The Proxel-Based Method: Formalisation, Analysis and Applications

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieurin
(Dr.-Ing.)

angenommen durch der Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg
von

Dipl.-Ing. M. Sc. Sanja Lazarova-Molnar

geboren am 26. Februar 1977 in Skopje, Mazedonien

Gutachter: Prof. Dr. Graham Horton
Doz. Dr. Rüdiger Hohmann
Prof. Dr. Helena Szczerbicka

Magdeburg, den 11. November 2005

Abstract

The proxel-based method is an intuitive approach to analysing discrete stochastic models, such as are described by stochastic Petri nets or queuing systems for example. The approach analyses models in a deterministic manner, avoiding the typical problems of discrete-event simulation (e.g. finding good-quality pseudo-random-number generator) and partial differential equations (difficult to set up and solve). The underlying stochastic process is a discrete-time Markov chain which is constructed on-the-fly by inspecting all possible behaviours of the model. The proxel-based simulation is shown to be very useful in analysing some classes of reliability models and fault-trees. In particular it is more efficient than the discrete-event approach applied to the same models, because the proxel-based method is less sensitive to the stiffness of the models. The goal of the thesis is to formally define this new method, study its behaviour under different circumstances, as well as show that it can be more suitable than some existing methods for certain classes of problems. Further, the thesis examines some of the application areas of the proxel-based method.

Zusammenfassung

Die proxel-basierte Methode ist ein intuitives Verfahren zur Analyse diskreter stochastischer Modelle, wie sie zum Beispiel durch stochastische Petrinetze oder Warteschlangensysteme beschrieben werden. Das Verfahren analysiert Modelle deterministisch und vermeidet daher die typischen Probleme diskreter ereignisorientierter Simulationen (z.B., hochwertige Pseudo-Zufallszahlengeneratoren zu finden) oder partieller Differentialgleichungen (schwierig aufzustellen und zu lösen). Der zugrunde liegende stochastische Prozess ist eine zeitdiskrete Markovkette, die "on-the-fly" konstruiert wird, indem man jedes mögliche Verhalten des Modells betrachtet. Es hat sich gezeigt, dass proxel-basierte Simulation sehr gut anwendbar ist, um beispielsweise stochastische Modelle für die Zuverlässigkeit oder Fehlerbäume zu analysieren. Insbesondere war sie bei diesen Modellen leistungsfähiger als die diskrete ereignisorientierte Simulation, da die proxel-basierte Methode auf die Steifheit der Modelle weniger empfindlich reagiert. Das Ziel der Dissertation ist, diese neue Methode formal zu definieren, ihr Verhalten unter unterschiedlichen Bedingungen zu studieren, sowie aufzuzeigen, für welche Kategorien von Problemen sie geeigneter ist als einige der vorhandenen Methoden. Weiterhin werden einige der Anwendungsgebiete der proxel-basierten Methode betrachtet und diskutiert.

Acknowledgments

The work on this Ph.D. dissertation would not have been such a positive experience without the support and engagement of many people. Here I would like to use this space and express my gratitude to all that helped me make it possible.

I am particularly thankful to:

- Graham Horton for his guidance and support that turned the work on my thesis in a wonderful experience with lots of possibilities to learn.
- Helena Szczerbicka and Rüdiger Hohmann for reviewing the thesis and the time and patience they committed to it.
- My parents and my sister for their unconditional love and support, as well as for believing in me and teaching me that "nothing is impossible".
- The most immediate participant in this journey and my life, my husband Žarko Molnar for always being beside me and letting me share all good and bad moments.
- All that have motivated and helped my work in one or another way.

Contents

1 Introduction **1**

1.1 Introduction . . . 1
1.1.1 Is It Stochastic or Too Complicated to Understand? . . . 2
1.1.2 Modelling and Simulation . . . 3
1.2 The Proxel-Based Method in the Big Picture . . . 4
1.3 Motivation . . . 5
1.4 Goals of the Thesis . . . 6
1.5 Organisation of the Thesis . . . 7

2 Preliminaries and Overview of Simulation Methods **9**

2.1 Preliminaries . . . 9
2.1.1 What is a Discrete Stochastic Model? . . . 9
2.1.2 The Method of Supplementary Variables . . . 12
2.1.3 Stochastic Petri Nets . . . 13
2.2 Existing methods for Analysis of Discrete Stochastic Models . . . 18
2.2.1 Discrete-Event Simulation . . . 18
2.2.2 Markov Chains . . . 19
2.2.3 Modern Approaches and Tools . . . 23
2.3 Summary . . . 28

3 The Proxel-Based Method **29**

3.1 Foundations of the Proxel-Based Method . . . 29
3.1.1 Description . . . 30
3.1.2 Definitions . . . 35
3.1.3 Basic Algorithm . . . 37
3.1.4 Characterisation of the Underlying Markov Chain . . . 39
3.2 On the Accuracy of the Method . . . 41
3.3 Supplementary Algorithms . . . 49
3.3.1 Heuristics for Determination of the Size of Δt . . . 49
3.3.2 Computation of Lifetimes of Discrete States . . . 51
3.4 Programming Aspects and Complexity . . . 53
3.5 Summary . . . 55

4 Variants and Special Topics **57**
4.1 Special Classes of Problems . . . 57
4.1.1 Unbounded Models . . . 57
4.1.2 Additional Supplementary Variables . . . 59
4.1.3 Rare-Event Models . . . 60
4.1.4 Instantaneous Rate Functions with Poles . . . 62
4.1.5 Finite Support Functions . . . 64
4.1.6 Number of Concurrently Active State Changes . . . 65
4.2 Variants of the Proxel-Based Method . . . 65
4.2.1 Stochastic Proxels . . . 65
4.2.2 Inclusion of Discrete Phases . . . 67
4.3 Summary . . . 70

5 The Proxel-Adapted Modelling Framework **71**
5.1 Motivation . . . 71
5.2 Basic Elements . . . 72
5.3 Graphical Description . . . 74
5.4 Adapted Algorithm . . . 75
5.5 Examples . . . 77
5.5.1 Queuing System . . . 77
5.5.2 Machine Model . . . 81
5.6 Experiments with the Example Models . . . 83
5.6.1 Experiments with the Queuing Model . . . 83
5.6.2 Experiments with the Machine Model . . . 85
5.7 Discussion and Conclusions . . . 87
5.8 Summary . . . 87

6 Applications **89**
6.1 Proxels in Reward Modelling and Performability Analysis . . . 89
6.1.1 Performability Modelling . . . 90
6.1.2 Performability and Proxels . . . 92
6.1.3 Experiments . . . 94
6.1.4 Conclusions . . . 100
6.2 Proxel-Based Simulation of a Warranty Model . . . 100
6.2.1 Description of the Warranty Model . . . 100
6.2.2 Experiments . . . 102
6.2.3 Conclusions . . . 105
6.3 Quantitative Evaluation of Fault Trees Using Proxels . . . 106
6.3.1 What is a Fault Tree? . . . 106
6.3.2 Quantitative Evaluation of Fault Trees and Proxels . . . 107
6.3.3 Conclusions . . . 108
6.4 Analysing Stochastic Petri Nets Using Proxels . . . 110
6.4.1 Example . . . 112
6.4.2 Conclusions . . . 116
6.5 Hybrid Models and Proxels . . . 116

6.5.1 What is a Hybrid Model? . . . 116
6.5.2 Proxel-Based Analysis of Hybrid Models . . . 117
6.5.3 Conclusions . . . 119
6.6 Proxel-Based Rare-Event Simulation . . . 120
6.6.1 Example . . . 121
6.6.2 Conclusions . . . 122
6.7 Proxels in Tuning Systems . . . 122
6.7.1 Example . . . 123
6.7.2 Conclusions . . . 125
6.8 Proxels in Teaching Simulation . . . 126
6.9 Summary . . . 127

7 Additional Experiments **129**
7.1 Experiments Concerning the Accuracy . . . 129
7.2 The Effect of Lifetimes of Discrete States on the Computational Complexity . . . 134
7.3 Models for Which Proxels Are a Bad Idea . . . 140
7.4 Summary . . . 141

8 Conclusions and Future Work **143**
8.1 Summary . . . 143
8.2 Critical Assessment of the Thesis . . . 144
8.3 Discussion of the Proxel-Based Method . . . 147
8.4 Implications for Further Research . . . 148
8.5 Conclusions or Back into the Big Picture . . . 148

A Frequently Used Probability Distributions **151**
A.1 Exponential Distribution . . . 151
A.2 Uniform Distribution . . . 151
A.3 Deterministic Distribution . . . 152
A.4 Normal Distribution . . . 152
A.5 Weibull Distribution . . . 152

B Proxel-Based Simulation Tool **153**
B.1 Description and Specifications . . . 153
B.2 Input Model Specification and Example . . . 153
B.3 Summary . . . 156

Bibliography **157**

List of Algorithms

2.1 Discrete-Event Simulation 19
3.1 Basic Proxel Algorithm (1) 38
3.2 Computing Lifetimes of Discrete States 52
5.1 Extended Proxel Algorithm (2) 76
6.1 Proxel Performability Algorithm 95
6.2 Proxel-Based Hybrid Model Analysis 118
6.3 Function ProcessDS-OFF() 119
6.4 Function ProcessDS-ON() 119

List of Figures

1.1 Illustration of a queuing system in a bank 2
1.2 Structure of the thesis 7

2.1 State transition diagram of an M/G/1 system 12
2.2 Firing of a transition 16
2.3 Example SPN of a repairable machine 16
2.4 Reachability graph of the example SPN shown in Figure 2.3 17
2.5 Examples of continuous phase-type distributions 25
2.6 Canonical form of discrete phase-type distributions 26

3.1 Comparison of simulation approaches 30
3.2 Example state diagram 33
3.3 The proxel tree of the first three time steps for the example model from Figure 3.2 34
3.4 State diagram of the cashier model 40
3.5 Example Markov chain with two states 42
3.6 Proxel-trees produced using two different discretisation time steps . . . 44
3.7 Solution values obtained using different time steps and the analytical solution for $\pi_A(\frac{1}{2\lambda})$ for $\lambda = 2$ 44
3.8 Illustration of the error produced by the basic assumption of the proxel-based method 45
3.9 Illustration of the four different integration approaches 47
3.10 Illustration of the three different integration approaches when using a large time step 48
3.11 Illustration of non-convergent solution values 50
3.12 Dependence of the computation time on the number of time steps for the two storage cases 54
3.13 Dependence of the total number of proxels generated and computation time on the size of the time step Δt 55
3.14 Dependence of the total number of proxels generated on the number of the time steps in the general case, i.e. before and after the point in time when the maximum number of states has been generated 56

4.1 Petri net model of an unlimited queue with one server 58
4.2 Proxel-based illustration of the initial part of the state space 58
4.3 Petri net representation of the machine failure model 60

4.4 CDFs of two exponential distributions with different rates 61
4.5 Example rare-event model 62
4.6 Illustration of the idea of having adaptive time steps 62
4.7 IRF of a Uniform distribution function on (0.5, 1.5) 63
4.8 Structure of the proxel tree for a model with two states, A and B, where B has a lifetime of $2.5 \times \Delta t$ 64
4.9 One path in the stochastic proxel variant 67
4.10 Inclusion of phases in a simple two state model 68
4.11 Proxel-phases state tree 69
4.12 Dependence on the total simulation and fitting time on the number of phases (Isensee et al. 2005) 70

5.1 Petri net description of the queuing system model 78
5.2 Graphical representation of the model using our proposed approach . . . 79
5.3 Petri net of the machine model 81
5.4 Graphical representation of the proxel-adapted machine model 82
5.5 The reachability graph of the queue model and its connection with the proxel-adapted model description 83
5.6 Transient probabilities of the discrete states (cases) described above . . 84
5.7 Transient probabilities of the discrete states described above 86
5.8 The mean number of failures as a function of the global simulation time 86

6.1 Simplified illustration of the initial steps in the proxel-based performability analysis 94
6.2 State diagram of the example model 96
6.3 Dependence of the performance with respect to the number of failures for case B 97
6.4 Illustration of the initial steps of the proxel-based performability analysis process for the example model including the discrete supplementary variables 97
6.5 Transient probabilities of the four different states 98
6.6 Transient performability of the model 99
6.7 Expected work accomplished of the model 99
6.8 Petri net of the warranty analysis model 101
6.9 GUI of the proxel-based special-purpose solver 103
6.10 Costs approximations using different time steps and extrapolation . . . 104
6.11 Costs approximations using different time steps and extrapolation . . . 104
6.12 Solution values for the costs regarding the different warranty strategies . 105
6.13 Example fault tree with the state-transition diagrams of the basic events 107
6.14 Results from the proxel-based simulation of the basic events 109
6.15 Results from proxel-based simulation of the top event 110
6.16 Petri net of a queuing system with one limited queue and two servers . . 112
6.17 Reachability graph of the Petri net from Figure 6.16 113
6.18 The first three levels of the markings tree based on the SPN in Figure 6.16 113

6.19 The first three levels of the proxel tree based on the SPN in Figure 6.16 114
6.20 Solution of the example SPN, obtained by proxel-based simulation . . . 115
6.21 Example hybrid model of a water tap . . . 117
6.22 Results from the proxel-based solution of the hybrid model shown in Figure 6.21 . . . 120
6.23 Example model for rare events . . . 121
6.24 Results from proxel-based simulation for discrete state DSS4 . . . 122
6.25 Example model for application of the proxel-based method in tuning systems . . . 123
6.26 Cost as a function of the threshold parameter, using different values for Δt . . . 124
6.27 Computation time for all of the tuning simulations as a function of the size of the time step . . . 125
6.28 Simple model used for teaching proxel-based simulation . . . 126
6.29 Teaching proxel-based simulation . . . 127

7.1 Example model . . . 130
7.2 Solution values for the example model from Figure 7.2 . . . 131
7.3 Comparison of solution values obtained using DES and proxel-based simulation . . . 132
7.4 Solution values obtained using DES for DS_3 at time $t = 10.0$ of the example model from Figure 7.1 . . . 133
7.5 Solution values for DS_3 at time $t = 5.0$ of the example model from Figure 7.1 . . . 134
7.6 Computation times using different integration approaches for the IRF for the example model from Figure 7.1 . . . 135
7.7 Two models with equal proxel complexities . . . 136
7.8 Solution values for the four-state model from Figure 7.7 excluding the dash-lined state changes . . . 137
7.9 Comparison of the numbers of proxels generated at each time step for the two models from Figure 7.7, excluding the dash-lined state changes . 137
7.10 State-transition diagram of the fault-tolerant machine . . . 138
7.11 The number of proxels generated at one level for the model described in Figure 5.4 in the case when the transition SC_0 has an age memory policy 139

B.1 The GUI of the proxel-based simulator . . . 154
B.2 Example model . . . 155
B.3 Plot of the solution of the example model . . . 156

List of Tables

5.1 Our modelling framework and SPNs: A parallel 74

6.1 Values for the warranty model strategy 103
6.2 Estimate for the cost using initial $\Delta t = 500$ miles 104
6.3 Estimate for the cost using initial $\Delta t = 1000$ miles 105

7.1 Richardson extrapolation for the solutions for DS_3 using time steps of 0.2, 0.1, 0.05, and 0.025 132
7.2 Development of the truncation error when using different thresholds for minimum probability 133
7.3 Computational complexity of the proxel-based simulation up to time $t = 10$ for two cases: A-including age memory state changes, and B-without age memory state changes (* The simulation was run only up to $t = 4.5$ because of the long running times) 141

List of Acronyms

CDF Cumulative distribution function

CTMC Continuous-time Markov chain

DES Discrete-event simulation

DTMC Discrete-time Markov chain

FEL Future event list

FSPN Fluid stochastic Petri net

GSPN Generalised stochastic Petri net

HSPN Hybrid stochastic Petri net

IRF Instantaneous rate function

MRGP Markov regenerative process

ODE Ordinary differential equation

PDE Partial differential equation

PDF Probability density function

SPN Stochastic Petri net

1 Introduction

1.1 Introduction

It has always been the dream of mankind to possess the ability of predicting the future for different reasons, but mainly for making our lives easier. The need to foresee things has been the main motivator for many research projects. That is also the case with simulation. One of the main goals of this research area is predicting how systems will behave in the future, based on the dependencies and features that have been discovered in them.

"What is the probability that it will rain tomorrow?" is just one of the many questions that simulation tries to answer. Computer systems with enormous computing power and huge memory capacities exist only for the purpose of computing the weather forecast. And it is an important question. To name one of the simplest examples: our holiday, for which we have been saving during the whole last year depends on the weather conditions. For the huge corporations it can be the issue that creates differences in terms of enormous sums of money, where the ability to foresee weather conditions means avoiding disasters by performing precaution-measures.

There is hardly any serious system today which is put into function without building a model first and studying its behaviour through simulation. In some systems the experiments performed on the real systems can be very expensive and potentially disastrous (e.g. nuclear plants or train systems) and simulation is the safe way to analyse their behaviour. The meaning of simulation itself justifies the need for more accurate, more efficient, and more reliable simulation approaches.

In this thesis we formalise the proxel-based method, which is designed for analysing systems which are characterised as stochastic, thereby supporting the decision-making processes regarding the their behaviour. In this chapter we provide the motivation for our research, the historical background of the area of simulation, as well as the goals of the thesis.

1.1.1 Is It Stochastic or Too Complicated to Understand?

The systems that we observe in the thesis and at which the proxel-based method is aimed are described as *stochastic*. The word "stochastic" is a synonym of "random", and an opposite of "deterministic". Therefore, *stochastic system* is a system, whose future behaviour can never be fully determined based on its current configuration. This definition captures already a great number of systems that we meet in our everyday lives.

One can argue about how random the randomness is, or whether we refer to as random for a process for which we have too little information. For instance, a classical example random process is the arrival of customers in a bank, as illustrated in Figure 1.1. Observed from aside it is random. Yet, if we knew everyone in the city and could read their minds, we can clearly predict when they want to go to the bank. If we knew what kind of cars they drove, their driving habits and the roads they took, as well as all remaining information, the sequence of arrival points in time could be precisely predicted. This is, however, a too idealistic assumption, and the information that we refer to is currently impossible to possess. Therefore, the way to go is to refer to those processes as random and search for some regularities and a way to describe them.

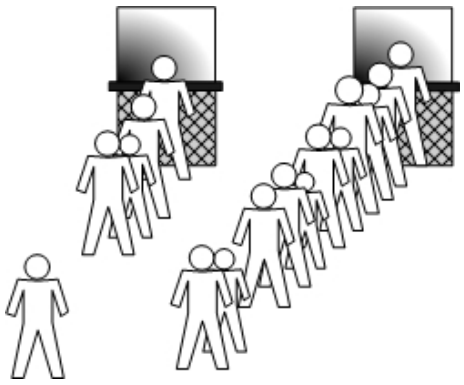


Figure 1.1: Illustration of a queuing system in a bank

Accordingly, it has been discovered that the randomness of different stochastic processes can differ, as well as characterise them. The amounts of time that cars need to get from the centre of the city to the main railway station differ. Yet, they all are within a limited time interval, and most of them are concentrated around one average length of time. Thus, to the lost driver that asks how much time it would take him/her to get from the centre of the city to the railway station, we answer "Drive straight, and in about 5 minutes you are there.". Therefore, the randomness of the travelling times from one point to another has some structure. So is the case with the inter-arrival times of visits to a restaurant, customers' inter-arrival times in a bank, or the inter-arrival times of cars passing by. The characterisation of the randomness is expressed by *distribution functions*, which are functions that describe how the probability is distributed among the set of possible values. In order to reveal the structure

of the randomness of one random process i.e. its distribution function, a large set of sample data needs to be collected and examined.

After a careful inspection of the collected set of data of interest (e.g. inter-arrival times of visits to a restaurant), the distribution functions can be guessed, and "approved" through many statistical tests (e.g. Chi-square tests), which is a process known as *input data modelling*. The distribution functions obtained through that process are then the prerequisite for designing of the model of the system whose behaviour we want to analyse.

1.1.2 Modelling and Simulation

In order to analyse a system, first its model needs to be built. "Model" is a very common word with various meanings, depending on the context. Throughout the thesis we refer to it as a description of a system, which can be used for studying the behaviour of the system by computers.

Usually, when one talks about simulation, he/she means "discrete-event simulation" (DES), which is the most straightforward way of simulating models. DES works by imitating a system's behaviour in a fast-forward fashion. For this purpose DES uses the distribution functions chosen in the *input data modelling* process and employs them for reproducing the random activities in the system being simulated. During the simulation runs, variables that track the relevant parameters are active, and based on their values, conclusions can be made about the behaviour of the system that is being simulated.

Let us again observe the classical example of a queue in front of a bank counter. Once the randomness of the inter-arrival times and the length of the service times have been described in form of distribution functions, that information is used for computer generation of "random" numbers which comply with the functions. The queue length one hour after the counter has opened is the requested parameter. For obtaining an estimate of the requested parameter, many simulation runs are carried out (e.g. 1000), where each simulates one possible system's behaviour during the first hour. At the end of each of those runs the queue-length is captured and stored in the tracking variable. At the end of all runs the average queue length is calculated. The more simulation runs are carried out, the more accurate the estimate for the real average is. The results are then delivered in a form of an interval, referred to as a *confidence interval*, which states the range of values that has a high probability of containing the parameter being estimated. Confidence intervals get narrower with the increase of the number of simulation runs. More on the topic of discrete-event simulation is presented in Section 2.2.1.

Discrete-event simulation is, however, just one of the many simulation approaches, which at the same time is the most straightforward one because it works in the most intuitive way by copying the behaviour of the system of interest, thereby using the advantage of computers to perform the imitation faster than the real execution times.

The speed results from the fact that the DES reproduces behaviour of models by only observing the points in time when the configuration of the system changes. With respect to intuitiveness, the proxel-based method can be compared to discrete-event simulation, except that it follows all possible behaviours of one system within one run, weighting each of the sample paths by distributing the probability accordingly to the distribution functions of the random processes that cause the state changes.

1.2 The Proxel-Based Method in the Big Picture

The area of simulation has a long history, and the methods that exist for simulation and analysis of stochastic systems can be generally divided into two classes: *experimental* and *numerical* methods.

In the case of experimental methods, the analysis is performed implicitly by observing the results obtained from many experiment runs. The most popular experimental simulation method is discrete-event simulation, which as already explained works by reproducing the behaviour of the systems. The reproducing of systems' behaviours is the reason why DES has to rely on random number generators which sample the random activities that are part of the system that is being analysed. The generation of random numbers and its use in analysing stochastic models is a reason why we refer to this method as *stochastic*. Once the model is built (i.e. programmed), the computer performs as many sample runs from the model as necessary to make conclusions about the model's behaviour. Therefore, the analysis in the case of DES is indirectly conducted, based on the observation of the sample runs. The biggest advantages of discrete-event simulation are its intuitiveness and its ability of simulating large-scale models for which deterministic solutions are intractable.

Numerical methods on the other hand, are designed for *analysing* stochastic models without incorporating any random behaviour. As a consequence, the simulation results that they deliver are always identical for the same model parameters and with a more controllable accuracy than DES. Numerical methods work by describing the flow of probability within one system, by mathematical models, usually using differential equations, which are then solved using different numerical methods. The differential equations are usually very complicated to set up and solve, and therefore not recommendable for large-scale models. One of the most popular numerical methods are Markov chains, which are briefly described in Section 2.2.2.

The proxel-based method contains features from both classes of methods for analysing stochastic models, and therefore can be seen as a sort of a hybrid of both classes. On one hand it does not employ random numbers, compared to the discrete-event simulation. However, compared to the existing numerical methods, the proxel-based method does not set up a system of differential equations, but dynamically describes and follows the flow of probability among the states of the model, in a very intuitive manner. This allows the method to be arbitrarily and to provide more accurate analysis of classes of models that until now were complicated and expensive to be analysed in a

deterministic manner. The only feasible choice for some of those models was discrete-event simulation, which can also become very expensive when high accuracy is needed. The proxel-based method represents now another option.

The method, as is the problem with other deterministic approaches, suffers from the problem of state-space explosion, and is therefore not recommended for solving large-scale models. We believe that in spite of this disadvantage, the method will be very useful for some classes of models, as presented further in the thesis.

1.3 Motivation

If a general stochastic model is to be analysed without imposing any restrictions, the choice of an analysis method narrows down to discrete-event simulation, as the only method that can handle any complex situation. However, DES has a stochastic nature and a few known drawbacks, and it can become very expensive for certain types of models. One such case are the rare-event models, as explained in Section 6.6. The proxel-based method is a deterministic alternative with comparable applicability as DES, that can be very promising, especially for the classes of models that are known as problematic for DES.

The proxel-based method was initially introduced by Graham Horton in (Horton 2002), as an alternative way of simulating discrete stochastic models. The method itself represents a progress in the direction of finding a widely applicable simulation method, which means a method that would be able to simulate a wide class of systems that exist in the real world. Because the approach is based on the description of the model and builds the solution algorithm directly from it, the proxel-based method is very promising and a number of applications was easily foreseeable. The method, just like discrete-event simulation, is able to simulate anything that can be brought in a certain form (i.e. model, program) to the computer.

The determinateness gives another dimension of the proxel-based method, i.e. unlike discrete-event simulation, the results do not depend on a quality of a random-number generator, and they are not in a form of a confidence interval. Instead, the results are in a form of a discretised function which approximates the transient probability. In the bank-queue example this would mean obtaining a series of n functions which would describe probabilities for having 0... *n-1* customers in the queue, for every time step, up to the maximum simulation time.

The method also offers solutions to classes of problems which are not trivial and hard to solve using the existing approaches, as pointed in Section 4.1. For instance, the fact that the method builds the state-space of a model dynamically or "on-the-fly" while simulating it, indicates that it can be applied for analysing unbounded models. Unbounded models are hard to imagine being analysed by using any of the existing deterministic approaches, as most of them require an a priori generation of the state-space. The method can also be easily applied to models which have state-dependent

distribution functions, i.e. functions which change depending on the discrete state and the time spent in it.

All of these speculations needed to be studied, experimented and tested for our speculations to be confirmed. Because of the novelty of the method, it required an exhaustive research to study its behaviour, as well as formalise it, in order to make it complete and self-contained so that the method can be offered to simulation practitioners as another alternative for analysing stochastic systems.

All of the above said represents a basic motivation for the thesis, and the basis for setting our goals as explained in the next section.

1.4 Goals of the Thesis

The motivation behind the thesis indirectly formed the goals of it. The main goal of the thesis is to develop and bring the proxel-based method to a level of an established simulation approach that can be offered to and be applied by simulation practitioners. This means a method for which it will be easy to spot when it would be efficient to be applied and when it would deliver acceptably accurate results, as well as when other alternative approaches would be the better choice.

The main goal resulted into the following sub-goals or tasks of the thesis:

- formalising the proxel-based method,
- analysing and studying its behaviour, and
- examining application areas.

Therefore, the more elaborate goal of this thesis is to formalise the proxel-based method, study its behaviour, thereby distinguish the classes of models for which it performs well and would be the natural choice for their analysis for accuracy or efficiency reasons, and finally discuss and examine its application areas.

Formalising the proxel-based method means providing definitions for all elements involved in the method, as well as provide an algorithm which defines the way the proxel-based method works. Additionally, we design a modelling framework which responds to the input requirements of the proxel-based method, and therefore contributes to the method's completeness.

Studying the behaviour of the proxel-based method involves many issues, among which the most important are analysing the accuracy of the method, which can be used to estimate the error that it makes, and analysing the computational complexity as a function of the size of the time step and the characteristics of the model being analysed. It is also essential to distinguish the classes of models for which the proxel-based method is especially recommended for different reasons.

Finally, the application areas demonstrate the usefulness of the newly introduced method in some already established and popular problem-domains in the area of simulation and modelling, and build foundations for further research regarding each of these areas.

In conclusion, there are two supplementary goals of the thesis. The first one is to communicate a clear message that the proxel-based method is an approach that can be successfully applied to certain classes of problems. The second supplementary goal of the thesis is a more subtle one, and that is to provide a useful textbook which will aid the further development of the proxel-based method and will serve as a generator for ideas with that respect.

1.5 Organisation of the Thesis

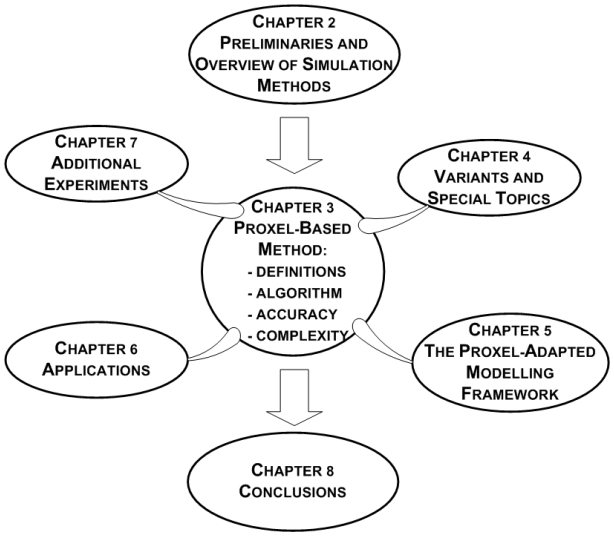


Figure 1.2: Structure of the thesis

The thesis is organised as follows. Chapter 2 introduces the preliminaries, which should provide the necessary background required for understanding the remaining chapters. Further in the same chapter, a brief overview of the existing approaches aimed towards simulation of discrete stochastic models is presented, which discusses both the traditional and the modern ones.

Chapter 3 proceeds with the introduction of the proxel-based method, as well as its formalisation and definitions. It also discusses one of the most important aspects, the method's accuracy and ways to improve it.

Once the proxel-based method is described and defined, Chapter 4 discusses the variations of the method that we have implemented and experimented with while searching for improvements to the basic approach. In the same chapter, the special classes of problems for which the proxel-based method is recommended are discussed.

In order to bring the proxel-based method one step further and turn it into a self-consistent method, we designed a modelling framework, whose goal is to provide such a description of the models that it would exploit all or most of the beneficial and unique properties of the proxel-based method. The specification of this modelling framework is the main topic of Chapter 5.

The following Chapter 6 discusses the numerous applications of the proxel-based method, some of which were initiated as a result of our cooperation with Daimler-Chrysler for their reliability models.

Chapter 7 presents and discusses some experiments regarding the accuracy and the computational complexity of the proxel-based method, which could not be placed elsewhere because they contain elements presented in more different chapters.

Finally, in Chapter 8 we conclude the thesis by discussing and summarising our contributions, as well as providing directions for future work.

Throughout the thesis, along with each chapter and problem area that we discuss, we present examples and results from experiments that illustrate the subjects that are discussed.

2 Preliminaries and Overview of Simulation Methods

In this chapter we introduce the reader to the basic terms and paradigms used throughout the thesis. Furthermore, we present the preliminaries necessary for understanding the issues treated in the thesis, and especially for understanding the basic principles of the proxel-based method. Also an overview of existing simulation methods is provided, which considers both the traditional, as well as the more significant modern approaches, discussing their advantages and drawbacks.

2.1 Preliminaries

This section addresses the issues, which we classified as preliminary or necessary for understanding the basic principles of the proxel-based method. Throughout the thesis we use the term *model* to refer to a description of a system in such a way that it can be used for making conclusions about its behaviour. The process of studying the behaviour of a system via its model we denote as *simulation*. We begin by describing what a discrete stochastic model is, a class of models on which the proxel-based method focuses.

2.1.1 What is a Discrete Stochastic Model?

Many real-life processes can be described by means of stochastic models, which basically signifies that the events that cause the state changes occur randomly. The randomness, however, conforms to some rules and possesses a structure, which makes it possible to describe and distinguish different types of randomness. The random processes that exist in the stochastic models are described by random variables, which are characterised by distribution functions. These distribution functions define the structure of the randomness for each of the random variables.

A stochastic model is therefore described by means of a stochastic process (Kulkarni 1995; Cox and Miller 1965), which again is defined as a family of random variables

$\{X(t) : t \in T\}$ that can take on values from the state space S, indexed by a parameter $t \in T$. The state space S is the set of configurations reachable by a model. The index can represent any physical quantity, although most often it is viewed as a time parameter. Depending on the character of the state and the parameter space, there are four classes of stochastic processes, derived from the following four combinations (Siegle 2002):

- discrete state space / discrete parameter space,
- discrete state space / continuous parameter space,
- continuous state space / discrete parameter space, and
- continuous state space / continuous parameter space.

A classical and one of the simplest examples of stochastic models is a queuing system. For instance, a bank counter with one server and one queue, where customers arrive at random, and so are the lengths of service times. This stochastic model has a discrete state space, enumerated by the number of people in the queue and the occupancy of the server, and discrete parameter space (i.e. time), because of the fact that state changes happen at discrete points in time. This is the class of models that proxel-based simulation focuses at, i.e. *discrete state space/discrete time*, which we refer to as *discrete stochastic models*.

There are two types of random variables, according to the number of values they can take on: discrete (countable number of values) and continuous (uncountably infinite number of possible values). We are interested in the latter one because the processes that we observe are defined on the time domain, which is continuous and infinite. Each continuous random variable is defined by a distribution function, which is the function that characterises the structure of the variable's randomness. It can be either a *cumulative distribution function* (CDF) or a *probability density function* (PDF). The CDF $F_X()$ is defined as follows

$$F_X(x) = Pr\{X \leq x\} \tag{2.1}$$

and expresses the probability (denoted by Pr) of receiving a draw less than or equal to x. The PDF $f_X()$, on the other hand, is the derivative of the cumulative distribution function, formally expressed as

$$f_X(x) = \frac{d}{dx} F_X(x). \tag{2.2}$$

Another function that characterises a random variable is the *instantaneous rate function* (IRF), sometimes referred to as *hazard rate function* or *failure rate function* in the literature. The easiest way to describe it leads to one of its other names i.e. "failure rate function". If X is a random variable that describes the failure times of one machine, then the IRF of X at time τ ($\mu_X(\tau)$) provides the rate with which a failure might

happen within a time interval $(\tau, \tau+\Delta\tau)$, where $\Delta\tau \to 0$, given that there has not been a failure for a time period of τ (Trivedi 2002). In general, the IRF expresses the rate with which an event might happen in a time interval $(\tau, \tau+\Delta\tau)$, where $\Delta\tau \to 0$, given that it has been pending for a time of τ. An event is *pending* if all of the preconditions for its happening are fulfilled and it has not happened yet. The IRF $\mu()$ is calculated as follows:

$$\mu(\tau) = \frac{f(\tau)}{1 - F(\tau)}. \tag{2.3}$$

The function $S(\tau) = 1 - F(\tau)$ is usually referred to as a *survival function* and expresses the probability that an event has not happened up to time $t = \tau$ i.e. $S_X(x) = Pr(X > x)$. The IRF is the function that we observe and use in the proxel-based approach, as explained further in Chapter 3.

We distinguish two classes of probability distributions: *finite support* and *infinite support* distributions, depending on the support of the IRF of the corresponding random variable. This classification is very important with respect to the efficiency of the proxel-based simulation, as described further in the thesis. Examples for finite support distributions are the Deterministic and the Uniform distributions and the Exponential one is an example for an infinite support distribution.

One interesting and characteristic class of stochastic processes is the class of Markov processes, which is discussed more in detail in Section 2.2.2. For that class there are already established and verified analytic solution methods because of its unique property, the "memorylessness", which means that the future behaviour of a model depends only on its current state, independently of the model's history. From a solution point of view, the "memorylessness" is viewed as an advantage. It, however, in many cases is viewed as a limitation for the reason that there are many processes that cannot be characterised as memoryless (e.g. wear-outs or processes with known deterministic duration).

When one speaks of the analysis of discrete stochastic models, there are two types of solutions that are meant by it: *transient* and *steady-state* solution. While the transient solution describes the behaviour of a model as a function of time, usually in form of state probabilities, the steady-state solution defines the model's long run behaviour, i.e. when in equilibrium. The definition of a solution, in general, depends on the model being analysed and the points of interest, which determine the parameters whose estimates are sought for.

Throughout the thesis most of the discrete stochastic models are described by means of state-transition diagrams that allow for *generally* distributed state changes. "Generally" means that there is no limitation on the type of distribution functions associated with the state changes.

2.1.2 The Method of Supplementary Variables

The method of supplementary variables was initially introduced by Cox in (Cox 1955a) for the purpose of analysing queuing systems, in which area it has a long history (Stidham 2002; Takagi 1991). The basic idea of the method is analysing the behaviour of non-Markovian models by contributing additional continuous variables to the states of the model which track the elapsed times of non-exponentially distributed events that have not happened yet (i.e. pending) (Cox and Miller 1965). The information that these variables carry is sufficient for calculating the probabilities for the possible future behaviours of the model, turning the model into a Markovian one. This is the main thought that exists also behind the proxel-based method.

Based on this hybrid description of the stochastic process, state partial differential equations are derived, which describe the dynamics of the process. The PDEs are further usually solved by application of Laplace transforms and z-transforms (German 2000b), which is a typical approach used in queuing theory. Numerical solution in time domain is only possible in case of finite state spaces, as stated in (German et al. 1995b), in which case there are two main restrictions:

- no generally distributed state changes are active in the beginning (caused by Dirac impulse in the initial conditions), and
- all general distributions have finite support.

In the case of a M/G/1[1] queuing system, German in (German 2000b) defines the underlying stochastic process as

$$\{(N(t), X(t)), t \geq 0\}, \tag{2.4}$$

where $N(t)$ is the number of customers in the system at time t, and $X(t)$ is a continuous supplementary variable, which tracks the elapsed service time at time t, and is referred to as *age variable*. The state transition diagram of the model is shown in Figure 2.1, where λ is the arrival rate of customers, and $\mu(x)$ is the instantaneous rate function of the non-exponentially distributed random variable that describes the service times.

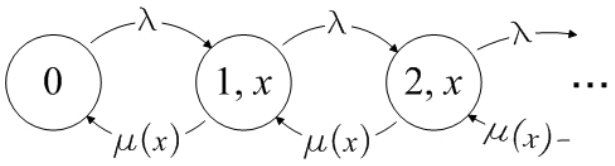


Figure 2.1: State transition diagram of an M/G/1 system

The following quantities are then needed in order to describe the dynamics of the underlying stochastic process:

[1] M/G/1 denotes a queuing system which consists of one queue and one server, where the arrivals are exponentially distributed, and the service times follow a general distribution function.

- *state probability* $\pi_n(t)$, denotes the probability of having n customers in the system at time t

$$\pi_n(t) = Pr\{N(t) = n\}, \tag{2.5}$$

- *age distribution function* $\Pi_n(t, x)$, denotes the joint probability of having n customers in the systems at time t and an elapsed service time of less than or equal to x

$$\Pi_n(t, x) = Pr\{N(t) = n, X(t) \leq x\}. \tag{2.6}$$

- *age density function* $\pi_n(t, x)$, which is derived from the probability that n customers are in the system and the elapsed service time is $(x, x + \Delta x)$, such that $\pi_n(t, x)\Delta x$ can be interpreted as the probability that there are n customers in the system and the elapsed service time is in a neighbourhood of x, i.e.

$$\pi_n(t, x) = \lim_{\Delta x \to 0} \frac{Pr\{N(t) = n, x < X(t) \leq x + \Delta x\}}{\Delta x} = \frac{\delta}{\delta x}\Pi_n(t, x), \text{ and} \tag{2.7}$$

- *state frequency* $\varphi_n(t)$, which is defined as the rate of service completions in state n at time t, and expressed as

$$\varphi_n(t) = \int_0^{x_{max}} \pi_n(t, x)\mu(x)dx. \tag{2.8}$$

After the definitions of the set of necessary quantities, the system of transient state equations is derived, which for this case consists of one ODE, two PDEs, one initial condition, two boundary conditions, and two integrals. The steady-state equations can be derived based on that. The analysis of the equations results in complicated z-transforms and Laplace transforms, which in this case can be manually solved. The analysis, however, results in a non-trivial problem when increasing the complexity of the models (German 1998).

The proxel-based method can be seen as an algorithmic implementation of the method of supplementary variables, avoiding completely the process of setting up and solving differential equations.

2.1.3 Stochastic Petri Nets

Stochastic Petri nets (SPNs) are a powerful high-level formalism, used mainly for describing and analysing discrete stochastic models, first proposed in their basic form by Carl Adam Petri in 1962 (Petri 1962; Reisig 1985; Haas 2002). They are a visual tool which provides a natural and intuitive representation of the user's conceptual model. In many cases it is a discrete-event system, meaning that the changes in the

system occur at certain points in time as a consequence of the completion of certain activities in the system. Activities in SPNs are associated with transitions, which can either fire instantly, as soon as certain conditions become true, as is the case with immediate transitions, or can be associated with probability distribution functions which determine the firing delays, as with timed transitions. Besides by the transitions, an SPN is defined by a finite number of places, a finite number of arcs and an initial state referred to as the initial marking of the Petri net.

Stochastic Petri nets are a formalism that has gone through many changes and has many variations (Puliafito et al. 1997), each of them introduced to overcome the limitations of its predecessor with respect to the classes of models they can describe. In this section we describe the class of Petri nets that is most relevant to the thesis and the models to be analysed, which is a non-Markovian class of SPNs, i.e. allows for generally (non-exponentially) distributed activities. The proxel-based method is, however, not limited to models described by means of stochastic Petri nets. Models described using this class of SPNs are used in Chapter 5 for describing our proxel-adapted modelling framework.

Formally, the class of SPNs that is treated employed in this thesis can be described in the following way:

$$SPN = (P, T, A, G, m_0), \tag{2.9}$$

where:

- $P = \{P_1, P_2, \ldots, P_n\}$ is the set of places, drawn as circles,
- $T = \{T_1, T_2, \ldots, T_m\}$ is the set of timed transitions along with their distribution functions, drawn as bars
- $A = A^I \cup A^O \cup A^H$ is the set of arcs, where A^O is the set of output arcs, A^I is the set of input arcs and A^H is the set of inhibitor arcs and each of the arcs has a multiplicity assigned to it,
- $G = \{g_1, g_2, \ldots, g_r\}$ is the set of guard functions along with the transitions they are associated to, and
- m_0 is the initial marking of the Petri net.

Each transition is represented as $T_i = (F, mp)$, where

$$mp \in \{enabling, age, immediate\}$$

is the type of memory policy if it is a timed transition or *"immediate"* if the corresponding transition is an immediate one. F is the cumulative distribution function associated with the transition if the transition is a timed one. Immediate transitions have a constant value instead of a distribution function assigned to them, which is used for computing the probability of firing of an immediate transition if more than one are enabled at once. The memory policy of a transition determines the behaviour of the transition's enabling time when it becomes disabled. It can either have age or

enabling memory policy , which specifies whether the time one timed transition was enabled until another fired will be remembered, which is the case with age memory policy, or reset, which is the case with enabling memory policy.

The sets of arcs are defined as

$$A^O = \{a_1^o, a_2^o, \ldots, a_k^o\}, A^I = \left\{a_1^i, a_2^i, \ldots, a_j^i\right\}, \text{and } A^H = \left\{a_1^h, a_2^h, \ldots, a_i^h\right\},$$

where $A^H, A^O \subseteq P \times T \times N, A^I \subseteq T \times P \times N$.

Input arc is an arc that unidirectionally connects a place to a transition. *Output arc*, on the other hand, connects a transition to a place. *Inhibitor arcs* connect places to transitions and block the transitions when the number of tokens in the place is equal to or greater than the multiplicity of the arc. We denote by the term *incoming/outgoing place* places that are connected to the *input/output arcs* correspondingly.

We denote by $M = \{m_0, m_1, m_2, \ldots\}$ the set of all reachable markings of the Petri net and is referred to as a *reachability set*. Each marking is a vector made up of the number of tokens in each place in the Petri net, $m_i = (|P_1|, |P_2|, \ldots, |Pn|)$. We distinguish two different kinds of markings, *vanishing*, if at least one immediate transition is enabled, and *tangible*, if no immediate transitions are enabled. The set of all reachable markings is the discrete state space of the Petri net. The changes from one marking to another are consequences of the firing of enabled transitions which move (i.e. destroy and create) tokens, creating the dynamics in the Petri net. This makes the firing of a transition analogous to an event in a discrete-event system. The markings of a Petri net, viewed as nodes, and the possibilities of movement from one to another, viewed as arcs, form the *reachability graph* of the Petri net. We denote the subset of the reachability graph that contains only the tangible markings by the term *reduced reachability graph*, which is basically equivalent to the *state-transition diagram* that we use throughout the thesis as a formalism for describing stochastic models, besides Petri nets.

A transition T_i is said to be *enabled* in a marking m if all of the following three conditions hold:

1. the number of tokens in each of the incoming places of T_i is greater than or equal to the multiplicity of the corresponding input arc,
2. the number of tokens in each of the places connected via inhibitor arcs to T_i is less than the multiplicity of the corresponding inhibitor arc, and
3. each of the guard functions associated with T_i evaluates to *false*.

In Figure 2.2 an example of a firing of a transition is shown. Based on the conditions for enabling a transition, T_1 is enabled, which means it is allowed to fire. That basically means that in each incoming place that many tokens are destroyed as the multiplicity of the corresponding input arc is, and in the outgoing places that many tokens are created as the multiplicities of the corresponding output arcs. After the firing, the transition T_1 is not enabled any more, as the first condition does not hold. In that case, we refer to the transition as *disabled*.

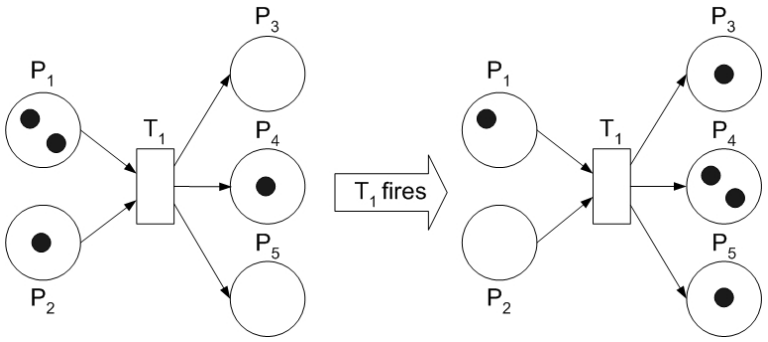


Figure 2.2: Firing of a transition

The effects of firing of all transitions in a Petri net are described by a so-called *incidence matrix*. The incidence matrix $P \times T \rightarrow Z$ is the matrix whose rows correspond to places and whose columns correspond to transitions. Column i denotes how the firing of T_i affects the marking of the net, i.e. how many tokens are removed and created into each place.

