Available online at www.sciencedirect.com
SCIENCE DIRECT®
Reliability Engineering and System Safety 84 (2004) 149–161
RELIABILITY ENGINEERING & SYSTEM SAFETY
www.elsevier.com/locate/ress

Modeling of system reliability Petri nets with aging tokens

V. Volovoi

School of Aerospace Engineering, Georgia Institute of Technology, 270 Ferst Dr., Atlanta, GA 30332-0150, USA

Received 12 July 2003; accepted 30 October 2003

Abstract

The paper addresses the dynamic modeling of degrading and repairable complex systems. Emphasis is placed on the convenience of modeling for the end user, with special attention being paid to the modeling part of a problem, which is considered to be decoupled from the choice of solution algorithms. Depending on the nature of the problem, these solution algorithms can include discrete event simulation or numerical solution of the differential equations that govern underlying stochastic processes. Such modularity allows a focus on the needs of system reliability modeling and tailoring of the modeling formalism accordingly. To this end, several salient features are chosen from the multitude of existing extensions of Petri nets, and a new concept of aging tokens (tokens with memory) is introduced. The resulting framework provides for flexible and transparent graphical modeling with excellent representational power that is particularly suited for system reliability modeling with non-exponentially distributed firing times. The new framework is compared with existing Petri-net approaches and other system reliability modeling techniques such as reliability block diagrams and fault trees. The relative differences are emphasized and illustrated with several examples, including modeling of load sharing, imperfect repair of pooled items, multiphase missions, and damage-tolerant maintenance. Finally, a simple implementation of the framework using discrete event simulation is described.

Keywords: Reliability; Dependability; Petri nets; Aging tokens

1. Background

A quick glance at the product line of any major vendor of system reliability software unequivocally leads to the following observation: presently, reliability block diagrams (RBDs) and fault tree analysis (FTA) are the only two frameworks widely available at the modeling phase for quantitative assessment of reliability metrics. Recent enhancements of these standard tools include more and more dynamic features, such as dependent events and spare modeling, which seemingly obviate the need for alternative, essentially dynamic, tools. In the past, it was necessary to resort to Markov chains in order to model some intricate dynamic interactions among failure modes. However, direct modeling of Markov states is unfeasible for all but very small-scale problems due to state-space explosion, and stochastic Petri nets (SPNs) can be employed as a compact preprocessor for creating larger Markov models in an automated fashion.

SPNs were proposed as a potentially attractive alternative for reliability modeling shortly after they were introduced more than two decades ago [1] and have been periodically proposed ever since [2–4]. However, modern dynamic FTAs [5–7] or some of the advanced RBDs can be also utilized in a similar fashion. Perhaps this can explain the fact that SPN applications to system reliability are mainly restricted to research, with few notable exceptions [8]. Ironically, an additional reason for this might be the very success of Petri nets in recent years, which has led to the proliferation of various ‘extensions’ of SPNs. The multitude of existing variations of Petri-net formalism might have excellent modeling capabilities, but lack a unified standard; as a result, what is referred to as SPNs can vary drastically from one application or paper to another. Needless to say, such ambiguity can be quite confusing for reliability practitioners, and by and large SPNs are perceived as a technique that is perhaps powerful but cumbersome and somewhat arcane. It is quite characteristic that SPNs are often absent from the list of compared techniques for system dependability (a measure of system performance that includes reliability, availability, and safety) [9,10]. Similarly, SPNs are only briefly mentioned (if at all) in books on the subject.

E-mail address: vitali.volovoi@ae.gatech.edu (V. Volovoi).

0951-8320/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2003.10.013

On the other hand, SPNs have evolved into a powerful and mature field of operational research [11,12] that enjoys wide applications in fields such as optical and computer networks [13] and flexible manufacturing systems [14]. To this end, there were commendable efforts to bring clarity into dependability modeling with SPNs [15] and associated terminology. One of the recommendations contained therein, which is followed in the present work was to avoid long and awkward abbreviations for a particular version of SPN: a subset of SPNs is specified and used throughout the paper, and when comparisons are made with different versions of SPNs, the distinctions are explicitly described. At the same time, the author falls short of following another recommendation in Ref. [15] of avoiding the word 'stochastic' and simply using the term 'Petri nets' instead, out of respect to a quite substantial body of literature where regular (untimed) Petri nets are successfully employed [16].

There might be yet another reason for the relative dearth of practical applications of SPNs to system reliability modeling, which is more directly addressed in the current work. The possibility of solving given SPN problems using discrete event simulation has been realized for some time and quite widely explored [17,18]. However, the development of formalism itself was tightly coupled with the solution capabilities for Markovian-type processes (which included semi-Markovian and regenerative processes). The present paper takes a fundamentally different approach: taking into consideration the fact that numerical solutions of the differential equations that govern underlying stochastic processes can be supplemented by increasingly cheap (from the computational standpoint) yet sophisticated Monte Carlo simulations [19,20], one can remove any restrictions on the modeled processes and concentrate instead on the needs of reliability engineers to model complex scenarios in a compact and transparent form. Determining the compatibility of the presented extensions of SPNs with the modern non-Markovian techniques as well as with other methods of 'direct' solutions for associated stochastic processes lies beyond the scope of the present paper, but presents an interesting subject for future research.

