2010 13th IEEE International Conference on Computational Science and Engineering

An Approximate Timing Analysis Framework for Complex Real-Time Embedded Systems

Yue Lu, Thomas Nolte and Johan Kraft
Mälardalen Real-Time Research Centre
Mälardalen University, Västerås, Sweden
{yue.lu, thomas.nolte, johan.kraft}@mdh.se

Abstract

To maintain, analyze and reuse many of today's Complex Real-Time Embedded Systems (CRTES) is very difficult and expensive, which, nevertheless, offers high business value in response to great concern in industry. In such context, not only functional behavior but also non-functional properties of systems have to be assured, i.e., Worst-Case Response Time (WCRT) of tasks has to be known. However, due to high complexity of such systems and the nature of the problem, the exact WCRT of tasks is impossible to find in practice, but may only be bounded. In addition, the existing relatively well developed theories for modeling and analysis of real-time systems are having problems which limit their application in the context. In this paper, we address this challenge by presenting a framework for approximate timing analysis of CRTES, namely AESIR-CORES, which provides a tight interval of WCRT estimates of tasks by the usage of two novel contributions. Our evaluation using three models inspired by two fictive but representative industrial CRTES indicates that AESIR-CORES can either successfully obtain the actual WCRT values, or have the potential to bound the unknown actual WCRT values from a statistical perspective.

1 Introduction

Many industrial embedded systems are very large, flexible, and highly configurable software systems, containing many event-triggered tasks being triggered by other tasks in complex, nested patterns. Consequently, they have a very complicated runtime behavior. Such systems may consist of millions of lines of code, and contain hundreds of tasks, many with real-time constraints. Examples of such systems include the robotic control system IRC 5, developed by ABB [1], as well as several telecom systems. In such systems, many tasks 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, 2) task offsets, and 3) runtime changeability of priorities and periods of tasks. We refer to systems with such characteristics as Complex Real-Time Embedded Systems (CRTES).

To maintain, analyze and reuse CRTES is very important, difficult and expensive, which, nonetheless, offers high business value responding to great concern in industry. For instance, one specific problem in maintenance, i.e., modifying the system after delivery to correct faults, improve performance or other attributes, or to adapt the product to a changed environment, is the risk for introducing timing-related errors. In particular, for the CRTES in safety-critical applications, both functional and temporal correctness are often equally important. Thus, not only the functional behavior of systems has to be assured, but also its temporal behavior, e.g., Worst-Case Response Time (WCRT) of the adhering tasks in systems has to be known. For instance, a failing industrial robot could halt an entire production line in a factory for hours, causing a huge financial loss. Software bugs that lead to slow response time in Anti-lock Brake System (ABS) in cars will cause loss of human lives, and recall of several hundreds of thousands of vehicles. In this work, we focus on Response-Time Analysis (RTA) of CRTES in safety-critical applications.

However, due to high complexity of such CRTES, the existing relatively well-developed theories for modeling and analysis of real-time systems are having problems which limit their application in the above context. For example, timing analysis methods such as RTA [5], are often not applicable, as their assumptions of *independent tasks* in the analysis do not hold in such CRTES. The results of such analyses thereby become overly pessimistic; often too pessimistic to be useful. Moreover, methods like RTA rely on the existence of a Worst-Case Execution Time (WCET) of each task. Correspondingly, the quality of the analysis is directly correlated to the quality of WCET estimates. In or-

978-0-7695-4323-9/10 $26.00 © 2010 IEEE
DOI 10.1109/CSE.2010.21

IEEE computer society

der to perform a safe analysis covering system worst-case scenarios, static WCET analysis [35] has to be adopted in the context, which makes the assumption that tasks are isolated in the analysis. Nevertheless, such assumptions make the option to use static WCET analysis to obtain task-level WCET estimates not proper, due to the fact that task intricate temporal dependencies cannot be well handled. Furthermore, today's WCET tools cannot analyze the complex high-performance CPUs used by many industrial systems.

The state of practice in industry is that many companies developing CRTES have no means for timing analysis, and are forced to rely on testing to find timing-related problems. However, timing errors can in most cases not be detected in *unit testing* as they only occur in the integrated system, when concurrent activities are interacting or interfering, under a very specific condition. Moreover, if errors related to timing and concurrency effects are discovered in testing of the entire system, they are typically hard to reproduce. Worse yet, it is not only extremely difficult and expensive to test all scenarios in the system, but also hard to predict how a product will be used. Enabling RTA of CRTES is a problem of high industrial relevance thereof.

One solution to the problem outlined above is to use a more detailed analysis model, which ideally should be a subset of the original software program with a certain level of abstraction, e.g., WCET estimates on jobs[1] [25]. Moreover, such models describe execution control flow on code-level with respect to the significance of task scheduling, communication and allocation of logical resources. Analyzing such detailed analysis models by using model checking techniques such as UPPAAL [7, 34], TIMES [3] can result in a state-space explosion issue, which in many cases makes such the exhaustive analysis not feasible in practice. The analysis methods such as Real-Time Calculus (RTC) [31] using count-based abstraction, do not support to model the detailed states information about tasks' intricate temporal dependencies, as noted above. Another alternative approach which avoids raising too large search state-space is to use Monte Carlo simulation-based methods, which can be described as keeping the best result from a set of randomized simulations. Nonetheless, the main drawback of Monte Carlo simulation is its low state-space test coverage, which subsequently decreases the confidence in the results of finding rare worst-case scenarios.

