2010 15th IEEE International Conference on Engineering of Complex Computer Systems

Statistical-based Response-Time Analysis of Systems with Execution Dependencies between Tasks

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

Abstract

This paper presents a novel statistical-based approach to Worst-Case Response-Time (WCRT) analysis of complex real-time system models. These system models have been tailored to capture intricate execution dependencies between tasks, inspired by real industrial control systems. The proposed WCRT estimation algorithm is based on Extreme Value Theory (EVT) and produces both WCRT estimates together with a probability of being exceeded. By using the tools developed, an evaluation is presented using three different simulation models, and four other methods as reference: Monte Carlo simulation, MABERA, HCRR and traditional Response-Time Analysis (basic RTA). Empirical results demonstrate that the benefit of the proposed approach, in terms of 1) reduced pessimism when compared to basic RTA and 2) validated guarantee of never being less than the actual response time values. The proposed approach also needs much fewer simulations compared to other three simulation-based methods.

1 Introduction

To date, most existing embedded real-time software systems have been developed in a traditional code-oriented manner. Many such systems are maintained over extended periods of time, sometimes spanning decades, during which the systems become larger and increasingly complex. As a result, these systems are difficult and expensive to maintain and verify. There are many industrial embedded systems consisting of millions of lines of C code, and containing 50 - 100 tasks or more, out of which many tasks have real-time constraints. One example of such systems is the robotic control systems developed by ABB [1]. Looking closer at these systems, contrary to the assumption in most real-time theory, i.e., independent tasks in the analysis model, tasks exhibit strong temporal dependencies, e.g., asynchronous message-passing, globally shared state variables (but not logical resources) and runtime changeability of periods and priorities of tasks, which vary the execution time of the tasks radically.

One desirable approach to avoid timing-related errors in such complex systems is to use schedulability analysis methods, such as Response-Time Analysis (RTA) [2, 3]. Nevertheless, RTA (and other schedulability analysis techniques), although providing the prediction about temporal behavior of execution in worst-case scenarios, rely on the existence of a fixed Worst-Case Execution-Time (WCET) of the tasks. Correspondingly, the quality of the analysis is directly correlated to the quality of the WCET estimates. Unfortunately, in the above described systems, the WCET of tasks obtained by static WCET analysis techniques may not easily be bounded. Sometimes a pessimistic WCET bound can be found, while in other cases the WCET is completely unbounded until the behavior of dependent tasks is known. Consider the following example in Figure 1, taken from an industrial robotic control system, where a task reads all messages buffered in a message queue and processes them accordingly:

```
1        msg = recvMessage(MyMessageQueue);
2              while (msg != NO_MESSAGE){
3                process_msg(msg);
4                msg = recvMessage(MyMessageQueue);
5        }
```

Figure 1. Iteration-loop wrt. message passing

By using static WCET analysis, the upper bound on the number of messages actually consumed by the task is equal to maximum queue size. Nonetheless, tasks with a higher significant priority may preempt the execution of the loop and refill the queue at runtime. A pessimistic assumption concerning the worst-case scenario, would be, for each preemption, that the maximum queue length is refilled. However, our evaluation work presented in [4] and [5] showed different results. In [4], the WCRT of the task under analysis is found when a loop has 42 iterations given a queue

978-0-7695-4015-3/10 $26.00 © 2010 IEEE
DOI 10.1109/ICECCS.2010.26

of length of 40. Whilst in [5], this number is bounded and typically smaller than the maximum queue size in the worst-case scenario. Consequently, the WCRT obtained by using basic RTA [6][1] with standard WCET estimation is 29.2% more pessimistic when compared to the known WCRT.

The other approach, which avoids the state-space explosion issue raised by model checkers such as UPPAAL [7] and TIMES [8], for instance, is to use simulation-based methods that sample the state space. The first type of simulation technology to use is Monte Carlo simulation, which can be described as keeping the highest result from a set of randomized simulations. Several frameworks already exist in this realm, such as the commercial tool *VirtualTime* [9] and the academic tool *ARTISST* [10]. However, the main drawback of using Monte Carlo simulation is the low state-space test coverage, which subsequently decreases the confidence in the results of finding rare worst-case scenarios. The other category is to apply an optimization algorithm (e.g., (meta)heuristic search algorithm), on top of Monte Carlo simulation, as in [4] and [11], which yield substantially better results, i.e., tighter lower bounds of the WCRT estimation.

Another approach is to use stochastic analysis of hybrid task sets in priority-driven soft real-time systems, as in [12]. Nevertheless, this approach does not allow for dependencies between tasks in the analysis, and the priority of jobs (a task is comprised by a sequence of jobs) and task periods are fixed.

In this paper, we present a novel statistical-based approach to response time analysis of systems with intricate execution dependencies between tasks. The proposed method uses samples collected by running Monte Carlo simulation as the input, and produces WCRT estimates on tasks along with a predictable probability of being exceeded, i.e., 10^{-9}. Specially, the contributions of this paper are four-fold:

- We propose a system model which captures the execution dependencies between tasks as mentioned previously, by using parametric WCET symbolic formula.
- We introduce the complexity of the problem about response time analysis and give the problem definition.
- We design an algorithm based on Extreme Value Theory (EVT) [13] and two adhering search algorithms used in the WCRT estimation, in order to get the best-fit estimated parameters of the distribution.
- We present empirical results to demonstrate that our solution can effectively reduce the pessimism when compared to basic RTA, while covering the best results obtained by using the simulation optimization-based methods as presented in [11] with much fewer simulations required to be run, 6% at most.

