A NEW MARKOV MODEL FOR DEPENDABILITY AND TEMPORAL EVALUATION OF HARD REAL-TIME SYSTEMS

CS Perkins and AM Tyrrell
Department of Electronics, University of York
Heslington, York, YO1 5DD, UK
Email: {csp|amt}@ohm.york.ac.uk

ABSTRACT

We present a new reliability model for hard real-time systems. This model uses a generic high-level formalism based upon a Markov chain with a lattice structure which represents the progress of a computation, allowing *both* functional and time correctness of the system to be modelled. This is an improvement on traditional system reliability models which typically focus on functional correctness, and do not adequately model the temporal properties of such systems. We provide an example of the application of this model to a recovery block system, and show that a number of important metrics may readily be derived from these results. We note an unusual feature of the failure profile data, from which we hope to derive measures of the independence of the alternates in a recovery block system.

INTRODUCTION

Reliability models for fault-tolerant systems are typically based around a probabilistic process which describes the system, either neglecting execution time information, or providing a partial ordering of events only. These models allow the failure probability for a particular system to be calculated, but do not provide for calculation of the timing properties of the system. The usefulness of this class of model must hence be questioned when applied to hard real-time systems, since such systems require both temporally and functionally correct behaviour.

In this paper we propose a new system reliability model for a generic hard real-time system. This is a discrete Markov model with a lattice structure that models the progress of a computation from its initial state to one of several final states: completed, detectable fault, hidden fault, and failed. Our model allows for both functional and temporal behaviour of a system to be represented in a single high-level model; and is derived from generic properties of hard real-time systems; hence being independent of any specific design/implementation technique for such systems.

The remainder of this paper is structured as follows: We begin with a description of our new system model, and a discussion of how this may be used to analyse system behaviour. This is then illustrated by application to a recovery block system, and a number of interesting results are derived. We then summarise these results, and give pointers to further research.

SYSTEM MODEL

In order for a system to be classified as *hard real-time* it must obey certain properties. In particular, the system must have well-defined execution time bounds, and the probability of the system exceeding those bounds must be known. Given this information, and in the absence of faults, such a system may be modelled as a simple Markovian state chain with probabilistic transition to a *completed* state (figure 1).

Such a model is, of course, overly simplistic and must be extended in order to account for the presence of faults within the system. We divide such faults into two classes: (1) Those which cause run-time errors and so are detectable *before* normal system completion; and (2) those faults which do not cause such errors, and so can only be detected by examining the final system state.

The first such class of fault may be modelled by the addition of a *detectable fault* state to the Markov model describing the system. A transition is made from each state in the basic state chain to this *detectable fault* state, with probability determined as discussed below (figure 2).

The second class of fault leads to a more complex model, requiring a parallel state chain to represent a system which is still functioning, but with a hidden fault. These parallel states mimic the function of the original state chain, and lead to the *hidden fault* and *failed* states (figure 3).

This then is our final model definition, comprising two parallel state chains, representing normal execution and execution with a hidden fault. These are interconnected with a lattice structure which models hidden fault occurrence and recovery. In addition a further two states are added to the lattice in order to represent run-time detectable faults. Standard Markovian analysis may

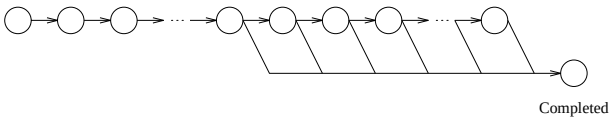

Figure 1: Basic state chain

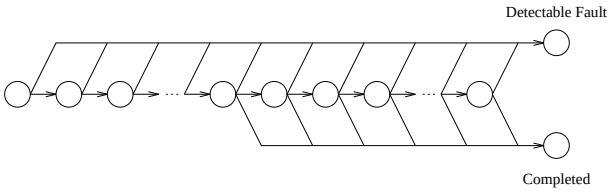

Figure 2: System model with detectable faults

then be performed to derive the system probability distribution amongst these states.