In this paper, we are aiming for addressing the above issue by proposing an approximate timing analysis framework for CRTES, namely *AESIR-CORES* (*Advanced and Enterprising Solutions for Innovative Research on COmplex Real-time Embedded Systems*). AESIR-CORES provides an interval on WCRT estimates of tasks, which consists of lower and upper bounds. In effect, the upper bound on the WCRT of tasks is given by a statistical approach to RTA of CRTES by combining Extreme Value Theory (EVT) with other statistical methods in order to produce a probabilistic WCRT estimate. Such a probabilistic WCRT estimate may have the potential to be considered as an upper bound on WCRT estimates, especially in the case where Conventional Timing Analysis (CTA) methods cannot be applied in practice. On the other hand, the lower bound on the WCRT of tasks is obtained through the simulation optimization-based method, called *HCRR* [9], which yields substantially better results in finding system worst-case scenarios when compared to the traditional Monte Carlo simulation in the CTA methods. In addition, it is worth stressing that in this paper, the issue *model validation* is not discussed. Instead we assume that the model (extracted from the target system) is a sufficiently accurate approximation of the modeled system from the perspective of interesting timing properties, such as the execution time and response time of the adhering tasks. However, we have presented some interesting results in the context of validating temporal simulation models in timing analysis of CRTES in [24], to which the interested readers can refer for details. In addition, the evaluation is done by using three simulation models inspired by two industrial CRTES developed by ABB and Arcticus systems [4].

Contributions: The main contributions of this paper are threefold:

1. We present AESIR-CORES, i.e., an approximate timing analysis framework for CRTES by using different analysis techniques, and we show how to use AESIR-CORES to obtain a WCRT interval on estimates of tasks. To our best knowledge, no such work has been done, even no similar concept has been proposed so far.

2. We discuss the statistical conviction on using RapidRT in AESIR-CORES to obtain a tighter upper bound on the WCRT estimate of tasks, i.e., can we statistically consider the values obtained by RapidRT as good quality and representative WCRT estimates of tasks given the system model and approach that we are using?

3. We evaluate AESIR-CORES by using three models inspired by two fictive but representative industrial CRTES, which shows that 1) concerning the cases where the actual WCRT value is known, AESIR-CORES can successfully bound the actual WCRT value in a tighter interval when compared to the interval given by the CTA methods, and 2) AESIR-CORES can provide an interval which has the potential to bound the actual but unknown WCRT values from a statistical perspective, especially in the cases where the CTA methods cannot be applied.

[1] A task consists of a sequence of jobs.

Organization: Section 2 introduces our modeling framework for CRTES and research problem. Next, Section 3 provides more details about AESIR-CORES and its two underlying different analysis methods HCRR and RapidRT, followed by Section 4 which gives the testbed and implementation details. The evaluation framework using three different simulation models inspired by two industrial CRTES, and the comparison with the CTA methods are presented in Section 5. Section 6 and 7 introduce the validity and scalability of AESIR-CORES, and related work respectively, before conclusions are drawn in Section 8.

2 Timing Analysis of CRTES

This section is split into two parts: Section 2.1 introduces our modeling and analysis framework for CRTES, and Section 2.2 gives the problem definition.

2.1 Modeling of CRTES

The target CRTES are described by the modeling language in the RTSSim simulation framework [21], which is quite similar to the commercial tool VirtualTime [30] and the academic tool ARTISST [11]. RTSSim allows for simulating system models containing detailed intricate execution dependencies between tasks, such as asynchronous message-passing, globally shared state variables, and runtime changeability of priority and period of tasks. In RTSSim, the system consists of a set of tasks, sharing a single processor. RTSSim provides typical RTOS services to the simulation model, such as Fixed-Priority Preemptive Scheduling (FPPS), Inter-Process Communication (IPC) via message queues, and synchronization (semaphores). The tasks in a model are described using C functions, which are called by the RTSSim framework. The framework provides an isolated "sandbox", where time is represented in a discrete manner using an integer simulation clock, which is only advanced explicitly by the tasks in the simulation model, using a special routine, *EXECUTE*. Calls to this routine models the tasks' consumption of CPU time.

All time-related operations in RTSSim, such as timeouts and activation of time-triggered tasks, are driven by the simulation clock, which makes the simulation result independent of process scheduling and performance of the analysis PC. The response time and execution time of tasks are measured whenever the scheduler is invoked, which happens for example at IPC, task switches, *EXECUTE* statements, operations on semaphores, task activations and when tasks end. This, together with the simulation clock behavior, guarantees that the measured response time and execution time are exact.

In RTSSim, the system S contains a set of non-blocking tasks. A task τ_i may not be released for execution until a certain non-negative time (the offset O_i) has elapsed after the arrival of the activating event. Each task also has a period T_i, a maximum arrival jitter J_i, and a priority P_i. Periods and priorities can be changed at any time by any task in the application, and offset and jitter can both be larger than the period. Tasks with equal priorities are served on a First Come First Served (FCFS) basis. The framework allows for three types of selections which are directly controlled by simulator input data: 1) selection of execution times, 2) selection of task-arrival jitter, and 3) selection of task control flow, directly or indirectly based on environmental input stimulus. In addition, Monte Carlo simulation can be realized by providing randomly generated (conforming to the uniform distribution) simulator input data, and gives output in terms of a set of traces, each of which contains the measured RT and ET data of each task invocation during simulation.