[1] Basic RTA in [6] is also called classical RTA or traditional RTA.

The remaining part of the paper is organized as follows: in Section 2, we describe a system model, where each task's WCET is represented as a symbolic formula. Section 3 introduces the complexity of RTA and gives the problem definition regarding the work. Section 4 presents the proposed method, i.e., *WCRTEVT*, and Section 5 describes the implementation of our testbed and a toolchain developed. The evaluation by using a set of case-study models, and scalability of the method are presented and discussed in Section 6 and Section 7 respectively. Section 8 introduces the related work, and finally, Section 9 concludes the paper and discusses future work.

2 System Model

In this paper we consider a system model inspired by industrial control systems containing a number of tasks communicating via asynchronous message-passing, sharing Globally Shared State Variables (GSSVs), being executed under Fixed-Priority Preemptive Scheduling (FPPS) on a single processor. More important, task periods and priorities can be changed by other tasks at runtime. Furthermore, each task is associated with b input buffers and g GSSVs. The novelty of the system model proposed in this work is to represent the WCET of tasks as a symbolic formula centering around the number of messages in the buffers and the value of the GSSVs. The reason for why parametric WCET representation is used in this context is: it captures the intricate task execution dependencies as described previously, by preserving them in terms of the parameters in the formula, which could for instance be used in basic RTA with the purpose of reducing the pessimism brought in by using a traditional, single numeric WCET representation.

In details, the system model S contains a set of non-blocking tasks, of which each task consists n *jobs*, where $n \in \mathbb{N}$. Each deadline-constrained task τ_i is a tuple $\tau_i(T_i, C_i^p, D_i, O_i, J_i, P_i)$, where T_i is the task period with maximum jitter J_i, constant offset O_i and a priority P_i, C_i^p is the WCET expressed as a function of b buffers (i.e., $U_{i,1}, ..., U_{i,b}$) and g GSSVs (i.e., $V_{i,1}, ..., V_{i,g}$) associated with task τ_i and execution time on jobs, D_i is the relative deadline ($max(C_i^p) \leq D_i \leq T_i$). The execution of task τ_i is divided into two types of sections. Firstly, the *non-volatile* (NV) section in which there is no input buffer and GSSV, and secondly, the *volatile* (V) section containing either one buffer, or a GSSV. In both sections, preemption caused by higher priority task is allowed.

2.1 Execution-Time Modeling

Each NV section in a task τ_i consists of h jobs $j_{i,x}$, where x is in the range of $[1, h]$. The WCET of the job $j_{i,x}$ is represented as $C(j_{i,x})$, and practically, such a value can be ob-

tained by using either static WCET analysis (which is safe but more pessimistic compared to the exact WCET) that are used in applications with hard real-time constraints, or through dynamic WCET estimates based on measurements with probability distribution when dealing with applications with soft real-time constraints. The corresponding WCET of the NV section containing h jobs is expressed as follows:

$$C_{i,nv} = \sum_{x=1}^{h} C(j_{i,x}) \tag{1}$$

The execution on a Volatile (V) section of a task τ_i is heavily dependent on the number of processed messages in the input buffer and data stored in the associated GSSV. Accordingly, the execution on a Volatile (V) section consists of two parts: *execution on GSSV* and *execution on message passing*.

The value of the GSSV $V_{i,j}$ determining the control branch in the model, where j is in the range of $[1, g]$ and g is the number of GSSVs associated to task τ_i, i.e., $Val(V_{i,j})$ associated with task τ_i is a function of the model time t and a set of Γ periodic tasks that can change the value of $V_{i,j}$ at runtime, given by (2).

$$Val(V_{i,j}) = f(t, \Gamma) \tag{2}$$

The WCET estimate on the task with respect to $V_{i,j}$ is:

$$C_i^p(V_{i,j}) = Sel(Val(V_{i,j}), C_{V_{i,j}}) \tag{3}$$

where $C_{V_{i,j}} = C_{V_{i,j},1}, ..., C_{V_{i,j},k}$, $C_{V_{i,j},k}$ is an execution-time specified in the *kth* branch of the control structure in the system, such as `if-else`, `switch-case`, Sel is a function returning the argument specified by the first argument, expressed as $Sel(x, y_0..., y_{n-1}) \mapsto y_x$. Since there are g GSSVs associated with task τ_i, the corresponding WCET of g V sections, considering the GSSVs, is expressed as follows:

$$\sum_{j=1}^{g} C_i^p(V_{i,j}) = \sum_{j=1}^{g} Sel(Val(V_{i,j}), C_{V_{i,j}}) \tag{4}$$

The message passing between two non-blocking tasks in the system model is asynchronous communication, i.e., the sending and receiving tasks place no constraints on each other in terms of completion, in the communication process. For each task τ_i either as the sending or receiving task, there are b input buffers associated, and the execution time of message passing primitives invoked (i.e., `sendMessage` and `recvMessage`) is denoted as $C^{msg\text{-}primitive}$. The execution time required to handle the messages (may include both message passing primitives and l jobs) in the buffer $U_{i,j}$ in task τ_i, can be expressed as follows:

$$C_i^p(U_{i,j}) = m_{xi,j} \times (C^{msg\text{-}primitive} + \sum_{x=0}^{l} C(j_{i,x})) \tag{5}$$

where $m_{xi,j}$ is either the number of messages sent by sending task τ_i to buffer $U_{i,j}$, or the number of messages received from buffer $U_{i,j}$ by the receiving task τ_i. Further, the value of $m_{xi,j}$ may not be bounded by the maximum size of the buffer $U_{i,j}$ as the preemption caused by higher priority tasks may preempt the execution in terms of refilling the buffer $U_{i,j}$ with more messages at runtime. Moreover, the value of $C^{msg\text{-}primitive}$ is assumed to be a safe upper bound obtained by using static WCET analysis.