Following there is an example Petri net for demonstrating the described concepts.

Example Petri Net

Figure 2.3 illustrates one example Petri net for the purpose of demonstrating how the Petri net formalism is used for modelling stochastic systems.

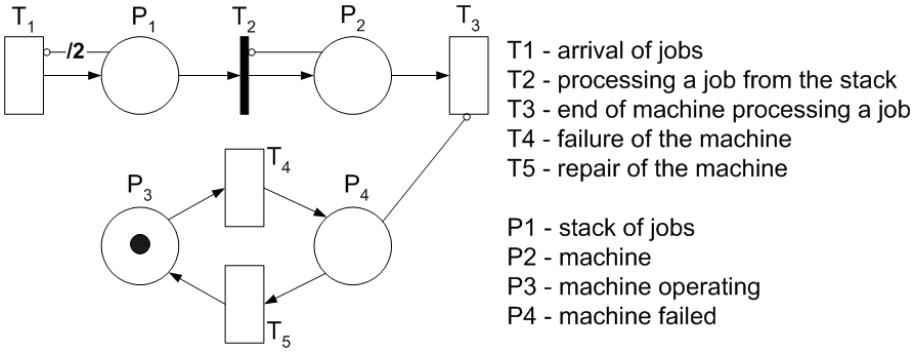


Figure 2.3: Example SPN of a repairable machine

The example is a model which is used further in the thesis for experimenting purposes and it illustrates a system of a repairable machine which processes jobs that queue in a stack with a capacity of two jobs. The meanings of each of the places and the transitions in shown in Figure 2.3 on the right-hand side. The firing of the transition T_1 represents an arrival of a job on the stack, which is represented by the place P_1. The inhibitor arc from P_1 to T_1 with a multiplicity of two shows the capacity of the stack. Transition T_2 fires only when the machine is free, i.e. does not process a job,

implemented by the inhibitor arc from P_2 to T_2. Places P_3 and P_4 model the two states of the machine, operating and failed, correspondingly. When the machine is failed, it cannot process jobs, therefore the transition T_3 is inhibited in that case. It might be expected, depending on the nature of the machine, that the transition T_3 has an age memory policy. This means that in case of a failure, the processing of a job does not restart, but resumes from the interruption point.

The initial marking of the Petri net is $m_0 = (0, 0, 1, 0)$, meaning the system is empty and the machine is in an operating state. The reachability set is the following:

$$\begin{aligned} M = \{&(0,0,1,0),(1,0,1,0),(0,0,0,1),(0,1,1,0),(1,0,0,1),\\ &(1,1,1,0),(0,1,0,1),(2,0,1,0),(2,1,1,0),(1,1,0,1),(2,1,0,1)\}. \end{aligned}$$

The reachability graph is shown in Figure 2.4. Because of the enabled immediate transition (T_2, drawn with thicker lines) in markings (1,0,1,0), (1,0,0,1), and (2,0,1,0), they are the vanishing ones in this Petri net and thereby encircled in dash-lined grey-coloured boxes. They need to be removed in order to construct the reduced reachability graph, which consists exclusively of tangible markings.

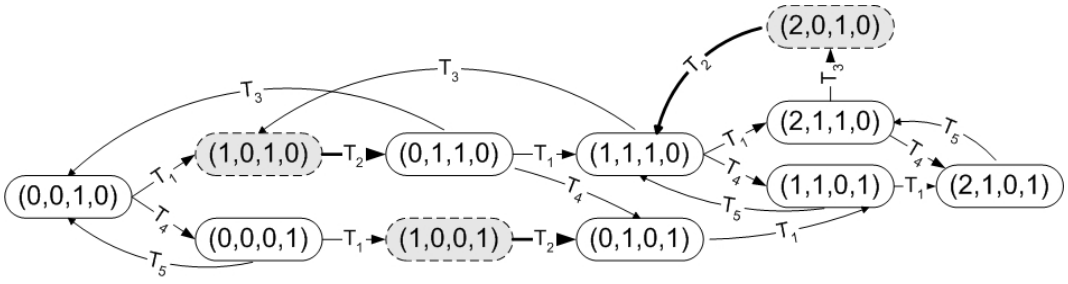


Figure 2.4: Reachability graph of the example SPN shown in Figure 2.3

The incidence matrix for the Petri net is:

$$\begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{ccccc} T_1 & T2 & T3 & T4 & T5 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 \end{pmatrix} \end{array}.$$

Further, in Section 2.2 we review the existing methods, which can be employed for analysing discrete stochastic models such as the one presented here in this section.

2.2 Existing methods for Analysis of Discrete Stochastic Models

In this section we provide a brief overview of the existing methods which are used for simulation and analysis of generally distributed (non-Markovian) discrete stochastic models. First we present the traditional methods: Discrete-Event Simulation and Markov Chains, after which we look into the more recent approaches to that problem area.

2.2.1 Discrete-Event Simulation

Discrete-event simulation (Law and Kelton 1999) is the most common way of simulating discrete models. It is at the same time the most straight forward and intuitive approach. The approach is based on a direct mimicking of the system's behaviour (simulating it), accomplished by describing every stochastic event by a random variable, and sampling values from it for every corresponding activity. Once the behaviour of the system is modelled in satisfactory way i.e. validated, it is then executed (i.e. replicated) many times, each time using independent random numbers (L'Ecuyer 1990). Finally the results of the relevant parameters (e.g. length of a queue) are delivered in form of confidence intervals, which provide the ranges in which the values of the parameters are with certain probabilities.

Discrete-event simulation works by storing a list of primary events, which is referred to as *future event list* (FEL) (Banks et al. 1998). Primary events are those that are triggered by time. The events that depend on some other conditions being fulfilled are referred to as secondary or conditional events. Algorithm 2.1 illustrates how discrete-event simulation works. The approach works by traversing the FEL and removing primary events from it as soon as they are processed (line 2). Processing a primary event means updating the state variables, which are the minimal and complete set of variables that describes the system at every point in time (line 4), checking if any new events are scheduled and inserting them in the FEL (line 5), and checking if any secondary events are triggered by the change of the state variables (line 6). If there are any secondary events that become enabled, they are processed immediately in the same way as the primary ones (lines 7 and 8).

As already stated, results from performing discrete-event simulation are delivered in form of confidence intervals for the wanted parameters, that is, the simulation run is carried out many times, after which it is possible to state that with a certain probability p (usually $p = 0.9$ to $p = 0.99$) the mean value of the parameter is within the stated confidence interval. The value $\alpha = 1 - p$ is referred to as the *level of significance*, whereas p is denoted as a *level of confidence*. The width of the interval becomes narrower with the increase of the number of independent simulation runs (i.e. replications).

The biggest advantage of discrete-event simulation is that it is the only approach that is able to simulate almost every model as long as there is a way to describe it. The

Algorithm 2.1: Discrete-Event Simulation

```
1  while FEL not empty do
2  |  Remove the first primary event E from FEL;
3  |  begin
4  |  |  Update state variables according to E;
5  |  |  Insert new events in FEL according to E;
6  |  |  if E enables a secondary event SE then
7  |  |  |  Update state variables according to SE;
8  |  |  |  Insert new events in FEL according to SE;
9  |  |  end
10 |  end
11 end
```

simulation itself is very straightforward and, as shown in Algorithm 2.1, very easy to be programmed.

The problem with discrete-event simulation is that for certain types of models, such as are models with big differences in rates of the events, a large number of replications is required in order to achieve narrow confidence intervals i.e. a higher accuracy. The second problem is the quality of the random-number generators, and it is definitely not a trivial task to find or design one that is acceptable (Niederreiter 1992; Marsaglia and Zaman 1990; Park and Miller 1988).

Compared to the other approaches, discrete-event simulation is developed for imitating the system. Therefore the analysis is implicit and derived by monitoring the modelled behaviour of the imitated system. That is unlike Markov chain analysis methods for example, where the analysis is direct.

Further, in Section 6.6 we show a typical case where the discrete-event simulation can become very expensive in computation time, and for which the proxel-based simulation is significantly more efficient (rare-event models). There are, however, many cases for which the discrete-event simulation succeeds and for which the memory requirements of the proxel-based method are so high that they are not realistically feasible. Such an example is shown in Section 7.3.

2.2.2 Markov Chains

Markov chains can be used for describing and analysing models which contain exclusively exponentially distributed state changes (Kulkarni 1995; Çinlar 1975; Kolmogorov 1950; Stewart 1995). There are many processes for which this condition does not represent a limit, and that is where Markov chains find their application. They are one of the most popular tools for modelling and analysing stochastic systems, which possesses both simplicity and the ability to model various classes of stochastic systems.

Depending on the character of the time (i.e. parameter) domain, there are two types of Markov chains: discrete-time Markov chains (DTMCs) and continuous-time Markov chains (CTMCs). They both characterise stochastic processes where transitions to the next state depend only on the current state, completely irrelevant of the history of the processes. This property is known as the *Markov property*, and is named after the Russian mathematician A.A. Markov who in the early 1900's systematically studied discrete-time stochastic processes satisfying this property. The Markov property is also known as "*memorylessness*", and is formally described as follows:

$$Pr\{X(s+t) = j |\, X(s) = i, X(p) = k, 0 \leq p < s\} = Pr\{X(s+t) = j |\, X(s) = i\} \tag{2.10}$$

where $\{X(t) : t \geq 0\}$ is a stochastic process, $i, j, k \in S$ (S is the state space), and $s, t \geq 0$. The class of stochastic models that satisfy the Markov property is referred to as *Markovian models*.

For processes that do not satisfy the Markov property (*non-Markovian*), there are fitting algorithms which convert the non-exponential distribution functions into Markov chains, resulting into the so-called *phase-type distribution functions*, which can be both discrete and continuous, as described further in Section 2.2.3. The biggest disadvantage of both approaches is the expensiveness to accurately fit finite support distributions, such as Uniform or Deterministic, as stated in (Isensee and Horton 2005a).

In the following we describe the two types of Markov chains, DTMCs and CTMCs in detail.

Discrete-Time Markov Chains

A DTMC is defined by its transition probability matrix $P = [p_{ij}]$, $i, j \in S$, i.e.

$$P = \begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0j} & \cdots \\ p_{10} & p_{11} & \cdots & p_{1j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ p_{i0} & p_{i1} & \cdots & p_{ij} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix}, \tag{2.11}$$

where

$$p_{ij} = Pr\{X_{k+1} = j | X_k = i\}, k \geq 0 \tag{2.12}$$

define the conditional probabilities for the one-step state changes, and its initial state probability vector π_0. Very often, the time that the model spends in each of the states in

a DTMC has no importance, and therefore the sequence of states from one observation might not have the notion of time at all. Instead, each of the state occupancies in the sequence can be referred to as a step, and not necessarily a time step. Each of the steps can, however, have a certain length of time associated with it (Δt), referred to as *time step* of the DTMC.

A DTMC is said to be *time homogeneous* if

$$Pr\{X_{k+1} = j \mid X_k = i\} = Pr\{X_{m+k+1} = j \mid X_{m+k} = i\}, \quad n = 0, 1, 2, ..., \quad m \geq 0, \quad i, j \in S. \tag{2.13}$$

Throughout the thesis whenever we talk about DTMCs we consider time-homogeneous DTMCs.

The basic approach to solving DTMCs is the so-called *power method*, which works in an iterative manner, computing the following equation

$$\pi_k = \pi_{k-1} P, \tag{2.14}$$

where π_k is the state occupancy probability row-vector at the k-th time step. This method computes the transient solution, but also the steady-state one, implicitly by iterating longer. The latter one can be computed only under the condition that it exists, depending on the model. If that is the case, then it is said that the model has a *limiting behaviour* and the steady-state solution is

$$\pi = \lim_{k \to \infty} \pi_k.$$

All row-vectors $x = [x_i]$ that satisfy the equations

$$xP = x \text{ and } \textstyle\sum_{i \in S} x_i = 1$$

are said to be *stationary solutions* of the DTMC. One DTMC can have infinitely many stationary solutions, and at the same time no steady-state one. In addition, the underlying stochastic process of the proxel-based method is a discrete-time Markov chain, as described in Section 3.1.4.

Continuous-Time Markov Chains

CTMCs, opposed to DTMCs, are characterised by a transition *rate* (or generator) matrix Q, which instead of probabilities, stores rates of the possible state changes $q_{ij} \geq$

$0, i, j \in S$, S is the state space, under the assumption that they are all exponentially distributed, in fulfilment of the memorylessness requirement, i.e.

$$Q = \begin{pmatrix} q_{00} & q_{01} & \cdots & q_{0j} & \cdots \\ q_{10} & q_{11} & \cdots & q_{1j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ q_{i0} & q_{i1} & \cdots & q_{ij} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix}. \tag{2.15}$$

The diagonal elements are chosen to ensure that the sum of the elements in every row is zero, i.e.

$$q_{ii} = -\sum_{i,j \in S, j \neq i} q_{ij}.$$

The mean time that the model spends in each state within one visit is known as *sojourn time*. It is exponentially distributed and can be computed as

$$E\left[sojourn\ time\ in\ state\ i\right] = -q_{ii}^{-1}. \tag{2.16}$$

The steady-state probability vector π for a CTMC is obtained as a solution to the following equations:

$$\pi \times Q = 0 \text{ and } \sum_{i \in S} \pi_i = 1,$$

where the second equation is the normalisation condition, and π_i are the steady-state probabilities for the different states of the CTMC and which system of equations provides a unique solution when existing. These equations come as a result of the so-called *balance equations* which basically state that the probability of entering a state is the same as the probability of leaving it.

The transient solution of a CTMC is obtained by solving the following Kolmogorov system of differential equations:

$$\frac{d\pi(t)}{dt} = \pi(t) \times Q,$$

whose solution has the following closed form:

$$\pi(t) = \pi(0) \times e^{Qt}.$$

Every CTMC can be embedded into a DTMC by defining small enough time step Δt that would be able to capture the rates of the state changes, which basically means

that $0 < \Delta t \leq min_i \left\{|q_{ii}^{-1}|\right\}$. The probability matrix P of the *embedded* DTMC is then constructed as

$$P = I + \Delta t \times Q. \tag{2.17}$$

This conversion is known as uniformisation or Jensen's method (Jensen 1953; Grassman 1990) and yields a numerical method for obtaining the transient probabilities. Also, a set of iterative methods is developed for obtaining the steady-state solution, such as Gauss-Seidel or SOR (Successive Overrelaxation), as presented and elaborated in (Stewart 1995). The iterative methods take advantage of the common property of generator matrices i.e. sparseness.

2.2.3 Modern Approaches and Tools

The class of Petri nets that contains only exponentially distributed and immediate transitions is referred to as *generalised SPNs* (Kartson et al. 1994; Marsan et al. 1984, 1998), and is one of the early forms of Petri nets. It is easily convertible to Markov chains[2], and thereby analysable by the numerical methods for solving Markov chains (Stewart 1995).

However, over the years there have been many extensions to the initial idea of generalised SPNs, all trying to overcome the limitations of having only exponentially distributed transitions. There are, namely, many processes in the real world that can not be classified as "memoryless", some of them even having activities with deterministic duration, or at least with small variances. Accordingly, there has been a series of approaches that aimed at analysing the class of so-called *non-Markovian* stochastic Petri nets (Trivedi et al. 1995; German 2002). We refer to the SPNs that allow other distributions in the models besides exponential as *non-Markovian SPNs*.

The analytical analysis methods for non-Markovian models can be classified as based on one of the following three approaches (Puliafito et al. 1997):

- Markov renewal theory (Choi et al. 1994),
- method of supplementary variables (Cox 1955a; German 2000b), and
- phase-type approximations (Neuts 1981; Bobbio et al. 2002; O'Cinneide 1999; Bobbio et al. 2000).

The method of supplementary variables is discussed in Section 2.1.2, therefore, in the following we briefly review the other two paradigms, and discuss some of the tools that implement them. The common property, however, for all three approaches is that they all ultimately construct a Markov process from a non-Markovian one.

[2]The reduced reachability graph of a GSPN is the state-space of the underlying CTMC.

Markov Renewal Theory

After in (Choi et al. 1994) it was shown that the class of SPNs where at most one non-exponential transition with enabling memory policy was enabled in each marking, belonged to the class of Markov Regenerative Processes (MRGPs), the Markov renewal theory became applicable for the analytical solutions of that class of SPNs, named as Markov regenerative SPNs. That lead to an active research for different approaches, as presented in (Telek et al. 1995; German et al. 1995a; German and Mitzlaff 1995; Horvath et al. 2000; Telek and Horvath 2001; Fricks et al. 1998) and implemented in tools, such as TimeNET(German et al. 1995a) and WebSPN (Bobbio et al. 1997).

According to the Markov renewal theory the stochastic process is considered at discrete points in time, referred to as *regeneration points*. At these points, the Markov property is valid for the stochastic process, i.e. the process probabilistically resets and goes into a state where the history is irrelevant. Formally, Markov regenerative processes are defined as follows:

Definition 2.1 – Markov regenerative process (MRGP). *$Z(t)$ is a Markov regenerative process (MRGP) if there exists a Markov renewal sequence $\{X_n, T_n; n \geq 0\}$ that*

$$Pr\{Z(T_n + t_1) = x_1, \ldots, Z(T_n + t_m) = x_m | Z(T_n), Z(u), 0 \leq u \leq T_n\} = Pr\{Z(T_n + t_1) = x_1, \ldots, Z(T_n + t_m) = x_m | Z(T_n)\}$$

for all $m \geq 1, 0 < t_1 < \ldots < t_m$ and $x_1, \ldots, x_m \in S$.

In other words, $Z(t)$ is a MRGP if there exists a Markov renewal sequence $\{X_n, T_n; n \geq 0\}$ of random variables such that all the finite dimensional distributions of $\{Z(T_n + t); t \geq 0\}$ given $\{Z(u); 0 \leq u < T_n, X_n = i\}$ are the same as those of $\{Z(t); t \geq 0\}$ given $X_0 = i$.

An example of a Markov regenerative process is a queuing system with one server, where the arrivals are exponentially distributed and the service times follow a general distribution function. The regeneration points for that stochastic process are the points in time that stand for service completion. At those points the probabilities for the possible events happening, i.e. customer arrival and service completion, are independent of the history. Therefore the sequence of the points in time that indicate service completion forms a Markov renewal sequence. Consequently the stochastic process is a Markov regenerative process.

It is interesting that both Markov renewal processes and the method of supplementary variables ultimately develop into same system of state equations, which yields the same analysis algorithm when solving for the steady-state behaviour, as pointed at (German 2000b). Hence both approaches, when solved analytically, have a limitation on the number of concurrently active non-exponential activities, which if neglected can result

in multi-dimensional differential equations which are hard to solve. There is, however, a difference in the way the algorithm is formulated, which when based on the Markov renewal theory, can also be solved in an iterative fashion, as shown in (German 2000a). Therefore the MRGP approach is more suitable for more complex models.

The biggest limitation of the application of the Markov renewal theory is that it can only be applied to processes classified as MRGPs, which means processes that contain points in time at which they restart probabilistically. The proxel-based method does not impose this requirement on models, and regarding the applicability, it is more general than the approaches based on Markov renewal theory.

Phase-Type Approximations

Phase-type distributions were initially described by Neuts in (Neuts 1981), and appear in general in two forms: *discrete* (Ciardo 1995; Isensee and Horton 2005a) and *continuous* (Bobbio et al. 2002). For both of them it is characteristic that they substitute generally distributed activities described by continuous distributions by *finite* and *absorbing* Markov chains, where the number of phases is the number of transient states of the associated Markov chain, and denotes the *order* of the phase-type approximation. The approximated phase-type distribution then describes random variables that are measured by the time that the Markov chain spends in its transient portion till absorption. The approximation is performed in different ways, some of the approaches work by solving a system of equations, as shown in (Osogami and Harchol-Balter 2003), and others by fittings using different optimisation algorithms, as shown in (Isensee and Horton 2005a; Bobbio et al. 2002).

Although not limited in their structure, there are a couple of phase-type distribution functions which are most common and popular. Some of the continuous ones are Erlang (Erlang 1917) and Coxian (Cox 1955b), as shown in Figure 2.5 in their three-phase form for the case of approximating a general distribution which describes the state change from state A to state B. The initial probabilities for the CTMC are denoted by p_i, whereas λ and λ_i refer to the transition rates.

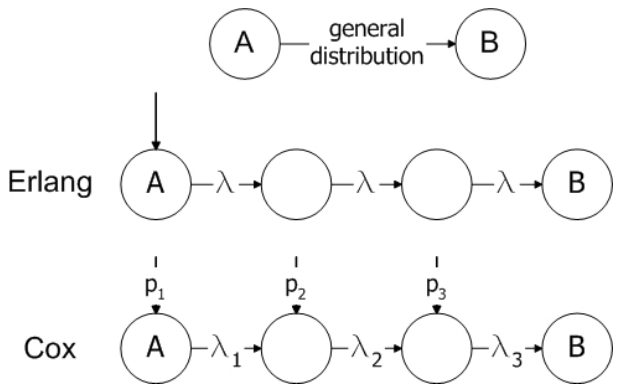


Figure 2.5: Examples of continuous phase-type distributions

In Figure 2.6 a minimal structure of discrete phase-type distributions is shown, referred to as a *canonical form*. The initial probabilities for the DTMC are denoted by p_i, whereas q_i and $1 - q_i$ refer to the transition probabilities. In (Bobbio et al. 2000) it has been proven that the canonical form is a minimal one and all other acyclic forms[3] can be reduced to a canonical of the same order.

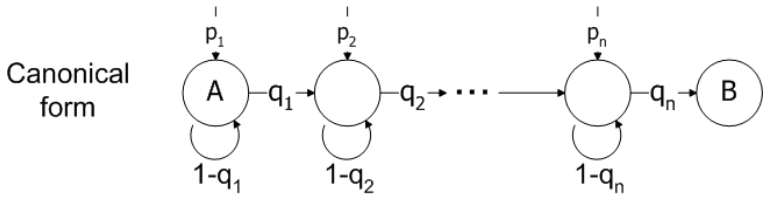


Figure 2.6: Canonical form of discrete phase-type distributions

The term *state* in the case of phase-type approximation refers to the combination of the discrete state and the phase, i.e. (DS, ϕ). The state space of the model is thus extended, and depending on the type of distribution functions that are being approximated, it can become quite complex, as is the case with uniform and deterministic distributions. The reason for this is that those distributions require a large number of phases for computing an acceptably accurate approximation.

In conclusion, phase-type approximation, being another approach for generating Markovian models from non-Markovian, can be seen as an attractive choice for performing numerical analysis of discrete stochastic models, thereby using the standard Markov chain solution methods. However, because of the enlargement of the state space due to the addition of phases, the method suffers from the well-known state-space explosion problem and distributions with finite support (such as uniform or deterministic) are badly approximated by the phase-type distributions. One solution to that problem is the combination of discrete phases-type approximation and proxels, which is explained in Section 4.2.2. Namely, the proxel-based method approximates very well distribution functions which are problematic for the phases, and discrete phase-type approximations approximate well distributions which are problematic for the proxels regarding the computational complexity.

Tools: TimeNET, Möbius

There are not many tools that provide numerical analysis of non-Markovian models. We choose to look at two of them, which we think deserved our attention: TimeNET and Möbius.

TimeNET (German et al. 1995a) is a tool developed for modelling and analysis of stochastic Petri nets with non-exponentially distributed firing times, mainly developed

[3]Definition - A discrete phase-type approximation is called *acyclic discrete phase-type approximation* if its states can be ordered in such a way that the stochastic matrix of the underlying DTMC is an upper triangular matrix. (Horváth 2003)

for performability modelling. It is developed and maintained by the performance evaluation group of the Computer Science Department at Technische Universität Berlin.

The type of analysis that TimeNET provides depends basically on the class of Petri net being analysed, i.e. its complexity. It computes numerical solutions, performs discrete-event simulation and phase-type approximations of general distribution functions, as well as structural analysis of the Petri net.

The disadvantage of the numerical analysis that TimeNET provides is that the analysis algorithms require that all transitions with non-exponentially distributed firing delays are mutually exclusive. If that is not the case, then the tool chooses discrete-event simulation as an analysis method, which can require long running times for sufficient accuracy (Zimmermann et al. 2000).

Some of the latest extensions allow the tool to model coloured stochastic Petri nets and carry out performability evaluation of stochastic Petri net models with non-exponentially distributed firing times, with a special emphasis on of manufacturing systems. Another novelty is the possibility of analysing models with a discrete time scale, which means that their underlying stochastic processes are Markovian if geometrically distributed firing delays are used. The geometric distribution is analogous to the exponential in the continuous-time case, and the tool allows switching between the two modes.

Möbius (Deavours et al. 2002) represents a multi-formalism software tool which resulted as a work of a complex project in which people from different universities took part. The tool tries to incorporate many of the established modelling and analysis methods to provide flexibility for the users, in order to avoid the case where everyone needs to spend a reasonable amount of time learning a new modelling formalism in order to analyse models. The modelling formalisms that it unifies are: stochastic extensions to Petri nets, Markov chains and extensions, and stochastic process algebras, and as analysis methods it offers discrete-event simulation and numerical solution techniques. The reason for providing this multi-formalism tool is the belief that one modelling or analysis method is not sufficient for studying the behaviour of the different types of complex systems that exist today which are mixture of many domains (Sanders et al. 2003).

Besides these two more advanced tools, there are only a few that implement deterministic or numerical approaches towards solving and analysing discrete stochastic models, mostly because of the limitations that the numerical approaches have with respect to the classes of analysable models. Discrete-event simulation is still the method of choice when it comes to analysing more complex models as it simply reproduces their behaviour and has no difficulties with that respect. The problem is then the accuracy of the solutions that it provides and the computationally expensive replications. Therefore, there is a need for deterministic and widely applicable method, for which we believe that the proxel based method is a good candidate, as will be explained in the next sections.

2.3 Summary

This chapter presents an overview of the basic concepts, terminology, and related work regarding the problems that we address in this dissertation. It defines what a discrete-stochastic model is and describes the main idea of the method of supplementary variables as a basis for the design of the proxel-based method. Finally, it provides a review of the most relevant and frequently-used methods for simulation and analysis of discrete stochastic models.

3 The Proxel-Based Method

This chapter provides an introduction to the proxel-based method along with definitions of the basic elements used for its description, and together with Chapter 2 defines the terminology used throughout the thesis. In Section 3.1, the foundations of the proxel-based method are established, after which in Section 3.2 its accuracy is discussed. Furthermore, in Section 3.3, additional algorithms are presented which describe two preprocessing steps which aid the implementation and the ease of use of the proxel-based method. Finally, in Section 3.4, the programming aspects and the complexity of the approach are discussed.

3.1 Foundations of the Proxel-Based Method

The goal of this section is to introduce the proxel-based method and establish its formal foundations. The proxel-based method is a new approach to analysing discrete stochastic models in a deterministic way. We believe that it is intuitive and easy to understand because it traces all of the possible developments in one model in a natural way. Thereby the probability is correspondingly distributed, weighting all possible paths that the model can take.

In general every simulation consists of three steps even though for some approaches some of the steps are merged: creating the user model, developing the computational model based on it, and finally solving the computational model using a certain solution algorithm. This process is illustrated in Figure 3.1. The PDE-based approaches contain all three steps, e.g. if the user model is described using a Petri net, the computational model is developed from the Petri net and results in a system of PDEs, which are then solved using a numerical method for solving PDEs. In the case of discrete-event simulation all three steps are merged and based on the description of the user model because DES solves one model by reproducing and observing its behaviour.

In the case of the proxel-based method, the user model yields directly the solution algorithm, without constructing a computational model, so the last two phases are

merged. This is a very advantageous feature of the method, which makes it very flexible in analysing different classes of discrete stochastic models.

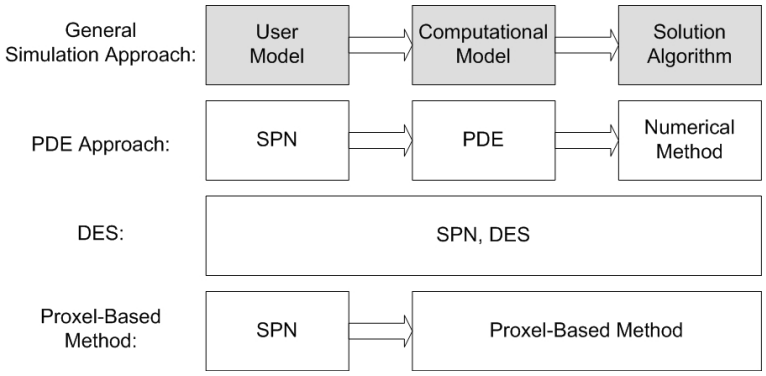


Figure 3.1: Comparison of simulation approaches

One advantage of the fact that the proxel-based method operates directly on the model is that it builds the state space on-the-fly (Deavours and Sanders 1998), ensuring that non-zero probabilities are assigned only to the truly reachable states, which are the only ones that are being stored. This yields a more rational memory use.

The proxel-based method belongs also to the class of methods that convert non-Markovian stochastic processes into Markovian ones, which the method achieves in a very intuitive manner deriving the solution algorithm directly from the description of the model. The proxel-based method does not involve setting up and solving partial differential equations (unlike the original implementation of the method of supplementary variables does (Cox 1955a; German 1998)) or approximating distribution functions by fitting (unlike phase-type approximations (Bobbio et al. 2002; Isensee and Horton 2005a)).

Because of the above described nature and features, the proxel-based method is very flexible and able to analyse a wide class of stochastic models. The approach, however, suffers from the well-known problem of state-space explosion, and that is its only real limitation until now.

In this section we provide a description of the method, along with an example, as well as definitions and algorithms that specify the proxel-based method.

3.1.1 Description

Proxel-based method is a numerical approach for analysing discrete stochastic models which implements the method of supplementary variables in a way that the partial differential equations are completely avoided, by tracking the flow of probability correspondingly to the behaviour of the model. This is achieved by employing the instantaneous rate function (defined in Section 2.1.1) which determines the rate with which one

event can happen based on the time that the event was scheduled and able to happen, where the meaning of "able" is *all preconditions for its happening are satisfied.*

An initial step to every proxel-based simulation is the determination of the size of the time discretisation step Δt. This should be arranged in coordination with the specifications of the distribution functions involved in the model, in a way that the size of the time step is small enough to capture the rates of all possible state changes. It is also a factor that determines the accuracy of each simulation. More on this subject is discussed in Section 3.3.1, which provides more insight into the way of determining an acceptable size of the time step and generalises a heuristics that we use for that purpose.

Proxel, which as a term is constructed as an analogy of the well-known *pixel* is an abbreviation of the phrase "**pro**bability **el**ement". It is the central computation unit of the proxel-based method and it describes every probabilistic configuration of the model in a minimal and complete way. This means that each proxel carries adequate amount of information for generating its successor proxels, or in other words for determining probabilistically how the model could behave from there on. This is the fact that turns a non-Markovian model into a Markovian one. The information that is found to be necessary and each proxel contains, is the following:

- discrete state that the model is in, which corresponds to the notion of "tangible marking" in Petri net terminology,
- age intensities of the possible (i.e. active) state changes in the current discrete state, which track the time that each of the possible state changes has been pending; they are necessary for determining the probabilities for each of the state changes happening,
- global simulation time, which describes the absolute time coordinate from the beginning of the simulation,
- route(s), which describes the sequence(s) of states via which the model has reached the current state contained in the proxel, and
- probability that the system is in the current discrete state having the actual age intensities and having been reached via the sequence of states declared in the route.

The formal definitions of the elements of the proxel are provided in Section 3.1.2.

We define the term *state* as a vector composed of a discrete state of the model and the age intensities of the possible (i.e. active) state changes.

Represented in a formal way, each proxels has the following format:

$$Proxel = (State, Time, Route, Probability), \text{ where}$$
$$State = (Discrete\ State, Age\ Intensity\ Vector),$$

and the *Age Intensity Vector* stores the age intensities of all relevant state changes.

As proxels store the necessary information for describing any probabilistic configuration of the model, the initial proxel that describes the initial configuration of the model at time $t = 0$ has the following form:

$$\begin{gathered} Initial\ Proxel = (Initial\ State, 0, \oslash, 1.0), \text{ where} \\ Initial\ State = (Initial\ Discrete\ State, \vec{0}). \end{gathered}$$

The symbol $\oslash$ denotes an empty route, which means that the model has no history yet. Once the initial proxel is generated, the simulator generates its successors based on the possible state changes and accordingly updates the age intensity vector and calculates its probabilities. In the proxels which represent different discrete state from the initial one, the appropriate age intensity (i.e. the one of the corresponding state change) is reset to zero, or becomes irrelevant. The latter situation corresponds to the state change being not represented in the age intensity vector. The age intensities of the possible (active) state changes in the proxel that represents the model staying in the initial discrete state in the next time step, are incremented by the size of the time step Δt, meaning that those state changes have now been pending for a time period of Δt. In this case the possible state changes are *aging* which usually increases the probability with which they might happen.

Once the second generation of proxels has been computed, the first one can be thrown away. The completeness of the information in the proxels supports this step. The next (i.e. third) generation is computed in the same way as the second one, except that the process of generating successors is now performed for more than one proxel. This procedure is carried out repeatedly until the predetermined end of the simulation time is reached. The structure that represents the generations of proxels as well as their relations in terms of successors and predecessors is referred to as a *proxel tree*.

During the computation of each generation of proxels, many proxels may occur that represent the same *state*. In the case where such proxel is generated, however not for the first time, it is not stored as new proxel, but instead the definition of the existing one is extended and updated. Its route parameter is now extended to the union of all routes, generated until the current point in time, that lead to that state; correspondingly its probability element is set to the sum of the state-matching proxels' probabilities. The proxels that have null or negligible probabilities are not stored. This leads to the conclusion that usually at some stage the width of the proxel tree reaches its maximum. This is so because most of the distribution functions have limited range of non-negligible values, i.e. if a state change is distributed according to a function other than exponential, usually it has a limited time within which it can happen with a non-negligible probability. The limit in the proxel-based method is determined dynamically which we believe is an essential advantage of the approach compared to the Markov chains-based approaches.

Example

We use a very simple example, as shown in Figure 3.2, to demonstrate how the proxel-based method works. The model is described using a simple state-transition diagram which allows generally distributed state changes. It is a two-state model which has two possible state changes that switch between the two discrete states, A and B.

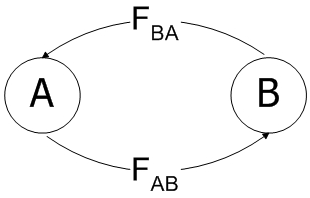


Figure 3.2: Example state diagram

Let A be the initial discrete state of this simple model and Δt the size of the time step. There is at most one age intensity that needs to be tracked in each discrete state, and correspondingly in each proxel. In A it is the age intensity of the state change from A to B, distributed according to the distribution function F_{AB}, and in B it is the one of the state change from B to A distributed according to F_{BA}. The initial proxel for this model is the following:

$$((A, 0), 0, \oslash, 1.0)$$

meaning that the model is in state A with age intensity of the state change from A to B equal to zero, at time $t = 0$ with a probability of 1.0, having no historical route of previously visited states ($\oslash$).

In the next time step, at $t = \Delta t$, the model can either be still residing in the discrete state A or have changed to the discrete state B, in which case the age intensity variable is reset and now it tracks the age intensity of the state change from B to A. The proxels that are thereby generated are the following:

$$((A, \Delta t), \Delta t, ((A, 0)), 1.0 - \mathit{probability}) \text{ and } ((B, 0), \Delta t, ((A, 0)), \mathit{probability}).$$

The first proxel denotes staying in discrete state A, whereas the second one a change to the discrete state B. The *probability* is approximated by the instantaneous rate function $\mu(\tau)$, integrated along the time step, where τ is the age intensity of the active state change i.e.

$$\mathit{probability} = \int_0^{\Delta t} \mu(x)dx,$$

which we approximate for our example by:

$$\mathit{probability} = \mu(0) \times \Delta t, \tag{3.1}$$

which in this case we interpret as the probability that the state change has happened within the interval $[0, \Delta t)$. The instantaneous rate function is computed from the distribution functions (CDF and PDF) using Equation (2.3) provided in Section 2.1.1. We also use more advanced approaches for integrating the IRF, which provide better approximation of the probabilities, as described in Section 3.2.

Now, let us observe the two generated proxels and compute their successors at time $t = 2\Delta t$. From the first one, $((A, \Delta t), \Delta t, ((A, 0)), 1.0 - probability)$, the following two can be computed:

$$((A, 2\Delta t), 2\Delta t, ((A, 0), (A, \Delta t)), *) \text{ and } ((B, 0), 2\Delta t, ((A, 0), (A, \Delta t)), *),$$

which again means that the model can stay in the discrete state A or have changed to B.

The second proxel from the second generation, $((B, 0), \Delta t, ((A, 0)), probability)$, analogously determines its successors as follows:

$$((A, 0), 2\Delta t, ((A, 0), (B, 0)), *) \text{ and } ((B, \Delta t), 2\Delta t, ((A, 0), (B, 0)), *).$$

The probabilities are omitted for reasons of simplicity. They can, however, be computed in the same manner as previously, using the IRF, relying only on the age intensities included in the proxels. The only difference is that this time they have to be multiplied by the parent-proxels' probability values because those are the amounts of probabilities that are available and being distributed.

The tree that describes the proxel generation process is referred to as a *proxel tree*, and for this simple example up to time $t = 2\Delta t$ is shown in Figure 3.3.

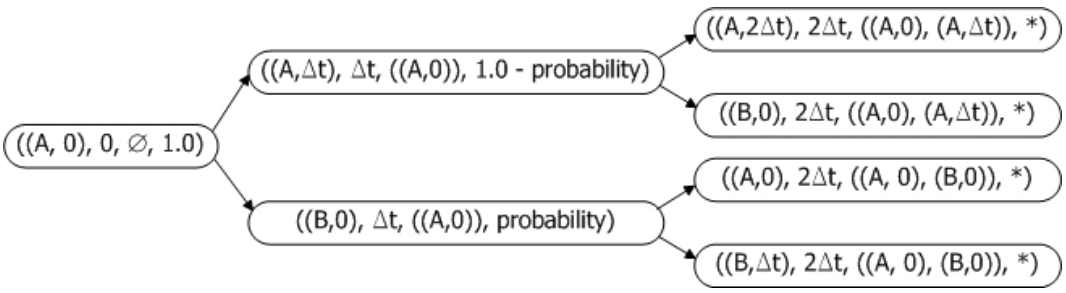


Figure 3.3: The proxel tree of the first three time steps for the example model from Figure 3.2

In the whole proxel-generation process we make one very important assumption in our approach. That is, we suppose that the model does at most one state change within a time period of Δt. The assumption is ultimately justified when $\Delta t \to 0$, which means that the smaller the time step, the higher accuracy can be achieved. The assumption, however, needs to be acknowledged, and is commented and elaborated in Section 3.2, which discusses the accuracy of the proxel-based method.

3.1.2 Definitions

In this section we provide definitions for all key concepts and elements that are relevant for the proxel-based simulation. We begin by describing the most basic terms, some of which have been already informally described and used in the previous Section 3.1.1.

Definition 3.1 – Discretisation time step. *Discretisation time step is the time step at which the proxel-based simulation advances, under the assumption that the probability of more than one state changes happening within one time step is negligible.*

Definition 3.2 – Discrete state. *A discrete state of a model is a configuration of the model in which the elapsed amounts of time of the active state changes are ignored. It is analogous to the concept of "tangible marking" in Petri nets.*

Definition 3.3 – State change. *A state change in a model is a change of the configuration of the model, defined by the discrete state. A state change happens when an event occurs. Analogously to the Petri net formalism (elaborated in Section 2.1.3), we allow two types of state changes according to their memory policy:* age *and* enabling.

Definition 3.4 – Active state change in a discrete state. *An active state change in a discrete state of a model is a state change which* can (i.e. is able to) *take place in the actual discrete state. (Example: In a queuing system, a customer cannot leave the service if the server is not occupied, which means that in a discrete state in which the server is empty, the state change "service completion" is* not *active)*

Definition 3.5 – Age intensity. *Age intensity of a state change is the amount of time that the state change has been active without it actually happening.*

There are, however, sometimes more than one active state changes (or ones with age memory policy) whose age intensities need to be separately tracked in the proxels. Therefore the age intensity definition extends to an age intensity vector. The age intensity vector has a corresponding mapping vector, which shows which state changes are associated with which components in the vector.

Definition 3.6 – Age intensity vector. *Age intensity vector of a discrete state is the vector of the age intensities of all active and age-memory state changes in one discrete state.*

Definition 3.7 – Mapping of an age intensity vector. *Mapping of an age intensity vector is a vector characteristic for each discrete state, which shows the association of the active and age-memory state changes to the components of the age intensity vector, when the model resides in that discrete state.*

Definition 3.8 – State. *A state is a combination of a discrete state and its age intensity vector.*

Definition 3.9 – Route. *A route is a sequence of states through which a model has passed before arriving in the state described by the proxel.*

Definition 3.10 – Proxel (1). *A proxel is a probability computation unit, made up of the following elements: a state, a global simulation time, a route(s) of states, and a probability for being in the given state at the stated point in time referred to as a global simulation time, given that the model has arrived there via the sequence(s) of states given by the route(s).*

The proxel-based method advances by distributing the initial probability of 1.0 for being in the initial state among all of the subsequent states of the model. Proxels are the computational units that store, keep track of, and characterise the flow of probability from one state to another. This means that the proxel-based simulation repeatedly generates from each proxel the set of successive proxels, until the end of the simulation time is reached. In this manner, a tree structure of proxels is created, which is referred to as the "proxel tree". The proxel tree is actually the state space of the model in terms of proxels, which enjoys the Markov property[1] if the arcs are to labelled by the probabilities for the state changes computed from the *IRFs*.

Definition 3.11 – Proxel tree. *Proxel tree is a tree-like data structure that represents the hierarchical state space of the model, having proxels as its nodes.*

As explained in the previous Section 3.1.1, during the process of generating the proxel tree, at any time step, one same state S may be generated many times, each time via a different sequence of predecessor states, i.e. a different route R. In order to obtain the total probability for that state and to optimise the storage of the proxels, the probabilities of all of that state's instances are summed up and a new proxel Px that represents the state is generated. This proxel is stored as a representative of the corresponding state S and its route parameter is extended to the union of all routes R_i that lead to that state at the current discrete time step. This can be formally represented in the following way:

$$Px = (S, t, \bigcup_{i:R_i \overset{t/\Delta t}{\rightarrow} S} R_i, Pr(\textit{model in state } S \textit{ at time } t)), \tag{3.2}$$

where

$$Pr(\textit{model in state } S \textit{ at time } t) = \sum_{i:R_i \overset{t/\Delta t}{\rightarrow} S} Pr(\textit{model in state } S \textit{ at time } t | S \textit{ reached via } R_i), \tag{3.3}$$

and $R \overset{n}{\rightarrow} S$ means "route R leads to state S in n steps".

[1] memorylessness

The formal representation of the transient probabilities of the discrete states at different points in time, as well as the calculation of proxels' probabilities in the next discrete time step based on the current one are the following:

$$Pr(\textit{model in discrete state } DS \textit{ at time } t) = \sum_{\vec{\tau}:S_k=(DS,\vec{\tau})} Pr(\textit{model in state } S_k \textit{ at time } t), \tag{3.4}$$

where

$$Pr(\textit{model in state } S_k \textit{ at time } t) = \sum_{i,j:R_j \overset{t/\Delta t-1}{\rightarrow} S_i} Pr(\textit{model in } S_i \textit{ at time } t-\Delta t) \times \mu_{ik}(\tau_{ik}) \times \Delta t, t \geq \Delta t \tag{3.5}$$

where τ_{ik} is the age intensity of the state change that caused the transition from S_i to S_k, and $\mu_{ik}()$ - the corresponding IRF. Δt is the size of the discretisation time step.

3.1.3 Basic Algorithm

Once we have defined all terms involved in the definition of the proxel-based method, in this section we present the basic algorithm of the proxel-based method. It operates on a standard state-transition diagram, which consists exclusively of discrete states, i.e. tangible markings, and allows generally distributed state changes. Every bounded SPN model can be reduced to this discrete state-space representation. It is an analogy of a reduced reachability graph. The proxel-based method itself is not limited to analysing models that have finite state-spaces because the method builds the state space dynamically[2]. The Algorithm 3.1 that we present here is the basic algorithm, and it is extensible for analysis of different and more complicated classes of models. It has several variations for analysing, for example, models described using our proxel-adapted modelling framework, which allows modelling of a more general class of discrete stochastic models (Chapter 5), or performability models (Section 6.1). The proxel structure with which this algorithm operates is simplified to contain only the discrete state, the age intensity vector and the proxel's probability. The route and the global simulation time parameters are implicitly considered, and therefore not stored in the proxels.

The meanings of the functions and symbols used in the Algorithm 3.1 are the following:

[2] Analysing unbounded models and a wider class of stochastic models is made possible using the proxel-adapted modelling framework which is presented in Chapter 5. There we present an extended proxel algorithm which can analyse a wider class of models.

