The Evolution of Stochastic Petri Nets

Antonio Puliafito[1], Miklos Telek[2], Kishor S. Trivedi[3]

[1] Istituto di Informatica, Università di Catania
Viale A. Doria 6, 95025 Catania - Italy
E-mail: ap@iit.unict.it

[2] Department of Telecommunications
Technical University of Budapest, 1521 Budapest, Hungary
E-mail: telek@pyxis.hit.bme.hu

[3] Center for Advanced Comp. & Comm.
Department of Electrical Engineering, Duke University
Durham, NC 27708-0291 - USA
E-mail: kst@ee.duke.edu

Abstract

Analytical modeling is a crucial part in the analysis and design of computer systems. Stochastic Petri Nets represent a powerful tool, widely used for dependability, performance and performability modeling. Many structural and stochastic extensions have been proposed so as to increase their modeling power. In this paper we review the main structural and stochastic extensions of Petri nets, by providing an updated treatment of the theoretical background and the possible areas of application. In particular, we focus our attention on Stochastic and Generalized Stochastic Petri nets (SPN and GSPN), Stochastic Reward Nets (SRN) and on non-Markovian Petri nets such as Markov Regenerative Stochastic Petri Nets (MRSPN). Fluid Stochastic Petri Nets (FSPN) are discussed.

1 Introduction

Analytical evaluation of computer/communication systems is increasingly becoming an integral part of the whole design process. Many diverse model specification techniques have been proposed. Markov chain models provide suitable framework for dependability, performance and performability evaluation. But, there are some difficulties in using Markov models. First, the state space grows much faster than the number of components in the system being modeled. A large state space can make a model difficult to specify correctly. Second, a Markov model of a system is sometimes far removed from the shape and general feel of the system being modeled. System designers may have difficulty in directly translating their problem into a Markov model.

These difficulties can be overcome by using a modeling technique that is more concise in its specification and whose form is closer to a designer's intuition about what a model should look like. One such approach is the use of Petri net models [32]. Stochastic Petri Nets can be used to automatically generate an underlying Markov model, which can then be analyzed to yield results in terms of the original Petri net model. This is a case where the "user-level representation" of a system is translated into a different "analytic representation". The analytic representation is processed and the results are cast back to the user-level representation.

The analytical tractability of Markov models is based on the exponential assumption of the distribution of the holding time in a given state. This implies that the future evolution of the system depends only on the current state and, based on this assumption, simple and tractable equations can be derived for both transient and steady state analysis.

Nevertheless, the exponential assumption has been regarded as one of the main restrictions in the application of Markov models. In practice there is a very wide

range of circumstances in which it is necessary to model phenomenon whose times to occurrence is not exponentially distributed. The hypothesis of exponential distribution thus allows the definition of models which can give a more qualitative rather than quantitative analysis of real systems. The existence of deterministic or other non-exponentially distributed event times, such as timer expiration, propagation delay, transmission of fixed length packets etc. gives rise to stochastic models that are non-Markovian in nature [29].

In recent years considerable effort has been devoted to enrich Petri nets formalism in order to improve their capability to easily capture system behavior and to deal with generally distributed delays. In this paper we review the main structural and stochastic extensions of petri nets, by providing an updated treatment of the theoretical background and the possible areas of application.

The paper is organized as follows. Section 2 defines the basic Petri Net model, while structural extensions are discussed in Section 3. Sections 4 and 5 introduces stochastic extensions and shows how SPN and GSPN models may be specified and solved. Stochastic reward nets (SRN) are introduced in Section 6, along with a review of performability measures. Non-Markovian PNs are discussed in Sections 7 and 8. Fluid Models are considered in Section 9, available analysis tools are discussed in Section 10, and a few concluding remarks and some possible extensions are given in Section 11.

2 Definition of the basic Petri Net Model

A Petri Net (PN) [32] is a 6-tuple, $(P, T, A, I(.), O(.), m_0)$:

- P is a set of "places."
- T is a set of "transitions ."
- A is a set of arcs.
- *I(.)*, the "input function," maps transitions to "bags" of places, where a "bag" is a set where members are allowed to appear multiple times.
- *O(.)*, the "output function," maps transitions to "bags" of places
- m_0 is the initial "marking" of the net, where a "marking" is the number of "tokens" contained in each place.

A transition t_i is said to be "enabled" by a marking m if and only if $I(t_i)$ is contained in m. Any transition t_i enabled by marking m_j can "fire". When it does, a token is removed from each place $I(t_i)$ and added to each place $O(t_i)$. This may result in a new marking m_k. If a marking enables more than one transition, the enabled transitions are said to be in conflict. Any of the enabled

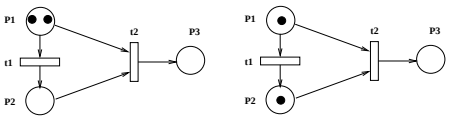

Figure 1: A PN before and after the firing of transition t_1

transitions may fire first. This firing may disable transitions which were previously enabled.

Petri nets are generally represented graphically. Places are drawn as circles and transitions as bars. The input and output functions are represented by directed arcs from places to transitions and transitions to places, respectively. Tokens are represented by black dots or numbers inside places.

The left side of Figure 1 shows a simple Petri Net with two markings. The PN has three places (*P1*, *P2* and *P3*) and two transitions (*t1* and *t2*). The input function maps *t1* to *P1* (the arc from *P1* to *t1*) and *t2* to *P1* and *P2*. The output function maps *t1* to *P2* (the arc from *t1* to *P2*) and *t2* to *P3*. In the intial marking there are two tokens in *P1*. This marking enables *t1*. The transition *t2* is not enabled because there is no token in place *P2*.

When transition t_1 fires, a token is removed from place P_1 and a token is deposited into P_2. The right hand side of Figure 1 shows the PN after transition *t1* has fired. In the new marking, both t_1 and t_2 are enabled and they are in conflict.

The analysis of Petri Nets revolves around investigating the possible markings. The Petri Net semantic does not state which of multiple simultaneously enabled transitions fires first, so a Petri Net analysis must examine every possible firing order.

Petri Nets can be used to capture the behavior of many real-world situations including sequencing, synchronization, concurrency, and conflict. The main feature which distinguishes PNs from queueing networks is the ability of the former to represent concurrent execution of activities. If two transitions are simultaneously enabled, this means that the activities they represent are proceeding in parallel. Transition enabling corresponds to the *starting* of an activity, while transition firing corresponds to the *completion* of an activity. When the firing of a transition causes a previously enabled transition to become disabled, it means that the interrupted activity was aborted before being completed.

3 Petri Net Structural Extensions

Many extensions to PNs have been proposed to increase either the class of problems that can be represented or

their capability to deal with the common behavior of real systems. In [13] Ciardo et al. define "modeling power" to be the ability of a Petri net formalism to capture the details of a system by allowing different types of firing time distribution functions. They also define "modeling convenience" to be the practical ability to represent common behavior by the introduction of structural extensions to the basic Petri net formalism. "Decision power" is defined to be the set of properties that can be analyzed. The generally accepted conclusion is that increasing the modeling power decreases the decision power. Thus each possible extension to the basic PN formalism requires an in depth evaluation of its effect upon modeling and decision power. An excellent overview of these tradeoffs can be found in [32].