2.2 Problem Formulation

In general, the actual value of WCRT of tasks in the real system is impossible to find, as its input WCET of tasks is undecidable in practice. The research in [13] recently proved that the time complexity of computing a WCRT of an independent task τ_i by using basic RTA (as shown by Equation 1) is NP-hard. Furthermore, referring to the system model with intricate task execution dependencies as the fact in CRTES, the WCET of tasks C_i is better off being represented in terms of the symbolic formula [25] in the corresponding timing analysis. Nevertheless, to use such tasks' parametric WCET representation is still an open and challenging question, which significantly increases the complexity of the problem, without a doubt. Therefore, one feasible solution is to find a tight interval consisting of lower and upper bounds on the WCRT of tasks, by using approximation techniques.

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil \times C_j \tag{1}$$

where $hp(i)$ is the set of all tasks with a priority higher than that of task τ_i.

Our research problem can be defined as follows: we are given a model M describing the target CRTES, which can be simulated on the RTSSim *simulation instance*[2] s. Let $R(s)$ denote the highest response time measured for the task under analysis in the simulation instance s. Given m simulation instances $s_1, ..., s_m$ as the samples in the entire search space S, i.e., $S \leftarrow s_1, ..., s_n$, where $n \in \mathbb{N}$. The goal of the

[2] A simulation in RTSSim is completely deterministic given a specific input, referred to as a *simulation instance*. Moreover, a simulation instance is represented as a set of sequences of integers, where each sequence is associated with either an arrival jitter of a specific task, or a specific execution time, or a specific environmental input stimulus.

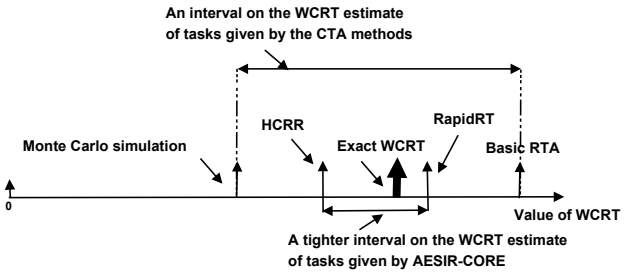

Figure 1. Illustration of applying different WCRT analysis methods in the system model of CRTES.

problem is then to find a WCRT estimate interval consisting of lower and upper bounds between which the actual WCRT s_* is. Moreover, the relationship between the results obtained by AESIR-CORES and the exact WCRT value of tasks in the system model describing CRTES is illustrated in Figure 1.

3 AESIR-CORES

In this section, we first show an overview of our research on timing analysis of CRTES, and then introduce the proposed analysis framework *AESIR-CORES*. In contrast to Conventional Timing Analysis (CTA) methods, AESIR-CORES performs approximate WCRT analysis by providing an interval on WCRT estimates of tasks. Such an interval consists of a lower bound obtained through a simulation optimization-based method (to be introduced in Section 3.2), and an upper bound derived from a statistical RTA method (to be introduce in Section 3.3).

3.1 Overview of Our Research

As shown in Figure 2, the entire process of our research starts with the model extraction work which is about to extract a set of models from a real system which depict the detailed task execution dependencies, based on program slicing [33]. Then a validation process will be performed in order to decide if the extracted models are sufficiently accurate approximation of the modeled system. If they are, then our proposed analysis framework AESIR-CORES using the simulation optimization-based method HCRR and the statistical-based RTA method RapidRT will perform RTA of tasks in the system; otherwise, the model extraction process has to be refined such that the extracted models can capture sufficiently accurate details about the target system, in the view of interesting system timing properties such as tasks' response time and execution time.

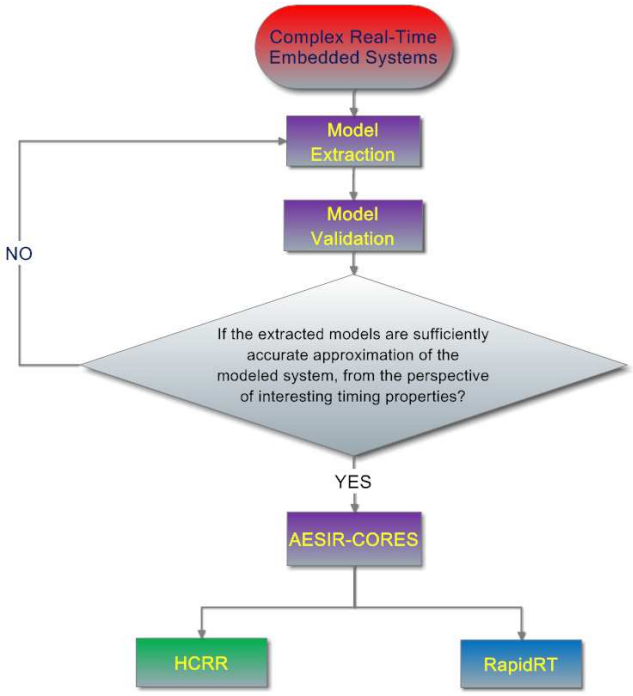

Figure 2. Overview of our research on timing analysis of Complex Real-Time Embedded Systems.

3.2 The HCRR Approach

HCRR is a meta-heuristic search algorithm based on hill climbing using random-restarts guides the traditional Monte Carlo simulation to find higher response times. Hill-climbing has the advantage of being one of the simplest meta-heuristics available, and is based on the idea of starting at a random point, and then repeatedly taking small steps pointing upward (to higher measured response time) whenever such search directions exist. If no such step exists, a local maximum has been reached.