Algorithm 3.1: Basic Proxel Algorithm (1)

Input: $\Delta t, t_{max}$

Initialize proxel set $PS(0)$ by inserting the initial proxel;
$switch = 0$;
for *i=1* **to** $[t_{max}/\Delta t]$ **do**
  foreach *proxel* $p = ((DS, \vec{\tau}), prob)$ *in the proxel set* $PS(switch)$ **do**
    foreach *active state change* C_i *in* DS **do**
      Get x as the mapping index of C_i in $\vec{\tau}$;
      Compute $succ(\vec{\tau}) = \vec{\tau} \times I_x$;
      Compute $succ(DS)$ as a successor discrete state of DS, resulting from the state change C_i;
      Generate new state $S = (succ(DS), succ(\vec{\tau}))$;
      Compute probability $prob_{calculated} = IRF(C_i, \tau_x) \times \Delta t$ for the state change C_i;
      Search for the state S in the states of the set of generated proxels;
      if $proxel_{found}$ *is the found proxel* **then**
        $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$;
      else
        Generate new proxel $p_{new} = (S, prob_{calculated})$;
        Store p_{new} in $PS(1 - switch)$;
      end
      $prob_{rest} = prob - prob_{calculated}$;
    end
    Generate new proxel $p_{new} = ((DS, succ_{stay}(\vec{\tau})), prob_{rest})$;
    Remove the processed proxel p from $PS(switch)$;
  end
  $switch = 1 - switch$
end

- Δt - size of the discretisation time step,
- t_{max} - maximum simulation time,
- DS - discrete state,
- $\vec{\tau}$ - age intensity vector,
- $prob$ - probability value of the proxel being processed,
- $IRF(C, a)$ - the value of the instantaneous rate function for the random variable that describes the state change C and its age intensity a,
- τ_x - x-th component of the age intensity vector $\vec{\tau}$,
- $succ(\vec{\tau})$ - the updated age intensity vector, and
- I_x - an identity matrix which has a zero on the place of the (x, x)-th element.

The proxel-based method operates by switching between two data structures which store proxels from two consequent time steps. In the algorithm, this is realised by switching between the two proxel sets, $PS(0)$ and $PS(1)$, through the variable *switch* (line 23), where one set is the source from which proxels are removed and processed and the other one (of the successive time step) is the storage for the successors of the processed proxels. In the beginning, the initial proxel set ($PS(0)$) contains only one proxel, which corresponds to the initial discrete state of the model (line 1). The algorithm shows how every proxel from the current tree ($PS(switch)$) is being processed and removed (lines 4 and 21). The processing means computing its successor proxels (lines 5-20), thereby checking for each of them whether a proxel which represents the same state already exists (line 11). If the answer is positive, then no new proxel is created, but the computed probability is added to the one of the existing proxel (lines 12 and 13). If that is not the case, then a new proxel is generated and stored into the proxel tree of the next time step ($PS(1 - switch)$), as shown in lines 15 and 16. The computation of the proxel that represents the model staying in the same discrete state and its probability (which is the leftover of the parent-proxel's probability, after the probabilities for the state changes have been subtracted) are shown in lines 18 and 20.

The mapping of the age intensity vector shows how the active state changes, as well as those that have age memory policy are mapped to the age intensity vector $\vec{\tau}$. Therefore, its updating means preserving the values of the age intensities of the age memory state-changes and of the ones that did not cause the state transition, and setting the age intensity of the state change that has actually happened to zero. This is realised by the multiplication of $\vec{\tau}$ by I_x in line 7.

The algorithm presented in this section represents just the basic proxel-based approach. Further, in Section 3.3, supplementary algorithms are shown, which aid and support the proxel-based algorithm and the process of choosing parameters.

3.1.4 Characterisation of the Underlying Markov Chain

The proxel-based method belongs to the class of approaches that reduce non-Markovian models to Markovian ones because it extends the state definition to the point that it completely determines the probabilities for the possible state changes. In this subsection we show the details of the reduction process that the proxel-based method performs, by constructing and discussing the characterisation of the underlying discrete-time Markov chain. In order to do that, we define a time step of size Δt and we assume that it is so small that the probability that more than one state change within Δt occurs is negligible. Recall that the state $S = (DS, \vec{\tau})$ is defined by a discrete state DS and a vector of age intensities $\vec{\tau}$. For two states S_i and S_j, where S_i is defined at discrete time step k and S_j is defined at discrete time step $k + 1$, we compute the probability p_{ij} for the state change from S_i to S_j is approximated as follows:

$$p_{ij} = \mu_{ij}(\tau)\Delta t \tag{3.6}$$

where $\mu_{ij}(\tau)$ denotes the value of the instantaneous rate function of the age of the state change that leads from the discrete state of the state S_i to the one of S_j. The probability that the model does not change the discrete state within the next time step (i.e. no state change happens in the interval $(\tau, \tau + \Delta t)$) is then 1.0 minus the sum of the probabilities of all possible (active) state changes.

For a given model we can thus define a discrete time Markov chain in which the unknowns correspond to the discretised states of the model and the coefficients of the probability matrix P are given by the Equation (3.6). For simplicity, we describe the case where the dimension of the age intensity vector is one; this is not a limitation of the proxel-based method.

We consider as an example a simple system which consists of a cashier who can be either free (DS_0) or busy (DS_1). The state diagram of the model is shown in Figure 3.4.

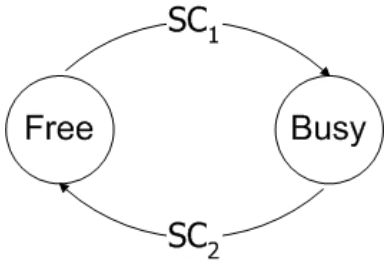


Figure 3.4: State diagram of the cashier model

We assume for simplicity reasons, that state changes SC_1 and SC_2 are exponentially distributed with rates λ and μ, respectively. This means that the instantaneous rate functions for both state changes have constant values λ and μ. We also assume, for brevity, that the maximum length of time that each transition can be enabled is $\tau_{max} = 3 \times \Delta t$. The transition probability matrix P for this example is as follows:

$$
\begin{array}{c}
 \\
\pi_{00} \\
\pi_{01} \\
\pi_{02} \\
\pi_{03} \\
\pi_{10} \\
\pi_{11} \\
\pi_{12} \\
\pi_{13}
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
\pi_{00} & \pi_{01} & \pi_{02} & \pi_{03} & \pi_{10} & \pi_{11} & \pi_{12} & \pi_{13}
\end{array} \\
\begin{pmatrix}
0 & 1-\lambda\Delta t & 0 & 0 & \lambda\Delta t & 0 & 0 & 0 \\
0 & 0 & 1-\lambda\Delta t & 0 & \lambda\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 1-\lambda\Delta t & \lambda\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mu\Delta t & 0 & 0 & 0 & 0 & 1-\mu\Delta t & 0 & 0 \\
\mu\Delta t & 0 & 0 & 0 & 0 & 0 & 1-\mu\Delta t & 0 \\
\mu\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 1-\mu\Delta t \\
\mu\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

We denote by π_{ij} the state of the DTMC which corresponds to the model being in discrete state DS_i and the age intensity of the transition being $j \times \Delta t$. Then, π_k is given by

$$\pi_k = [\pi_{00}, \pi_{01}, \pi_{02}, \pi_{03}, \pi_{10}, \pi_{11}, \pi_{12}, \pi_{13}]$$

If the state changes are not exponentially distributed and their IRFs are correspondingly the functions of the age intensities: $\lambda()$ and $\mu()$, then the probability matrix P has the following form:

$$
\begin{array}{c}
 \\ \pi_{00} \\ \pi_{01} \\ \pi_{02} \\ \pi_{03} \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13}
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} \pi_{00} & \pi_{01} & \pi_{02} & \pi_{03} & \pi_{10} & \pi_{11} & \pi_{12} & \pi_{13} \end{array} \\
\begin{pmatrix}
0 & 1-\lambda(0)\Delta t & 0 & 0 & \lambda(0)\Delta t & 0 & 0 & 0 \\
0 & 0 & 1-\lambda(\Delta t)\Delta t & 0 & \lambda(\Delta t)\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 1-\lambda(2\Delta t)\Delta t & \lambda(2\Delta t)\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mu(0)\Delta t & 0 & 0 & 0 & 0 & 1-\mu(0)\Delta t & 0 & 0 \\
\mu(\Delta t)\Delta t & 0 & 0 & 0 & 0 & 0 & 1-\mu(\Delta t)\Delta t & 0 \\
\mu(2\Delta t)\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 1-\mu(2\Delta t)\Delta t \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}.
$$

If the DTMC is explicitly created and intended to be solved using one of the Markov chains' solution methods, then a decision on the maximum age intensity τ_{max} must be made in advance, and the whole Markov chain needs to be stored. The Markov chain approach would also force the state changes bounded by τ_{max} to happen when this point is reached, which is an unrealistic assumption. The proxel-based method avoids this condition by adapting the length of τ_{max} as needed; when probabilities for states at a new discrete time step are generated whose values fall below a certain threshold, then these are simply ignored. If that is not the case, the proxel simulator continues to prolong the enabling times of transitions which are still producing non-negligible probabilities.

The DTMCs for more complex models can be generated analogously. The number of states of the chain can grow exponentially in the number of concurrently enabled transitions - a well-known drawback of the supplementary variable approach. Another drawback when using pure homogeneous Markov chain approach instead of proxels, is that many of the features which make the proxel-based method flexible (such as allowing models to exhibit time-dependent behaviour, as shown in Section 4.1), are lost. If those are to be converted into Markov chains, then it would result into non-homogeneous Markov chains[3], which require precomputed values of all coefficients of the matrices and can require a large amount of memory for storing them (van Moorsel and Wolter 1998). The proxel based method, on the opposite, computes the coefficients dynamically, as needed, from the description of the model.

3.2 On the Accuracy of the Method

There are two major factors that affect the accuracy of the proxel-based simulation, as well as some minor ones too. Briefly described, the two sources of error are:

- the assumption that at most one state change within one time step happens, further referred to as the *basic assumption* of the proxel-based method, and

[3] A finite non-homogeneous Markov chain is defined by an infinite sequence $P_1, P_2, P_3, \ldots$ of $n \times n$ stochastic matrices, which define the transition probabilities that can be different at different time steps. The matrix P_i is the probability transition matrix for the i-th step. A homogeneous Markov chain with probability transition matrix P is the special case: $P, P, P, \ldots$, and that is the case that we observe throughout the thesis.

- the numerical integration of the instantaneous rate function, used for computing probabilities of state changes.

That is the subject that is treated in this section, or in other words: the accuracy of the proxel-based method and how it can be improved.

The first major factor follows from the fact that the proxel-based method is based on the assumption that the probability of more than one state change happening within one time step is negligible. As the size of the time step approaches zero, this becomes true, so theoretically the assumption is an acceptable one. This, however, means that the greater the size of the time step, the lower is the accuracy of the simulation. The relation of both is elaborated in this section.

In order to estimate the error that is produced by the assumption of allowing at most one state change within time of Δt, we compare the numerical solution obtained using the proxel-based method to the one obtained analytically, by solving a continuous-time Markov chain. In order to isolate this source of error we make it independent of the type of distribution functions, and we choose a model with exponential distributions for the comparison because of the simplicity of obtaining the analytical solution. The constant rates ensure that the second source of error, i.e. the IRF integration method, is neutralised.

Let us observe the continuous-time Markov chain shown in Figure 3.5. The model consists of two states, A and B, connected via two state changes, which have equal rates of λ.

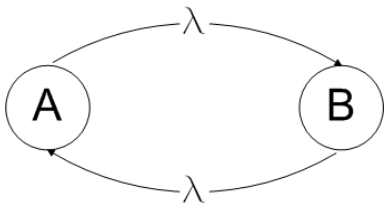


Figure 3.5: Example Markov chain with two states

We solve the balance equations to get the analytical solution of the model, and then compare it with the solutions obtained using the proxel-based method with different sizes of the time step.

We denote by $\pi_A(t)$ and $\pi_B(t)$ the transient solutions at time t of states A and B, correspondingly. The system of balance equations that characterises this model is the following:

$$\frac{d\pi_A}{dt} = -\lambda\pi_A + \lambda\pi_B = -\frac{d\pi_B}{dt}.$$

The analytical solution to this system of ODEs is then the following:

$$\pi_A(t) = 0.5 + 0.5e^{-2\lambda t}$$

$$\pi_B(t) = 0.5 - 0.5e^{-2\lambda t},$$

which given the assumed initial conditions

$$\pi_A(0) = 1.0$$
$$\pi_B(0) = 0.0,$$

expanded as a Taylor series has the following form:

$$\pi_A(t) = 1 - \lambda t + \lambda^2 t^2 - \frac{2}{3}\lambda^3 t^3 + \frac{1}{3}\lambda^4 t^4 - \frac{2}{15}\lambda^5 t^5 + \dots$$
$$\pi_B(t) = \lambda t - \lambda^2 t^2 + \frac{2}{3}\lambda^3 t^3 - \frac{1}{3}\lambda^4 t^4 + \frac{2}{15}\lambda^5 t^5 + \dots.$$

The approximation that the proxel based method is making when using the starting point for integration, as shown in Equation (3.1), produces an error term for each step of order Δt^2. This in turn makes the proxel-based method a first-order method for a sufficiently small Δt. The solutions which the proxel-based method produces for $t = \Delta t$, knowing that the IRF is constantly λ are the following:

$$\pi_A(\Delta t) = 1 - \lambda \Delta t$$
$$\pi_B(\Delta t) = \lambda \Delta t,$$

which are exactly the first-order terms of the Taylor expansion.

Let us observe the proxel tree of the Markov chain shown in Figure 3.6, using two different values for the size of the discretisation time step, the second one twice as small as the first one. Note that because of the Markov property of memoryless, there is no need for an age variable. Therefore, the form of the proxel in the proxel-tree shown in Figure 3.6 is the following: (*discrete state, probability*). The value of the instantaneous rate function for the exponential distribution is constant and in this case it is λ. Therefore, the size of the time step has to be smaller then $\frac{1}{\lambda}$. For our example, we choose the following sizes of the time step: $\Delta t_1 = \frac{1}{2\lambda}$ and $\Delta t_2 = \frac{1}{4\lambda}$.

The solutions obtained using the proxel-based method and the analytical ones are illustrated in Figure 3.7, whereby the analytical solution is mapped to the time step $\Delta t = 0.0$. In the figure, the proxel-generated solutions converge towards the analytical one as $\Delta t \to 0$. This feature (the convergence of the solution values) of the proxel-based method makes it possible to extrapolate solutions obtained with larger time steps (correspondingly having shorter computation times), in order to get solutions of higher accuracy.

Our experience has shown that the use of *Richardson's extrapolation method* (Richardson 1927) for performing the extrapolation yields satisfiably accurate results, which

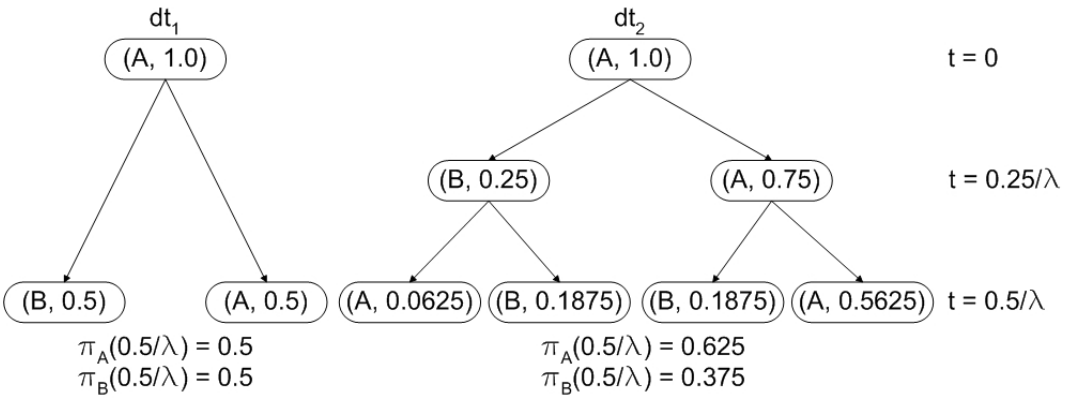


Figure 3.6: Proxel-trees produced using two different discretisation time steps

are very close to the analytical ones. The Richardson's extrapolation method is known as a method that by combining lower-order results yields new results of a higher order (Lambert 1973). In the case shown in Figure 3.7, the value obtained using the extrapolation of the four shown values is $\pi_A(0.25) = 0.68396123809524$, whereas the analytical solution is $\pi_A(0.25) = 0.6839$.

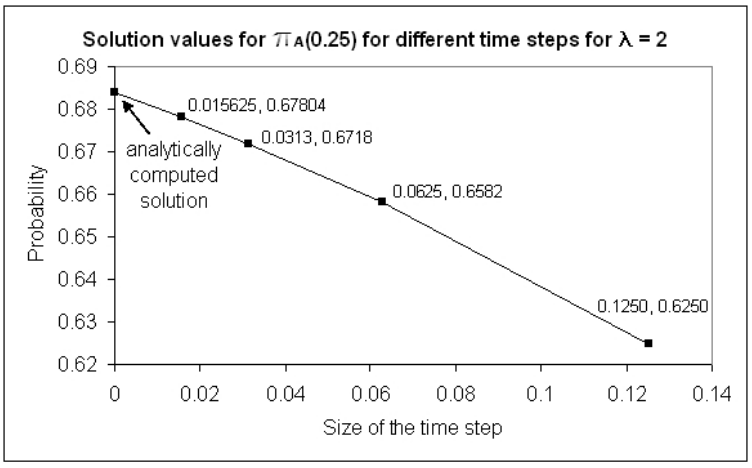


Figure 3.7: Solution values obtained using different time steps and the analytical solution for $\pi_A(\frac{1}{2\lambda})$ for $\lambda = 2$

To extrapolate the conclusions about the accuracy of the proxel-based method regarding solely the basic assumption and excluding the properties of the distribution functions, we can state that the proxel-based method is a first-order method because its local truncation error is $O(\Delta t^2)$, where Δt is the size of the time step. The reason for that is the assumption of having at most one state change within one time step, which means that the local truncation error that accumulates is of the order of the neglected probabilities. This is further concretised for the cases of Uniform and Weibull distributions.

The process of truncating probabilities when increasing the size of the time step, is illustrated in Figure 3.8. The dashed lines show the paths that are neglected as a consequence of the assumption; the corresponding discrete states are coloured grey. In the first case, when Δt is halved, there is only one situation that is ignored by the proxel-based method when using a time step of Δt, and that is the route (A, B, A). When Δt becomes even smaller, the number of those situations increases, but the order of the ignored probabilities is always at least $O(\Delta t^2)$, as already shown by the solutions of our example model from Figure 3.5.

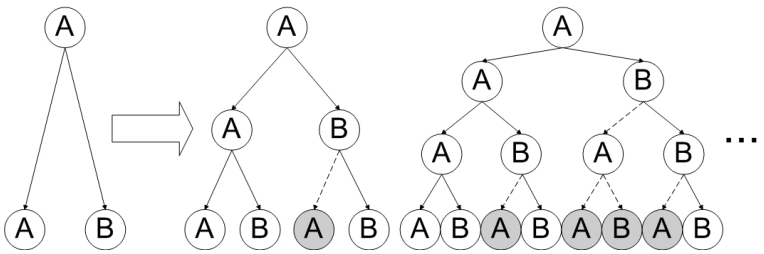


Figure 3.8: Illustration of the error produced by the basic assumption of the proxel-based method

Now let us observe explicitly the approximations that the proxel based method creates for the IRF of some of the other distributions when calculating the probabilities (the second major accuracy factor) and the error that it thereby makes. We inspect in more detail the cases when the simplest integration method is used, i.e.

$$\int_{\tau}^{\tau+\Delta t} \mu(x)dx \approx \mu(\tau) \times \Delta t,$$

which is known to be a first order integration method. In that case the extracted approximations for the Uniform and the Weibull distributions are the following:

$Uniform(a, b)$ The integration of the IRF of a Uniform distribution defined on the interval $[t, t + \Delta t]$, expanded as a Taylor series is the following:

$$\int_{t}^{t+\Delta t} \mu(x)dx = \frac{\Delta t}{b-t} - 1/2\,\frac{\Delta t^2}{(-b+t)(b-t)} + 1/3\,\frac{\Delta t^3}{(-b+t)^2(b-t)} - \ldots$$

on the interval (a,b). Outside of that interval, the value of the IRF is zero.

The proxel-based method approximates it by the first term of the Taylor series, thereby making an error for each time step of order Δt^2.

If the state changes in the model shown in Figure 3.5 are uniformly distributed on (a, b), then the Taylor series of the state equations are the following:

$$\pi_A(t) = 1 - \frac{t}{b} + 1/2\,\frac{t^2}{b^2}$$

$$\pi_B(t) = \frac{t}{b} + 1/2\,\frac{t^2}{b^2}.$$

The solutions obtained using the proxel-based method for $t = \Delta t$ are again the first order terms, meaning

$$\pi_A(\Delta t) = 1 - \frac{\Delta t}{b}$$

$$\pi_B(\Delta t) = \frac{\Delta t}{b},$$

that is if the points in time we have observed belong to the interval (a, b).

Weibull(a, b) The integration of the IRF of a Weibull distribution on the interval $[t, t + \Delta t]$, expanded as a Taylor series is the following:

$$\begin{aligned}\int_t^{t+\Delta t} \mu(x)dx = &\left(\frac{t}{a}\right)^b b\Delta t t^{-1} \\ &+ 1/2\left(\frac{t}{a}\right)^b b\,(b-1)\,\Delta t^2 t^{-2} \\ &+ 1/6\left(\frac{t}{a}\right)^b b\,(b-1)\,(b-2)\,\Delta t^3 t^{-3} + \ldots\end{aligned}$$

which the proxel-based method approximates to the first term of the Taylor series, making an error for each time step of order Δt^2.

If the state changes in the model shown in Figure 3.5 are distributed according to a Weibull distribution function with $b > 1$, then the Taylor series of the state equations contain only higher than first-order non-constant terms, and the constant terms are exactly the approximations that the proxel-based method makes, thereby producing an error that is of order higher than Δt^2.

The described approximations use the simplest integration method for the approximation of the IRF, which is usually not the case in our implementations. There, we often use, as further described, the midpoint, trapezoid, Simpson's rule or Gauss quadrature (Hoffman 1992), which yield smaller error and a better approximation of the integration of the instantaneous rate function.

The numerical approach used for integration of the instantaneous rate function is the second major factor that affects the accuracy of the results. As mentioned, there are four major integration approaches that are being used, which in the following are presented along with their local errors:

- Mid-point rule, which approximates the function using the middle point and has local error of $O(\Delta t^3)$, i.e.

$$\int_\tau^{\tau+\Delta t} \mu(x)dx \approx \mu\left(\tau + \frac{\Delta t}{2}\right) \times \Delta t,$$

- Trapezoid rule, which approximates the function using the two end points and has local error of $O(\Delta t^3)$, which is same as the Mid-point, i.e.

$$\int_{\tau}^{\tau+\Delta t} \mu(x)dx \approx (\mu(\tau) + \mu(\tau + \Delta t)) \times \frac{\Delta t}{2},$$

- Simpson's rule, which works by increasing the accuracy of the functions used to approximate the integrand, uses three points, and has local error of $O(\Delta t^5)$ i.e.

$$\int_{\tau}^{\tau+\Delta t} \mu(x)dx \approx \left(\mu\left(\tau\right) + 4\mu\left(\tau + \frac{\Delta t}{2}\right) + \mu\left(\tau + \Delta t\right)\right) \times \frac{\Delta t}{6}, \text{ and}$$

- Gauss quadrature, which uses a careful selection of the points at which the function is evaluated, uses two points, and has local error of $O(\Delta t^5)$, which is the same as the one of the Simpson's rule i.e.

$$\int_{\tau}^{\tau+\Delta t} \mu(x)dx \approx \left(\mu\left(\tau + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)\Delta t\right) + \mu\left(\tau + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)\Delta t\right)\right) \times \frac{\Delta t}{2}.$$

The graphical representations of the four approaches are shown in Figure 3.9. The original function is $f(x)$, and $f^*(x)$ is its approximation. The shaded area is the approximated integral.

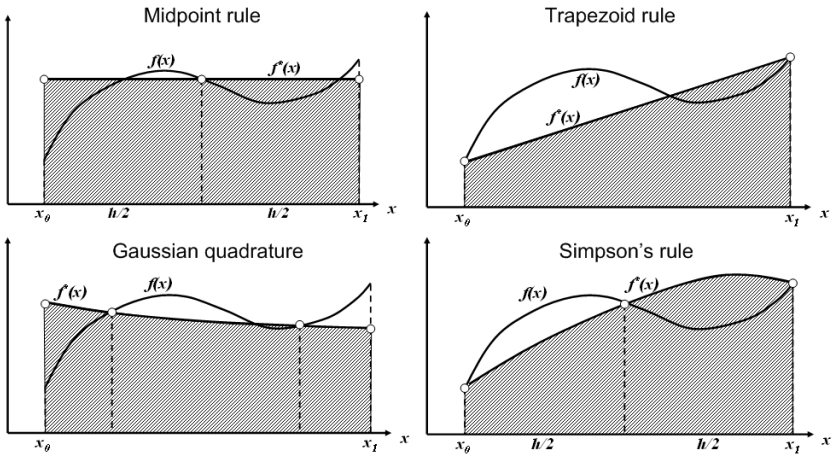


Figure 3.9: Illustration of the four different integration approaches

The results of one comparison of the first three integration approaches are shown in Figure 3.10. The Gauss quadrature has a comparable accuracy to the Simpson's rule and therefore is not presented in the figure. The difference is in some cases not meaningful, but when high accuracy is required then it is not neglectful. The model that is analysed consists of two states which are connected via two uniformly

distributed state changes on a range $(1, 2)$. The size of the time step is $\Delta t = 0.5$ and it was chosen to be large for easier spotting of the differences. It is to be observed that not all integration approaches handle the sharp edges of the Uniform distribution in the same way. It depends a lot on the points which are being used for approximating the integral, and as explained further in Section 4.1.4, require a special treatment.

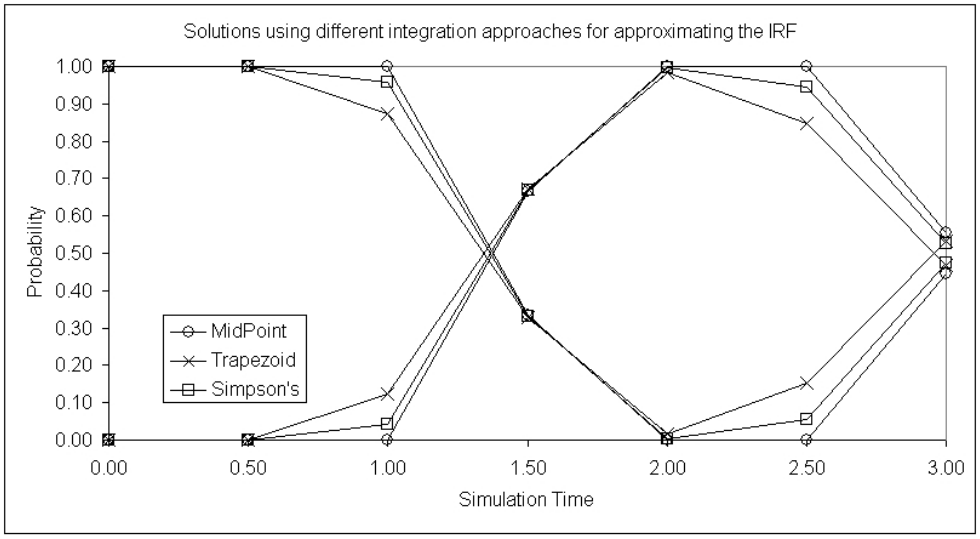


Figure 3.10: Illustration of the three different integration approaches when using a large time step

When deciding which integration method to choose, firstly it depends on the accuracy that we want to achieve, and secondly on the functions that we are trying to approximate. The advantage of Gauss quadrature over Simpson's rule and of mid-point over trapezoid is that they need fewer function evaluations for achieving the same accuracy, which makes them usually the methods of choice.

One minor factor which affects the accuracy of the simulation is the neglecting of proxels with probabilities less than a predefined threshold i.e. the value of the threshold. The method operates by throwing away proxels with probabilities which are less than the predefined threshold (typically between 10^{-12} and 10^{-15}). This, however, leads to probability leaking out, which means that at later points in simulation the sum of all probabilities of the proxels from one level is not equal to 1.0 any more. The error can be decreased by decreasing the value of the threshold, taking into account the distribution functions and rareness of the events.

Experiments regarding the minimum probability threshold as an error factor are presented in the experiments' Chapter 7. Its influence, however, is meaningful only when states with very small probabilities get involved, such as occur in rare-event models.

In conclusion, the proxel-based method can be considered as a first-order method in the worst case, i.e. its accuracy gets improved by using more accurate integration methods and depends also on the distribution functions involved in the models.

3.3 Supplementary Algorithms

In this section we discuss two supplementary steps, which aid the proxel-based simulation. The first one shows how to automate the determination of the size of the time step, whereas the second one shows an algorithm for computation of the lifetimes of the discrete states, which makes the computational complexity predictable and assists the storage strategy.

3.3.1 Heuristics for Determination of the Size of Δt

One important issue when performing proxel-based analysis of discrete stochastic models is the size of the discretisation time step Δt. Although for small models it is uncomplicated and straightforward to do it manually, the task can become very cumbersome when dealing with more complex models. Therefore, here we generalise the approach that we have been using for determining the size of Δt and provide its formalisation.

We are interested in calculating the largest acceptable size that the time step can have in order for the method to deliver reasonable solutions, even though with a lower accuracy. A *reasonable* solution is a solution, which together with the solution obtained for the halved step size, converges towards the exact one. To provide such a solution the time step should be able capture all of the state changes in the model, which means not skipping any of the points where they are most likely to happen. Such points are the ones where the value of the instantaneous rate function is the highest or mean values of the distributions. If a size of the time step satisfies this condition, then it is referred to as an *acceptable* size of the time step.

An example for an *unacceptable* step-size is $\Delta t \geqslant b$ if the state change is distributed according to a Uniform distribution function on (a, b), or $\Delta t \geqslant 1/\lambda$ if the state change is distributed according to an Exponential distribution function with a rate of λ. For illustration, in Figure 3.11, the sizes of the time step that are greater than 1.0 are unacceptable step-sizes because the solution values do not get closer to the "true solution" with the decrease of the size of the time step. The clear idea of what an unacceptable time step is, provided the key to what an acceptable size of a time step is and the approach that we present here.

As expected, the size of Δt depends directly on the distribution functions involved in the model and their parameters. We define a measure called "relatively acceptable time step" (RATS) for every state change, and define it for every type of distribution function $F()$ as the half of its mean value. For most of the distributions, it is sufficient that the time step is less than the mean value, the half of the mean only ensures that the size of the time step is an acceptable one. Its values for some of the more characteristic probability distributions are the following:

- if a state change SC is distributed according to $F() \sim Uniform(a, b)$, then $RATS(SC) = \frac{b+a}{4}$,

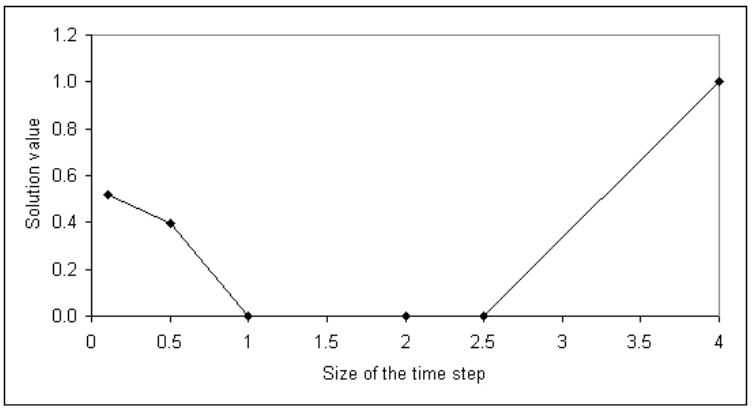


Figure 3.11: Illustration of non-convergent solution values

- if a state change SC is distributed according to $F() \sim Exponential(\lambda)$, then $RATS(SC) = \frac{1}{2\lambda}$,
- if a state change SC is distributed according to $F() \sim Normal(\mu, \sigma)$, then $RATS(SC) = \frac{\mu}{2}$,
- if a state change SC is distributed according to $F() \sim Weibull(\alpha, \beta)$, then $RATS(SC) = \frac{\alpha}{2} \times \Gamma(\frac{1}{\beta} + 1)$, where $\Gamma()$ is the Gamma function[4].

In other words, we chose them to be half of the mean values of the corresponding random variables. The decision is experience-based, constructed as a conclusion from the many experiments that we have carried out.

Once the *RATSs* for all state changes are computed, the minimum of all of them is calculated and that provides the "globally acceptable time step" (*GATS*) for the proxel-based simulation of the model. Formally expressed, if M represents a discrete stochastic model, then the global acceptable time step for M is defined as:

$$GATS(M) = \min_{\forall SC \in M} RATS(SC). \tag{3.7}$$

The smaller the size of the time step is, the higher the accuracy of the solutions is. Therefore, the acceptable size of the time step is just an acceptable one. Further, depending on whether the accuracy or the speed is important (or both), the user can choose to use either extrapolation or a very small step size for improving the accuracy.

[4]The Gamma function was first introduced by the Swiss mathematician Leonhard Euler (1707-1783) in his goal to generalize the factorial to non integer values. It belongs to the category of the special transcendental functions and is defined by the following integral: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. There are many numerical approaches for its approximation, some of the more popular being Lanczos Approximation and Stirling's Formula.

3.3.2 Computation of Lifetimes of Discrete States

Prior to the proxel-based analysis, if the analysis is to be carried out on a bounded state space (i.e. the model has a limited number of discrete states), then a preprocessing step can be carried out for computing the lifetimes of the discrete states. The computed lifetimes provide a preview of the computational complexity and the memory requirements of the concrete proxel-based analysis, and are used for computing the keys of the proxels in the binary tree. A unique key is assigned to every proxel, which is computed based on the state that the proxel represents, i.e. the combination of the discrete state and the age intensity vector. Therefore, it is necessary to be able to predict the largest value that each age intensity can have.

When finite support distributions are associated with the state changes in a model, then it is predictable that the model can spend a limited amount of time in each discrete state. When the state changes are distributed according to infinite support distributions, then theoretically the model can spend an infinite amount of time in each discrete state. The probabilities for staying in the discrete states, however, decrease as time increases. At a certain point in time, they become so small that we can treat them as negligible. We decide that the probabilities for staying in the discrete states are small enough when they become smaller than the predefined minimum probability threshold ϵ, which is usually around 10^{-15}. This makes it possible to determine simulation characteristic lifetimes of the discrete states.

A *lifetime* of a discrete state determines the longest time that the model can spend in that discrete state regarding the concrete simulation parameters ϵ, and is calculated based on the distribution functions that are associated with the active state changes in the actual discrete state. The procedure for the calculating the lifetimes is described in the Algorithm 3.2. Recall that the IRF could be better approximated using any of the four integration approaches, as described in Section 3.2. In the algorithm presented here, for simplicity reasons we use the most simple approximation, which is based on the starting point of each interval.

The symbols used in the algorithm have the following meanings:

- t_{max} is the maximum simulation time,
- Δt is the size of the time step,
- $prob_{exit}(DS)$ is the total probability for exiting the discrete state DS,
- $prob_{exit}(DS, SC)$ is the probability for exiting the discrete state DS through the state change SC,
- $prob_{stay}(DS)$ is the probability for not leaving the discrete state DS,
- $lifetime(DS)$ is the lifetime of the discrete state DS, and
- $\mu_{SC}(t)$ is the value of the instantaneous rate function of the random variable that describes the state change SC, having an age intensity equal to t.

Algorithm 3.2: Computing Lifetimes of Discrete States

Input: $\Delta t, t_{max}$

1 **foreach** *discrete state DS in the model M* **do**
2 $\quad t = 0;$
3 $\quad lifetime(DS) = 0;$
4 $\quad prob_{exit}(DS) = 0;$
5 $\quad prob_{stay}(DS) = 1.0;$
6 $\quad$ **for** $t = 0$ **to** t_{max} *in steps of* Δt **do**
7 $\quad\quad$ **foreach** *active state change SC in DS* **do**
8 $\quad\quad\quad prob_{exit}(DS, SC) = \mu_{SC}(t) \times \Delta t;$
9 $\quad\quad\quad prob_{exit}(DS) = prob_{exit}(DS) + prob_{exit}(DS, SC);$
10 $\quad\quad$ **end**
11 $\quad\quad prob_{stay}(DS) = prob_{stay}(DS) \times (1.0 - prob_{exit}(DS));$
12 $\quad\quad$ **if** $prob_{stay}(DS) < \epsilon$ **then**
13 $\quad\quad\quad lifetime(DS) = t;$
14 $\quad\quad$ **end**
15 $\quad$ **end**
16 $\quad$ **if** $lifetime(DS) = 0$ **then**
17 $\quad\quad lifetime(DS) = t_{max};$
18 $\quad$ **end**
19 **end**

The algorithm works by calculating the probabilities for leaving the states through any of the state changes at every discrete time step (lines 8 and 9) and thereby the probability for not leaving the discrete state (line 11), until the end of simulation time t_{max} is reached (line 6). If the probability for not leaving the discrete state is less than ϵ (line 12), that means that at that point in time there is a probability of zero or negligible probability to stay in the same discrete state. Therefore the point in time at which that happens is assigned as the lifetime of the actual discrete state (line 13). If that point in time, where the probability for staying in the same discrete state is negligible, is not reached within the maximum simulation time t_{max}, then the maximum simulation time t_{max} is assigned as the lifetime of the actual state for the actual proxel-based simulation (lines 16 and 17).

The lifetimes of the discrete states in a model directly influence and determine the complexity of the simulation. The reason for that is that they are a factor that determines the real state space of the model in terms of proxels, thereby considering only the truly reachable states. If the model has no more than one concurrently active state changes with different activation times, then we propose considering the sum of the lifetimes of all states in a model as a measure for the complexity of that model's proxel-based analysis. In cases where this condition does not hold, the number of supplementary variables in each state has to be considered too, as it increases the number of their possible combinations when generating states i.e. proxels. The presence of age memory state changes is a typical case for having more than one active state changes

with different activation times. In that case we define a *maximum number of states generated from a discrete state* as follows.

Definition 3.12 – Maximum number of states generated from a discrete state (MNSDS). *Maximum number of states generated from a discrete state* (MNSDS) *is the maximum number of combinations of the state vector, considering the lifetimes of the discrete states.*

MNSDS is equal to the ratio of the lifetime of the discrete state and Δt when there are no more than one active state changes with different activation times. If that is not the case (i.e. when there are age memory state changes), MNSDS is calculated as a product of the lifetime of the actual discrete state and the lifetime(s) of the state(s) in which the age memory state change(s) is (are) active. MNSDS is directly used for calculating the key of each proxel.

The effect that lifetimes have on the computational complexity of the proxel-based simulation is supported and illustrated with concrete examples in the experiments' Section 7.2.

3.4 Programming Aspects and Complexity

The basic implementation of the proxel-based algorithm is based on two interchangeable data structures, which store proxels from two sequential time steps. After many experiments (Balaprakash 2004), binary tree was the data structure of choice. We needed an easily searchable data structure because of the multiple generation of the same states within one generation of proxels, which as time advances, happens more often. For speeding up the searches, a balanced binary tree (AVL tree) was implemented (Foster 1965; Lazarova-Molnar and Horton 2003a), which showed as more time consuming than the non-balanced version because the balancing required a large portion of the computation time. Results from an experiment regarding the comparison are shown in Figure 3.12, from which it could be observe that the AVL tree implementation requires more than twice the time that the non-balanced tree implementation needs. Therefore, our decision was to continue using the non-balanced binary tree as a data structure for storing proxels.

A unique key for each state is computed based on the information contained in the state, i.e. discrete state and age intensity vector. In order to optimise the key, lifetimes of the discrete states are computed as a preprocessing step, which are basically the longest times that the model can reside in each of the discrete states before a state change happens. When the model contains no age memory state changes, the lifetimes directly generate the upper bounds of the age intensities in each discrete state. When that is not the case, the upper bounds are computed by multiplying the lifetime of each state that tracks the age memory state change with the lifetime of the discrete state in which the age memory state change is active, that is the MNSDS for every discrete state. The presence of age memory state changes can significantly increase the complexity of the

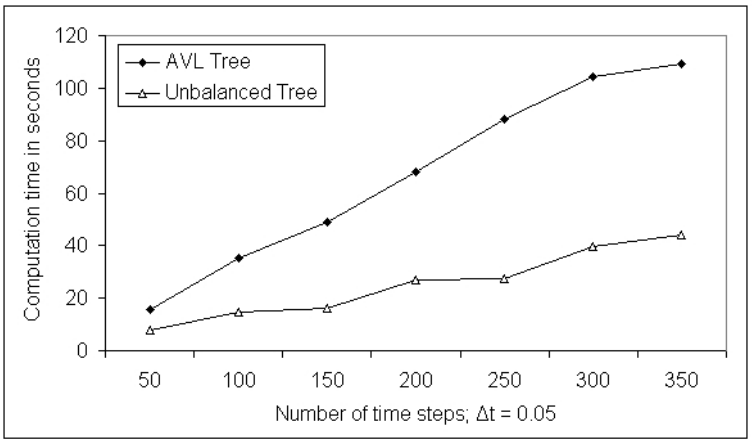


Figure 3.12: Dependence of the computation time on the number of time steps for the two storage cases

simulation, as it yields a greater number of truly reachable states, because the states are result of the combinations of the age intensities. This is illustrated by an example in Section 7.2, which demonstrates the effect that the lifetimes have on the computational complexity of the proxel-based simulation.

Although the first impression is that the method has a polynomial complexity in the order of the branching factor of the proxel tree, that is not true. Usually, depending on the type of the distribution functions and their parameters, there is a point in time, from where on, the number of proxels generated at each level does not change. It stays constant because of the limited lifetimes of the discrete states, and at that point in time all truly reachable states are generated. This shows, that although it seems that the proxel tree grows exponentially, it actually does so only up to some point in time (dependent on the distribution functions) and from there on grows linearly.

In order to practically illustrate the computational complexity of the proxel-based simulation as a function of the size of the time step, we choose again the model from Figure 3.5, thus setting both state changes to be distributed uniformly on (0.0,5.0), and we simulate initially up to time $t = 5.0$. The lifetimes of both discrete states are equal to the maximum simulation time, which is on purpose so that we can observe the complexity before the proxel-tree stops growing in width. In Figure 3.13 is illustrated the dependence of the total number of proxels generated and computation time on the size of the time step Δt. The figure confirms the quadratic growth, which is the case until the maximum number of reachable states is reached. From there on, the dependence is linear, as illustrated in Figure 3.14, in which case the simulation is run beyond time $t = 5.0$.

An advantageous feature of the proxel-based method is the dynamic building of the state-space, thereby rationally using the memory. Even if the underlying stochastic process is a Markov chain, it is not completely in the memory from the beginning of the simulation, but instead it is dynamically built and stored.

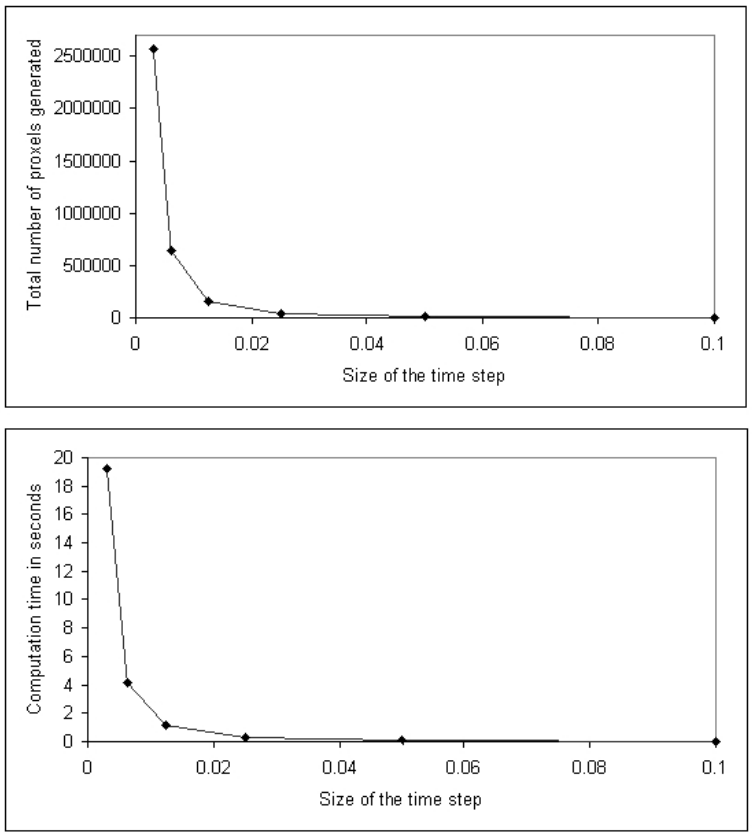


Figure 3.13: Dependence of the total number of proxels generated and computation time on the size of the time step Δt

3.5 Summary

In this chapter we establish and present the formal background of the proxel-based method, focusing on its basic algorithm. We show what are the supplementary pre-processing steps, which can aid the proxel-based simulation, such as computing the lifetimes of the discrete states and the heuristics for determining an acceptable size of the time step. Further we discuss the method's accuracy, after which, in conclusion, the programming and implementation aspects are addressed. In the upcoming chapters we discuss some special issues of the method, describe the modelling framework which was designed to exploit its advantageous properties, as well present some of the applications of the proxel-based method.

