Description Framework for Proxel-Based Simulation of a General Class of Stochastic Models

Sanja Lazarova-Molnar, Graham Horton
Institute for Simulation and Graphics
University of Magdeburg
Universitaetsplatz 2,
39106 Magdeburg,
Germany
sanja@sim-md.de, graham@sim-md.de

Keywords: proxel-based method, supplementary variables, framework, Petri nets.

ABSTRACT

The motivation for this paper is to raise the recently introduced *proxel-based* simulation method to a higher level by defining its own model description framework, which at the same time would allow us to enhance the description and analysis of discrete stochastic models beyond the potential of stochastic Petri nets.

The Proxel-based method is designed for transient analysis of discrete stochastic models, which are commonly described using stochastic Petri nets (SPNs). The approach is based on the method of supplementary variables, meaning it performs the analysis in a deterministic manner. It, however, works in a purely algorithmic style, without employing partial differential equations. Experiments and applications have so far shown the method to be promising, especially in analysing classes of models which are known to be difficult or problematic to be deterministically analysed using standard methods (such as Markov chains and partial differential equations). Infinite state space models, such as queuing systems with unbounded queues (with generally distributed arrivals and processing times) where servers can fail, and the number of failures and servers' age determine the parameters of the distribution function of the processing time, is one of those problematic cases.

As mentioned, up to now, a starting point for the proxel-based analysis was the Petri net model of the system to be simulated. The models, however, had to be adapted, using hard coding, to transform them into an appropriate input for the proxel-based simulator, mainly for efficiency reasons, but also for allowing properties that were not supported by SPNs. The transformation was implicitly the model description approach that we present and formalise in this paper.

We believe that the modelling framework that we describe here will be able to exploit all or most of the beneficial properties of the proxel-based method and describe the model in a way that is directly analysable by the proxel-based simulator. The framework is based upon Petri net features and modifies and extends them according to the properties and needs of the proxel-based method.

Our approach is supported and demonstrated by experiments and characteristic examples, as well as comparison to the formalism of SPNs.

INTRODUCTION AND OVERVIEW

Discrete stochastic models can be analysed in two ways, deterministically or stochastically. Deterministic methods have the advantage of not having to deal with "random" numbers and their randomness (quality), and provide results which are of higher accuracy. Discrete-event simulation, being a stochastic approach, delivers solutions of discrete stochastic models in form of confidence intervals, whose size in the ideal case is zero, i.e. contains only one value. Decreasing the size of the confidence interval means increasing the number of independent replications of the simulation run, and that by a value that is proportional to the square root of the increase multiplier of the number of replications.

Proxel-based simulation, introduced in [1], delivers the solutions of the models in form of a function, meaning the desired result of having a confidence interval of size zero is accomplished within one run of the proxel-based simulation, which at the same time observes all possible behaviours of the model. The completeness of the solutions contributes to the process of making statements and conclusions about behaviours of models. Our experience has shown that the method (at least theoretically) is able to analyse a wide class of models i.e. practically everything we have tried until now. This gives us the motivation to bring the proxel-based method to a higher level than its current development. One of the aspects regarding that is the way the models are being described.

The direct motivation for this paper is a practical application of the proxel-based method to a reliability modelling problem for DaimlerChrysler, described in [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 into 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.

Currently, the input to our proxel-based simulation tool is the reachability graph of a stochastic Petri net [3], which has the disadvantage that the models have to be bounded. In this paper we propose a new description approach, in which this problem does not exist any more and which exploits the other advantageous properties of the method. The direct input of the Petri net is also an option, but it would need optimising and adaptations before sending it to the proxel-based simulator.

The goal of the paper is thus to establish a modelling framework which is customized to the features and properties of the proxel-based method. In order to achieve this, we start with a general description of how the proxel-based method works, after what we define the elements needed to uniquely describe a model and compare them with a popular formalism: stochastic Petri nets. Further we show the adaptation of the proxel-based algorithm based on the described modelling framework. We use two examples to demonstrate the complete procedure of describing and analysing different models and present some computational results. Finally, we discuss our approach and present our ideas for future research with that respect.