2. Present approach

In what follows, it is assumed that the failure distributions of individual components of a system are given, and the dependability measures (including reliability and availability) of a stochastic system are sought. Furthermore, the system is assumed to be dynamic (its properties change with time), distributions are not limited to exponential ones (unless stated otherwise), and large-scale applications should be feasible. Particular attention is paid to the descriptive power of the methods.

There is an ample body of literature describing SPNs, and for a detailed description the reader is referred to Refs. [11,12,21]. Only concepts relevant to the discussion are described below.

A Petri net is a directed graph with two disjoint types of nodes: places (denoted as circles) and transitions (denoted as rectangles). A directed arc can connect a place to a transition (an input arc) or a transition to a place (an output arc). The places connected to a given transition by input or output arcs are called the input or output places, respectively, for this transition. A common extension of SPNs introduces the concept of arc multiplicities (integer numbers). Each place can be assigned a non-negative number of tokens (denoted as small filled circles). A combined assignment for all the places in the model is referred to as a marking of the system, which fully characterizes the system.

Under certain conditions, a transition can 'fire' and remove a token (or more generally, a number of tokens that is equal to the multiplicity of the corresponding input arc) from its input place while depositing a token (number of tokens equal to the multiplicity of the corresponding output arc) to its output place. Generally speaking, the number of removed and added tokens does not have to coincide, and these two parts of firing can be considered as independent events corresponding to the 'death' and 'birth' of the tokens, respectively. However, a combination of a token's removal from an input place and a token's deposit into an output place can be also considered to be a single action of moving the token from the input to the output. If all the tokens are indistinguishable, then both interpretations are equivalent. As discussed below, for labeled tokens the difference is significant, and the latter interpretation is particularly relevant for aging tokens.

The firing of a transition corresponds to any discrete event of the modeled system, and represents a fundamental feature of Petri nets: the ability to graphically depict the dynamic behavior of a system. Naturally, a key property is the definition of firing policies for each transition; in order to fire, a transition must be enabled first. A transition is considered to be enabled if each of its input places has at least as many tokens as the multiplicity of the corresponding input arc. Most of the SPN extensions introduce a so-called inhibitor arc (denoted as an arc terminated with a small hollow circle). While these inhibitors do not increase the SPN's representational power, they can make the model more compact. In the presence of inhibitor arcs, there is an additional condition that a transition can be enabled only when all of the inhibitor input places for this transition have fewer numbers of tokens than the corresponding arc multiplicity. Original Petri nets did not contain a concept of time [22], and an enabled transition would fire instantaneously. The introduction of deterministic time delays (that is, firing takes place if a transition is enabled for a specified amount of time) has led to timed Petri nets that were later extended to SPNs where such delays are random variables based on given distributions.

In the so-called colored Petri nets [23], tokens are distinguished (labeled), which allows the implementation of different firing policies for different tokens. Moreover, by specifying different labels for a token that enters and leaves a transition, colored Petri nets permit a dynamic change of the labeling of a token upon its passing through a transition. The ability to move a token from one place to another and alter its labels in the process provides a flexible mechanism for tracking relevant changes in the system. This mechanism turns out to be particularly useful in SPNs with aging tokens. In existing high-level Petri nets, the changes in token labeling are fully determined by the firing occurrence itself, and are not affected by the transitional process; that is, a given token can change its property in a discrete fashion upon the firing of a transition, but as long as it stays in the same place its properties remain the same.

Another extension of SPNs allows for so-called marking dependence when the firing policies can depend on the marking in a quite general way. As discussed below, while the employment of this feature can be beneficial under certain circumstances, it can also lead to compromising model modularity and consequently, a lack of clarity [24]. The introduction of aging tokens can greatly reduce the need for marking dependence. The standard SPNs imply that, when a transition is disabled, it 'forgets' its previous enabled time; the clock is reset when the transition is enabled again and the corresponding firing time is resampled. This is referred to as *enabling memory* or *preemptive repeat different (prd)* policy. Alternatively, a transition can 'remember' the time that it was enabled (that is, the transition clock is stopped but not reset). This is called age memory or *preemptive resume (prs)* policy in accordance with the classification introduced in Ref. [25] (see also Refs. [21,26]). Finally, there is a third type of memory policy, *preemptive repeat identical (pri)* [27], when the clock is reset but no resampling is done, and the same sampled time is used. As shown below, these different policies can be equivalently modeled in the proposed extension of SPNs, where the memory is assigned not to the transition but to the token instead.

