Using Design of Experiments to Optimise a Decision of Sufficient Testing

Mahnaz Malekzadeh[1], Iain Bate[1,2], Sasikumar Punnekkat[1]
[1]Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden
[2]Department of Computer Science, University of York, York, UK
{mahnaz.malekzadeh, sasikumar.punnekkat}@mdh.se, iain.bate@york.ac.uk

***Abstract*—Testing of safety-critical embedded systems is an important and costly endeavor. To date researchers and practitioners have been mainly focusing on the design and application of diverse testing strategies, but leaving the test stopping criteria as an ad hoc decision and an open research issue. In our previous work, we proposed a convergence algorithm that informs the tester when the current testing strategy does not seem to be revealing new insight into the worst-case timing properties of tasks and hence should be stopped. This algorithm was shown to be successful but its trial and error tuning of parameters was an issue. In this paper, we use the *Design of Experiment (DOE)* approach to optimise the algorithm's performance and to improve its scalability. During our experimental evaluations the optimised algorithm showed improved performance by achieving relatively the same results with 42% less testing cost as compared to our previous work. The algorithm also has better scalability and opens up a new path towards achieving cost effective non-functional testing of real-time embedded systems.**

I. Introduction

Embedded systems are often used in safety-critical applications where failures can lead to catastrophic damage to people or environment. Testing is an extremely important part of the development and certification process but is also one of the most expensive parts. Therefore, testers have to determine whether there is any benefit in running the current testing strategy further. Currently this is at best a qualitative decision. Such a decision also plays an important role in the *As Low As Reasonably Practicable (ALARP) principle* which is an underpinning concept in most safety standards. According to the ALARP principle, risk-tolerability depends on practicability of further risk-reduction which is a cost-benefit analysis, i.e., it must be feasible to demonstrate that the cost of reducing the risk further would outweigh the benefit gained.

We addressed this decision challenge quantitatively in our previous work [1] for the important problem of estimating the Worst-Case Response Time (WCRT) of Real-Time Systems (RTS) [2], i.e., a convergence algorithm was proposed based on the ALARP principle to decide when to stop testing the RTS as it was unlikely that significant new information would be obtained. The algorithms checked whether High WaterMark (HWM), i.e., the Maximum Observed Response Time (MORT) is increasing at a sufficiently fast rate as well as the distribution of response times is varying significantly.

The convergence algorithm had a set of parameters, here called *controllable factors*, which were informally tuned using limited trial and improvement experiments. The contributions of this paper are as follows:

- To use the *Design of Experiments (DOE)* approach to optimise the algorithm through tuning the controllable factors such that a better decision of when to stop testing is made and the analysis itself is more scalable.
- To present an evaluation that shows the optimisation does in fact improve the algorithm's performance and scalability.

The remainder of this paper is structured as follows. Section II describes the background, system model and the simulation environment. The convergence algorithm is stated in III. Section IV includes a motivational example followed by the problem formulation in Section V. Then, the DOE approach, the experimental results and evaluations are presented in Section VI and VII respectively. Section VIII finally draws the conclusions.

II. Background

In this section, firstly, the worst-case timing problem of RTS is described to show in what sense our algorithm helps to address it compared to the traditional Response-Time Analysis (RTA) [3] techniques. Secondly, the system model and the simulation environment for which the algorithm has been applied and evaluated are described.

A. Worst-Case Timing Properties Problem

There are diverse testing strategies being used to discover defects in embedded software systems [4], [5], [6]. However, they do not answer the question of when to stop testing, more specifically, in a quantified way. Our previous work addressed this challenge for the worst-case timing properties of RTS.

The reason to focus on WCRT problem is that the traditional RTA techniques are incapable of capturing features inhabiting complex real-time embedded systems, thus, resulting in inaccurate WCRT analysis. They are based on simplified assumptions of systems and compute an absolute WCRT provided that the load on a system is bounded and the exact WCET of each task within the system is determined. However, such a *deterministic* RTA does not apply in a real system with complex control flow behaviour of tasks due to dynamic calls, dynamic jumps [7] and explicit, implicit dependencies, e.g., complex transactions in an engine control system [8] and global state shared variables in robots' control system [9] respectively. Furthermore, RTA techniques rely on the exact

WCET estimation which itself is hard due to the advanced hardware features, temporal and execution dependencies between tasks [7], etc. In contrary, our convergence algorithm based on testing allows us not to depend on an abstract system model nor the exact WCET estimation which makes it suitable for real complex embedded systems.