PROXEL-BASED METHOD

The proxel-based method works by observing all of the possible behaviours of the model, each with a determined computable probability, based on the distribution functions which describe the events, as well as the time they have been pending (denoted as *age intensity*) [1][4]. The unit that stores all necessary information for turning a non-Markovian model into a Markovian one is referred to as *proxel* (probability element). Among the others, it contains the *discrete state*, the *age intensities* of the possible events in that discrete state, and the *probability*, resulting into the following structure:

Proxel = (*State*, *Time*, *Route*, *Probability*), where
State = (*Discrete State*, *Age Intensity Vector*).

The *Route* parameter contains the sequence of states via which the model has reached the actual state. Time advances in discrete steps and the point where the simulation starts is the initial discrete state. From there on, based on the possible state changes (associated with events) new proxels are generated for the subsequent time step. The values of the corresponding age intensities are updated with respect to the event that has caused the state change.

We illustrate the basic proxel-based method using a very simple model illustrated in Figure 1. The model consists of two discrete states (*A* and *B*), communicating via two state changes.

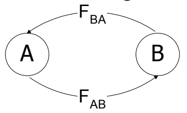


Figure 1. Example state diagram

Let *A* be the initial discrete state of this simple model and *dt* the size of the time step. There is only one age intensity that should be memorised in each proxel. In *A* it is the age intensity of the state change distributed according to F_{AB}, and in B it is the one distributed according to F_{BA}. The initial proxel for this model is the following:

((*A*, 0), 0, ∅, 1.0), where ∅ represents an empty route.

In the next time step the model can either stay in the discrete state *A* or change to the discrete state *B*, in which case the age intensity is reset and now it tracks the age of the state change associated with the distribution function F_{BA}. The proxels that are generated are the following:

((*A*, *dt*), *dt*, ((*A*, 0)), 1.0- *probability*) and ((*B*, 0), *dt*, ((*A*, 0)), *probability*)

The probability is calculated from the instantaneous rate function ($\mu(\tau)$) with the age intensity (τ) as a parameter, multiplied by the size of the time step:

$$probability = \mu(0)*dt.$$

In general instantaneous rate function can be computed as:

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

where *f* is the density function and *F* is the cumulative distribution function.

Now, let us observe both generated proxels and generate their successors. From the first one, ((*A*, *dt*), *dt*, ((*A*, 0)), 1.0- *probability*), the following two can be computed:

((*A*, 2*dt*), 2*dt*, ((*A*, 0), (*A*, *dt*)), *) and ((*B*, 0), 2*dt*, ((*A*, 0), (*A*, *dt*)), *).

The second proxel from the second generation, ((*B*, 0), *dt*, ((*A*, 0)), *probability*), determines the following successors:

((*A*, 0), 2*dt*, ((*A*, 0), (*B*, 0)), *) and ((*B*, *dt*), 2*dt*, ((*A*, 0), (*B*, 0)), *).

The probabilities are omitted for reasons of simplicity. The are, however, computed in the same manner as previously. The process of generation of proxels for the first three described time steps is illustrated in Figure 2.

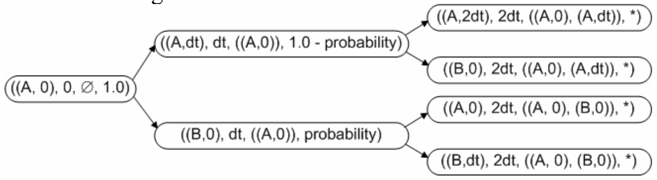

((A, 0), 0, ∅, 1.0)
((A,dt), dt, ((A,0)), 1.0 - probability)
((B,0), dt, ((A,0)), probability)
((A,2dt), 2dt, ((A,0), (A,dt)), *)
((B,0), 2dt, ((A,0), (A,dt)), *)
((A,0), 2dt, ((A, 0), (B,0)), *)
((B,dt), 2dt, ((A, 0), (B,0)), *)

Figure 2. The proxel tree of the first three time steps

It can be realised that we make one very important assumption in our approach. That is, we suppose that the model does at most one state change within time of *dt*. The assumption is ultimately correct when $dt \to 0$, which means that the smaller the time step, the higher accuracy can be achieved.