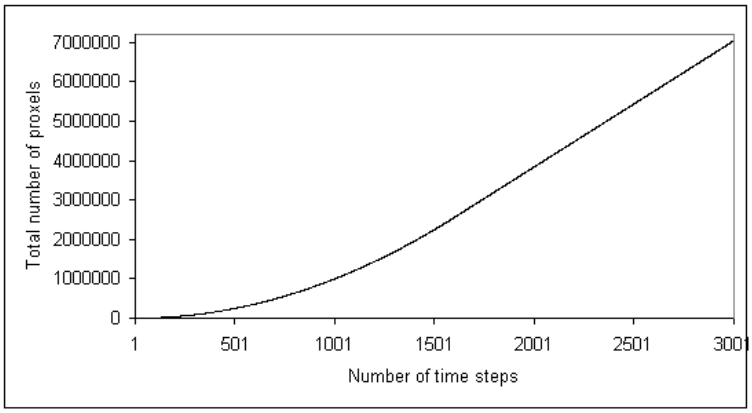


Figure 3.14: Dependence of the total number of proxels generated on the number of the time steps in the general case, i.e. before and after the point in time when the maximum number of states has been generated

4 Variants and Special Topics

In this chapter we address two main topics which we consider to be of especial importance when considering using the proxel-based method. The first section reviews different specific or characteristic classes of stochastic models which were found to be exceptional in different manners for the method and discusses them. The second section introduces two basic ideas for variations of the proxel-based method which were discovered while searching for improvements of the basic algorithm, along with a discussion of their advantages and drawbacks.

4.1 Special Classes of Problems

In this section we discuss and analyse the features and classes of stochastic models which were found to be from different aspects, characteristic for the proxel-based method, both in positive and negative sense. In other words, this section should also serve as an answer to the question whether it is a good idea to analyse a given model using the proxel-based method, and which are the issues that one should be careful about and require to be handled in a special way.

4.1.1 Unbounded Models

Unbounded stochastic models have always been a burden when their analysis was to be performed by any of the existing numerical approaches (Markov chain-based). In order to solve this class of models numerically, the state-space has to be bounded, which basically means modifying the original model, additionally affecting the accuracy of the analysis. The reason for the bounding is because all existing numerical approaches, as shown in Figure 3.1 are based on a mathematical computational model, which is then solved using numerical solution algorithms. The numerical solution algorithms need to be operate on the computer, therefore all of their unbounded quantities need to be bounded because the computer can only deal with finite elements and data structures as computation time and memory have are limited. On the other hand we

know nothing about the solution of the models we are analysing, so the decision on how bound the unbounded quantities is not a trivial one.

Real-life processes do not always have the feature of having a limited discrete state-space, so unbounded models are quite common. The formalism of stochastic Petri nets is capable of successfully describing unbounded models, which is the reason why we use them for illustration of this important feature of the proxel-based simulation. The proxel-based method possesses the ability to operate on the Petri net directly, without having to create the reachability graph prior to the simulation. This means that there is no requirement for bounding the state-space for analysing one model.

We illustrate by an example what the above described feature means. The most common and at the same time simplest example for the above described situation is a queuing system which consists of an unbounded queue and one server. The Petri net for this model is shown in Figure 4.1. Place P_1 corresponds to the queue, whereas P_2 represents the server. Transitions T_1 and T_3 accordingly represent the arrivals of the customers and the completions of service. Transition T_2 represents the secondary event of a customer moving to service when the server is free (i.e. $|P_2| = 0$), a condition which is implemented by an inhibitor arc from P_2 to T_2.

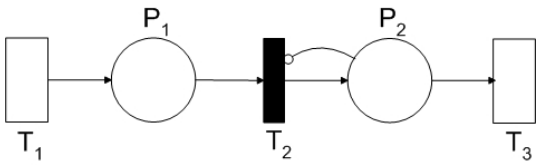


Figure 4.1: Petri net model of an unlimited queue with one server

Instead of building the reachability graph, the proxel-based method can operate directly on the Petri net, i.e. creating for each marking the set of reachable markings on-the-fly, based on the Petri net specifications. More specifically, reachable markings can be computed directly from the incidence matrix of the Petri net.

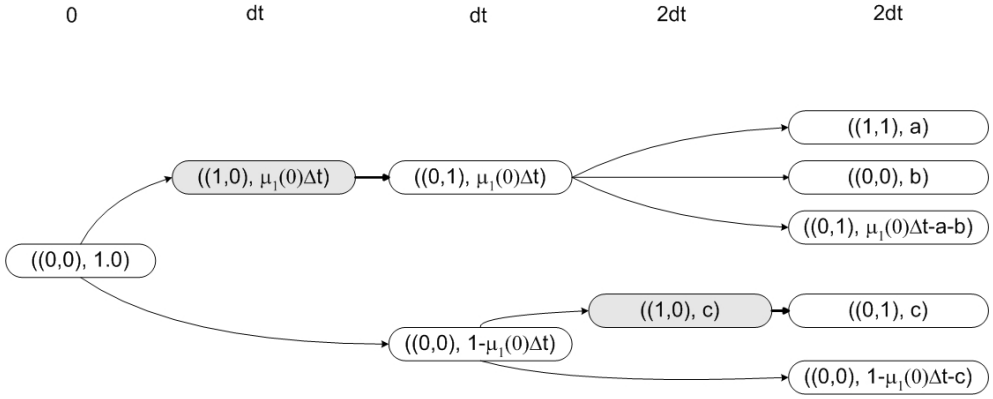


Figure 4.2: Proxel-based illustration of the initial part of the state space

In Figure 4.2, the initial levels of the corresponding proxel tree are shown, the age intensity variable is omitted for simplicity of the figure. The grey-coloured proxels represent vanishing markings, whereas the white ones stand for the tangible markings. The structure of the proxels in the figure contains only the marking i.e. the discrete state and the probability i.e. (m, p). Symbols a,b and c stand for the probabilities of the state changes, which can be calculated using any of the IRF integration formulas shown in Section 3.2.

The reachability graph is dynamically created as the proxel-based simulation advances in time, which is feasible because the proxel-based method operates directly on the user model, generating the state space on-the-fly and not requiring in advance generated reachability graph. This means that the proxel tree is generated solely based on the description of the Petri net. The on-the-fly creation of the state-space makes one important and problematic class of models analysable in a deterministic manner, i.e. the class of *unbounded discrete stochastic models*.

Further, in Chapter 5, we present a modelling framework which describes unbounded models in a way that the models thereby created can be used as a direct input to a proxel-based simulator. In Section 6.4 the application of the proxel-based method for analysing SPNs is presented.

4.1.2 Additional Supplementary Variables

As already defined in Chapter 3, the term *state* in the proxel-based terminology refers to the vector composed of the discrete state of the model (in Petri nets terminology - marking) and the age intensities of the active state changes in that discrete state. The additional components to the discrete state of the model, i.e. the age intensities, are known as supplementary variables.

One feature of the proxel-based method is that the supplementary variables can also be of discrete type and can be employed for tracking any quantity relevant for the analysis of the model. The additional variables can be also parameters of the instantaneous rate function that determines the probability for a state change happening. By contrast, if those types of models were to be analysed using DTMCs or PDEs, every such additional variable would add to the complexity of the system and the model could result in a very complicated system of equations which can be very expensive to solve.

An example for the described situation is a model of a machine failure where the failure is a function of the number of times that the machine has failed previously, a case which is also described in (Sule and Castro 2002). The number of failures does not need to be limited, as it was already explained in the previous subsection. The Petri net for this model is shown in Figure 4.3. It consists of two places OK (i.e. machine is operating) and F (i.e. machine has failed), and two transitions T_F (i.e. machine fails) and T_R (i.e. machine gets repaired). The number of failures is presented as a supplementary variable in the proxel.

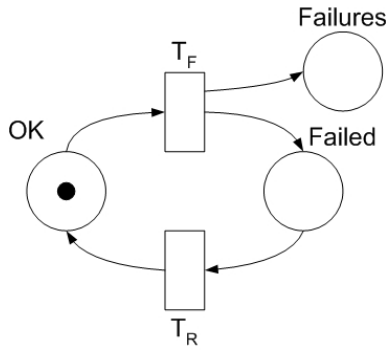


Figure 4.3: Petri net representation of the machine failure model

The Petri net for this model contains a place that tracks the number of failures. A typical deterministic approach, which operates on the reachability graph, would imply bounds on both, the number of failures and the age intensities. The proxel-based method is free of those limitations and the bounds are computed dynamically, based on the probabilities for the state changes happening. When the probability for a state change becomes negligible, i.e. falls bellow a certain threshold (in the ideal case equal to zero), that means that the "real" boundary of that parameter has been reached. A proxel Px for this example would have the following format:

$$Px = ((Marking, AgeIntensity), |Failures|, Time, Route, Probability).$$

By this feature of the proxel-based method of using the supplementary variables for tracking other quantities besides age intensities, we can use the method for a more efficient analysis of discrete stochastic models that contain unbounded elements which have impact on the distribution functions.

4.1.3 Rare-Event Models

Rare-event models are known to be very expensive when performing standard discrete-event simulation (sometimes in orders of months and years, as stated in (Görg et al. 2001)). As their name reveals, rare-event models contain events which happen very rarely with respect to other events in the model, i.e. there is a big difference in the rates at which the events in the model occur. As discrete-event simulation works by reproducing the behaviour of the models, it usually takes a large number of simulation runs for such rare events to happen, making it very difficult to obtain a reasonable estimate for the searched parameters, usually in form of narrow confidence intervals.

In Figure 4.4 the CDFs of two exponential distributions are illustrated which have different rates. From the figure it is evident that the probability of the event with the higher rate happening is, at any point in time, higher than the one of the event with a lower rate. In this case the difference ratio of the rates is 10. This means that on average on every 10 events described by the faster rate, one slower event happens.

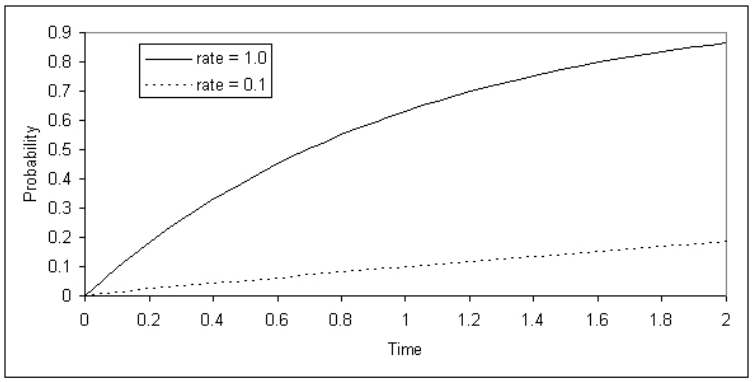


Figure 4.4: CDFs of two exponential distributions with different rates

The problem that discrete-event simulation has, is an issue that the proxel-based method is less sensitive to. By design the proxel-based method explores all possible behaviours of one model and reaches every possible state, which shows that the it gives equal importance to all events in the model. Therefore, even a state that is to be reached by the most rare event in the model, as early as in the first time step, obtains its very small probability contained in the proxel that describes it. DES, on the contrary, relies on values of random numbers to explore the behaviour of the model, which are not a guarantee that all states will be visited.

However, rare-event models bring some disadvantages for the proxel-based method too. The presence of rare-events in a model means big differences in the rates of the events. When using our heuristics for deciding on the size of Δt presented in Section 3.3.1, we choose a *globally acceptable time step*. This means that the proxel-based simulation is currently forced to move at equally sized time steps, which are minimum of the relatively acceptable ones for all state changes. Consequently, the state changes associated with rare events are tracked by using such small time steps, even though it is not necessary. This observation led to the idea of having adaptive time steps, which is a research in progress and is treated in (Wickborn 2004; Horton and Lazarova-Molnar 2003).

The main thought behind the *adaptive time steps* is to allow the simulation to advance at different pace for all state changes depending on their distribution functions. In an example model from Figure 4.5, in which the state change from A to C is associated with a rare event relatively to the other state changes, the structure of the proxel tree which uses adaptive time step would look similarly to the one in Figure 4.6. The main goal of the idea of having adaptive time steps is improving the efficiency of the proxel-based simulation by reducing the number of processed proxels.

The ability to accurately analyse rare-event models, we consider to be a great advantage of the proxel-based method, because the problem of the rare-event simulation has been around for a long time, and there has been a number of methods that try to overcome it (Heidelberger 1995; Glynn and Iglehart 1989; Hsieh 2002; Kelling 1996). Proxel-based method makes an important contribution with that respect. Example regarding

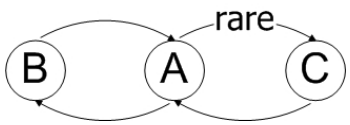


Figure 4.5: Example rare-event model

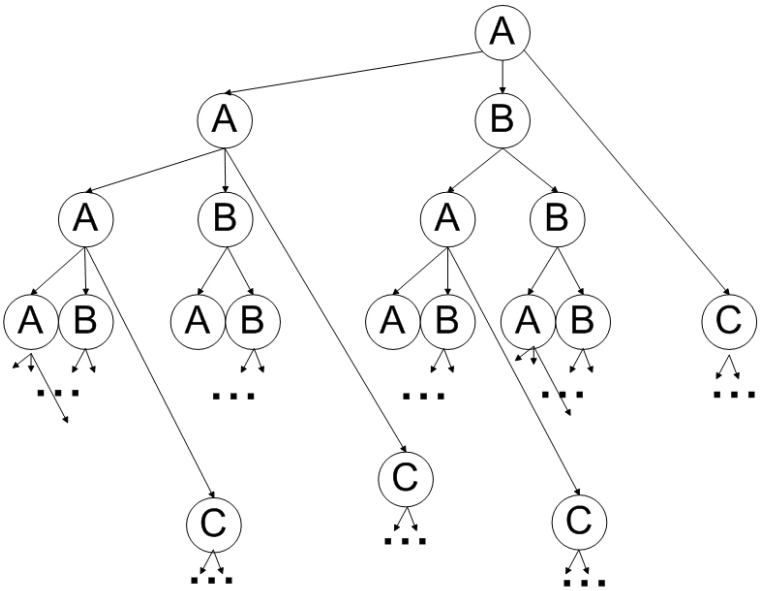


Figure 4.6: Illustration of the idea of having adaptive time steps

analysis of rare-event models is presented in Section 6.6, where it is discussed and treated as one of the application areas of the proxel-based method.

4.1.4 Instantaneous Rate Functions with Poles

Poles are points in which the value of a function goes to infinity. An example for a function that has a pole is the IRF of the Uniform distribution. The poles in the instantaneous rate functions are problematic for the proxel-based method and require a special treatment because the IRF needs to be integrated in steps of Δt and one of these steps contains the pole, as shown in Figure 4.7 where $\Delta t = 0.4$.

In Figure 4.7, the IRF of the Uniform distribution defined on (0.5, 1.5) is illustrated as an example. It is very important to properly handle those situations because that signifies that the state change that is associated with a such distribution has to happen by the pole point at latest, and if there are other competing state changes at the same time interval, then the probabilities for their happening have to be set to zeros, and the whole parent probability to be transferred to the state change associated with the IRF with a pole.

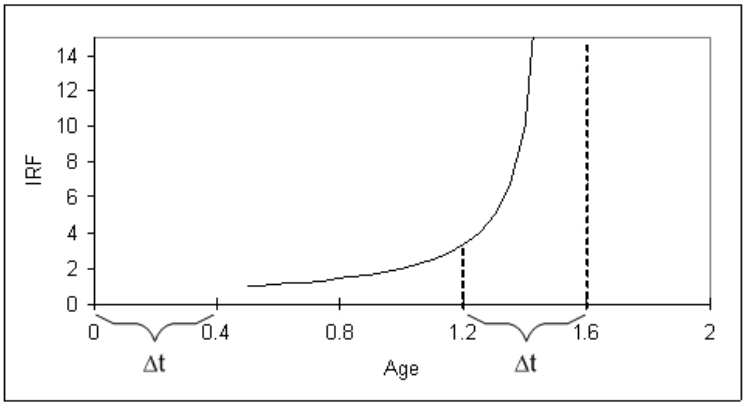


Figure 4.7: IRF of a Uniform distribution function on (0.5, 1.5)

A problem that occurs at the computation of probabilities at such points is that because of the infinity, the points approaching the pole have very high values, which means that there is a possibility that probability values greater than 1.0 are computed. This is a fact which helps in spotting those points and handling them properly. Therefore the implementations of the functions that we use in our programs are redefined to make sure that such points are captured. The most problematic distributions with that respect are Uniform and Deterministic, in which cases when evaluating the whole Δt neighbourhood needs to be examined and if the pole point is contained, an extremely high value based on Δt is yielded, which would immediately push the probability of staying in the discrete state bellow the minimum probability *threshold* (e.g. $\frac{1}{threshold \times \Delta t}$) and cancel it out.

The case is clear when only one state change of that type (with an IRF with a pole) is active. The situation is, however, more delicate when there are two or more concurrently active state change distributed according to finite support functions whose domains' upper boundaries are equal. An example for that is a state, in which there are two active state changes that have equal deterministic durations. When the simulation time reaches that point, there is a danger that the algorithm computes probability values for both state changes that are greater than 1.0 (because the rate goes to infinity, meaning they have to happen). Therefore, it is important to capture that point in time and properly handle it, which in the proxel-based simulation is achieved by distributing the parent-proxel's probability proportionally to their rates, amongst the active pending state changes. In the described case, our approach is to sum up the probabilities and calculate each probability by dividing the probability with the sum, ensuring that the probabilities of all state changes sums up to one. This means is that in the above described case both probabilities are equal, i.e. 0.5.

Even though poles require a specific treatment, they do not add to the complexity of the proxel-based method. Therefore, the goal of this section is their acknowledgement as a special case in the proxel-based simulation and provide an idea of how they are to be handled.

4.1.5 Finite Support Functions

Finite support distributions are distributions whose IRFs have finite support. Examples for such functions are again Uniform and Deterministic. The benefit from having state changes which follow finite support distribution functions comes from the proxel-based algorithm, which does not store proxels which carry zero probability values. Therefore the number of the proxels generated as a consequence of such a state change happening (i.e. one whose IRF has a finite support) is limited because the time range when such a state change can happen is limited too. In other words, the lifetimes of the corresponding discrete states are bounded. The finiteness ensures that the proxel-tree stops growing in width after some time, or once all possible states are generated. This is illustrated in Figure 4.8, which shows the structure of the proxel tree for a two state model with discrete states A and B where the lifetime of B is $2 \times \Delta t$ because the distribution function associated with the state change from B to A is Uniform defined on the interval $(0, 2.5 \times \Delta t)$. Once a path in the proxel tree reaches a point at which a discrete state must change, the branching in that direction stops, thereby reducing the growth of the tree.

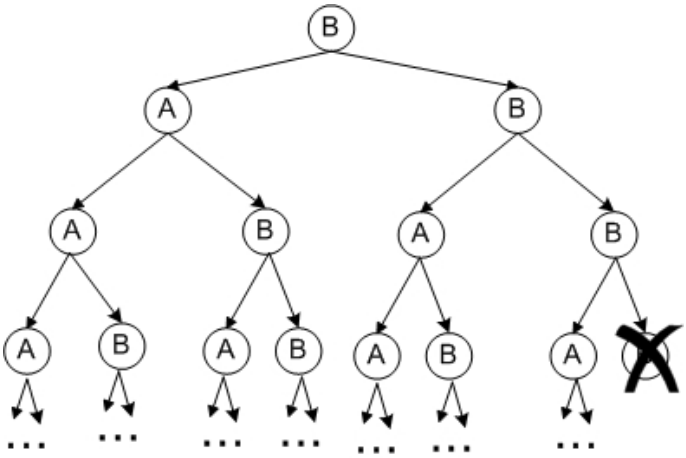


Figure 4.8: Structure of the proxel tree for a model with two states, A and B, where B has a lifetime of $2.5 \times \Delta t$

The presence of finite support functions simplifies the process of computing lifetimes of the discrete states in one model, described in Section 3.3.2, because its time range during which they can occur limits the lifetimes of the corresponding states. It is sufficient to have only one state change with a finite support IRF to limit the lifetime of the corresponding discrete state because that indicates the longest time that the model can reside in the actual discrete state. For distributions with finite support IRFs, no truncation is needed for determining the lifetime and a more accurate prediction of the complexity of the computation is possible. Therefore, we observe them as a favourable property of the models analysed by the proxel-based method.

4.1.6 Number of Concurrently Active State Changes

The lifetime of a discrete state depends solely on the rate at which the probability for leaving the state flows. This rate is depended on the concurrently active state changes associated with the discrete state. With every addition of a new state change to the model the rate at which the model leaves the corresponding discrete state increases. Therefore, the lifetime of the affected discrete state decreases which means reduction in the number of proxels that are stored and the computational complexity of the simulation because proxels which carry zero probability are not stored.

Hence, the increased number of concurrently active state changes is in general an indicator for a more efficient proxel-based simulation and therefore such models belong to the favoured classes of stochastic models. The statement, however, cannot be generalised to being more than just an indicator, as it depends on the points in time when those state changes become active.

The more concurrently active state changes there are that have different activation times (i.e. different age intensities at same points in time), the more combinations of the age intensities are possible, resulting into a larger state space. This issue is explained more in detail in Section 3.3.2. From that point of view, the problem is not trivial, and the presence of age memory state changes makes it only more complicated and increases significantly the computational complexity. The change of the complexity with respect to the memory policies of the state changes is shown in Section 7.2, where experiments regarding the lifetimes of the discrete states are performed.

4.2 Variants of the Proxel-Based Method

In this section we discuss two variants of proxel-based method which are currently being developed as attempts to improve its efficiency. The first one is the variant of stochastic proxels, which is a research in progress and still at a level of speculation. The second one is a promising variant, which combines phase-type approximations with proxels and has been able to improve the efficiency of the proxel-based method for certain classes of models.

4.2.1 Stochastic Proxels

The proxel-based method works by exploring and generating all possible paths that a model can take, when advancing in steps of Δt. What if there was a knowledgeable way of generating a subset of all paths which would be representative such that it would yield an approximation of the solution of the model?

This is the basic idea of the stochastic proxels, except that the “knowledgeable” is implemented by “stochastic”, i.e. the subset of paths is generated on a random basis. Each path has an equal probability to be generated, which ensures a “fair” treatment to all events.

The idea for the variation of the proxel-based method under the name of *stochastic proxels* was derived by comparing the standard proxel-based method and discrete-event simulation, as well as by observing the problems that the latter has with respect to rare events, and the state-space explosion of the former one. In this section we present the basic idea for this variant because it is still one of the speculative research areas that are yet to be studied.

Rare events are problematic because their probability of happening is so low that by reproducing the system, which is what discrete-event simulation does, they almost never happen, so it takes a high number of replications for them to occur. The proxel-based method on the other hand treats all events with equal importance, so the problem of neglecting an event is irrelevant.

The stochastic variant of the proxel-based method is an experimental approach, which works similarly to discrete-event simulation in that that it carries out many experiments which are then used for obtaining the necessary measures. It works by following one proxel route in each experiment, which route is chosen on a random basis. The difference to DES is that the routes are advancing in equally sized time steps (just like the standard proxels), which is compensated by the probability values. Each probability is calculated as shown in Chapter 3 for the standard proxel-based method. By this variant of the method we avoid the calculation of all paths, which is what the standard algorithm does, and only a random selection of all sample paths is generated.

The method of stochastic proxels uses a uniformly distributed random variable for deciding which state change to happen, which preserves the fairness towards the events, and avoids the state-space explosion problem because the memory that it requires is same as for discrete-event simulation, which is very low compared to the standard proxel-based method. The difference from the basic proxel-based method, as explained in Chapter 3 is that the stochastic variant computes only a subset of the paths that the model can take, selected on a uniformly random basis. This means that the random factor is present, as well as that results are again confidence intervals.

One problem that we faced whilst researching the stochastic variant develops from the fact that the further the point in time for which a solution is sought, the smaller the probabilities for each separate path are, resulting into a non-negligible error produced by the computer when registering them. This can however be avoided by scaling the probabilities at certain points in time and needs to be handled in advance.

In Figure 4.9, the proxel tree of the model shown in Figure 3.2, is shown. The route (A, A, B, B) is one possible path for the first four equidistant points in time, followed in one of the experiments of the stochastic proxels variant. The procedure for the generation of any random path is as follows:

1. The initial state of the model is A.
2. Using a uniformly distributed random variable, one of the two paths (1 or 2) is chosen; The probability is calculated in the same way as it is for the standard proxel-based method. If the models chooses to stay in A, then the age variable is increased by Δt, else it is set to zero.

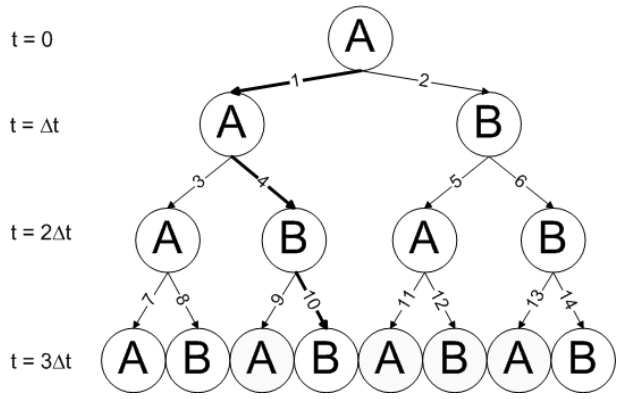


Figure 4.9: One path in the stochastic proxel variant

3. Step 2 is repeated until the end of simulation time t_{max}, at each step for the choice of possible state changes including the option of staying in the same discrete state.

Depending on the predefined number of replications, the above described procedure is repeated that many times. Next the transient probabilities for all discrete states at time $t = t_{max}$ are summed up. Each separate probability is calculated as the ratio of the sum of probabilities for that discrete state and the overall probability sum, i.e.

$$Pr(\text{model in discrete state } DS_i \text{ at time } t_{max}) = \frac{\sum_{\forall replications} Pr\{DS_i \text{ at time } t_{max}\}}{\sum_{\forall replications, \forall i} Pr\{DS_i \text{ at time } t_{max}\}}.$$

The stochastic variation suffers from the same problem of long computation times when high accuracy is needed because of it stochastic nature, as is the case with discrete-event simulation. It, however, also benefits from the same advantage as discrete-event simulation of having very low memory requirements, and additionally from the fair treatment of all events i.e. not neglecting the rare ones. We believe that the stochastic variant can be an efficient alternative for models that are too complex for the proxel-based method and contain rare events. The method, however, still needs to be further researched in order get more meaningful conclusions about its applicability.

4.2.2 Inclusion of Discrete Phases

As described in Section 2.2.3 phase-type distribution functions can be very efficient and accurate when approximating for distribution functions such as Weibull and Normal. On the other hand, they are very expensive to fit when approximating finite support distributions like Uniform and Deterministic which require large number of phases to be approximated more accurately.

Above described features of the discrete phase-type approximations yielded another variation of the proxel-based method, which is achieved by its combination with discrete phases, as described in (Isensee et al. 2005). The basis for the combination is

the fact that the underlying model of the proxel-based simulation can be described as a DTMC which is also what the discrete phases are. Therefore, the implementation of the discrete phases for substituting some of the distributions fitted well into the existing framework. An illustrative example is a model which consists of two state changes distributed according to a Weibull and a Uniform distributions correspondingly. The uniformly distributed state change is very expensive to approximate using discrete phase-type distribution functions and is therefore simulated with proxels. By contrast, in most cases, depending on the parameters a Weibull can be sufficiently well approximated by using only a few phases.

When combining proxels and phases, a new element is added to the proxel i.e. the phase (or phases) in which the model is, thus extending the definition of a proxel. The extended proxel structure is the following:

$$Proxel = (State, Time, Route, Probability), \text{ where}$$
$$State = (Discrete\ State, Age\ Intensity\ Vector, Phase\ Vector).$$

The *Phase Vector* contains information about the current phases of the model, in the same way as the *Age Intensity Vector* carries information about the relevant age intensities. The other elements have the same meanings as explained in Section 3.1.1.

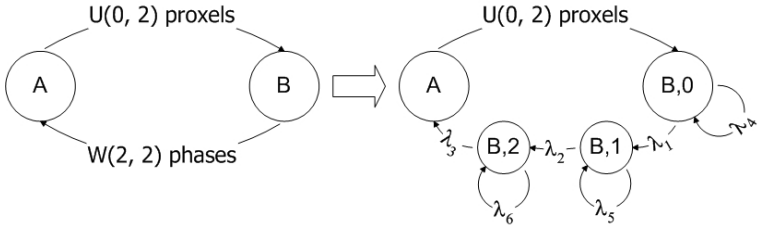


Figure 4.10: Inclusion of phases in a simple two state model

In Figure 4.10 an example model is shown for the purpose of illustrating the basic idea of the approach of combining discrete phases with proxels. The model consists of two discrete states, and two state changes, one of which is Weibully distributed and can be accurately fitted using discrete phases. In the same figure, the model development is shown after the inclusion of phases. The state change from B to A is approximated using three phases, whereas the state change from A to B is left unchanged because of the fact that proxels are more suitable for approximating Uniform distribution. When the model is in state A, the supplementary variable is the age intensity, and whenever the model is in state B, the supplementary variable denotes the phase of the discrete state B. The probabilities λ_i can be computed using one of the fitting algorithms described in (Isensee and Horton 2005a) and they do not require age variables because of the property of memorylessness. The state development for the first five time steps is shown in Figure 4.11, in which the variable supplementing the discrete state A denotes an age intensity, and the one supplementing the discrete state B denotes a phase.

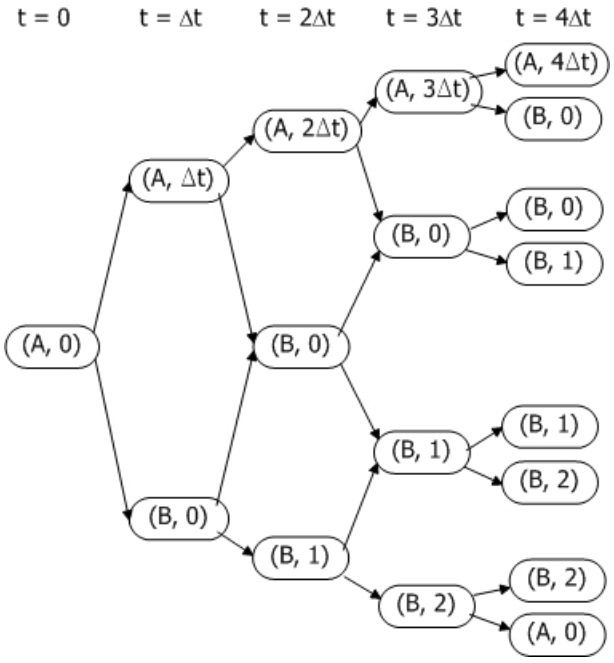


Figure 4.11: Proxel-phases state tree

The concept of phases fits well in the existing framework because the underlying processes of both approaches are DTMCs. This results in a straightforward implementation of the combined approach, for which recently a tool has been introduced (Isensee and Horton 2005b). The basic algorithm of this approach of combining phases and proxels is presented in (Isensee et al. 2005). There, we also present results from experiments, which show that by including phases in the proxel-based simulation, an improvement of factor ca. 2.5 is achieved regarding the simulation time. The computation of the approximation, however, can take a significant amount of time too. In the experiments we present in (Isensee et al. 2005) the combined phase-proxel simulation which yielded most acceptable accurate results took 22.390s (with $\Delta t = 0.05$), for which the distribution fitting took 28.719s, summing up to 51.109s. In the experiment two distributions are approximated by discrete phases, each of order 16, and two are simulated by proxels. The pure proxel-based simulation for this case took longer i.e. 56.421s. This shows that if a very high accuracy is needed, then the fitting can be computationally very expensive too. If, however, a less accurate fitting is satisfiable, then the computation time can be reduced significantly as explained further.

The total simulation and fitting times for the three cases, when using 4, 8 and 16 phases, are shown in Figure 4.12. The computation times grow rapidly as the number of phases increases, which is mostly due to the fitting times that can be computationally very expensive when the number of phases is large. The problem is that sometimes when a distribution which requires a large number of phases is approximated, it can require correspondingly a longer computation fitting time, meaning phases are not always the more efficient solution. Concerning that problem there is an ongoing research

for deciding when to use phases and when not, based on the characteristics of the distributions associated with the state changes and the lifetimes of the discrete states in the model. To conclude, the combination of phases and proxels creates and open door for the stochastic models that are too complex for the standard proxel-based method and represents an extension which would broaden the class of models that can be analysed using the proxels-based method.

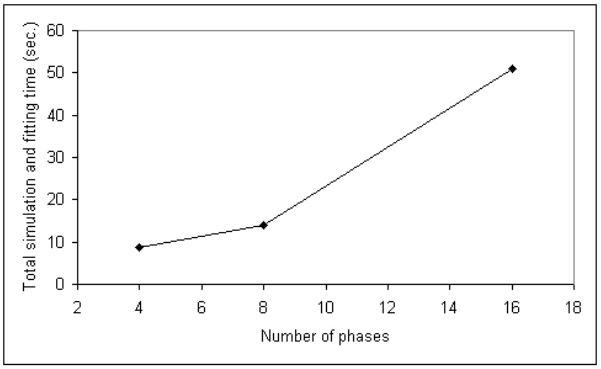


Figure 4.12: Dependence on the total simulation and fitting time on the number of phases (Isensee et al. 2005)

4.3 Summary

In this chapter we treat issues that we think are important for the proxel-based simulation, especially when deciding whether to apply it or not and what to be careful about. For some classes of models, other approaches might be more suitable or also out of consideration because of the specific description of the model. With that respect Section 4.1 can be used as an indicator or help when making the decision and point out the characteristic features of the method.

Additionally, we review two ideas for variations of the proxel-based method for which there is an ongoing research in our research group. The variants can also be considered as starting points and directions for future research on the proxel-based method and its improvement.

5 The Proxel-Adapted Modelling Framework

This section describes the framework that we developed for describing models, which is customized to the features and properties of the proxel-based method (Lazarova-Molnar and Horton 2005a). We start by elaborating the motivation for the development of the framework, after which a detailed description and definitions of the basic elements follow. Some of the terms are redefined terms from Chapter 3. Additionally, we present two examples to demonstrate the complete procedure of describing and analysing discrete stochastic models and show computational results from their analysis. Finally, in Section 5.7 we discuss and critically assess our approach.

5.1 Motivation

The direct motivation for developing this modelling framework is a practical application of the proxel-based method to a reliability modelling problem for DaimlerChrysler, elaborated in Section 6.2. There, we were able to adapt a Petri net model so that it could be analysed effectively using proxels, at the same time exploiting all the benefits of the method. This resulted in a substantial simplification of the initial model. As a result of this project, we realised that there was a necessity of having a "better" way of describing the models, so that their analysis can benefit the most from the properties of the proxel-based method.

Previously, the input to our proxel-based simulation tool was the reachability graph of a stochastic Petri net (Haas 2002), which has the disadvantage that the models have to be bounded. Here we introduce our new description approach, in which this problem does not exist any more and which exploits the other beneficial properties of the method, some of which are explained in Section 4.1. The direct input of the Petri net is also an option, but it requires optimising and adaptations before sending it to the proxel-based simulator.

In the next section we start by defining the elements needed to uniquely describe a model and compare them with a popular formalism: the one of stochastic Petri nets.

5.2 Basic Elements

The modelling framework that we introduce in this chapter represents a compact way of modelling, which distinguishes between the finite and the infinite parts of one model, using this knowledge to handle them efficiently for their further analysis, more specifically for the proxel-based analysis. The goal of the framework is to contribute to the development of a modelling and analysis tool that would be able to represent and deterministically analyse a wide class of discrete models. This means having a way to describe models such that they could be immediately analysable by the proxel-based simulator without making any changes or adaptations to the model itself and at the same time exploiting the positive features of the proxel analysis method. Having these properties in mind, our framework consists of the elements defined in the following.

Definition 5.1 – Discrete state. A discrete state *(DS) of the model is one configuration of it, independent of time. It is represented as a combination of the* partial discrete state *(PDS) and the vector of all* countable quantities *($\vec{Q}$) in the model.*

$$DS = (PDS, \vec{Q})$$

Definition 5.2 – Countable quantity. *A countable quantity (Q) is a discrete quantity in the model whose value changes according to a specific rule and which does not have to be bounded. Usually it is used when the values it can take on are infinite and it is a part of the discrete state of the model. It can be observed as a discrete supplementary variable.*

Definition 5.3 – Partial discrete state. *A partial discrete state (PDS) of the model is a part of the description of the discrete state the model is in, excluding the time dependent* countable quantities.

Definition 5.4 – Event. *An event (E) is any action which takes the model into a different discrete state (analogous to a state change). With respect to time, there are two kinds of events,* timed *and* conditional*, depending on whether the event is scheduled at a certain point in time or depends only on certain conditions being fulfilled. With respect to the memory policy of the time that one event has been waiting to happen, there are again two kinds of timed events,* age *and* restart*. The first one remembers the time that the event has been* active *i.e. ages, whereas the second one has no memory.*

Definition 5.5 – Age intensity. *An age intensity of an event is the pending time during which the event could have happened, but it has not.*

Definition 5.6 – Active event. *An active event in a discrete state is an event which fulfils all preconditions for it to happen, as defined by the* rules.

Definition 5.7 – Relevant event. *A relevant event in a discrete state is an event whose age intensity has an impact on the possible future state changes. These are the active events and the age memory events in the model. Only timed events can be marked as relevant because only they can age.*

Definition 5.8 – State. *A state (S) in one model is the combination of a* discrete state *and the age intensities of the* relevant events *in the actual discrete state. It completely describes a configuration in which the model can be.*

$$S = (DS, AgeIntensities(RelevantEvents(DS)))$$

Definition 5.9 – Rule. *A rule (R) is a description of the dynamics in the model, as a consequence of an event happening, and the preconditions for that event to happen. It describes the changes in the model that an event provokes, in terms of discrete state changes. Every event has a corresponding set of rules.*

Definition 5.10 – Lifetime of a discrete state. *A lifetime of a discrete state is the longest amount of time that one model can spend in a particular discrete state without making any state changes. It is dependent on the functions that describe the dynamics of the model and the minimum probability threshold. It can be calculated using Algorithm 3.2.*

Definition 5.11 – Proxel. *A proxel is a tuple, representing a dynamic computation unit which completely describes any state of the model with respect to the global simulation time and the* sequence of states *that leads to it, in terms of the probability for being there. It is composed of the following elements:* state of the model *(S), vector of the* age intensities *of the relevant events ($\vec{\tau}$),* global simulation time *(t), the* sequence of states *that lead to it (*Route*) and the* probability *(Pr) for all of the previous elements.*

$$Proxel = (S, \vec{\tau}, t, Route, Pr)$$

Age intensities of the relevant events are stored in an age intensity vector - $\vec{\tau}$. In order to reduce the complexity of the proxel structure, there are discrete state-dependent mappings of the age intensities. This means that each component of the vector has a different meaning (i.e. represents the age of a different event) in every discrete state. Based on these definitions, every model can be represented as a tuple

$$(PDS, E, Q, R, (PDS_0, Q_0)),$$

where each of the elements has a meaning, described as follows:

- PDS is the set of partial discrete states:

$$PDS = \{PDS_1, PDS_2, \ldots, PDS_n\},$$

- E is the set of events:

$$E = \{E_1, E_2, \ldots, E_m\},$$

- Q is the set of countable quantities:

$$Q = \{Q_1, Q_2, \ldots, Q_l\},$$

- R is the set of rules which describe the effect of, and the conditions under which, each event happens in terms of discrete state changes:

$$R = \{R_1, R_2, \ldots, R_j\},$$

- DS_0 is the *initial discrete state*, represented as a vector of the initial partial discrete state and the initial values of the countable quantities:

$$DS_0 = (PDS_0, Q_0).$$

Rules can as well be time dependent, just as the parameters of the distribution functions that describe the activation times of the events. This means that different complex situations can be modelled, such as, for example, a queuing system in which customers in the afternoon arrive usually in couples and with a lower frequency than at other times during the day.

In the proposed formalism, some of the defined terms have corresponding Petri net elements, as shown in Table 5.1.

Table 5.1: Our modelling framework and SPNs: A parallel

our approach	***SPNs***
discrete state	marking
event	transition
age intensity	activation time of a timed transition
relevant event	marking-enabled or age memory timed transition
set of rules	incidence matrix

In general, when compared to the Petri net formalism, our proposed framework can be interpreted as a proxel-adapted compromise between the Petri net itself and its reachability graph, where the finite part of the Petri net is the basis for the partial discrete state space and the infinite part for the countable quantities. This framework for describing discrete stochastic models makes their proxel-based simulation and analysis straightforward as will be shown with concrete examples.

5.3 Graphical Description

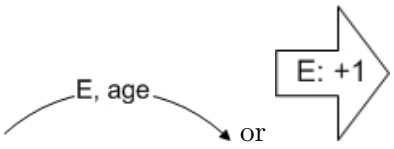
To support the comprehension and enhance the intuitiveness of our modelling framework, we designed a corresponding graphical representation. This also contributes to

a more understandable illustration of the models and their dynamics. Each of the elements has its corresponding graphical symbol, as follows:

- Partial discrete state (labelled circle):

- Countable quantity (labelled square):

- Event (labelled arrow):



The descriptions of the effects (e.g. changes of the quantities) that the events cause and the properties of the events (e.g. age; restart is the default) are labelled on the graphical symbols. The arrows (i.e. events) can also connect two corresponding partial discrete states to show the switch between the two. If an event causes a change of a countable quantity then the arrow, its graphical representation, is attached to the square that represents that particular countable quantity, as shown in Figure 5.2.

5.4 Adapted Algorithm

In this section we present the algorithm for the proxel-based analysis of general models. It is based on Algorithm 3.1 presented and explained in Section 3.1.3, and adapted accordingly to use our modelling framework as a starting point.

In Algorithm 5.1 the new adapted algorithm is shown. For simplicity reasons the route and the simulation time components are excluded. There is no need to explicitly store them as they are implicitly considered.

The meanings of the variables used in the algorithm are the following:

- PDS is a partial discrete state,
- Q is a vector of the countable quantities,

Algorithm 5.1: Extended Proxel Algorithm (2)

Input: $\Delta t, t_{max}$

Initialize proxel set $PS(0)$ by inserting the initial proxel;
$switch = 0$;
for *i=1* **to** $[t_{max}/\Delta t]$ **do**
  foreach *proxel* $p = ((PDS, Q), \vec{\tau}, prob)$ *in the proxel set* $PS(switch)$ **do**
    foreach *active event* E_i *in* PDS **do**
      Calculate probability $prob_{calculated}$ for the event E_i;
      Generate new state $S = (succ(PDS), succ(Q))$ according to rules (E_i);
      if $pds(S)$ *enables set of conditional events* $cE = \{cE_i\}$ **then**
        Generate set of new states $nS = \{nS_i\}$ according to cE_i
        foreach nS_i *in* nS **do**
          Calculate probability $prob_{calculated}$ for the event cE_i;
          if *search(*nS_i *in* $PS(1 - switch)$*)* $= proxel_{found}$ **then**
            $proxel_{found} = (nS_i, (prob(proxel_{found}) + prob_{calculated}))$
          end
          Generate new proxel $p_{new} = (nS_i, prob_{calculated})$;
          Store p_{new} in $PS(1 - switch)$;
        end
      end
      else if *search(*S *in* $PS(1 - switch)$*)* $= proxel_{found}$ **then**
        $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$
      end
      else
        Generate new proxel $p_{new} = (S, prob_{calculated})$
      end
      $prob_{rest} = prob - prob_{calculated}$;
    end
    Generate new proxel $p_{new} = ((PDS, Q), succ_{stay}(\vec{\tau}), prob_{rest})$;
    Store p_{new} in $PS(1 - switch)$;
    Remove the processed proxel p from $PS(switch)$;
  end
  $switch = 1 - switch$
end

- $pds(S)$ is a function which extracts the partial discrete state from a state S,
- $\vec{\tau}$ is the age intensity vector which contains the age intensities of the relevant events,
- *prob* is the probability of the proxel that is being processed,
- $search(S\ in\ PS)$ is a function that searches for a state S in a set of proxels PS and returns the corresponding proxel if successful,
- $rules(E_i)$ is the set of rules associated with an event E_i, and
- $succ(\vec{\tau})$ and $succ_{stay}(\vec{\tau})$ are the newly calculated successors of the age intensity vectors for leaving a discrete state (according to the set of rules $rules(E_i)$) and staying there, correspondingly.

The probability of a proxel ($prob_{calculated}$ in lines 6 and 10) is approximated by integrating the instantaneous rate function (IRF) along the interval $(t, t + \Delta t)$, using any of the numerical approaches presented in Section 3.2.

Algorithm 5.1 differs from Algorithm 3.1 in that that it checks at every event if any conditional events have become enabled (line 8). If the answer is positive, it implements the corresponding state changes without updating the simulation time, i.e. within the same time step.

The proposed modelling formalism provides an appropriate input format for the proxel simulator because the proxel-based method operates on the description of the model and the framework allows a high degree of flexibility in the model description. The rules that are part of the framework are easily adaptable to describe any situation that we have examined until now. At the same time the framework provides a compact input of the state, which makes the proxel-based simulation more efficient.

5.5 Examples

In this section we use two concrete examples to illustrate the terms defined in the previous section and show some of the advantages of our proposed modelling framework. The first example is a model of a queuing system and the other one of a fault-tolerant machine. The examples are classical, yet they contain features which make them difficult to be analysed using standard approaches (discussed in Section 4.1), as described in detail along with the model descriptions.

5.5.1 Queuing System

The first example model is a queuing system that consists of two queues and one server, which can be interpreted as a system that operates with two types of customers which arrive according to two different distribution functions and are served by one employee. The queues are unbounded. This would already present a problem when

using a standard deterministic approach when the arrivals are generally distributed. In that case, the first pre-processing step that would have to be carried out would be to bound the lengths of the queues, i.e. impose a restriction on the model, which means having to change the model for every alteration of the distribution functions or their parameters. Proxels, however, can analyse the model without making any adaptations to it as they create the state space on-the-fly while simulating the behaviour of the model. The new modelling framework aids this process by providing a description of the models making them directly analysable by the proxel-based method without making any modifications.

The Petri net of the queuing model is shown in Figure 5.1. It consists of two places for the queues and one place for the server. The inhibitor arc illustrates the limitation of the server of serving one customer at a time.

