Proxel-Based Simulation for Fault Tree Analysis

Sanja Lazarova-Molnar
sanja@isg.cs.uni-magdeburg.de
Institut für Simulation und Graphik, Universität Magdeburg
Universitätsplatz 2, D-39106 Magdeburg

Graham Horton
graham@isg.cs.uni-magdeburg.de
Institut für Simulation und Graphik, Universität Magdeburg
Universitätsplatz 2, D-39106 Magdeburg

Abstract

Fault trees are used to describe the failure behaviour of complex technical products and systems. We are interested in the case where the basic events can themselves be small simulation models with non-trivial time-dependent behaviour. These models can be very stiff and at the same time require a high degree of solution accuracy. We propose proxel-based simulation as an alternative to the usual Monte Carlo approach and give some arguments in its favour. Computational experiments are performed to evaluate the method's usefulness. We discuss techniques for accelerating the analysis and give an outlook for a general simulation tool.

1 Goals of the Paper

The goal of this paper is to show that fault trees can be analysed using proxel-based simulation. At the same time we want to show that the proxel-based method can be very suitable for fault-tree analysis and provide very accurate results. We will also make a comparison to the standard approaches.

2 Preliminaries

The formal description of the proxel-based method is given in [1][2], therefore because of the limitation of the space, only a short introduction of the method will be given. There will be a brief description of fault trees too.

2.1 Proxel-based Simulation

Proxels were recently introduced [1] as a new technique for analysing discrete-state stochastic models such as queueing systems or stochastic Petri nets. For this class of model, the analysis is usually carried out using discrete-event Monte Carlo simulation. By contrast, proxel-based simulation is deterministic, and works with the state-space view of the model. Normally, this approach entails constructing and solving partial differential equations [5], but Proxels yield a purely algorithmic approach to the simulation, in which differential equations are avoided completely. The goal of the new approach is to develop an easily-understood, deterministic algorithm, which, for certain classes of model, may prove to be more efficient than discrete-event simulation.

A Proxel is a basic computational unit for the simulation algorithm, which represents the probability that the simulation model has reached a specific state at a given time via a specific path. We use the term "Proxel" as an abbreviation of "probability element" by analogy to the well-known "pixel" in Computer Graphics.

The idea behind the simulation algorithm is to discretise the continuous stochastic process of the user model using a discrete time step *dt*. This yields a computational model consisting of a set of discrete states at each time point, each of which has a certain probability of occurring. The proxel simulation algorithm creates the discrete states on the fly and tracks probability as it is redistributed between these states as time progresses.

We illustrate the idea intuitively using the simple user model in Figure 1. This model consists of three discrete states OK, NR and F. Arrows indicate the state changes (activities) that are possible. The delays for each activity are described by probability distributions. We assume the model to be in state OK at time t=0.

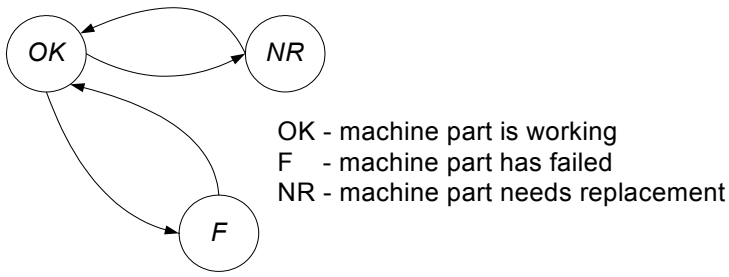

Figure 1: A simple discrete model

We now choose a discrete time step *dt*. Figure 2 shows the states of the model at times 0, *dt*, and 2**dt*. The state is composed of two parts - the discrete state of the user model and the length of time that an activity has been going on without a state change occurring. The latter is known as the age intensity; it is needed to compute the probabilities for each state at the new time step, as will be explained in the next section. For this simple model, only one age intensity variable is needed. For more general models, several will be necessary.

At time $t = 0$, the model is in state (OK, 0). At time $t = dt$, the model can have done any of three things – transition to state F or NR, or remain in state OK. The probability for each of these occurrences can be computed from the probability distributions. States

at time $t = 2dt$ and later are generated correspondingly. Proxel-based simulation is the repeated computation of the new set of proxels as simulation time progresses.

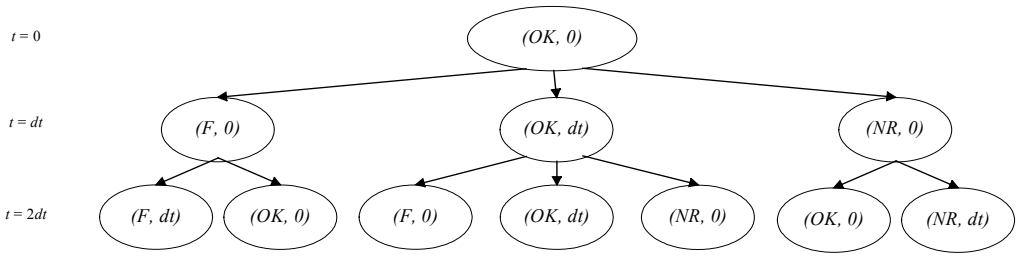

Figure 2: The first few states reached by the model

In Figure 2 it looks as if the proxel tree grows exponentially as time progresses. However, because there is only one age intensity variable and the model is quite simple, the tree reaches a maximal set of reachable states very quickly. This means that after some point in simulation time the width of the tree does not increase. Thus, proxel simulation is efficient for small models with only one age intensity variable.