Advantages of HCRR come from the combination of a strictly local improvement part, which quickly converges to high response times, with diversification mechanisms (jump-back to equal candidates, and full restarts) that are important to avoid local maxima. As shown in [9], HCRR yields substantially better results than Monte Carlo simulation. In details, the HCRR algorithm begins by choosing as starting point the best simulation instance from m randomly selected candidates using Monte Carlo simulation. Then, in each iteration, $k \cdot \text{len}(curr)$ random values of the current simulation instance $curr$ (which has len($curr$) input values) used before RT($curr$) are selected and modified using the neighborhood procedure Nbh. The response time for the task under analysis is measured by running an RTSSim call on a neighbor nb. Modifications suggested by Nbh that

increase response time are accepted, and changes that decrease response time are rejected. Modifications that have equal response time are rejected but saved for future reference, as described below.

A pure hill-climbing procedure is susceptible to getting stuck in local maxima, and can therefore exhibit less than satisfactory performance on many problems. In order to avoid convergence to locally maximal areas and to improve the probability of finding a true global maximum, two different diversification mechanisms were implemented. First of all, the algorithm jumps back to a previously encountered, randomly selected simulation instance with an equal response time to the current instance after non-improving simulations. This distributes focus over a number of equal instances, which can help in avoiding small local maxima. The second mechanism performs a full restart of HCRR from a random point after non-improving simulations. We call the *jump-back threshold* and the *random-restart threshold*. For details, the interested readers can refer to [9].

3.3 The RapidRT Approach

RapidRT is based on Extreme Value Thoery (EVT) [16, 17], which was first codified in 1958 and is a separate branch of statistics for dealing with the tail behavior of a distribution. EVT is used to model the risk of the extreme, rare events, without the vast amount of sample data required by a brute-force approach. Example applications of EVT include risk management, insurance, hydrology, material sciences, telecommunications.

RapidRT is a recursive procedure which, as the first two arguments, takes n reference data sets each of which contains m simulation traces containing tasks' response times. For each reference data set, the algorithm returns the WCRT estimation with a probability of being exceeded, e.g., 10^{-9}, which is the third algorithm argument. For instance, Airbus [2] uses such the value 10^{-9} which is at the highest development assurance level in the safety-critical system domain. Next, RapidRT will verify if the sampling distribution consisting of n WCRT estimates given by EVT for all n reference data sets (we refer to such the sampling distribution as the *EVT distribution* hereafter) conforms to a normal distribution or not, according to the result given by the non-parametric Kolmogorov-Smirnov test [22] (the KS test hereafter). If it is, then RapidRT will calculate the confidence interval (i.e., CI hereafter) of the EVT distribution, at the given confidence level 99.7%, and choose the upper bound of the CI as the final WCRT estimate. This invents a new hard statistic constraint, i.e., from the statistical perspective, given the modeled system, the possibility of the existence of the actual WCRT which is higher than the WCRT estimate given by RapidRT is no more than 1.5×10^{-12} (i.e., $(100\% - 99.7\%)/2 \times 10^{-9}$). Otherwise, if the EVT distribution cannot be fitted to a normal distribution, a *resampling* statistic *bootstrap* [29] will be adopted to obtain the upper bound of the CI of the EVT distribution. Furthermore, in our evaluation, the EVT distributions for all the three evaluation models conform to a normal distribution, therefore the bootstrap test will not be introduced in this paper.

RapidRT consists of the following three steps: 1) construction of the referenced data sets, 2) WCRT estimation of each referenced data set using EVT, and 3) derivation of a final WCRT estimate that is given by the algorithm. In addition, the outline of the algorithm is as follows:

1. Construct n reference data sets for the WCRT estimates by running m Monte Carlo simulations for each reference data at first, and then choosing the highest maximum value of response time of the task under analysis in each simulation. Consequently, the sampling distribution of response-time (RT) data per reference data set consists of the m highest maximum RT data of m simulations.

2. Perform the WCRT estimates on the task under analysis per each reference data set, i.e., est_i where $1 \leq i \leq n$.

   (a) Set the initial block size b to 1, for each reference data set.

   (b) If the number of blocks $k = \left\lfloor \frac{m}{b} \right\rfloor$ is less than 30, the algorithm stops as there are not enough samples to generate an estimate.

   (c) Segment m response times into blocks of size b, and for each of the $\left\lfloor \frac{m}{b} \right\rfloor$ blocks find the maximum values.

   (d) Estimate the best-fit Gumbel parameters μ and β to the block maximum values by using a proposed lower-part binary search algorithm introduced in [26].

   (e) Calculate a WCRT estimate based on the best-fit Gumbel Max parameters estimated through Step *d*), i.e., μ, β, and a target acceptance probability P_e, i.e., 10^{-9}.

3. After verifying if the EVT distribution (i.e., $EST \leftarrow est_1, ..., est_n$) can successfully be fitted to a normal distribution by using the KS test, RapidRT will return a result, i.e., $\overline{EST} + 3\sigma_{EST}$ (the sum of *mean* value and 3 *standard deviation* of EST at the confidence level 99.7%).

4 Implementation

Our testbed is running Microsoft Windows XP Professional, version 2002 with Service Pack 3. The computer is equipped with the Intel Core Duo CPU E6550 processor, 2GB RAM and a 4MB L2 Cache. The processor has 2 cores and 1 frequency level: 2.33 GHz.