3. SPNs with aging tokens

The key feature of the present formulation is the introduction of tokens with memory (or aging tokens): a token's label can change continuously during the period the transition is enabled. Such tokens can coexist in conjunction with regular colored and uncolored tokens. The value of these counters can affect the firing policies of transitions that are enabled by such tokens. As in colored Petri nets, there might be more than one counter associated with a given token; for example, it can be a real number between 0 and 1 (indicating the fraction of a part's lifespan that was 'spent'), or the actual time that the token was enabling a transition. Such continuously changing counters can naturally facilitate modeling of accumulated damage, or any other property that is continuously changing with time. These aging counters can be complemented by discrete counters existing in colored nets, such as the number of repairs for the part. The latter feature can be useful, for example, where a limited number of repairs are allowed for a given part before it is scrapped. These dynamic features are only possible when a token can be 'preserved' by a transition and both parts of firing (removing and depositing) are treated as parts of a single action: moving the aging token from an input to an output place. Obviously, ambiguities must be avoided if multiple inputs and output places and/or arc multiplicities are present: the route of all the tokens that are preserved by the transition must be uniquely specified by matching (labeling) corresponding output and input arcs. A death or a birth of an aging token can still occur (just like with regular tokens).

Unlike existing SPNs with memory-enabled transitions, the current formalism assigns the memory to tokens instead, and unlike the existing colored SPNs, the token labels affecting firing policies can change continuously (i.e. the value of these labels, or counters, may change gradually while the token is enabling a transition, and not only in a discrete fashion upon the firing of a transition). Aging of a token occurs when the token enables a transition that has a matching firing policy for this type of token. Obviously, at any given marking, an aging token can enable, at most, one such transition per counter in order to define the aging of this counter in a unique fashion; however, in addition, this token can enable an arbitrary number of transitions that do not affect its age. In order to take full advantage of aging tokens, it is useful to distinguish several modes of firing for a given timed transition (stochastic or deterministic):

(1) 'Standard' mode. Once the transition is enabled, a stochastic distribution is sampled or a deterministic delay is provided, and an associated single clock starts. When this clock runs down, the transition fires a number of tokens that is equal to the multiplicity of the corresponding output arc. Tokens are chosen at random from the input place. The procedure is repeated as long as the transition remains enabled.
(2) 'Series' or first-in-first-out (FIFO) mode, similar to the standard mode in that only one clock per transition is active at any given point in time; however, the tokens are fired in the order they arrive in the input place. The distinction between FIFO and standard mode is relevant only for labeled tokens.
(3) 'Multiple' (parallel) mode, useful for a more compact description of similar processes. For each token in the input place, the firing delay is determined independently, and a corresponding separate clock is assigned. The fundamental distinction of this mode is that the firing policies are intimately related to properties of both token and transition, so that an analogy with a key (token) and a lock (transition) can be made. Both

multiple arcs for standard transitions and multiple transitions are designed to engage several tokens at the same time. However, the former essentially lump tokens into a single packet that is fired as a single token, while the latter preserve the individuality of each token. For clarity purposes, in the present formulation of multiple mode arc, multiplicity affects only enabling (and not the firing itself).

Multiple transition is the most commonly used mode in SPNs with aging tokens, and in fact it can be used to mimic the behavior of the two other modes. Indeed, for a series transition, this modeling is fairly straightforward (see Fig. 1 for an example with three tokens), although the series transition provides a more compact representation, especially if the input place, denoted as I, is an input to other transitions as well.

Remarkably, representing standard transitions is seemingly more cumbersome than series ones. Effectively, the difficulty stems from the need to remove the influence of token properties on the firing policy of the transition (Note that a token that enables a standard transition need not even be present when the transition is fired). Such a token-neutral firing using multiple transitions is possible if an auxiliary token is introduced. Fig. 2a depicts a series transition T_0 from the input place (denoted as I) to the output place (denoted as O). In Fig. 2b seven transitions, T_i, $i = 1, \ldots, 7$ are introduced to provide the same functionality (only T_4 is a multiple transition with all others being of the immediate type). As soon as a token appears at I for the first time, T_1 fires. All the tokens from I are returned there, but in addition an auxiliary token is 'born', which enables T_4 that has the same timing properties as T_0. The inhibitors are arranged in such a way that, if all tokens are removed from I, then T_3 is fired (thus disabling T_4), and conversely, if a token is moved to I, the auxiliary token is fired back (via T_2) and T_4 is enabled again. The firing of T_4 causes all the tokens from I to be fired and enable T_5 and T_7. Since T_5 has a higher priority, it fires a randomly chosen token to O. All other tokens are returned to I via T_7; a new auxiliary token is born and the clock for T_4 is restarted. Note that the use of non-aging auxiliary tokens is equivalent to *prd* policy for a standard transition, while aging auxiliary tokens can mimic *prs* policy. Furthermore, *pri* policy can be modeled using

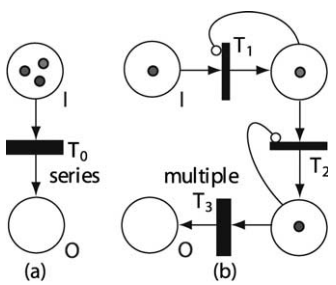


Fig. 1. Series transition with three tokens: (a) modeling using series transition; (b) equivalent modeling using multiple transitions.

