Proceeding of the First International Conference on Modeling, Simulation and Applied Optimization, Sharjah, U.A.E. February 1-3, 2005

A FRAMEWORK FOR PERFORMABILITY MODELLING USING PROXELS

Sanja Lazarova-Molnar, Graham Horton

University of Magdeburg
Department of Computer Science
Universitaetsplatz 2, 39106 Magdeburg, Germany
sanja@sim-md.de

ABSTRACT

Performability modelling and analysis is used as a measure of both performance and dependability of, mainly, fault-tolerant systems. Proxels are a paradigm that provides deterministic analysis of discrete stochastic models which contain general types of distribution functions, which is not otherwise possible using the standard approaches, such as discrete-event simulation or Markov reward modelling. Because of the flexible definition of the proxels, they can easily be extended to provide a widely-applicable approach to performability modelling.

Proxels work in such a way that they track the complete behaviour of the system and correspondingly distribute the probability, depending on the distribution functions and the time spent in each discrete state without anything happening (denoted as age intensity of the state change). Therefore the term state with respect to the proxels has been extended to include the age intensities of the relevant state changes.

In this paper we establish the framework for proxel-based performability modelling and analysis, as well as for general reward modelling. Two types of rewards are considered, impulse (associated with transitions) and rate rewards (associated with discrete states). The advantage of the proxel-based simulation is that both types of rewards can be modelled as functions of any other time-dependent quantity of the model, and models with general types of distribution functions can be analysed in a deterministic way.

For the purpose of demonstrating the method, we will include an example which will show in detail how the method works. Discussion and comparisons with some of the existing approaches will be included too, which will provide more insight into the advantages of the proxel-based method.

1. GOALS AND INTRODUCTION

The goal of this paper is to show how the basic proxel-based method can be adapted for doing performability analysis of discrete stochastic models. In that sense, establishing the formalities that are behind the adaptation process are the subject of the paper.

The motivation for this new application field of the proxel-based method is a recent practical experience with the method for analysing a warranty model, a project which we carried out for DaimlerChrysler, as described in [1]. The model that we needed to analyse involved manipulating impulse rewards in order to analyse the costs. It turned out to be a straightforward adaptation of the basic proxel method, without imposing any additional significant complications with respect to the computational and memory complexity of the implementation. Based on that experience we realised that the method can be useful for carrying out a more general performability analysis which also includes reward rates and that is the topic that we treat here.

As a beginning this section will give a short overview on the performability theory and the proxel-based method, as well as present and discuss some of the more popular existing approaches. Further we will describe and explain why the proxels are a good option for carrying out performability analysis of discrete stochastic systems and describe how it is to be achieved. To aid the comprehension and show how it works in practice, in the experiments' section we introduce an example model which we than use to test two different cases of performability modelling. Finally, in our last section we give a brief summary and an outlook of our previous and future research with respect to the topic of the paper.

1.1. Performability Modelling

Performability modelling [2] is a modelling approach to simultaneously evaluating both the performance of a stochastic system and its dependability. This measure is of an especial importance when analysing fault-tolerant systems whose performance depends on many components which fail and get repaired, i.e. behave, in a stochastic manner.

Performance alone is a measure of how efficient one system is (can be measured as throughput, response time, etc.), whereas dependability is a measure of its ability to function correctly over a specific period of time i.e. how reliable it is. Reliability (or dependability) of a discrete and stochastic system *S* can be expressed mathematically in the following form:

$$R(t) = Pr(S\, operates\, correctly\, in\, [0,t)). \qquad (1)$$

If we now denote the lifetime of a system by *L*, and *F* is the distribution function of *L*, then the reliability of the system at time *t* can be computed as

$$R(t) = Pr(L > t) = 1 - F(t). \qquad (2)$$

In common words, performability modelling tries to evaluate and answer the following question:

How much work will be done (lost) in a given interval by a given system including the effects of its failures and repairs?

and find the function that describes it. The work is then the accumulated performance over the given time interval, given that it changes over time depending on the operativeness of the separate components of the system i.e. the *performability*.

The performance of the system is measured using a reward function $P(DS, \tau)$ which evaluates its efficiency in each of its discrete states (*DS*) and can also be dependent on the time spent in it i.e. τ, or the global simulation time.