Extensions which affect only modeling convenience can be removed by transforming an extended PN into an equivalent PN so they can usually be adopted without introducing any analytical complexity. These kind of extensions provide a powerful way to improve the ability of PNs to model real problems. Some extensions of this type have proven so effective that they are now considered part of the standard PN definition. They are:

- arc multiplicity
- inhibitor arcs
- transition priorities
- marking-dependent arc multiplicity

Arc multiplicity is a convenient extension for representing the case when more than one token is to be moved to or from a place. If the multiplicity of the input arc connecting place p to transition t is n, it means that transition t is enabled if and only if there are at least n tokens in place p. When transition t fires, n tokens are removed from place p. If the multiplicity of the output arc from the transition t to the place p is n, then when t fires, n tokens are deposited in place p. The standard notation is to denote multiple arcs as a single arc with a number next to it giving its multiplicity.

Inhibitor arcs are another useful extension of standard PN formalism. An inhibitor arc from place p to transition t disables t for any marking where p is not empty. Let $H(.)$ indicate the inhibition function mapping transitions to bags of places. If inhibitor arcs are allowed, a transition t_i is said to be enabled by marking m if and only if $I(t_i)$ is contained in m and for each $p_j \in H(t_i)$ (i.e. $\forall j$ such that $h_j(t_i) > 0$), then $m_j < h_j(t_i)$. Graphically, inhibitor arcs connect a place to a transition and are drawn with a small circle instead of an arrowhead. It is possible to use the arc multiplicity extension in addition to inhibitor arcs. In this case a transition t is disabled whenever place p contains at least as many tokens as the multiplicity of the inhibitor arc. Inhibitor arcs are used to model contention of limited resources to represent situations in which one activity must have precedence over another.

Another way to represent the latter situation is by using *transition priorities*, an extension in which an integer "priority level" is assigned to each transition. A transition is enabled only if no higher priority transition is enabled. However, the convenience of priorities comes at a price. If this extension is introduced, the standard PNs ability to capture the entire system behavior graphically is partially lost.

Practical situations often arise where the number of tokens to be transferred (or to enable a transition) depends upon the system state. These situations can be easily managed adopting *marking-dependent arc multiplicity*, which allows the multiplicity of an arc to vary according to the marking of the net. Marking dependent arc multiplicities allow simpler and more compact PNs than would be otherwise possible in many situations. When exhaustive state space exploration techniques are employed, their use can dramatically reduce the state space.

As an example, consider the PN model for a ISDN channel depicted in Figure 2, which was originally presented in [36]. The system behaves as follows:

- Voice and data packets arrivals are modelled through transitions $Tarrival - voice$ and $Tarrival - data$, respectively;
- Voice and data processing time are modeled though transitions $Tser - voice$ and $Tser - data$;
- The transmitter contains a buffer (place $data$) to store a maximum of k data packets. This is modeled by the inhibitor arc from place $data$ to transition $T_{arrival-data}$ with multiplicity k;
- A voice packet can enter the channel (place $voice$) only if there are no packets waiting to be transmitted. This is modeled by the inhibitor arcs to transition $T_{arrival-voice}$.
- If a voice transmission is in progress, data packets cannot be serviced, but are buffered. This priority mechanism is modeled by the inhibitor arc from place $voice$ to transition $T_{ser-data}$;
- The data buffer can eventually be flushed, if some asynchronous event (transition $flush$) occurs. This is modeled by the marking-dependent arc multiplicity between place $data$ and transition $remove$.

4 Stochastic Petri Nets

In the previous section, we discussed structural extensions to the basic Petri Net model. They increased the convenience of the Petri Net representation without increasing the modeling power. That is, even with the extensions, the PN models can provide only a qualitative

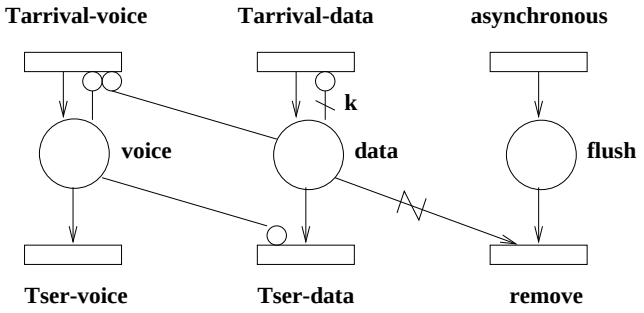

Figure 2: Using PN structural extension to model an ISDN channel

analysis; they cannot capture time dependencies. Modeling power can be increased by associating random firing times with either the places or the transitions. When waiting times are associated with places, a token arriving into a place enables a transition only after the place's waiting time has elapsed. When waiting times are associated with transitions, a transition becomes enabled as soon as all of its input place contained a token, but it fires only when it was enabled for the period of the waiting time, which is referred to as firing time in this case.

Stochastic Petri Net (SPN) models ([30]) increase modeling power by associating exponentially distributed random firing times with the transitions. A transition's firing time represents the amount of time required by the activity associated with the transition. It is counted from the instant the transition is enabled to the instant it actually fires, assuming that no other transition firing affects it.

Like basic PN models, SPN models can have more than one transition enabled at a time. To specify which transition will fire among all of those enabled in a marking, an "execution policy" has to be specified. Two alternatives are the "race policy" and the "preselection policy". Under the race policy, the transition whose firing time elapses first is assumed to be the one that will fire. Under the preselection policy, the next transition to fire in a given marking is chosen from among the enabled transitions using a probability distribution independent of their firing times. SPN models use the race policy; we will not consider the preselection policy any more.

Once a policy for choosing among enabled transitions has been determined, a "memory policy" is needed for what happens to the enabled transitions that did not fire. With the new marking, some of these may still be enabled and some may no longer be enabled. In [1] the semantics of different memory policies has been discussed and the following alternatives have been proposed and examined:

- **Resampling policy**. All enabled transitions resample a new firing time from their distribution. This does not seem to model practical situations.
- **Enabling memory policy**. If a transition is still enabled, its firing time is not resampled but is reduced by the time it has been enabled without interruption. Transitions that have been disabled have no memory of the amount of work already done and must resample a new firing time once they are reenabled. This is the policy used for SPN models.
- **Age policy**. As with the enabling memory policy, transitions that remain enabled do not resample their firing time; their firing time is reduced by the amount of time they have been enabled. However, with this policy a transition that was enabled and becomes disabled without firing retains its memory of the work already done and picks up from there without resampling when it becomes enabled again. This is more realistic than the enabling memory policy, but also more difficult to implement. Details on this memory policy and a way to implement it can be found in [8].

Generalized Stochastic Petri Nets (**GSPN**), proposed by Ajmone Marsan et al. ([2]), are an extension of Stochastic Petri Nets obtained by allowing the transitions of the underlying PN to be immediate as well as timed. Immediate transitions (drawn as thin black bars) are assumed to fire in zero time once enabled. Timed transitions (represented by rectangular boxes or thick bars) are associated with an array R of rates just as in SPNs.