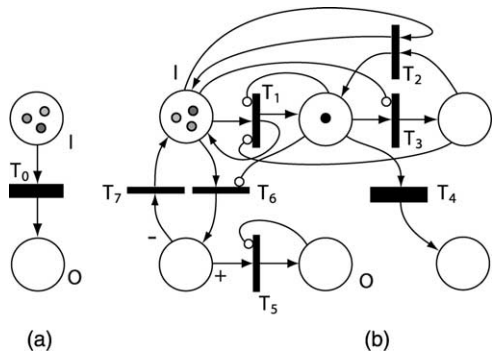


Fig. 2. Standard transition: (a) modeling using standard transition; (b) equivalent modeling using multiple transitions (diagram also provides a means to model *pri* transitions).

the same diagram as well by assigning T_4 a deterministic delay, $t_L = t_{max} + \epsilon$, where t_{max} is the upper time limit of the model and ϵ is an arbitrary small delay. The stochastic nature of the firing of T_4 is reflected in the initial age of the auxiliary token. If $F(t)$ is a cumulative distribution function (CDF) of the *pri* policy, then the following CDF for the age of the auxiliary token can be given:

$$\hat{F}(x) = \begin{cases} 1 - F(t_L[1-x]), & \text{when } x \geq 0 \\ 0, & \text{when } x < 0 \end{cases} \quad (1)$$

For immediate transitions the concept of a clock is irrelevant, and if several tokens enable an immediate transition simultaneously, the fired token is chosen at random (however, one can introduce priorities in these situations as well, for example, based on token colors), and the arc multiplicities have the standard meaning.

Introducing aging tokens into Petri-net formalism has important ramifications for system reliability modeling of aging and repairable systems. Several salient features of real-life systems can be modeled more transparently; these include aging systems with load sharing, warm spares, multiphase missions, and imperfect repair of pooled parts. In conjunction with the features that already exist in Petri nets, this provides a very powerful tool for modeling system reliability.

4. Application of aging tokens

4.1. Example A: two units in a series

To illustrate these concepts, let us consider two repairable units that are connected in series. Failure of any of the components results in the failure of the system. It is a common practice to model such a system as two RBD blocks connected in a series, or as a fault tree with two basic events connected with an 'OR' gate. What is less commonly recognized is that an assumption of the failure of these two units to be independent is in fact only an approximation when availability of the system is calculated: it is reasonable

to expect that, after one unit fails and the system goes down, the failure rate of the second unit changes (for example, goes to zero). For the quantitative implications of this, see Ref. [28]. Explicit extraneous constructions are required to model this subtle yet very simple situation with either FTAs or RBDs.

Let us first assume that the failure and repair rates are constants for both units (λ and μ, respectively), so that we can model this system using a Markov chain. As shown in Fig. 3a, there are only two possible states: O, when both units are operating, and I, where one of the units has failed and another one is idling. Obviously, the transitional rate from O to I is 2λ (corresponding to the failure of one of the units), while from I to O the rate is μ.

Fig. 3b depicts how the problem can be modeled with SPNs. Each of the two units can be in three possible states: up, idle (standby), or down. There are three places corresponding to each of these states, and two tokens denoting each unit. The presence of a token in a given place indicates that a unit is in the corresponding state. Initially both tokens are in the 'up' place, indicating that both units are operating; a failure transition is therefore enabled. This transition is multiple, i.e. there is an independent clock associated with each enabling token. It is worth noting that compared to the Markov chain representation the requirement for the two units to be identical is relaxed here (different distributions can be assigned to the firing time for each of the colors). At the same time, the transition from up to 'standby' is disabled despite the presence of tokens in the transition's input place due to the inhibitor of multiplicity two. When a unit fails, which implies that one of the two clocks in 'failure' transition runs to zero, then the corresponding token is fired and deposited into the 'down' place. This event has two consequences: the 'repair' transition is enabled and the corresponding clock initiated (so the repair of the unit can occur); in addition, the inhibitor becomes disabled (since there is only one token in its input place) so that the other token (corresponding to the remaining 'good' unit) is immediately fired to the standby place. It stays there as long as the failed unit is in the down place (an inhibitor ensures that), but once the failed unit is repaired, the token corresponding to the good unit returns to the up place.

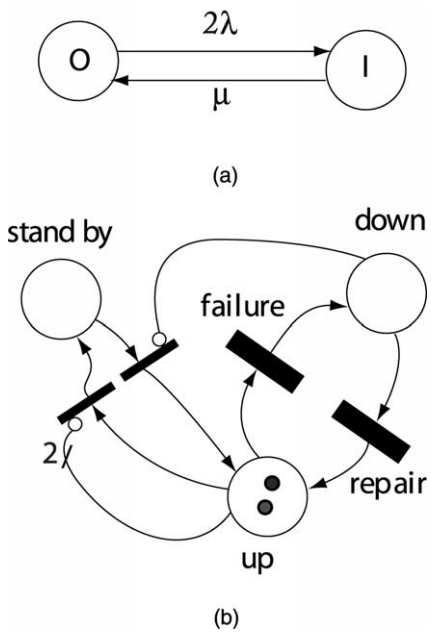


Fig. 3. Two units connected in a series: (a) modeled using Markov chains; (b) modeled using SPN.

