Influential Nuisance Factors on a Decision of Sufficient Testing

Mahnaz Malekzadeh[1(✉)] and Iain Bate[1,2]

[1]Mälardalen Real-Time Research Centre, Mälardalen University, Västerås 72123, Sweden
[2]Department of Computer Science, University of York, York YO105DD, UK
email: mahnaz.malekzadeh@mdh.se and iain.bate@york.ac.uk

Abstract. Testing of safety-critical embedded systems is an important and costly endeavor. To date work has been mainly focusing on the design and application of diverse testing strategies. However, they have left an open research issue of when to stop testing a system. 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 system tasks, hence, should be stopped. This algorithm was shown to be successful while being applied across task sets having similar characteristics. For the convergence algorithm to become *robust*, it is important that it holds even if the task set characteristics here called *nuisance factors*, vary. Generally speaking, there might be either the main factors under analysis, called *design factors*, or nuisance factors that influence the performance of a process or system. Nuisance factors are not typically of interest in the context of the analysis. However, they vary from system to system and may have large effects on the performance, hence, being very important to be accounted for. Consequently, the current paper looks into a set of nuisance factors that affect our proposed convergence algorithm performance. More specifically, it is interested in situations when the convergence algorithm performance significantly degrades influencing its reliability. The work systematically analyzes each nuisance factor effect using a well-known statistical method, further, derives the most influential factors.

Keywords: Testin; Safety; ALARP; Nuisance Factor; Real-Time System; ANOVA; Analysis of Variance

1 Introduction

Testing is an important part of the development and certification process in safety-critical systems in which failure can lead to catastrophic damage to people or environment. However, it 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 testing the Worst-Case Response Time (WCRT) of Real-Time Systems (RTS) [2] in which the correctness of the software not only depends on the functional correctness but also on the timely delivery of the computational results. In [1], We proposed a convergence algorithm 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 algorithm checked whether the *High WaterMark (HWM)*, which represents the *Maximum Observed Response Time (MORT)* during testing, is increasing at a sufficiently fast rate as well as the distribution of response times is varying significantly.

Our convergence algorithm got a set of design factors which were initially tuned using limited trial and improvement experiments. Further in [3], we used the *Design of Experiments (DOE)* approach to tune the design factors such that a better decision of when to stop testing is made and the analysis itself is more scalable. The experimental results showed that the tuning did improve the algorithm performance and scalability.

The convergence algorithm, so far, has been evaluated with task sets having similar characteristic. However, to have a robust algorithm, it is important that it holds when the task set characteristics change, i.e., in the presence of nuisance factors. A nuisance factor may be sometimes *unknown* and *uncontrolled*, i.e., we do not know that it exists and is even changing during the experiments. Such a nuisance factor may affect the process output. A design technique called *randomization* is used which helps averaging out the nuisance factor effect. However, there is a potentially serious problem with randomized experiment if the nuisance factor *significantly* affects the process output. To cut off the nuisance factor effect, firstly, the nuisance factor with large effect on the output has to be identified which helps to systematically control the nuisance source of variability. Secondly, when the nuisance factor becomes *known* and *controllable*, we can eliminate its effect using appropriate design techniques which, in effect, leads to robustness to conditions that can not be easily controlled.

The contributions of this paper are to address the concern raised by the presence of nuisance factors for our convergence algorithm and are as follows.

- To propose a set of nuisance factors to find out whether they have any significant effect on the convergence algorithm performance, also called *response*. The intuition behind choosing each factor is its effect on the worst-case timing properties of the task set.
- To systematically analyze the effect of each nuisance factor through *analysis of variance (ANOVA)* to see whether the factor does in fact influence the algorithm response and to eventually identify the most influential nuisance factors.

– To take the first step towards robust design of the algorithm by identifying under what conditions the algorithm response significantly degrades which also relates to the reliability, i.e., the likelihood of the algorithm failure in the presence of a nuisance factor. Robust design of the algorithm, further, tries to reduce the failure.

The remainder of this paper is structured as follows. Section 2 describes the background of the work. The convergence algorithm, system model and simulation environment in which we run our experiments are stated in 3. Section 4 includes the problem statement and the ANOVA approach for identifying the nuisance factors followed by the experimental results in Section 5. Section 6 finally states the conclusions and future work.