When both immediate and timed transitions are enabled in a marking, only the immediate transitions can fire; the timed transitions behave as if they were not enabled. When a marking m enables more than one immediate transition, it is necessary to specify a probability mass function according to which the selection of the first transition to fire is made. The markings of a GSPN can be classified into "vanishing" markings, in which at least one immediate transition is enabled, and "tangible" markings, in which no immediate transitions are enabled. The reachability graph of a GSPN can be converted into a CTMC by eliminating vanishing markings and solved using known methods, as described in the following section.

5 SPN and GSPN Analysis

Let *M(t)* be the marking of a stochastic Petri net at time t^+. *M(t)* is a right-continuous, piecewise constant, continuous-time stochastic process. It is called the (right continuous) "marking process" or the stochastic process "underlying" the stochastic Petri net.

The probabilistic study of a Petri net is made by examining the marking process, $M(t)$, obtained by constructing

a *reachability graph* for the net. Each node in this graph corresponds to a marking. If the firing of a transition t_g changes the marking of the PN from m_1 to m_2, an arc is drawn from the node corresponding to marking m_1 to the node corresponding to marking m_2. Depending on whether the transition is immediate or timed, the arcs are labeled with probabilities or rates. The transition rate from marking m_j to marking m_k is the sum of the rates of all those transitions that are enabled in the former, and whose firing generates the latter.

If a Petri net is characterized by:

- a finite set of all possible markings (*reachability set*);
- firing rates which are time-independent;
- initial marking reachable with a probability other than zero by any marking in the reachability set,

then the associated stochastic process will have a finite, stationary, irreducible and continuous time state space.

According to the type of firing time distributions allowed, the SPN formalism may correspond to a wide range of well-known stochastic processes. For example, assuming an enabling memory policy:

- If firing times are geometrically or exponentially distributed, a discrete or continuous-time Markov chain (DTMC or CTMC) is obtained, respectively [30].
- If firing times can have general distributions, a generalized semi-Markov process (GSMP) is obtained [24].
- Under certain conditions restricting the type of distribution that concurrently enabled transitions can have, a semi-Markov process (SMP) is obtained [19].

In the following we consider only exponentially distributed firing times with time-independent firing rates. Generally distributed timed transitions are introduced in the next section together with the analysis technique to be adopted.

If the memoryless property holds, the stochastic process underlying a Petri net becomes a Markov chain, so from an analytical point of view the solution of an SPN is equivalent to the solution of the associated continuous Markov chain [2].

Analyzing GSPNs is a bit more complicated because of the presence of vanishing markings. Because vanishing markings enable at least one immediate transition, vanishing markings have no effect on the state probabilities of the marking process. The *Extended Reachability Graph* (ERG) of a GSPN has to have its vanishing markings to be identified and eliminated. This results in a *reduced reachability graph* in which the vanishing states

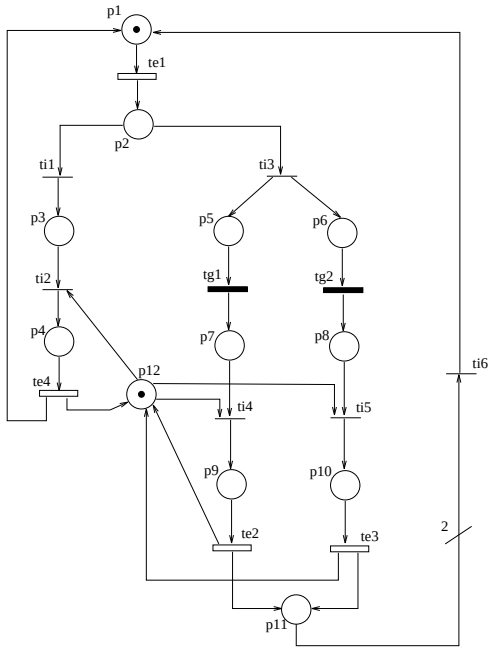

Figure 3: PN model of a Client-Server system.

are no longer present explicitly, but only by virtue of their modification of the transition rate between tangible states. The ERG thus has to be searched for vanishing markings.

Given a vanishing marking, m_b, (reached by a tangible marking, m_a) and a tangible one, m_c, which according to the reachability graph are separated by a set, K, of vanishing markings, we compute the probability P_{bc}, of activation of the path between m_b and m_c as:

$$P_{bc} = \prod_{i \in K} P_i$$

where P_i is the branching probability of the immediate transition m_i. The vanishing markings belonging to K can be eliminated by introducing an arc directly connecting m_a to m_c, and by suitably modifying the probability distribution associated with the generic transition t activated in m_a. If t is an EXP transition, the transition between m_a and m_c will have a constant rate given by:

$$\lambda_{ac} = P_{bc} \sum_{t \in \varepsilon(m_i)} \lambda_t$$

where $\varepsilon(m_i)$ is the set of EXP transitions enabled in m_i and λ_t is their firing rate.

As an example of the generation of the extended and reduced reachability graph, consider a system based on a client-server paradigm, whose Petri net model is shown in Figure 3 while the reachable markings are identified in Table 1. This model was originally presented in [33].

	$p1$	$p2$	$p3$	$p4$	$p5$	$p6$	$p7$	$p8$	$p9$	$p10$	$p11$	$p12$
$m1$	•											•
$m2*$		•										•
$m3*$			•									•
$m4$				•								
$m5$					•	•						•
$m6*$						•	•					•
$m7*$					•			•				•
$m8$						•			•			
$m9$						•					•	•
$m10$								•	•			
$m11*$								•			•	•
$m12$										•	•	
$m13*$											•	•
$m14$					•					•		
$m15$					•						•	•
$m16$							•				•	•
$m17*$									•		•	
$m18$							•			•		

Table 1: Reachable markings for the Petri net of Figure 3

The system being examined is made up of a client requesting (transition **te1**) a service which can be supplied with a probability of $1-pr$ (transition **ti3**) by two servers working parallelly, and with a probability of pr (transition **ti1**) by accessing a resource (place **p12**) shared by the two servers. In the first case a request forwarded by the client is split (*fork*) into two subrequests each addressed to a different server (places **p5** and **p6**). It is assumed in the definition of the model that a generic I/O request is considered to be concluded when all the servers have served the subrequests they are assigned (fork-join synchronization). When a server has processed its subrequest, it accesses the shared resource to record its processing results (transitions **te2** and **te3**). When both servers have accessed the shared resource and the information requested is thus reconstructed and available, a join operation is performed and the processed result is returned to the client (transition **ti6**). With the probability pr, on the other hand, it is assumed that the information requested by the client is already available in the shared resource, so the request is met by accessing the resource, retrieving the data and communicating it to the client (transitions **ti2** and **te4**).

