# Challenges with Probabilities in Response-Time Analysis of Real-Time Systems

Thomas Nolte*†, Meng Liu*, Björn Lisper*
*MRTC / Mälardalen University, Västerås, Sweden
†Software Systems / ABB Corporate Research, Västerås, Sweden

**Abstract**

In this paper we present and discuss some of the key open problems and challenges with using probabilities in Response-Time Analysis (RTA) of "real" real-time systems, i.e., challenges inherent in the real software to be analyzed.

## I. Introduction

The Worst-Case Response-Time (WCRT) of a task is the maximum time after activation within which the task will finish its execution. The WCRT is constructed by the Worst-Case Execution Time (WCET) of the task itself, as well as the interference of other higher priority tasks, along with potential blocking by lower priority tasks if blocking is allowed by the task model. The interference of other higher priority tasks is constructed using the higher priority tasks' WCET and period. Hence, the WCET of tasks in a real-time system is central in deriving the WCRT of a particular task. Usually the WCET is derived using static analysis of the source code in a context of a specific processor on which the task is to execute on. The context is needed to get the correct timing characteristics.

Now let's consider that the probability of the WCET to actually happen is rather low. If we further consider that the tasks are independent from each other, then the probability of the actual WCRT to happen will be derived by multiplication of the corresponding WCET probability of the task under analysis along with the respective WCET probability of all instances of interfering and blocking tasks executing during the busy period constituting the response time. Even with a fairly high probability of the WCET to happen it is easy to see that the probability of the WCRT to happen quickly becomes very small.

Probabilistic WCRT has the potential to avoid over-allocation of system resources such as processor capacity which, if applied in RTA for real systems, increases resource efficiency and decreases hardware cost. Hence, it is tempting to make use of the probability in the context of WCRT, such that overly pessimistic (sufficiently unlikely) WCRT values can be disregarded. Instead, using a threshold to derive more realistic (sufficiently accurate) analytical response time values would be more applicable.

In search of the holy grail of probabilistic real-time analysis techniques a large number of approaches for using probabilities in real-time analysis theory have been developed over the years. However, most of these approaches are developed around a set of simplifying assumptions restricting the problem and unfortunately also restricting the usefulness of the approach in the setting of a real system.

In this paper we try to go to the bottom of the challenges with using probabilities in RTA for real-time systems. We identify a set of key challenges inherent in such analysis, and we discuss the implication of these challenges in this context of related work. Finally, we point towards the future with an attempt to see how probabilities can be used in the context.

## II. Challenges

Below we outline 6 key challenges in achieving useful probabilities in RTA. Note that when doing WCET analysis for WCRT analysis, it is easier to capture the exact worst-case related to each challenge not having to keep track of the different states outlined below. However, when going for a probabilistic RTA, a more detailed probabilistic execution time is needed along with various state information in order to produce meaningful probabilistic response times. For each challenge outlined below we describe the known assumptions made to address the challenge. Note that it is possible to construct a system and software taking one or more challenge's assumptions into consideration, however it can be concluded from complying with these assumptions that they make the analysis less useful for application to real systems.

*Challenge 1 – SOFTWARE INTERNAL STATE*

The software internal state will affect the exact execution time of a particular execution instance of a task. Internal variables of the task may contain data that affect the exact software code to be executed and how, e.g., which software parts to execute and for how many iterations, which in turn will affect the exact execution time of a particular instance of task execution. Known assumptions made to solve the challenge:

A1.1 Forbid tasks to have internal states, i.e., each activation of a task is independent from other activations of the same task and therefore the task's execution time can be modeled as independent from the software internal state.

A1.2 Make sure that the internal state of a task will not influence its execution time.

*Challenge 2 – SOFTWARE EXTERNAL STATE*

Sharing of resources among the tasks will affect the exact execution time of a task instance. If critical sections are protected by locks, protocols for synchronization among real-time tasks can be used. There exist analysis for the worst case behavior of such protocols, however the exact execution time of a particular task execution instance is not known as this depends on how and when the other tasks sharing the same resource will execute. Known assumptions made to solve the challenge:

A2.1 Forbid synchronization and sharing of data among tasks, i.e., all tasks must be independent from each other.

A2.2 Make sure that the execution time of a task will not depend on effects related to shared data.

*Challenge 3 – SOFTWARE ARCHITECTURE STATE*

Due to usage of services and drivers located together with the task on a processor or core of a multi-core, e.g., a particular driver or service that is used by several tasks possibly also resident outside of the processor hosting the task under analysis, the exact execution time of a task instance will depend on the usage of these services and drivers. Note that it may be very difficult to characterize the exact behavior of service and driver usage by external clients compared to internal clients. Known assumptions made to solve the challenge:

A3.1 Avoid usage of shared services and drives in the software architecture.

A3.2 Enforce predictability of shared services and drivers, e.g., by hosting them in predictable containers such as servers.

*Challenge 4 – HARDWARE ARCHITECTURE STATE*

The hardware state will affect the execution time of the software executing on the hardware. The hardware state is affected by all software that is executing on the hardware. For example, the contents of caches will have an impact on the time that it takes to execute instructions and manipulate data. If two processor cores are sharing a cache, a task running on one core may interfere with the tasks running on the other core even though they are otherwise independent from each other. Known assumptions made to solve the challenge:

A4.1 Reset the hardware state before execution of a task instance, and to not allow for any preemption of the task before it has finished its execution.

A4.2 Disregard for the effect of hardware related interference. Depending on what hardware is used, the effect of the hardware internal state on the execution time of a task could be so small that it could be incorporated into, e.g., the execution time of the task, and therefore disregarded.

A4.3 Make the hardware to behave random, and therefore the effects of it can be modeled accurately.

*Challenge 5 – STATE OF THE ENVIRONMENT*

The state of the environment in which the task is executing may affect the exact execution time of a particular task execution instance. For example, a software tracking obstacles may have its execution time depend on the number of obstacles that it is currently tracking. Another example is the speed of a vehicle that may affect the execution time of a control software running in the vehicle, e.g., where the speed generate data to be processed by a task instance. Known assumptions made to solve the challenge:

A5.1 Disregard the environment from the analysis. In the context of a worst-case analysis this is a common solution, however if we would like to derive a correct distribution of execution- and response-times, the environment's affect on the execution time should be dealt with.

A5.2 Make sure that the implementation of the software in the task will not vary its execution time depending on the state of the environment, e.g., it is possible to implement (at least some) algorithms to have constant execution time.

## III. Discussion and Open Problems

We have contributed to the state-of-the-art in probabilistic and statistical analysis of real-time systems, e.g., [1], [2]. Most related work, including our much of our own, disregard either all or most of these challenges, i.e., they can be considered as open problems, in particular the combination of solutions to several of the challenges. Typically related work tries to address at most one of the challenges. We believe that addressing all challenges 1-5 is required for a general applicability of probabilistic real-time analysis in the context of "real" real-time systems, i.e., systems that are not subject to too many restricting assumptions. Hence, given current state-of-the-art, we have a lot of significant research challenges to explore in the upcoming years.

## References


[1] G. A. Kaczynski, L. Lo Bello, and T. Nolte, "Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems," in *12th IEEE Intl. Conference on Emerging Technologies and Factory Automation (ETFA'07)*, Sep. 2007, pp. 101–110.

[2] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of real-time embedded systems," in *33rd IEEE Real-Time Systems Symposium (RTSS12)*, December 2012, pp. 351–362. [Online]. Available: http://www.es.mdh.se/publications/2618-