This simple example is somewhat unrepresentative in the sense that the modeling using Markov states looks more compact (and on the surface there seems to be no particular reason to use SPNs instead). However, a distinguishing feature of the SPN model is that it can handle not only two but also an arbitrary number of units (tokens need to be added and the multiplicity of the first inhibitor appropriately adjusted, but otherwise the model will look exactly the same). Obviously, the Markov model, especially in the case of units with different failure rates, will grow significantly in size. The underlying reasons for such a difference are quite fundamental and are covered elsewhere [24]. Here it suffices to say that the Markov model is of a global kind: any local change in the system corresponds to a different (global) Markov state. On the other hand, SPNs provide local modeling: only the combined marking of all the places characterizes the system. A direct consequence of this difference is that Markov chains use a global (explicit) logical branching where all the logical permutations must be specified upfront, while SPNs use local (implicit) logical rules, which results in more efficient models when the problem sizes increases. In the case of a system with only two parts there might be fewer global states than local ones, but for larger systems the situation is practically always reversed.

Let us now attempt to relax the assumption of constant failure rates (however convenient this assumption may be, its consequences are quite detrimental to the model's fidelity [29,30]). Obviously, Markov chains cannot be used directly here. However, there are existing extensions of SPNs that provide a mechanism to model such situations. More specifically, if one uses *prs* policies for failure transitions, then when the failed part is repaired and the unfailed part is activated, the failure transition will remember its age; in other words, a transition has an age memory. Let us show how the same functionality can be achieved when the memory is assigned to a token instead of a transition. A single aging label is introduced with the token's age defined as the current value of the corresponding CDF. This value can be used to calculate the 'equivalent time,' i.e. the time with the same value of CDF [31,32] corresponding to a transition in a new place. Since in this particular example the token returns to the same place, the equivalent time is equal to the time that the token was enabled. However, as shown in the following examples, that is not always the case. When the token returns to the up

state, the firing time is determined by taking the age of the part into consideration. This example demonstrates the use of aging tokens, but does not present clear advantages over assigning memory properties to transitions. Let us proceed with examples where these advantages are more transparent.

4.2. Example B: a repairable system with warm spares

Let us consider a repairable system with two active units and a 'warm spare' (the term describes a situation where a standby unit can fail but the corresponding distribution is different from that of an active component). If the transition rates are constant, such a system is often modeled using Markov chains [33]; this requires 18 states and 36 arcs for a general case when all three components are different (for indentical components four states are sufficient to model the system). The discussion from the previous example about 'global' versus 'local' modeling is quite relevant here. Using SPNs, the model has a much simpler form (Fig. 4) and is easier to read. As in Ref. [33], the system is considered to be down if fewer than two units are operating, but the remaining unfailed unit can fail nevertheless (alternative assumptions can be implemented using the procedure outlined in the previous example). Similar to the previous example, there are three possible states for each unit (each state is represented by a corresponding place). Initially there are two tokens in the up place, which disables an immediate transition for a third token from the 'spare' to the up place by means of an inhibitor of multiplicity two. A spare or an active component can fail, which is modeled by two appropriate timed multiple transitions to the down place. Once any of the two active components fails, the 'activation' immediate transition becomes enabled and a token corresponding to a spare moves to the up place (assuming that a spare is available). It is interesting to note that, depending on the choice of the type of the repair transition, different repair policies can be modeled: a series type will be equivalent to a single (first-in-first-out) repair team, while a multiple transition would model unlimited repair recourses. What is also remarkable is that the model actually remains correct for a general k-out-of-n case, as long as the tokens are added and the arc multiplicity is adjusted. While in many respects this model is very similar to Example A, there is one fundamental difference in modeling the systems with varying hazard (failure) rates: damage now can accumulate in two different transitions (spare and active failures), and therefore the modeling with *prs* transitions is far from straightforward, if not impossible. One can envision merging the up and spare places and instead modeling the transitions from the spare to up state for each component by implementing an elaborate marking dependence; however, such a solution will lead to the loss of visual clarity that the original non-aging model possesses.

The introduction of aging tokens, on the other hand, permits comfortable modeling of the aging of system without any marking dependence. In fact, the visual representation of the model will stay the same (Fig. 4). As in Example A, each token is assigned a label that records the component's age, A_γ. Let us assume that at $t = t_0$ the component is new $A_\gamma(t_0) = 0$, while F_a and F_s are CDFs corresponding to the firing policies of active and spare failures, respectively. Let us further consider a scenario when one of the active components fails first at t_1. Then a token corresponding to a spare component moves to the up place, and its age $A_3 = F_s(t_1)$ is used to calculate an equivalent starting time [31,32], $t_s = F_a^{-1}(A_3)$, which then is used to sample F_a.