One of the most common tools for modelling and performability analysis are Markov-reward models, which operate by first constructing the continuous-time Markov chain that represents the model:

$$X = \{X(t), t \geq 0\} \tag{3}$$

and assigning *reward rates or functions* to each of the states in it. The reward rates estimate the performance of the system in each state. They can be obtained using performance analysis of the system, which means running the system in every possible configurations and evaluating the measure that characterises its performance, which is than denoted as a reward rate of that configuration (state).

The goal of performability modelling is to obtain some the following measures, depending on their relevancy:

- the *expected performance* of the system at a certain point in time, taking in account the effects of failures, repairs and all of the other possible conditions of the system,
- the *time-averaged performance* of the system over a time interval $(0, t)$,
- the *amount of work accomplished* over a time interval $(0, t)$, etc.

The last one (the amount of work accomplished) is calculated as the accumulated reward in the time interval $[0, t)$ and denoted as $Y(t)$. The time-averaged performance ($W(t)$) is then calculated as

$$W(t) = Y(t)/t. \tag{4}$$

Y(t) is generally more useful for Markov-reward models that contain absorbing states, and W(t) for the rest of them. The biggest drawback of this method is that it is not directly applicable to models that contain generally distributed events i.e. it can only analyse models with exponentially distributed activities.

One another common tool for performing performability modelling are *stochastic reward nets* (SRNs) [4], which are based on *generalised stochastic Petri nets*. Because of the equivalence between the GSPNs and Markov chains, SRNs can be mapped onto Markov-reward models by associating rewards to all tangible markings.

Recently there has been a tool introduced, TimeNET [5], that carries out performability analysis of a wide class of stochastic Petri nets (SPNs) with generally distributed firing delays (including even coloured SPNs). The tool, however, has restrictions on the number of non-exponentially distributed times with respect to the quality and type of analysis that it can provide. Its deterministic approach works on basis of a reachability graph, which limits the tool to bounded models. None of this limitations exist for the proxel-based method, which on the contrary, creates the state space on-the-fly and does not have any restrictions with respect to the number and type of distribution functions.

1.2. Proxels

Proxel-based method is a recently introduced approach [3] for deterministic analysis of discrete stochastic models. Its basis is the method of supplementary variables [6], which introduces additional variables to the discrete states of the model, which track the time that the model spends there without a state change happening with respect to the possible state changes and based on it calculates their probabilities. The elapsed time is referred to as an *age intensity* of the discrete state change. The vector which captures the relevant age intensities in a certain discrete state combined with the discrete state forms what is known as *state* in the proxel terminology.

The function which defines the rate with which a state change happens within a small $dt \to 0$, if it has been pending for a time period of *t* is known as the *instantaneous rate function* (IRF) and calculated as follows

$$IRF(t) = \frac{f(t)}{1 - F(t)} \tag{5}$$

where *f*(*t*) and *F*(*t*) are the density and the cumulative distribution functions, correspondingly. This makes it possible to compute probabilities for any state change if we know the amount of its pending time.

The method works by observing all of the possible developments in the dynamics of the model at every time step, in terms of state changes, with respect to the discretised simulation time. Having said that, proxel is a computation unit that tracks the behaviour and is defined as following

$$Proxel = (DS, \vec{\tau}, t, R, Pr) \tag{6}$$

where

- DS is the discrete state of the system,
- $\vec{\tau}$ is the age intensity vector which stores the age intensities of the relevant state changes with respect to DS,
- t is the global simulation time,
- R is the route of discrete states that lead to DS, and
- Pr is the probability for having all of the previous conditions fulfilled, i.e. that the model is in the discrete state DS with an age intensities τ at time t, having been through the route of discrete states R.

More about the method and the algorithms behind it can be found in [3][7]. In the following sections we explain the proxel-based method in conjunction with the additions for tracking performability measures. There is also presented an algorithm that shows the way that the method works.

2. PERFORMABILITY MODELLING USING PROXELS

In this section we describe how performability modelling can be carried out by using proxels and formalise the connection. Performability analysis in general works by accumulating and manipulating rewards. There are two types of rewards that are treated in this paper: rate and impulse rewards. Rate rewards are associated with discrete states and impulse rewards are associated with state changes. When measuring performability of one system, rate rewards are the evaluations of the performance of the system in different discrete states. Both can either be constant values, or functions of different parameters.