In a summary, by combining the two parts of task execution time, i.e., V and NV sections in task τ_i, the parametric WCET estimate of task τ_i is a function expressed as follows:

$$C_i^p = \sum_{j=1}^{b} C_i^p(U_{i,j}) + \sum_{j=1}^{g} C_i^p(V_{i,j}) + \sum_{j=1}^{c} C_{i,nvj} \tag{6}$$

where c is the number of NV sections in task τ_i.

2.2 System Modeling

Practically, the system model presented previously is specified by the modeling language used in RTSSim [14], which can be considered as a domain-specific language describing both architecture and behavior of task-oriented real-time systems developed in C, and running on a single processor. Its syntax and semantics are as expressive as the C programming language, and it includes the typical RTOS services to the task models, such as task scheduling (e.g. FPPS), IPC via message passing and synchronization (semaphore). RTSSim also measures response time and execution time for each finished instance of a specified task, and reports the maximum value observed during the simulation. We intentionally miss out the details for the sake of space, the interested reader can refer to [14] for a thorough description of RTSSim.

3 Worst-Case Response-Time Analysis

In this section, we present 1) the time complexity of computing the WCRT of a set of independent tasks, 2) the feasible solution to RTA of the systems with intricate task execution dependencies as introduced in Section 2, and finally, 3) give the problem definition.

3.1 Problem Complexity

As proposed in [6], the WCRT of a set of independent tasks can be numerically obtained using the following recurrence relation:

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil \times C_j \tag{7}$$

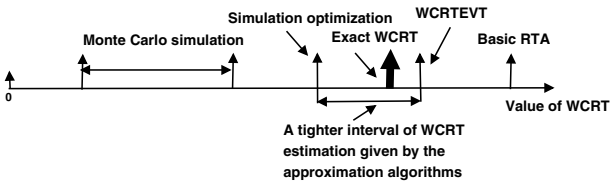

Figure 2. Illustration of applying different WCRT analysis methods in the system model presented in Section 2.

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

Equation 7 can iteratively be solved using fix-point iteration [15, 16]. Starting with $R_i^0 = C_i$ and iterating until $R_i^{n+1} = R_i^n$ is guaranteed to yield the smallest possible solution and thus the response time for task τ_i. The research in [17] recently proved that the complexity of computing such WCRT of an independent task τ_i is NP-hard, unless $P = NP$. Further, referring to the system model with intricate task execution dependencies as introduced in Section 2, the WCET of tasks C_i, i.e., the input to Equation 7, is better off being represented in terms of the symbolic formula expressed by Equation 6. Nevertheless, to use tasks' parametric WCET representation in Equation 7 is still an open and challenging question, which significantly increases the complexity of the problem. Therefore, one feasible solution is to find a tighter interval consisting of the lower and upper bound of the WCRT of tasks, given by the approximation algorithms.

3.2 Problem Formulation

In [11], a tighter lower bound of the WCRT of tasks in system models with intricate task execution dependencies as introduced in Section 2 is obtained by using a simulation optimization-based method, where a heuristic search algorithm runs on top of the traditional, Monte Carlo simulation. In this paper, we present another approximation method, *WCRTEVT* based on *Extreme Value Theory*, with the purpose of determining a tighter, meaningful upper bound of the WCRT of tasks, when compared to basic RTA. Consequently, the problem can be defined as follows. We are given a model, which can be simulated on RTSSim simulation instance 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 space S, i.e., $S \leftarrow s_1, ..., s_i, ..., s_n, n \in \mathbb{N}$, the goal of the problem is then to find an estimation that is bigger than any instance s_i in space S. Moreover, the relationship between the results obtained by different analysis methods and the exact value of the WCRT of the task on focus in the system model is illustrated in Figure 2.

4 WCRT Estimation Based on EVT

Extreme Value Theory (EVT) [13] is a separate branch of statistics for dealing with the tail behavior of a distribution. It is used to model the risk of the extreme, rare events, without the vast amount of sample data required by a brute-force approach. The example applications are hydrology, material sciences, telecommunications etc.

There are three models in EVT, i.e., the Gumbel (type I), Frechét (type II) and Weibull distributions (type III), which are intended to model random variables that are the maximum or minimum of a large number of other random variables. It is worth noting that the Frechét distribution is bounded on the lower side ($x > 0$) and has a heavy upper tail, while the Weibull model relates to minima (i.e., the smallest extreme value). Since the purpose of this work is to find the higher response time of tasks concerning rare worst-case scenarios, we therefore use the maximum case in the Gumbel distribution, referred to as Gumbel Max in the reminder of the paper.

The proposed method, *WCRTEVT* is shown in Algorithm 1. It is a recursive procedure which, as first argument, takes m data sets of which each contains N samples of the response time of the task under analysis. The algorithm returns the WCRT estimation with a predictable probability of being exceeded which is the second algorithm argument (i.e., 10^{-9} being for instance adopted by Airbus [18] in the safety-critical system domain). The outline of our WCRTEVT algorithm is as follows, which is discussed in greater detail in the following sections.