4.3. Example C: load sharing

Let us consider a subsystem where two parts with aging failure distribution (e.g. Weibull) share a common load. Fig. 5 provides an illustration of how such a system can be modeled within the proposed framework. Initially, the two parts are in an undamaged state, so the two corresponding labeled tokens are in place A. Both of these tokens enable the first transition that has matching (possibly different) aging policies for each of these tokens. A (second) immediate transition from place A to C is disabled due to the presence of an inhibitor of multiplicity two. Once one of the parts fails, a corresponding token moves to place B; this enables the second transition, which fires immediately and moves the other token corresponding to the still-operating part into place C. This in turn enables a third transition that has a matching (faster) aging policy, corresponding to this

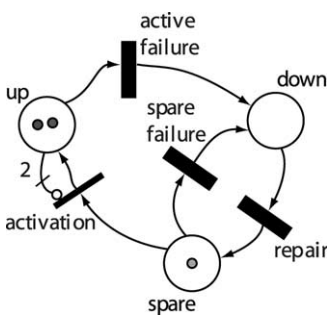


Fig. 4. SPN model for a two-out-of-three repairable system with a warm spare.

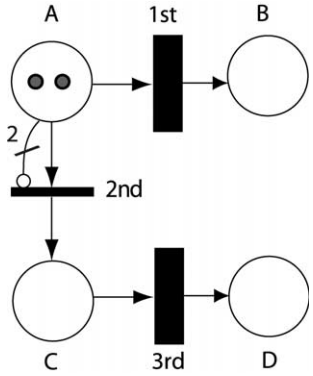


Fig. 5. Modeling of a shared load for a two-component system.

part carrying the load alone. The amount of damage accumulated when the token was staying in place *A* can be used to determine the firing time based on the distribution for the third transition. Such models need not be restricted to time scaling, as is commonly done in modeling multiphase changes in loads (when the effects of changing loads are modeled by changing the clock speed), and more rigorous modeling based on the equivalent time (see the previous example) is implemented instead.

An SPN without aging tokens would consist only of the top part of Fig. 5, and the important features of the system would not be modeled graphically. Instead, the definition of the firing policy for the first transition would include a statement to the effect that if a token is marked in State *B* (one of the two parts has failed) then the firing policy changes (which would further complicate this situation since for aging parts such a change would also depend on *when* the second part fails). Similar modeling problems exist in multiphase systems (see Example D); in fact, the difficulties associated with such modeling probably explain why there are no published results (to the best of the author's knowledge) for Petri-net modeling of shared load problems or multiphase missions where different phases require different general (non-exponential) firing policies for the same unit.

4.4. Example D: multiphase missions

Next, let us consider a simple example of an aging part undergoing a two-phase periodic mission (Fig. 6): transitions e_1 and e_2 determine the duration of the first (place *A*) and second phases (place *B*), respectively (such durations can be either deterministic or non-deterministic). During each phase the part can fail (which corresponds to the firing of the transition f_1 or f_2, respectively). In this case, transitions f_1 and f_2 have aging policies for the token, but not e_1 and e_2.

The modeling difference of the proposed framework with existing SPN techniques is discussed in more detail in Ref. [34], where a relatively more elaborate example of phased mission systems [35] is modeled using aging without resorting to marking dependence.

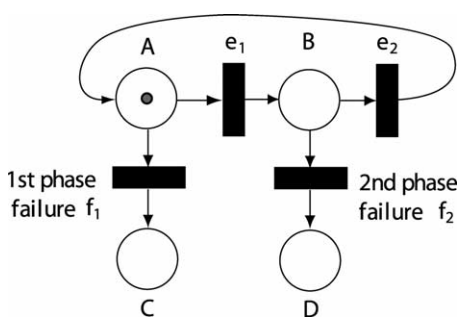


Fig. 6. Modeling of accumulated damage in a periodic mission consisting of two different phases.

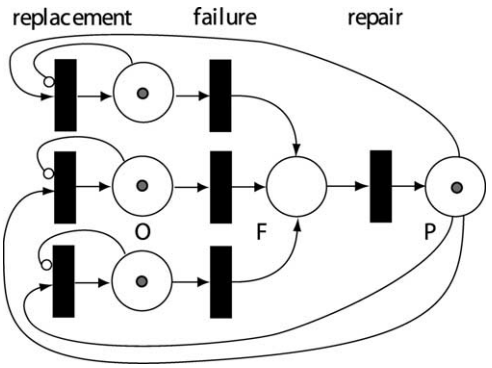


Fig. 7. Repairing with shared pool of spares.

4.5. Example E: pooled repair

Let us consider a fleet of three similar units that share a pool of spares (Fig. 7): when a part fails at any of the units, a spare from the pool is used (if available); in the meantime, the failed part is sent for repairs. When repaired, the failed part is added to the pool of spares. If the repair restores the part to its pristine, 'like-new' state, there would be no problems modeling such a situation in standard SPNs; however, very often such an assumption is unrealistically optimistic. Let us instead consider so-called 'minimal repair' (quite commonly used in reliability): the part is repaired to the state where it was just before failure.