In the proxel-based approach the model is represented as a stochastic process $X = \{X(t), t \geq 0\}$ which is defined on a set of discrete states $DS = \{DS_0, DS_1, \ldots\}$, whose state changes are distributed according to certain distribution functions. We are currently working on a new proxel-adapted model description framework, but there has not yet been a publication on that to which we could refer to, so we will use the way of describing models that we have been using until now, which is by using standard state space diagrams. Performance in that case is defined as a function of the discrete states of the model:

$$rr : DS \rightarrow \Re, \text{ or} \tag{7}$$
$$rr : DS \times \Re \rightarrow \Re \tag{8}$$

where *DS* is the discrete state space of the model. The second definition (8) shows that the performance can also be a function of another parameter, besides the discrete state (such as simulation time, age intensity, or the number of times that it has failed, calculated as impulse reward). Impulse rewards are defined in a similar way, except that they are functions of the state changes. Analogously we have:

$$ir : DS \times DS \rightarrow \Re, \text{ or} \tag{9}$$
$$ir : DS \times DS \times \Re \rightarrow \Re. \tag{10}$$

If the performance of the system is dependent on anything else besides the discrete state of the model, then that element has to be included in the discrete state of the system as an additional variable. For example, in a situation where the performance of the system decreases following a given function, proportional to the number of times that a certain component has failed, the number of failures has to be included in the discrete state. This is also a very realistic assumption, which gives an additional credit to the proxel-based simulation for being able to handle such complex configurations.

Based on the given definitions, we can define the relevant measures in perfomability analysis. Some definitions are extracted from [2]:

- *Steady state performability* (SSP) is defined as

$$SSP = \sum_{i \in DS} \pi_i rr_i \tag{11}$$

  where π_i is the steady state probability of the *i*-th state,

- *Transient performability* (TP) is defined as

$$TP(t) = \sum_{i \in DS} p_i(t) rr_i \tag{12}$$

  where $p_i(t)$ is the transient probability of discrete state DS_i at time *t*

- *Expected work accomplished* (EW) is defined as

$$EW(t) = \int_0^t TP(s)ds \tag{13}$$

  and presents the accumulated transient performability to time *t*.

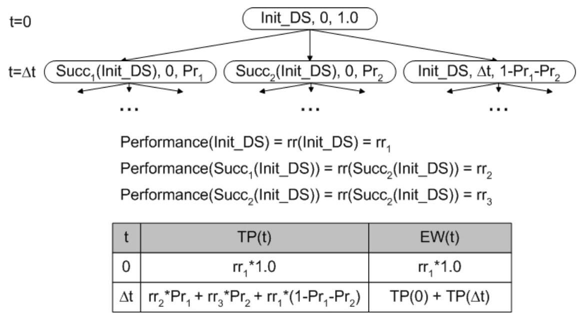
In the proxel-based method, performability measures are computed by tracking the behaviour of the model and accordingly updating them. Therefore, the *transient performability* (14) and the *expected work* (15) for every time step are computed according to:

$$TP(k\Delta t) = \sum_{\forall proxel(prox)\ at\ k\Delta t} Expr,$$
$$\text{where } Expr = Pr(prox)(rr(DS(prox)) + ir(DS(pre(prox)), DS(prox))) \tag{14}$$

$$EW(k\Delta t) = \sum_{i=0}^{k} TP(i\Delta t)\Delta t,\ where\, k = \lceil t/\Delta t \rceil \tag{15}$$

correspondingly, where $Pr(prox)$ is the probability of the proxel $prox$ and $DS(prox)$ is its discrete state component, $pre(prox)$ is the predecessor proxel of $prox$. The steady state performability can be implicitly calculated from these two measures.

In Figure 1 an illustration of the proxel-based method and the calculation of the performability measures is shown. The figures are, however, simplified to aid the comprehension. The proxel in this figure has only three components i.e. discrete state, one age intensity and probability value. $Succ_i(DS)$ denotes one of the successor discrete states (i-th) of the discrete state DS, and $Init\_DS$ is the initial one.