As shown in Figure 3, the two main components in the AESIR-CORES toolchain are *SimOpti* and *ThinkStati*, which are prototypes of both HCRR and RapidRT as executable programs with a simple user interface developed using Microsoft's C# programming language and .NET framework 2.0. Both SimOpti and ThinkStati read the same output of the RTSSim simulator, i.e., one text file *out.txt* which contains m lines of simulation results representing the highest value of response time for a specific task observed during each simulation in m simulation runs. For RapidRT, in particular, it first generates the text file *yblock.txt* for each reference data set after segmenting the samples, then produces the WCRT estimation on tasks under analysis according to the best-fit Gumbel Max parameters (verified and returned by using our proposed *lower-part binary search* algorithm in [26] and the commercial software *EasyFit* [12]) and the acceptance probability, i.e., 10^{-9} in this work. The output of ThinkStati is a text file containing the EVT distribution, which is used by EXCEL 2007 to construct the CI at the confidence level 99.7%. Concerning the Chi-squared test required by ThinkStati, it is done by using EasyFit. Specifically, given the text file *yblock.txt* which contains a certain number of samples generated by ThinkStati, as the input, the Chi-squared test engine embedded in EasyFit will return the results in terms of rejecting or not rejecting the H_0 or the *null hypothesis*. Such the null hypothesis (H_0) is expressed as *if the maxima of blocks are conforming to the Gumbel Max distribution, and the corresponding parameters of the Gumbel Max distribution should not be rejected, and they will be used to generate a WCRT estimate of the task on focus.*

5 Empirical Results

In this section, we first introduce three evaluation models (including one validation model) inspired by two industrial CRTES, and then we compare our solution against the Conventional Timing Analysis methods (CTA) as reference: Monte Carlo simulation and the basic RTA using the *Response-Time Computation Formula* (RTCF) without blocking (as shown in Equation 1 in Section 2.2). It is worth stressing that though there are many considerable research in RTA of systems with excessive blocking, shared resources, etc, our evaluation models do not have such behaviors. We therefore use the basic RTA (out of other RTA methods) as a representative reference method in the evaluation.

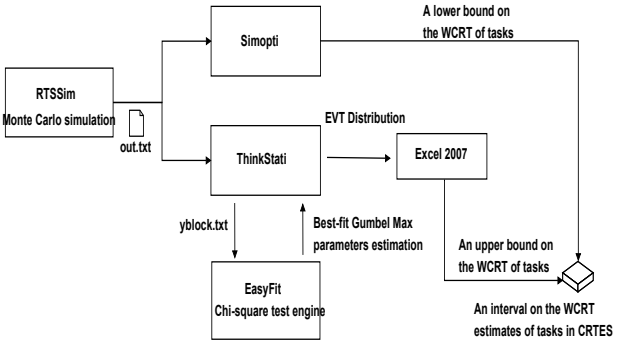

Figure 3. The toolchain in AESIR-CORES timing analysis framework.

5.1 Evaluation Models

Two of three evaluation models, i.e. Model 1 (M1) and Model for Validation (MV), have similar architecture and analysis problems as one industrial real-time application in use at ABB. M1 is representing a control system for industrial robots developed by ABB Robotics, which is not possible to analyze using methods such as RTA [5, 23]. M2 is constructed from a test application used by Arcticus Systems, which develops the Rubus RTOS used in many vehicular systems. We also use a simplified version of M1, making the basic RTA is applicable, for validation (MV). The sole purpose of MV is to investigate if the interval on the response time estimation given by AESIR-CORES can bound the known actual WCRT. The scheduling policy is FPPS for all models, apart from M1 (where FPPS is used as base but one task changes its priority during runtime). In addition, in all evaluation models, the basic time unit is one *simulation time unit* (*tu*).

5.1.1 Model 1 (M1)

This model is designed to include some behavioral mechanisms from the ABB system which RTA cannot take into account:

- tasks with intricate dependencies in temporal behavior due to IPC and Globally Shared State Variables (GSSVs);
- the use of buffered message queues for IPC, where triggering messages may be delayed;
- tasks that change scheduling priority or periods dynamically, in response to system events.

The modeled system controls a set of electric motors based on periodic sensor readings and aperiodic events. The

calculations necessary for a real control system are, however, not included in the model; the model only describes behavior with a significant impact on the temporal behavior of the system, such as resource usage (e.g., CPU time), task interactions and important state changes. The model contains four periodic tasks with the parameters shown in Table 1 (a lower valued priority is more significant).

Table 1. Task parameters for Model 1.

Task	Priority	Period	Offset	Depends on
PLAN	5	40 000	0	UI
CTRL	4 or 2	10 000 or 20 000	0	PLAN, IO, UI
IO	3	5 000	500	Sensor
DRIVE	1	2 000	1 2000	CRTL, UI

The environmental input stimulus in this problem is a sequence of integers from zero to two, denoting the number of external events that are generated by a sensor, measured in one IO task period. The IO task then sends equally many messages to the CTRL task. The CTRL task may change priority and periodicity in response to two specific events in the model. The PLAN task is responsible for planning the movement of the physical object connected to the motors. The CTRL task calculates control signals for the motors with respect to coordinates sent from the PLAN task and the IO events provided by the IO task. The DRIVE task actuates the motors based on the CTRL task output, which impacts the execution time of the CTRL task. Moreover, the DRIVE task will change the priority of the CTRL task according to different conditions.

The model also describes a user interface (UI) which generates sporadic events which impact the system behavior. There are three types of user interface events: START, STOP and GETSTATUS. The START and STOP events make the system change between two system modes, IDLE and WORKING, with different temporal behaviours. The GETSTATUS event makes the PLAN, CTRL and DRIVE tasks send a status message to the UI, which increases the execution time of those task instances. The task in focus of analysis is the CTRL task. The details of the model are described in [21].