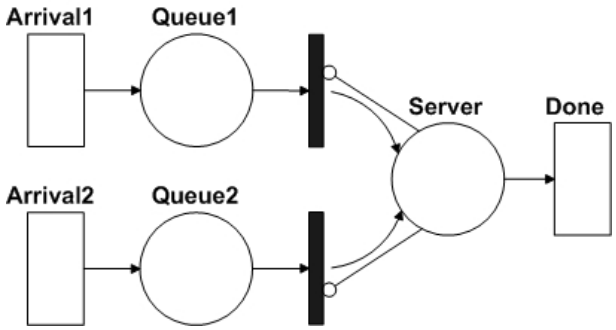


Figure 5.1: Petri net description of the queuing system model

In this model we recognise two candidates for countable quantities, i.e. the numbers of people in the both queues, and two partial discrete states, i.e. server-free and server-busy, which in this case are sufficient to describe the discrete state space. We label the countable quantities by Q_1 and Q_2 correspondingly, and the two partial discrete states by B (as busy) and F (as free).

Five events can be distinguished in this model:

- E_1 - customer enters the first queue,
- E_2 - customer enters the second queue,
- E_3 - customer from the first queue moves to the server,
- E_4 - customer from the second queue moves to the server, and
- E_5 - customer completes service.

which operate according to the following rules:

- $E_1 \rightarrow Q_1 + 1$,
- $E_2 \rightarrow Q_2 + 1$,
- $E_3 \rightarrow Q_1 - 1$, (if F then B),

- $E_4 \rightarrow Q_2 - 1$, (if F then B),
- E_5 (if B then F).

Three of the rules include if-conditions, where the if part describes the precondition(s) for each of the events to happen. For example, the server needs to be free (F) so that a customer from either of the queues can occupy it (events E_3 and E_4).

In Figure 5.2 a graphical representation of the model, using the proxel-adapted description framework, is shown.

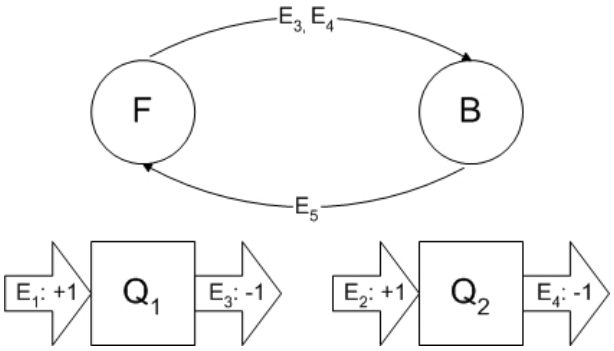


Figure 5.2: Graphical representation of the model using our proposed approach

As already stated, there are two types of events: timed and conditional. In our model, E_3 and E_4 represent conditional events, whereas all of the others are timed. This concept corresponds to the Petri net concepts of timed and immediate transitions.

The initial partial discrete state of the model is F and the initial values of the countable quantities are both zeros (i.e. the queues are empty in the beginning), meaning that the initial discrete state is $(F, 0, 0)$.

The relevant events, which can only be timed, are the following:

- In F - E_1 and E_2
- In B - E_1, E_2 and E_5.

Because at any partial discrete state there are at most three relevant events, the age intensity vector consists of three components and has the following mappings: in $F \sim (E_1, E_2, /)$ and in $B \sim (E_1, E_2, E_5)$, where the "/" symbol means that there is no mapping for that component, i.e. that component in the vector is inactive.

The general *discrete state vector* for this model, which describes the structure of the discrete states of the model is the following:

$$(PDS, Q_1, Q_2), \text{where } PDS \in \{F, B\}, \text{and } Q_1, Q_2 \in Z_0^+.$$

Correspondingly, the initial state vector is the following:

$$((F, 0, 0), (0, 0, /)).$$

According to the event rules, the following states could be reached in the next time step:

1. $((F, 1, 0), (0, \Delta t, /))$, via E_1 i.e. customer arrival in the first queue,
2. $((F, 0, 1), (\Delta t, 0, /))$, via E_2 i.e. customer arrival in the second queue, and
3. $((F, 0, 0), (\Delta t, \Delta t, /))$, no event has happened.

In the first two cases there are conditional events activated, i.e. the arrived customer moves to service because the server is free. Therefore, the model behaviour takes the corresponding directions immediately, in the same simulation time step. The *corrected* possible states to be reached at time step $t = \Delta t$ are:

1. $((B, 0, 0), (0, \Delta t, 0))$, via E_1 and E_3,
2. $((B, 0, 0), (\Delta t, 0, 0))$, via E_2 and E_4, and
3. $((F, 0, 0), (\Delta t, \Delta t, /))$, no event has happened.

The probabilities for the developed successive states are omitted in the descriptions above. However, they can be computed using the instantaneous rate function, using the Equation (2.3).

Once the probability for a state has been computed, it is attached to the state vector to form the proxel. The probability for staying in the same discrete state in the next time step is calculated as a subtraction of the probabilities for leaving the discrete state from the probability for being there.

If the model is to be analysed by the proxel-based method, using Petri net representation, one of the following two approaches must be used. The first one is to construct the reachability graph, which again means bounding the state space of the model a priori. The second approach is to build the state space on-the-fly, which avoids the disadvantage of the former method, but it leads to complications when storing the proxels, because that is the worst complexity that the format of one proxel can have i.e. the number of tokens at all places in the Petri net.

The proposed approach is a combination of the two. It uses an array for enumerating the partial discrete states, and additional components where necessary: the infinite places (i.e. the countable quantities).

The goal of this example is to show how the modelling framework works when used as an input formalism to the proxel-based simulation. Particularly in this example, there is no reduction of the computational complexity compared to the direct Petri net simulation because the dimension of the discrete state vector is equal to the number of places in the Petri net. The saving is meaningful when the finite part of the Petri net is represented by more than one place, as is the case in the next example, where three places are represented by one component in the discrete state i.e. the partial discrete state.

5.5.2 Machine Model

The second example illustrates some additional features of the method including the possibility of modelling rewards. The model represents a fault-tolerant machine which has scheduled almost regular maintenance and fails according to a given distribution function, which is a function of the age of the machine and the number of failures that have happened until that point in time. The Petri net that describes the model is shown in Figure 5.3.

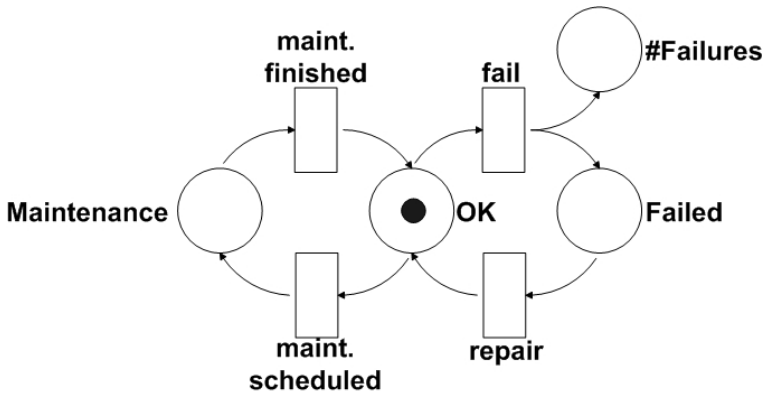


Figure 5.3: Petri net of the machine model

The state space of this model is unbounded because the number of failures is not limited. As is the case with the previous example, this would mean limiting it in order to analyse the model numerically.

According to our proxel-adapted description framework, the model has the following partial discrete states:

- OK - machine operates normally,
- F - machine is failed,
- M - machine is being maintained,

and one countable quantity ♯F, which tracks the number of failures.

The model contains four events, each distributed according to the assigned distribution function:

- E_1 - machine goes to maintenance mode $\sim F_1()$,
- E_2 - maintenance completed $\sim F_2()$,
- E_3 - machine fails $\sim F_3()$,
- E_4 - repair completed $\sim F_4(\sharp F, t)$ (repair time is a function of the number of failures $\sharp F$ and the age of the machine i.e. simulation time t),

which obey the following rules:

- $E_1 \rightarrow$ (if OK then M), age memory,
- $E_2 \rightarrow$ (if M then OK),
- $E_3 \rightarrow$ (if OK then F), $\sharp F + 1$,
- $E_4 \rightarrow$ (if F then OK).

The specific functions and parameters are provided in Section 5.6 i.e. the experiments section. The graphical representation of the machine model is illustrated in Figure 5.4.

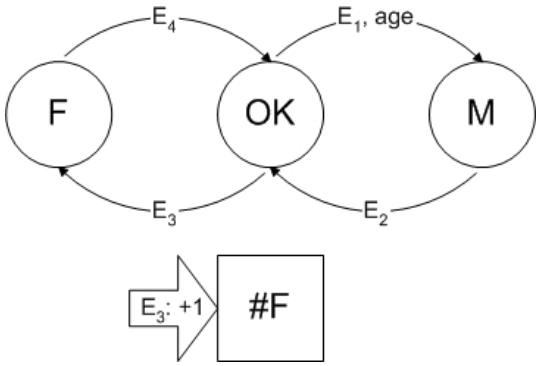


Figure 5.4: Graphical representation of the proxel-adapted machine model

The age intensity vector is two-dimensional, with the following mappings of the events' age intensities to the discrete states: $OK \sim (E_1, E_3)$, $M \sim (E_2, /)$, and $F \sim (E_1, E_4)$. The age intensity of the event E_1 is present in both partial discrete states OK and F, because it has an age memory policy and its elapsed time needs to be remembered, whereas in M it is the event that got the model into that state and it is not active any more, which is why it is not included into the mapping vector. The discrete state vector has the form (PDS, $\sharp$F), where PDS $\in \{OK, M, F\}$ and $\sharp$F$\in Z_0^+$. Therefore the initial state is

$$((OK, 0), (0, 0)),$$

resulting in the following subsequent states:

1. $((F, 1), (\Delta t, 0))$, via E_4 i.e. machine has failed; the number of failures is also increased,
2. $((M, 0), (0, 0))$, via E_1 i.e. machine goes to maintenance, and
3. $((OK, 0), (\Delta t, \Delta t))$, no event has happened.

From the two example models it can be seen that the proxel-based method operates by exploring all of the possible behaviours of the model in an intuitive way, only by

following the rules attached to the events in the model. Therefore the properties of the models, which are usually considered as undesired, fit very well into the existing framework.

5.6 Experiments with the Example Models

In this section we present results of some of the experiments performed using the two examples described in Section 5.5. The goal of this section is to provide a general impression about the performance of the proxel-based method and the kinds of questions it can answer, given that the models are described using the proposed formalism. In the first subsection we present results concerning the queuing model and in the second one results concerning the machine model.

5.6.1 Experiments with the Queuing Model

The queuing model is interesting because its state space is unbounded. Therefore, the reachability graph of the model would be unbounded as well, as represented symbolically in Figure 5.5. The rounded boxes represent tangible markings and the rectangles vanishing markings. The two big ovals represent the partial discrete states, as defined by our proxel-adapted modelling framework.

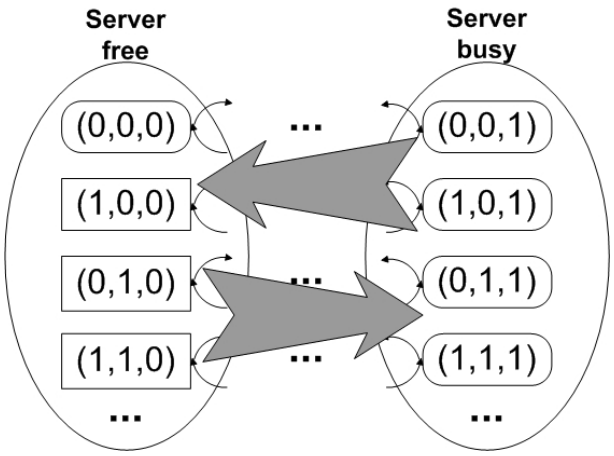


Figure 5.5: The reachability graph of the queue model and its connection with the proxel-adapted model description

For the experiments we use the following distribution functions:

- $E_1 \sim$ Uniform (0.8, 1.6),
- $E_2 \sim$ Uniform (1.0, 1.8), and
- $E_5 \sim$ Exponential (2.0),

for describing the corresponding timed events.

The solutions that the analysis provides are the transient probabilities for the following discrete states:

(1) Server is busy and there are no customers in the queues,
(2) Server is busy and there are no customers in the first queue and the second queue is not empty,
(3) Server is busy and there are no customers in the second queue and the first queue is not empty,
(4) Server is busy and both queues have at least one customer, and
(5) Server is free and there are no customers in the queues.

The results are shown in Figure 5.6 in the corresponding order. Even though the model contains unbounded elements, still a smooth transient solution is feasible. The only other option that can analyse unbounded models without imposing restrictions, the DES, would need a large number of replication to achieve such a smooth result.

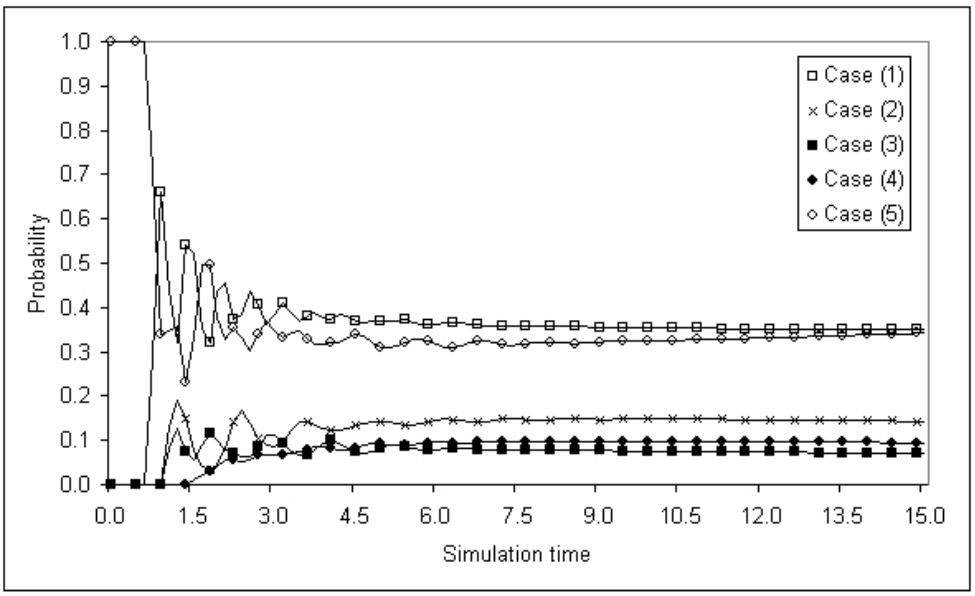


Figure 5.6: Transient probabilities of the discrete states (cases) described above

The computation time for this experiment was 500 seconds, using a time step $\Delta t = 0.2$, simulating up to time $t = 15$. Observing the results shown in Figure 5.6 we can come to a conclusion that this model, given the distribution functions, has a limiting behaviour i.e. the transient probabilities approach the steady state ones as time goes to infinity. The highest probability have the states described in case (1) and (5) i.e. where both of the queues are empty and the server is either busy or free. This is also to be expected, given that the rate of service is quite fast compared to the arrival rates.

5.6.2 Experiments with the Machine Model

The machine model is more interesting because it exploits the feature of having the distribution functions of the events depend on the values of the countable quantities and the global simulation time.

The distribution functions that we use for describing the events for the purpose of experimenting with this model are the following:

- $E_1 \sim Deterministic(20.0)$,
- $E_2 \sim Exponential(3.0)$,
- $E_3 \sim Uniform(10.0, 20.0)$, and
- $E_4 \sim Uniform(1.0 + 0.1 * (\sharp F + t), 15.0 + 0.05 * \sharp F)$, depending on both the number of failures that have happened $\sharp F$ and the global simulation time t.

The probabilities that we are interested in are the ones for the following discrete states:

(1) The machine has not had any failures,
(2) The machine has had one failure,
(3) The machine has had two or more failures, and
(4) The system is being repaired.

The results from the corresponding proxel-based analysis are shown in Figure 5.7, where it can be observed how the probability of the machine having more than two failures, i.e. case (3), slowly approaches the value 1.0, as expected. The probability of the discrete state in which the machine is failed, i.e. case (4), has not stabilised during the predefined simulation time.

The second thing that might be of interest in this model is the number of failures and its transient development. The result of the analysis of that parameter is shown in Figure 5.8.

The computation time for this experiment was 0.06 seconds, using a time step $\Delta t = 1.0$, running the simulation up to time $t = 80$.

Based on the experiments, it is apparent that the proxel-based analysis provides complete solutions to the model questions i.e. transient solutions. The obtained solutions support the process of drawing conclusions about the behaviour of the model, and make it possible to predict it for points in time for which it has not yet been simulated. Another goal of this model is to show that the proxel-based method can be seen as a promising tool in reward and performability modelling (Haverkort et al. 2001), which is described in detail in Section 6.1.

Both series of experiments in this section analyse models which have properties that are usually found as unwanted and problematic. However, this is not an obstacle for the proxel-based simulation. Just for a comparison, the models could have been analysed

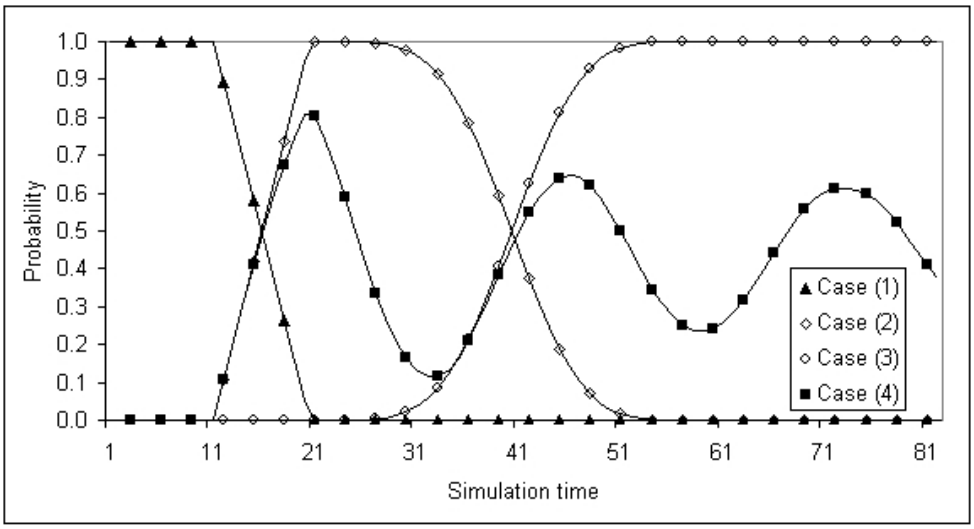


Figure 5.7: Transient probabilities of the discrete states described above

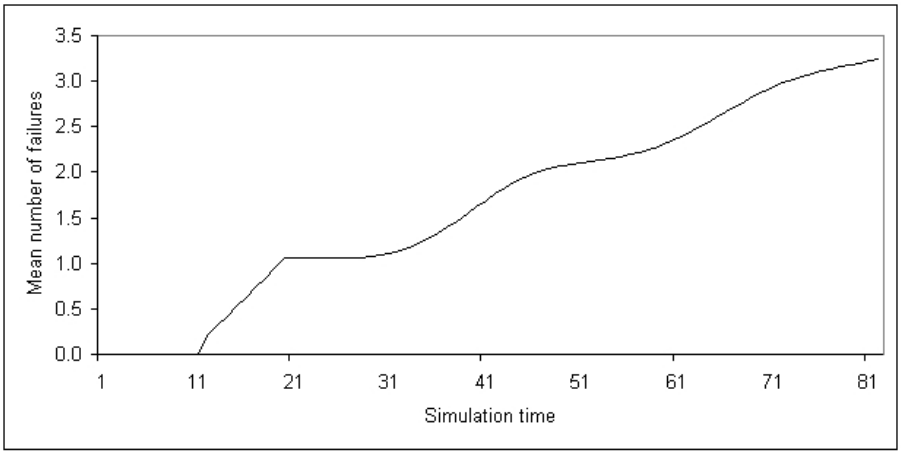


Figure 5.8: The mean number of failures as a function of the global simulation time

using discrete-event simulation, but then the results would have been in a form of confidence intervals requiring a large number of independent replications to achieve or get closer to the accuracy that the proxel-based method provides. This is shown in Section 6.2, where for the model that is analysed the discrete-event simulation needs 20 to 30 hours, whereas the proxel-based simulation for achieving the corresponding accuracy, a couple of seconds to a couple of minutes.

5.7 Discussion and Conclusions

The motivation for developing this modelling framework is a practical application of the proxel-based method to a reliability modelling problem for DaimlerChrysler (Lazarova-Molnar and Horton 2004). This experience made us realise the need for a description method which would be able to handle unbounded models with as few variables as possible, which lead to observing the complete state space on two levels, as illustrated in Figure 5.5. We recognised that some of the elements of the model can be observed as discrete supplementary variables of the so-called partial discrete states and that their behaviour, because of regularities, can be described by rules. This observation lead to the idea of having countable quantities and rules.

We also allow models described by our description framework to have additional features that are supported by the proxel-based method. One of these is having distribution functions that depend on the discrete states or the simulation time, which is a very realistic assumption and very complicated to analyse using partial differential equations. On the other hand, it is definitely doable by discrete-event simulation, nevertheless there we may encounter the problem of having extremely long simulation times if we try to achieve the quality of solutions that the proxels provide. Another problem one might come across when using discrete-event simulation are rare events, to which the proxel-based method is undoubtedly less sensitive because all of the events there are equally important, as shown in (Lazarova-Molnar and Horton 2003a).

One disadvantage is that probably the proposed framework is less intuitive than the formalism of stochastic Petri nets, but that is still hard to say because it is new. One of our future goals is to provide a tool which would convert SPNs into our proxel-adapted descriptions.

Finally, given the mentioned properties of the proxel-based method, we believe that our proposed framework contributes to making the method a generally applicable tool for analysing stochastic models.

5.8 Summary

In this chapter we introduce a modelling framework which responds to the input requirements and analysis capabilities of the proxel-based method. For that purpose, we revise the definitions of some elements and introduce new ones, as well as update the basic algorithm for proxel-based simulation. We illustrate our modelling framework by two examples which demonstrate different features of the method.

6 Applications

The proxel-based method is theoretically not limited in the classes of problems that it can address. The main reason for that is that the method works by creating the solution algorithm directly from the description of the model, skipping the step of developing a computational model, which enables the method to deal with any situation that can arise. In this section we report on applications of the proxel-based method, for which we believe that it can be useful. Each of the application areas is supported by concrete examples.

6.1 Proxels in Reward Modelling and Performability Analysis

Performability modelling and analysis is used as a measure of both performance and dependability of, mainly, fault-tolerant systems. Because of the flexible definition of the proxels, they can easily be extended to provide a widely-applicable approach to performability modelling.

In (Lazarova-Molnar and Horton 2005b) we establish the framework for proxel-based performability modelling and analysis, and implicitly for general reward modelling. Two types of rewards are considered: impulse and rate rewards. Impulse rewards occur with state changes and are associated with them, whereas rate rewards are 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.

The motivation for this application field of the proxel-based method is a practical experience with the method for analysing a warranty model, a project which we carried out for DaimlerChrysler, as described in Section 6.2. The model that needed to be analysed involved manipulating only impulse rewards in order to analyse the costs. It required a slight adaptation of the existing framework of the basic proxel-based method for tracking the rewards, without imposing any additional significant difficulties with respect to the computational and memory complexity of the implementation. Based on that experience we recognized that the method can be useful for carrying out a

more general performability analysis which also includes rate rewards, which approach is described in the following.

To start with, we provide a short overview of performability theory and discuss some of the more popular existing approaches. Further we describe and explain why the proxels are a good option for carrying out performability analysis of discrete stochastic models and describe how it is to be achieved. To aid comprehension and show how it works in practice, in the experiments section we introduce an example model which we then use to test two different cases of performability modelling.

6.1.1 Performability Modelling

Performability modelling (Haverkort et al. 2001) is a modelling approach for simultaneously evaluating both the performance of a stochastic system and its reliability. This measure is of exceptional 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 a system is provided the system is correct. It can be measured as throughput, response time, etc. Reliability, on the other hand, is a measure of the system's ability to function correctly over a specific period of time i.e. describes how reliable it is. Reliability of a discrete and stochastic system *Sys* can be expressed mathematically in the following form:

$$R(t) = Pr(Sys \text{ operates correctly } in\, [0, t)). \tag{6.1}$$

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

$$R(t) = Pr(L > t) = 1 - F(t) \tag{6.2}$$

which is nothing but the survival function, as defined in Section 2.1.1.

Using common words, performability modelling evaluates and answers the following question:

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

and computes the function that describes it. The completed amount of work is then the accumulated performance over a given time interval, taking into account the operativeness of the separate components of the system, referred to as *performability*.

The performance of one system is measured using a reward function $rr(DS, \tau)$ which evaluates its efficiency in each of its discrete states and can also be dependent on the time spent in it 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 and assigning *reward rates* or *functions* to each of its states. The reward rates estimate the performance of the system in each state. The estimates can be obtained using performance analysis of the system, which means running the system in every possible configuration and evaluating the measure that characterises its performance, which is then denoted as a reward rate of that state in the Markov chain.

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

- *expected performance* of a system at a certain point in time,
- *time-averaged performance* of a system over a time interval $(0, t)$, and
- *amount of work accomplished* over a time interval $(0, t)$,

all of them taking into account the effects of failures, repairs, as well as the other possible conditions of the system.

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

$$W(t) = Y(t)/t. \tag{6.3}$$

$Y(t)$, which denotes the amount of work accomplished at time t, is generally more useful for Markov reward models that contain absorbing states, and $W(t)$ for the others. The biggest drawback of the Markov reward model is that it is not directly applicable to models that contain generally distributed events i.e. it can only analyse models which contain exclusively exponentially distributed activities.

Another common tool for performing performability modelling are *stochastic reward nets* (SRNs) (Muppala et al. 1994), which are based on *generalised stochastic Petri nets* (GSPNs) (Kartson et al. 1994). Because of the equivalence between GSPNs and Markov chains, SRNs can be mapped onto Markov-reward models by associating rewards to all tangible markings of the reachability graph.

TimeNET (Zimmermann et al. 2000) which was reviewed in Section 2.2.3, carries out performability analysis of a wide class of stochastic Petri nets with generally distributed firing delays, including 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 numerical approach works on the 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.

6.1.2 Performability and Proxels

In this section we describe how performability modelling can be carried out using the proxel-based method, as well as formalise the approach by developing an algorithm. Performability analysis in general works by accumulating and manipulating rewards. There are two types of rewards that are treated by the proxel approach: 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 use standard state-transition diagrams for describing the models. We define *performance* rr in form of a reward rate as a function on the discrete state space DS of the model:

$$rr : DS \rightarrow \Re, \text{ or} \tag{6.4}$$

$$rr : DS \times \Re^n \rightarrow \Re, n \in N, \tag{6.5}$$

where $\Re$ is the set of real numbers and N the set of natural numbers. A higher reward rate means a higher performance of the system.

The second Expression (6.5) shows that the performance can also be a function of another parameter besides the discrete state (such as the simulation time, age intensity, or the number of times that a state change has happened, calculated as an impulse reward), and that is supported by the proxel-based method.

Impulse rewards ir are defined in a similar way, except that they are functions of the state changes. Analogously we have:

$$ir : DS \times DS \rightarrow \Re, \text{ or} \tag{6.6}$$

$$ir : DS \times DS \times \Re^n \rightarrow \Re, n \in N. \tag{6.7}$$

If the performance of the system is dependent on anything else besides the elements which are by definition part of the proxel, then that element has to be included in the state of the system as an additional variable. For example, in a situation where the performance of the system decreases following a given function, which is proportional to the number of times that a certain component has failed, then the number of failures has to be included in the state description. The situation described is also a very realistic assumption (also treated in (Sule and Castro 2002)), which gives an additional credit to the proxel-based simulation for being able to handle such complex configurations.

Based on Expressions (6.4) to (6.7), if DS is the discrete state-space of the model, we specify the relevant measures in performability analysis that can be evaluated by the proxel-based method as follows (Haverkort et al. 2001):

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

$$SSP = \sum_{i \in DS} \pi_i rr(i) \tag{6.8}$$

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

- *Transient performability* (TP), defined as

$$TP(t) = \sum_{i \in DS} p_i(t) rr(i) \tag{6.9}$$

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

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

$$EW(t) = \int_0^t TP(s) ds \tag{6.10}$$

  and presents the integrated transient performability up 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* $TP()$ and the *expected work* $EW()$ for every time step are computed according to the Equations (6.11) and (6.12) correspondingly.

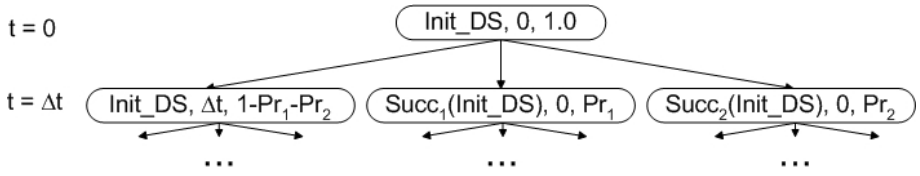
$$TP(k \times \Delta t) = \sum_{\forall proxel\, Px \text{ at } k\Delta t} [Pr(Px) \times (rr(ds(Px)) + ir(ds(pre(Px)), ds(Px)))] \tag{6.11}$$

$$EW(k \times \Delta t) = \sum_{i=0}^{k} TP(i \times \Delta t)\Delta t, \text{ where } k = \lceil t/\Delta t \rceil \tag{6.12}$$

$Pr(Px)$ is the probability that the proxel Px carries, whereas $ds(Px)$ is a function that extracts the discrete state from the proxel Px. $pre(Px)$ is the predecessor (i.e. parent) proxel of Px. The steady state performability can be implicitly calculated from these two measures.

An illustration of the proxel-based method, along with the computation of the performability measures is shown in Figure 6.1 in the table below the proxel-tree. The figure is, however, simplified to aid the comprehension. The proxels in this figure consist of only three components: discrete state, one age intensity and a 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 discrete state of the model, i.e. the one that the model occupies at time $t = 0$. The model in this case has three discrete states: the initial one and two successors of it. Their performances are denoted by their reward rates:

rr_1, rr_2, and rr_3. We assume that there are also two impulse rewards for the two state changes from the initial discrete state to the two successors, denoted by ir_1 and ir_2.



Performance(Init_DS) = rr(Init_DS) = rr_1
Performance($Succ_1$(Init_DS)) = rr($Succ_1$(Init_DS)) = rr_2
Performance($Succ_2$(Init_DS)) = rr($Succ_2$(Init_DS)) = rr_3
ir(Init_DS, $Succ_1$(Init_DS)) = ir_1
ir(Init_DS, $Succ_2$(Init_DS)) = ir_2

t	TP(t)	EW(t)
0	rr_1*1.0	TP(0)Δt
Δt	rr_1*(1-Pr_1-Pr_2) + rr_2*Pr_1 + rr_3*Pr_2 + ir_1*Pr_1+ ir_2*Pr_2	TP(0)Δt + TP(Δt)Δt

Figure 6.1: Simplified illustration of the initial steps in the proxel-based performability analysis

The proxel-based performability analysis is described by Algorithm 6.1. There, in lines 20, 21, 25, and 26 the calculations of transient 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 20 and 21, i.e. when the state changes happen, whereas in the case that the model stays in the same discrete state, they are omitted. Algorithm 6.1 is an extension of the basic algorithm for proxel-based analysis, which again operates based on two parallel data structures, one of which stores the proxels from the previous step and the other 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.

The algorithm shows 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 proxel-based performability analysis on a simple example model works.

6.1.3 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 consists of two different processors that operate at different speeds and have different properties. The performance of the separate processors are functions of the numbers of failures of the processors.

Algorithm 6.1: Proxel Performability Algorithm

Input: $\Delta t, t_{max}$

Initialize proxel set $PS(0)$ by inserting the initial proxel;
$switch = 0$;
$EW(0) = 0$;
for *i=1* **to** $[t_{max}/\Delta t]$ **do**
  $TP(0) = 0$;
  foreach *proxel* $p = ((DS, \overrightarrow{\tau}), prob)$ *in the proxel set* $PS(switch)$ **do**
    foreach *active state change* C_i *in* DS **do**
      Get x as the mapping index of C_i in $\overrightarrow{\tau}$;
      Compute $succ(\overrightarrow{\tau}) = \overrightarrow{\tau} \times I_x$;
      Compute $succ(DS)$ as a successor discrete state of DS, resulting from the state change C_i;
      Generate new state $S = (succ(DS), succ(\overrightarrow{\tau}))$;
      Compute probability $prob_{calculated} = IRF(C_i, \tau_x) \times \Delta t$ for the state change C_i;
      Search for the state S in the states of the set of generated proxels;
      if $proxel_{found}$ *is the found proxel* **then**
        $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$
      else
        Generate new proxel $p_{new} = (S, prob_{calculated})$;
        Store p_{new} in $PS(1 - switch)$;
      end
      $TP(i) = TP(i) + rr(succ(DS)) \times prob_{calculated} + ir(C_i) \times prob_{calculated}$;
      $EW(i) = EW(i) + rr(succ(DS)) \times prob_{calculated} \times \Delta t + ir(C_i) \times prob_{calculated}$;
      $prob_{rest} = prob - prob_{calculated}$;
    end
    Generate new proxel $p_{new} = ((DS, succ_{stay}(\overrightarrow{\tau}))prob_{rest})$;
    $TP(i) = TP(i) + rr(DS) \times prob_{rest}$;
    $EW(i) = EW(i) + rr(DS) \times prob_{rest} \times \Delta t$;
    Remove the processed proxel p from $TS(switch)$;
  end
  $switch = 1 - switch$
end

Description of the Example Model

The state diagram of the model that we elaborate and experiment with is illustrated in Figure 6.2.

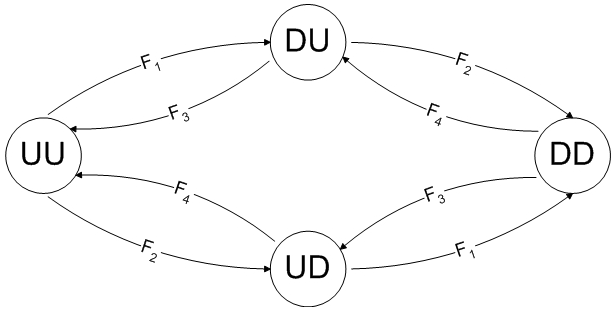


Figure 6.2: State diagram of the example model

The model has four discrete states:

- UU i.e. both processors up,
- UD i.e. faster processor up and slower down,
- DU i.e. slower processor up and faster down, and
- DD i.e. both processors down,

and each of them has a performance value in form of a reward rate associated with it. The assumption is that one of the processors is initially 10 times as fast as the other one, but the performance also depends on the number of failures that have happened to each of them. This means that the performance of the system decreases with each failure that has happened, which is a realistic behaviour situation. The functions which describe the reward rates of the four discrete states are the following:

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

where f_1 and f_2 denote the number of times each of the processors has failed. In Figure 6.3 the dependence of the performance on the total number of failures for the discrete state UU is shown. The amount of work that the slower computer can accomplish in a time step Δt without any failures is the unit for measuring the amount of work done. The failure distribution functions that we use for the experiments are the following:

- $F_1 \sim Weibull(15.0, 1.5)$,
- $F_2 \sim Uniform(3.0, 6.0)$,

- $F_3 \sim Deterministic(3.0)$, and
- $F_4 \sim Uniform(1.0, 5.0)$.

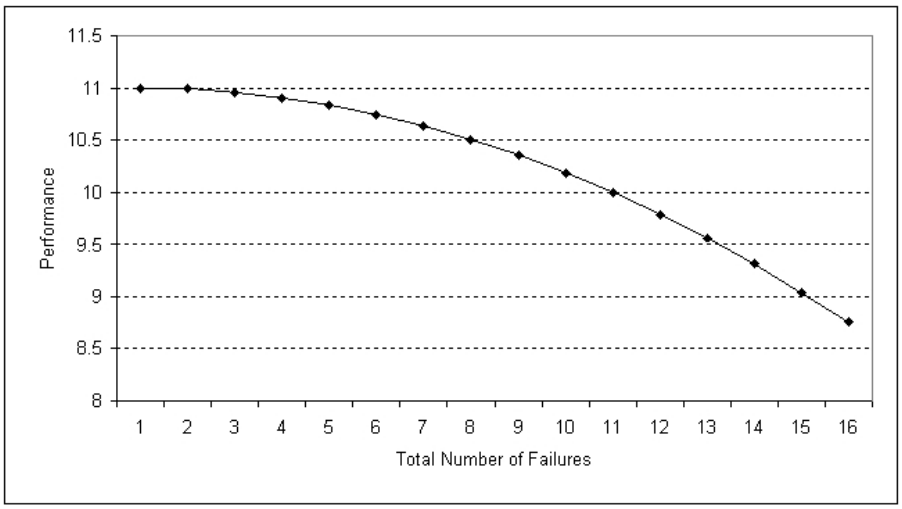
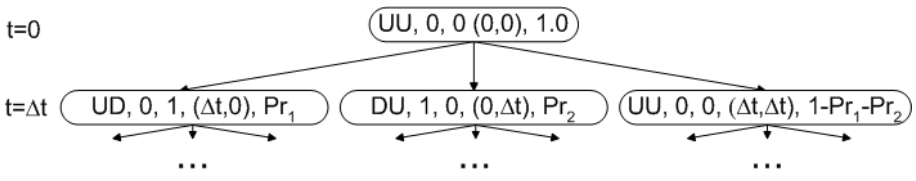


Figure 6.3: Dependence of the performance with respect to the number of failures for case B

In the same manner as in Figure 6.1, the illustration of the proxel-based performability analysis of this model is presented in Figure 6.4. 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, their elapsed operating and repair times. There are two additional discrete variables to the state of the model which trace the numbers of failures for both processors correspondingly. 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, than in a case in which the performance is constant.



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 6.4: Illustration of the initial steps of the proxel-based performability analysis process for the example model including the discrete supplementary variables

The results of the experiments with this model follow in the next subsection.

Results of the Performability Evaluation of the Example Model

In the experiments we analysed the model presented in Figure 6.2 with a 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 6.5, based on which it can be pointed out that the model goes ultimately into a steady-state, in which the discrete state in which both processors are up (i.e. UU) has the highest probability, followed by the discrete state in which the faster processor is up and the slower is down (i.e. UD). The transient performability evaluation for the same model, which is shown in Figure 6.6, depicts the behaviour of the model in that, that the transient performability stays relatively high, and decreases slowly because the transient probabilities of the discrete states DD and DU slowly increase and become constant correspondingly.

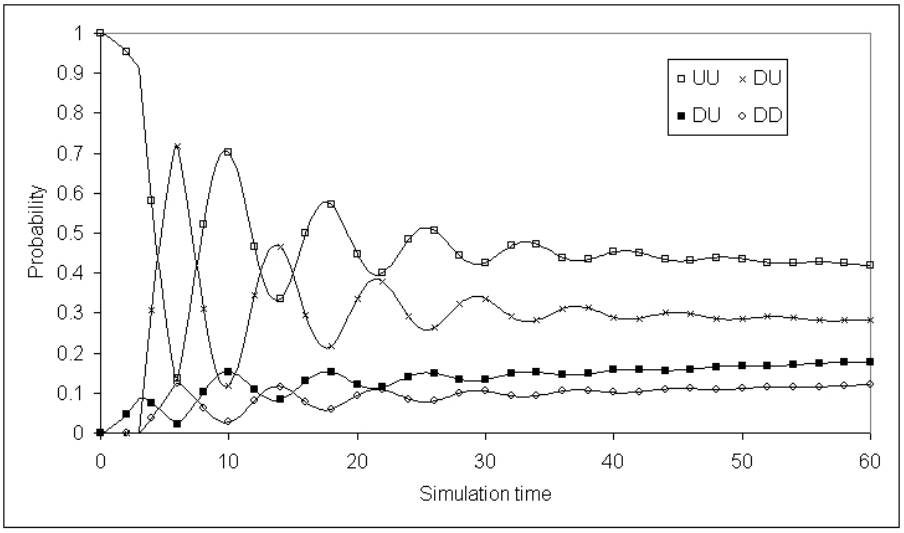


Figure 6.5: Transient probabilities of the four different states

The transient solution of the expected work accomplished is shown in Figure 6.7, which is the integral of the transient performability solution, as shown in Figure 6.6. In the figure it is visible how the expected work increases almost linearly, which is because of the fact that the transient performability decreases very slowly.

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

The number of proxels that are generated and the computation time for the experiment are as follows:

Number of Proxels	10 160 034
Computation Time	ca. 49s

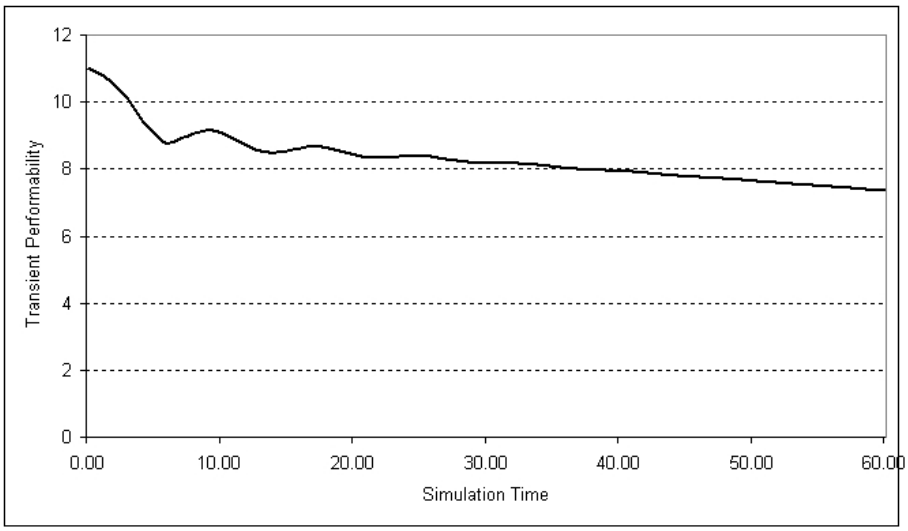


Figure 6.6: Transient performability of the model

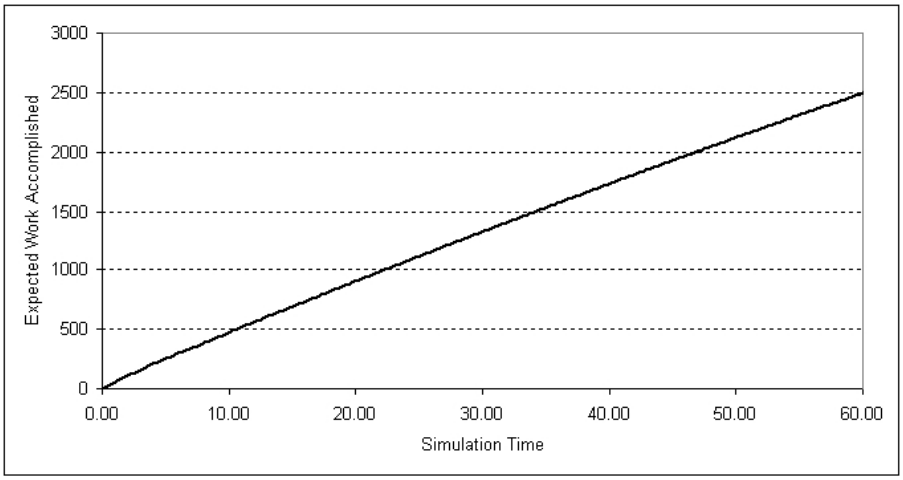


Figure 6.7: Expected work accomplished of the model

which is an acceptable computation time regarding the smoothness and clearness of the obtained solutions.

The model that we analysed is an interesting case because it involves discrete parameters for controlling the performance of the system. If the model was to be analysed using a PDE approach, the complexity of the computational model would have been increased significantly with the introduction of every additional dependence. The DES approach would have also needed a large number of replications to deliver such a smooth transient solution as the one shown in Figure 6.6. This model is also not bounded, which demonstrates another advantage of the method i.e. the ability to build the state space on-the-fly.

6.1.4 Conclusions

Performability modelling represents one application of the proxel-based method, which is introduced and formalised in this section. Because of the flexibility of the proxel-based method and the proxel definition, the method is easily extended to include additional variables for tracking different relevant quantities. Therefore, the treatment of both impulse and rate rewards did not represent an additional load, but instead fitted straightforward in the existing framework.

6.2 Proxel-Based Simulation of a Warranty Model

The analysis of stochastic models is a common task in reliability modelling. The problem of long and expensive simulations is always present and thus also the need for faster and cheaper approaches. Models which are stiff, or whose measures have a large variance, require a large number of replications when Monte Carlo simulations are used; the model considered in this paper required 20 to 30 hours of computation time. Since the proxel-based method is effectively a discretisation of a system of differential equations, it provides a much more controlled convergence towards the solution as the computation progresses. This fact was exploited in the warranty analysis model to achieve results of comparable accuracy in a time range from a few seconds up to a few minutes (Lazarova-Molnar and Horton 2004).

In this section the warranty model that was studied is described and results of an experiment are provided, which illustrate the behaviour of the method and the application of a corresponding software tool which was designed to automate the analysis.

6.2.1 Description of the Warranty Model

The model that was analysed was used to predict the warranty costs for automobiles using different warranty strategies. These strategies contained a race condition between a time-based and a mileage-based expiration threshold, whereby the simulation time unit t is measured in miles, and physical time is converted into an equivalent number of miles. Figure 6.8 shows a stochastic Petri net of this model.

The goal of the analysis of this model is to predict the manufacturer's costs for different types of vehicles and analyse different warranty strategies. During the warranty period, the manufacturer incurs costs whenever the vehicle fails and must be repaired. As shown in Figure 6.8, the warranty runs out whenever one of the two conditions is satisfied: either the warranty period has run out or the warranty mileage is reached. The costs are a decision factor in constructing the warranty strategy.