Performance(Init_DS) = rr(Init_DS) = rr_1

Performance($Succ_1$(Init_DS)) = rr($Succ_2$(Init_DS)) = rr_2

Performance($Succ_2$(Init_DS)) = rr($Succ_2$(Init_DS)) = rr_3

t	TP(t)	EW(t)
0	rr_1*1.0	rr_1*1.0
Δt	rr_2*Pr_1 + rr_3*Pr_2 + rr_1*(1-Pr_1-Pr_2)	TP(0) + TP(Δt)

Figure 1: *Simplified illustration of the initial steps in the proxel-based performability analysis*

The algorithm that describes the proxel-based performability analysis is shown in Figure 2. There, in lines 13, 14, 18, and 19 the calculations of transitional performability and expected work are shown, the latter one being of a cumulative nature with respect to the former one. It can be noticed that the impulse rewards only appear in lines 13 and 14, i.e. when the state changes happen, whereas in the case that the model stays in the same discrete state, they are omitted. In the figure the basic algorithm for proxel-based analysis is also shown, which operates based on two parallel data structures (in this case list), one of which stores the proxels from the previous step and the another for the proxels that are currently being calculated. This is sufficient because every proxel contains all of the necessary information in order to compute its successors.

It is noticeable that performability modelling does not introduce any additional complications to the existing algorithm and fits straightforwardly into the existing framework. In order to show this practically in the next section we show how the analysis on a simple example model work.

3. EXPERIMENTS

Here we present a specific model which we found to be appropriate for demonstrating how the proxel-based performability analysis works. The model represents a computer system which has two different processors that operate at different speeds and have different properties.

Firstly we will present experiments where the performances of the separate processors are constant and then ones where they are functions of additional parameters of the state of the model.

```
1. TP(0)=0, EW(0)=0; index=0;
2. generate proxel_list(index) = {initial_proxel};
3. for i = 1 to [t/dt]
4.     TP(i)=0; index = 1-index;
5.     for each proxel(j) in proxel_list(1-index)
6.     rest_pr = probability(proxel(j));
7.         for each scp = state_change_possible(discrete_state(proxel(j)))
8.             pr = probability for the state change given the age_intensity(proxel(j));
9.             ai = update(age_intensity(proxel), scp);
10.            if ss =succ_state(proxel(j), scp) not in proxel_list(index)
11.                generate np = new_proxel(ss, ai, pr) in proxel_list(index) with probability pr;
12.            else add probability pr to the existing one;
13.            TP(i)= TP(i) + rr(discrete_state(np))*pr + ir(scp)*pr;
14.            EW(i)= EW(i) + rr(discrete_state(np))*pr*dt + ir(scp)*pr;
15.            rest_pr = rest_pr - pr;
16.        end_for
17.        generate new_proxel(discrete_state(proxel(j)), age_intensity(proxel(j))+1, rest_pr)
               in proxel_list(index);
18.        TP(i)= TP(i) + rr(discrete_state(proxel(j)))*pr;
19.        EW(i)= EW(i) + rr(discrete_state(proxel(j)))*pr*dt;
20.    end_for
21.end_for
```

Figure 2: *Simplified algorithm of the proxel-based performability analysis*

3.1. Description of the Example Model

The state diagram of the model that we elaborate and experiment with is illustrated in Figure 3.

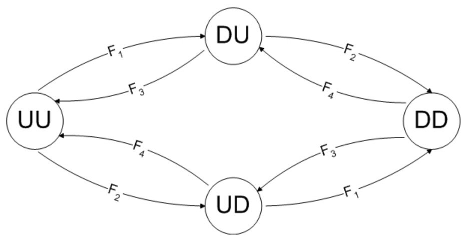


Figure 3: *State diagram of the example model*

The model has four discrete states:

- UU (both machines up),
- UD (faster machine up and slower down),
- DU (slower machine up and faster down), and
- DD (both machines down),

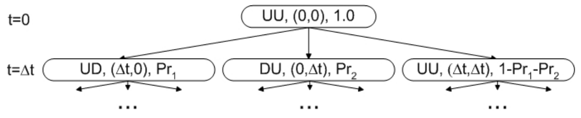
and each of them has a performance value (reward rate) associated with it. The assumption is that one of the processors is 10 times as fast as the other one. That means that we can associate performance values for the four states as follows