5.1.2 Model 2 (M2)

M2 is based on a test application from Arcticus systems, developers of the Rubus RTOS [4] which is used in heavy vehicles. This model uses a pipe-and-filter architecture, where tasks trigger other tasks through trigger ports, forming transactions. M2 contains 3 periodic transactions and one interrupt-driven task, in total 11 tasks. The interrupt has a small jitter, while the other transactions are strictly periodic. The parameters of tasks and their execution times are given in Table 2.

Table 2. Task parameters for Model 2.

Task	Period	Offset	Jitter	Priority	Execution
swcIT_1	5 000	500	100	0	[100, 200]
swcIT_2	5 000	500	100	0	[100, 200]
swcA_1	5 000	0	0	1	[400, 500]
swcA_2	10 000	0	0	1	[400, 500]
swcA_3	30 000	0	0	1	[400, 500]
swcB_2	10 000	0	0	1	[400, 500]
swcB_3	30 000	0	0	1	[400, 500]
swcA_et2	10 000	0	0	2	[500, 600]
swcA_et3	30 000	0	0	2	[500, 600]
swcB_et2	10 000	0	0	2	[500, 600]
swcC_et1	30 000	0	0	2	[500, 600]

M2 is less complex than M1 in the sense that there exist no shared variables or IPC via message passing which can impact the tasks' timing and functional behavior. Instead, the tasks have large variations in execution times, which makes the state space of this model very large. For this model, the evaluation focuses on the end-to-end response time of the transaction which contains the tasks with the lowest priority.

5.1.3 Validation Model (MV)

MV is constructed based on M1, but the adhering task execution dependencies are simplified in that

- GSSVs have been removed,
- priority and period are strictly static,
- explicit loop bounds have been added manually,
- the constant offset of tasks is removed.

As a consequence, MV has considerably lower complexity, which makes both using the basic RTA by using the RTCF without blocking as shown by Equation 1 in Section 2.2 to calculate the WCRT of tasks under analysis, and achieving the exact WCRT by using both Monte Carlo simulation and HCRR [9], feasible.

5.2 Results Comparison

Before we present experimental results, it is interesting to note that the WCRT estimate of tasks given by RapidRT in AESIR-CORES is in the floating-point representation. This is due to the fact that we are using the upper bound on the CI of the EVT distribution at the confidence level 99.7%, for reference data sets. In addition, the results regarding different evaluation models are shown in Table 3. In particular, we can see:

- For MV, when the actual WCRT value of the task under analysis is known, the upper bound on the WCRT estimate given by AESIR-CORES is 13.13% (i.e., (5 982−5 196.68)/5 982×100%) less pessimistic when compared to the upper bound obtained by the basic RTA in the CTA methods. Though both of AESIR-CORES and the CTA methods find the actual WCRT value, the interval given by AESIR-CORES is 47.60% (i.e., {(5 982 − 4 332) − (5 196.68 − 4 332)} ÷ (5 982 − 4 332) × 100%) tighter than the one derived from the CTA methods.

- For M1, when the actual WCRT value of the task under analysis is not known, 1) the basic RTA in the CTA methods cannot perform analysis since all the underline assumptions cannot be made. Therefore, the upper bound on the WCRT of the task under analysis is `NA` (i.e., not applicable). However, by using AESIR-CORES, such an upper bound can be derived from RapidRT under a hard statistical constraint, i.e., 1.5×10^{-12}, which could be considered to bound the actual WCRT value (refer to Section 6.1 for the validity of statistically considering the result given by RapidRT as a sound upper bound). On the other hand, the lower bound given by AESIR-CORES, i.e., 792 tu is higher than the one given by the CTA methods, which shows that AESIR-CORES can significantly improve the lower bound of the WCRT estimate of the task under analysis in the sense of finding higher response times. Clearly, the interval on the WCRT of the task under analysis obtained by AESIR-CORES is much tighter than the one given by the CTA methods, especially in the cases where the latter cannot be applied.

- The same conclusion can be drawn for M2, except that the lower bound given by AESIR-CORES is 297 (i.e., 6299 − 6002) tu higher than the one given by the CTA methods. While the WCRT estimate of the task on focus given by RapidRT is 7 262.77 tu, which we believe is an upper bound on the WCRT of the task under analysis from the statistical point of view.

Table 3. The intervals on WCRT estimates given by AESIR-CORES and the CTA methods for three evaluation models.

	AESIR-CORES	CTA methods	Actual WCRT
MV	[4 332, 5 196.68]	[4 332, 5 982]	**4 332**
M1	[8 474, 8 698.29]	[7 682, NA]	Unknown
M2	[6 299, 7 262.77]	[6 002, NA]	Unknown

6 Validity and Scalability of AESIR-CORES

This section discusses the validity of the WCRT interval given by AESIR-CORES and the scalability of our analysis framework.