1. Construct n reference data sets by running m Monte Carlo simulations, and then choosing the n best simulations with the highest maximum value of response time of the task under analysis.
2. Perform the WCRT estimates on the task under analysis per each reference data set.
   (a) Set the initial block size b to 100, for each reference data set.
   (b) If the number of blocks $k = \left\lfloor \frac{N}{b} \right\rfloor$ is less than 30, then algorithm stops since there are not enough samples to generate an estimate.
   (c) Segment N response times into blocks of b, and for each of the $\left\lfloor \frac{N}{b} \right\rfloor$ blocks find the maximum values.
   (d) Estimate the best-fit Gumbel parameters μ and β to the block maximum values by following two procedures using two different proposed search algorithms as introduced in Section 4.2.3.
   (e) Calculate the WCRT estimation based on the estimated Gumbel parameters, i.e., μ, β, and a target acceptance probability P_e, i.e., 10^{-9}.
3. Return the lowest WCRT estimation of all the refer-

ence data sets.

4.1 The Reference Data Sets

In order to construct the input data sets to the WCRTEVT, there are m Monte Carlo simulations in RTSSim to run at first. Then the n best simulations with the highest maximum value of response time, are selected as the reference data sets. For each reference data set, there are N samples of the response time taken from the task under analysis. The value of N depends on simulation length, i.e., how long the simulation will run. Due to that the upper bound of the simulation length in the software program developed in C is $2^{31} - 1$, and in order to make sure that there are enough samples used for both estimation and validation for each reference data, we set the simulation length to be 2 000 000 000 which is quite near to the upper bound, for all the simulation models used for the evaluation purpose as introduced in Section 6. Correspondingly, for instance, there are 199 990 samples, almost half of which i.e., 99 990 samples are used in WCRT estimation for the model MV and M1 (refer to Section 6), whilst the rest of the samples are used for the validation purpose. Moreover, empirical evidence suggests that such number of samples in the estimation part of the reference data set is usually sufficient to make a good estimate. Therefore, in this work, N is either 100 000 for the model MV and M1, or 33 340 for the model M2, which is slightly different with 99 990 and 33 334 samples that are practically used by the simulator in the evaluation respectively.[2] Furthermore, the construction is showed in rows 1-3 in Algorithm 1, where x_i in line 3 is the highest response time of the task under analysis observed in simulation per each data set.

4.2 WCRT Estimation of the Reference Data Sets

4.2.1 Blocking of N Samples

In order to avoid the risk of mistakenly fitting raw response time data, that may not be from random variables, to Gumbel Max, we use the method of block maxima [13] as proposed in [19]. This is done by grouping N response time samples in each reference data set into k blocks of size b, and then choosing the maximum value from each block to construct a new set of sample "block maximum" values, i.e., $Y \leftarrow y_{i,1}, ..., y_{i,k}$, $y_{i,k} \leftarrow maxima(S) \leftarrow N_{(k-1)\times b+1}, ..., N_{kb}$ as shown in row 6, 9 and 10 in Algorithm 1. The samples at the end of the execution sequence in a simulation that do not completely fill a block are discarded. For instance, if there are 9 samples per data set, i.e., {1119, 1767, 2262, 2287, 1792, 2687, 1942, 1842, 1692}, and b (i.e., the size of the blocks) is 2, then the last sample (i.e., 1692) in the sequence is discarded since it can not be grouped in the 4 (i.e., $\left\lfloor \frac{9}{2} \right\rfloor$) blocks.

[2] The differences are less than 0.018% (i.e., (33 340 − 33 334) ÷ 33 334 = 0.018%, (100 000 − 99 990) ÷ 99 990 = 0.01%).

4.2.2 Search Space and Initial Value of Block Size b

The selection of b is a trade-off between the quality of fit to the Gumbel Max distribution, and the number of blocks (i.e., k) in each data set available and used in the estimation of the Gumbel parameters. Generally, the higher value we choose for b, the more likely it is that the block maximum values will follow a Gumbel Max, but the fewer samples, i.e., blocks, we will have available to use in the parameter estimation. Further, in order to use Chi-square test (as introduced in Section 4.2.3), the number of blocks should not be less than 30. For instance, for the model MV and M1, there are 99 990 samples in the sampling distribution per each reference data set, the corresponding block size b should be no bigger than 3 333, i.e., 99 990 ÷ 30. Therefore, 3 333 is the upper bound of the entire search space of block size b concerning the best-fit Gumbel parameters. For the lower bound of b, its theoretical value is 1, corresponding to 99 990 blocks. Unfortunately, so many blocks are extremely hard to fit to a Gumbel Max, although they are very good for parameter estimation. Moreover, empirical results showed that block size b fitting to the best-fit parameters for the Gumbel Max distribution is usually larger than 100. Therefore, in this work, the initial value of b is chosen as 100. While just in case that block size b concerning the best-fit Gumbel Max parameters located in the range of 1 and 100, Algorithm 1 will find it by performing the lower bound binary search as introduced in the following Section 4.2.3.