FORMAL BACKGROUND

The goal of any model is, when simulated, to mimic the behaviour of the given system as closely as possible. That is the reason why there are so many modelling formalisms, each of them being appropriate for some classes of models or analysis approaches, and possibly having problems with others.

The modelling framework that we propose represents a compact way of modelling, distinguishing 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 section.

Definitions and Terminology

In this section we define the terms necessary for describing our modelling framework. The following paradigms can be recognised in any discrete system, independent of the formalism used to model it:

Definition 1: *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 2: *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 3: *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 4: *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: *Age intensity* of an event is the pending time during which the event could have happened, but it has not.

Definition 6: *Active event* in a discrete state is an event which fulfils all preconditions for it to happen, as defined by the *rules*.

Definition 7: *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 8: *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.

$$S = (DS, \mathit{Age\ Intensities}(\mathit{Relevant\ Events}\ (DS)))$$

Definition 9: *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 10: *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 can be calculated.

Definition 11: *Proxel* (from **pro**bability **el**ement) 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.

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

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 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 (constructs) 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 described in the following table:

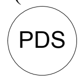
our approach	**Petri nets**
discrete state	*marking*
event	*transition*
age intensity	*activation time of a transition*
relevant event	*marking-enabled or age memory transition*
set of rules	*incidence matrix*

In general, when compared to the Petri net formalism, our proposed framework can be seen as 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.

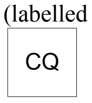
Graphical Description of the Models

To support the comprehension and enhance the intuitiveness of our modelling framework presented in this paper, 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:

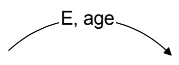
- Partial discrete state (labelled circle):



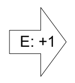
- Countable quantity (labelled square):



- Event (labelled arrow):



or



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 can also connect two corresponding partial discrete states to show the switch between the two, in a case they cause one. 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 4.

THE ADAPTED PROXEL ALGORITHM

In this section we present the algorithm for the proxel-based analysis of general models. The proxel simulation algorithm was introduced in [1] and developed further in [12][5]. Essentially, the proxel-based method builds a discrete-time Markov chain [6] on the fly, which approximates the behaviour of the model at discrete time steps, using the method of supplementary variables [7].

The simulation starts with the initial discrete state. From there on, based on the event rules, it is checked which of the events are active and correspondingly new proxels are generated for the subsequent time step. The values of the components of the age intensity vector are updated with respect to the event that has caused the state change, remembering the values of the ones that have aging memory. For every proxel, prior to its generation it is tested whether a proxel which represents the same state has been already generated for that time step. If the answer is positive then no new proxel is introduced and the computed probability is added to the probability of the existing one, as well as the set of routes extended to include the new one. This procedure is repeated until the end of the simulation time. The values of the proxels are recorded and manipulated, depending on the questions that the analysis is supposed to answer.

An interesting and delicate situation is when the discrete state of the model activates some conditional events. In this case, the proxel generated is not stored, but its successors generated by the happening of the conditional events. The proxel that initiated the new proxels, but was not stored is referred to as a *vanishing proxel*. This approach is similar to the on-the-fly elimination of vanishing states in the analysis of stochastic Petri nets [8][9].