- $rr(UU) = 11\,WU$
- $rr(UD) = 10\,WU$
- $rr(DU) = 1\,WU$
- $rr(DD) = 0\,WU$

where the amount of work that the slower computer can accomplish in a time step Δt is the unit for measuring the amount of work done (*WU*). The failure distribution functions are the following:

- $F_1 \sim Weibull(15.0, 1.5)$
- $F_2 \sim Uniform(3.0, 6.0)$
- $F_3 \sim Deterministic(3.0)$
- $F_4 \sim Uniform(1.0, 5.0)$

In the same manner as in Figure 1, the illustration of the proxel-based performability analysis of this model is presented in Figure 4. The age intensity vector in this case has two components because that is the maximal number of concurrently active state changes. The first component maps the age intensity of the first processor and the second one of the second processor.



t	TP(t)	EW(t)
0	rr(UU)	rr(UU)
Δt	rr(UD)*Pr_1 + rr(DU)*Pr_2 + rr(UU)*(1-Pr_1-Pr_2)	TP(0) + TP(Δt)

Figure 4: *Illustration of the initial steps of the proxel-based performability analysis process for the example model*

We carried out two sets of experiments with this model, the results of which follow in the next subsection.

3.2. Results of the Performability Evaluation of the Example Model

In the first experiment session (we denote it as *Case A*), we analysed the model presented in Figure 3 having constant reward rates as given in the previous subsection. In the both sets of experiments the simulations were carried out with time step $\Delta t = 0.2$ up to simulation time $t = 60$.

The simulation results for the transient solution of the model are shown in Figure 5, where it can be observed that the model goes ultimately into a steady state, in which the discrete state where the slower processor is up and the faster down (DU) has the highest probability, the second one being the discrete state where both processors are down (DD, whose reward rate is zero). The other two discrete states have zero steady state probabilities. Transient performability evaluation for the same model is shown in Figure 6 where it can be observed that it also has a steady state behaviour (as expected) which is the product of the discrete state DU and its reward rate, as the only non-zero value.

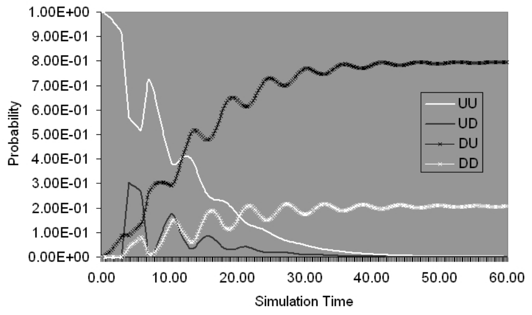


Figure 5: *Transient probabilities of the four different states*

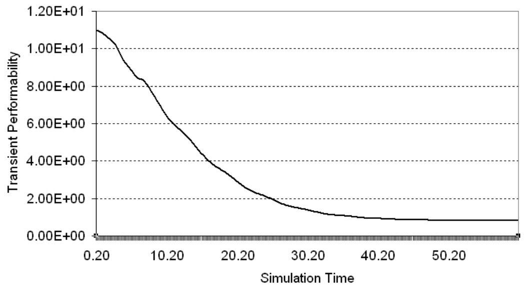


Figure 6: *Transient performability of the model (Case A)*

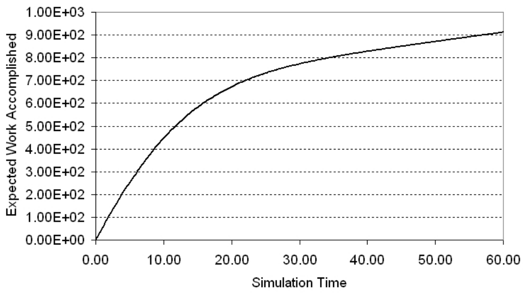


Figure 7: *Expected work accomplished of the model (Case A)*

The transient solution of the expected work accomplished is shown in Figure 7, which is the integral of the transient performability solution, as shown in Figure 6.

It is obvious how the solutions that are obtained are complete and support the process of making decisions about the system we are analysing, which is a great advantage of the proxel-based method.