Our model may therefore be used to determine *both* the functional and the temporal correctness of a system. The *functional* correctness is indicated by the probability distribution of the system between the four final states of our model: completed, detectable fault, hidden fault, and failed. The *temporal* correctness is indicated by plotting the timing profile to show the distribution of these probabilities with respect to time.

Implicit in the above discussion has been the precise nature of the transition probabilities of the lattice model. We divide these into four categories:

Probability of completion, p_c. This is the probability that the system completes execution at any given time step. It is independent of the occurrence of faults, and must be derived from knowledge of the algorithm used by the process and/or test data. This is the transition probability for the arcs leading to the *completed* and *hidden fault* states.

Probability of detectable fault, p_d. The probability that the system fails in such a manner that can be detected before the normal completion. It may be estimated from test data, or from experience with similar systems. This is the transition probability for arcs leading to the *detectable fault* and *failed* states.

Probability of hidden fault, p_f. This is the probability that a fault occurs which does not give rise to an error detectable at run-time. Such a fault may be detected after completion of the process, and hence may be estimated based on the results of a system acceptance test. This probability, together with the probability of hidden recovery, defines the transition probabilities on the arcs interconnecting the two main state chains of our model.

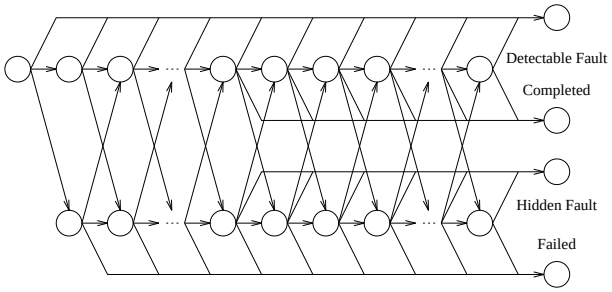

Figure 3: System model with hidden faults

Probability of hidden recovery, p_r. The probability that the system recovers silently from a hidden fault. May be estimated in a similar manner to the probability of a hidden fault.

With the exception of the completion probability, p_c, these transition probabilities are expected to be uniform, and to follow a random-fault model (Laprie and Kanoun, 1992; Musa, 1979; Perkins and Tyrrell, 1995).

It can therefore be seen that the parameters required by our model may readily be estimated based on test data from a real system. Our model is therefore of use in a predictive role: Given preliminary test data for a component we derive a reliability and timing prediction. A number of these may then be combined to predict the behaviour of an entire system.

For example, a hard real-time system may be composed of a number of components, such as recovery blocks, which combine to form a complete system. Each of these components will comprise a number of alternates. It is envisaged that the model described herein is suitable for application at the level of the individual alternate of the recovery block or other such structure. This will enable timing properties for the entire fault tolerant structure, and hence the complete system, to be derived.

APPLICATION EXAMPLE

The recovery block (Randell, 1975) is a technique which uses multiple versions of a program block to attempt to ensure success in the presence of system failures. The first version is known as the *primary* and the second and subsequent versions are known as *alternates*. The primary is executed, and an acceptance test evaluated. If this fails, the alternates are executed in series until