The only thing necessary for computing one generation of proxels are the proxels computed in the previous time step (the predecessor proxels). The algorithm operates based on two interchangeable data structures (*proxel sets*): one which contains the proxels from the previous time step and another one which stores their successors, i.e. the proxels being computed in the current time step. Some of the desired features of the data structures are a fast search and access to the proxels. Currently we are using a binary tree. A successful attempt has been made in using a hash function-based array, but the function for computing the key did not involve any knowledge about the structure of the proxels. Therefore, we believe that the storage technique can still be improved, since it is still one of the problematic elements of the implementation, especially considering the state-space explosion problem [10].

The general structure of the algorithm is the shown as follows. Please note that for simplicity reasons the route and the simulation time components are excluded, there is also no need to explicitly store them as they are implicitly considered.

The following are the explanations of the variables (abbreviations) used in the algorithm:

- *PDS* is a partial discrete state,
- *Q* is the vector of the countable quantities,
- $\vec{\tau}$ is the age intensity vector (contains age intensities of relevant events),
- *prob* is the probability of the proxel,
- *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 the 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 calculated using the *instantaneous rate function* (IRF).

The lifetime of a state plays an important role for the efficiency of the simulation implementation. Currently we use that information to find the maximal value that an age intensity in one state can have, and use it for calculating a unique key for each proxel, which is needed for the search procedure.

```
Input: $\Delta t, t_{max}$
1  Initialize proxel set $PS(0)$ by inserting the initial proxel;
2  $switch = 0$;
3  for $i$=1 to $\lceil t_{max}/\Delta t \rceil$ do
4      foreach proxel $p = ((PDS, Q), \vec{\tau}, prob)$ in the proxel set $PS(switch)$ do
5          foreach active event $E_i$ in $PDS$ do
6              Calculate probability $prob_{calculated}$ for the event $E_i$;
7              Generate new state $S = (succ(PDS), succ(Q))$ according to rules $(E_i)$;
8              if $pds(S)$ enables set of conditional events $cE = \{cE_i\}$ then
9                  Generate set of new states $nS = \{nS_i\}$ according to $cE_i$
                   foreach $nS_i$ in $nS$ do
10                     Calculate probability $prob_{calculated}$ for the event $cE_i$;
11                     if $search(nS_i$ in $PS(1 - switch)) = proxel_{found}$ then
12                         $proxel_{found} =$
                           $(nS_i, (prob(proxel_{found}) + prob_{calculated}))$
13                     end
14                     Generate new proxel $p_{new} = (nS_i,\ prob_{calculated})$;
15                     Store $p_{new}$ in $PS(1 - switch)$;
16                 end
17             end
18             else if $search(S$ in $PS(1 - switch)) = proxel_{found}$ then
19                 $proxel_{found} = (S, (prob(proxel_{found}) + prob_{calculated}))$
20             end
21             else
22                 Generate new proxel $p_{new} = (S,\ prob_{calculated})$
23             end
24             $prob_{rest} = prob - prob_{calculated}$;
25         end
26         Generate new proxel $p_{new} = ((PDS, Q), succ_{stay}(\vec{\tau}), prob_{rest})$;
27         Store $p_{new}$ in $PS(1 - switch)$;
28         Remove the processed proxel $p$ from $PS(switch)$;
29     end
30     $switch = 1 - switch$
31 end
```

The proposed modelling formalism is an appropriate input format for the proxel simulator because the method is an algorithmic approach and it can operate based on any rules attached to the model. Therefore our approach was based on the necessity to find a way to describe the changes in the model in terms from one time step to another associated with every event, in order to analyse models effectively.

EXAMPLE MODELS

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. One is a model of a queuing system and the other one models machine failures. The examples are classical, but they contain features which make them difficult to be analysed using standard approaches, as described in detail along with the model descriptions.

Queuing System

The first example model is a queuing system that consists of two queues and one server, which can be interpreted as two types of customers which arrive according to two different distribution functions and are being served by one employee. The queues are unlimited. 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, as will be shown in this section.