2 Background

This section describes the worst-case timing properties of real-time systems, the problems associated with the traditional timing analysis techniques and in what sense our algorithm tries to tackle those problems, followed by a related work. Safety-critical embedded systems are expected to work properly under extreme and uncontrollable conditions which significantly raises the requirements on their dependability and reliability. They also have real-time characteristics need to be fulfilled as part of the safety requirements which makes the worst-case timing analysis an important and necessary task. The traditional Response-Time Analysis (RTA) [4] techniques, however, are incapable of capturing features inhabiting complex real-time systems, thus, resulting in inaccurate WCRT analysis. They also depend on the exact *Worst-Case Execution Time (WCET)* of each task which itself is hard to be gained due to the advanced hardware features, temporal and execution dependencies between tasks [5], et cetera. Our convergence algorithm looks into the MORT of the tasks during testing and their distributions using HWM and a statistical test respectively such that it depends neither on an abstract system model nor the exact WCET estimation which also makes it suitable for real systems.

To the best of our knowledge there is no similar work on making a decision of sufficient testing, e.g., the authors in [6] look into the HWM and the distributions of the WCET in multi-path, therefore realistic programs, to estimate the WCET. They collect execution times by running the program under analysis and pick the HWM within randomly formed blocks of data, then, examine whether the HWMs matches one of the *Extreme Value Theory (EVT)* [7] distributions. They compare two successive distributions to see whether they have converged, thus, no more observations need to be collected. They estimate the WCET using the resulted EVT distribution. Although we use HWM and the worst-case timing distributions similar to theirs, our convergence algorithm, firstly, applies the HWM on the MORTs as it is a relatively cheap test. Then, it looks into the MORT distributions to see whether they are getting converged rather than the WCET distributions. Eventually, our goal is to derive a stopping point for testing rather than WCET estimation in [6].

3 Convergence Algorithm

Our proposed convergence algorithm in [1] decides when to stop testing the RTS as no significant new information will be determined without clairvoyance. The system model assumed and the task set simulator used by the convergence algorithm are as follows.

- *System Model* comprises a set of applications. Each application consists of tasks which are assigned unique priorities according to some policy. Each *periodic* task τ_i gives rise to an infinite sequence of invocations separated by a period T_i. T_i represents the minimum time between successive invocations. A task performs an amount of computation bounded by C_i during each invocation which has to be completed by its deadline D_i. The time difference between completion and release time of a task is called its *response time*.
- *Task Set Simulator* generates testing data that allows a ground truth to be established and 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 [8], 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 generates a set of preemptive tasks with no overheads and the following characteristics.
  - Total utilisation of the task set which falls within the range [80%, 100%].
  - Each task utilisation U_i which is generated using the *UUniFast* algorithm [9] to generate random tasks with uniform distributions.
  - Each task i execution time (depicted by C_i) which is set using the following equation.

$$C_i = U_i T_i \tag{1}$$

  The simulation duration is set to 10^{13} when the MORTs of the tasks within a set of 10 fall within 5% of the last MORT observed during the whole simulation. All timings are in microseconds.

Algorithm 1 presents the convergence algorithm having the following design factors: α, λ, i, δ and *NumSet*. 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 factor *NumSet* defines how many data sets of the MORT distributions to be generated. The factor λ defines the number of bins and is to assort response times into equally-sized bins, each of size *BinSize*, 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.

For each task, the algorithm takes two overlapping distributions depicted by X and Y such that Y is a superset of X (Line 7), i.e., to gradually examine testing data for convergence. The algorithm, firstly, checks whether the HWM is increasing (Line 9) and if it has not been increased for i successive analysis iterations (Line 14). If the HWM is passed, the algorithm checks whether the distribution models of response times are being refined using the *Kullback-Leibler*

DIVergence (KLDIV) test [10]. Otherwise, the HWM test is reset (Line 10). The criterion for the the *KL DIV* test being passed is that the test result falls below the δ threshold (Line 17). The algorithm stops further analysis provided that both the HWM and KL DIV tests are passed, otherwise, the HWM test is reset (Line 25) and further datasets would be analysed (Line 27). *SPMORT* in Line 19 and 31 corresponds to the MORT value, observed for each task, when the algorithm stops. 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 32, 33).