4.2.3 The Best-fit Gumbel Max Parameters Estimation

The estimation of the parameters of the Gumbel Max distribution is the core of WCRTEVT, which is also an iterative procedure as shown in rows 8-35 in Algorithm 1. In this paper, we introduce two procedures using two different search algorithms, i.e., *lwbsearch* and *upbsearch* which can find a proper value of b in the search space introduced in Section 4.2.2 producing the best-fit Gumbel Max parameters estimation. The algorithm *lwbsearch* is invoked at first as shown in rows 8-26 in Algorithm 1, which focuses on searching for the value of b to be as low as possible. In this way, there are more blocks, i.e., the bigger value of k, used as samples in the estimation. However, in some cases, *lwbsearch* may fail in finding such a value of b in best-fit tests. If this is the case, then *upbsearch* will be adopted, which is showed in rows 27-35 in Algorithm 1. Moreover, the best-fit test is, in terms of examining the estimated Gumbel parameters, a goodness-of-fit (GOF) test, i.e., Chi-square test at α-value of 0.05. Chi-squared test is used to determine if a

sample comes from a population with a specific distribution [20], i.e., Gumbel distribution in this work. Further, the null and alternative hypotheses are:

- H_0: the data, i.e., the maxima of the blocks follow the Gumbel max distribution;
- H_a: the data, i.e., the maxima of the blocks do not follow the Gumbel max distribution.

Note that other more advanced (meta)heuristic search algorithms can be applied. While the empirical results concerning the three models presented in Section 6.1 show that the two proposed algorithms worked well enough to reach the goal. For the sake of space, we will not show the implementation of the two search algorithms proposed. Instead, we illustrate how they work together with using a concrete example based on the data shown in Table 1.

Table 1. Illustration of using two proposed search algorithms.

step	*b*	*ALG*	χ^2	*step*	*b*	*ALG*	χ^2
1	100	lwb	×	9	2600	upb	×
2	200	lwb	×	10	2500	upb	×
3	400	lwb	×	11	2450	upb	×
4	800	lwb	×	12	2425	upb	√
5	1600	lwb	×	13	2437	upb	√
6	3200	lwb	×	14	**2443**	upb	√
7	2400	upb	√	15	2446	upb	×
8	2800	upb	×	16	2444	upb	×

The columns *b*, *ALG* and χ^2 in Table 1 represents *block size*, *search algorithm* and *result of Chi-square test at α-value of* 0.05 respectively (Chi-square test at α-value of 0.05 is referred to as χ^2 test without indicating α-value of 0.05 in the following context). Moreover, *lwb* stands for the algorithm *lwbsearch*, *upb* means the algorithm *upbsearch*, √ is *not reject* χ^2 test and × is *reject* χ^2 test. At the beginning, Algorithm 1 will try to find the value of *b* that is not bigger than 3 200 (which is close to the upper bound of the search space of *b* i.e., 3 333) and not be rejected by χ^2 test, by doubling the block size *b*. If such *b* value i.e., b_{dou} is found, then *lwbsearch* will start searching the lowest value of *b* in the range of $[100, b_{dou}]$, i.e., b^*, in the way of the lower part binary search, under the condition that the corresponding χ^2 test is not rejected. b^* is used to estimate the parameters for the Gumbel Max distribution which are considered as the best-fit parameters. While, as shown through Steps 1 to 6 in Table 1, b^* cannot be obtained by using *lwbsearch*. Therefore, *upbsearch* is followed to perform with the purpose of obtaining the highest value of *b* in the way of the upper part binary search. E.g., at Step 7, b_{lwb} (i.e., the lower bound of *b*) is set to be 1 600 (corresponding to Step 5) and b_{upb} (i.e., the upper bound of *b*) is 3 200 (corresponding to Step 6). The new *b* value to be verified by using χ^2 test at Step 7 is 2 400, i.e., (3 200 + 1 600)/2. Further, there is one point to highlight, i.e., at Step 16, the value of b is 2 444 which fails in χ^2 test. While at Step 14, when b value is 2 443, the corresponding χ^2 test passes. Hence b^* is ensured to be 2 443.

4.2.4 The WCRT Estimations Formula

The two parameters of the Gumbel Max distribution: a location parameter μ and a scale parameter β, are used in the Gumbel percent-point function, which returns the WCRT estimation that the block maximum *Y* cannot exceed with a certain probability P_e, as shown in Equation 8. Its implementation is the function *wcrtevt* with the arguments *b* (block size), *l* (location parameter), *s* (scale parameter) and P_e (acceptance probability) (refer to line 19 and 33 in Algorithm 1). How to obtain the best-fit Gumbel Max distribution parameters practically are explained in the following Section 5.

$$est = \mu - \beta \times log(-log((1 - P_e)^b)) \tag{8}$$

4.2.5 Selecting the Lowest WCRT Estimation

As the last step in WCRTEVT, the lowest WCRT estimate is selected as the WCRT estimate on all *m* data sets. This is confirmed by the empirical results by evaluating a validation model presented in Section 6.2.

5 Implementation

In this section, our testbed and the toolchain including the implemented tools are introduced in details.

5.1 Testbed

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.

5.2 Toolchain

The toolchain proposed in this work is showed in Figure 3. Moreover, the relevant tools are introduced in the following sections.

5.2.1 RTSSim

In this work, RTSSim is extended with another functionality of recording N samples of response time of a specific task during each simulation, and storing such information in a separate text file. When there are m simulations specified to execute in the batch mode, then RTSSim will generate one text file *out.txt* (which contains m lines simulation results of the highest response time for a specific task observed during each simulation), and m data set files of which each contains N response time samples of the task corresponding to each simulation.