B. System Model

The system model comprises a set of applications. Each application Θ_i is modeled as a task graph. Each task graph consists of a number of tasks and the communication between them. So, an application is modeled as a directed acyclic graph $\Theta_i = (A_i, B_i)$ where A_i denotes the set of tasks and $B_i \subset A_i \times A_i$ represents the set of communications between tasks. Task *j* in application Θ_i is indicated by $\tau_{ij} \in A_i$ and has a release period h_j. The time difference between completion and release time of a task is called its *response time*.

C. Simulation Environment

For evaluations, a task set simulator is used which allows a ground truth to be established and also allows careful control of the task set characteristics, including complexity. Two ground truths are available for comparison: static analysis which in this particular situation gives an exact safe result [10], and a HWM but with significantly longer simulation. Longer simulation is possible due to the nature of the simulator, however, such increased testing would be prohibitively expensive in a real system.

The simulator executes a randomly generated task set for a given duration (*SimDur*). Before the simulation starts, a random number of tasks are generated each with a *Best Case Execution Time (BCET)* and WCET. For all the tasks, except the control scheduling tasks, a period is randomly chosen within the range [MinPeriod, MaxPeriod]. Then, the deadlines of the tasks are set to be equal to the periods. Each time a task is released it is given a random execution time, according to a normal distribution, in the range [BCET, WCET]. The tasks are scheduled based on the Deadline Monotonic Priority Ordering (DMPO), i.e., the shorter the deadline, the higher the priority. Then, a scheduler controls which tasks are in the delay and run queues, and which task is currently executing.

III. Convergence Algorithm

Our earlier proposed convergence algorithm decides when to stop testing the RTS as no significant new information will be determined without clairvoyance. To accomplish its task, the algorithm relies on, firstly, HWM to examine whether the response times are increasing at a sufficiently fast rate. The HWM test helps to avoid doing expensive probabilistic technique too much. Secondly, the algorithm uses *Kullback-Leibler DIVergence (KLDIV)* test [11] to check whether the distribution models of response times are being refined.

The convergence algorithm is presented in Algorithm 1 with the following controllable factors: α, λ, *i*, δ and *NumSet*. As it is shown the response times of a task set and the proposed *Stopping Point (SP)* by the convergence algorithm form the input and the output of the algorithm respectively. The response time distributions are generated running the simulator for time *SimDur*. The factor *NumSet* defines how many data sets containing the WCRT distributions to be generated. The factor λ is to assort response times into the equally-sized bins to foil the outliers effect and to improve scalability, i.e., instead of saving every single response time, the frequencies of the response times falling in the range [s, s + BinSize] are recorded which occupy much less memory space.

For each task, the algorithm takes two overlapping data sets depicted by X and Y such that Y is a superset of X (Line 6), i.e., to gradually examine test data for convergence and to avoid further cost as soon as the convergence occurs. It, then, checks whether the HWM is increasing (Line 7) and if it has not been increased for *i* successive analysis iterations (Line 12), it goes for *KL DIV* test, otherwise, the HWM test is reset (Line 8). The criteria for *KL DIV* test being passed is that the test result falls below the δ threshold (Line 15). The algorithm stops further analysis provided that both the HWM and KL DIV tests are passed, otherwise, the HWM test is reset (Line 23) and further dataset would be analysed (Line 25).

It is worth highlighting that the higher priority tasks in the task set tend to converge sooner than the lower priority tasks. However, the algorithm stops only if the latest task within the task set converges (Line 30, 31). The latter provides the higher priority, thus, safety-critical tasks better WCRT analysis, e.g., our results in Section VII-B show that MORT at SP for the highest priority tasks is almost equal to LM (within at most 0.01%), offering acceptable estimates of worst-case timing properties observed during the simulation.