Algorithm 1: The *Convergence Algorithm*

```
   Input: ResponseTimes
   Output: AlgorithmStoppingPoint
 1 BinSize ← MaxPeriod/λ;
 2 foreach Task ∈ {TaskSet} do
 3 |   X = 1;
 4 |   Y = 1;
 5 |   OldMORT = 0;
 6 |   while Y <= NumSet do
 7 |   |   Y ← α * X;
 8 |   |   CurrentMORT = Maximum(ResponseTimes ∈ Y);
 9 |   |   if (CurrentMORT > OldMORT) then
10 |   |   |   HWMCounter ← 0;
11 |   |   end
12 |   |   else if (CurrentMORT <= OldMORT) then
13 |   |   |   HWMCounter ← HWMCounter + 1;
14 |   |   |   if (HWMCounter >= i);
15 |   |   |   then
16 |   |   |   |   run KL DIV test;
17 |   |   |   |   if (KLDIV <= δ);
18 |   |   |   |   then
19 |   |   |   |   |   save task testing time and MORT when the algorithm passes both tests:
   |   |   |   |   |   Task(TestingTime, SPMORT);
20 |   |   |   |   |   break;
21 |   |   |   |   end
22 |   |   |   end
23 |   |   end
24 |   |   else
25 |   |   |   HWMCounter ← 0;
26 |   |   end
27 |   |   X ← X + 1;
28 |   |   OldMORT ← CurrentMORT;
29 |   end
30 end
31 foreach Point ∈ {TaskSet(TestingTime, SPMORT)} do
32 |   LatestConvergence ← Maximum(TestingTime);
33 |   Return Task(LatestConvergence, MORT at LatestConvergence);
34 end
```

4 Approach and Problem Formulation

For any testing strategy to be valid for a range of systems, it needs a clear understanding of what parameters of the system could make it invalid. These parameters are called nuisance factors. The nuisance factors, in our case, relate to those factors seem to be influential on the MORT based on the scheduling theory. More specifically, we focus on the factors which lead to more complex timing behaviour of the task set while they are being changed. We limit this work to the following nuisance factors: *Period*, *Offset*, *Number of tasks*, *Harmonic vs. nonharmonic* periods where the latest corresponds to the way task period is

generated. Harmonic period requires that every task period evenly divides every longer period which is not the case for nonharmonic period.

4.1 The ANOVA Approach

As stated earlier, we use the ANOVA method to identify a set of nuisance factors which are the most influential on the convergence algorithm response. The ANOVA method determines whether any of the nuisance factors contributes to the variability transmitted to the response, further, decides which set of factors are significant at a given confidence level. In particular, we are interested in the *p-value* [11] which, in statistics, is a function of the observed sample results used for testing a statistical hypothesis. Before the test, a threshold value is chosen and is called the significance level of the test, traditionally 5% or 1% [1]. If the *p-values* are equal to or smaller than the significance level, then, it suggests that the observed data are inconsistent with the assumption of the null hypothesis correctness, thus, that hypothesis has to be rejected. In our approach, the null hypothesis suggests that the nuisance factor under analysis has no effect on the convergence algorithm response, thus, the *p-value* smaller than the significance level suggests that the null hypothesis must be rejected and eventually, derives a set of influential nuisance factors.

As the response times distributions being used in our analysis do not follow a normal distribution, the parametric ANOVA, which assumes normal distributions of data, should be replaced. Hence, we use the non-parametric analysis of variance test, called *Kruskal Wallis* test that does not depend on such an assumption. In the rest of the paper, however, we use the term ANOVA for simplicity.

4.2 Problem Formulation to Identify Nuisance Factors

We observe and analyze the effect of each nuisance factor through a set of response metrics. The response metrics relate to the algorithm performance and are defined such that the smaller values indicate better performance is achieved. They also form our ANOVA approach inputs and are as follows.

- $M_{achieve}$: Closeness of the algorithm SPMORT to the LM while LM corresponds to the last MORT 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, the smaller $M_{achieve}$, the better performance is gained.

$$M_{achieve} = \frac{LM - SPMORT}{LM} \tag{2}$$

  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_{cost}: The cost of testing in terms of the time that has been spent to generate and to analyse testing data. Similar to $M_{achieve}$, smaller values of M_{cost}

indicate better performance.

