RapidRT: A Tool For Statistical Response-Time Analysis of Complex Industrial Real-Time Embedded Systems

Yue Lu[1,2], Thomas Nolte[1], Liliana Cucu-Grosjean[2], and Iain Bate[3]
[1]Mälardalen Real-Time Research Centre, Västerås, Sweden
[2]INRIA Nancy-Grand Est, Nancy, France
[3]Department of Computer Science, University of York, York, United Kingdom
yue.lu@mdh.se

***Abstract*—RapidRT is a tool for statistical response time analysis of Complex Industrial Real-Time Embedded Systems (CIRTES). A key feature of this tool is that it does not require worst-case execution times of tasks to be known for the computation of a probabilistic task worst-case response time estimate. The presented tool is a step towards bridging the gap between academic research and industrial practice.**

I. Motive for the Research

Nowadays, many existing industrial embedded systems are very complex, large, flexible, and highly configurable software systems. Such systems often consist of millions of lines of code and contain hundreds of tasks, many of which are with real-time constraints and being triggered by other tasks in a complex and nested pattern. More importantly, in such systems, tasks may have *intricate dependencies* in their temporal behavior, such as 1) asynchronous message-passing and globally shared state variables, which may decide important control-flow conditions with major impact on task execution time as well as task response time and, 2) runtime changeability of priorities and periods of tasks and, 3) task offsets. All these dependencies result in very complicated system timing behavior (i.e., multi-modal task response time distributions). We refer to systems with such characteristics as *Complex Industrial Real-Time Embedded Systems* (CIRTES).

To maintain, analyze and reuse such CIRTES is very difficult and expensive, nevertheless, it offers high business value in response to great concern in industry. For CIRTES, not only the functional behavior of systems has to be assured, but also the temporal behavior, i.e., the Worst-Case Response Time (WCRT) of the adhering tasks in systems has to be known. However, when static Response-Time Analysis (RTA) methods [1] are used in timing analysis of CIRTES, the results are often overly pessimistic, and hence they are less useful to the practitioner. Additionally, the fundamental assumption in RTA is that accurate task WCET estimates are available. For CIRTES, these estimates are difficult and/or impossible to obtain. This is not only because of underlying modern processor architectures (with features such as cache, pipelines, branch-prediction, out-of-order execution), but also because of the intricate temporal and execution dependencies between tasks as we mentioned previously. Another interesting issue is protecting intellectual property, i.e., the non-accessibility issues in the source code and/or object code of CIRTES, makes it hard to perform or even prevents the state-of-the-art WCET analysis of tasks in CIRTES. Going forward, when combined with the fact that most existing CIRTES tend to be probabilistic in nature, this advocates moving toward statistical RTA. Such analysis computes a probabilistic task WCRT estimate according to a given task reliability requirement, which the system has to meet after some changes are made.

To solve the above mentioned problems, we have developed RapidRT [2], a tool for trace-based statistical RTA of CIRTES which consists of a novel sampling method and a statistical inference (which processes the achieved sample with more powerful and valid statistical techniques). Furthermore, our tool comes with a friendly GUI (conforming to Windows 7 style) and a rich set of tool features that allow users to better understand tasks timing behavior by providing some relevant statistical characteristics, as well as allow users to specify and analyze a wide variety of method parameters at ease.

II. Theoretical Foundation Underlying RapidRT

The underlying RapidRT methods mainly consist of two parts: a proposed sampling method (which achieves qualified analysis samples) and a statistical inference (which uses Extreme Value Theory (EVT) [3], other statistical methods and search algorithms to compute a probabilistic task WCRT estimate). To be specific, our sampling method is a two-step procedure, i.e., *the collection of representative task Response Time (RT) samples* and *the collection of the sample satisfying the independent and identically distributed (i.i.d.) assumption* (i.e., i.i.d. task RT samples). Furthermore, based around some statistical rationales given in [4], we also provide a means to determination of different sample sizes in algorithm. Next, we decompose the given task reliability requirement (of which the default value is 10^{-9} according to the highest development assurance level in the safety-critical system domain and used for instance by Airbus) in some probabilities used by the statistical inference in different

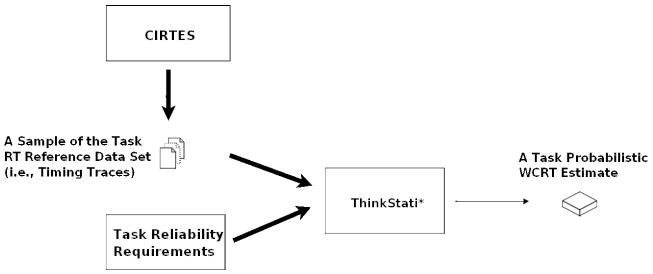

Figure 1. The work flow about RapidRT.

context. Particularly, Figure 1 illustrates the work flow about RapidRT.