The reachability and reduced reachability graphs are shown in Figures ??a and b; the states $m2$, $m3$, $m6$, $m11$, $m13$ and $m17$ are vanishing states which thus disappear from the reduced reachability graph. The initial marking is given by $m1 = (1, 0, 0, 0)$. Once the reduced reachability graph is obtained, the generator matrix for the underlying CTMC can be constructed as follows:

Let P^{VV} be the matrix such that $P^{VV}_{i,j}$ is the probability of transition from a vanishing marking i to vanishing marking j. Let P^{VT} be the matrix such that $P^{VT}_{i,j}$ is the probability of transition from a vanishing mark-

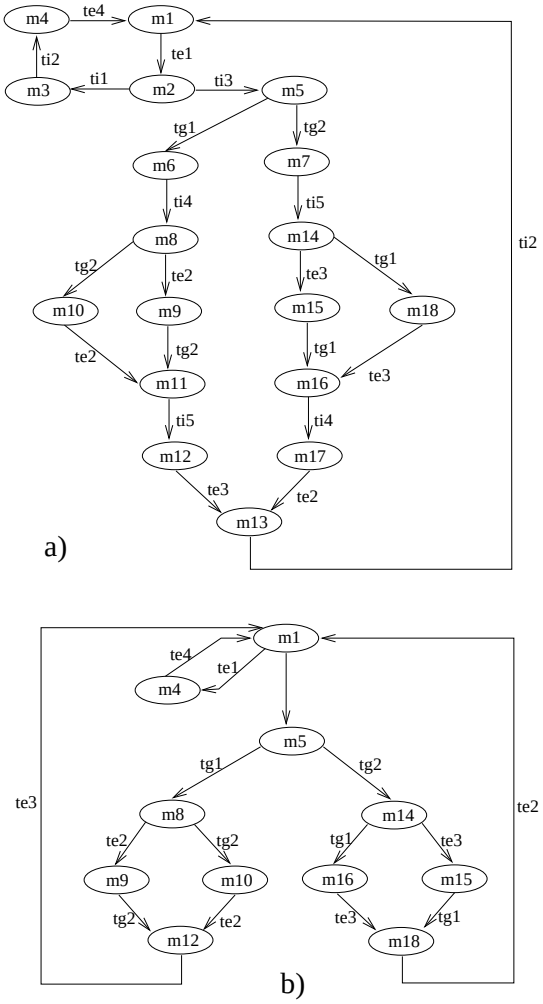

Figure 4: Extended and reduced reachability graphs of Client Server system.

ing i to tangible marking j. Let U^{TV} be the matrix in which $U^{TV}_{i,j}$ denotes the rate at which a tangible marking i changes to vanishing marking j. Let U^{TT} be the matrix in which $U^{TT}_{i,j}$ denotes the rate at which a tangible marking i changes to tangible marking j. Then $U = U^{TT} + U^{TV}(1 - P^{VV})^{-1}P^{VT}$ is the matrix describing the rate of transition from each tangible marking to other tangible markings. The diagonal entries are ignored and the generator matrix is defined as:

$$Q_{i,j} = \begin{cases} U_{i,j} & \text{if} \quad i \neq j \\ -\sum_{k \in T, k \neq i} U_{i,k} & \text{if} \quad i = j \end{cases}$$

Standard solution techniques for steady state and transient analysis of Markov chains can be adopted from this point on. The Markov chain analysis gives probabilities for the GSPN markings. From these probabilities and the reachability graph, the following transient and steady-state measures can be computed:

- throughput and utilization for a transition
- expected number of tokens in a place
- probability to have n token in a place

By properly combining the previous measures, more complex measures can be obtained, according to the problem to be solved.

6 Stochastic Reward Nets

Stochastic Reward Nets (SRN) introduce a stochastic extension into *SPNs* consisting of the possibility to associate reward rates with the markings. The reward rate definitions are specified at the net level as a function of net primitives like the number of tokens in a place or the rate of a transition. The underlying *Markov model* is then transformed into a *Markov reward model* thus permitting evaluation of not only performance and availability but also a combination of the two. The SPNP tool, which implements this extension of Petri nets, allows the reachability graph to be generated automatically and easily provides dependability, performance and performability measures [14].

6.1 Markov reward models

A Markov reward model consists of a continuous parameter Markov chain $Z = \{Z(t), t \geq 0\}$ with a finite state space S, and of a *reward function* r with $r : S \rightarrow R$. Z is completely described by its infinitesimal generator matrix Q and by the initial probability vector $\pi = [\ldots, \pi_i, \ldots]$. For each state $i \in S$, $r(i)$ represents the reward obtained per time unit. In general, the reward associated with a state denotes the performance in that state. A negative value of the reward index denotes a loss, rather than a gain. In general, we can think of reward functions $r(i,t)$, also dependent on time. The reward definition we use is called *rate-based* reward, to indicate that the system produces at a rate $r(i)$ for all the time it remains in state $i \in S$. We can also speak of impulse-based reward models, for which we define the reward function $r : S \times S \rightarrow R$, which associates a reward $r(i,j)$ to each transition from state $i \in S$ to $j \in S$. Each time a transition from state i to j occurs, the cumulative reward of the system instantaneously increases by $r(i,j)$. Besides, these rewards can depend on time $r(i,j;t)$. In general, several combinations of the different reward function definitions can exist within the same model. Henceforth we will only refer to (time-independent) rate-based Markov reward models.

According to the application to manage, we can give different meanings to states, rates and rewards in an MRM. From a Markov reward model we can get four types of measures: measures in *steady state*, transient measures, cumulative measures and distributions of cumulative measures. In the following treatment, let $\Pi = [\ldots, \pi_i, \ldots]$ denote the vector of state probabilities at steady state, $\Pi(t) = [\ldots, \pi_i(t), \ldots]$ the vector of state probabilities at time t, and $X(t) = r_{Z(t)}$ the system reward rate at time t.

Measures at steady states and transient conditions Measures at steady state can be obtained if the vector of steady state probabilities of the system is known. The expected reward rate in steady state $E[X]$ is a typical measure:

$$E[X] = \sum_{i \in S} r_i \pi_i \tag{1}$$

Once we get the vector of state probabilities at time t, we can compute the expected instantaneous reward as

$$E[X(t)] = \sum_{i \in S} r_i \pi_i(t) \tag{2}$$

Cumulative measures Cumulative measures denote the amount of work supplied by the system during a certain interval $[0,t]$.These measures can be obtained by integrating transient measures during the interval:

$$Y(t) = \int_0^t X(\tau) d\tau \tag{3}$$

It is often useful to evaluate the time averaged cumulative reward $W(t) = \frac{Y(t)}{t}$, which can also be seen as the average rate according to which the reward is accumulated at time t. Applying one or the other depends on whether absorbing states are present or not in the system structure state model [37].