The Petri net of the queuing model is shown in Figure 3.

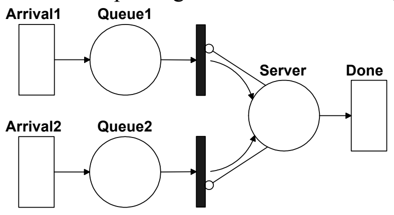


Figure 3. Petri net description of the queuing system model

In this model we recognise two candidates for countable quantities (number of people in the both queues) and two partial discrete states (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:

$E_1 \to Q_1+1$,

$E_2 \to Q_2+1$,

$E_3 \to Q_1-1$, (if F then B),

$E_4 \to Q_2-1$, (if F then B),

$E_5 \to$ (if B then F).

Three of the rules include *if*-conditions, where the *if* part describes the precondition(s) for that 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 4 a graphical representation of the model, using the proxel-adapted description framework, is shown. 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 initial values of the countable quantities are both zeros.

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.

Therefore, the age intensity vector consists of three components and has the following mappings: in $F \sim (E_1, E_2, /)$ and in $B \sim (E_1, E_2, E_5)$, the "/" symbol means that there is no mapping for that component, i.e. the place is free.

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

$$(PDS, Q_1, Q_2), \text{ where } PDS \in \{F, B\}, \text{ and } Q_1, Q_2 \in \mathbb{Z}_0^+ .$$

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, dt, /)), via E_1 (customer arrival in the first queue),
2) ((F, 0, 1), (dt, 0, /)), via E_2 (customer arrival in the second queue), and
3) ((F, 0, 0), (dt, dt, /)), no event has happened.

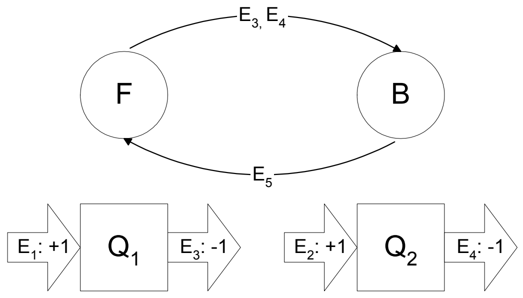


Figure 4. Graphical representation of the model using our approach

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 $t = dt$ are:

1) ((B, 0, 0), (0, dt, 0)), via E_1 and E_3,
2) ((B, 0, 0), (dt, 0, 0)), via E_2 and E_4, and
3) ((F, 0, 0), (dt, dt, /)), 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.

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 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 in function of the proxel-based simulation. Particularly in this example, the reduction of the computational complexity compared to the direct Petri net simulation is nothing, 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 (the partial discrete state).

Machine Model

The second example illustrates some additional features of the method including the possibility of doing reward modelling. The model represents a machine which has almost regular maintenance scheduled and fails according to a given distribution function, which is also 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.

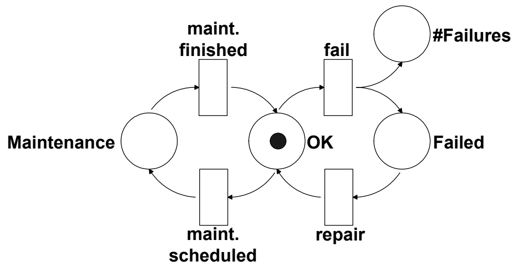


Figure 5. Petri net of the machine model

The state space of this model is unbounded too because the number of failures is not limited. This would mean limiting it in order to analyse the model deterministically using (for example) the partial differential equation approach or some of the Markov chains' solution methods.

