Exploiting Execution Dynamics in Timing Analysis Using Job Sequences

Leonie Ahrendts and Rolf Ernst
TU Braunschweig

Sophie Quinton
Inria Grenoble Rhône-Alpes

Editor’s note: Performance verification for real-time systems is based on system models for which it is possible to formally derive safe timing parameters such as worst-case response times. In order to make such a verification manageable, the models are often oversimplified compared to the real loads and, thus, the results, even if safe, are overly pessimistic. This article presents modeling and analysis techniques based on richer models that allow for a more accurate representation of real workloads and produce less pessimistic results. In particular, the authors argue for sequence-based approaches in both modeling the workload and specifying the imposed constraints. They also show how such approaches can be applied in industrial practice.

—Petru Eles, Linköping University

**DEriving** tight bounds on the timing behavior of a real-time computing system is known to be a challenging verification problem. Verification itself is difficult, but another major problem is that of identifying a precise yet analyzable system model for which safe (i.e., possibly approximate but always correct) parameter values can be obtained in practice through measurements or formal approaches.

A real-time computing system consists of a set of software tasks that compete for processing and communication resources and that are served according to a scheduling algorithm. A task is executed repeatedly and each of its instances is called a job. A task can, therefore, be seen as an infinite sequence of jobs over time. The creation of a job is triggered by an *activation event*, and the amount of service requested by a job is called its *workload*. Jobs may access shared resources such as memory during execution. For performance verification, a task is modeled using bounds on its timing parameters. To derive such bounds, it is common to

- characterize the best-case/worst-case parameters that can be observed for a single job of this task and
- then attribute these extreme parameters to every job of the considered task. Characteristic parameters of a job include its execution time, access times to shared resources, communication delays, as well as the temporal distance to the activation event of the subsequent job (simply called job distance in the sequel). This procedure leads to a safe but pessimistic timing model of a task.

A similar approach is commonly chosen for specifying the constraints imposed on a real-time computing system: The hardest timing constraint that applies to one job of a given task is adopted for all jobs of that task. One such example is a task deadline that must be met for every job. In this paper, we advocate the use of *job sequences* to describe the best-case/worst-case timing parameters and constraints of a task. These should be formulated for sequences of $n$ consecutive jobs (also called $n$-sequences in the following). By this means, execution dynamics and, therefore, variability...
in task behavior can be taken into account. For instance, in an $n$-sequence of jobs of the same task, the temporal distance between the first and last activation events is guaranteed to be larger than $n$ times the minimum job distance. Similarly, $n$ consecutive jobs of a task have a maximum cumulative workload that is smaller than $n$ times the worst-case execution time. On the constraint side, it may be tolerable for some jobs in a given $n$-sequence to miss their deadline.

In various works on real-time computing systems, specific problems have been successfully solved by considering sequences of jobs for modeling and/or constraint specification. We believe that a rigorous and consistent use of job sequences for task modeling and constraint specification could represent an important step toward tighter bounds on system timing behavior. In addition, the effort required to model timing parameters and to derive constraints for potentially any $n \in \mathbb{N}$ can be reduced with appropriate mathematical methods.

In the rest of this paper, we first survey and discuss existing work based on job sequences for either modeling or constraint specification. We then show as an example how typical worst-case analysis (TWCA) achieves substantial improvements in accuracy by systematically using job sequences for both modeling and constraint specification. We illustrate its practical significance by industrial case studies.

Using job sequences for task modeling and constraint specification

In this section, we discuss seminal research contributions that exploit the properties of job sequences for modeling or constraint specification of real-time computing systems. Note that, although these powerful abstractions for the description of job sequences exist, they are often not used to their full potential in practice.

Task modeling using timing parameters based on job sequences

An activation event may be caused by a periodic timer interrupt, or by a measured variable falling below or exceeding a threshold value, an alarm indicating a specific incident like a timer overflow or a fault. Many activation events thus have an aperiodic nature, and their timing depends on the dynamics of the system environment. The execution time of a task, on the other hand, may vary due to several reasons: data-dependent control flow, variable resource usage, or access times as in, e.g., memory accesses.