6.1 Validity of the WCRT Interval Given by AESIR-CORES

In AESIR-CORES, as we are aiming for bounding the actual WCRT value of tasks in CRTES that is either known or unknown depending on different system complexities, it is therefore important to show the validity of such an interval on the WCRT given by AESIR-CORES. This typically lies in the validity of the upper bound given by using RapidRT. Put it in the other way, we have to answer the question, i.e., *can we statistically consider the values obtained by RapidRT as good quality and representative WCRT estimates of tasks given the system model and approach that we are using?*

First, looking at Step 1 in RapidRT, the sample size for all reference data sets in the analysis is statistically sufficient in the sense of successfully covering the parameter of the underline population, i.e., the maximum value of the entire search space of RT data of the task on focus in the system model. In effect, as shown by the evaluation of the model MV, the result given by 500 000 samples (i.e., 50 reference data sets each of which contains 10 000 samples) successfully covered the actual WCRT value. Moreover, we use Simple Random Samples (SRS) [29] by running Monte Carlo simulation which eliminates the bias on the sampling in terms of giving every possible sample of a given size the same chance to be chosen. This also gives us the confidence that no matter how big the population is, the inference based on the sampling distribution collected by SRS can successfully estimate the parameters of the underline population [29]. Therefore, we believe that the step about the construction of the sampling distribution for each reference data set is sound in the sense that it has a statistically large enough sample size and accuracy of the inference regarding the actual WCRT value of tasks. It is however possible to construct cases where this method would fail to produce a safe overestimation. For instance, a task may contain an *if-statement* with a very unlikely condition, which is never true during measurements. But when it is true, the execution time of the task will be increased vastly, e.g., with a factor 1 000. Such a task is unlikely, but possible in practice. However, if using simulations as a base for this analysis, it is possible to insert detailed monitoring of the execution, without affecting the simulation results (execution and response times). This since execution time is typically consumed explicitly in simulators like RTSSim, by incrementing the simulation clock. Through extra simulation monitor-

ing it should be possible to keep track of what behaviors of the tasks that have been explored, and thereby determine if there are unexplored control branches, which possibly may result in vastly higher execution time and thereby response time. Besides, more advance measurement-based tracing mechanisms can also be used to remedy the situation.

Second, looking at Step 2 in RapidRT, for each reference data set, the usage of EVT to create a corresponding Gumbel Max distribution in the WCRT estimation allows us to statistically state that the probability of existence of an exact WCRT value that is greater than the resulting EVT estimate is no more than 10^{-9}. Such the probability is at the highest development assurance level in the safety-critical system domain, adopted by industrial companies.

Finally, looking at Step 3 in RapidRT, we use an upper bound of the CI of the EVT distribution which ensures that the possibility of having an actual WCRT value larger than the estimate given by RapidRT is less than 1.5×10^{-12} (i.e., $(100\% - 99.7\%)/2 \times 10^{-9}$). In other words, the probability of the WCRT estimate given by RapidRT to be lower than an unknown exact value of the WCRT of tasks is no more than 1.5×10^{-12}. Hence, we believe that the result given by RapidRT can be statistically considered as a good quality upper bound on the WCRT estimate of tasks, especially in the case where the basic RTA cannot be practically applied.

6.2 Scalability of AESIR-CORES

Table 4. The computation time corresponding to the number of simulations required to execute by each method in AESIR-CORES. The time unit is one second (*s*).

	MV	*M1*	*M2*
HCRR	*44.39 s*	*44.39 s*	*1 716.73 s*
RapidRT	*1 573.36 s*	*4 805.90 s*	*2 086.23 s*

In AESIR-CORES, the input to HCRR and RapidRT i.e., a number of simulation traces containing response time data of tasks in the system model, is collected by running Monte Carlo simulation, which in general scales to larger size systems [8]. Specifically, in our evaluation, the computation time cost by each method in AESIR-CORES at the testbed (introduced in Section 4) is reasonably affordable, i.e., the maximum computation time is 4 805.90 s (which is approximately equal to 1.3 hours). Though one disadvantage of the current implementation of AESIR-CORES, is that, in ThinkStati one proposed search algorithm has not yet been integrated in the toolchain. Therefore, manual effort on finding the best-fit Gumbel distribution parameters is necessary. Nevertheless, we would like to separate the scalability of the method from this issue, since the integration can be managed given a reasonable effort, with the purpose of toolchain automation.

7 Related Work

This section introduces the work that are not mentioned previously, but related. Moreover, such work are mainly falling into three categories, i.e., existing RTA techniques, simulation optimization, and statistical timing analysis. For RTA, the current work on execution precedence constraints of tasks is presented in [6]. The latest work on temporal dependencies between tasks, in terms of offset, is presented in [27]. In [32], Samii et al aim to find extreme response times for distributed systems by optimizing a set of simulation parameters for models containing temporal attributes and communication. They use a genetic algorithm [14] to explore combinations of task execution times in order to maximize end-to-end response time. However, flow of control within tasks is not considered. The analysis framework by Kim et al [20] also has a similar basis of temporal task attributes. In the view of statistical timing analysis, an interesting approach features the use of stochastic task execution times in RTA of priority-driven soft real-time systems [19] and schedulability analysis [28]. Nonetheless, this approach currently does not allow for execution dependencies between tasks in the analysis. Burns [10] presents another probabilistic framework extending RTA to incorporate a probabilistic characterization of task arrivals and execution times. However, task execution dependencies such as runtime changeability of task priorities and periods, and message-passing, are not taken into consideration. Other related work includes [18] presenting how likely a WCET estimate generated by EVT will be exceeded in the future. Here, the search algorithm concerning the best-fit Gumbel distribution parameters is done in a simple way, by only doubling the block size.