By means of the description framework presented in this paper, the model has the following *partial discrete states*:

OK – machine is operating normally,

F – machine has failed,

M – machine is being maintained,

and one *countable quantity #F*, which counts the number of failures. The model contains four events, each distributed according to the assigned distribution function (the specific functions and parameters are given in the experiments' section):

E_1 – machine goes to maintenance mode ~ $F_1(\tau)$,

E_2 – maintenance completed ~ $F_2(\tau)$,

E_3 – machine fails ~ $F_3(\tau)$,

E_4 – repair completed ~ $F_4(\tau, \#F, t)$,

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*), *#F* +1,

$E_4 \rightarrow$ (if *F* then *OK*).

The graphical description of the model is shown in Figure 6. The age intensity vector is two-dimensional, with the following mappings of the events' age intensities to the discrete states: *OK* ~ (E_1, E_3), *M* ~ ($E_{2,}$ /), and *F* ~ (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 anymore, which is why it is not included into the mapping vector. The discrete state vector has the form (*PDS*, *#F*), where *PDS* $\in$ {*OK*, *M*, *F*} and *#F* $\in$ Z_0^+ and therefore the initial state is

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

resulting in the following subsequent states:

1) ((F, 1), (dt, 0)), via E_4 (machine has failed, the number of failures is also increased),
2) ((M, 0), (0, 0)), via E_1 (machine goes to maintenance), and
3) ((OK, 0),(dt, dt)), 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 a very intuitive way. This is so, even though the models have properties which are usually considered as undesired when it comes to analysing discrete stochastic models.

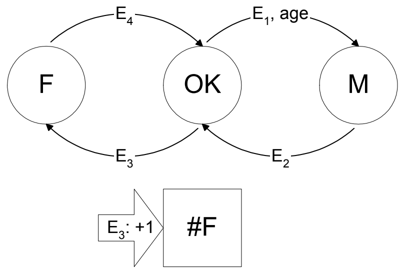


Figure 6. Graphical representation of the proxel-adapted machine model

EXPERIMENTS

In this section we will present results of some of the experiments performed using the two previously presented examples. 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 will present experiments concerning the queuing model and in the second one results concerning the machine failure model.

Experiments with the Queuing Model

The queuing model is interesting because its state space is unbounded – a property which is usually considered to be an obstacle when it comes to deterministic analysis of stochastic models. The reachability graph of the model would be unbounded, as represented in Figure 7. The rounded boxes represent tangible markings and the rectangles - vanishing markings. The two big ovals represent the partial discrete states.

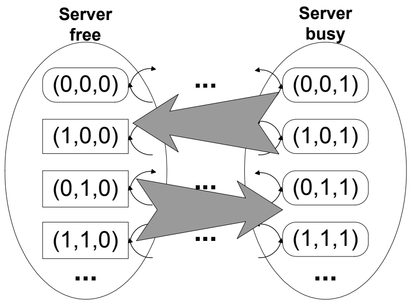


Figure 7: The reachability graph of the queue model

In the experiments we used the following distribution functions

E_1 ~ Uniform (0.8, 1.6),

E_2 ~ Uniform (1.0, 1.8), and

E_3 ~ Exponential (2.0),

for describing the corresponding timed events.

The questions that the analysis is supposed to answer are calculating the transient probabilities of 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 8 in the corresponding order. The computation time for this experiment was 500 seconds, using a time step dt = 0.2, simulating up to time t = 20. Observing the results shown in Figure 6 (which us to time t = 15 for having a better overview) 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.

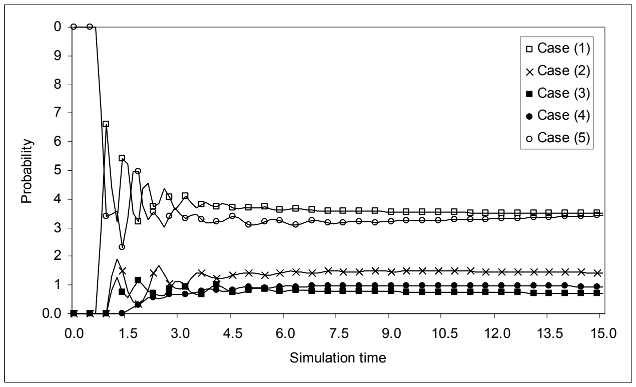


Figure 8. Transient solution of the queuing system up to t = 15

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 were used for the experiments with this model are the following:

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

The probabilities that we are interested in computing are for the following discrete states:
(1) There are no failures,
(2) There is one failure,
(3) There are two or more failures, and
(4) The system is being repaired.