From (2) we can get:

$$E[Y(t)] = E[\int_0^t X(\tau)d\tau] = \int_0^t E[X(\tau)]d\tau = \sum_{i \in S} r_i \int_0^t \pi_i(t) \quad (4)$$

the expected value of the accumulated reward $Y(t)$ until time t. The vector $\Pi(t) = [\ldots, \pi_i(t), \ldots]$ is obtained solving the system $d\Pi/dt = \Pi(t) \cdot Q$. Yet, if we do not need to know (for other purposes) state probabilities $\pi_i(t)$ at time t, we can avoid the computational overhead required by their evaluation. Let

$$L_i(t) = \int_0^t \pi_i(\tau)d\tau$$

the expected total time in which a CTMC remains in state i during the interval $(0,t)$; let $\mathbf{L}(t)$ denote the row vector consisting of elements $L_i(t)$. Integrating both sides of the *Kolmogorov* differential equation for continous parameter Markov chains we get the following relation:

$$\frac{dL(t)}{dt} = L(t)Q + \Pi(0)$$

We can rewrite (4) as:

$$E[Y(t)] = \sum_{n=0}^{n} r_i L_i(t) = rL^T(t)$$

Once $E[Y(t)]$ is known, $E[W(t)]$ can be computed as $E[Y(t)]/t$.

Distribution of cumulative measures The knowledge of cumulative measures enables us to decide if the system is able to perform a certain amount of work y in a finite interval $(0,t)$. If $F(t,y) = P\{Y(t) \leq y\}$ denotes the distribution function of the accumulated reward, the complementary distribution:

$$F^c(t,y) = 1 - F(t,y) = P\{Y(t) > y\}$$

enables us to answer the following question: "*What is the probability that the system can perform a certain amount of work in a specified time interval?*". If we know the probability that the system produces more than y in the interval $(0,t)$, we can characterize the system completely. In fact, if we begin our study of a system only analyzing the expected values of the measures of interest, we can often achieve less significant results; in fact, the average value does not give any indication about the probability of occurrence of the single values, and thus prevents us to grasp so called *tail effects*. In [37] we find several examples of multiprocessor systems, revealing a behavior that is not deducible from the mere analysis of the average values.

7 Memory policies in non-Markovian SPNs

To define a Stochastic Petri Net with generally distributed firing times besides the distribution function of the firing time $G_g(x)$ of the transition t_g, the instance of time when the firing time is sampled from this distribution and the dependence of the remaining firing time on the preceeding enabled periods of the transition also need to be specified.

Up to 1993, most definitions of non-Markovian SPNs were based on the Phase-type approximation of the general distributions. The firing time was implicitly assumed to be *resampled* each time the corresponding transition becomes enabled after firing or a restart of the firing time accumulation. The recent approach based on the Markov renewal theory allows the consideration of a more general definition/classification of the behavior of timed transtions with generally distributed firing time. This concept is based on the careful examination of the memory of a general transition through which it effects the future evolution of the marking process of the non-Markovian SPN.

In general, the memory of a single transition can be considered as composed by two elements: an *active time* and a *resampling time*.

1. *The active time:*
   is the time since the general transition was enabled after the last firing of the transition or a reset of the activity of the transition (i.e., assuming that a timed transition represents the completion of an activity with a random time requirement, then the active time is the amount of time while the system tries to complete this activity.) To account for the real executed activity of a transition t_g an age variable a_g is introduced. This age variable is reset to 0 at the end of each active period, and always positive inside an active period [8].
2. *The resampling time*:
   is the time during which the actual firing time level γ_g maintains its value without a resampling from its distribution $G_g(x)$ (i.e., the *resampling time* counts the time from which the current value of the firing time level was set).

By combining the *active time* with the *resampling time* Bobbio et al. construct the firing policy in which the age variable a_g can be reset, but the firing time γ_g remains effective and unchanged in the successive enabling periods [5]. This policy was not considered earlier as the phase-type approximation is not able to capture the stochastic behavior of a general transition with this policy. A possible manifestation of this new firing policy, is the *preemptive repeat identical (pri)* policy, that has been formulated for the first time, in the context of *SPN*, in

[5].

In general, the memory policy must be formulated in terms of a combination of the *active time* and the *resampling time*. By borrowing terminology from the queueing theory, the following taxonomy can be introduced.

A *PN* transition can be:

- *Preemptive Resume (prs)*:
  The active time and the sampled time are synchronized. They can be concluded only by the firing of a transition.
- *Preemptive Repeat Different (prd)*:
  The active time and the sampled time are synchronized. They are concluded every time the transition fires or become disabled.
- *Preemptive Repeat Identical (pri)*:
  The active time and the sampled time are not synchronized. The active time is concluded every time the transition fires or become disabled, but the sampled time can be concluded only by the firing of a transition.

8 Markov Regenerative Stochastic Petri Nets

The first definition of the class of Markov Regenerative Stochastic Petri Nets (MRSPN) comes from [12]:

Definition 1 *A SPN is called a Markov Regenerative Stochastic Petri Net if its marking process is a MRGP.*

MRGPs [26] (or Semi Regenerative Processes [16]) are discrete state continuous time stochastic processes with embedded Regenerative Time Points [39], at which the process enjoys the Markov Property. Based on the concept of memory of the general transitions, Regenerative Time Points (RTPs) can be defined as follows:

Definition 2 *A regeneration time point in the marking process is an instance of time when all the active and sampled time intervals of the general transitions are concluded.*

The importance of this definition comes from the fact that MRSPNs can be studied by the results available for MRGPs [16, 26]. The analysis methods of MRSPNs published in the literature so far are based on one of the following approaches:

- Markov renewal theory [12, 8]
- Method of supplementary variable [22, 20]
- Approximate analysis by phase type expansion [7, 18]

8.1 Analysis by Markov Renewal Theory

By the memoryless property of the MRGPs in the RTPs the analysis of a MRSPN can be divided into independent subproblems which are the stochastic (subordinated) processes between the consecutive RTPs, called regeneration periods. The measures required for the transient analysis of MRSPNs based on the Markov regenerative theory are commonly refered to as global and local kernels. The global kernel describes the occurence of the consecutive RTP:

$$K_{ij}(t) = Pr\{M_{(1)} = j\,,\, \tau_1^* \leq t | \,\mathcal{M}(0) = i\}$$

where $\mathcal{M}(t)$ denotes the marking process τ_1^* is the next RTP and $M_{(1)}$ is the right continuous state of the marking process at the next RTP. The local kernel describes the state transitions probabilities before the next RTP:

$$E_{ij}(t) = Pr\{\mathcal{M}(t) = j\,,\, \tau_1^* > t | \,\mathcal{M}(0) = i\}$$

This measure is a function of the memory policy of the transition dominating the regenerative period, i.e. the transition whose sampling period coincidences with the tagged regenerative period. For a prd type general transition the analysis is given in [12], for a prs type in [8] and for a pri type in [5].

Based on the global and the local kernels the transient analysis can be carried out in the time domain :

$$V_{ij}(t) \;=\; E_{ij}(t) + \sum_k \int_0^t d\,K_{ik}(y)\; V_{kj}(t-y)$$

or in the transform domain :

$$\mathbf{V}^{\sim}(s) \;= [\mathbf{I} - \mathbf{K}^{\sim}(s)]^{-1}\; \mathbf{E}^{\sim}(s)$$

where $V_{ij}(t)$ denotes the state transition probability over $(0,t)$, i.e.:

$$V_{ij}(t) = Pr\{\mathcal{M}(t) = j \,|\, \mathcal{M}(0) = i\}$$

and $\mathbf{V}^{\sim}(s)$ is the Laplace-Stieltjes transform of the transition probability matrix.