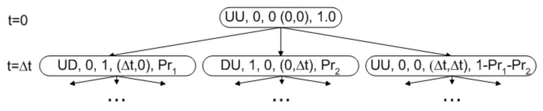
In the second set of experiments (denoted as *Case B*) we added an additional dependency to the reward rates, which now are functions of the impulse rewards that track the numbers of failures of each of the processors. This means that the performance of the system decreases with each failure that has happened, which is another realistic behaviour. The transient solution of the model in the second case is the same as in Figure 5 because we do not perform any changes on the basic model.

It is interesting that in this case (Case B) there are two additional discrete variables to the discrete states of the model which trace the numbers of failures for both processors correspondingly. The proxel-scheme for this case is shown in Figure 8. The first additional variable next to the discrete state counts the failures of the faster processor and the second one of the slower one. The consequence of this change in the model is a larger state space, and therefore a higher number of proxels generated.

The functions which describe the reward rates of the four discrete states are the following:

- $rr(UU) = 11 - (f_1 + f_2)^2 \times 0.005$
- $rr(UD) = 10 - f_1^2 \times 0.005$
- $rr(DU) = 1 - f_2^2 \times 0.005$
- $rr(DD) = 0$

where f_1 and f_2 denote the number of times each of the processors has failed. In Figure 9 the dependence of the performance on the total number of failures for the discrete state UU is shown.



t	TP(t)	EW(t)
0	rr(UU, 0, 0)	rr(UU, 0, 0)
Δt	rr(DU, 0, 1)*Pr_1 + rr(DU, 1, 0)*Pr_2 + rr(UU, 0, 0)*(1-Pr_1-Pr_2)	TP(0) + TP(Δt)

Figure 8: *Illustration of the initial steps of the proxel-based performability analysis process for the example model including the discrete supplementary variables (Case B)*

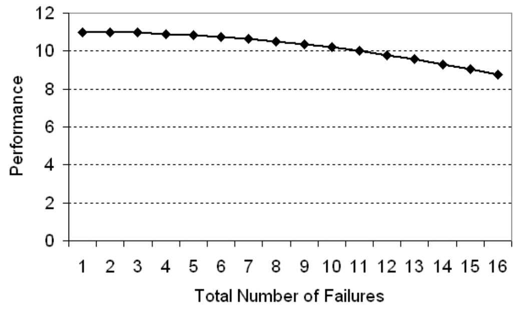


Figure 9: *Dependence of the performance with respect to the number of failures for case B*

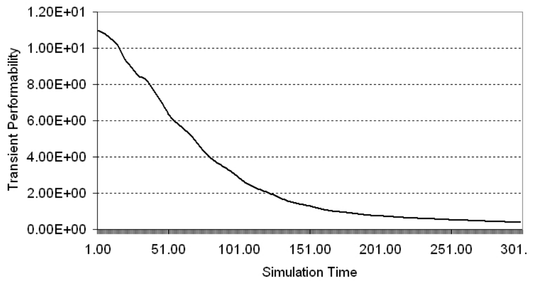


Figure 10: *Transient performability of the model (Case B)*

In Figures 10 and 11 it can be noticed that the both performability measures that we analysed have lower values this time, which is again expected, given that now they depend on the number of failures that have happened, decreasing the performance of the whole system faster than in the previous case.

As a comparison of both cases in the following table, the number of proxels that are generated and the computation times are shown.

Cases	**Case A**	**Case B**
Number of Proxels	4 119 252	10 160 034
Computation Time	ca. 15s	ca. 49s

It is obvious that in *Case B*more proxels are generated because there the state space is extended to include the numbers of failures of the both processors. This model is also not bounded, which

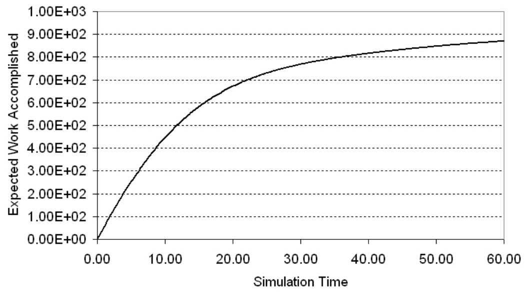


Figure 11: *Expected work accomplished of the model (Case B)*

shows another advantage of the method i.e. the ability to build the state space on-the-fly.