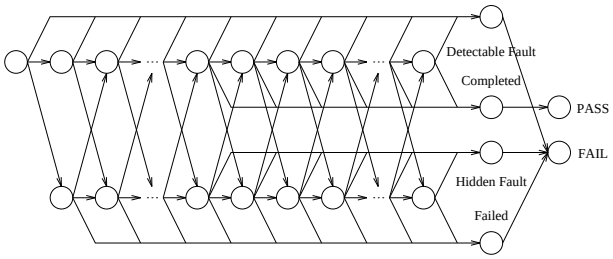

Figure 4: Alternate Model With Acceptance Test

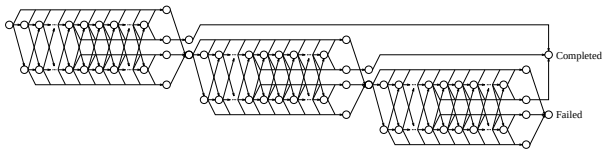

Figure 5: Recovery Block Model

one succeeds. In order for the entire system to operate successfully under hard real-time constraints, it is necessary for each alternate to operate under such constraints. Each alternate in the recovery block may, therefore, be viewed as a generic hard real-time system, and the model developed herein is applicable. In order to model the full recovery block, an acceptance test model is also required. This must map from the output states of the alternate to the final pass/fail states. A generic acceptance test will be fallible, that is, it will not correctly classify all systems, and will take a finite amount of time. For reasons of simplicity and tractability of the analysis, the test modelled here will, however, be assumed *infallible* (Csenki, 1993), and will take unit time. The study of systems with fallible acceptance tests is the subject of current research. This combined alternate and acceptance test model is illustrated in figure 4.

Several such systems may be combined in order to model a complete recovery block. This is illustrated in figure 5 for a recovery block consisting of a primary and two alternates.

In order to illustrate the applicability of our model, a system such as that in figure 5 has been analysed. For the purpose of this example, the primary and the two alternates were selected as follows (The completion probabilities for these systems are illustrated in figure 6):

Primary. A slow but reliable system, where the completion probability increases with time. For example some form of iterative solution or stepwise refinement technique.

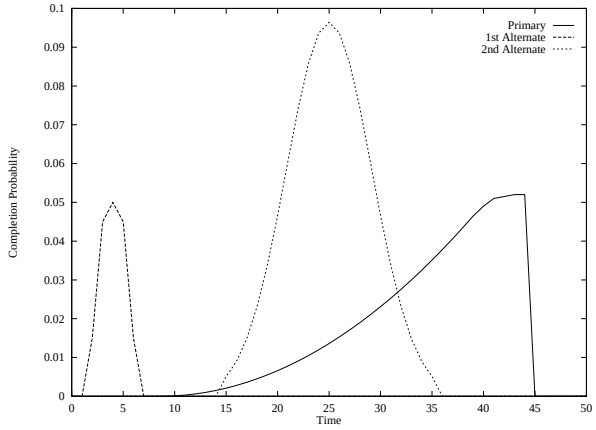

Figure 6: Basic Alternate Completion Profiles

	p_f	p_r
Primary	0.001	0.001
1st Alternate	0.010	0.010
2nd Alternate	0.010	0.005

Table 1: Alternate Parameters

System	1st	2nd	3rd
RB1	1	2	3
RB2	1	3	2
RB3	2	1	3
RB4	2	3	1
RB5	3	2	1
RB6	3	1	2

Table 2: Alternate Orderings

1st Alternate. A fast but unreliable system. For example a naive linear interpolation algorithm applied to a somewhat nonlinear system.

2nd Alternate. A reliable, medium speed system. The completion profile of this system follows a "bell-shaped" curve. For example an algebraic solution to a set of equations, where the completion time is somewhat data dependent.

There are three other parameters to the alternate model: Probability of detectable fault, p_d; probability of hidden fault, p_f; and probability of hidden recovery, p_r. In these tests p_f and p_r will be fixed for each alternate (see table 1), and p_d will be varied between 0.00001 and 0.10000.

In a recovery block system with three alternates there are a total of six possible orderings of the execution of these alternates. These orderings are shown in table 2. The behaviour of the recovery block system