The results of the analysis are shown in Figure 9. The second thing that could be of interest in this model is the number of failures and its transient development. This is shown in Figure 10. The computation time for this experiment was 0.06 seconds, using a time step dt = 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. Based on the solutions, it is easy to draw conclusions about the behaviour of the model, and sometimes to predict it for points in time for which it has not been yet simulated. Another goal of this model was to show that the proxel-based method can be seen as a promising tool in reward and performability modelling [11].

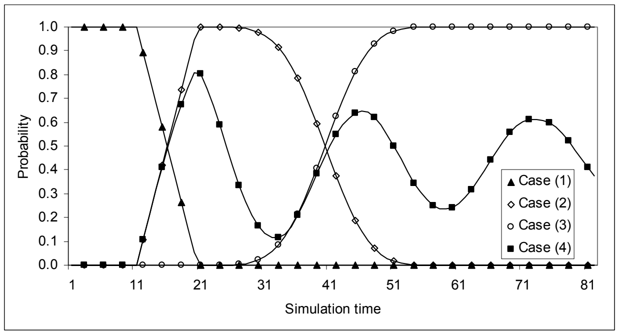


Figure 9. Transient solution of the machine model

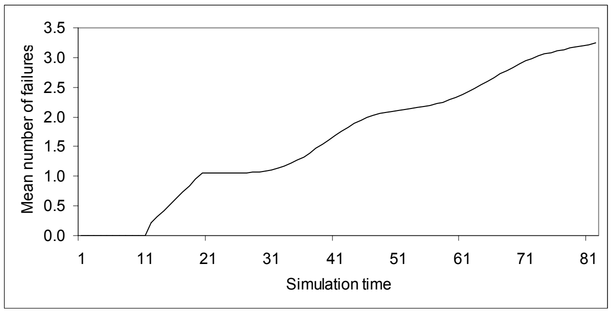


Figure 10. Number of failures as a function of the simulation time

Both series of experiments in this section analysed models which have properties that are usually found as unwanted and problematic. However, this was not an obstacle for the proxel-based simulation. Just for a comparison, the models could have been analysed using discrete-event simulation, but then the results would have been in a form of confidence intervals requiring a huge amount of independent replications to achieve or get closer to the accuracy that the proxel-based method provides. This is shown in [2], where the discrete-event simulation needed 20 to 30 hours, whereas the proxel-based simulation for achieving the corresponding accuracy, a couple of seconds to a couple of minutes.

DISCUSSION
The motivation for developing this modelling framework is a practical application of the proxel-based method to a reliability modelling problem for DaimlerChrysler [2]. This experience meant necessity of 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 shown in Figure 7) and recognizing 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 behaviours (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 [12].

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 and is further to be developed. 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 will contribute to making the method a generally applicable tool for analysing stochastic models.

SUMMARY AND OUTLOOK

We are interested in studying the proxel-based simulation of discrete systems and in developing tools for performing this type of simulation. Proxel simulation is a state space-based approach, and builds the Markov chain approximation of the model on-the-fly. It has proved essential to develop a new modelling paradigm which is based on familiar concepts and provides a more appropriate and less limiting input format to the proxel simulator. One important aspect of this paradigm is to distinguish between bounded and unbounded quantities in the model, which allows for more efficient storage techniques during the simulation. In addition, extensions such as time-dependent parameters of events' distribution functions are also possible. One future extension of the modelling framework will be to include uncountable quantities, which should expand the proxel-based approach to the analysis of hybrid models.