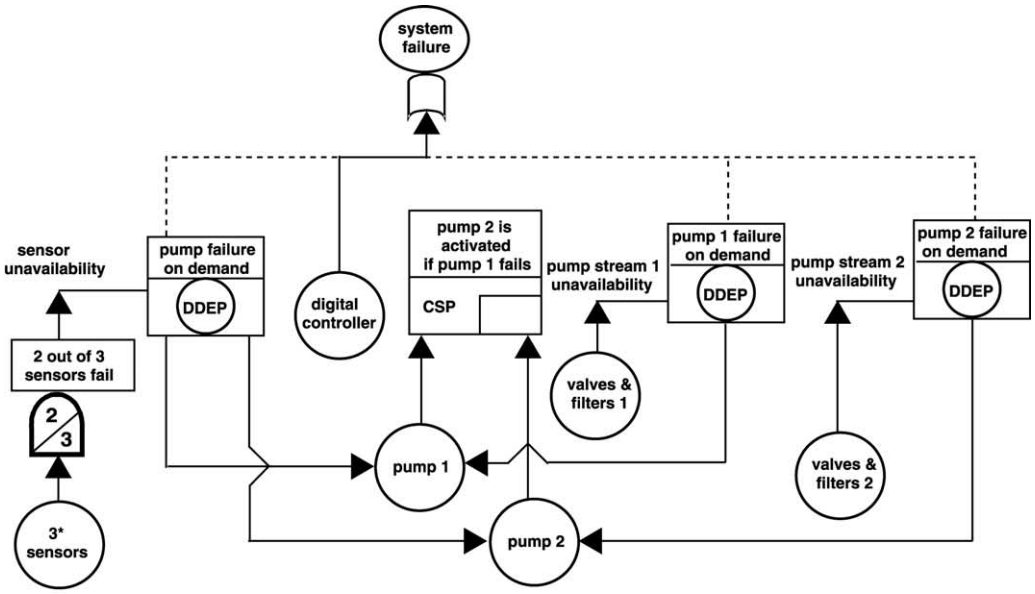
In this model, only failure transitions have aging policies for the tokens associated with them. The counter for a given token records the age of a part, which is used to sample the corresponding failure distribution. More sophisticated policies of repair can be modeled with the proposed technique as well (for example, the repair counter for a token can be activated, and only a limited number of repairs will be allowed before the part is discarded).

Even such a relatively simple situation presents formidable challenges for modeling using high-level Petri nets with memory-enabled transitions (once the part fails, the corresponding transition fires; even if such a transition preserves the age, keeping the track of all possible combinations is not practical because a different part can replace the failed unit). Similar challenges exist in standard commercial software packages that use RBDs.

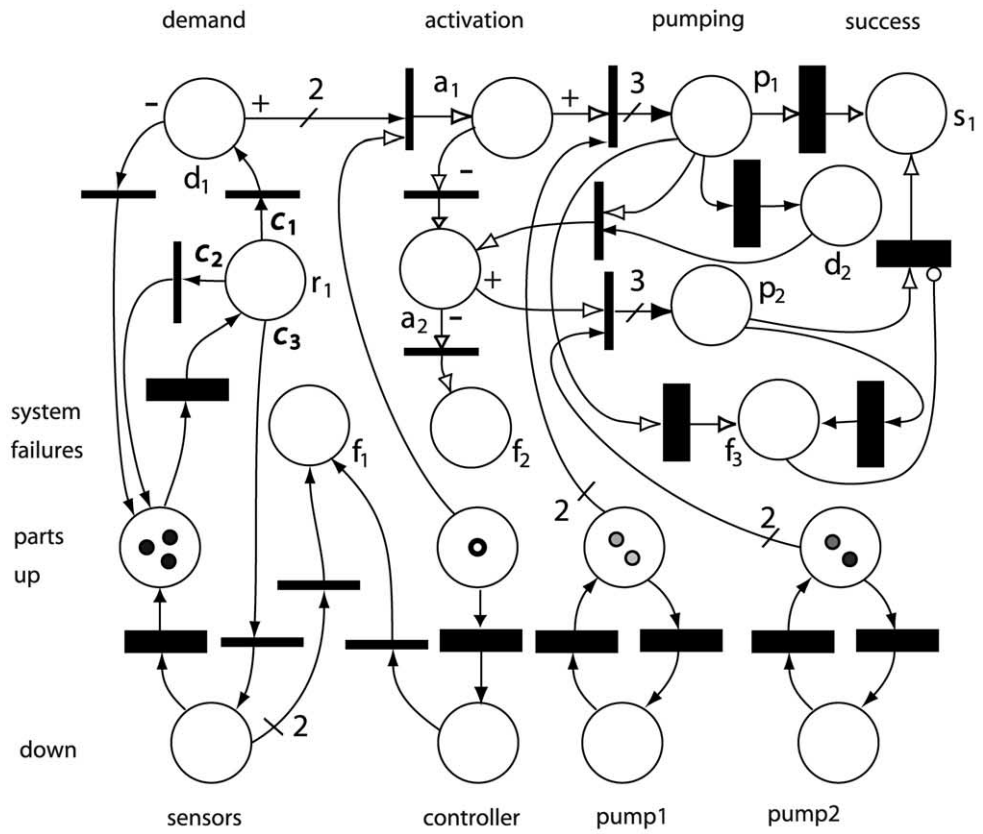
4.6. Example F: system on demand

To further explore the flexibility of SPNs with aging tokens, let us turn to the first example from Ref. [6] where a new construction (demand dependency gate, DDEP) is introduced for dynamic fault trees. Therein, as a motivating example for the introduction of this new construction, a computer-controlled hypothetical sprinkler system (HSS) is considered. The system is comprised of three sensors, two pumps, and a digital controller. Each pump has a support stream (valves and filters) that must be operational for the pump to start. The sensors send temperature readings to

