parallel composition of 2 tasks (T_1, T_2), while S_4 contains 4 tasks (T_1,..., T_4). The third column specifies the period and the worst case execution time for each task. In order to perform a more thorough comparison, we have considered two different scheduling policies; EDF and RM. According to the CARTS tool, the minimum budget that the resource supplier should provide each 100 time units, for which the component S_1 is schedulable under EDF and RM, is **32.5**. For the same parameters, the minimum budget we have computed in our framework using Uppaal SMC is **33**, which is very close to that obtained by CARTS. The two tools produce almost identical results. CARTS has the advantage of being an extremely fast method, while our approach is extremely flexible and configurable.

5.1 Uppaal SMC counterexample for one CARTS result

During the schedulability analysis of a specific component configuration, we obtained a result from CARTS that was in conflict with our own results.

This was for the specific case (bold gray numbers) of component S_3 in Table 2. According to CARTS's computations, the minimal necessary budget for S_3 to be schedulable under EDF and RM is **42.5**. With the use of Uppaal SMC, we first estimated the minimal budget to be 45, which has a considerable difference with the results from CARTS.

Our estimation using Uppaal SMC immediately produced a counterexample trace which shows that Task1 (T_1) can miss its deadline with

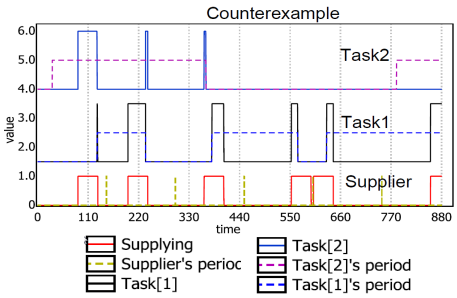


Fig. 14. Counterexample for the deadline missing of T_1 in S_3 with the budget 43 under RM in Table 2

a supplier budget of 43. The counterexample is depicted in Fig. 14 in terms of a plot that was also produced by UPPAAL SMC. The bottom of the plot shows the supplier; the dashed spikes represent the length of the supplier period and the solid line illustrates when the supplier is supplying. Each of the other two groups illustrates the behavior of a task. The solid line shows when the task is executing, and the dashed line goes up when the task is released and down when the task has finished its computation. Approximately at time 880, Task1 is executing on its third period but fails to complete before its deadline. In order to confirm our findings, we also calculated the minimum supplier budget according to the theory underlying the CARTS tool. We calculated this both using the equations from [21] and equations from [20]. The results of such calculations confirmed our findings in that we calculated the minimal budget to be 45. This leads us to conclude that there must be an error in the implementation of CARTS while the underlying theory is correct. We reported this anomaly and it has been confirmed by the developers of the CARTS tool that CARTS has an implementation error.

			Δ_{init} of supplier's period							
			Δ_{init} = 100%		Δ_{init} = 50%		Δ_{init} = 20%		Δ_{init} = 0%	
S_i	T_i	P, WCET	EDF	RM	EDF	RM	EDF	RM	EDF	RM
1	T_1	500, 30	100, 33	100, 33	100, 33	100, 33	**100, 29**	**100, 29**	**100, 26**	**100, 26**
	T_2	500, 100	(33%)	(33%)	(33%)	(33%)	**(29%)**	**(29%)**	**(26%)**	**(26%)**
2	T_1	170, 30	100, 47	100, 48	100, 47	100, 48	**100, 42**	**100, 44**	**100, 38**	**100, 44**
	T_2	500, 100	(46 %)	(48 %)	(46 %)	(47 %)	**(42 %)**	**(44 %)**	**(38 %)**	**(44 %)**
3	T_1	250, 40	150, 45	150, 45	150, 45	150, 45	150, 44	150, 45	**150, 40**	**150, 40**
	T_2	750, 50	(30 %)	(30 %)	(30 %)	(30 %)	(30 %)	(30 %)	**(27 %)**	**(27 %)**
4	T_1	10000, 1406	5000, 1406	5000, 1406	5000, 1406	5000, 1406	**5000, 1325**	5000, 1406	**5000, 1057**	**5000, 1057**
	T_2	40000, 2826	(28 %)	(28 %)	(28 %)	(28 %)	**(26 %)**	(28 %)	**(21 %)**	**(21 %)**
5	T_1	80000, 6890	50000, 15082	50000, 15082	50000, 15082	50000, 15082	**50000, 12531**	**50000, 14948**	**50000, 12400**	**50000, 13261**
	T_2	100000, 8192	(30 %)	(30 %)	(30 %)	(30 %)	**(25 %)**	(30 %)	**(29 %)**	**(27 %)**
	T_3	200000, 2644	10000, 1875	10000, 2155	10000, 1875	10000, 2155	10000, 1875	**10000,2060**	**10000,1872**	10000,1985
	T_4	1000000, 5874	(18.8 %)	(21.6 %)	(18.8 %)	(21.6 %)	(18.8 %)	(20.6 %)	**(18.7 %)**	**(19.9 %)**