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

5 Experimental Results

This section presents the experiments and ANOVA results to identify which nuisance factors are the most influential on the convergence algorithm performance. The task set simulator described in 3 is used in the experiments.

The ANOVA test is run at two phases: Phase 1 and Phase 2. Phase 1 includes 20 experiments, called *sample size*, for each level of the potential nuisance factor under analysis and it is called a *low resolution* phase as allows us to identify the most influential nuisance factors at relatively low cost. Phase 2, here called *high resolution* phase, includes the influential nuisance factors identified in Phase 1, however, with bigger sample size. The sample size is determined such that at least 90% power would be associated with the ANOVA test. The potential nuisance factors are analyzed at the following levels.

- Period is analyzed in overlapping ranges each starts at 50000 and ends in the following upper bounds: {200000, 400000, 600000, 800000}.
- Harmonic vs. nonharmonic period includes task sets of the same characteristics except the way periods are generated.
- Offset is analyzed in overlapping ranges starting at 10000 and ending at upper bounds as follows: {50000, 100000, 200000, 300000}. The analysis also includes a level with no offset.
- Number of tasks includes experiments of {10, 30, 50} tasks within each task set.

The ANOVA results from Phase 1 are shown in Table 1. We are interested in *p-values* smaller than 0.05 that show the corresponding nuisance factor is significant on the observed response with 95% confidence. Based on the achieved *p-values*, the nuisance factors {Harmonic vs. nonharmonic, Offset, Number of tasks} are the most influential on at least one of the response metrics. For scalability reason, the set {Harmonic vs. nonharmonic, Number of tasks} is chosen to be analyzed further at the high resolution phase as they are associated with much smaller *p-values* rather than the offset, i.e., they are much more significant.

Table 1: Phase1 - ANOVA results

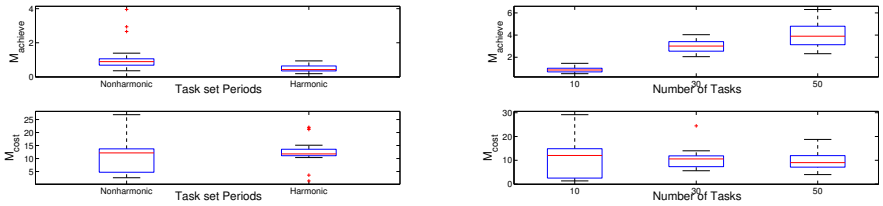
Metrics	$M_{achieve}$	M_{cost}
Period	0.5057	0.2288
Harmonic vs. nonharmonic	**0.0002**	0.7455
Offset	**0.0242**	**0.0139**
Number of tasks	**2.9186E-10**	0.8237

Table 2: Test Power & Sample Size

Nuisance Factors	Test Power	Sample Size
Harmonic vs. nonharmonic	**15%**	**265**
Number of tasks	**86%**	**27**

The box plots in Figure 1 show the algorithm performance in Phase 1 for the most infuential nuisance factors including Harmonic vs. nonharmonic period and Number of tasks. In each box plot, the *central box* represents the central

50% of the data with lower and upper boundary lines are at the 25%, 75% quantile of the data respectively. The central line indicates the median of the data and the two vertical lines extending from the central box indicating the remaining data outside the central box that are not regarded as outliers. The + sign presents outliers. In both figures the horizontal axis shows the nuisance factor levels while the vertical axis presents the response metrics values for the whole sample size. As stated earlier, smaller values of $M_{achieve}$ and M_{cost} indicate



(a) Nonharmonic, Harmonic Periods (b) Number of Tasks

Fig. 1: Algorithm Performance

that the algorithm performance has been improved, i.e., when $M_{achieve}$ and M_{cost} decrease it implies that the algorithm stops closer to the last MORT and is spending less effort to propose when to stop testing respectively. It can be seen that for both factors $M_{achieve}$ significantly degrades, i.e., becomes larger as the task sets get more complex timing behaviour while M_{cost} does not change very much. For example, $M_{achieve}$ for nonharmonic periods is 2.3 times more than harmonic periods while M_{cost} difference is 1.1 times. Also, for the task sets of size 50, $M_{achieve}$ is 4.5 times more than task sets of size 10 whereas M_{cost} does not significantly differ (1.1 times).