At the moment we are working on a general proxel analysis tool, which would be based on the described framework. The framework currently is tested on hard-coded examples, the results of which were illustrated in the experiments section.

An important factor for the performance of the proxel-based simulation is the lifetime of the states, from where we come to the conclusion that the more competing events there are, the shorter the lifetime of a state is. This in turn gives to a smaller state space, which leads to better performance of the computation [12]. Currently we have created a pre-processing tool which calculates the lifetimes of all the discrete states in the model a priori and thus prepare the model for better performance.

For many common probability distributions, a discrete-time phase representation can be considerably more efficient than a proxel (i.e. supplementary variable) representation. In addition, methods are now available for computing the phase approximation quickly [13]. One topic for future development is thus the integration of phase-type models into the framework described here.

Finally, in conclusion, the work presented here represents another milestone towards a full understanding of the discrete-time Markov chain approach to the simulation of discrete stochastic models and to the building of a tool which is able to perform such a simulation efficiently and reliably.

Reference List

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

[2] Sanja Lazarova-Molnar, Graham Horton: *Proxel-Based Simulation of a Warranty Model.* European Simulation Multiconference, Magdeburg, 2004.

[3] Peter J. Haas: *Stochastic Petri Nets.* Springer-Verlag, 2000.

[4] Sanja Lazarova-Molnar, Graham Horton. *Proxel-Based Simulation for Fault Tree Analysis*. 17. Symposium Simulationstechnik (ASIM 2003), SCS European Publishing House 2003.

[5] Sanja Lazarova-Molnar, G. Horton: *Proxel-Based Simulation of Stochastic Petri Nets.* Simulation und Visualisierung 2003. SCS Verlag 2004.

[6] Vidyadhar G. Kulkarni: *Modeling and Analysis of Stochastic Systems,* Chapman & Hall/CRC, 1996.

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

[8] Gianfranco Ciardo, Andrei S. Miner: *Storage alternatives for large structured state spaces*, Proc. 9th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, LNCS 1245, pages 44–57, St. Malo, France, June 1997.

[9] Daniel D. Deavours and William H. Sanders: *"On-the-fly" solution techniques for stochastic Petri nets and extensions*. In Proc. 7th Int. Workshop on Petri Nets and Performance Models, (PNPM'97), pages 132–141, St. Malo, France, June 1997.

[10] Antti Valmari: *The state space explosion problem*. In *Advances in Petri Nets*. Springer-Verlag, 1999.

[11] Boudewijn R. Haverkort, Raymond Marie, Gerardo Rubino, Kishor Shridharbhai Trivedi: *Performability Modelling : Techniques and Tools.* John Wiley & Sons, Ltd, 2001.

[12] Sanja Lazarova-Molnar, Graham Horton: *An Experimental Study of the Behaviour of the Proxel-Based Simulation Algorithm.* Simulation und Visualisierung 2003. SCS Verlag 2003.

[13] Claudia Isensee, Graham Horton: *Approximation of Discrete Phase-Type Distributions*. 38th Annual Simulation Symposium, San Diego, CA, USA, 2005.

SANJA LAZAROVA-MOLNAR was born in Skopje, Macedonia and went to "Sts. Cyril and Methodius" University in Skopje where she obtained her degree in Computer Science in 2000. She continued her studies in Magdeburg from where she obtained her Masters degree in Computational Visualistics in 2002. Since 2002 she is a member of the Simulation and Modelling group at the university in Magdeburg working towards her PhD in the field of Simulation. She can be reached at sanja@sim-md.de

GRAHAM HORTON studied Computer Science at Erlangen University, Germany and graduated with a Masters degree in 1989, followed by PhD and Habilitation degrees in 1991 and 1998, respectively. Since 2001 he is Professor of Simulation at the Computer Science Department at the University of Magdeburg, Germany. He can be reached at graham@sim-md.de