Task Models: RapidRT supports a task model which contains non-blocking periodic tasks on a single processor. Typically, each task is comprised of n *jobs* (where $n \in \mathbb{N}$) and is a tuple $\tau_i(T_i, J_i, O_i, P_i, C_i, D_i)$, where T_i is task period with maximum jitter J_i, constant offset O_i, a priority P_i, and C_i is the task WCET. It is interesting to note that the exact knowledge of C_i is not required in RapidRT. Furthermore, since in CIRTES, tasks have their deadlines derived from their periods as normally defined. Therefore, in this work, we use *implicit* task deadline, i.e., the task deadline D_i is equal to the task period T_i. Besides, although the *Fixed Priority Preemptive Scheduling* is used as baseline, some tasks may change their priorities and periods at runtime, in response to system events.

III. The Design of RAPIDRT

The RapidRT implementation consists of two parts, and its first part, i.e., our sampling method, is implemented in a relatively straightforward manner, based around *Monte Carlo Simulation* (MCS). To be specific, MCS is realized by providing some generated simulator input data (conforming to a uniform distribution), of which output is a set of timing traces. The other part of RapidRT, i.e., the statistical inference, is implemented as an executable program with a friendly GUI developed using Microsoft's C# programming language. As shown in Figure 2, after loading a number of timing traces containing task response time data, RapidRT visualizes the corresponding timing behavior of the task by presenting some descriptive statistics, such as *minimum*, *maximum*, *mean*, *standard deviation*, *variance* and *probability density function*. Moreover, RapidRT provides two ways of obtaining a probabilistic task WCRT estimate, by using either some given values of parameters (as default) or user defined values. In addition, RapidRT supports multiple threads UI, e.g., the software execution process can be terminated at runtime if users press the button *Stop*. Figure 2 shows an overview of the tool GUI, which is comprised of four different working views, i.e., *SYSTEM TASK TREE* view, *DESCRIPTIVE STATISTICS OF TASK RT POPULATION* view, *PARAMETERS SETTING* view and *ANALYSIS OUTPUT* view.

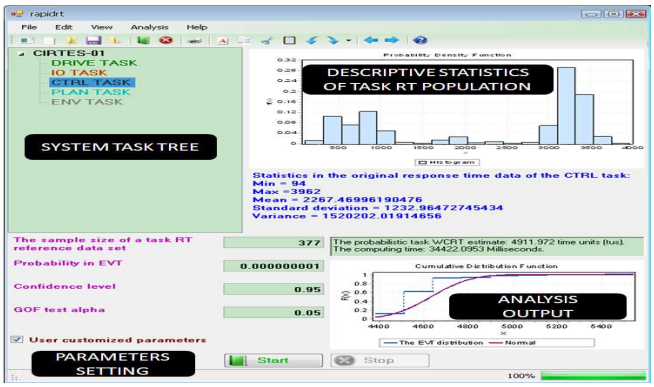

Figure 2. RapidRT Graphical User Interface (GUI).

IV. Concluding Remarks and Strategic Plan

This paper has given an overview of RapidRT, a trace-based statistical Response Time Analysis (RTA) tool for complex industrial embedded real-time systems. Particularly, we presented an overview of the underlying theoretical foundation, the tool work flow and the corresponding design. Though RapidRT is currently not an open source software for free download and use, we are interested in reaching the following milestones in our strategic plan:

- Method validation by using more case studies in industrial settings.
- Using statistical timing analysis method for multi-core or multi-processor real-time software in both RTA and WCET analysis [5].
- The development of a light weight command-line version which enables our tool to be integrated with other existing tool chains.
- Using our proposed analysis framework in some other research studies, which are not only limited to real-time operating systems and scheduling, and the possible commercialization of our developed software.

References

[1] *Handbook of Real-Time and Embedded Systems*. Chapman and Hall/CRC (July 23, 2007), 2007.

[2] Y. Lu, T. Nolte, J. Kraft, and C. Norström, "A Statistical Approach to Response-Time Analysis of Complex Embedded Real-Time Systems," in *Proc. of RTCSA' 10*, August 2010.

[3] E. Gumbel, *Statistics of Extremes*. Columbia University Press, 1958.

[4] "How to Determine a Sample Size," extension.psu.edu/evaluation/pdf/TS60.pdf.

[5] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A trace-based statistical worst-case execution time analysis of component-based real-time embedded systems," in *Proc. of ETFA' 11, Work-in-Progress session*, September 2011.