The discrete-event simulation of this model resulted in long computation times, owing partly to the rarity of the failures, but mostly to the large impulse reward i.e. the cost of a failure. The large reward combined with the rarity of failures creates a big variation

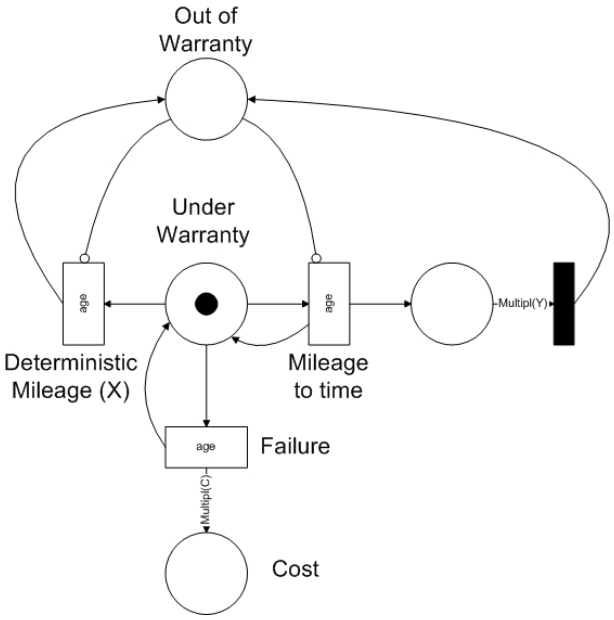


Figure 6.8: Petri net of the warranty analysis model

in the results. The proxel-based method, as previously mentioned, is less sensitive to the stiffness of models and results in much shorter computation times. Because of its deterministic nature, the solutions obtained with the proxel-based method can clearly show trends in the behaviour of one model.

The model is analysed using the following parameters:

- Y- the number of years under warranty,
- X - the mileage under warranty, and
- C - average costs per failure,

and the following distribution functions:

- $f()$ - failure distribution function, and
- $g()$ - time to mileage distribution function.

In the proxel-based simulation, the number of years is included in the state vector in the same manner as the age intensities are; therefore the model results in only two discrete states, "*under warranty*" i.e. U and "*out of warranty*" i.e. O. The state vector in the general case has the following form:

(discrete state, age intensity of the failure transition, number of years passed),

with the initial state being:

$$(U, 0, 0).$$

The things that could happen in the next time step are:

(1) one year could have passed, which probability can be computed from the instantaneous rate function of g, or

(2) failure could have happened, which probability can be computed from the instantaneous rate function of f, and the costs are correspondingly incremented, or

(3) the global simulation time, which is the mileage of the car, could be reached X, which would end the simulation, or

(4) nothing, for which the probability is 1.0 minus the probabilities of everything else happening, whereby the age intensities are incremented correspondingly.

The corresponding states are the following:

(1) $(U/O, \Delta t, 1)$,

(2) $(U, 0, 0)$,

(3) $(O, \Delta t, 0)$, and

(4) $(U, \Delta t, 0)$.

The discrete state in case (1) can either be U or O, depending on the parameter Y which states the number of years that the vehicle is under warranty. If the discrete state is O, then the simulation ends just like in the case (3).

The proxel-based simulation traces the flow of probability and the accumulation of costs. The goal is to provide an approximation for the expected costs. The approximations differ in their accuracy based on the size of the time step being used. The solution values, however, converge monotonically towards the true solution as the size of the time step decreases. This makes it possible to perform an extrapolation of solution values obtained with larger time steps and thus obtain a better approximation. The fact that the simulation can be run using larger time steps means an enormous saving of computation time.

6.2.2 Experiments

In the following we provide an example computation for the warranty analysis model. For this purpose we use the following (fictitious) values for the required parameters:

- $C = 1000$ \$
- $f \sim Exponential(0.00001 \text{ per mile})$
- $g \sim Weibull(4500 \text{ miles}, 1.5)$

Table 6.1: Values for the warranty model strategy

X (miles)	5000	10000	15000	20000
Y (years)	1	2	3	4

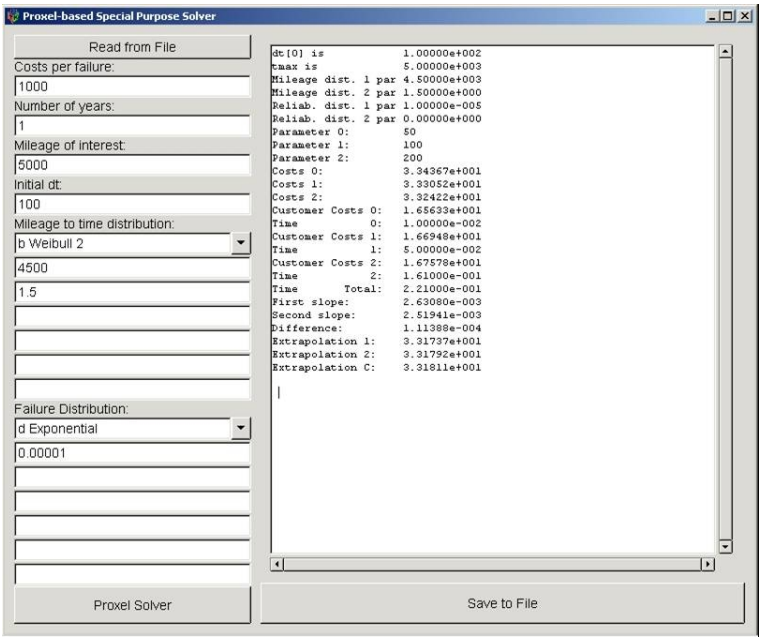


Figure 6.9: GUI of the proxel-based special-purpose solver

The corresponding values for X and Y are given in Table 6.1.

The user interface of the special-purpose proxel-based tool for the simulation using the first values of X and Y is shown in Figure 6.9. It allowed an automated and faster analysis of the warranty strategies.

Using an initial value of 100 for Δt, the approximations for the costs are given in Figure 6.10. In the same figure, the converging values of the costs for $\Delta t = 100$, 50, 25 are obtained using proxel-based simulation, whereas the value for $\Delta t = 0$ is obtained using extrapolation. The overall computation time for this simulation was 0.24 seconds. In Figure 6.11 the results of the same simulation are shown, using an initial time step of 300. The computation time was 1.02 seconds. This illustrates how the initial size of the time step influences the solution. Even though the directly computed solutions are quite different, the extrapolated solutions are very similar. Thereby, the computation time differs with a factor greater than four. When running more complex models, simulating a longer mileage or a period of time, the difference can be much more significant. This shows that the extrapolation can increase the accuracy of the solutions at a much lower cost.

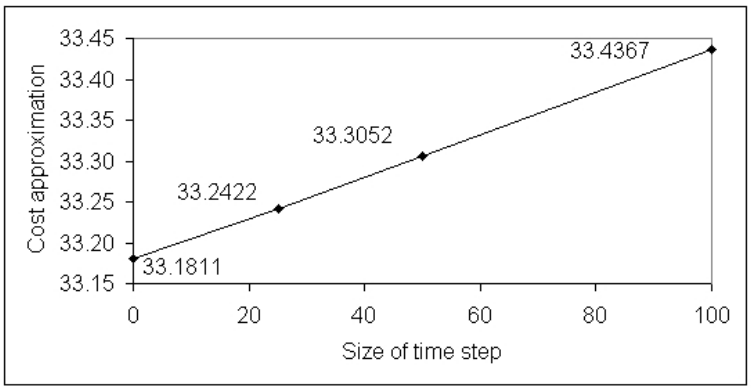


Figure 6.10: Costs approximations using different time steps and extrapolation

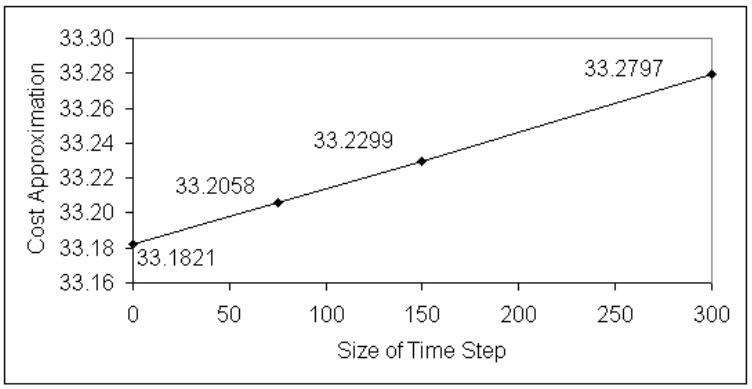


Figure 6.11: Costs approximations using different time steps and extrapolation

From Tables 6.2 and 6.3, it can be seen that the saving in computation time when using larger time step is large and the accuracy affected only a little. This also proves that the proxel-based method is very flexible and especially suitable when a rough approximation is needed quickly.

From the cost analysis of the model, it could be discovered how the costs vary for the four different cases (values of X and Y). In the example presented, the change of the costs is linear, as shown in Figure 6.12, which is to be expected, since the failure distribution is exponential, i.e. it has a constant instantaneous rate function.

Table 6.2: Estimate for the cost using initial $\Delta t = 500$ miles

X (miles)	5000	10000	15000	20000
Y (years)	1	2	3	4
Costs ($)	33.1675	72.5313	112.7430	153.3080
Comp. Time (s)	0.020	1.462	11.456	46.967

Table 6.3: Estimate for the cost using initial $\Delta t = 1000$ miles

X (miles)	5000	10000	15000	20000
Y (years)	1	2	3	4
Costs ($)	33.1442	72.4817	112.6660	153.2040
Comp. Time (s)	0.010	0.160	1.212	4.757

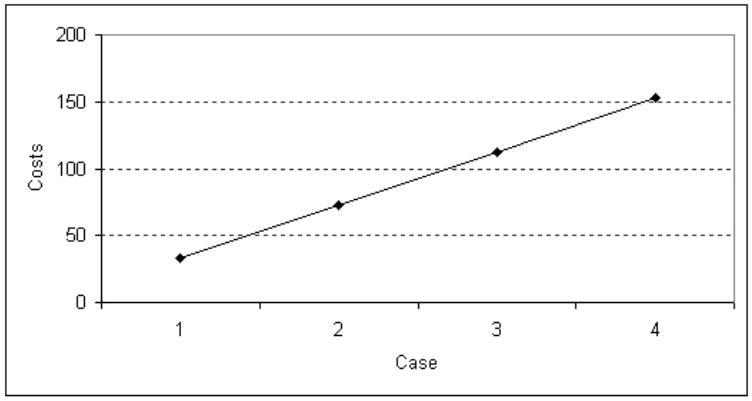


Figure 6.12: Solution values for the costs regarding the different warranty strategies

6.2.3 Conclusions

This positive experience with the proxel-based method opened the doors to our further cooperation with DaimlerChrysler with respect to analysing their models with the proxel-based method. The short computation times are significant from the point of view of providing finer analysis of the costs within reasonable amounts of time, which was almost impossible using the discrete-event simulation approach. The latter approach was only able to do rough prediction of the costs for predetermined coarse points, where each simulation was very expensive, with duration of about 20-30 hours. In a time significantly less than that, the proxel-based simulation could provide a more accurate smooth prediction and therefore provide a better decision concerning the warranty strategy.

This is, however, only one way in which the proxel-based simulation is more efficient than the discrete-event simulation approach. Another one would be getting rough approximations of the costs within even smaller amounts of time. The flexibility of the method makes it possible to compromise the accuracy with the computation time when only an approximate idea of the solutions is needed. However, the sacrifice in terms of accuracy is not too big as it was also shown by the experiments.

The success of the approach has also led us to construct a proxel-based simulation tool for analysing the times to failure for non-trivial basic events in fault trees, which

approach is discussed in the next Section 6.3. These models are small, but a high degree of accuracy is required; these are ideal conditions for the proxel method.

6.3 Quantitative Evaluation of Fault Trees Using Proxels

Fault trees are a very popular tool for describing the failure behaviour of complex technical products and systems (Vesely et al. 1981). One of its common uses is estimating computer systems reliability, as treated in (Apthorpe 2001; Manian et al. 1998a; Kaiser and Gramlich 2004). The basic elements of fault trees are usually failures of different components of one system, which combination of failures determines the failure of the system as a whole. Therefore, these basic elements are usually described by one single event which most often is a failure of a component, and therefore referred to as *basic events*.

We are interested in the case where the basic events can themselves be small simulation models with non-trivial time-dependent behaviour (Lazarova-Molnar and Horton 2003b). 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 discrete-event simulation approach (Dutuit et al. 1997) and give some arguments in its favour. Computational experiments are presented to evaluate the method's usefulness and demonstrate how it addresses the problem.

6.3.1 What is a Fault Tree?

In complicated systems failures result as 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, referred to as a *top event*, 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 referred to as a "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, i.e. the 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 the probability for failure of the system as a function of time.

Fault trees can also be qualitatively analysed, which in practice means determining the minimal *cutsets* for the top event (Lazarova-Molnar and Horton 2003b). Cutsets define the sets of events, which if all occur, can lead to system failure. A cutset can be one basic event, or a combination of more of them. A *minimal cutset* is the smallest set of events that causes the top event to occur. Cutsets with the smallest number of events

have the highest probability of happening, which is the reason why their computation is important. The knowledge obtained via qualitative analysis is further used for computing the availability and reliability measures of the system during quantitative analysis.

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 6.13. The system fails if one of the components C and D fail, and both of A and B. A and B are repairable components, whereby A has an intermediate state before it fails, which has a random duration described by a known random variable. C and D are two copies of a non-repairable component. 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:

$$(\text{A and B}) \text{ and } (\text{C or D}),$$

and its probability is computed using simple probability algebra functions on the probabilities of the basic events (Helstrom 1984).

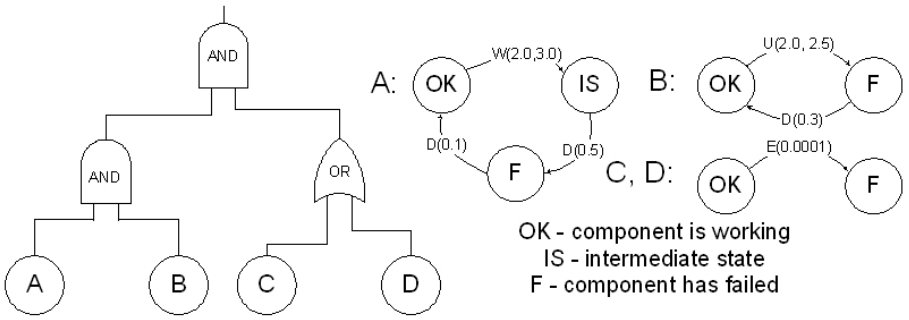


Figure 6.13: Example fault tree with the state-transition diagrams of the basic events

6.3.2 Quantitative Evaluation of Fault Trees and Proxels

When each basic event is a simple failure, as in cases C and D in Figure 6.13, 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 explain how proxel-based simulation can be used to solve that problem. An alternative supplementary-variable based approach for models with only one active age intensity variable is implemented for example in the tool TimeNET (Zimmermann et al. 2000). Another very popular and advanced tool for fault-tree analysis is Galileo (Sullivan et al. 1999), which

analyses fault trees using combinatorial methods based on binary decision diagrams (BDDs) (Akers 1978; Bryant 1986) and Markov methods. Examples for other more popular fault-tree tools are SHARPE which uses a combination of methods (Sahner and Trivedi 1996), MCI-HARP which uses discrete-event simulation for analysing non-Markovian processes (Boyd and Bavuso 1993), and DIFtree, which extends both the BDD and Markov analytical approaches and incorporates discrete-event simulation as well (Manian et al. 1998b).

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.

Example

The fault tree that we use in order to explain how the proxel-based analysis work is the one presented in Figure 6.13. In Figure 6.14, the proxel-based simulation results of the basic events are shown. The simulation was carried out using a time step $\Delta t = 0.01$, up to time $t = 50$.

In Figure 6.15, which illustrates the transient solution of the top event, it is evident that the proxel-based fault tree analysis provides the probability of the top event as a function of time, and is able to trace the change of probability on a very detailed level. The probability distribution of the top event is calculated from the ones of the basic event using the following expression:

$$(Pr(A)Pr(B))(1 - (1 - Pr(C))(1 - Pr(D))),$$

which is constructed using basic probability algebra (Helstrom 1984; Trivedi 2002). In Figure 6.15 it is noticeable how the probability of the top event increases and decreases, the reason for which is the fact that there are repairable components in the system which fail and get repaired.

6.3.3 Conclusions

The proxel-based analysis of fault-trees provides deterministic and accurate results. The biggest advantage of the method is that it does not deliver just one probability

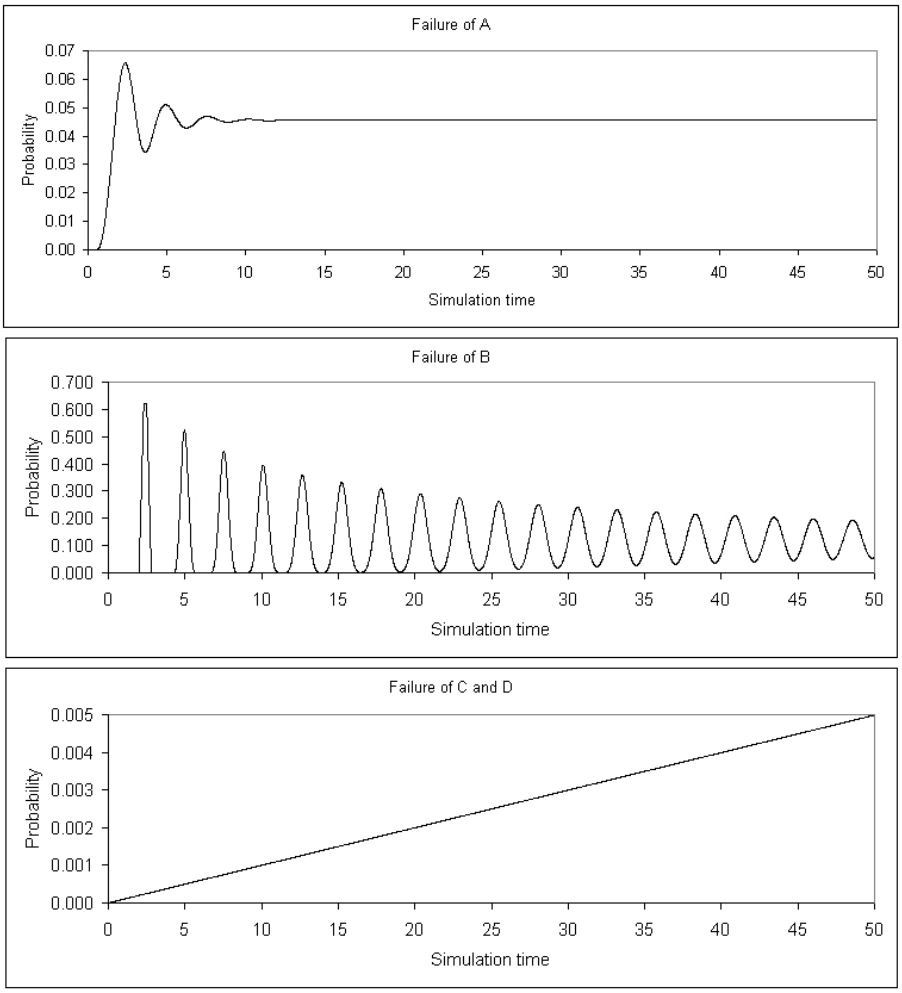


Figure 6.14: Results from the proxel-based simulation of the basic events

value stating how reliable one system is, but it delivers results in form of a function, which allows a higher level of detail and more insight into the behaviour of the top event in a relatively short computation time than what a DES would provide, as shown in Figure 6.15. This allows a closer and more detailed observation of the behaviour of the whole system, as a function of its components. The proxel-based 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 (Lazarova-Molnar and Horton 2003a,c). 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. Along with the experiments, a general fault-tree analysis tool was developed, which provided more general models to be analysed.

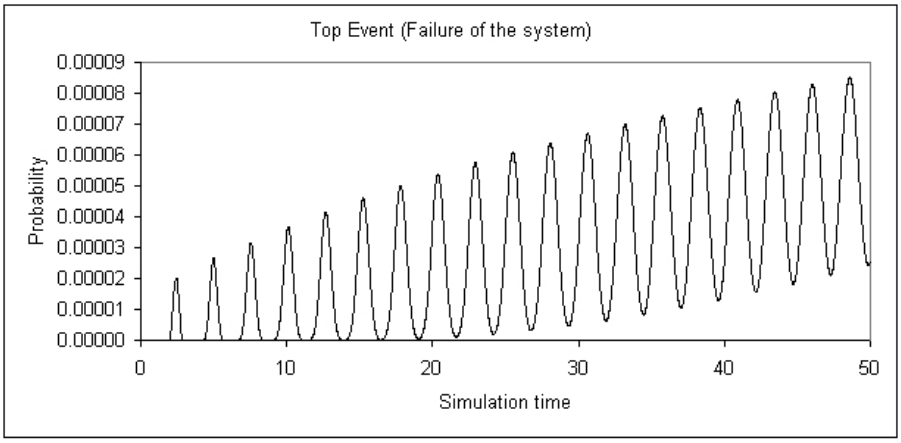


Figure 6.15: Results from proxel-based simulation of the top event

6.4 Analysing Stochastic Petri Nets Using Proxels

In this section we describe the proxel-based algorithm when the underlying stochastic process is described by means of SPNs. Special attention must be given to the age memory policy transitions, as well as the immediate ones, which are basically the elements that make the difference when comparing SPNs to standard state-transition diagrams, which elements are, however, an advantage when it comes to modelling because they simplify the process significantly. The formal description of SPNs that we use in this section is the one provided in Section 2.1.3.

A proxel Px is then defined as follows:

$$Px = (S, t, R, Pr).$$

$S = (m, \vec{\tau})$ is the *state of the model*, where m is the marking and $\vec{\tau}$ is the *age intensity vector* which contains the elapsed enabling times of a set of timed transitions. t is the global simulation time and Pr denotes the probability that the model is in state S at time t, given that it has been reached through the sequence of states that represent tangible markings, $R = (S_1, S_2, \ldots, S_s)$. The null sequence is denoted by $\oslash = ()$. The age intensity vector $\vec{\tau}$ is needed for a complete definition of the state of the Petri net, because the firing rate of each timed transition is dependent on how long it has been enabled. As expected, the instantaneous rate function is used for calculating it. It takes as a parameter τ, which is the enabling time of the timed transition that caused the change from the previous to the current marking, and is computed from the distribution function that is associated with the transition.

For each timed transition $T_i = (F_i, type_i)$ with distribution function F_i and $type_i \neq$ "*immediate*", the IRF $\mu_i(\tau)$ is calculated as shown in Section 2.1.1. The instantaneous rate function is used for calculating the probability with which a timed transition might fire within the time interval $(\tau, \tau + \Delta t)$ if it has been enabled for time τ. We

approximate this probability by $\mu_i(\tau) \times \Delta t$, which can also be exchanged for different numerical integration approaches, as shown in Section 3.2.

Immediate transitions are associated with vanishing markings, therefore the proxels that represent vanishing markings are referred to as *vanishing proxels*. This also applies to the states that represent vanishing markings, they are referred to as *vanishing states*. The model spends zero time in those markings and therefore the vanishing proxel is instantly replaced by its successor proxels without advancing their simulation time component in the proxels. The probability values of the successor proxels of a vanishing proxel are computed as a product of the probability value of the vanishing proxel and the probability that the corresponding immediate transition would fire. The latter is computed using the following formula:

$$Pr(T_i\ fires) = \frac{p_i}{\sum_{T_j\ is\ enabled} p_j},$$

where $T_{i/j} = (p_{i/j}, immediate)$.

As already stated in Section 4.1.1, the proxel-based method can operate on the Petri net itself, without having to construct the reachability graph, building the state-space on-thy-fly.

The formal representation for the case of SPNs of the calculation of the probabilities of the different markings at different time steps as well as the calculation of the proxels' probabilities in the next discrete time step based on the previous one is the following:

$$P(model\ in\ marking\ m\ at\ time\ t) = \sum_{\vec{\tau}:S_k=(m,\vec{\tau})} P(model\ in\ state\ S_k\ at\ time\ t),$$

where

$$P(model\ in\ S_k\ at\ time\ t+\Delta t) = \sum_{i,j:R_j \overset{t/\Delta t}{\rightarrow} S_i} P(model\ in\ S_i\ at\ time\ t) \times \mu_{il}(\tau_{il}) \times \Delta t \times p_{lk},$$

where τ_{il} is the age intensity of the timed transition that caused the state change from S_i to S_l, having μ_{il} as the IRF of the corresponding timed transition, and S_l is an intermediate vanishing state, in which case p_{lk} is the probability with which the relevant immediate transition that is enabled in the vanishing marking fires. If there is not an intermediate vanishing marking, then $S_l = S_k$ and $p_{lk} = 1$. $\mu_{il}(\tau_{il})$ and p_{lk} are zero if the corresponding state changes are not possible.

One important aspect when constructing the proxel tree is dealing with the different memory policies of the timed transitions in the Petri net. Age policy transitions "remember" their activation times when they become disabled owing to another transition

firing. When re-enabled, the clock that measures the time until the transition fires resumes from where it left off. For this reason, every age policy transition may require an additional component in the age intensity vector, which adds to the complexity of the simulation. If M is the set of reachable markings, then the dimension D of the age intensity vector is bounded by:

$$D = \max_{m \in M} \{|enabled\ enabling\ memory\ transitions\ in\ m| + |age\ policy\ transitions|\}$$

The first components of the age intensity vector are the age intensities of all timed transitions that have enabling policy and have been enabled at different points in time. The second component contains the age intensities of the age memory transitions. The values of the variables corresponding to enabled transitions increase with time during which the timed transitions they represent remain enabled. Each of them is set to zero when the corresponding timed transition fires. The age intensities of the currently disabled age memory transitions remain constant until the transitions are once again enabled. There is an algorithm which creates a nearly optimal mapping as presented in (Balaprakash 2004), so that at every point in time only the relevant age policy transitions are represented in the age intensity vector, and at the same time it has the lowest possible dimension.

6.4.1 Example

In this section an example is presented, which shows how the proxel-based method functions when the Petri net contains both types of transitions, i.e. timed and immediate. For this purpose, the Petri net model shown in Figure 6.16 is used. This SPN represents a limited queuing system with two servers, where people choose one when both of them are free, based on a probability value. For simplicity reasons the maximal number of people in the queue was chosen to be two.

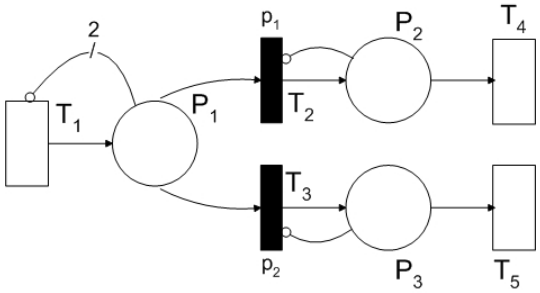


Figure 6.16: Petri net of a queuing system with one limited queue and two servers

The initial marking is $m_0 = (0, 0, 0)$. The five transitions modelled in the Petri net are defined as follows:

- $T_i = (F_i, enabling), i = 1, 4, 5$, and
- $T_i = (p_i, immediate), i = 2, 3$.

Two of the transitions are immediate, and three of them are timed ones. Because of the possibility that all three of the timed transitions are enabled at the same time with different elapsed times, the dimension of the age intensity vector is three. The reachability graph of the Petri net is shown in Figure 6.17. Vanishing markings are shaded, and the arcs representing immediate transitions represented by thicker lines. In Figure 6.18 the tree illustrates all possible developments of the Petri net within two discrete time steps, in terms of markings.

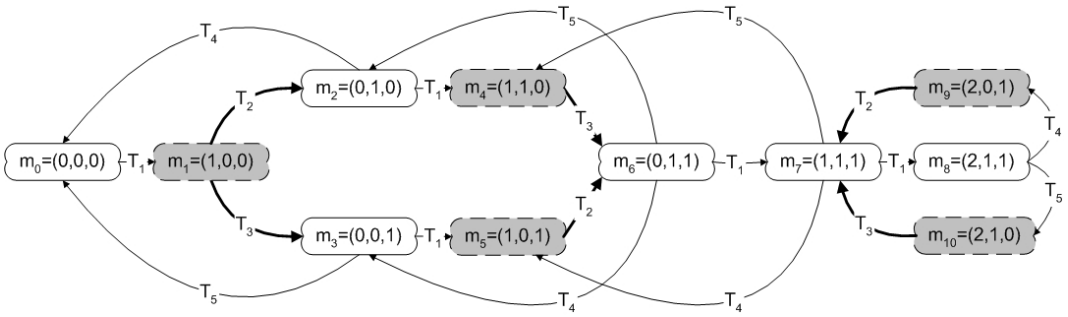


Figure 6.17: Reachability graph of the Petri net from Figure 6.16

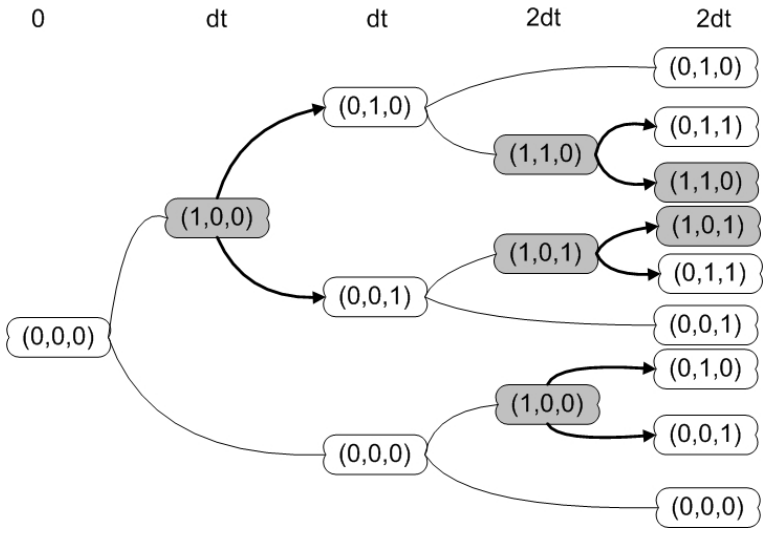


Figure 6.18: The first three levels of the markings tree based on the SPN in Figure 6.16

The age intensity vector complies with the following mapping (T_1, T_4, T_5), i.e. the three timed transitions that have enabling memory policy are consequently mapped to the three components of the age intensity vector. The beginning part of the proxel-tree for this model is presented in Figure 6.19. The proxel analysis operates on the complete reachability graph, without advancing the time in the case of vanishing markings. This

means that whenever the model is in a vanishing marking, the probabilities of the subsequent proxels are instantly calculated by multiplying the probability of the vanishing proxel with the probabilities associated with the relevant immediate transitions. The simulation time during this operation does not advance, as shown in Figure 6.19.

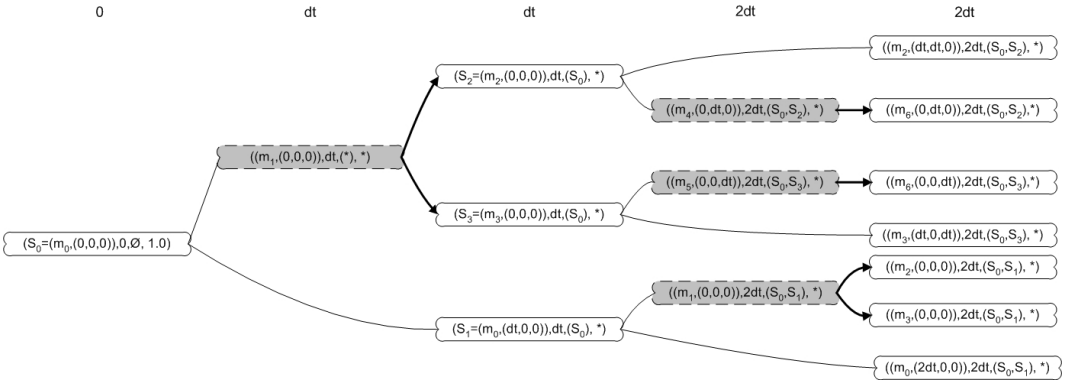


Figure 6.19: The first three levels of the proxel tree based on the SPN in Figure 6.16

The values of the probabilities are not stated explicitly in Figure 6.19, instead they are replaced by asterisks. In Figure 6.20, a result from the proxel-based simulation of this Petri net is presented. The value that is used for Δt is 0.05, and the simulation is run up to time t = 10, i.e. there are 200 discrete time steps. The computation took 1198 seconds on a 1.2 GHz Pentium III notebook. The distribution functions and the probability values associated with the transitions that are used in the simulation are the following:

- $F_1 \sim Uniform(0.1, 2.1), F_4 \sim Uniform(2.0, 3.0), F_5 \sim Exponential(0.5)$
- $p_1 = 0.3, p_2 = 0.7$

From Figure 6.20 one can spot that the model goes into a steady state, as well as the sharp edges of the solution functions, which occur because of the presence of uniformly distributed state changes.

Further follows an output of the implementation of this example which shows the order in which the proxels are processed can be seen, as well as the exact probability values, given the specific parameters for the distribution functions and the probability values. *AddProxel* shows the proxels that are generated and stored into the proxel tree, *Processing* shows a proxel that is currently being processed i.e. taken from the proxel tree and its successor proxels are being generated. *Vanishing* shows the vanishing proxels that are being processed i.e. their successor proxels being stored. Only the proxels with nonzero probability values are actually stored. That explains why the size of the tree in the first three steps is one. Because of the uniformly distributed transition T_1 on (0.1, 2.1), which is the only one enabled in the initial marking m_0, the state change can happen only after $t = 0.1 = 2\Delta t$, as confirmed by the output of our program. It can also be spotted that always when a vanishing proxel is generated (in this case marking m_1), it instantly triggers its successor-proxels (representing markings

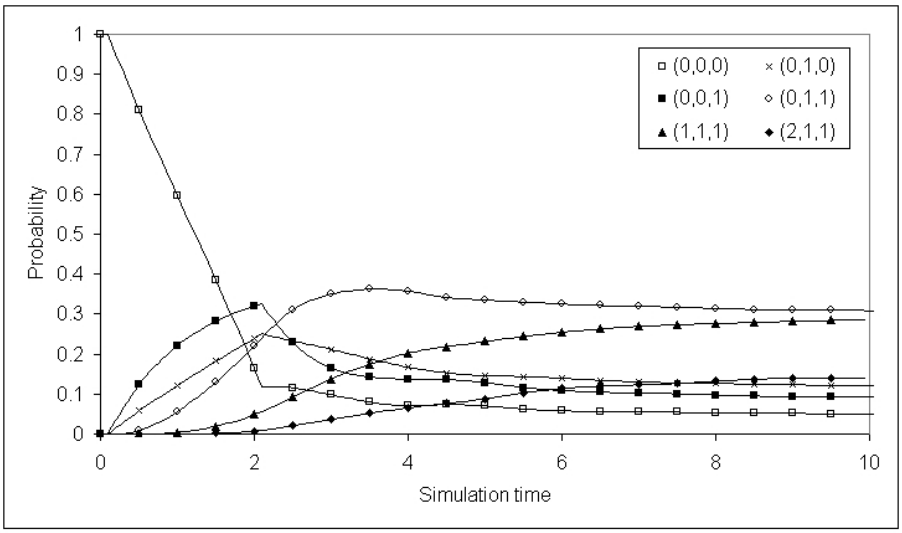


Figure 6.20: Solution of the example SPN, obtained by proxel-based simulation

m_2 and m_3) and distributes its probability to them, which in STEP 3 is most noticeable because of the non-zero probability.

```
AddProxel  ((m0, ( 0dt,  0dt,  0dt)),  0dt, Route, 1.000000)

STEP 1
Size of tree 1

Processing ((m0, ( 0dt,  0dt,  0dt)),  1dt, Route, 1.000000)

Vanishing  ((m1, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel  ((m2, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel  ((m3, ( 0dt,  0dt,  0dt)),  1dt, Route, 0.000000)
AddProxel  ((m0, ( 1dt,  0dt,  0dt)),  1dt, Route, 1.000000)

STEP 2
Size of tree 1

Processing ((m0, ( 1dt,  0dt,  0dt)),  2dt, Route, 1.000000)

Vanishing  ((m1, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel  ((m2, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel  ((m3, ( 0dt,  0dt,  0dt)),  2dt, Route, 0.000000)
AddProxel  ((m0, ( 2dt,  0dt,  0dt)),  2dt, Route, 1.000000)

STEP 3
Size of tree 1

Processing ((m0, ( 2dt,  0dt,  0dt)),  3dt, Route, 1.000000)
```

```
Vanishing  ((m1, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.025000)
AddProxel  ((m2, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.007500)
AddProxel  ((m3, ( 0dt,  0dt,  0dt)),  3dt, Route, 0.017500)
AddProxel  ((m0, ( 3dt,  0dt,  0dt)),  3dt, Route, 0.975000)
```

6.4.2 Conclusions

Proxel-based method can be successfully applied for analysis of stochastic Petri nets which contain generally distributed transitions. The fact that the proxel-based method builds the state-space on-the-fly, makes it directly applicable on a Petri net itself, without having to build the reachability graph beforehand. This makes the method very suitable for analysing Petri nets with unbounded state-space. In order to optimise a Petri net for input to the proxel-based simulator, our proxel-adapted modelling framework that is described in Chapter 5 can be used as an intermediate step and measure of how to adjust the Petri net for more efficient proxel-based analysis.

6.5 Hybrid Models and Proxels

As stated in (Ciardo et al. 1999), hybrid models have gained increasing attention because of the employment of digital controllers and elements in environments that are typically analog, such as power generators, chemical plants, or water distribution systems. Hybrid stochastic Petri nets (HSPNs) and Fluid stochastic Petri nets (FSPNs), as extensions of the SPN formalism allow a more exact modelling of this type of systems because of the addition of fluid (i.e. continuous) elements, i.e. places. For the first time the fluid extension of SPNs is presented in (Trivedi and Kulkarni 1993) for representing continuous state variables along with the discrete ones. It started a series of research on the topic of FSPNs and their analysis, as well as increasing the complexity of the models being analysed by introducing dependencies between the continuous and the discrete part, as treated in (Horton et al. 1996) and (Ciardo et al. 1999). The latter one deals with more complex models which can only be solved by means of discrete-event simulation. In the meantime, a parallel research of an extension of SPNs by continuous places under the name of hybrid SPNs (HSPNs) is described and defined in (Alla and David 1998). In this section we will briefly describe the main properties of the hybrid models and show how they can be analysed using the proxel-based method.

6.5.1 What is a Hybrid Model?

Hybrid models are structures which at the same time incorporate discrete, as well as continuous quantities. When simulated by computers, they are all ultimately discretised. Therefore, the advantage in using hybrid models essentially consists in the modelling comfort, i.e. representing the real systems as intuitively as possible.

HSPNs are an extension to the formalism of SPNs that allows modelling of continuous quantities along with the discrete ones. It contains the so-called *fluid* type of a place,

which models the continuous quantities. Correspondingly, a new type of arc is defined which allows flowing from and to the fluid places according to the rates associated with it.

For our proxel-based hybrid models' analysis we use the typical place transition diagram in which we allow for continuous states, which act similarly as the fluid places in the HSPNs. More insight is given in the next Section 6.5.2 which elaborates one example hybrid model, and at the same time explains how our hybrid state-transition diagram functions.

6.5.2 Proxel-Based Analysis of Hybrid Models

Because the numerical solution of hybrid models turns into an unpleasant and very complicated task, besides discrete-event simulation which is described in (Ciardo et al. 1999)), we propose the proxel-based method as an alternative for at least some classes of hybrid models, i.e. the less complex ones.

In order to carry out proxel-based simulation of hybrid stochastic models, a discretisation of the continuous (i.e. fluid) elements should be performed prior to the simulation for efficiency reasons. It results into having two discretisation steps: Δt for the simulation time, and Δf for the continuous part of the state, or the fluid elements.

As an example for demonstrating how proxels can be employed in analysing hybrid models, we use the model shown in Figure 6.21. The model is described using a simple state-transition diagram and can be interpreted as a water tap with a sink. The tap has two discrete states: ON and OFF. When the tap is in the discrete state ON then the water flows into the sink which represents the continuous state, from where it flows out in the both discrete states: ON and OFF. The rates with which the water flows in and out are denoted by φ_i, and the state changes by SC_i.

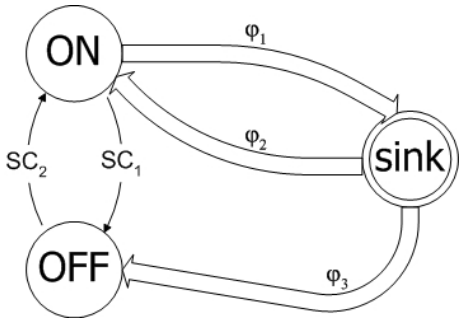


Figure 6.21: Example hybrid model of a water tap

The proxel structure for the model shown in Figure 6.21 is the following:

$$Proxel = (State, Time, Route, Probability), \text{ where } State = (DS, Q_{sink}, \tau)$$

where

- $DS \in \{ON, OFF\}$ is the discrete part of the state,
- Q_{sink} is the water quantity in the sink, and
- τ is the age intensity of SC_1 when in discrete state ON, and age intensity of SC_2 when in OFF.

The other elements, i.e. *Time, Route,* and *Probability* have the standard meanings as described in Chapter 3. Accordingly, the initial proxel is then $((OFF, 0, 0), 0, \oslash, 1.0)$

This model can be analysed by the proxel-based method using the Algorithm 6.2, where the proxel structure is simplified to contain only the state and its probability, which for the initial proxel means $((OFF, 0, 0), 1.0)$. The algorithm is an extension of the basic Algorithm 3.1, presented in Section 3.1.3. The water quantity is treated as another supplementary variable which requires a discretisation step too, denoted by Δf, in order to avoid having an infinite number of possible values, i.e. states. The rates with which it flows can be defined as functions of the age intensity too. The algorithm works in the same way as the basic one, except that now the water quantity according to the flowing rates is updated too, as shown in Functions ProcessDS-OFF and ProcessDS-ON in line 2 correspondingly.

Algorithm 6.2: Proxel-Based Hybrid Model Analysis

```
   Input: Δt, Δf, t_max
 1 Initialize proxel set PS(0) by inserting the initial proxel;
 2 switch = 0;
 3 for i=1 to [t_max/Δt] do
 4 |   foreach proxel p = ((DS, Q_sink, τ), prob) in the proxel set PS(switch) do
 5 |   |   switch the value of DS do
 6 |   |   |   case OFF
 7 |   |   |   |   ProcessDS(OFF);
 8 |   |   |   end
 9 |   |   |   case ON
10 |   |   |   |   ProcessDS(ON);
11 |   |   |   end
12 |   |   end
13 |   end
14 |   switch = 1 − switch
15 end
```

Example results from the proxel-based analysis up to time $t = 5.0$, using $\Delta t = 0.1$ and $\Delta f = 0.1$ are shown in Figure 6.22. There we observe separately the solutions for the state *ON* and for the state *OFF*. First there are two three-dimensional plots which show the probabilities for each time step and each water quantity discretisation step for both states. Bellow there are the solutions for having an empty sink in each of the states, and the water quantity probabilities at time $t = 5.0$. Each of the results obtained for every water discretisation step represent probabilities for a range of water

Function `ProcessDS-OFF`

Compute probability $prob_{calculated} = IRF(SC_2, \tau) \times \Delta t$;
Compute water quantity $Q_{sink}^{new} = max\{Q_{sink} - \varphi_3(\tau) \times \Delta t, 0\}$;
Discretise $Q_{sink}^{new} = \lfloor Q_{sink}^{new} / \Delta f \rfloor \times \Delta f$;
Generate new state $S = (ON, 0, Q_{sink}^{new})$;
Search for the state S in the states of the set of generated proxels;
if *$proxel_{found}$ is the found proxel* **then**
    $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$
end
Generate new proxel $p_{new} = (S, prob_{calculated})$;
Store p_{new} in $PS(1 - switch)$;
Generate new state $S = (OFF, \tau + \Delta t, Q_{sink}^{new})$;
Generate new proxel $p_{new} = (S, prob - prob_{calculated})$;
Store p_{new} in $PS(1 - switch)$;

Function `ProcessDS-ON`

Compute probability $prob_{calculated} = IRF(SC_1, \tau) \times \Delta t$;
Compute water quantity $Q_{sink}^{new} = max\{Q_{sink} - \varphi_3(\tau) \times \Delta t, 0\}$;
Discretise $Q_{sink}^{new} = max\{Q_{sink} - (\varphi_1(\tau) - \varphi_2(\tau)) \times \Delta t, 0\}$;
Generate new state $S = (OFF, 0, Q_{sink}^{new})$;
Search for the state S in the states of the set of generated proxels;
if *$proxel_{found}$ is the found proxel* **then**
    $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$
end
Generate new proxel $p_{new} = (S, prob_{calculated})$;
Store p_{new} in $PS(1 - switch)$;
Generate new state $S = (ON, \tau + \Delta t, Q_{sink}^{new})$;
Generate new proxel $p_{new} = (S, prob - prob_{calculated})$;
Store p_{new} in $PS(1 - switch)$;

quantity, i.e. the probability for every $k \times \Delta f$ is in fact a probability for water quantity range $((k - 0.5) \times \Delta f, (k + 0.5) \times \Delta f)$. The smaller Δf, the smoother and better the approximation of the solutions will be.

6.5.3 Conclusions

The algorithm presented here demonstrates an idea of how hybrid models can be analysed using proxels. The problem, however, is very complex and computationally very expensive. This is a subject which is treated in a Master thesis carried out at our department, completed by Lakshmi Devi Baskar (Baskar 2005) and the main conclusion from that work is that the proxel-based method is appropriate only for relatively small hybrid models.