For the purpose of the steady state analysis an MRSPN the following measures of the subordinated processes should be known:

$$\alpha_{ij} \;= E[\int_0^{\infty} I_{\mathcal{M}^{(i)}(t)=j} dt]$$

$$\pi_{ij} = Pr\{M_{(1)} = j \mid \mathcal{M}(0) = i\}$$

α_{ij} is the expected time a subordinated process starting from state i spends in state j, and π_{ij} is the probability that the subordinated process starting from state i is followed by a subordinated process starting from state j.

Indeed the matrix $\mathbf{\Pi} = \{\pi_{ij}\}$ is the transition probability matrix of the *DTMC* embedded at the RTPs.

The analysis of these measures are also conditional to the type of the dominant transition of the subordinated processes. For a prd type general transition the analysis is given in [3], for a prs type in [38] and for a pri type in [9]. These measures can also be obtained from the global and local kernels either in the time or the transform domain:

$$\alpha_{ij} = \int_{t=0}^{\infty} E_{ij}(t)dt = \lim_{s \to 0} E_{ij}^{\sim}(s)/s$$

$$\pi_{ij} = \lim_{t \to \infty} K_{ij}(t) = \lim_{s \to 0} K_{ij}^{\sim}(s)$$

The steady state analysis of an MRSPN based on these measures is a 2 step method:

Step 1: Evaluate $P = \{p_i\}$ the unique solution of:

$$P = P\mathbf{\Pi} \quad ; \qquad \sum_i p_i = 1$$

Step 2: The steady-state probabilities of the *MRGP* become:

$$v_j = \lim_{t \to \infty} Pr\{\mathcal{M}(t) = j\} = \frac{\sum_k p_k \ \alpha_{kj}}{\sum_k p_k \ \alpha_k}$$

8.2 Method of Suplementary Variables

The **marking process** ($\mathcal{M}(t)$) together with the **age variable** (A) of the dominant transition of an MRSPN with at most one active prd or prs type general transition is a **Markov process** over the state space $S \times R$, where S is the set of reachable tangible markings and R is the (sub)set of positive real numbers. The joint process can be analyzed by the method of supplementary variables [17] as shown in [22]. Following the concept and the notations of [20] we briefly summarize the approach.

The state space is divided in two parts. S^E is the set of states in which no general transition is active ($A \doteq 0$), and S^g is the set of states in which one general transition is active. The superscript ^{E} refers to the states in S^E and the superscript ^{g} (or ^{h}) refers to the states in S^g. The probability of being in state n at time t is $\pi_n(t) = Pr[\mathcal{M}(t) = n]$. The so called, "age rate" describes the state probability together with the age value at time t:

$$p_n(t,x) = \frac{Pr[\mathcal{M}(t) = n, x < A \leq x + dx]}{dx} \cdot \frac{1}{1 - F^g(x)}$$

The firing time distribution of transition t_g is $F^g(x)$. And the matrix, referred to as branching probability matrix, describes the state transition due to the firing of a general transition is denoted by $\Delta_{i,j}^{g,h}$

With the of use proper vectors and matrices a system with prd transitions is characterized as follows. A partial differential equation describes the process evolution in S^g

$$\frac{\partial}{\partial t}\mathbf{p^g}(t,x) + \frac{\partial}{\partial x}\mathbf{p^g}(t,x) = \mathbf{p^g}(t,x)\mathbf{Q^g}$$

An ordinary differential equation describes the process evolution in S^E.

$$\begin{aligned} \tfrac{d}{dt}\pi^{\mathbf{E}}(t) \quad = \quad & \pi^{\mathbf{E}}(t)\mathbf{Q}^{\mathbf{E},\mathbf{E}} + \\ & \sum_g \int_0^{\infty} \mathbf{p^g}(t,x)dF^g(x)\mathbf{\Delta}^{\mathbf{g},\mathbf{E}} + \\ & \sum_g \pi^{\mathbf{g}}(t)\mathbf{Q}^{\mathbf{g},\mathbf{E}} \end{aligned}$$

State probabilities can change by the firing of an exponential transition (1st term). After firing of a general transition only exponential transitions are enabled (2nd term). The same phenomenon occurs after disabling a general transition (3rd term).

The boundary condition is given by:

$$\begin{aligned} \mathbf{p^g}(t,0) \quad = \quad & \pi^{\mathbf{E}}(t)\mathbf{Q}^{\mathbf{E},\mathbf{g}} + \\ & \sum_h \int_0^{\infty} \mathbf{p^h}(t,x)dF^h(x)\mathbf{\Delta}^{\mathbf{h},\mathbf{g}} + \\ & \sum_h \pi^{\mathbf{h}}(t)\mathbf{Q}^{\mathbf{h},\mathbf{g}} \end{aligned}$$

A general transition g can be activated by the firing of an exponential transtion in S^E (1st term), by the firing of a general transition (2nd term) or by disabling the active general transtion (3rd term).

Probabilities of states with active general transition is given by:

$$\pi^{\mathbf{g}}(t) = \int_0^{\infty} \mathbf{p^g}(t,x)(1 - F^g(x))dx$$

Finally the initial conditions are $\pi^{\mathbf{E}}(0)$ and $\mathbf{p^g}(0,x) = \pi^{\mathbf{g}}(0)\delta(x)$

The analysis of the transient behavior by the supplementary variable approach is based on a numerical evaluation of the system of equations. An iterative algorithm based on the fixed size (h) discretization of the continuous variables proposed by German et al. [23] consists of the following steps:

1. Compute age rates in the next time instant

   $$\mathbf{p^g}(ih, jh) = \mathbf{p^g}((i-1)h, (j-1)h)e^{\mathbf{Q^g}h}$$

   and set $\mathbf{p^g}(ih, 0) = 0$
2. Compute the state probabilities $\pi^{\mathbf{g}}(ih)$ by $\mathbf{p^g}(ih, jh)$, $j = 0, 1, \ldots$

3. Compute the state probabilities $\pi^{\mathbf{E}}(ih)$ by the ordinary differential equation
4. Compute the activation rate of general transitions $\mathbf{p}^{\mathbf{g}}(ih, 0)$ by the boundary conditions
5. Check the convergence and go back to step 2 or start with the next time instant $(i+1)h$

The transient behavior of an MRSPN by the supplementary variable approach can be easily obtained by vanishing the derivatives according to the time in the above set of equations. The analysis based on the obtained description requires the transient analysis of the state probabilities and the cumulative state probabilities of the subordinated CTMCs.

8.3 Phase type expansion

The set of non-Markovian Stochastic Petri Nets with prs or prd type transitions can be approximately analyzed by the method of phase type expansion. When the firing times are all phase type distributed [31], this approach gives the exact solution.

The analysis method is composed by the following steps:

Step 1: Approximate firing time distribution of all the timed transitions by a Phase type distribution (an overview of approximation methods and tools can be found in [6]).