Using job sequences for modeling the arrival of activation events and workload can greatly improve the accuracy of the model when the timing of these parameters is subject to high variability. This important observation is at the core of Network Calculus [1], and has been exploited by a host of work in communication theory. Network Calculus was later adapted and proposed as a method for real-time system design under the name of Real-time Calculus [2], [3]. Both Network Calculus and Real-time Calculus use as fundamental modeling concepts event arrival curves and workload curves, which describe the best-case and worst-case task parameters for job sequences. For instance, the upper event arrival curve $\alpha_i'(\Delta t)$ of a task $\tau_i$ bounds from above the number of activation events that may occur in any time interval $\Delta t$. The relation to job sequences is even more obvious if one considers the pseudo inverse $\delta_i'(n)$ of $\alpha_i'(\Delta t)$, which we call the distance function: $\delta_i'(n)$ returns the minimum temporal distance between the first and last activation events in any sequence of $n$ consecutive jobs of task $\tau_i$. Similarly, the upper workload curve $\gamma_i'(n)$ of a task $\tau_i$ bounds from above the workload requested by any $n$ consecutive jobs.

Event arrival and workload curves provide an expressive task modeling approach that can yield more accurate analysis results. Figure 1 illustrates the striking difference between event arrival and workload curves obtained based on

- worst-case parameter values for a sequence of jobs and
- linear extrapolations of worst-case parameter values of a single job. We mean by linear extrapolation with respect to event arrival that the minimum interarrival time of any two jobs is used as period. Linear extrapolation with respect to requested workload is the weighting of the worst-case execution time with the number of activation events. In contrast, the nonlinear, job sequence-based worst-case models represent tighter upper bounds $\alpha_i'(\Delta t)$ and $\gamma_i'(n)$ since they are based on the observation of more than
Constraint specification based on job sequences

The classical timing constraint for real-time systems is the deadline of a task, specifying the maximum allowed response time of any job of this task. Satisfying this constraint guarantees a maximum reaction time that fits the time constants of the system and its environment.

For systems with control or imaging applications, it has been demonstrated that deadline misses can actually be tolerated without any impact on their functional correctness (see [6]–[8]) as long as the pattern of deadline misses is precisely known. Such robust systems are called weakly-hard real-time systems. A tolerable pattern of deadline misses is usually defined as a $(m, k)$ constraint, where at most $m$ deadline misses in $k$ consecutive task executions are allowed. This implies that for weakly-hard systems a response time constraint is a function of the past system behavior. $(m, k)$ constraints thus capture variability in timing constraints over a sequence of $k$ jobs. Weakly-hard systems are usually verified as if they had hard real-time constraints. Specifying $(m, k)$ constraints for them rather than a single deadline clearly increases their likelihood to be successfully verified. Interestingly, from the guaranteed satisfaction of a given $(m, k)$ constraint, one can infer satisfaction of constraints for other values of $k$ [9].

Discussion

We have seen so far that several standalone approaches exist that exploit execution dynamics in the timing analysis of real-time computing systems. On the one hand, the consideration of job sequences allows refined modeling of event arrival and workload. On the other hand, weakly-hard constraints improve accuracy in constraint specification by introducing requirements over a sequence of jobs, taking into account the inherent robustness of systems toward occasional deadline misses.

Furthermore, the presented concepts—task modeling based on job sequences and weakly-hard constraints—share the mathematical property that they describe or constrain job sequences in a cumulative manner. Cumulative functions do not preserve knowledge about the individual timing behavior of one job. The shaded area between the linear and nonlinear sequence models in Figure 1 illustrates the gain in accuracy.

Let us now shortly discuss the options for deriving in practice such tight and expressive event arrival and workload curves over job sequences. Event arrival curves that are formally derived are tight if the behavior of the event source is either analytically known or enforced. Periodic event arrivals with jitter fall, for instance, in the first category. Shaped event streams fall in the second category. The workload of a job sequence can be formally bounded using, e.g., a multiframe task model [3], [4] whenever knowledge about the task structure and functional behavior is available. Besides, it is not a problem if upper event arrival curves or upper workload curves are only known up to a specific $n$ value, as the concept of subadditive extension can be applied [5].

The tightness of formal bounds relies on the detailed information on the software and hardware platforms in use. Such information is not always available and then formally derived bounds are pessimistic. When platforms with complex performance-enhancing and power-saving features are used, this pessimism is so large that the practical usability of these formal bounds is disputable. A complementary approach is, thus, to derive bounds on event arrival and workload by measurements over execution traces. An execution trace of a task is a list of observed activation instants and execution times of an actual job sequence. Although it has the character of experiments and an uncertainty thus formally remains, trace recording is a widely used and accepted technique in industrial practice.

**Figure 1. Bounding event arrival and workload.**
each job in the considered sequence, but summarize the timing characteristics of the sequence. This approach is mathematically elegant, because it contains all required information for performance verification but condenses at the same time several equivalent worst cases in one description. The introduced event arrival curves and workload curves are cumulative since they describe the worst-case aspects of task behavior with regard to a time interval $\Delta t$ (event arrival curve) or a sequence of $n$ jobs (workload curve). Likewise weakly-hard constraints define a budget of deadline misses for a sequence of $n$ jobs, which generally includes several allowed patterns of jobs with missed deadlines.