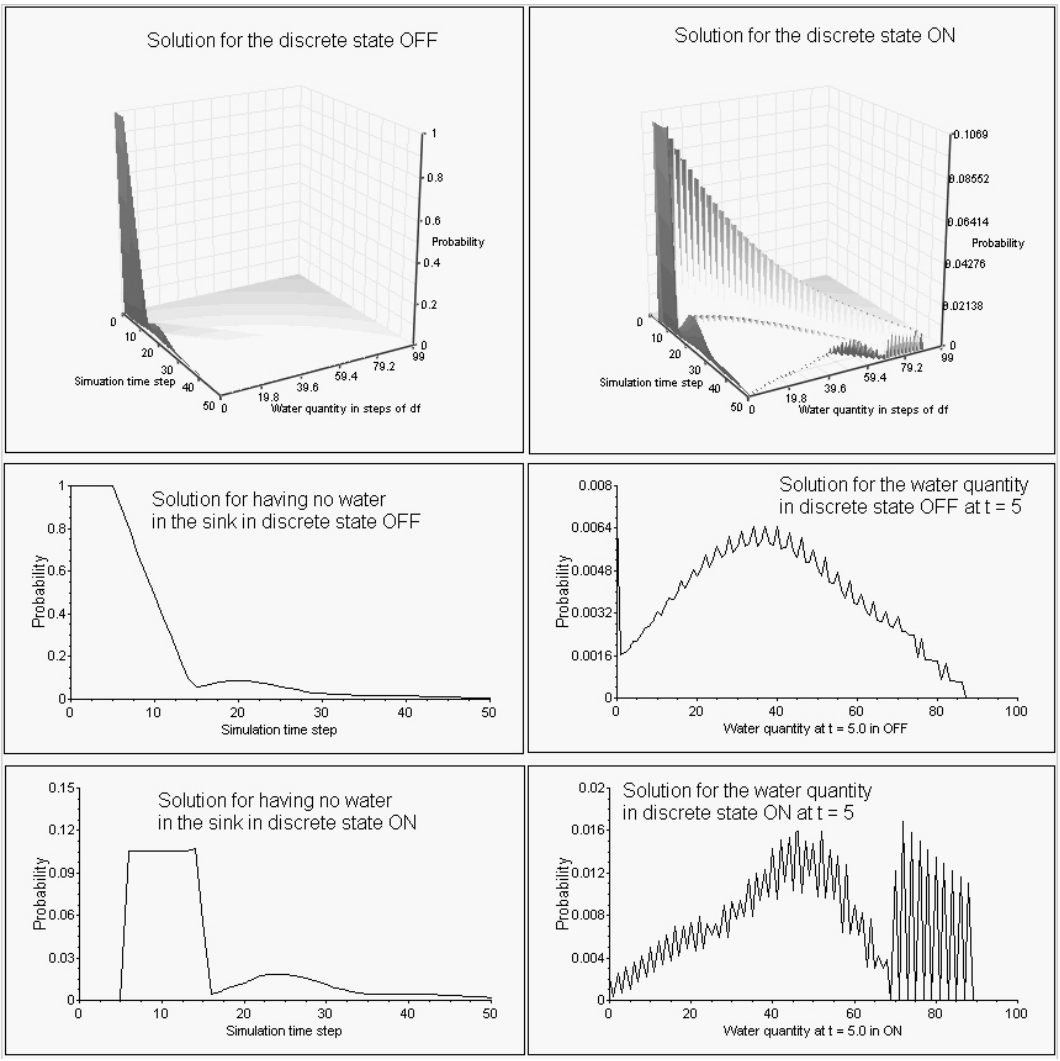


Figure 6.22: Results from the proxel-based solution of the hybrid model shown in Figure 6.21

6.6 Proxel-Based Rare-Event Simulation

As already discussed in Section 4.1.3, rare-event models are an interesting class of models because the proxel-based method is less sensitive to the difference in rates within one model, and provides fair treatment to all events. On the other hand, rare-event models require long computation times when using discrete-event simulation for their analysis (Görg et al. 2001). In a models that contains a rare event it can take a large number of replications in the discrete-event simulation for that event to occur. From that point of view, the proxel-based method is seen as an especially suitable for analysing rare-event models. The proxel-based simulation computes the rare events along with all the others, yielding equal importance to all events. In order to demonstrate this feature of the proxel-based method we use an example model

which contains two rare events and analyse it using both DES and the proxel-based method.

There are many methods which aim at overcoming the problem known as "rare-event simulation". Two of the more popular ones are RESTART (Kahn and Marshall 1953; Kelling 1996) and *importance sampling* (Glynn and Iglehart 1989). They are both based on discrete-event simulation. The basic idea of RESTART is by partitioning the state space into a series of nested subsets to observe the rare-event as an intersection of a nested sequence of events. Importance sampling technique is also known as one of the variance-reduction techniques in Monte Carlo methods, and is based on the idea of scaling the occurrence of the rare-events such that they occur more frequently, thereby speeding up the simulation. The comparison of both approaches is presented in (Garvels 2000).

In general, it is very difficult to talk about comparison of solutions obtained using discrete-event simulation and ones obtained using proxel-based simulation because the quality of the results is different. Discrete-event simulation relies on random numbers and because of that the results will always be stochastic in nature and depend on the quality of the random number generator. On the other hand, the proxel-based simulation is completely free of random behaviour. Therefore, the quality of the results depends only on the size of the time step, making it more analogous to the discretised PDE approach.

6.6.1 Example

In Figure 6.23, a model with two rare events is presented. There are four exponentially distributed state changes with rate parameters equal to 1, and two with rate parameters equal to 0.0001. The model is in state DSS1 or DSS2 most of the time, and very rarely changes to the discrete state DSS3, from where it also most often transits back to DSS2, and only very rarely to DSS4.

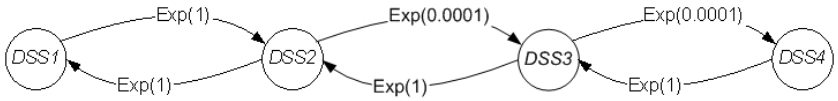


Figure 6.23: Example model for rare events

The model in Figure 6.23 was analysed using the discrete-event simulation package SIMPLEX3 (Schmidt 2000). After 5000 independent replications, each up to time $t = 10000$, which took about 25 minutes computation time, the simulator did not once visit the discrete state DSS4.

The same model was analysed using proxel-based simulation, using a time step of $\Delta t = 0.05$. The transient solution for DSS4 is presented in Figure 6.24. It can be noticed that somewhere around $t = 7.8$ the probability converges to a value around $4.5e - 9$. The total running time of the program, up to maximum simulation time $t = 50$, was 4.2 seconds.

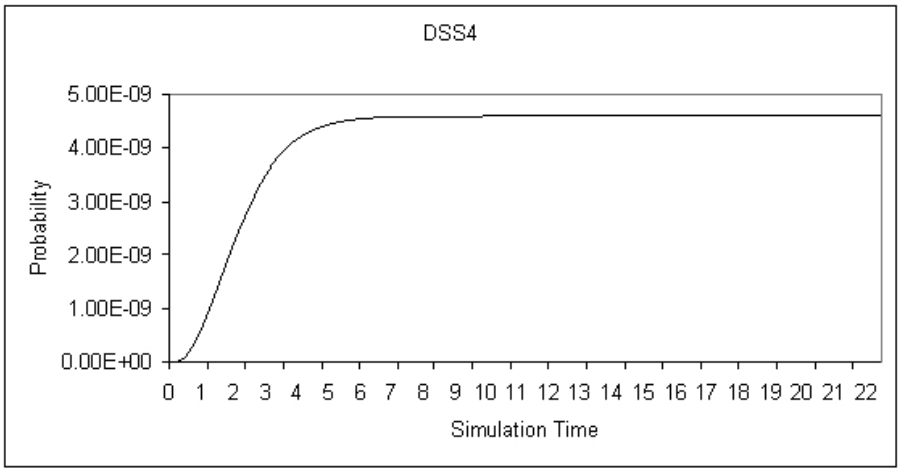


Figure 6.24: Results from proxel-based simulation for discrete state DSS4

These experiments illustrated that the proxel-based simulation can be particularly useful for analysing small, stiff models for which a high level of accuracy is required, such as occur in safety and reliability modelling, and thereby significantly reducing the computation times.

6.6.2 Conclusions

Rare-event simulation is one of the problematic domains in simulation. As stated in Section 4.1.3, there are many methods that approach the problem, and the proxel-based method can be seen as an alternative for approaching this class of models which provides transient solutions for relatively small models - within relatively short computation times.

6.7 Proxels in Tuning Systems

In this section we present a rather speculative idea of another application of the proxel-based method, which still needs to be further researched. That is the use of the proxel-based method for tuning parameters in different systems.

Because of the fast and flexible analysis of some classes of models, the proxel-based simulation can be especially useful in tuning systems. Proxels are especially suitable for rough and fast evaluation of stochastic models. The extrapolation as an instrument for achieving higher accuracy of the results can also be employed for increasing the quality of the optimisation solutions. This makes the proxel-based method a very suitable tool for performing fast evaluation of systems, thereby ensuring that all events are taken into account. The convergence of the solutions can be used for spotting trends in the results obtained using different time steps.

6.7.1 Example

We use a basic example for illustrating our idea. The model that we analyse describes a system which consists of a storage area (i.e. stock) with a limited capacity which has to be regularly refilled, and from which items are being removed according to a certain distribution function. The parameter that defines the expense of running the system is a threshold of items in the storage. When the threshold is reached, a refill order is issued, which takes a random amount of time to arrive. The model of the system, built using our proxel-adapted modelling framework, is shown in Figure 6.25.

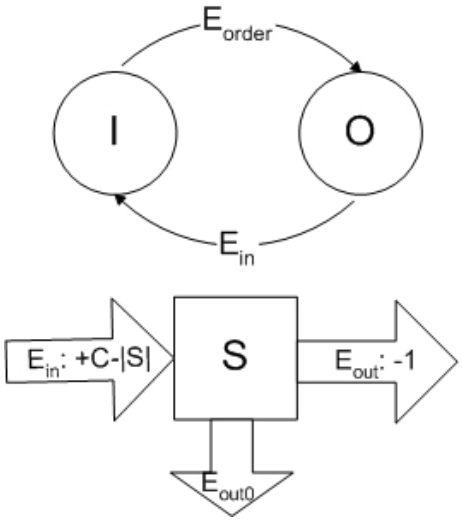


Figure 6.25: Example model for application of the proxel-based method in tuning systems

The model consists of two partial discrete states: I as idle and O as order, for the configuration when the order is issued and it has not yet arrived. Additionally, there is one countable quantity S which tracks the number of items in the stock. Its capacity is denoted by C. The model has four events, as follows:

- E_{in} - order has arrived $\sim F_1(\tau)$,
- E_{out} - one item is removed from the stock when the stock has at least one item $\sim F_2(\tau)$,
- E_{out0} - one item is requested from the stock when it is empty $\sim F_2(\tau)$, and
- E_{order} - order is issued,

which obey the following rules:

- $E_{in} \rightarrow$ (if O then I), $+C - |S|$
- $E_{out} \rightarrow$ (if $S > 0$ then $S - 1$),
- $E_{out0} \rightarrow$ (if $S = 0$ then S),

- $E_{order} \rightarrow$ (if $(I$ & $(S \leq Threshold))$ then $O)$.

Events E_{in}, E_{out}, and E_{out0} are timed, whereas the event E_{order} is a conditional one because it happens as soon as the threshold is reached. The timed events are distributed according to the following distribution functions:

- $E_{in} \sim$ Uniform (3.0, 4.0), and
- E_{out} and $E_{out0} \sim$ Exponential (1.0).

The rewards in the model are the following:

- $rr(O/I) = 10 * |S|$ per day, the cost of storing the items, and
- $ir(E_{out0}) = 1000$, the cost of not being able to provide an item when there is a request.

Our goal is to tune the *Threshold* parameter such that the costs of managing the stock are minimal. We base our decision on observations for a simulation time of 50 days, using an initial size of the time step $\Delta t = 0.4$ days. Further we observe the computation times and the accuracy of the results as the size of the time step decreases.

The discrete state structure of the model is (PDS, S), where $PDS \in \{I, O\}$. The age intensity vector mapping is: in $I \sim (E_{out}, /)$ and in $O \sim (E_{out}, E_{in})$, and correspondingly the initial state is

$$((I, C), (0, /)).$$

The results obtained using different step sizes are presented in Figure 6.26. The threshold range is from 2 to 9, and the stock capacity is 10. The dependency of the computation time on the size of the time step is shown in Figure 6.27, where once again the high increase in computation time when decreasing the time step is shown.

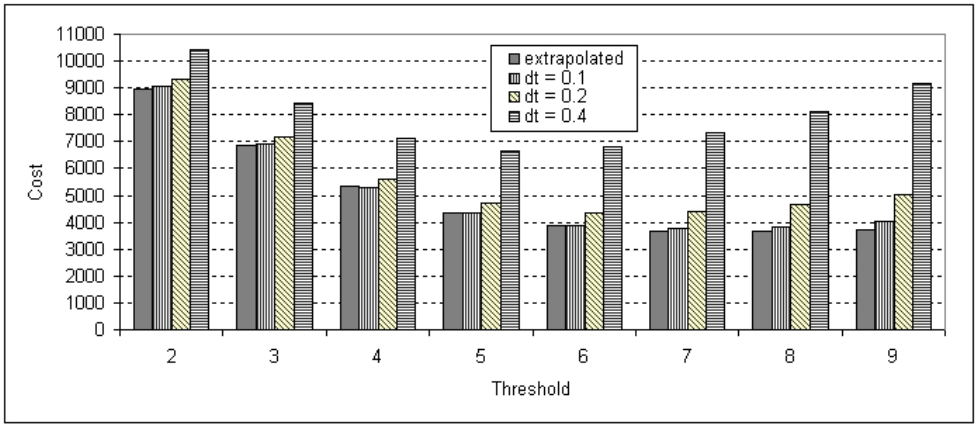


Figure 6.26: Cost as a function of the threshold parameter, using different values for Δt

If one observes Figure 6.26, the trend of the cost regarding the thresholds can be noticed:

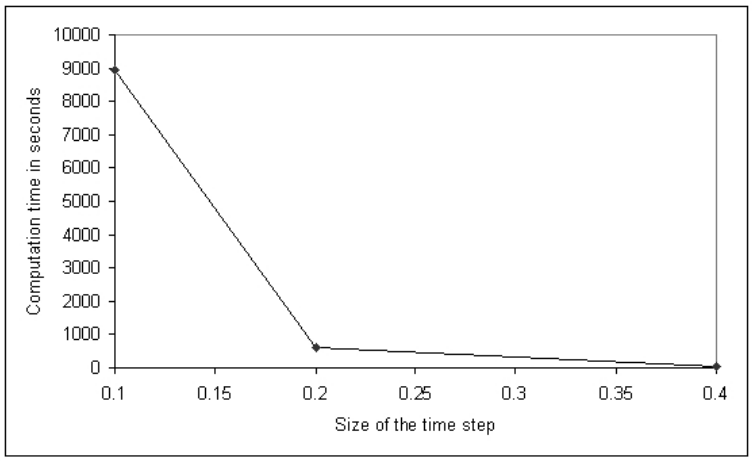


Figure 6.27: Computation time for all of the tuning simulations as a function of the size of the time step

- for $\Delta t = 0.4$ the minimum cost is for a threshold equal to 5,
- for $\Delta t = 0.2$ the minimum cost is for a threshold equal to 6,
- for $\Delta t = 0.1$ the minimum cost is for a threshold equal to 7,

and what is interesting is that the extrapolated solutions yields minimum cost for a threshold equal to 8.

This example demonstrates how the proxel-based method can be applied for tuning parameters of systems for optimising some of its measures (e.g. cost, throughput, performance). Thereby, one can rely that all events are taken into account and every solution is meaningful. For instance, two solutions obtained using different time steps can be used for spotting in which direction the solutions converge i.e. the trends in the solutions. By contrast, because of the stochastic nature of DES, the results that it would produce would vary a lot when the impulse rewards are very large. Correspondingly it would take a large number of replications for the obtained solutions to start converging and provide an idea of the trendline of solutions.

6.7.2 Conclusions

Based on this example we believe that the proxel-based method can be efficiently used for tuning parameters of more complex systems because the extrapolation yields shorter computation times than when simulating using very small time steps, and thereby achieves comparable accuracy. If speed is even more important, one can make the compromise of running proxel-based simulation with a large time step for obtaining a rough approximation for observing the trends in the behaviour of the system, using this information for making quick decisions. This, we observe as a very positive feature of the proxel-based method, especially in cases where models contain rare events that have

significant impulse rewards (e.g. costs), for which discrete-event simulation can become very expensive in terms of computation times for obtaining a reasonable estimate. The proxel-based method can, in such cases, be used for spotting the trend in the models because it provides fair treatment for all events in the model.

6.8 Proxels in Teaching Simulation

Discrete-event simulation, as a concept, is not a difficult one to teach. The reason for that is that the simulation is simply an imitation of system's behaviour, which is accomplished by mimicking its stochastic activities by using pseudo-random numbers. Therefore, we refer to DES as method that possesses a stochastic nature.

As soon as one states that stochastic systems can be analysed by using methods which advance deterministically, the whole simulation issue becomes very confusing. The partial differential equations only add to the confusion, and suddenly turn away students and their interest. Proxel-based method offers an alternative way of explaining simulation of stochastic models by using deterministic approaches.

The proxel-based method follows the natural and expected behaviour of a model and on-the-fly, consequently calculates the probabilities for the model taking each of the development paths. Therefore, we believe that the method can be especially useful in teaching simulation and analysis of discrete stochastic models and studying their behaviour. The usefulness is more applicable for understanding the numerical approaches, as the proxel-based method illustrates the flow of probability in a much more natural and intuitive way than it is approached by partial differential equations or any of the other methods which are based on mathematical models.

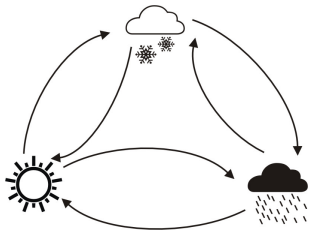


Figure 6.28: Simple model used for teaching proxel-based simulation

In our lectures we use the proxel-based approach to present the concept of behaviour and probability flow in stochastic models. One illustrative and intuitive example which we use, is also known as a classic in the area of simulation, that is the simplified weather prediction model, as shown in Figure 6.28. The model consists of three discrete states: *sunny*, *rainy* and *snowy*, and 6 possible state changes among all three discrete states. For a more intuitive explanation we use a time step Δt of one day. Using an appropriate illustration of the proxel-based approach, the responses from the students with respect to the comprehension of the paradigm of a "stochastic model" and the proxel-based method are very positive. It is a very pleasant experience to teach the analysis of

stochastic models without showing a single differential equation. The proxel-tree that we develop for this weather model is the one shown in Figure 6.29. The little balloons in the figure represent the age intensities of the relevant state changes in each discrete state, which are supposed to be memorised, and p_1 and p_2 are the corresponding probabilities.

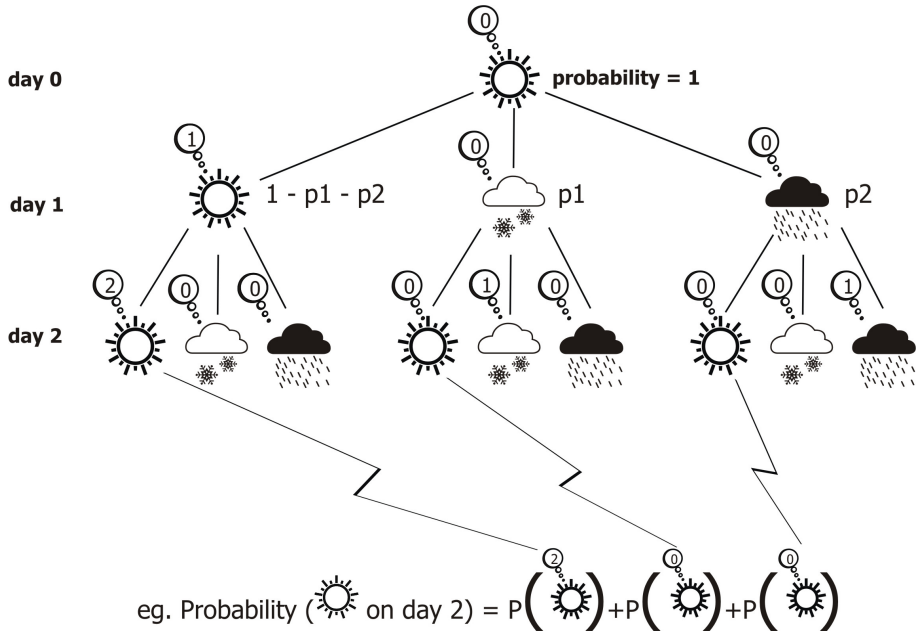


Figure 6.29: Teaching proxel-based simulation

We teach the proxel-based method regularly for the fourth year within our advanced course "Advanced Discrete Modelling". The responses from the students can be summarised as positive, especially based on the exams, where most of the students had no problems answering questions which were connected to the proxel-based method. The method allows a high level of interaction in the classes too, which we believe is because of the fact that the students are able to understand and follow the way the proxel-based method advances. We plan to continue this practice in the future too, and believe that the method will become a recognised tool for teaching deterministic simulation methods.

6.9 Summary

In this chapter we present mostly confirmed, but also some speculated applications of proxel-based method. The method is, however, not restricted to the here presented applications, as we believe that theoretically the method has a wide applicability. We believe that this can serve as good starting point, and that many alternative applications of the method will arise, mostly owing to the algorithmic and flexible definition of the proxel-based method. Some of these applications can be treated as directions

for *future work*, because even though the basic formalisation for each of them is laid, there is still a lot of work to be done with respect to experimenting and testing them on real-life situations and models.

7 Additional Experiments

In this chapter we present and discuss results of experiments performed regarding the accuracy and computational complexity of the proxel-based method, observed under different circumstances and for different types of models. The experiments presented in this chapter are additional in that, that they require a combined knowledge from different previous chapters, and it was hard to fit them within any of them. As a measure for the computational complexity of the experiments we use the number of stored proxels and the computation time. Sometimes, however, the size of the proxel can also make a difference and should be considered if the number of age variables significantly differs.

All experiments are performed on a Pentium III/1200MHz computer running Windows XP, with 512MB of main memory. For most of them the tool presented in the Appendix B was used, which was designed to automate the proxel-based simulation and make the experimenting process easier.

7.1 Experiments Concerning the Accuracy

As already explained in Section 3.2, the accuracy of the proxel-based method is a function mainly of the size of the time step - Δt, but also of other parameters. Two of them are the threshold for the minimum probability allowed ϵ and the choice of numerical approach for the integration of the IRF. The experiments in this section demonstrate how the accuracy depends on those parameters and show what they mean in practice.

Accuracy and Extrapolation

For the first set of experiments we use the model shown in Figure 7.1, which has different distribution functions for its state changes. We use different sizes of the time step for observing the development of the error and the trend of the solution values.

The goal of this set of experiments is to provide an idea of how the accuracy of the proxel-based method changes when changing the size of the time step and the other control parameters of the proxel-based method. Further we show the impact that the extrapolation has on improving the accuracy.

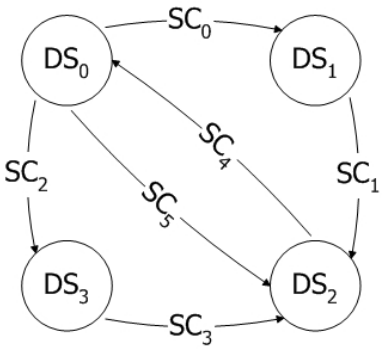


Figure 7.1: Example model

The distribution functions that describe the six state changes in the model, shown in Figure 7.1, are the following:

- $SC_0 \sim Uniform(1.5, 2.0)$,
- $SC_1 \sim Uniform(1.5, 2.5)$,
- $SC_2 \sim Exponential(2.0)$,
- $SC_3 \sim Normal(2.0, 1.0)$,
- $SC_4 \sim Exponential(2.0)$, and
- $SC_5 \sim Uniform(2.0, 3.0)$.

We treat all distribution functions as non-memoryless i.e. we employ supplementary variables even for the Exponentially distributed state changes. The purpose of this is to provide results which are meaningful for the general case.

In Figure 7.2 the transient solution values for the four different discrete states for time $t = 10$ are shown. The solutions are calculated using four different sizes of the time step, 0.025, 0.05, 0.1, and 0.2. It is evident that the method is at least a first-order one and that as $\Delta t \to 0$, all four solution values converge towards the "accurate" solution. The truncation error, which is produced by cutting off proxels with a probability values that are less than the threshold ϵ is 8.68915e-011. The value is computed as the sum of all probability values that are cut off. The value of ϵ in this case is 10^{-15} and $\Delta t = 0.025$.

The lifetimes of the discrete states, computed using the Algorithm 3.2 for $\Delta t = 0.025$, as a preprocessing step, are as follows:

- $lifetime(DS_0) = 80 \times \Delta t = 2.0$

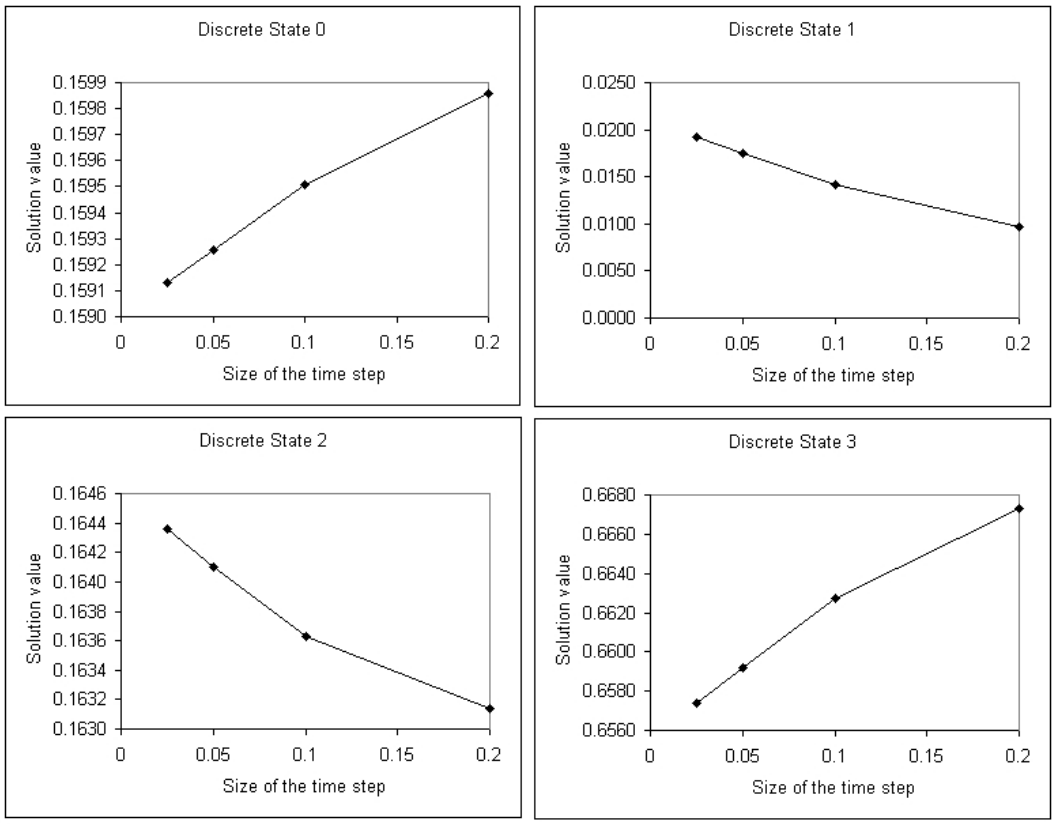


Figure 7.2: Solution values for the example model from Figure 7.2

- $lifetime(DS_1) = 100 \times \Delta t = 2.5$
- $lifetime(DS_2) = 400 \times \Delta t = 10.0$
- $lifetime(DS_3) = 353 \times \Delta t \approx 9.0$

where $t_{max} = 10 = 400 \times \Delta t$. Therefore, the sum of the lifetimes of the discrete states in the model is 23.5. The worst case complexity for this number of discrete states appears when the sum of lifetimes is 40 because all state changes have enabling (restart) memory policy.

We now compare the solutions obtained using the proxel-based method with the solutions obtained using discrete-event simulation in order to get a better idea about the accuracy of the proxel-based method. For that purpose we randomly choose the probability of the discrete state DS_3 as our point of interest. In order to get a narrow confidence interval for its transient solution at time $t = 10$, the discrete-event simulation needed computation time of 50 to 60 minutes. The size of the thereby obtained confidence interval is ca. 0.002 and the mean value is 0.655406. The level of confidence is 0.99 and the number of replications - 10^6. The graphical illustration of this comparison is shown in Figure 7.3, where one part of the solutions is zoomed to enable a closer observation of the results obtained using DES in form of confidence intervals (marked

by a dashed line) and the ones obtained using proxel-based simulation (marked by a full line).

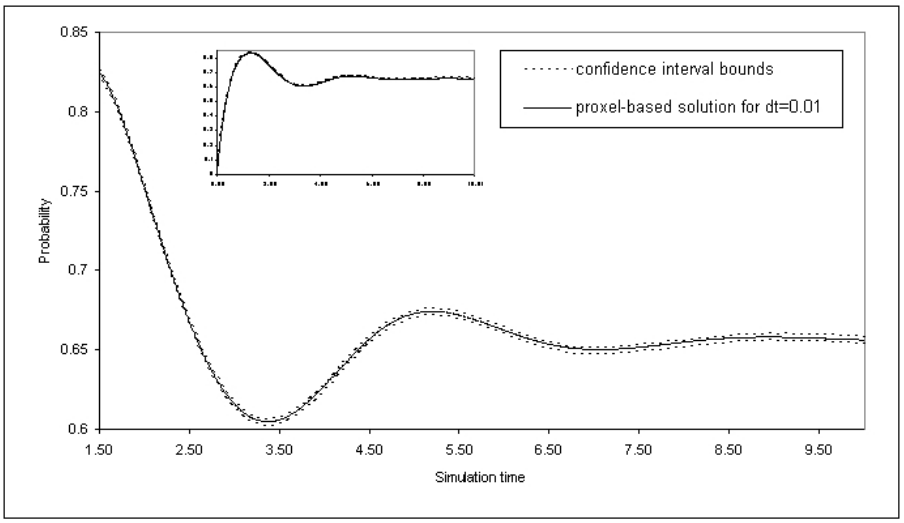


Figure 7.3: Comparison of solution values obtained using DES and proxel-based simulation

In Table 7.1 the Richardson extrapolation of the proxel-based solutions is provided, which takes into account solution values obtained using four different time steps, and extrapolates them to obtain a more accurate solution (i.e. 0.65549395238095). The total computation time, which is the sum of the separate ones, is 3.56 seconds.

In a comparable computation time as of the proxel-based method of ca. 3 seconds, using discrete-event simulation and performing different numbers of independent replications, we obtained the results shown in Figure 7.4. There one can see that the solutions obtained using DES do not converge (especially the mean values of the confidence intervals) with the increase of the number of independent replications. The described behaviour is because of the stochastic nature of the discrete-event simulation as an approach. The solutions obtained using DES start converging only when the number of replication increases significantly, a number which is dependent on the model description. However, even in that case the results do not converge monotonically.

Table 7.1: Richardson extrapolation for the solutions for DS_3 using time steps of 0.2, 0.1, 0.05, and 0.025

$\Delta t \backslash O()$	$O(\Delta t)$	$O(\Delta t^2)$	$O(\Delta t^3)$	$O(\Delta t^4)$	com. time
0.2	0.667327				0.06 s
0.1	0.66271	0.658093			0.2 s
0.05	0.659217	0.655724	0.6549343(3)		0.8 s
0.025	0.657358	0.655499	0.655424	0.65549395238095	2.5 s

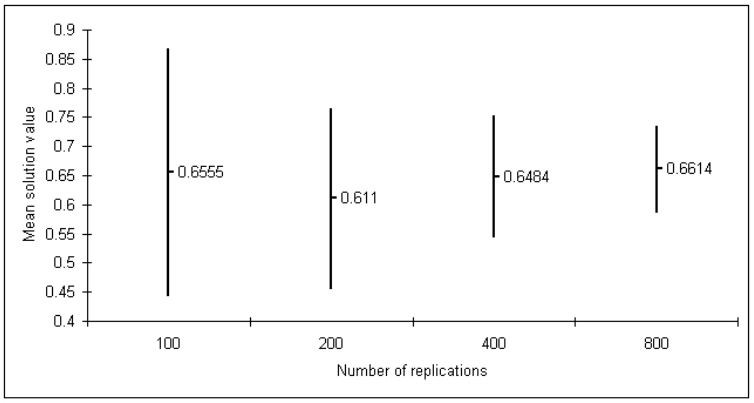


Figure 7.4: Solution values obtained using DES for DS_3 at time $t = 10.0$ of the example model from Figure 7.1

Accuracy and Minimum Probability Threshold

The second set of experiments concerning the accuracy is performed to observe the effects of the minimum probability threshold ϵ. It produces a truncation error as a result of cutting off nodes from the proxel tree, which have very small probability values (the value of the threshold is usually between 10^{-12} and 10^{-15}). For the experiments we use again the model from Figure 7.1, this time however with a rare event which contributes to generating small probabilities, with which we want to emphasise the effect of their truncation. For that purpose we set the distribution function of SC_0 to $Exponential(10^{-12})$, thereby reducing significantly the probability of the state DS_1.

Table 7.2: Development of the truncation error when using different thresholds for minimum probability

minprob	10^{-12}	10^{-13}	10^{-14}	10^{-15}	10^{-20}	10^{-25}
t. error	8.12E-11	8.76E-12	2.40E-12	1.37E-12	1.53E-19	0
solution	0	0	1.63E-15	7.03E-14	3.27E-13	3.27E-13

In Table 7.2 the truncation errors and solution values for the transient solution of DS_1 at $t = 10$ for different values of ϵ are shown. The solution values for the threshold equal up to 10^{-12} and 10^{-13} are both zero, and it stabilises once the threshold becomes 10^{-20}, in which case the error is very small and the solution does not differ much with the one for threshold equal to 10^{-25} where the error is zero. Therefore, we can state that the solution value for the probability of the discrete state DS_1 at time $t = 10$ is 3.27E-13, which when high accuracy is needed, and rare events are present, can not be neglected. This means that the value of the threshold for the minimum probability needs to be adapted to the model, and its distribution functions and parameters.

Accuracy and Integration Approaches

Further we observe the effect that the numerical approach for integration of the IRF has on the accuracy of the solutions and computation times. For that purpose we use again the model from Figure 7.1 and choose one discrete state (in this case the one marked as DS_3) and we observe its transient solution values for time $t = 5.0$, using different sizes of the time step Δt and different integration approaches for the IRF (mid-point, trapezoid, Simpson's rule, and Gaussian quadrature). The results of those experiments are illustrated in Figure 7.5. It can be observed that the Gaussian quadrature integration provides a little faster convergence than the other methods, and they all converge towards the same value.

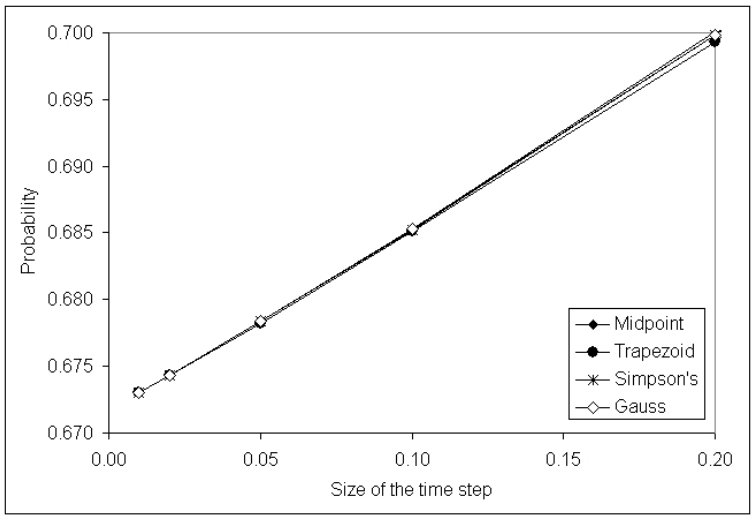


Figure 7.5: Solution values for DS_3 at time $t = 5.0$ of the example model from Figure 7.1

In Figure 7.6 the computation times are shown for the four integration methods for the case of $\Delta t = 0.01$ and $t_{max} = 10$. The differences are because of the different number of points at which the functions are evaluated, which makes the Simpson's rule the most expensive one. The figure illustrates that the computation times for the trapezoid approach and Gauss quadrature are very similar, which is because both of the integration approaches use two-point function evaluations. Although at first view, all computation times seem comparable, when performing more exhaustive simulation it can make big difference.

7.2 The Effect of Lifetimes of Discrete States on the Computational Complexity

The increment in the size of the discrete state space of one model can significantly increase the computational and memory complexity of the proxel-based simulation of

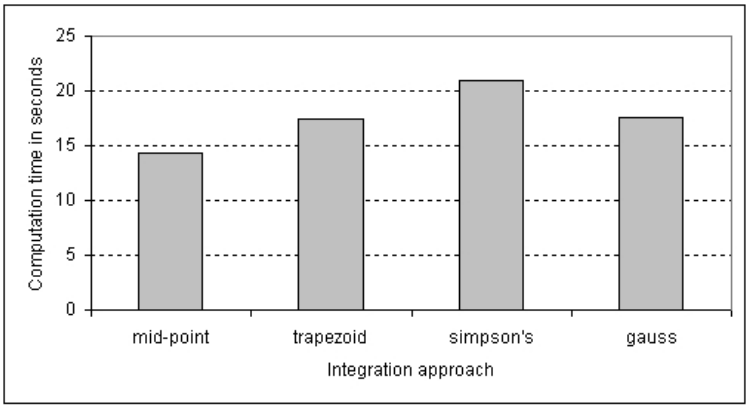


Figure 7.6: Computation times using different integration approaches for the IRF for the example model from Figure 7.1

one model. That, however, is not a function of only the size of the discrete state space, but also of the lifetimes of the discrete states in the model, which is determined by the number of concurrently active state changes and their distribution functions. The notion of lifetimes and their computation is elaborated in Section 3.3.2, and it determines the complexity of the proxel-based simulation of one model. It is important to notice that the discrete state space or the reachability graph, in the terminology of the Petri net formalism, is not a measure at all about the computational complexity of its proxel-based simulation. Experiments that treat the issue of the effect of the lifetimes on the computational complexity are demonstrated in this section.

As it was pointed out in Section 4.1.6, the number of concurrently active state changes affects the computational complexity of the proxel-based simulation. That number, together with the characteristics and the parameters of the random variables that are used to describe the state changes, determine the lifetimes of the discrete states. The effect of having more concurrently active state changes can be observed from two aspects. The first one is the number of age variables, which increases when having more state changes to track, making the proxel structure more complex, whereas the second aspect is beneficial, which is the lifetimes of the discrete states, which usually reduce with the increase of the number of concurrently active state changes. This in turn reduces the memory requirements of the simulation.

Conclusion can be made that the number of concurrently active state changes is not a constraining factor of the proxel-based simulation, it can be seen more as a beneficial or neutral one, unless they are activated at different points in time.

Lifetimes of the discrete states in one model are the factor that determines the complexity of its proxel-based simulation. Therefore, the experiments presented in this section demonstrate the relation of the computational complexity of the proxel-based simulation of one model and the sum of the lifetimes of its discrete states. The model that we choose for the first set of experiments excludes age memory state changes,

which are present in the models from the second set of experiments and in that case the complexity is higher, as is illustrated.

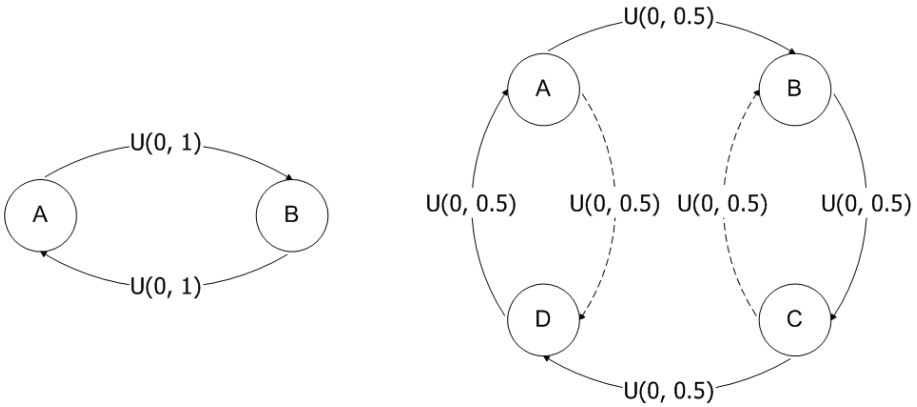


Figure 7.7: Two models with equal proxel complexities

Let us observe the models shown in Figure 7.7, excluding the state changes illustrated by dashed lines. The model with two discrete states seems on the first view as less complex than the four-state one. However, when analysed using the proxel-based method, both of the models have equal complexities because the sums of the lifetimes are equal. We chose this example, because the uniform distribution function has the nice property of having a finite support, making it simpler to determine the lifetimes of the discrete states.

When chosen $\Delta t = 0.05$, then the lifetimes of both discrete states in the two-state model are $20 \times \Delta t$, resulting into a maximal number of 40 proxels. In the four-state model, the lifetimes of the discrete states are $10 \times \Delta t$ resulting again into a maximal number of 20 proxels.

If we now add the state changes represented by the dashed lines to the four-state model, then the lifetimes of the discrete states A and C shorten insignificantly (because of the increased probability of leaving the discrete state). This results again into almost the same computational complexity i.e. number of proxels, which is 36. The computation times of the both simulations are the same too. Transient solution of the four-state model excluding the dashed state changes is shown in Figure 7.8. The model has a steady-state solution $Pr(A) = Pr(B) = Pr(C) = Pr(D) = 0.25$.

If we now exchange the state change from D to A in the original four-state model from Figure 7.7 for a Normally distributed one with parameters: $\mu = 1.0$ and $\sigma = 0.3$ then the lifetime of state D extends to $38 \times \Delta t$. Accordingly, the maximum number of proxels generated at each level of the proxel tree is 68, which is the sum of the lifetimes of all four states.

In Figure 7.9 the comparison of the number of proxels generated at each discrete time step for the 2-state and 4-state models is shown. It is evident that the 4-state model reaches the maximum number of proxels sooner than the 2-state one, which is because

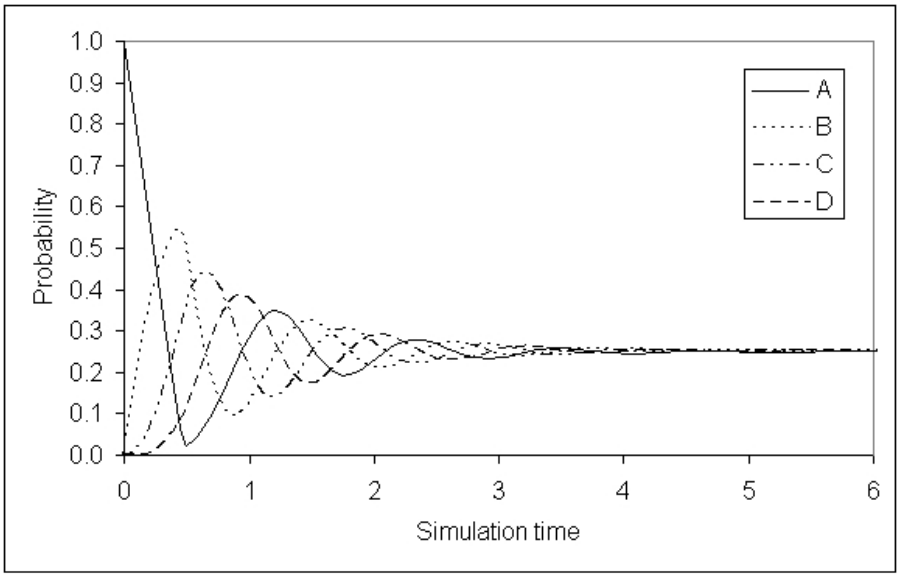


Figure 7.8: Solution values for the four-state model from Figure 7.7 excluding the dash-lined state changes

of the greater number of state changes which generate accordingly more proxels at each time step.

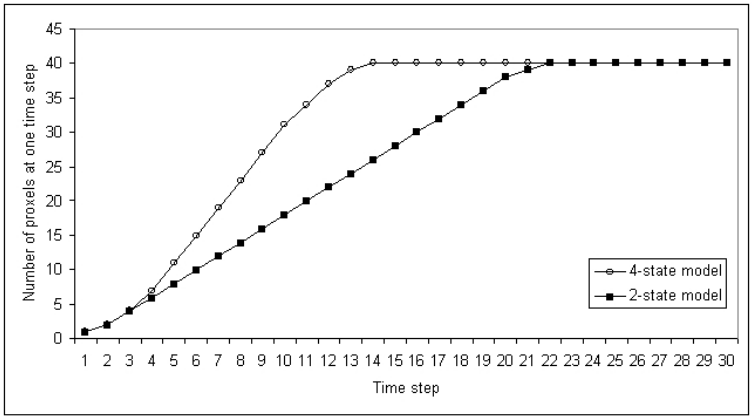


Figure 7.9: Comparison of the numbers of proxels generated at each time step for the two models from Figure 7.7, excluding the dash-lined state changes

Special Case - Age Memory State Changes

As mentioned in Section 3.3.2, the relation between the stored state-space (i.e. the memory complexity) of the proxel-based simulation of one model involves the number of concurrently active state changes which have different activation times, resulting into different age intensities at same points in time. A typical case that illustrates the

described situation is when a model contains state changes that are characterised by an age memory policy. For that purpose we carry out two sets of experiments based on one model in which the memory policy of one transition varies. The model is the same as the one presented in Section 5.5.2, in Figure 5.4, except that this time we do not observe the number of failures, thereby reducing it into the state-transition diagram shown in Figure 7.10. The state change, whose memory policy is varied is SC_0, which means that when the machine fails, the tracking of time until the next maintenance does not restart, but continues from where it stopped as soon as the machine goes back to being functional, i.e. discrete state *OK*.