5.2.2 ThinkStati

The core part of the toolchain, i.e., *ThinkStati*, is a prototype of WCRTEVT as an executable program with a simple user interface developed using Microsoft C# programming language and .NET framework 2.0. The software 1) reads one output of the RTSSim simulator, i.e., the reference data set file containing N (i.e., either 100 000 or 33 340 for the different evaluation models as explained in Section 4.1) samples of response time of the task on focus, at first, then 2) generates a text file *yblock.txt* for each reference data set after segmenting the samples as introduced in Section 4.2.1, then 3) produces the WCRT estimation on tasks under analysis according to the best-fit Gumbel Max parameters (verified and returned by *EasyFit* introduced in the following section) and the acceptance probability, i.e., 10^{-9} in this work. Due to limited time, the investigation on how to use the interfaces in EasyFit, concerning the verification result of the estimated Gumbel Max parameters, has not been done yet. Correspondingly, the two proposed search algorithms have not been implemented in ThinkStati.

5.2.3 EasyFit

Since Chi-square test is required by ThinkStati, in order to decide if the estimated parameters of the Gumbel distribution are conforming to the underline samples after the segment, a commercial software *EasyFit* [20] is used in the toolchain. Given the text file *yblock.txt* which contains certain number of samples generated by ThinkStati, as the input, the Chi-square test engine embedded in EasyFit will return the results in terms of the success or failure of the hypothesis test concerning the acceptance of H_0 or *null hypothesis*. If Easyfit derives the conclusion that H_0 or null hypothesis is satisfied, which means that the estimated parameters are conforming to the underlying distribution and should not be rejected; Otherwise, the parameters estimated have to be rejected.

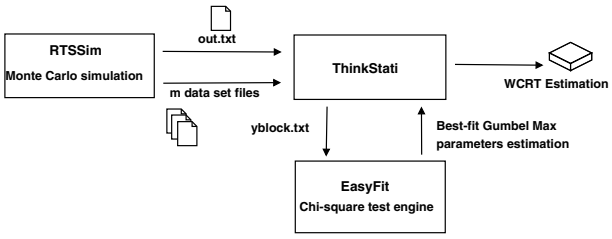

Figure 3. The toolchain in this work.

6 Empirical Results

In this section, we firstly introduce three models used for method evaluation including one validation model, then compare our solution with four other methods as reference: Monte Carlo simulation, MABERA, HCRR, and basic RTA by using the response-time computation formula (RTCF) expressed by Equation 7. Furthermore, the models are inspired by two different industrial control systems.

6.1 Evaluation Models

The three models, i.e., Model 1 (M1), Model for Validation (MV) and Model 2 (M2), have similar architecture and analysis problems as two industrial real-time applications in use at ABB [1] and Arcticus Systems [21]. M1 is representing a control system for industrial robots developed by ABB Robotics, which is not possible to analyze using methods such as RTA [2, 3]. M2 is constructed from a test application used by Arcticus Systems [21], which develops the Rubus RTOS used in many vehicular systems. We also use a simplified version of Model 1 for validation (MV). The sole purpose of this model is to investigate how close the response time estimation given by WCRTEVT is to the true, known 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), M2 and MV both use fixed priorities. Furthermore, both M1 and MV could be described by the system model proposed in Section 2. Besides, Table 2 shows the relevant information about the number of tasks, jobs, message queues, and GSSVs contained by each evaluation model.

6.1.1 Model 1 (M1)

This model represents a control system for industrial robotics, developed by ABB. It is designed to include some behavioral mechanisms from the ABB system which RTA can not take into account:

1. tasks with intricate dependencies in temporal behavior due to Inter-Process Communication (IPC) and globally shared state variables;

2. the use of buffered message queues for IPC, which vary the execution time of tasks dramatically;
3. 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 details of the model are described in [14].

6.1.2 Validation Model (MV)

MV is constructed based on M1, but the adhering task execution dependencies are simplified in that 1) globally shared state variables have been removed, 2) priority and period are strictly static, 3) loop bounds have been added manually, and 4) the constant offset of tasks is removed. As a consequence, MV has considerably lower complexity, which makes both using the RTCF in basic RTA to calculate the WCRT of tasks under analysis, and achieving the exact WCRT by using simulation-based methods, e.g., Monte Carlo simulation and HCRR [11], feasible.

6.1.3 Model 2 (M2)

M2 is based on a test application from Arcticus systems, developers of the Rubus RTOS [21] which is used in heavy vehicles. This model uses a pipe-and-filter architecture, where tasks trigger other tasks through trigger ports, forming transactions. The model contains three periodic transactions and one interrupt-driven task; in total 11 tasks. The interrupt has a small jitter, while the other transactions are strictly periodic. 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 M2, the evaluation focuses on the end-to-end response time of the transaction which contains the tasks with the lowest priority. More details of the model can be found in [22].

Table 2. The number of the adhering tasks, jobs, message queues, and GSSVs in the evaluation models.

	Tasks	*Jobs*	*Message Queues*	*GSSVs*
MV	5	35	7	0
M1	5	37	8	10
M2	4	13	0	0

6.2 Results Comparison

Before we present the results in the view of comparing WCRTEVT with the other four reference methods, the number of simulations required by WCRTEVT has to be decided. The bigger such value is, the better coverage the sampling distribution will give, whilst the more time will be consumed. Empirical results show that, for WCRTEVT, 600 simulation runs of which each simulation length is 2 000 000 000 (as introduced in Section 4.1) are sufficient to obtain the qualified reference data sets. However, the number of simulation runs in WCRTEVT could be optimized when more investigation is conducted.

Table 3. Results comparison for three evaluation models, when the different simulation budgets, i.e., the number of simulations to run, are given to the methods.