IV. Motivational Example

In our previous work, the controllable factors of the algorithm were tuned based on the Trial and Improvement (TI) approach. However, there might be other tunings to the controllable factors which result in the improved performance and scalability. To check our hypothesis we applied two different tunings depicted by $tuning_1$ and $tuning_2$ on the same low priority, thus, *risk tolerable* task within a randomly chosen task set. Table I summarizes the results and shows that the $tuning_1$ has about 13 times higher cost and 1.5 times more occupied space compared to $tuning_2$, however, the MORT in $tuning_1$ is about 1% higher than the MORT in $tuning_2$. Low priority of the task causes tolerability against risks making an ALARP judgement feasible, i.e., according to the ALARP principle 1% improved WCRT analysis in $tuning_1$ does not justify 13 and 1.5 times more testing cost and computational space respectively. So, it can be concluded that the algorithm in $tuning_2$ makes a better ALARP decision rather than $tuning_1$.

This example underlines the need for more careful formalization of the problem and detailed scientific methodology for tuning the parameters.

V. Problem Formulation

As stated earlier, we used DOE to optimise our convergence algorithm. The DOE approach has the following inputs and

Algorithm 1: The *Convergence Algorithm*

```
Input: $ResponseTimes$
Output: $AlgorithmStoppingPoint$
$BinSize \leftarrow MaxPeriod/\lambda$;
foreach Task ∈ {TaskSet} do
    $X = 1$;
    $Y = 1$;
    while $Y <= NumSet$ do
        $Y \leftarrow \alpha * X$;
        if ($CurrentMORT > OldMORT$) then
            $HWMCounter \leftarrow 0$;
        end
        else if ($CurrentMORT <= OldMORT$) then
            $HWMCounter \leftarrow HWMCounter + 1$;
            if ($HWMCounter >= i$);
            then
                run KL DIV test;
                if ($KLDIV <= \delta$);
                then
                    save current task stopping point coordinates:
                    Task(CurrentTime, CurrentMORT);
                    break;
                end
            end
        end
        else
            $HWMCounter \leftarrow 0$;
        end
        $X \leftarrow X + 1$;
        $OldMORT \leftarrow CurrentMORT$;
    end
end
foreach Point ∈ {TaskSet(CurrentTime, CurrentMORT)} do
    $LatestConvergence \leftarrow Maximum(CurrentTime)$;
    Return Task(Maximum(CurrentTime ), MORT);
end
```

TABLE I
ALGORITHM PERFORMANCE

Tunings	SPMORT	SP Testing Cost ($*10^9$)	Space ($*10^6$)	LM
$tuning_1$	80459	1792	283	81699
$tuning_2$	79332	137	182	81699

outputs to eventually draws the final tunings.

The inputs to the DOE approach are the following controllable factors from Algorithm 1:

- α: With the feasible range [2, NumSet) forms *Y* in each analysis iteration. However, its range is limited to [2, 6] as bigger values result in bigger *Y*, i.e., unnecessary test data may being analyzed whilst the convergence has already occurred. In other words, small α values help to identify the convergence as soon as possible.
- λ: With the feasible range [1, MaxPeriod] defines the number of bins. The smaller the λ, the more scalable the algorithm becomes. However, very small λ values threaten the accuracy of the distribution models. So, λ is set such that it results in both acceptably accurate distributions and scalable algorithm. Accordingly, its range is limited to [200, 1200] as higher values notoriously degrade scalability and lower values endanger accuracy of the distributions.
- *i*: With the feasible range [1, ∞) defines the number of HWM tests to be done before KL DIV test. The range is chosen to be [10, 90] for performance and scalability reasons, i.e., the lower bound is set to 10 as smaller values may weaken the test and the higher bound is set to 90 as bigger values may make the test too greedy to be useful.
- δ: With the feasible range [0, 1] where 0 presents the absence of difference between two distributions. The smaller δ, the less difference exists between the two distributions, thus, limiting its range to [0.000001, 0.1].
- *NumSet*: With the feasible range [2, *SimulationTime*/*t*] defines the number of data sets to be generated by the simulator where *t* specifies how frequent the data is logged. The smaller the *NumSet*, the more scalable the algorithm becomes. The range is set to be [8000, 10000] as lower values of *NumSet* causes bigger *t* and logging data less frequent which is in odds with the principle of ALARP, i.e., less data sets including more, possibly unnecessary, test data.

The outputs of the DOE approach are the following response metrics which measure the performance and scalability of the convergence algorithm and assess the relative quality of each each combination of values of factors, called *candidate*.