Another interesting situation to model is when each of the processors gets replaced by a new one if its performance falls bellow a certain threshold. Because of the limited space that will not be covered in this paper as the examples already show that such a situation is analysable. So is the case for many other classes of real-case systems, whose performability analysis is definitely possible using proxels, at least theoretically. That is a statement we can make now, as until now we have not been able to find a class of models for which it would not work.

4. SUMMARY AND OUTLOOK

Proxel-based method as a recently introduced one, until now has found an application in analysing different classes of stochastic models. Some of them, worth mentioning here are: analysing fault-trees [8] and warranty analysis [1].

In this paper another application (inspired by the previous practical experience) of the proxel-based method was introduced and formalised, which is performability modelling and analysis of mainly fault tolerant systems. Because of the flexibility of the proxel-based method and the proxel definition it is highly adaptive in analysing complex and realistic situations, which means it has very low requirements (if any) on simplifying of the model in order to analyse it. Therefore the treatment of rewards (both impulse and rate) did not represent an additional load, but instead fitted straightforward in the existing framework.

The algorithmic nature of the approach allows to model any kind of dependencies, thus bringing the model closer to the realistic system that it tries to mimic, resulting into analysis results which until now was not able to be delivered using any of the standard methods. The intuitiveness of the method is another advantage. It definitely does not require special implementation skills in order to model any system.

The accuracy of the proxel-based method is a function of the size of the time step that is being used, which is also true for the simulation time. The benefit from that is that the method is highly flexible and compromisible when fast and rough results are needed at the expense of accuracy. This can be especially useful for optimisation for example, when more simulations are needed with different parameters and the relations among their performability evaluations are important.

We are aware that the memory complexity of the current implementation is high with respect to the state space of the model. Therefore we are working on improving it by replacing the infinites support functions by discrete phases [9], which until now has given some promising results. The method itself performs very well for distribution functions with finite support because they have a limited range in which proxels are generated and stored, because proxels with zero probability are not stored. Another condition from which the method can benefit with respect to the model is having a higher number of concurrently active state changes, which makes the time that the model can spend in one discrete state shorter, making its age intensity values limited, resulting into a similar effect as the previous condition.

So, it is obvious that classes of models can be distinguished for which the proxel-based method performs well and those for which it will be extremely expensive. However, we consider one of the strongest advantages to be its (until now) unrestricted application and its ability to analyse models which are hard to be imagined as doable using other analysis methods.

What still needs to be done with respect to the proxel-based performability analysis is testing it on real models that are difficult or maybe impossible to be analysed using the standard approaches. Another point of future research is adapting the new proxel-adapted modelling framework to include rewards.

5. REFERENCES

[1] Lazarova-Molnar, S., and Horton, G. ”Proxel-Based Simulation of a Warranty Model”, European Simulation Multiconference, Magdeburg, 2004.

[2] Haverkort, B.R., Marie, R., Rubino G., and Trivedi, K., Performability Modelling: Techniques and Tools, John Wiley & Sons, Ltd, 2001.

[3] Horton, G., ”A new paradigm for the numerical simulation of stochastic Petri nets with general firing times”, Proceedings of the European Simulation Symposium 2002, Dresden, Society for Computer Simulation, 2002.

[4] Muppala, J.K., Ciardo, G., and Trivedi, K. ”Stochastic Reward Nets for Reliability Prediction”, Communications in Reliability, Maintainability and Serviceability: An International Journal published by SAE International, Vol. 1, No. 2, July 1994.

[5] Zimmermann, A., Freiheit, J., German, R., and Hommel, G. ”Petri Net Modelling and Performability Evaluation with TimeNET 3.0”, 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS’2000).

[6] German R., Performance Analysis of Communication Systems. Modeling with Non-Markovian Stochastic Petri Nets, John Wiley & Sons, Ltd, 2000.

[7] Lazarova-Molnar, S., and Horton, G. ”An Experimental Study of the Behaviour of the Proxel-Based Simulation Algorithm”, Simulation und Visualisierung 2003, SCS Verlag 2003.

[8] Lazarova-Molnar, S., and Horton, G. ”Proxel-Based Simulation for Fault Tree Analysis”, 17. Symposium Simulationstechnik (ASIM 2003), SCS European Publishing House, 2003.

[9] Isensee, C., Lazarova-Molnar, S., and Horton, G. ”Combining Proxels and Discrete Phases”, ICMSAO’05, American University of Sharjah, UAE.