	MC	*MABERA*	*HCRR*	*Basic RTA*	*WCRTEVT*
MV	4332	4332	4332	5982	**4574.556**
M1	7682	8065	8474	NA	**8610.766**
M2	6031	6002	6299	NA	**6368.742**

As shown in Table 3, when different simulation budgets (refer to Table 4) are given to different methods in order to obtain the highest WCRT of the task under analysis, for MV, the WCRT estimation achieved by WCRTEVT is 5.6% (i.e., $(4\,574.556 - 4\,332)/4\,332 \times 100\%$) more pessimistic than the exact value derived by HCRR and MC (Monte Carlo simulation), but 23.5% (i.e., $(5\,982 - 4\,574.556)/5\,982 \times 100\%$) less pessimistic when compared to the value obtained by basic RTA. While for the other two models, i.e., M1 and M2, to which the basic RTA cannot be applied, the WCRT estimations given by WCRTEVT cover the best results obtained by both MC (Monte Carlo simulation), MABERA and HCRR as presented in [11]. This shows that WCRTEVT has a potential to provide meaningful results, i.e., as a tighter upper bound of the WCRT estimation in the response time analysis of real-time systems with more complex execution dependencies between tasks, specially when basic RTA cannot be applied.

Table 4. The number of simulations required to execute by each method.

	MC	*MABERA*	*HCRR*	*WCRTEVT*
MV	10000	10000	10000	**600**
M1	8140000	8140000	10000	**600**
M2	10000	10000	10000	**600**

Table 5. The computation time corresponding to the number of simulations required to execute by each method.

	MC	*MABERA*	*HCRR*	*WCRTEVT*
MV and M1	36133.46 s	36133.46 s	44.39 s	**1716.73 s**
M2	22.3 s	22.3 s	22.3 s	**318.31 s**

Table 6. The results obtained by the methods as reference when given the same simulation budget, i.e., 600 runs.

	MC	*MABERA*	*HCRR*	*WCRTEVT*	*True WCRT*
MV	3987	4332	3962	**4574.556**	4332
M1	7279	7379	8349	**8610.766**	NA
M2	5976	4951	5942	**6368.742**	NA

Moreover, as shown in Table 4, the number of simulations required to execute MC, MABERA and HCRR, are far more than the simulations executed in WCRTEVT. Especially, for the most realistic model M1, both MC and MABERA spend 13565.67 times (i.e., (8 140 000 – 600)/600) as many simulations compared to what is required by WCRTEVT. While for HCRR, such number is much lower, but still 15.67 times (i.e., (10 000 – 600)/600) as many compared to WCRTEVT. In other words, WCRTEVT only needs 6% ($600/10\,000 \times 100\%$) simulation budget to produce the WCRT estimation which covers the best results found by HCRR, i.e., the best of three referenced simulation-based methods.

Regarding the computation time consumed by each method used in the evaluation, Table 5 shows that the computation time spent by WCRTEVT is either 37.67 times (i.e., $(1\,716.73 - 44.39) \div 44.39$) (for the model MV and M1), or 13.27 (i.e.,, $(318.31 - 22.3) \div 22.3$) (for the model M2) times, as much as the time consumed by the fastest referenced method HCRR, respectively. This is because the number of samples in each data set currently chosen in WCRTEVT (which is not optimal) is more than the optimal number of samples required by the other simulation-based methods as introduced in [11]. However, by using WCRTEVT to obtain the results covering the best results found by other referenced methods can be managed at most in 1716.73*seconds*, i.e., 28 minutes approximately, which is reasonably acceptable.

On the other hand, if all the methods are given by the same simulation budget, i.e., 600 simulations as executed by WCRTEVT, with the exception of MV, both MC, MABERA and HCRR could not reach the highest known response time value as shown in Table 6. For MV, only MABERA finds the true WCRT, i.e., 4 332. This shows that when given the same simulation budget, WCRTEVT could perform the best-effort response time estimations.

7 Scalability of the Method

In this work, the reference data sets to WCRTEVT are collected by running Monte Carlo simulation, which in general scales to the larger size systems that are for instance constructed by creating independent "subsystems" where each subsystem is a complete model out of the three models introduced in Section 6.1. More details of using "subsystems" for scalability evaluation can be found in [22]. Moreover, the maximum number of the task's response time samples in each reference data set is constant, i.e., 100 000, which is easily handled by Chi-square test engine in EasyFit with the estimated Gumbel distribution parameters returned in a few seconds, on our testbed. One disadvantage of the current implementation of WCRTEVT, i.e., in ThinkStati, is that the two proposed search algorithms haven't been integrated yet, 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 when extra time and more investigation are given, the integration could be managed given a reasonable effort, with the purpose of toolchain automation.

8 Related work

This section introduces the work that are not mentioned in Section 1, but related. [19] presents the work on predicting on how likely a WCET estimate generated by EVT will be exceeded in the future, for a single trace. Moreover, the search algorithm concerning the best-fit Gumbel distribution parameters is done in a simple way, by only doubling the block size. A tool for statistical analysis of hard real-time scheduling algorithms is introduced in [23]. The data which are used for statistical analysis can be collected by running multiple simulation instances. However, the underline simulation model cannot capture the tasks execution dependencies inspired by real industrial control systems as introduced in this work. Moreover, there is no discussion on which kind of statistical analysis that could be performed. In [24], another probabilistic framework extending RTA to incorporate a probabilistic characterization of task arrivals and execution times is presented. However, task execution dependencies such as runtime changeability of tasks priority and period, message-passing, are not taken into consideration.