Fig. 15. Enhancement of resource usability using the maximal offset of task's initial period. The cases marked with a grey background are verified using symbolic model checking

6 Related Work

In an engineering setting, providing effective parameters that make a system realizable is very practical in terms of time and cost. In this paper, while we explore the schedulability analysis of hierarchical scheduling systems by profiting from the technological advances made in the area of model checking, we propose a compositional analysis approach to determine and increase the potential configurations making much more hierarchical scheduling systems schedulable.

The concept of hierarchical scheduling systems was first introduced as 2 levels systems in [10], and then generalized as a real-time multi-level system by [15]. An

example of the increasing use of hierarchical scheduling systems is the standard ARINC 653 [2] for avionics real-time operating systems.

Several compositional analysis techniques [20,12,11,21,1,7] have been proposed. An analytical compositional framework was presented in [21] as a basis for the schedulability analysis of hierarchical scheduling systems. Such a framework relies on the abstraction and composition of system components, which are given by periodic interfaces. The interfaces state the components timing requirement without any specification of the tasks concrete behavior. In [19], the authors extend their previous work [21] to a hierarchical scheduling framework for multiprocessors based on cluster-based scheduling. They used analytical methods to perform the analysis. However, in both [21] and [19], the proposed framework is limited to a set of formulas describing an abstraction of the system entities, given in terms of periodic interfaces, without any specification of the tasks behavior and interaction. CARTS (Compositional Analysis of Real-Time Systems) [18] is tool that implements the theory given in [21,19]. Compared to our approach CARTS is a mature tool that is easy to use. On the other hand, we provide a more detailed modeling and analysis.

As common traits, analytical approaches assume computations with deterministic Execution Time usually coincident with the Worst Case Execution Time (WCET), and they provide pessimistic results [7]. Recent research within schedulability analysis gives tremendous attention to model-based approaches, because of their expressiveness that allows for modeling more complicated behavior of systems, and also due to the technological advances made in the area of model-based simulation and analysis tools. In [4], the authors analyzed the schedulability of hierarchical scheduling systems using the TIMES tool [1,3], and implemented their model-based framework in VxWorks [4]. They constructed an abstract task model as well as scheduling algorithms focusing on the component under analysis. However, the authors not only consider the timing attributes of the component under analysis but also the timing attributes of the other components that can preempt the execution of the current component. Thus, the proposed approach is not fully compositional. The authors of [7] provided a compositional framework modeled as preemptive Time Petri Nets for the verification of hierarchical scheduling systems using the ORIS tool [16]. They only analyze systems using two specific scheduling algorithms severely restricting the class of systems they can handle. In [9], the authors introduced a model-based framework using Uppaal for the schedulability analysis of single layered scheduling systems, modeling the concrete task behavior as a sequence of timed actions.

We have been inspired by the work in [9] but generalizing and lifting it to a compositional approach for hierarchical scheduling systems. Resource efficiency constitutes one of the most important factors in the performance evaluation of hierarchical scheduling systems. Such resources are often represented by either periodic [20] or explicit deadline periodic [11] resource models. The resource models represent an interface between a component and the rest of the system. In [13], the authors introduced the Dual Periodic Resource Model (DPRM) and presented an algorithm for computing the optimal resource interface, re-

ducing the overhead suffered by the classical periodic resource models. In [17], the authors introduced a technique for improving the schedulability of real-time scheduling systems by reducing the resource interferences between tasks.