- $M_{achieve}$: Closeness of the algorithm MORT at SP (*SPMORT*) to the last MORT (*LM*) observed during simulation assuming that virtually infinite test data resources are available. SPMORT has to be reasonably close to LM when the algorithm stops, thus, giving candidates with smaller $M_{achieve}$ higher rank.

$$M_{achieve} = \frac{LM - SPMORT}{LM} \tag{1}$$

Ideally, LM has to be equal to WCRT from static analysis. However, in practice, it is not scalable especially for a low priority task. It is also less important as we want to make an ALARP decision.

- M_{alarp}: Closeness of SPMORT to a quantified MORT called *ALARP MORT (AM)*. Ideal is that the algorithm stops later than AM but not far from it. By stopping too soon before AM, SPMORT becomes far from LM and by stopping too late after AM, it may result in higher cost without gaining useful new findings as it has already fallen within the ALARP region, e.g., in this paper SPMORT within 5% of LM defines the ALARP region. Candidates with smaller M_{alarp} are ranked higher while candidates causing the algorithm stops before AM are discarded.

$$M_{alarp} = \frac{SPMORT - AM}{SPMORT} \tag{2}$$

- M_{cost}: The cost of testing in terms of the time has been spent to generate and to analyse test data.

$$M_{cost} = \frac{TestingTimeatSP}{TestingTimeatLM} \tag{3}$$

- M_{space}: Computational memory space; i.e., how much space the generated test data occupies.

$$M_{space} = SizeofDataSets \tag{4}$$

The metrics $M_{achieve}$ and M_{alarp} relate to the algorithm performance while M_{space} addresses scalability. M_{cost} relates to both performance and scalability.

- M_{ed}: Measures the overall quality of a candidate based on the response metrics $\{M_{achieve}, M_{alarp}, M_{cost}, M_{space}\}$. Assume there are *n* controllable factors and *m* response metrics denoted by $X_1 - X_n$ and $M_1 - M_m$ respectively. A candidate consisting a set of tunings $C_\kappa = \{X_{\kappa 1}, ..., X_{\kappa n}\}$ experimentally maps to a set of response metrics $RM_\kappa = \{M_{\kappa 1}, ..., M_{\kappa m}\}$. An ideal candidate C_{ideal} results in $M_i = 0$ for each $M_i \in RM_{ideal}$. However, such an ideal candidate does not necessarily exist. M_{ed} for each candidate is defined based on the *Euclidean Distance (ED)* between two points C_κ and C_{ideal} such that the candidates closer to C_{ideal} get a higher rank.

$$M_{ed} = \sum_{i=1}^{m} w_i (s_i M_i)^2 \tag{5}$$

  Weighting value depicted by w_i is introduced such that larger value shows greater importance of its associated metric. A scaling factor s_i normalizes each metric absolute value such that $s_i M_i$ locates in the range [0, 1]. M_{edMax} presents the worst candidate for a given set of weightings.

$$M_{edMax} = \sum_{i=1}^{m} w_i \tag{6}$$

  s_i is defined as $1/max(M_i)$ where max(M_i) is the largest value of metric M_i observed during all experiments.
  In this work, we set *RM* = $\{M_{achieve}, M_{alarp}, M_{cost}\}$, i.e., the response metrics measuring the performance are the most important ones. M_{space}, in the second place, helps to find the more scalable candidates. For each $M_i \in RM$, the associated w_i is set to 1 to give them the equal importance. All M_{ed} values fall in the range [0, $\sqrt{3}$] where 0 and $\sqrt{3}$ (calculated by Equation 6) show the theoretically ideal and the worst candidates respectively.

VI. DOE Approach

The DOE approach is to optimise the controllable factors such that the performance and scalability of the convergence algorithm are improved. There are totally five controllable factors involved in the optimisation which may lead to combinatorial explosion if being optimised in an inappropriate fashion, e.g., assume we are interested in assessing the quality of four points called *levels* for each factor within its range. Then, totally 4^5 = 1024 combinations of these levels across all the factors have to be assessed. If each candidate is assessed by running three simulations, it means 3072 simulations in total, which may be prohibitively expensive. DOE, however, is based upon a three-phase method to systematically explore each factor's range. Firstly, it analyses the variance of LMs with respect to *SimDur* and determines *SimDur* such that the scalability is improved. Secondly, it allows insignificant factors with minor or no effect on the performance to be identified and discarded in an earlier phase called *factor screening* before digging deeper the most significant factors. This helps to avoid the combinatorial explosion and to enhance the scalability. The three-phase DOE is as follows.