8 Conclusions and Future Work

This paper has provided a link between traditional Response-Time Analysis (RTA) engineering and the more complicated domain of Complex Real-Time Embedded Systems (CRTES). In effect, we have presented and proposed an approximate timing analysis framework for CRTES, namely AESIR-CORES, by using a collection of novel approximation techniques. Specifically, two methods are outlined for obtaining a tight interval of the Worst-Case Response Time (WCRT) of tasks in CRTES, consisting of lower and upper bounds; the methods are based on simulation optimization and statistical RTA techniques. Finally, in the evaluation of our technical contributions, we have used three simulation models describing two fictive but representative industrial applications. Moreover, we have discussed

the validity of results given by AESIR-CORES, in particular from the perspective of statistically considering the results as a tighter interval on the WCRT estimate of tasks.

Our future work will investigate ways in which to develop more advanced sampling strategy for covering unlikely conditions in the statistical RTA of CRTES, and conduct more evaluations of AESIR-CORES in industrial circles. Finally, we will consider using the proposed statistical RTA for CRTES together with trace-driven techniques in the context of analyzing real systems execution, as well as validating temporal simulation models extracted from CRTES by using statistical hypothesis testing.

References

[1] Website of ABB Group. www.abb.com.

[2] Airbus, www.airbus.com/en/, 2009.

[3] 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 *FORMATS' 03*, number 2791 in LNCS, pages 60–72. Springer-Verlag, 2003.

[4] Website of Arcticus Systems. www.arcticus-systems.se.

[5] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority pre-emptive scheduling: an historical perspective. *Real-Time Systems*, 8(2/3):129–154, 1995.

[6] I. Bate and A. Burns. An integrated approach to scheduling in safety-critical embedded control systems. *Real-Time Syst.*, 25(1):5–37, 2003.

[7] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In M. Bernardo and F. Corradini, editors, *SFM-RT' 04*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[8] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte. Best-effort simulation-based timing analysis using hill-climbing with random restarts. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-236/2009-1-SE, Mälardalen University, June 2009.

[9] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte. Simulation-based timing analysis of complex real-time systems. In *RTCSA' 09*, pages 321–328, August 2009.

[10] A. Burns, G. Bernat, and I. Broster. A probabilistic framework for schedulability analysis. In *EMSOFT' 03*, pages 1–15, 2003.

[11] D. Decotigny and I. Puaut. ARTISST: an extensible and modular simulation tool for real-time systems. In *ISORC' 02*, pages 365–372, 2002.

[12] EasyFit, www.mathwave.com/products/easyfit.html, 2010.

[13] F. Eisenbrand and T. Rothvoß. Static-priority real-time scheduling: Response time computation is np-hard. In *RTSS' 08*, pages 397–406, 2008. IEEE Computer Society.

[14] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[15] J. Goossens and C. Hernalsteen. A tool for statistical analysis of hard real-time scheduling algorithms. In *SS' 98*, page 58, 1998. IEEE Computer Society.

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

[17] J. S. J. Beirlant, Y. Goegebeur and J. Teugels. *Statistics of Extremes: Theory and Applications*. Wiley Press, 2004.

[18] S. H. J. Hansen and G. Moreno. Statistical-based wcet estimation and validation. In *WCET' 09*, pages 123–133, 2009.

[19] G. A. Kaczynski, L. L. Bello, and T. Nolte. Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems. In *ETFA' 07*, pages 101–110. IEEE Industrial Electronics Society, September 2007.

[20] K. Kim, J. L. Diaz, L. L. Bello, J. M. Lopez, C.-G. Lee, and S. L. Min. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Trans. Comput.*, 54(11):1460–1466, 2005.

[21] J. Kraft. RTSSim - A Simulation Framework for Complex Embedded Systems. Technical Report, Mälardalen University, March 2009.

[22] A. M. Law and D. M. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 1999.

[23] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[24] Y. Lu, J. Kraft, T. Nolte, and C. Norström. A statistical approach to simulation model validation in timing analysis of complex real-time embedded systems. In *WATERS' 10*, July 2010.

[25] Y. Lu, T. Nolte, I. Bate, and C. Norström. Timing analyzing for systems with task execution dependencies. In *COMPSAC' 10*. IEEE, July 2010.

[26] Y. Lu, T. Nolte, J. Kraft, and C. Norström. A statistical approach to response-time analysis of complex embedded real-time systems. In *RTCSA' 10*, August 2010.

[27] J. Mäki-Turja and M. Nolin. Efficient implementation of tight response-times for tasks with offsets. *Real-Time Systems Journal*, 40(1):77–116, October 2008.

[28] S. Manolache, P. Eles, and Z. Peng. Schedulability analysis of applications with stochastic task execution times. *ACM Trans. Embed. Comput. Syst.*, 3(4):706–735, 2004.

[29] D. S. Moore, G. P. Mccabe, and B. A. Craig. *Introduction to the practice of statistics*. W. H. Freeman and Company, New York, NY 10010, sixth edition, 2009.

[30] Rapita systems, www.rapitasystems.com, 2008.

[31] Website of Real-Time Calculus. http://www.mpa.ethz.ch/.

[32] S. Samii, S. Rafiliu, P. Eles, and Z. Peng. A simulation methodology for worst-case response time estimation of distributed real-time systems. In *DATE' 08*, volume 10-14, pages 556–561. IEEE, March 2008.

[33] F. Tip. A survey of program slicing techniques. *Journal of programming languages*, 3:121–189, 1995.

[34] Uppaal, www.uppaal.com, 2009.

[35] R. Wilhelm, J. Engblom, A. Ermedahl, . Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem—overview of methods and survey of tools. *Trans. on Embedded Computing Sys.*, 7(3):1–53, 2008.