2.2 Fault trees

In complicated systems failures result as a combinations of different factors (such as components failing, human errors etc.). The manner in which each of these factors affects the failure of the system as a whole can be represented in a logical function. Every factor that is not further decomposed is referred to as "*basic event*". The graphical representation of this behaviour of one system as a logical function of the basic events is identified as "*fault tree*". The logical interconnections of the basic events that lead to the top event are known as *gates*, which define the types of the relationships.

Fault tree analysis is a deductive analysis which provides a method in which the causes for a certain undesired event (top event) are determined. The fault tree in itself is a qualitative model, which can be evaluated quantitatively, which is the objective of the proxel-based fault tree analysis. This means obtaining probability for failure of the system as a function of time.

Basic events can typically be modelled using small Petri nets that have only one token and no more than five to six places. This means that only one age variable is needed which makes the proxel-based simulation a good choice for analysing the basic events.

As an example we present a failure of a system which is modelled as a fault tree that consists of four basic events i.e. four components that fail: A, B, C and D, as shown in Figure 3. The system fails if one of the components C and D fail, and both of A and B. The probability distributions of the activities are shown on the arcs in each model. The top event in this case is represented with the following logical function of the basic events:

(A and B) and (C or D)

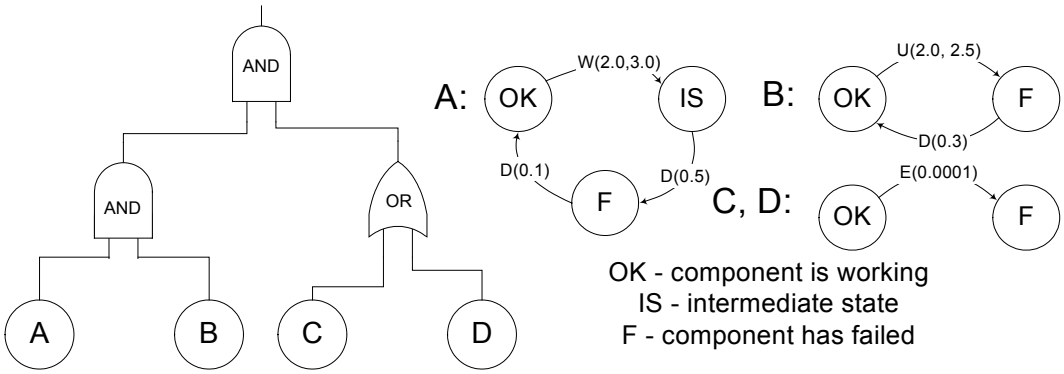

Figure 3: Example fault tree with the state diagrams of the basic events

3 Proxel-based analysis of fault trees

When each basic event is a simple failure, as in cases C and D in Figure 3, then the fault tree can be analysed by performing simple probability algebra. For non-trivial basic events, discrete-event simulation is usually used. Here, we propose the use of proxel-based simulation for reasons of efficiency and simplicity. An alternative supplementary-variable based approach for models with only one active age intensity variable can be found, for example in the tool TimeNET [6].

3.1 Description

The basic events of the fault trees can usually be modelled by very simple SPNs which contain only one token. The proxel-based simulation works very well for this type of models because it usually requires only one age variable. The proxel-based analysis also releases the typical fault tree analysis limitation of having only non-repairable systems.

The way it works is that primarily all of the basic events are separately simulated, which can be done sequentially or in parallel. This makes it possible to check the models for errors for each of the basic events. After that the logical function that describes the top event is applied to these results from the proxel-based simulation which provides the probability of the top event as a function of time. The results of the proxel-based fault tree analysis are deterministic and their accuracy can be easily controlled by the choice of the discrete time step.

3.2 Example

The fault tree that we use in order to explain how the proxel-based analysis work is the one presented in Figure 3. In Figure 4, the proxel-based simulation results of the basic events are shown. The simulation was carried out with time step $dt = 0.01$ to time 50.