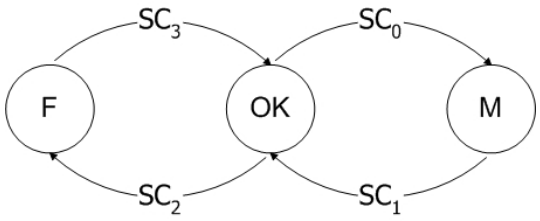


Figure 7.10: State-transition diagram of the fault-tolerant machine

The distributions that describe the stochastics of the state changes in the model are the following:

- $SC_0 \sim Uniform(15.0, 17.0)$,
- $SC_1 \sim Uniform(1.5, 2.5)$,
- $SC_2 \sim Exponential(0.05)$, and
- $SC_3 \sim Normal(2.0, 1.0)$.

$\Delta t = 0.2$ and it is simulated up to time $t = 50$. In the first set of experiments all of the state changes have enabling (i.e. restart) memory policy and the lifetimes of the discrete states are the following:

- $lifetime(OK) = 85 \times \Delta t = 17.0$
- $lifetime(F) = 34 \times \Delta t = 6.8$
- $lifetime(M) = 12 \times \Delta t = 2.4$

resulting into a total of $131 \times \Delta$, meaning that 131 is exactly the maximum number of proxels generated at each level.

In the second set of experiments we let the state change SC_0 have an age memory policy. In the case when there are state changes with age memory policy involved, the number of truly reachable states increases greatly, as the number of combinations of the values for the age intensities increases[1]. In that case predicting an upper bound of the number of proxels being generated at each step has to include this factor, which

[1]The latter is known as MNSDS, and is explained and defined in Section 3.3.2

implies that the lifetimes of all states which keep track of the age intensity of the age memory state change(s) has to be multiplied by the lifetime of the state in which the age memory state change is *active*. The sum gives the upper bound of the proxels generated at one level. The operation is to be carried out on the multiples of Δt, and in this particular case it is:

$$lifetime(OK) \times lifetime(OK) + lifetime(F) \times lifetime(OK) + lifetime(M) = 10127, \quad (7.1)$$

which shows that in this case the proxel-based simulation is significantly less efficient than in the case when models do not have concurrently active state changes, activated at different points in time. In Figure 7.11 the number of proxels generated at each level is shown when simulated up to time $t = 50$ for the described case. The computation time was 53.867 seconds and the maximum number of proxels at one level 6370, which is a lot less than the number obtained using the Equation (7.1). The formula, however, defines an upper bound which is based on all possible combinations. One of the future work areas would be to improve the estimate and generalise it.

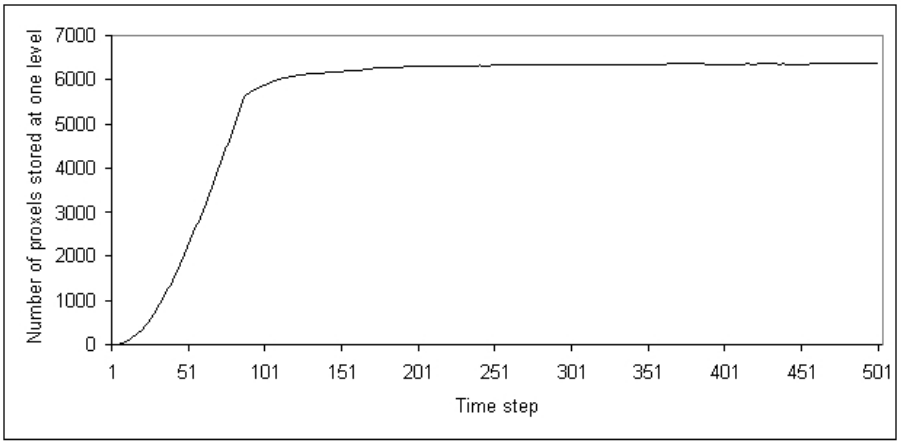


Figure 7.11: The number of proxels generated at one level for the model described in Figure 5.4 in the case when the transition SC_0 has an age memory policy

Summary

The experiments presented in this section show the direct dependence of the computational and memory complexity on the lifetimes of the discrete states in the model. This is an important issue regarding the proxel based-method and its improvements, particularly with respect to the employment of discrete phases. The "a priori" computation of the lifetimes supports the process of making decision whether to substitute one general distribution by a phase-type one in the proxel-based simulation. The combination of discrete phase-type distributions and proxels is discussed in Section 4.2.2.

7.3 Models for Which Proxels Are a Bad Idea

In Section 7.1 we show an example for which the proxel-based simulation is more efficient than discrete-event simulation. In Chapter 4 we discuss the special cases for which proxel-based simulation would be a good choice for reasons of efficiency and accuracy. Based on those conclusions, here we create a model for which the proxel-based simulation fails i.e. the computational complexity and the memory requirements are so high that discrete-event simulation would be the method of choice.

If we observe the favourable features of the models regarding proxel-based simulation, we can easily come to a list of features which would lead to very low-efficient proxel-based simulation, to name few of them:

- long lifetimes,
- presence of age memory state changes,
- state changes which requires small size of Δt,

which already provide an idea for constructing a model for which discrete-event simulation would be more suitable. These are also factors which would not influence discrete-event simulation in a significant way. DES is more affected from the relatively big variances of the distributions, as well as from the big differences in the rates, such as are to be found in rare-event models. Therefore, the nature of the factors that influence the efficiency of the simulation of the both approaches (DES and proxel-based) is different, thus meaning that the classification of models analysable by the one or the other method is not unambiguous.

In order to satisfy the criteria for inefficient proxel-based simulation we decided to use again the model from Figure 7.1, thereby changing the memory policies of the transitions, i.e. SC_0, SC_4 and SC_5 have now age memory policy. This increased the number of supplementary variables, which for this model is now four (we ignore the fact that one of them is exponentially distributed and treat as if it was not), and also increased significantly the number of truly reachable states, as the age memory transitions created a big number of combinations of the age intensities of the relevant state changes (MNSDS, Section 3.3.2). For illustration we use the Table 7.3, which illustrates the computational complexities of the proxel-based simulations for both cases. In the case where $\Delta t = 0.025$ we cancelled the simulation because it was becoming already very expensive and took long time, already significantly more than the discrete-event simulation.

This experiment illustrates an example model for which DES would be the better choice for simulation and provides an idea for the types of models for which proxel-based simulation would not be the analysis method of choice. The main reason for that is the state-space explosion, which is especially aggressive when state changes with age memory policy are part of the model, as is the case with the model analysed in this section.

Table 7.3: Computational complexity of the proxel-based simulation up to time $t = 10$ for two cases: A-including age memory state changes, and B-without age memory state changes (* The simulation was run only up to $t = 4.5$ because of the long running times)

$\Delta t \backslash O()$	A-proxels	B-proxels	A-com. time	B-com. time
0.2	55053	3306	0.83 s	0.06 s
0.1	877721	13572	15 s	0.2 s
0.05	12692092	54145	287 s	0.8 s
0.025	29842678*	215926	5390 s*	2.5 s

7.4 Summary

In this chapter we show the results of a series of experiments which were performed mostly for providing an idea about the accuracy and computational complexity of the proxel-based method, but also for comparing it with the most popular simulation approach, the discrete-event simulation. We chose discrete-event simulation as a method for comparison because of the same level of applicability, i.e. both methods can analyse any type of model as long as it is described in a form that the computer can handle. The largest part of the experiments are, however, provided throughout the thesis, along in the corresponding sections. Therefore, the experiments presented in this chapter are experiments that needed knowledge from more different chapters, and this was the most adequate place for them.

8 Conclusions and Future Work

8.1 Summary

To conclude the thesis, we begin by summarising our main contributions.

In Chapter 3 we established the formal foundations of the proxel-based method and defined the elements and terms involved in the method. In the same chapter we presented the basic proxel-based algorithm which operates on state-transition diagrams, as well as two supplementary algorithms, which provide a more efficient and straightforward proxel-based simulation. The first one shows an algorithm for calculating lifetimes of the discrete states in a model for predicting the computational complexity of its simulation and calculating the keys of the proxels (for accessing them in the data structure where they are stored). The second supplementary algorithm is a heuristics which generalises our approach for determining an acceptable size of the time step for the proxel-based simulation. In Chapter 3 we also examine thoroughly the accuracy of the proxel-based method, as well as the programming aspects and the computational complexity.

Further, in Chapter 4 we pay attention to the special issues regarding some characteristic classes of models that are specific or favoured by the proxel-based method i.e. for which it performs well compared to the other simulation and analysis approaches, or cases which need to be handled in a special way. There we also show the basic ideas behind the two variants of the proxel-based method that we have been working on and for which we believe could improve the performance of the method for some classes of models.

In order to bring the proxel-based method one step further, we designed a modelling framework which describes models in a way that they could be directly analysed using the proxel-based method. The complete description of it, along with some concrete examples is presented in Chapter 5.

The numerous applications of the proxel-based method which we have developed, as well as some speculative ideas for some promising future application areas are shown

in Chapter 6. This demonstrates the wide applicability of the proxel-based method and provides basic ideas for its further utilization.

In order to carry out most of the experiments throughout the work on the thesis we designed a proxel-based simulation tool, which is briefly described in Appendix B, which in the experiments' Chapter 7 is used to support many of the claims made in the thesis, especially with respect to the accuracy and the computational complexity.

Finally, this section provides a critical assessment of the thesis and conclusions based on the work on the thesis, as provided in the sections that follow. Furthermore, we also discuss the implications for future research.

8.2 Critical Assessment of the Thesis

The main goal of the thesis is to bring the proxel-based method from an idea to the level of an established method which can be offered to and used by simulation practitioners. This main goal resulted into the following three subgoals (or tasks) of the thesis:

- formalising the proxel-based method,
- analysing and studying its behaviour, and
- examining possibilities for its application.

Altogether, the three subgoals support one more subtle goal of the thesis, which is to provide a textbook that will aid the further development of the proxel-based method. In the following we discuss what each of the three tasks means and how their requirements are met.

Formalising the Proxel-Based Method

Formalising the proxel-based method meant defining all elements involved in the description of the method, thereby creating a basis for further research, development and improvement of the method. The first and basic part of the formalisation process is presented in Section 3.1.2, which defines the elements when models are described by means of state-transition diagrams, and further uses them for designing the basic proxel algorithm in Section 3.1.3. The provided definitions aid in avoiding ambiguities in the further development of the proxel-based method. The term "state" is a typical example for a confusion, unless defined. What is referred to as a "state" in Markov chains, in our case is referred to as a "discrete state".

The second part of the formalisation process which extended the definition process is presented in Section 5.2, and in general in Chapter 5. This is the chapter where we present the design of our proxel-adapted modelling framework, which complements the method by providing a way for describing the models such that it directly prepares them for their proxel-based simulation. The modelling framework allows modelling of

all classes of models that the proxel-based method can simulate, thereby optimising them for a more efficient simulation and providing a clear idea of what types of models the proxel-based method can analyse. Among others, this includes unbounded models, as well as models that have state changes distributed according to state-dependent distribution functions. The framework can either be used for direct modelling, or as an intermediate step for preparing the models for the proxel-based simulation. The main advantage of the proxel-adapted description framework is that it provides a clear idea of what types of models the proxel-based method can analyse and what is the most efficient form for them to be input to the simulator.

The paradigm of lifetimes of discrete states was introduced, which provides a way for predicting the computational complexity of the proxel-based simulation of one model. In the case when the model contains no age memory state changes, the sum of the lifetimes provides directly the maximum number of proxels generated at one level. The ability to predict the complexity can in turn be used for developing more effective storage strategies, as well as for making decisions whether to include discrete phase-type approximations.

The formalisation provided in the thesis is concluded regarding the current level of development of the proxel-based method. However, with every extension of the method, the formalisation will need to be updated and extended too. The experience with the inclusion of phases treated in (Isensee et al. 2005) shows that the formalisation is easily upgradable and provides a basis for communicating ideas regarding the proxel-based method.

Analysing and Studying the Method's Behaviour

Analysing and studying the behaviour of the proxel based method included studying the accuracy of the method, as well exploring classes of problems for which it performs well. The method was demonstrated to be at of least first-order accuracy, depending on the distribution functions involved in the model that is being analysed. The basic assumption that "the probability of more than one state changes happening within a time step is negligible" creates a first-order error, as well as the numerical integration methods used for approximating the probabilities for the state changes, in the worst case. The fact that the solutions converge is further utilised for performing extrapolation in order to achieve solutions of higher accuracy. The extrapolation yields shorter computation times when higher accuracy is needed, as the computation time and memory complexity increases significantly when decreasing the size of the time step, which is the other option for improving the accuracy.

In Chapter 4 we discuss the specific issues that affect the proxel-based method. These issues are very important to be acknowledged, as they are considered to be very significant aspects of the behaviour of the proxel-based method, both as strengths and drawbacks. Chapter 4 together with the experiments throughout the thesis demonstrate the level of detail of study and analysis of the proxel-based method. This aids

the main goal of the proxel-based method being a studied and examined tool with known advantages and weaknesses.

We believe that we captured the most specific and common aspects of the behaviour of the proxel-based method. As with the formalisation process, studying the behaviour of a relatively new method is a dynamic issue which will advance and develop with every new situation that may arise in future.

Examining Possibilities for Application of the Proxel-Based Method

Our third subgoal was to examine possibilities for applications of the newly introduced proxel-based method. That was not a difficult task, having in mind the advantages of the proxel-based method and its flexibility, especially regarding the classes of models it is able to analyse. The method was easily extended and adjusted to be applied to each of the following areas:

- Performability and Reward Modelling, the definition of the additional variables which keep track of rewards fit well into the existing framework,
- Warranty Analysis, which is our most practical application that solved a real-life problem by cutting down computation times of order of 20-30 hours to computation times of order of a couple of minutes,
- Fault Tree Analysis, the proxel-based method allows high flexibility in the definition of the basic events, letting them be defined as small stochastic models, providing transient solutions for the top event,
- Stochastic Petri Nets, by the ability to build the reachability graph on the fly, the proxel-based method can analyse SPNs without imposing limitations on the models,
- Hybrid Models, by discretising the continuous places, they are nothing but a supplementary variable in the definition of the proxel,
- Rare Event Simulation, the fact that the proxel-based method gives equal importance to all events in the model makes it very appropriate for analysing rare-event models, and
- Tuning Systems, which is a rather speculative and not fully researched area, for the reason that all events have equal importance and the applicability to reward modelling, the proxel-based method can be used for obtaining quick estimates of the trends in one system, and use that information for making decision about modifying some of its parameters.

We believe that the list can easily grow with the future research on the method and its improvement. At the moment the real obstacle to the proxel-based method is the problem of state-space explosion, and thus the memory complexity. Therefore, it is very hard to imagine analysing large-scale models using the proxel-based method,

which means that all of its current applications are limited to small models, for which it can offer high accuracy transient solutions.

The ongoing research on the combination of the proxel-based method with discrete phases, as described in Section 4.2.2, seems to offer an improvement with respect to the state-space explosion (Isensee et al. 2005). However, it is still not clear by what factor which is a subject of future research.

8.3 Discussion of the Proxel-Based Method

As already stated, the proxel-based method represents a new way of analysing discrete stochastic models. The approach is completely deterministic and because of its property of developing the solution algorithm directly from the model description, very flexible. Its flexibility means a wide applicability of the method, and an easily extendible framework for analysing many classes of models. To name some of them: performability models, fault trees, as well as the others presented in Chapter 6.

The proxel-based method, however, suffers from the well-known problem of state-space explosion, which is its real limitation currently. That is the reason why the method in its current form can only be applied to relatively small models. The lifetime factor can be used for a more precise definition of what a small model is. Thereby, the models are not limited by anything else, and their stiffness is also not seen as a problem.

There are many plans for improvement and lessening the problem of state-space explosion, one of them being the inclusion of phase-type distributions, but they are all ultimately just small steps. Ideally, the problem could only be solved by having an unlimited source of memory.

The proxel-based method can be also seen as a special case of a partial differential equation solver which forgets about the equations and is derived solely based on the model description. That is also the interesting thing, that when people solve problems they forget about the problems they are solving (in this case: analysing stochastic models), and instead spend long time blocked in the intermediate state (i.e. PDEs). With that respect, the proxel-based method is special because it goes back to the source i.e. the model and derives the solution approach directly from there.

Another way one can look at the proxel-based method, as already shown in Section 3.1.4, is as solving a special dimension-changing discrete-time Markov chain. The proxel-based method works actually as a DTMC solver, whereby the dimension of the probability matrix dynamically changes to include the newly generated states.

When compared to discrete-event simulation, the proxel-based method can be seen as its duality. They both operate directly on the model, whereby DES has variable time steps that are generated based on ”random” numbers and a probability of one for each path within each run. By contrast, the proxel-based method has constant time step and a different probability for all paths that the model can go through.

These different interpretations of the proxel-based method show once again that even though the simulation methods seem as very different, they are all in fact related and can transform into one another.

8.4 Implications for Further Research

With respect to the future research possibilities, we see them as a wide spectrum of applications of the proxel-based method and its improvements. The combination with phases (presented in Section 4.2.2) is a very promising perspective and we believe it might yield a very significant improvement in the computational complexity of the method. The future research regarding the phase-type approximations is generalising the approach presented in 4.2.2, deciding for which classes of problems and when it is suitable, as well as developing an estimate of the improvement factor.

As mentioned in Section 4.1.3, the proxel-based method can suffer significantly when advancing at a constant pace, i.e. when moving with an equal sized time step for all state changes in a model. The next future improvement of the method is to design an approach, which will modify this behaviour of the proxel-based method and use an adaptive size of the discretisation time step.

Another promising direction for future research is the stochastic variant of the proxel-based method (as described in Section 4.2.1), which would extend the applicability of the method to larger models. It is still not very clear what the weaknesses of the approach are, as well as its strengths, and that is yet to be analysed.

The notion of lifetime and its effect on the computational complexity can be used for investigating and optimising the data structures being used, as well as deciding whether to use phases for some of the state changes.

Also, some of the applications need real-life examples to be tested at and applied to. Therefore, one of the future tasks is to search possibilities for applications of the proxel-based method in the industry, which would reconfirm its usefulness.

8.5 Conclusions or Back into the Big Picture

The area of simulation of discrete stochastic models is enriched by another alternative approach, the proxel-based method. In the thesis we present the formalisation of the new method, show its strengths and weaknesses, and discuss its application possibilities. By doing that, we show that some “difficult” classes of stochastic models become deterministically analysable with the proxel-based method. That is our major contribution regarding the simulation community.

The proxel-based method is a widely applicable analysis method for discrete stochastic models. Its advantage consists in the feature of developing the solution algorithm

directly from the model description, which makes the method flexible and easily extendible. The second advantage is that it is a deterministic approach, as opposed to discrete-event simulation, meaning the proxel-based method does not have to rely on random numbers and does not reproduce the behaviour of models in order to implicitly analyse them. Therefore, the proxel-based method is designed to *analyse* discrete stochastic models, as opposed to *reproduce*.

The method itself has a big potential for further improvement as described in the "future work" Section 8.4. With that respect, the formalisation presented in the thesis contributes to its further development by providing a convention for describing the method which will support the further research and ease the communicating of ideas regarding the proxel-based method. The presented study and analysis of the method pay special attention to discovering its weaknesses and strengths, and provide implications for the classes of models for which it is suitable.

It is a fact that the power and the capacities of computers constantly grow with every new generation, which is a factor that influences the applicability of the proxel-based method in that that it allows more and more complex models to be analysable. The real strength of the method is, however, that besides on the size of the models, it has no limitations with respect to the types of discrete stochastic models that it can analyse.

This thesis can be seen as a contribution to the further development of the proxel-based method. Many ideas for improvement of the method in different ways, have been presented, and most of them have the potential to result in constructive realisations. They are, mostly, aimed at improving the efficiency and therefore increasing the applicability of the method to more complex models.

Finally, we believe that this work is one of the initial steps towards the establishment of a significant method, which one day will be implemented in every useful simulation tool and will have a chapter in every book on simulation.

A Frequently Used Probability Distributions

Here we describe some properties of the probability distribution used in the examples throughout the thesis. We focus particularly on the instantaneous rate functions.

A.1 Exponential Distribution

The exponential distribution with a parameter λ i.e. $Exponential(\lambda)$ is characterised by the following density $f()$ and cumulative distribution functions $F()$:

$$f(x) = \lambda e^{-\lambda x} \text{ and } F(x) = 1 - e^{-\lambda x}.$$

The exponential is the only memoryless function, which very well illustrated by its instantaneous rate function $\mu()$:

$$\mu(x) = \lambda,$$

which is always constant independent of the age intensity.

A.2 Uniform Distribution

The uniform distribution defined on an interval (a, b) i.e. $Uniform(a, b)$ is characterised by the following density $f()$ and cumulative distribution functions $F()$:

$$f(x) = \begin{cases} 0 & x < a \\ \frac{1}{b-a} & a < x < b \\ 0 & x > b \end{cases}, \text{ and } F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & x > b \end{cases}.$$

The instantaneous rate function $\mu()$ goes to infinity as the age gets closer to b:

$$\mu(x) = \frac{1}{b-x},$$

and is defined on (a, b).

A.3 Deterministic Distribution

The deterministic distribution defined in the value a i.e. $Deterministic(a)$ is characterised by the following density $f()$ and cumulative distribution functions $F()$:

$$f(x) = \begin{cases} 0 & x \neq a \\ \infty & x = a \end{cases} \text{ and } F(x) = \begin{cases} 0 & x < a \\ 1 & x \geq a \end{cases} .$$

The instantaneous rate function $\mu()$ goes to infinity when the age intensity is equal to a:

$$\mu(x) = \begin{cases} 0 & x \neq a \\ \infty & x = a \end{cases}$$

A.4 Normal Distribution

The Normal distribution defined with parameters μ and σ i.e. $Nor(\mu, \sigma)$ is characterised by the following probability density function $f()$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The expression for the cumulative distribution function of the normal distribution does not exist in a simple closed formula. It is computed numerically. This is also valid for its IRF.

A.5 Weibull Distribution

The Weibull distribution defined with parameters α and β i.e. $Weibull(\alpha, \beta)$ is characterised by the following density $f()$ and cumulative distribution functions $F()$:

$$f(x) = \beta\alpha^{-\beta}x^{\beta-1}e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \text{ and } F(x) = 1 - e^{-\left(\frac{x}{\alpha}\right)^{\beta}}.$$

The instantaneous rate function $\mu()$ is defined as follows:

$$\mu(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1}.$$

B Proxel-Based Simulation Tool

For testing and experimenting purposes we built a tool that implements the proxel-based method. In this chapter we introduce this tool, which we also used for most of the experiments in the thesis. First the input format to the proxel-based simulator is explained, after which its options are and the way it works are described. The whole process is demonstrated by a concrete example.

B.1 Description and Specifications

The proxel-based simulator has a graphical user interface (GUI), which makes it very easy for use and experimenting. The simulator was created in the early stages of the work on this thesis and therefore its requirement for the input is a reachability graph of a Petri net (or a state-transition diagram), with some extra information as explained further in detail.

In Figure B.1 the graphical user interface of the simulator is shown. The model input is through a file, which has to comply with a format that is described in the next section. Once the model is loaded, it is shown in the left edit-box. In order to analyse the models, first a preprocessing step is performed which calculates the lifetimes of each of the states, which is necessary for an efficient computation of the keys of the proxels. After that, the model is solved using the proxel-based method and the results are displayed in the right-hand edit-box. Additionally, the results can either be saved in a file or plotted directly by the tool.

B.2 Input Model Specification and Example

The specification of the input file that describes the model contains the following elements:

- maximum simulation time t_{max},

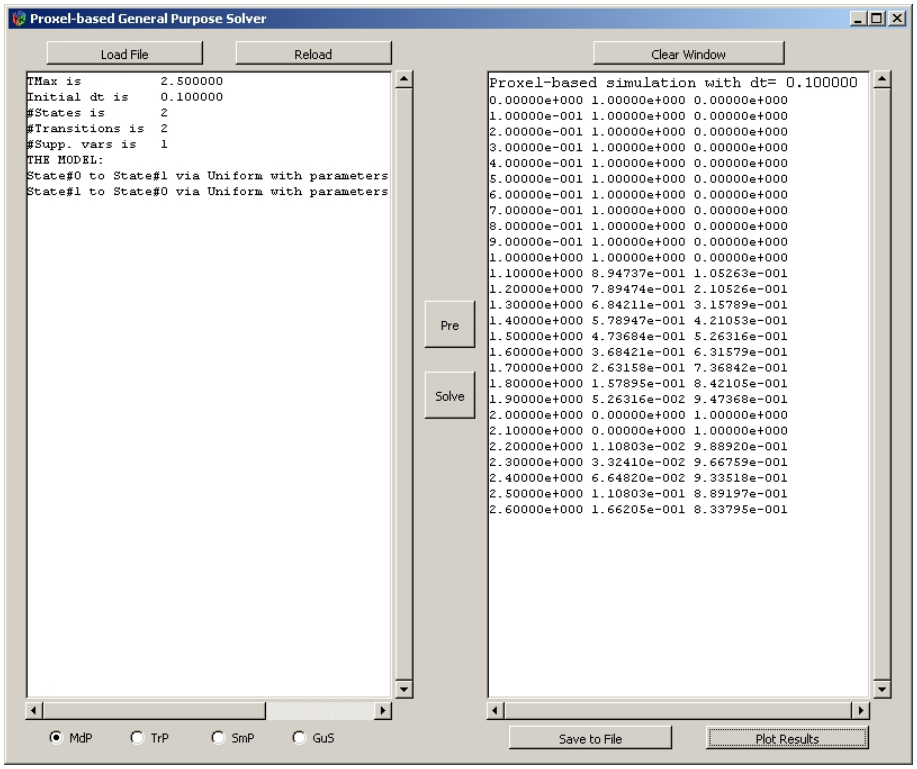


Figure B.1: The GUI of the proxel-based simulator

- size of the discretisation time step Δt,
- number of discrete states,
- number of state changes,
- maximum number of supplementary variables,
- all state changes in the following form: incoming state, outgoing state, distribution, parameters, memory policy, identity, and
- all discrete states in the following form: discrete state, age intensity mapping vector.

The identity parameter ensures that the simulator recognises if the same transition is further enabled in the next marking because in a reachability graph different arcs may represent same transitions from a Petri net.

In order to illustrate how they are represented in the input file, we use the example model represented in Figure B.2. The input file for that model is the following:

```
2.5
0.1
2
```

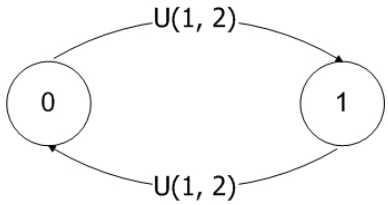


Figure B.2: Example model

```
2
1

0 1 U 1.0 2.0 E 0
1 0 U 1.0 2.0 E 1

0 0
1 1
```

meaning the model is simulated up to time $t_{max} = 2.5$, using a time step of $\Delta t = 0.1$. The model has two discrete states (0 and 1) and two transitions (0 and 1) which are both distributed according to Uniform(1,2) (described by: U 1.0 2.0), have enabling memory policy (E), and one supplementary variable at most in each of the two discrete states. If a state change has age memory policy, then it is specified by the letter "A". Other distribution functions that are implemented are:

- Exponential (E),
- Bathtub (B),
- Weibull (W),
- Normal (N), and
- Deterministic (D).

In the discrete state 0, the age of the 0-th transition needs to be tracked (described by: 0 0), whereas in the discrete state 1, the one of the 1-st transition (described by: 1 1).

Once this model is input to the proxel-based simulator, it appears in the left-hand text-box in the following form:

```
TMax is          2.500000
Initial dt is    0.100000
#States is       2
#Transitions is  2
#Supp. vars is   1
THE MODEL:
```

```
State#0 to State#1 via Uniform with parameters:  1.00  2.00
State#1 to State#0 via Uniform with parameters:  1.00  2.00
```

meaning that the model is loaded. The four radio buttons under the left-hand text box allow the user to choose one of the four integration methods for approximating the instantaneous rate function, as described in Section 3.2. Once the model has been analysed (solved), the solution can be graphically presented, which in this case results into the plot shown in Figure B.3.

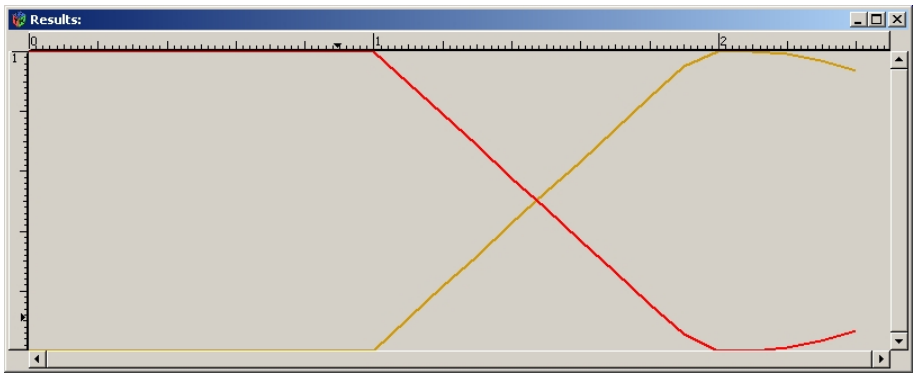


Figure B.3: Plot of the solution of the example model

The plot provides an instant insight into the model's behaviour and eases the model verification process.

B.3 Summary

The proxel simulator is a tool which was created for the purpose of studying the behaviour of the proxel-based method, and it surely satisfied its purpose. The tool, however, can be extended to capture the class of unbounded models, for which purpose the proxel-adapted modelling framework can be used for specifying the models. This is one of the implications for future work.

Another implementation of the proxel-based method exists as a part of a simulation tool, which is being developed and used by DaimlerChrysler and was a topic of a Diploma thesis completed at our Simulation and Modelling Group in cooperation with DaimlerChrysler, by Fabian Wickborn (Wickborn 2004).

Bibliography

Akers, S. (1978). Binary decision diagrams. *IEEE transactions on Computers*, C-27(6):509–516.

Alla, H. and David, R. (1998). Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8(1):159–188.

Apthorpe, R. (2001). A probabilistic approach to estimating computer system reliability. In *Proceedings of the 15th USENIX System Administration Conference — LISA 2001*, pages 31–46. USENIX.

Balaprakash, P. (2004). Preprocessing of petri nets and an improved storage strategy for proxel based simulation. Master's thesis, University of Magdeburg.

Banks, J., Carson II, J. S., and Nelson, B. L. (1998). *Discrete-Event System Simulation*. PUB-PRENT-I, second edition.

Baskar, L. D. (2005). Proxel-based simulation for fluid stochastic Petri nets. Master's thesis, University of Magdeburg.

Bobbio, A., Horváth, A., Scarpa, M., and Telek, M. (2000). Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm. Technical report, Budapest University of Technology and Economics.

Bobbio, A., Horvath, A., and Telek, M. (2002). Phfit: A general phase-type fitting tool. In *Proceedings of 12th Performance TOOLS*.

Bobbio, A., Puliafito, A., Scarpa, M., and Telek, M. (1997). WebSPN: Non Markovian stochastic petri net tool. In *18th Int. Conf. on Application and Theory of Petri Nets, (Toulouse (France))*.

Boyd, M. A. and Bavuso, S. J. (1993). Simulation modeling for long duration spacecraft control systems. In *Annual Reliability and Maintainability Symposium*, pages 106–113.

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691.

Choi, H., Kulkarni, V. G., and Trivedi, K. S. (1994). Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:335–357.

Ciardo, G. (1995). Discrete-time Markovian stochastic Petri nets. Technical Report TR-95-9.

Ciardo, G., Nicol, D. M., and Trivedi, K. S. (1999). Discrete-event simulation of fluid stochastic Petri nets. *IEEE Trans. Software Eng.*, 25(2):207–217.

Çinlar, E. (1975). *Introduction to stochastic processes.* Prentice-Hall Inc., Englewood Cliffs, N.J.

Cox, D. R. (1955a). The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. *Proceedings Cambridge Philosophical Society*, 51(3):433–441.

Cox, D. R. (1955b). A use of complex probabilities in the theory of stochastic processes. *Cambridge Phil, Soc.*, 51:313–319.

Cox, D. R. and Miller, H. D. (1965). *The Theory of Stochastic Processes.* Chapman and Hall.

Deavours, D. D., Clark, G., Courtney, T., Daly, D., Derisavi, S., Doyle, J. M., Sanders, W. H., and Webster, P. G. (2002). The möbius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969.

Deavours, D. D. and Sanders, W. H. (1998). On-the-fly solution techniques for stochastic Petri nets and extensions. *IEEE Trans. Softw. Eng.*, 24(10):889–902.

Dutuit, Y., Rauzy, A., and Signoret, J. P. (1997). Monte-Carlo simulation to propagate uncertainties in fault trees encoded by means of binary decision diagrams. In *1st International Conference on Mathematical Methods in Reliability, MMR'97*, pages 305–312.

Erlang, A. (1917). Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektroteknikeren*, 13.

Foster, C. C. (1965). Information retrieval: information storage and retrieval using avl trees. In *Proceedings of the 1965 20th national conference*, pages 192–205, New York, NY, USA. ACM Press.

Fricks, R. M., Puliafito, A., Telek, M., and Trivedi, K. S. (1998). Applications of non-Markovian stochastic Petri nets. *Performance Evaluation Review*, 26(2).

Garvels, M. (2000). *The Splitting Method in Rare Event Simulation.* PhD thesis, Faculty of mathematical Science, University of Twente, The Netherlands.

German, R. (1998). Markov regenerative stochastic Petri nets with general execution policies: Supplementary variable analysis and a prototype tool. In *TOOLS '98: Proceedings of the 10th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*, pages 255–266, London, UK. Springer-Verlag.

German, R. (2000a). Iterative analysis of Markov regenerative models. *Lecture Notes in Computer Science: Computer Performance Evaluation: Modeling Techniques and Tools*, 1786:156–170.

German, R. (2000b). *Performance Analysis of Communication Systems. Modelling with Non-Markovian Stochastic Petri Nets*. John Wiley & Sons, Ltd.

German, R. (2002). Non-Markovian analysis. *LNCS 2090: Lectures on Formal Methods and Performance Analysis*, pages 156–182.

German, R., Kelling, C., Zimmermann, A., and Hommel, G. (1995a). TimeNET: A toolkit for evaluating non-Markovian stochastic Petri nets. *Performance Evaluation*, 24(1-2):69–87.

German, R., Logothetis, D., and Trivedi, K. S. (1995b). Transient analysis of Markov regenerative stochastic Petri nets: a comparison of approaches. In *Proceedings of Sixth International Workshop on Petri Nets and Performance Models - PNPM '95 in Durham, North Carolina, USA*.

German, R. and Mitzlaff, J. (1995). Transient analysis of deterministic and stochastic Petri nets with TimeNET. In *Messung, Modellierung und Bewertung von*, pages 209–223.

Glynn, P. W. and Iglehart, D. L. (1989). Importance sampling for stochastic simulations. *Manage. Sci.*, 35(11):1367–1392.

Görg, C., Lamers, E., Fuß, O., and Heegaard, P. (2001). Rare event simulation. COST report 257.

Grassman, W. K. (1990). Finding transient solutions in Markovian event systems through randomization. In *1st International Conference on the Numerical Solution of Markov Chains, Raleigh*, pages 375–395, Raleigh, North Carolina.

Haas, P. J. (2002). *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer.

Haverkort, B., Marie, R., Rubino, G., and Trivedi, K. (2001). *Performability Modelling: Techniques and Tools*. John Wiley & Sons, Ltd.

Heidelberger, P. (1995). Fast simulation of rare events in queueing and reliability models. *ACM Trans. Model. Comput. Simul.*, 5(1):43–85.

Helstrom, C. W. (1984). *Probability and Stochastic Processes for Engineers*. Macmillan Publishing Company, New York, NY, USA.

Hoffman, J. D. (1992). *Numerical methods for engineers and scientists*. McGraw-Hill, Inc.

Horton, G. (2002). A new paradigm for the numerical simulation of stochastic Petri nets with general firing times. In *Proceedings of European Simulation Symposium, Dresden, October 2002*. SCS Verlag.

Horton, G., Kulkarni, V. G., Nicol, D. M., Trivedi, and S., K. (1996). Fluid stochastic Petri nets: Theory, applications, and solution techniques. Technical Report TR-96-5.

Horton, G. and Lazarova-Molnar, S. (2003). A reduced-stiffness method for the transient solution of discrete-time Markov chains. In *Proceedings of Numerical Solution of Markov Chains 2003, University of Illinois at Urbana-Champaign.*

Horváth, A. (2003). *Approximating non-Markovian Behaviour by Markovian Models.* PhD thesis, Budapest University of Technology and Economics.

Horvath, A., Puliafito, A., Scarpa, M., and Telek, M. (2000). Analysis and evaluation of non-Markovian stochastic Petri nets. In *Computer Performance Evaluation: Modelling Techniques and Tools, 11th International Conference, TOOLS*, pages 171–187.

Hsieh, M. (2002). Adaptive Monte Carlo methods for rare event simulation. In *WSC '02: Proceedings of the 34th conference on Winter simulation*, pages 108–115. Winter Simulation Conference.

Isensee, C. and Horton, G. (2005a). Approximation of discrete phase-type distributions. In *Proceedings of Annual Simulation Symposium 2005 in San Diego, USA.*

Isensee, C. and Horton, G. (2005b). Fast simulationwithout randomness: A simulation tool combining proxels and discrete phases. In *18. Symposium Simulationstechnik (ASIM 2005).*

Isensee, C., Lazarova-Molnar, S., and Horton, G. (2005). Combining proxels and discrete phases. In *Proceedings of The International Conference on Modeling, Simulation and Applied Optimization 2005 in Sharjah, U.A.E.*

Jensen, A. (1953). Markoff chains as an aid in the study of markoff processes. *Skand. Aktuarietiedskr.*, 36:87–91.

Kahn, H. and Marshall, A. W. (1953). Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research*, 1:263–278.

Kaiser, B. and Gramlich, C. (2004). State-event-fault-trees - a safety analysis model for software controlled systems. In *SAFECOMP*, pages 195–209.

Kartson, D., Balbo, G., Donatelli, S., Franceschinis, G., and Conte, G. (1994). *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, Inc., New York, NY, USA.

Kelling, C. (1996). A framework for rare event simulation of stochastic Petri nets using RESTART. In *WSC '96: Proceedings of the 28th conference on Winter simulation*, pages 317–324, New York, NY, USA. ACM Press.

Kolmogorov, A. (1950). *Foundations of the Theory of Probability.* Chelsea, New York.

Kulkarni, V. G. (1995). *Modeling and analysis of stochastic systems.* Chapman & Hall, Ltd., London, UK.

Lambert, J. D. (1973). *Computational methods in ordinary differential equations.* Wiley.

Law, A. M. and Kelton, D. M. (1999). *Simulation Modeling and Analysis*. McGraw-Hill Higher Education.

Lazarova-Molnar, S. and Horton, G. (2003a). An experimental study of the behaviour of the proxel-based simulation algorithm. In *Proceedings of the Simulation und Visualisierung 2003*.

Lazarova-Molnar, S. and Horton, G. (2003b). Proxel-based simulation for fault tree analysis. In Hohmann, R., editor, *17. Symposium Simulationstechnik (ASIM 2003)*, pages 99–104, Erlangen, Ghent. SCS European Publishing House.

Lazarova-Molnar, S. and Horton, G. (2003c). Proxel-based simulation of stochastic Petri nets containing immediate transitions. In *On-Site Proceedings of the Satellite Workshop of ICALP 2003 in Eindhoven, Netherlands. Forschungsbericht Universität Dortmund. Dortmund 2003.*

Lazarova-Molnar, S. and Horton, G. (2004). Proxel-based simulation of a warranty model. In *Proceedings of the European Simulation Multiconference 2004 in Magdeburg, Germany.*

Lazarova-Molnar, S. and Horton, G. (2005a). Description framework for proxel-based simulation of a general class of stochastic models. In *Summer Computer Simulation Conference 2005.*

Lazarova-Molnar, S. and Horton, G. (2005b). A framework for perfomability modelling using proxels. In *Proceedings of The International Conference on Modeling, Simulation and Applied Optimization 2005 in Sharjah, U.A.E.*

L'Ecuyer, P. (1990). Random numbers for simulation. *Commun. ACM*, 33(10):85–97.

Manian, R., Dugan, J. B., Coppit, D., and Sullivan, K. J. (1998a). Combining various solution techniques for dynamic fault tree analysis of computer systems. In *HASE*, pages 21–28.

Manian, R., Dugan, J. B., Coppit, D., and Sullivan, K. J. (1998b). Combining various solution techniques for dynamic fault tree analysis of computer systems. In *Third IEEE International High-Assurance Systems Engineering Symposium*, page 21.

Marsaglia, G. and Zaman, A. (1990). Toward a universal random number generator. *StatProbLett*, (8):35–39.

Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1998). Modelling with generalized stochastic Petri nets. *SIGMETRICS Perform. Eval. Rev.*, 26(2):2.

Marsan, M. A., Conte, G., and Balbo, G. (1984). A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122.

Muppala, J., Ciardo, G., and Trivedi, K. (1994). Stochastic reward nets for reliability prediction. *Communications in Reliability, Maintainability and Serviceability: An International Journal published by SAE International*, 1(2).

Neuts, M. F. (1981). *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Johns Hopkins University Press, Baltimore, MD, 1st edition.

Niederreiter, H. (1992). *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics.

O'Cinneide, C. A. (1999). Phase-type distributions: Open problems and a few properties. *Stochastic Models*, 15(4):731–757.

Osogami, T. and Harchol-Balter, M. (2003). A closed-form solution for mapping general distributions to minimal PH distributions. In *Computer Performance Evaluation / TOOLS*, pages 200–217.

Park, S. K. and Miller, K. W. (1988). Random number generators: good ones are hard to find. *Commun. ACM*, 31(10):1192–1201.

Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2. Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, 1966, Pages: Suppl. 1, English translation.

Puliafito, A., Telek, M., and Trivedi, K. S. (1997). The evolution of stochastic Petri nets.

Reisig, W. (1985). *Petri Nets, An Introduction*. EATCS, Monographs on Theoretical Computer Science. SPRINGER.

Richardson, L. (1927). The deferred approach to the limit, i-single lattice. *Trans. Roy. Soc. London*, 226:299–349.

Sahner, R. A. and Trivedi, K. S. (1996). SHARPE: Symbolic hierarchical automated reliability and performance evaluator: Introduction and guide for users. Duke Univ.

Sanders, W. H., Courtney, T., Deavours, D., Daly, D., Derisavi, S., and Lam, V. (2003). Multi-formalism and multi-solution-method modeling frameworks: the Möbius approach. In *Proceedings of the Symposium on Performance Evaluation - Stories and Perspectives*, pages 241–256, Vienna, Austria.

Schmidt, B. (2000). *The Art of Modelling and Simulation: Introduction to the Simulation System Simplex3*. SCS-Europe Publishing House, Ghent.

Siegle, M. (2002). Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties. Universitaet Erlangen-Nuernberg, 2002.

Stewart, W. (1995). *Introduction to the numerical solution of Markov chains*. Princeton University Press.

Stidham, S. J. (2002). Analysis, design, and control of queueing systems. *OR*, 50(1):197–216.

Sule, A. A. and Castro, I. T. (2002). Discrete-time analysis of a repairable machine. *Journal of Applied Probability*, 39(3):503–516.

Sullivan, K. J., Dugan, J. B., and Coppit, D. (1999). The galileo fault tree analysis tool. In *Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing*, pages 232–5, Madison, Wisconsin. IEEE.

Takagi, H. (1991). *Queueing Analysis - A Foundation of Performance Evaluation. Volume 1: Vacation and Priority Systems, Part 1.* North-Holland, Amsterdam.

Telek, M., Bobbio, A., Jereb, L., Puliafito, A., and Trivedi, K. S. (1995). Steady state analysis of Markov regenerative SPN with age memory policy. In *Messung, Modellierung und Bewertung von*, pages 165–179.

Telek, M. and Horvath, A. (2001). Transient analysis of age-MRSPNs by the method of supplementary variables. *Performance Evaluation*, 45(4):205–221.

Trivedi, K. S. (2002). *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., Chichester, UK, UK.

Trivedi, K. S., Bobbio, A., Ciardo, G., German, R., Puliafito, A., and Telek, M. (1995). Non-Markovian Petri nets. In *SIGMETRICS '95/PERFORMANCE '95: Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 263–264, New York, NY, USA. ACM Press.

Trivedi, K. S. and Kulkarni, V. G. (1993). FSPNs: Fluid stochastic Petri nets. In *Application and Theory of Petri Nets*, pages 24–31.

van Moorsel, A. P. A. and Wolter, K. (1998). Numerical solution of non-homogeneous markov processes through uniformization. In *Proceedings of the 12th European Simulation Multiconference on Simulation - Past, Present and Future*, pages 710–717. SCS Europe.

Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasl, D. F. (1981). *Fault Tree Handbook*. U. S. Nuclear Regulatory Commission, NUREG-0492, Washington DC.

Wickborn, F. (2004). Entwicklung eines erweiterten proxel-basierten Petri-netz-simulators. Master's thesis, Univeristy of Magdeburg.

Zimmermann, A., Freiheit, J., German, R., and Hommel, G. (2000). Petri net modelling and performability evaluation with TimeNET 3.0. *11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'2000)*.