Step 2: Based on the net description, the phase type model and the memory policy of the transitions compose the expanded state model of the stochastic process which is a CTMC over the state space $S \times T_1 \times \ldots \times T_n$, where S is the set of reachable tangible markings and T_g is the set of the phases of the phase type model of transition t_g.

Step 3: Analyze the expanded CTMC.

Step 4: Evaluate the marking probabilities and the other required Petri net measures.

Cumani [18] has developed a package called ESP which automatically performes Step 2 - 4.

9 Fluid Stochastic Petri Nets

Recognizing the increasing use of stochastic fluid flow models in performance analysis, Trivedi and Kulkarni introduced the class of Fluid Stochastic Petri Nets [40]. In this class they extend the traditional integer token concept by introducing real (positive) tokens assigned to the continuous places of the net. This way a marking of an FSPN is described by a vector of integer and real numbers representing the number (amount) of tokens (token) m_d (m_c) at the discrete (continuous) places P_d (P_c), where m_d (m_c) is a vector with size $\#P_d$ ($\#P_c$).

In the first model defined in [40] the continuous part of a marking does not affect the discrete stochastic process defined by the number of tokens in the discrete places. i.e. the evolution of the discrete part of the marking (m_d) is through discrete transitions where firing rates depend only on m_d. The evolution of the continuous part of the marking m_c is governed by flow rate functions which depend only on m_d. Let S be the set of reachable discrete markings ($m_d(t) \in S$) and Q be the infinitesimal generator of discrete part of the marking process (which is a CTMC). The fluid level at place $i \in P_c$ at time t is characterized by the resultant flow rate depending on the discrete state $n \in S$:

$$\frac{dX_i(t)}{dt} = \begin{cases} r_i(n) & \text{if} \quad X_i(t) > 0 \\ max\{r_i(n), 0\} & \text{if} \quad X_i(t) = 0 \end{cases} \tag{5}$$

Define the transient distribution function $H(t, \vec{x}, n) = P\{X_i(t) \leq x_i, i \in P_c, m_d(t) = n \in S\}$ and the row vector of size $\#S$ $\vec{H}(t, \vec{x}) = \{H(t, \vec{x}, n)\}$. In accordance with the famous result of stochastic fluid models $\vec{H}(t, \vec{x})$ satisfies [40]:

$$\frac{\partial \vec{H}(t, \vec{x})}{\partial t} + \sum_{i \in P_c} \frac{\partial \vec{H}(t, \vec{x})}{\partial x_i} \mathbf{R}_i = \vec{H}(t, \vec{x})\mathbf{Q}, \quad \vec{x} > 0 \tag{6}$$

with the boundary condition $H(t, \vec{x}, n) = 0$ if $x_i = 0$ and $r_i(n) > 0$. In the expression above $\mathbf{R}_i = diag(r_i(n))$.

The steady state behavior of FSPNs is characterized by eliminating the time dependent behavior from (6). Analytical evaluation of the transient as well as the steady state behavior of FSPN with multiple fluid places is very hard. Numerical techniques are being explored. An FSPN with a single fluid place results in a traditional stochastic fluid flow model for which the steady state analysis based on the spectral decomposition of $\mathbf{Q}$ is possible [4].

An extension of the first FSPN model was proposed by Horton at al. [25]. In the extended FSPN class, mutual affecting of the continuous and the discrete parts of the marking process is allowed. Using the same notation to indicate dependence on the fluid levels at fluid places we have $\mathbf{Q}(\vec{x})$, $r_i(n, \vec{x})$ and $\mathbf{R}_i(\vec{x})$. In this case the system behavior modifies to [25]:

$$\begin{aligned} &\frac{\partial \vec{H}(t, \vec{x})}{\partial t} + \sum_{i \in P_c} \frac{\partial (\vec{H}(t, \vec{x})\mathbf{R}_i(\vec{x}))}{\partial x_i} \\ &= \vec{H}(t, \vec{x})\mathbf{Q}(\vec{x}) - \int_0^{x_F} \ldots \int_0^{x_1} \vec{H}(t, \vec{x}) \frac{\partial \mathbf{Q}(\vec{x})}{\partial x_i} dx_i \ldots dx_F \end{aligned} \tag{7}$$

with the boundary conditions $H(t, \vec{x}, n) = 0$ if $x_i = 0 \& r_i(n, \vec{x}) > 0$ and

$$\lim_{x_i \to \infty} \frac{\partial \vec{H}(t, \vec{x})}{\partial x_i} = 0 \ .$$

Numerical analysis of these FSPNs with single fluid places can be found in [25] while a discrete event simulation method is proposed by Ciardo et al. [15].

10 Some Available Petri Net Analysis Tools

Concurrent with the stochastic evolution of Petri Nets, effort has been devoted to defining and implementing automatic solution tools for dependability, performance and performability analysis of PN models. In this section we briefly indicate the most well known tools for analytic evaluation. Tools based on simulation will not be covered here.

GreatSPN [11] is based on graphical input and provides steady state as well as transient analysis of GSPNs mainly for evaluating performance measures. Structural analysis (P and T invariant) is also allowed.

ESP [18], implements an approximate solution based on the use of phase-type distributions instead of only exponential distributions. Thus, a wider class of PNs can be solved. Both *prs* and *prd* memory policies are allowed.

UltraSAN [35] is based on a class of SPN models known as stochastic activity networks and has been primarily used for performability analysis. Steady state as well as transient simulation are also available.

Hierarchical approach is adopted in SHARPE [36]. It provides a variety of probabilistic, discrete-state models used to assess the reliability and performance of computer and communication systems. GSPN is one of seven model types. Originally provided with a textual input, now a graphical interface is also available [34].

SPNP [14] is a C-based tool for the analysis of Stochastic Reward Nets, an extension of PNs which allows specification of rewards directly at the net level. Steady state, transient and cumulative measures can be computed.

DSPNexpress [28] deals with deterministic and stochastic Petri nets. Under the assumption that at most one deterministic transition is enabled in a given time instant, steady state probabilities are computed.

As an extension, TimeNET [21] allows the numerical evaluation of GSPNs and DSPN at steady state and in transient condition under the same structural constraints of DSPNexpress. Stationary approximation can be used for concurrent DSPN. Simulation is also allowed.

A new modeling tool for the analysis of non-Markovian stochastic Petri nets that relax some of the restrictions present in previously available modeling packages has been recently presented [10]. This tool, called WebSPN, provides a discrete time approximation of the stochastic behavior of the marking process which results in the possibility to analyze a wider class of Petri net models with *prd* and *prs* concurrently enabled generally distributed transitions. WebSPN is developed in Java and implements some powerful communication mechanisms which allow a generic user connected to the Internet to remotely access the tool regardless of the type of his/her hw/sw platform.

11 Conclusions

Petri nets represent a powerful tool for specifying and analyzing computer and communication systems in order to evaluate dependability, performance and performability measures. In this paper we surveyed the Petri Net literature and discussed structural and stochastic extensions. We presented Petri net notation and showed the way to deal with SPN and GSPN models. Reward extension was also discussed. We specifically focused on non-Markovian Petri nets, their solution technique and the way to deal with prs and pri memory policies. Fluid stochastic PNs were also reviewed.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-15:832–846, 1989.