In this paper, we argue for systematically applying sequence-based approaches in both modeling and constraining. As will be demonstrated in the following, this is an important step to significantly reduce pessimism of formal timing analysis results and make the verification of highly loaded, industrial real-time computing systems possible.

**Verifying highly loaded systems**

Highly loaded real-time computing systems, which actually work in industrial practice, are often rejected by formal timing analysis. The discrepancy between measurements and formal analysis can be considerably reduced, if tighter upper bounds on event arrival and workload are applied as described in the previous section on “Task modeling using timing parameters based on job sequences.” This standalone approach is, however, often not sufficient.

System feasibility observed in practice suggests that the event arrival and workload demand of tasks must be most of the time below the obtained upper bounds. In the transient overload situations, which may happen, there is experimental evidence that many systems tolerate a limited number of deadline misses. The functional robustness toward $m$ deadlines misses in a sequence of $k$ consecutive jobs can even be proven [7], [8]. It seems, therefore, reasonable to combine sequence-based modeling with weakly-hard constraints introduced in the constraint specification based on job sequences section.

One key issue is how to formally provide $(m, k)$ guarantees, considering the schedulable and unschedulable phases of system behavior. The verification method TWCA [10] proposes a possible solution. First, event arrival curves and workload curves for each task are derived, which are true upper bounds for most of the run time. Such event arrival curves and workload curves are called typical, because they capture the predominant timing behavior of tasks (e.g., the periodic workload but not the rare sporadic workload). Those typical curves describe a less service-demanding job behavior than the worst-case curves: Figure 2a shows a typical event arrival curve $a^{+\text{typ}}(\Delta t)$ and a worst-case event arrival curve $a^*(\Delta t)$ for a given task, where by definition we have $a^{+\text{typ}}(\Delta t) \leq a^*(\Delta t)$. In the example, the typical event pattern is periodic, while in the worst-case additional sporadic activation events occur. Figure 2b illustrates a typical workload curve $\gamma^{+\text{typ}}(n)$ and a worst-case workload curve $\gamma^*(n)$ for a given task, where again $\gamma^{+\text{typ}}(n) \leq \gamma^*(n)$. In a phase of typical system behavior, a certain maximum
typical execution time $TCET$ is never exceeded, while execution times larger than $TCET$ may occur in the worst case.

The difference between the worst-case curve and the typical curve is monotonically increasing, both for events and workload. In contrast to the approaches presented in the previous section, however, important differences are not only obtained for longer job sequences but also for a single job. On the one hand, the typical event arrival curve does not assume the minimum interarrival time even for a single job. On the other hand, the typical workload curve does not attribute the worst-case execution time $WCET$ to a single job but the maximum typical execution time $TCET$. On the basis of typical event arrival curves and workload curves, highly loaded real-time systems can be proven schedulable in phases of typical behavior. To verify the worst-case behavior, TWCA quantifies the maximum distance between the typical and worst-case curves. The additional activation events contained in the worst-case event arrival curve but not in the typical event arrival curve can be considered as cause for transient overload in the system. It is actually possible to bound the occurrence of these overload events in $\Delta t$ by the event arrival curve $\alpha^{WCET}(\Delta t)$. Similarly, jobs that exceed the typical execution time are a potential source of overload. The maximum number of jobs that exceeds the typical execution time in $\Delta t$ can be bounded by an event arrival curve $\alpha^{TCET}(\Delta t)$.

From the comparison of the typical and worst-case workload curves follows, moreover, that the amount of additional workload in a sequence of $n$ consecutive jobs cannot be larger than $\gamma^{TCET}(n)$. TWCA now derives the maximum number of missed deadlines in a job sequence of given length $k$ as a function of the number of overload events and the amount of additional workload. As we will see in the following section, highly loaded real-time systems with weakly-hard constraints have been successfully verified using the TWCA method.

Case studies

The significance of TWCA results for industrial practice has been demonstrated by several major use cases.

Automotive communication networks

In [11], the timing behavior of automotive controller area network (CAN) buses has been investigated. Automotive CAN buses have seen a massive increase in utilization in recent years due to larger message sizes and a rapidly growing number of messages. CAN messages are time-triggered and/or event-triggered. If the minimum interarrival time is used for the modeling of event arrival in a system that is dominated by event triggering, the theoretical worst-case utilization of working systems exceeds 100% and may reach 500%. Tight nonlinear event arrival curves allow for a much more accurate performance analysis. Yet such an improved worst-case response time (WCRT) analysis still discards many systems, which have proven functional in extensive simulation. The reason is that the occasional loss of messages can actually be tolerated, and the modeling of response time constraints in the form of $(m, k)$-guarantees better represents the actual system requirements.