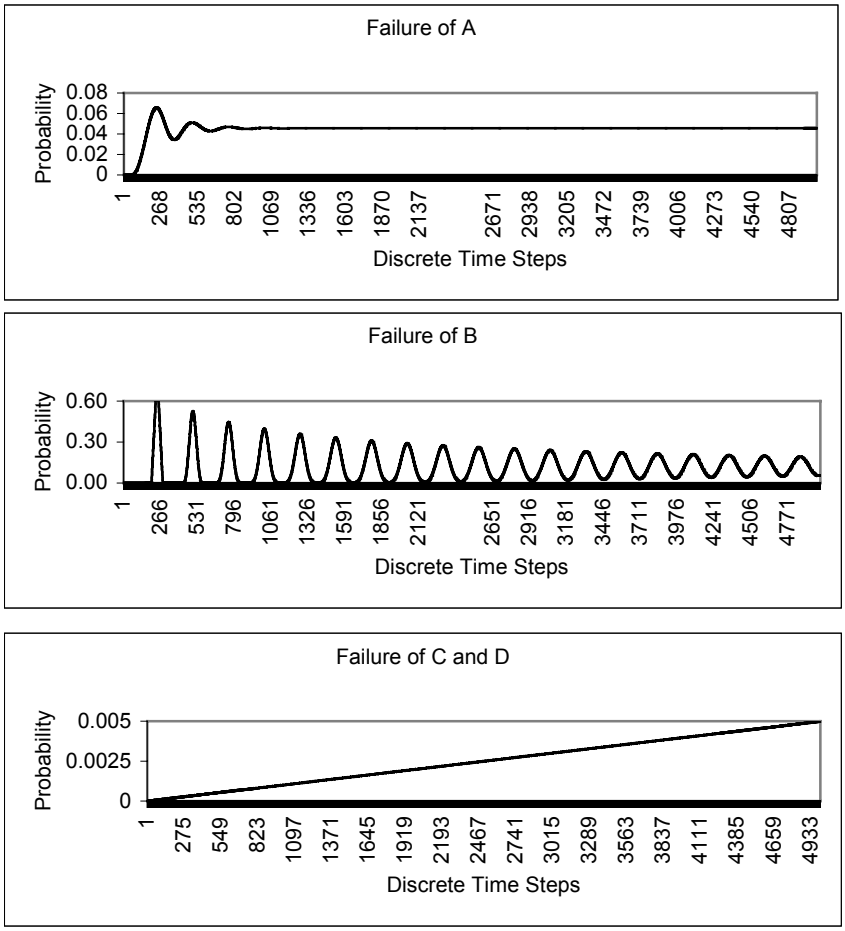

Figure 4: Results from proxel-based simulation of basic events

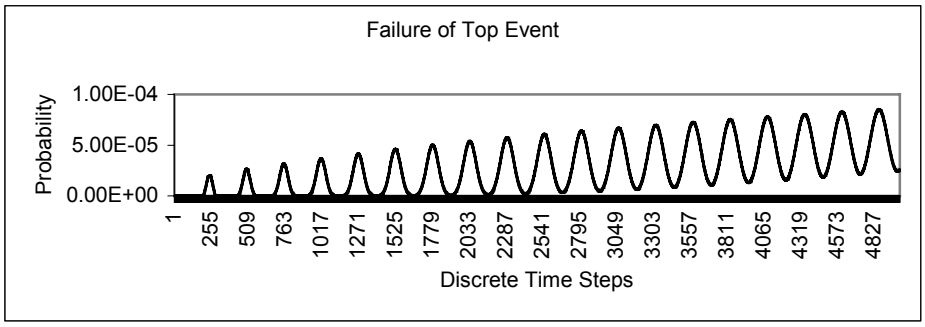

Figure 5: Results from proxel-based simulation of the top event

In Figure 5 it is evident that the proxel-based fault tree analysis provides accurate and precise results and is able to trace the change of probability on a very detailed level. It is noticeable how the probability of the top event increases and decreases because of the fact that there are repairable components in the system.

4 Conclusion and future work

The proxel-based analysis of fault-trees provides deterministic and accurate results. The simulation can be very efficient in some cases (compared to the standard approaches). Two of those cases are models that contain rare events and models that contain activities' times distributed according to functions with finite support [2][3]. In the other cases it performs quite well too because the models which describe the basic events are simple models which contain only a few states.

Development of a more general fault-tree analysis tool is planned, which will provide more general experiments to be carried out. This should also give us some insight into how the proxel-based fault tree analysis could be improved and made more efficient.

5 Literature

[1] *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.

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

[3] *Lazarova-Molnar, S., and Horton, G.*: Proxel-Based Simulation of Stochastic Petri Nets containing Immediate Transitions. On-Site Proceedings of the Satellite Workshop of ICALP 2003 in Eindhoven, Netherlands. Forschungsbericht Universität Dortmund. Dortmund 2003.

[4] *Vesely, W. E., Goldberg, F. F., Roberts, N. H., Haasl, D. F.*: Fault Tree Handbook. Washington, DC: 1981.

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

[6] *Zimmermann, A., Freiheit, J., German, R., 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), LNCS 1786, Springer-Verlag, Schaumburg, Illinois, USA, 2000.