- *Phase 1: Choice of simulation duration*
  This phase determines the simulation duration (*SimDur*) through the analysis of variance such that the LM of each task within a task set sufficiently becomes stable to represent the long-term behaviour. We evaluate each candidate through simulation where smaller *SimDur* improves scalability. Thus, it is important to determine *SimDur* such that it favours not only stability of the LMs but also scalability.
- *Phase 2: Factor Screening*
  This phase identifies the minimal set of controllable factors or factor pairs which are the most influential predictors of the convergence algorithm response metrics using the ANOVA method [5]. The ANOVA method determines which set of factors and their interactions are significant at a given confidence level. The insignificant factors identified by the ANOVA are discarded to avoid combinatorial explosion and to improve scalability.
- *Phase 3: Factor tunings*
  This phase is to derive a set of tunings associated with the best quality candidate by sampling each significant factor's range at *high resolution*. The high resolution phase is feasible as levels with polynomial experimental cost growth can be increased while controllable factors with exponential experimental cost growth has been already dropped by the factor screening phase.

Our analysis results of the three-phase DOE approach are presented in Section VII.

VII. Experimental Results and Evaluations

This section, firstly, presents our experimental results from the three-phase DOE approach and the derived optimisation solution. Secondly, it includes evaluation of the DOE tunings compared to the TI tunings showing improved performance and scalability by the DOE approach over the TI method.

A. Three-Phase DOE Results

- Phase 1: Choice of simulation duration
  Let us denote the task set LMs observed during time *t* by LM_t and assume that LM for each task eventually converges to WCRT calculated by the static analysis (*StaticWCRT*), i.e., $LM_t \rightarrow StaticWCRT$ while *t* $\rightarrow \infty$. If at time *k*, LM_k becomes sufficiently close to StaticWCRT, e.g., within η% of LM_∞, we conclude that LM_ks are stabilised and any further variation would be due to the experimental error. In this paper, LMs become stable at *SimDur* = 10^{13} with η equal to 5 for a set of 10 tasks.

TABLE II
PHASE2: *p-values* FOR p = $\{\alpha, \lambda, \delta, i, NumSet\}$ AND RM = $\{M_{achieve}, M_{alarp}, M_{cost}, M_{space}\}$

Metrics	$M_{achieve}$	M_{alarp}	M_{cost}	M_{space}
α	**0.0028**	**0.0034**	**0.0001**	0.5791
λ	**0.0029**	**0.0032**	**0.0003**	**0**
δ	**0.0024**	**0.0039**	**0**	0.5811
i	0.4575	0.4142	0.9805	0.665
NumSet	0.6039	0.5614	0.6952	**0.0031**
$\alpha\lambda$	**0.003**	**0.003**	**0.0046**	0.5204
$\alpha\delta$	**0.0029**	**0.0031**	**0.0012**	0.1815
αi	0.4297	0.4481	0.9303	0.5201
α*NumSet*	0.5846	0.5798	0.7069	0.7382
$\lambda\delta$	**0.0028**	**0.0033**	**0.0002**	0.5378
λi	0.4316	0.4483	0.4783	0.5236
λ*NumSet*	0.5901	0.5747	0.5613	**0.0129**
δi	0.4016	0.4744	0.1848	0.6486
δ*NumSet*	0.5565	0.599	0.3649	0.0736
iNumSet	0.9702	0.9882	0.9238	0.2752