[2] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[3] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Lecture Notes in Computer Science*, V. 266, pp. 132–145. Springer Verlag, 1987.

[4] D. Anick, D. Mitra, and M.M. Sondhi. Stochastic theory of data handling system with multiple sources. *Bell System Techn. J.*, 61:1871–1884, 1982.

[5] A. Bobbio, V.G. Kulkarni, A. Puliafito, M. Telek, and K. Trivedi. Preemptive repeat identical transitions in Markov Regenerative Stochastic Petri Nets. In *6-th IEEE Int. Conf. on Petri Nets and Performance Models - PNPM95*, pp. 113–122, 1995.

[6] A. Bobbio and M. Telek. A benchmark for PH estimation algorithms: results for Acyclic-PH. *Stochastic Models*, 10:661–677, 1994.

[7] A. Bobbio and M. Telek. Computational restrictions for SPN with generally distributed transition times. In D. Hammer K. Echtle and D. Powell, editors, *First European Dependable Computing Conf. (EDCC-1), Lecture Notes in Computer Science*, V. 852, pp. 131–148. Springer Verlag, 1994.

[8] A. Bobbio and M. Telek. Markov regenerative SPN with non-overlapping activity cycles. In *IEEE Int. Computer Performance and Dependability Symp. - IPDS95*, pp. 124–133, 1995.

[9] A. Bobbio and M. Telek. "Combined preemption policies in MRSPN," in *Fault-tolerant systems and software* (R. M. et al, ed.), pp. 92–98, Narosa Publishing House, New Delhi, India, 1995.

[10] A. Bobbio, A. Puliafito, M. Scarpa, and M. Telek. WebSPN: Non Markovian Stochastic Petri Net Tool In *18th Int. Conf. on Application and Theory of Petri Nets*, (Toulouse (France)), June 23-27, 1997.

[11] G. Chiola. *GreatSPN* 1.5 Software architecture. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*, pp. 121–136. Elsevier Science Publishers, 1992.

[12] H. Choi, V.G. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:337–357, 1994.

[13] G. Ciardo. Toward a definition of modeling power for stochastic Petri net models. In *Proc. IEEE Int. Workshop on Petri Nets and Performance Models*, pp. 54–62, Madison, 1987.

[14] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proc. IEEE Int. Workshop on Petri Nets and Performance Models - PNPM89*, pp. 142–151,1989.

[15] G. Ciardo, D. Nicol and K.S. Trivedi. Discrete-event simulation of Fluid Stochastic Petri Nets. *7-th Int. Conf. on Petri Net and Performance Models* (Saint Malo (France)), June 1997.

[16] E. Cinlar. *Introduction to Stochastic Processes.* Prentice-Hall, Englewood Cliffs, 1975.

[17] D.R. Cox. The analysis of non-markovian stochastic processes by the inclusion of supplementary variables. *Proc. of the Cambridge Phylosophical Society*, 51:433–440, 1955.

[18] A. Cumani. Esp - A package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proc. IEEE Int. Workshop Timed Petri Nets*, pp. 144–151, Torino (Italy), 1985.

[19] J. Dugan, K. Trivedi, R. Geist, and V.F. Nicola. Extended stochastic Petri nets: applications and analysis. In *Proc. PERFORMANCE '84*, Paris, 1984.

[20] R. German. New results for the analysis of deterministic and stochastic Petri nets. In *IEEE Int. Computer Performance and Dependability Symp. - IPDS95*, pp. 114–123, 1995.

[21] R. German, C. Kelling, A. Zimmermann, and G. Hommel. *TimeNET - A toolkit for evaluating non-markovian stochastic Petri nets.* Report No. 19 - Technische Universität Berlin, 1994.

[22] R. German and C. Lindemann. Analysis of stochastic Petri nets by the method of supplementary variables. *Performance Evaluation*, 20:317–335, 1994.

[23] R. German, D. Logothetis, and K. Trivedi. Transient analysis of Markov Regenerative Stochastic Petri Nets: a comparison of approaches. In *6-th IEEE Int. Conf. on Petri Nets and Performance Models - PNPM95*, pp. 103–112, 1995.

[24] P.J. Haas and G.S. Shedler. Stochastic Petri nets with timed and immediate transitions. *Stochastic Models*, 5:563–600, 1989.

[25] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fuid Stochastic Petri Nets: Theory, Applications and Solutions. *ICASE Techn. Rep. 96-5*, 1996.

[26] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems.* Chapman Hall, 1995.

[27] C. Lindemann. DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets. *Performance Evaluation*, 22:3–21, 1995.

[28] D. Logothetis, K. Trivedi, and A.Puliafito. Markov regenerative model. In *IEEE Int. Computer Parallel Distributed Systems - IPDS95*, Apr. 24-26 1995.

[29] M.K. Molloy. On the integration of delay and throughput measures in distributed processing models. Technical report, Phd Thesis, UCLA, 1981.

[30] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, 1981.

[31] J.L. Peterson. *Petri net theory and the modeling of systems.* Prentice Hall, Englewood Cliffs, 1981.

[32] A.Puliafito, M.Scarpa, and K.S.Trivedi. Petri nets with k simultaneously enabled generally distributed timed transitions. to appear in *Performance Evaluation*, 1997, North-Holland.

[33] A. Puliafito, O. Tomarchio, and Lorenzo Vita. Porting SHARPE on the WEB: Design and Implementation of a Network Computing Platform using JAVA., In *9th IEEE Int. Conf. on Modelling Tech. and Tools for Comp. Perf. Eval.*, (Saint Malo (France)), June 1997.

[34] W.H. Sanders, W.D. Obal II, M.A. Qureshi, and F.K. Widjanarko. The UltraSAN modeling environment. to appear in *Performance Evaluation*, North-Holland.

[35] R.Sahner, K.S.Trivedi, and A.Puliafito. *Performance and reliability analysis of computer systems: an example based approach using the SHARPE software package.* Kluwer Academic Publishers, Oct. 1995.

[36] R. Smith, K. Trivedi, and A.V. Ramesh. Performability analysis: Measures, an algorithm and a case study. *IEEE Transactions on Computers*, C-37:406–417, 1988.

[37] M. Telek, A. Bobbio, L. Jereb, A. Puliafito, and K. Trivedi. Steady state analysis of Markov regenerative SPN with age memory policy. In H. Beilner and F. Bause, editors, *8-th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation, Lecture Notes in Computer Science*, V. 977, pp. 165–179. Springer Verlag, 1995.

[38] M. Telek. *Some advanced reliability modelling techniques.* CSc/Phd Thesis, Hungarian Academy of Science, 1995.

[39] K.S. Trivedi and V.G. Kulkarni. FSPNs:fluid stochastic Petri nets. In *Proc. 14-th Int. Conf. on Application and Theory of Petri Nets*, Chicago, 1993.