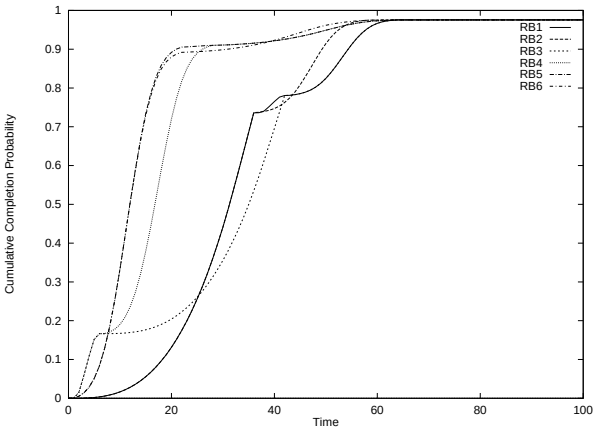

Figure 7: System Completion Probability, $p_d = 0.00001$

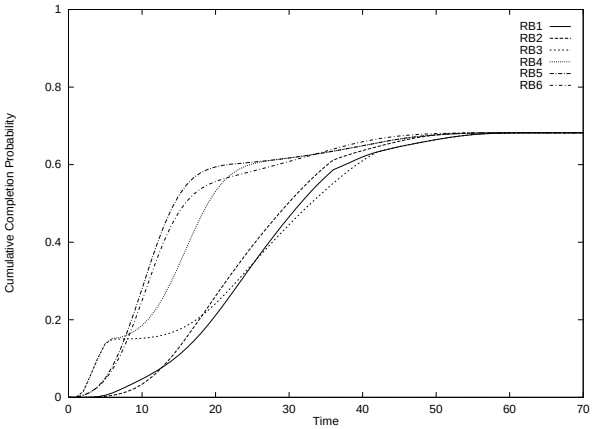

Figure 8: System Completion Probability, $p_d = 0.05000$

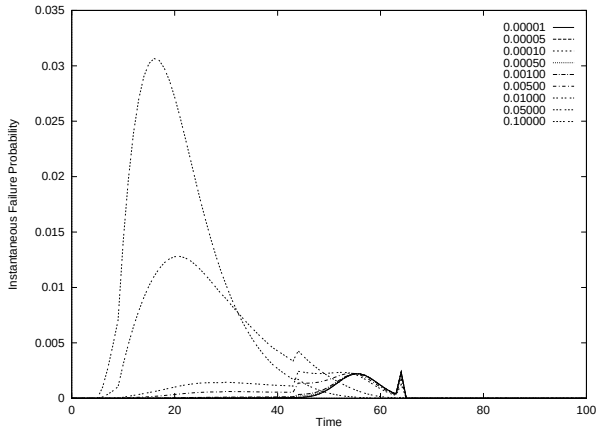

Figure 9: Instantaneous System Failure Probability

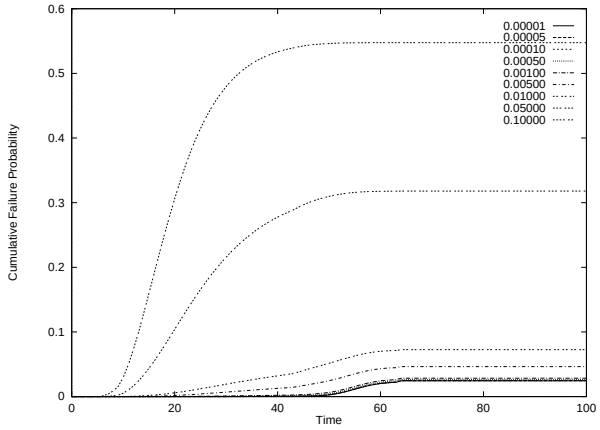

Figure 10: Cumulative System Failure Probability

has been simulated for all possible alternate orderings and a range of different forward failure probabilities. This leads to a large number of plots of system completion/failure probability *vs.* time. This data may then be analysed to determine a number of system performance metrics.

As might be expected, the ordering of the execution of the alternates and the forward failure probability has a large effect on the recovery block completion profile. Consider the data shown in figures 7 and 8: Figure 7 shows a system where the failure rate is low ($p_d = 0.00001$), and the alternates are executing almost sequentially, with failures occuring almost entirely due to time overrun. Since each alternate has a unique completion profile, we see the system completion profile change, depending on the ordering of the alternates. It can be seen that this ordering has no effect on the overall system completion probability, but does affect the time at which the system is likely to complete: The cumulative completion probability increases at different rates for the different alternate orderings.