The work presented in [11], therefore, applies sequence models and subsequently TWCA to the CAN case study. First, tight upper event arrival curves are derived based on the specified and measured timing of message dispatch. Then sporadic dispatch events are identified, which can be interpreted as overload events potentially causing deadline misses during transient workload peaks. Based on TWCA, for each message, an $(m, k)$-guarantee is obtained. In the case study, it could be shown that in at most 15% of 10,000 executions, a CAN message transmission takes longer than in the overload-free case. For many of the 212 messages, the percentage is significantly below 15%. This experimental result was the first to formally show that an increase of the classical CAN bus load is actually tolerable.

Automotive software: Engine management

Automotive software applications integrate a large number of interdependent functions. The engine management ranks among the most complex software applications and is composed of about 20 container tasks including around 1500 functions that are scheduled by an open systems and corresponding interfaces for automotive electronics-compliant operating system. These container tasks are a source of strong but well understood execution time variations. The average system utilization is usually above 90%, while the worst-case utilization in those systems exceeds easily 100%. Despite this evidence of overload, extensive simulation often suggests the functional correctness of the investigated software systems. This discrepancy can be attributed
to inaccurate utilization analysis that does not take into account the variability of execution times.

The system-level timing feasibility test proposed in [12] for an engine control application shows that with a workload curve $\gamma(t)$ that describes the execution demand of job sequences, significantly tighter WCRT can be derived. While with linear workload modeling, five out of 20 tasks are found to be infeasible in formal performance analysis, workload modeling with response to job sequences improved the accuracy of results such that only two out of 20 tasks are bound to complete after their deadline. Since the involved control applications are inherently robust toward occasional deadline misses, $(m, k)$-guarantees for the two unschedulable tasks are derived. TWCA is applied to this problem of computing the $(m, k)$-guarantees in [12], yet the overload is not caused by additional sporadic activations in this use case. In contrast, it is caused by the execution times of tasks that are occasionally longer than the TCET. In the case study, the 100- and 200-ms tasks could each tolerate three deadline misses in 20 consecutive executions, and as few as one deadline miss in 20 executions could actually be guaranteed by TWCA.

Modern real-time computing systems with performance-enhancing features have high variability in event arrival and workload. At the same time system requirements with regard to job completion are not static but often depend on system history. For instance, there may be a precisely defined budget for deadline misses of jobs.

**AN APPROACH TO DEAL** with these dynamic system characteristics is to model and constrain sequences of jobs rather than focusing on the behavior of a single job in isolation. Event arrival curves, workload curves, and weakly-hard constraints are the existing abstractions that allow making worst-case statements about the sequences of jobs. The systematic and rigorous use of this more detailed modeling and constraint formulation allows designing systems with formal worst-case guarantees where established methods for formal performance analysis are not applicable due to their pessimism. Since the approach is compatible to the existing engineering methods of measuring and trace recording, it provides an opportunity to improve design verification and optimization where current design practice has to live with unsafe simulation and prototyping.

This paper has presented as an example the TWCA method that is based on an analysis of the impact of transient overload. The gained accuracy narrows significantly the gap between the verification results of formal performance analysis and simulations that are currently used for validation in industrial practice.

**Acknowledgments**

This work has received funding from the German Research Foundation under the contract number TWCA ER168/30-1. This work has also been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

**References**


Leonie Ahrendts is a Ph.D. student with the Embedded System Design Automation Group at the Institute of Computer and Network Engineering, TU Braunschweig. Her current research concerns the analysis of real-time systems with an emphasis on weakly-hard real-time systems. Other areas of interest are fault-tolerant real-time computing systems and communication networks. She received a master’s degree in electrical engineering from TU Braunschweig in 2015.

Rolf Ernst is a Professor at the Technische Universität Braunschweig, where he chairs the Institute of Computer and Network Engineering covering embedded systems research from real-time systems to challenging automotive and aerospace applications. He received a PhD in electrical engineering and information technology from the University of Erlangen. He is an IEEE Fellow.

Sophie Quinton is a Researcher at Inria Grenoble Rhône-Alpes in France. She received a PhD degree from the University of Grenoble in 2011. Her research focus is mostly on real-time schedulability analysis and contract-based design and verification of embedded systems.

Direct questions and comments about this article to Leonie Ahrendts, Institute of Computer and Network Engineering, TU Braunschweig, 38106, Braunschweig, Germany; e-mail: ahrendts@ida.ing.tu-bs.de.