- Phase 2: Factor Screening
This phase includes experiments designed by the *full factorial* DOE [12] method, thus, including all possible combinations of the factors and their levels. To run the experiments the task set simulator described in II-C is used. Let us assume there are p controllable factors at q evenly-spaced levels. As we have no priori knowledge about a specific region which may offer good quality candidates, the entire range of each factor is explored and the response metrics for each candidate are derived. Then, the ANOVA method is applied to see to what extent each factor influencing the response metrics. In particular, we are interested in *p-values* [13] derived by ANOVA which are significant with 95% confidence ($p < 0.05$). The confidence level depends on the application requirements. However, 95% is generally a good and widely-used choice unless there are forcing reasons to favour an alternative value [13]. To reduce the experimental noise which may unfairly rate a candidate in a single experiment, s multiple experiment instances called *repetitions* are performed for each candidate. Each of the s repetitions is instantiated by a different pseudo-random number generator seed in the simulator to drive stochastic behaviour.
Our optimisation problem includes $p = 5$, $q = 2$ (low and high bounds of each factor's range) and $s = 3$, resulting in $2^5 * 3$ = 96 experiments. Table II shows the factors' *p-values* achieved by applying the ANOVA at 95% confidence level. The results show factors α, λ, δ and their interactions have significant influence on RM = $\{M_{achieve}, M_{alarp}, M_{cost}\}$. The factors λ, *NumSet* and their interaction are influential on M_{space}. The factor i has no significant effect on any response metric. Eventually, $p = \{\alpha, \lambda, \delta\}$ includes the significant factors influencing performance. The factor *NumSet*, influential on M_{space}, and i with no significant effect are discarded. Then, each significant factor's range is adjusted through candidate quality assessment based on M_{ed}. It is observed that the high quality candidates are associated with low value of α and high value of δ whilst they get either low or high values of λ. Insignificant factors i and *NumSet* are set based on the candidates with the lowest M_{space}; i.e., 30 and 8000 respectively. The adjusted ranges in FS phase are shown in Table III.

TABLE III
PHASE2: RANGE ADJUSTMENT BASED ON M_{ed}

Factors	α	λ	δ	i	*NumSet*
Initial Range	[2, 6]	[200, 1400]	[0.00001, 0.1]	[10, 90]	[8000, 10000]
Phase 2 Adj.	[2, 4]	[200, 1400]	[0.05, 0.1]	30	8000

- Phase 3: Factor tunings
This phase consists of two sub-phases in which q is gradually increased to conform to the ALARP principle and to favour scalability. In the first sub-phase, we identify the region offering good quality solution within each factor's range at lower resolution, then increase the resolution over interesting regions in the second sub-phase getting more candidates to be assessed. Each sub-phase includes the ANOVA method to confirm the significance of previously identified influential factors and M_{ed} assessment to rank the candidates. The sub-phases can increase until neither significant factors are identified nor a better candidate is discovered. Eventually, a candidate CS_{κ} with the lowest associated M_{ed} is chosen as the optimisation solution.
The first sub-phase includes p = 3, q = 3 and s = 3; thus, $3^3 * 3$ = 81 experiments. The ANOVA results presented in Table IV show α is influencing M_{cost}. Then, based on M_{ed}, α range is narrowed to [2, 3]. The ANOVA shows no more significant factors influencing performance at this stage. Oddly, the ANOVA results for λ do not conform to Phase 2. However, it is in accordance with the quality assessment derived in Phase 2, i.e., both high and low values of λ were associated with the high quality candidates indicating its neutral effect on performance. Factors α and λ show significant impact on M_{space}. Accordingly, the factor λ with no effect on performance is dropped at this point and is set to the lower bound for scalability. The significant factors' ranges and insignificant factors values are shown in Table V.
The second sub-phase includes $p = \{\alpha, \delta\}$, $q = \{2, 5\}$ and s = 3; resulting in 30 experiments. The best quality M_{ed}, as the optimisation solution, is associated with C_{κ} = {2, 200, 0.0625, 30, 8000} for $p = \{\alpha, \lambda, \delta, i, NumSet\}$ respectively.

B. DOE vs. TI

This section evaluates the quality of the C_{κ} achieved by the DOE method compared to the TI approach based on 100 experiments each including a set of 10 tasks. For evaluations, we assume there is one safety-critical task, offering a safety-critical service, associated with the highest priority in the task set and the rest are risk-tolerable tasks. This helps to

TABLE IV

PHASE3: *p-values* FOR SIGNIFICANT $p = \alpha, \lambda, \delta$ AND RM = $\{M_{achieve}, M_{alarp}, M_{cost}, M_{space}\}$

Metrics	$M_{achieve}$	M_{alarp}	M_{cost}	M_{space}
α	0.2164	0.148	**0.0118**	**0.0255**
λ	0.639	0.8512	0.6664	**0**
δ	0.7096	0.693	0.1151	0.8597
$\alpha\lambda$	0.1361	0.1411	0.4896	0.2015
$\alpha\delta$	0.6513	0.7637	0.2526	0.2619
$\lambda\delta$	0.2824	0.4057	0.5328	0.9401

TABLE V

PHASE3: RANGE ADJUSTMENT BASED ON RESPONSE METRICS

Factors	α	λ	δ	*i*	*NumSet*
First sub-phase Adj.	[2, 4]	[200, 1400]	[0.05, 0.1]	30	8000
Second sub-phase Adj.	[2, 3]	200	[0.05, 0.1]	30	8000

assess the performance of the convergence algorithm for the risk-tolerable tasks in isolation where the ALARP principle is applicable, further, to investigate the quality of the timing analysis with respect to safety-critical task's requirements. The preemptive tasks are not associated with overheads due to context switching, changing processor frequencies, etc and have the following characteristics.