Similar results are obtained for systems with greater failure rates, for example figure 8 with $p_d = 0.05000$. Such systems exhibit a reduced overall completion probability, as is expected due to the greater failure rate; and also show a greater divergence between the different alternate orderings. That is, a greater failure rate makes the ordering of the alternates more important for determining the system's completion time.

From information such as this a number of important system metrics may be derived, such as system completion probability, and hence reliability; and mean completion time. Accurate knowledge of a system's completion profile will also allow derivation of more optimistic scheduling strategies, which utilise a knowledge of the system's expected execution time bounds to derive a more efficient schedule than that possible by applying worst-case bounds.

In addition to this completion profile data, it is also

possible to derive system failure profile data (figures 9 10). Such data shows a very interesting feature: For a given failure rate the system failure profile is fixed, independent of the order of execution of the alternates. At first, this appears surprising: since the completion profile of the system changes with the differing orderings of alternates, the failure profile might reasonably be expected to do so too. In practice this is not so, due to a fundamental assumption of the recovery block model. This model assumes that the alternates comprising the recovery block system are independent; that is the performance of an alternate measured in isolation is the same as its performance when used in the recovery block. With this in mind, it becomes less surprising that the failure profile for the recovery block system is identical regardless of the order of execution of the alternates: If each alternate is regarded as a function transforming an input signal, then whatever order those "black-box" functions are combined the result is the same.

A number of studies have been conducted into the independence of the versions in multiversion software, (Eckhardt and Lee, 1985; Knight and Leveson, 1986). From these studies, it is clear that the alternates in a multiversion system *cannot* be assumed independent, and coincident errors are likely. This would imply that the reliability of an alternate fed only those input points on which the previous alternate has failed, is likely to be worse than the reliability of the same alternate fed a random selection of input points.

This has important consequences for the model presented above, since if the failure profile of an alternate changes depending on the ordering of the alternates, then the failure profile of the system as a whole will change accordingly. This leads to a potential method of measuring the degree of independence between the alternates in a recovery block system, by measuring the deviation of the system failure profile as the ordering of alternates is varied. This is the subject of current research.

SUMMARY AND CONCLUSIONS

To summarise; it has been noted that current reliability models are not sufficient for use in hard real-time systems design, since they do not adequately model the temporal properties of such systems. We propose a new model to overcome these limitations. This new model uses a generic high-level formalism based upon a Markov chain with lattice structure which represents the progress of a computation with respect to both time and functional correctness. We provide an example of the application of this model to a recovery block system, and show that a number of important metrics may readily be derived from these results.

One limitation of our recovery block model is that it assumes independent behaviour of alternates. For systems where this is not the case, we hope to extend our model to utilise the variation in failure profile data to derive a measure of the coupling between the alternates.

ACKNOWLEDGEMENTS

This research was supported by the UK Engineering and Physical Sciences Research Council.

REFERENCES

Csenki, A. (1993). Reliability analysis of recovery blocks with nested clusters of failure points. *IEEE Transactions on Reliability*, 42(1):34–43.

Eckhardt, D. E. and Lee, L. D. (1985). A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Transactions on Software Engineering*, SE-11(12):1511–1517.

Knight, J. C. and Leveson, N. G. (1986). An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1):96–109.

Laprie, J.-C. and Kanoun, K. (1992). X-Ware reliability and availability modeling. *IEEE Transactions of Software Engineering*, 18(2):130–147.

Musa, J. D. (1979). Validity of execution-time theory of software reliability. *IEEE Transactions on Reliability*, R-28(3):181–191.

Perkins, C. S. and Tyrrell, A. M. (1995). Reliability models for hard real-time systems. In *Proceedings of the 2nd IMA Conference on the Mathematics of Dependable Systems*, University of York.

Randell, B. (1975). System structure for software fault tolerance. *IEEE Transactions on Software Engineering*, SE-1:220–231.

BIOGRAPHY

Mr Perkins received a BEng degree in Electronic Engineering in 1992. Since that time he has been a DPhil research student at the University of York, Department of Electronics. His research interests are in the fields of software reliability modelling and fault tolerant design. He is an associate member of the IEE.