9 Conclusions and Future Work

This paper has presented work on performing worst-case response time analysis for system models with intricate execution dependencies between tasks, by using the proposed statistical-based method based on extreme value theory. Specially, we have presented and validated the method by using three models inspired by two real industrial control systems, which shows the benefit over basic RTA, in terms of reduced pessimism, and much fewer simulations required to execute when compared to three other referenced simulation-based methods. Contrary to existing stochastic real-time analysis, the proposed method is not restricted by the assumption that tasks are independent, that the job-level priority is fixed and that the worst-case scenario only happens in the case of the critical instance. As part of future work, our effort would be spent on evaluating much larger system models by using "subsystems", and addressing toolchain automation. More important, we will investigate the possibility of evaluating the proposed method on real systems, by applying more industrial standards on the toolchain. The investigation on confidence interval of the WCRT estimation given by WCRTEVT is also one interesting issue to assess.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research via the strategic research centre PROGRESS.

References

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

[2] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historical perspective," *Real-Time Systems*, vol. 8, no. 2/3, pp. 129–154, 1995.

[3] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[4] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A metaheuristic approach for best effort timing analysis targeting complex legacy real-time systems," in *Proc. of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 08)*, April 2008, pp. 258–269.

[5] Y. Lu, T. Nolte, I. Bate, and C. Norström, "Timing analyzing for systems with execution dependencies between tasks," in *Track on Real-Time Systems, The 25th ACM Symposium on Applied Computing (SAC2010)*. ACM, March 2010.

[6] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal (British Computer Society)*, vol. 29, no. 5, pp. 390–395, October 1986.

[7] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, ser. LNCS, M. Bernardo and F. Corradini, Eds., no. 3185. Springer–Verlag, September 2004, pp. 200–236.

[8] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times - a tool for modelling and implementation of embedded systems," in *TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. London, UK: Springer-Verlag, 2002, pp. 460–464.

[9] "Rapita systems, www.rapitasystems.com, 2008."

[10] D. Decotigny and I. Puaut, "ARTISST: an extensible and modular simulation tool for real-time systems," in *Proc. of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '02)*, 2002, pp. 365–372.

[11] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte, "Simulation-based timing analysis of complex real-time systems," in *The 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 09*, August 2009, pp. 321–328.

[12] 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 *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07)*. IEEE Industrial Electronics Society, September 2007, pp. 101–110.

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

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

[15] N. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "Hard real-time scheduling: The deadline-monotonic approach," in *in Proc. IEEE Workshop on Real-Time Operating Systems and Software*, 1991, pp. 133–137.

[16] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, pp. 284–292, 1993.

[17] F. Eisenbrand and T. Rothvoβ, "Static-priority real-time scheduling: Response time computation is np-hard," in *RTSS '08: Proceedings of the 2008 Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 397–406.

[18] "Airbus, www.airbus.com/en/, 2009."

[19] S. H. J. Hansen and G. Moreno, "Statistical-based wcet estimation and validation," in *9th Int'l Workshop on Worst-Case Execution Time Analysis*, 2009, pp. 123–133.

[20] "Easyfit, www.mathwave.com/products/easyfit.html, 2009."

[21] "Website of Arcticus Systems," www.arcticus-systems.se.

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

[23] J. Goossens and C. Hernalsteen, "A tool for statistical analysis of hard real-time scheduling algorithms," in *SS '98: Proceedings of the The 31st Annual Simulation Symposium*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 58–65.

[24] A. Burns, G. Bernat, and I. Broster, "A probabilistic framework for schedulability analysis," in *Proceedings of the Third International Conference on Embedded Software (EMSOFT 2003*, 2003, pp. 1–15.

Algorithm 1 $WCRTEVT(m, P_e)$

$RT \leftarrow rt_1, ..., rt_m \leftarrow MonteCarlo(m, rnd\_inst())$
$n \leftarrow \frac{m}{100}$
$X \leftarrow x_1, ..., x_i, ..., x_n \leftarrow selectHRT(n, RT)$
for all x_i such that $1 \leq i \leq n$ **do**
  $b \leftarrow 100$
  $k \leftarrow \left\lfloor \frac{N}{b} \right\rfloor$
  $success \leftarrow false$
  while $k \geq 30$ **and** $success = false$ **do**
    $S \leftarrow s_{i,1}, ..., s_{i,k} \leftarrow segment(N, b)$
    $Y \leftarrow y_{i,1}, ..., y_{i,k} \leftarrow maxima(S)$
    if $passChiSquareTest(Y) > 0$ **then**
      $lwb \leftarrow \frac{b}{2}$
      $upb \leftarrow b$
      $b \leftarrow \left\lfloor \frac{lwb + upb}{2} \right\rfloor$
      while $success = false$ **do**
        $success \leftarrow lwbsearch(b, Y)$
        if $success = true$ **then**
          $l, s \leftarrow ChiSquareTest(Y)$
          $est_i \leftarrow wcrtevt(b, l, s, P_e)$
        end if
      end while
    else
      $b \leftarrow 2 \times b$
      $k \leftarrow \left\lfloor \frac{N}{b} \right\rfloor$
    end if
  end while
  $upb \leftarrow b$
  $b \leftarrow \frac{b + \frac{b}{2}}{2}$
  while $success = false$ **do**
    $success \leftarrow upbsearch(b, Y)$
    if $success = true$ **then**
      $l, s \leftarrow ChiSquareTest(Y)$
      $est_i \leftarrow wcrtevt(b, l, s, P_e)$
    end if
  end while
end for
$EST \leftarrow est_i, ..., est_n$
$rt_{est} \leftarrow min(EST)$
return rt_{est}