- The tasks involved in the control software (sensor, calculation, actuator) get random execution time in the range [500, 1000].
- The Proportional-Integral-Derivative (PID) calculation task has random execution time in the range [2500, 5000].
- For the tasks not being involved in the control software, the execution time is in the range [2000, 20000].
- The tasks' *non-harmonic* periods are randomly chosen in the range [50000, 130000] such that each task set utilisation falls within the range [80%, 100%].
- The *SimDur* is set to 10^{13} from Phase 1, Section VII-A. All timings for the tasks are in microseconds.

The overall performance and scalability of the convergence algorithm is calculated for the highest priority tasks and the risk-tolerable ones separately using the following metrics. For all the metrics, the smaller the value, the better performance and scalability are achieved.

$$AlgorithmAchievement(AA) = mean(M_{achieve} * N) \quad (7)$$

$$AlgorithmEffort(AE) = mean(M_{cost} * N) \quad (8)$$

$$AlgorithmSpace(AS) = mean(M_{space} * N) \quad (9)$$

The N parameter normalizes the results such that all the metrics' values locate in the range [0%, 100%].

Figure 1 shows AA, AE and AS metrics of the DOE and TI approaches for the highest priority tasks across all the 100 experiments. In addition, Table VI summarizes the results in the form of total mean values. The results show that DOE has got better performance and scalability compared to TI, i.e., AE TI is 9.5 times greater than AE DOE (67% vs. 7% respectively) while AA TI has been improved only 0.03% compared to AA DOE. In essence, the DOE approach has gained AA almost equal to TI but with significantly less effort.

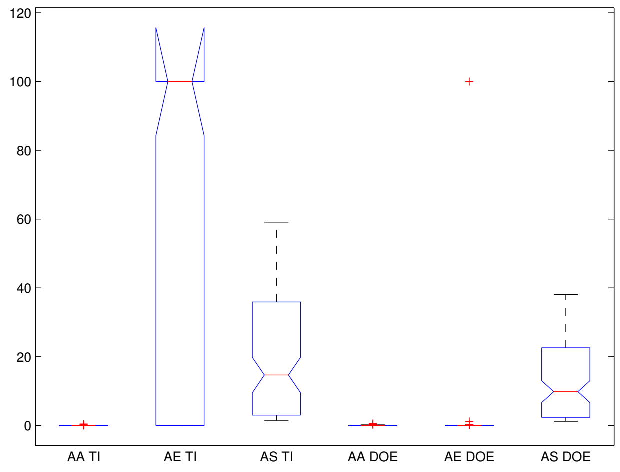

Fig. 1. The overall algorithm performance for the Safety-Critical Task

TABLE VI

THE OVERALL CONVERGENCE ALGORITHM PERFORMANCE FOR THE SAFETY-CRITICAL TASK

Approach	AA	AE	AS
TI	0.02	67.00	20.40
DOE	0.06	7.04	13.23

DOE also has less space complexity (13%) than TI (20%), resulting in enhanced scalability.

Then, Figure 2 presents the evaluation results corresponding to the risk-tolerable tasks and Table VII summarizes them. The results show that the DOE method got improved performance and scalability compared to TI. It can be seen that the TI approach totally achieves 0.7% greater AA than the DOE method. However, it does not justify 4 times more effort, i.e., according to the ALARP principle the TI achievement is disproportionate with respect to its associated cost. The DOE approach also has got better scalability compared to the TI (3% less space).