The results also imply that the reliability of the algorithm decreases either when we introduce nonharmonic periods to the system or increase the number of tasks which would also help to investigate the algorithm failure modes. To evaluate the algorithm performance, we introduce a quantified MORT called *ALARP MORT (AM)* relying on the ALARP principle. Ideal is that the algorithm stops later than AM but not far from it, i.e., by stopping too soon before AM, the MORT value at SP becomes far from the 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. In this paper, the MORT values within 5% of the LM defines the ALARP region.

In order to proceed with the high resolution phase, firstly, we calculate the ANOVA test power achieved by sample size 20 in Phase 1 for each influential nuisance factor and the corresponding response metrics which includes the set {Harmonic vs. non-harmonic, Number of tasks} and response metric $M_{achieve}$ respectively. Secondly, we calculate the sample size of the high resolution phase

such that 90% power of the ANOVA test would be achieved. Table 2 shows that for both nuisance factors the test power achieved by sample size 20 is less then 90%. Then, the required sample size is calculated, shown in column 3, such that at least 90% power would be associated with the ANOVA test in Phase 2.

Phase 2 includes 265 and 27 experiments for harmonic vs. nonharmonic periods and number of tasks respectively to achieve the ANOVA results with 90% power. Table 3 shows the results from Phase 2 which conform to Phase 1, i.e., both nuisance factors are influential on $M_{achieve}$, however, with much smaller *p-values* rather than Phase 1. M_{cost} also shows to be affected in Phase 2 for harmonic vs. nonharmonic factor, i.e., the test gets more powerful to identify the transmitted variability as the sample size is significantly increased.

Table 3: Phase 2 - ANOVA Results

Metrics	$M_{achieve}$	M_{cost}
Harmonic vs. nonharmonic	**1.6954E-27**	*0.0039*
Number of tasks	**4.7434E-12**	0.1158

6 Conclusions and Future Work

Testing as an important part of the development and certification process is very expensive. Therefore, it is extremely important to determine when to stop testing. Our previous work, firstly, proposed a convergence algorithm to make a quantified ALARP judgement of when sufficient testing has been done. Secondly, the algorithm was tuned based on the DOE approach to improve its performance and scalability. This paper focuses on the nuisance factors that vary from system to system and may affect the performance of the convergence algorithm. The reason is that there will be a huge bias in the experimental analysis caused by the nuisance factors if they significantly influence the algorithm response. So, it is very important that they are identified and their effect is controlled early on in the experiments. This paper proposes a set of nuisance factors, derived from system task set characteristics, that may potentially affect the convergence algorithm. Then, it systematically identifies whether the nuisance factors, in fact, influence the algorithm response.

This work also forms a stepping stone towards our future work. The future work, firstly, is around stress testing the algorithm based on the algorithm failure modes identified in this work, i.e., it uses the nuisance factor levels where the algorithm response significantly degrades. Secondly, it tries to remove the effect of the nuisance factors by robust design of the algorithm.

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 *IEEE International Symposium on High-Assurance Systems Engineering*, 2014, pp. 57–64.
2. H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed. Kluwer Academic Publishers, 1997.
3. M. Malekzadeh, I. Bate, and S. Punnekkat, "Using design of experiments to optimise a decision of sufficient testing," in *Euromicro Conference series on Software Engineering and Advanced Applications*, 2015.
4. 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.
5. 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.
6. L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs." in *Proceedings of the Euromicro Conference on Real-Time Systems*, 2012, pp. 91–101.
7. E. Gumbel, *Statistics of Extremes*, ser. Dover books on mathematics. Dover Publications, 2004.
8. 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.
9. E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, no. 1-2, pp. 129–154, 2005.
10. S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics (AMS)*, vol. 22, no. 1, pp. 79–86, 1951.
11. G. Box, J. Hunter, and W. Hunter, *Statistics for experimenters: design, innovation, and discovery*, ser. Wiley series in probability and statistics. Wiley-Interscience, 2005.