In contrast, we propose a model-based framework for the modeling of hierarchical scheduling systems with a generic resource model, while we use Uppaal and Uppaal SMC to analyze the schedulability of components in a compositional manner. We also introduce two novel techniques for improving resource efficiency, and computing the minimum resource supply of system components.

7 Conclusion

In this paper we have presented a compositional methodology for schedulability analysis using a combination of statistical and symbolic model checking. The methodology could be instantiated with any modeling formalism supporting both a lightweight statistical analysis and a more costly formal verification.

The methodology we propose is instantiated in a concrete approach using Parameterized Stopwatch Automata (PSA), Uppaal SMC and Uppaal. Our approach is model based, compositional and highly configurable. We have compared the results we obtained on different system configurations with results obtained from the CARTS tool. The results from the two tools are almost identical. We discovered one case with a large difference, which has been confirmed as an implementation error by the developers of CARTS. Our configurable approach can be instantiated and updated for many different applications and system configurations including scheduling policies. We have investigated two specific techniques for enhancing the resource utilization: a new resource model and offset manipulation. Both techniques are investigated using statistical model checking. We also provided a faster method for estimating the minimal budget of a supplier, instead of performing a binary search of potential budgets. The main contribution of the paper is that systems, which cannot be proven schedulable using classical analytic approaches, can potentially be proven schedulable using our approach.

********** A perspective of this work could be a study of the impact of the two techniques, we proposed for the enhancement of resource utilization, on the systems energy efficiency ***add ref to ERTS 2014****

References

1. T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Times: A tool for schedulability analysis and code generation of real-time systems. In K. G. Larsen and P. Niebert, editors, *FORMATS*, volume 2791 of *LNCS*, pages 60–72. Springer, 2003.
2. ARINC 653. Website. https://www.arinc.com/cf/store/documentlist.cfm.
3. 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.

4. M. Behnam, T. Nolte, I. Shin, M. Åsberg, and R. Bril. Towards hierarchical scheduling in VxWorks. In *OSPERT 2008*, pages 63–72.
5. 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. LNCS Volume 8348.
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. A. David, K. G. Larsen, A. Legay, and M. Mikucionis. Schedulability of herschel-planck revisited using statistical model checking. In *ISoLA (2)*, volume 7610 of *LNCS*, pages 293–307. Springer, 2012.
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. A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using edp resource models. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, pages 129–138, 2007.
12. A. Easwaran, M. Anand, I. Lee, L. T. X. Phan, and O. Sokolsky. Simulation relations, interface complexity, and resource optimality for real-time hierachical systems, 2009.
13. 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.
14. R. J. B. Mike Holenderski and J. J. Lukkien. An efficient hierarchical scheduling framework for the automotive domain. In S. M. Babamir, editor, *Real-Time Systems, Architecture, Scheduling, and Application*, pages 67–94. InTech, 2012.
15. A. K. Mok, X. A. Feng, and D. Chen. Resource partition for real-time systems. In *Proceedings of RTAS '01*, pages 75–84. IEEE Computer Society, 2001.
16. ORIS. Oris tool website. http://www.oris-tool.org/.
17. L. T. X. Phan and I. Lee. Improving schedulability of fixed-priority real-time systems using shapers. In *Proceedings of RTAS '13*, pages 217–226, Washington, DC, USA, 2013. IEEE Computer Society.
18. L. T. X. Phan, J. Lee, A. Easwaran, V. Ramaswamy, S. Chen, I. Lee, and O. Sokolsky. CARTS: a tool for compositional analysis of real-time systems. *SIGBED Rev.*, 8(1):62–63, Mar. 2011.
19. I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. In *ECRTS*, pages 181–190. IEEE Computer Society, 2008.
20. I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS*, pages 2–13. IEEE Computer Society, 2003.
21. I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.
22. K. Tindell. *Adding time-offsets to schedulability analysis.* University of York, Department of Computer Science, 1994.