We also evaluates SPMORT vs. LM of the DOE approach for the highest and the lowest priorty tasks. The results show that the SPMORTs for the highest priority tasks are almost equal to the LMs, i.e., within at most 0.01 of LM which is an acceptable estimate. It also suggests that further versions of the algorithm may include extra criteria for the safety-critical

TABLE VII

THE OVERALL CONVERGENCE ALGORITHM PERFORMANCE FOR THE RISK-TOLERABLE TASKS

Approach	AA	AE	AS
TI	0.30	45.47	10.88
DOE	1.21	10.23	7.05

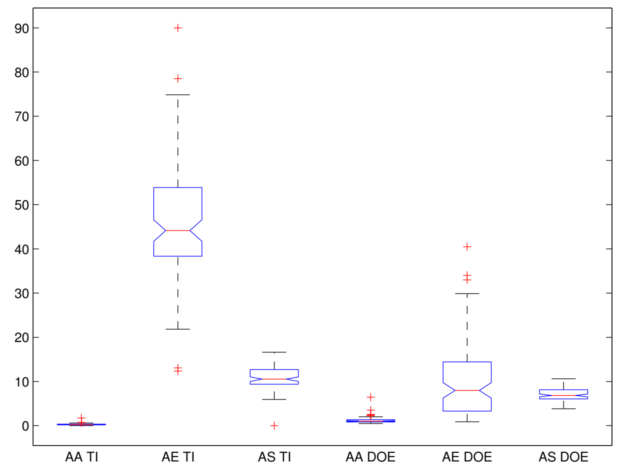

Fig. 2. The Overall Convergence Algorithm Performance for the Risk-Tolerable Tasks

tasks, i.e., the algorithm stops when criteria for both the safety-critical tasks' and ALARP are fulfilled. For the lowest priority tasks, the SPMORT is within at most 6% of the LM in average. However, based on the ALARP principle and the evaluation results for the risk-tolerable tasks, further improvement is not justified due to the significant cost.

VIII. Conclusions

Testing as an important part of the development and certification process is very expensive. Therefore, it is significantly important to determine when to stop testing. Our previous work proposed a convergence algorithm to make a quantified ALARP judgement of when sufficient testing has been done. This paper has proposed a method based on DOE to derive a set of tunings for the convergence algorithm's controllable factors to improve the performance and scalability. The derived candidate is evaluated against the TI approach which had been used in our previous work. The evaluation shows that the DOE method leads to the better performance and scalability. Future work will investigate how the convergence algorithm can be robust, i.e., it can be tuned such that it holds across many task sets and different systems.

Acknowledgement

We acknowledge the Swedish Foundation for Strategic Research (SSF) SYNOPSIS Project for supporting this work.

References

[1] M. Malekzadeh and I. Bate, "Making an ALARP Decision of Sufficient Testing," in *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE)*, 2014, pp. 57–64.

[2] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed. Kluwer Academic Publishers, 1997.

[3] N. C. Audsley, A. Burns, R. I. Davis, K. Tindell, and A. J. Wellings, "Fixed priority pre-emptive scheduling: An historical perspective," *Real-Time Systems*, vol. 8, no. 2-3, pp. 173–198, 1995.

[4] B. Beizer, *Software Testing Techniques (2Nd Ed.)*. Van Nostrand Reinhold Co., 1990.

[5] C. Kaner, J. L. Falk, and H. Q. Nguyen, *Testing Computer Software, Second Edition*, 2nd ed. John Wiley & Sons, Inc., 1999.

[6] B. Beizer, *Black-box Testing: Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, Inc., 1995.

[7] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Muller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem–overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, pp. 1—53, 2008.

[8] I. Bate and A. Burns, "An integrated approach to scheduling in safety-critical embedded control systems," *Real-Time Syst.*, vol. 25, no. 1, pp. 5–37, Jul. 2003.

[9] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A metaheuristic approach for best effort timing analysis targeting complex legacy real-time systems," in *the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008, pp. 258–269.

[10] I. Bate and A. Burns, "An integrated approach to scheduling in safety-critical embedded control systems," *Real-Time Systems Journal*, vol. 25, no. 1, pp. 5–37, 2003.

[11] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics (AMS)*, vol. 22, no. 1, pp. 79–86, 1951.

[12] D. Montgomery, *Design and Analysis of Experiments, 8th Edition*. John Wiley & Sons, Incorporated, 2012.

[13] G. Box, J. Hunter, and W. Hunter, *Statistics for experimenters: design, innovation, and discovery*, ser. Wiley series in probability and statistics. Wiley-Interscience, 2005.