the digital controller, and when a threshold is exceeded, the controller activates the pump. If the first pump is not able to start or fails during the service, the backup (second) pump is activated. HSS is available on demand if at least two of the sensors are operational and at least one pump and the controller are functional. Once the pump is operating, only the digital controller and pump system are required. The key concept in DDEP is separating the analysis into two phases:



(a)



(b)

Fig. 8. Models for a hypothetical sprinkler system: (a) reliability part modeled in dynamic fault trees; (b) combined SPN model for reliability and availability.

standby and demand. A steady-state availability of the standby components is used as an input to DDEP and incorporated into the reliability assessment of the demand phase. Fig. 8a depicts the resulting dynamic fault tree for the reliability part HSS system. In addition to DDEP, cold-spare gates [5] are employed. Neither the model's availability of the support systems nor the demand are discussed. For a detailed description of this model, the reader is referred to Ref. [6].

Fig. 8b depicts an SPN model for the same system. In addition to the features modeled in Fig. 8a, simple availability models for each component as well as a simulation of the demand are provided. The lower part of the graph refers to the standby mode. All transitions in the model are multiple. The tokens that denote the three sensors are originally in the up place. After a certain time (t_1) these sensors are queried for temperature readings (tokens are fired to r_1), and three options exist for each sensor: the reading is above the threshold (probability c_1), the reading is normal (probability c_2), or there is no correct reading (the sensor is faulty, probability c_3). (It is assumed that faulty sensors are immediately detected: if needed, this model could be further refined to reflect the possibility of undetected sensor failures.) The tokens are moved in accordance with these options to the corresponding places, and if at least two sensors are down, then f_1 receives a token. Similarly, if the controller fails, f_1 receives a token as well. The place f_1 corresponds to the 'failure to detect the demand' mode. Otherwise, sensors are read with t_0 time interval, until place d_1 receives at least two tokens, when a controller token (CT) is moved into place a_1. This is a colored token, but here, in order to be distinguished in black and white as well, a hollow token is used, and the same is true for the corresponding arc arrowheads that help to visualize the token's movements. Here a_1 is an input place for two immediate transitions. The one with the higher priority (the corresponding arc is denoted with a $+$) will fire first if both of the colored tokens corresponding to the first pump itself and its supporting system are available. Then all three colored tokens are moved into p_1, indicating that the first pump has started to operate. Otherwise, the CT moves to a_2, and similar construction allows initiation of the second pump (p_2).

If the second pump has failed to start, the CT is moved to f_2, indicating that the second failure mode occurred: failure to respond to the demand. Once the first pump starts to operate, there are two options: either the service will be successfully completed (and the CT moves to s_1)—which occurs after a specified time t_s—the corresponding transition's deterministic delay is t_s, or a failure occurs. An uncovered failure of the controller is modeled by firing the CT to f_3 (which is the third mode of failure that occurs during the operation). On the other hand, if a stream or a pump itself fails during the operation, and one of the corresponding tokens moves to d_2, then the CT is moved to a_2 and the spare is activated. Such spare modeling is possible due to the fact that the CT is an aging token and therefore has a label that records the time the CT has spent at p_1, while the transition from p_2 has a matching firing policy, so that the timed delay is $t_s - t_1$. Furthermore, if the failures of each component are non-exponentially distributed, the token's labels allow recording of the age of the component, so that the model remains valid. For example, the token corresponding to the first pump can remember the age it accumulated during the standby phase, which will affect the sampling of its failure time during the operation.

4.7. Example G: damage tolerance

A simple damage tolerance model for a single component is provided below (Fig. 9). A token corresponding to a component is initially marked in the place denoted as B, indicating that the component is operating. The age of this token reflects the maximum flaw size. The counter is initialized with a value characteristic of a new component (corresponding to age zero). The token enables two transitions: a failure transition and the 'Time Between Inspections (TBI)' transition. The value of the counter changes (reflecting the fact that the flaw size increases) as the clocks associated with both transitions race to zero. If the clock associated with the failure 'wins' this race, the token is fired from B to C, and the 'game' is over. Obviously, at this point the value of the token's counter corresponds to the critical flaw size, and the component fails. On the other hand, if the TBI clock reaches zero first, the token is fired to A, and its age translates into the characteristic flaw size that is less than the critical flaw size. Next, an inspection takes place (in the example this inspection is denoted as instantaneous, but delays associated with inspections can be modeled as well). With a certain probability, p (which can depend on the value of the counter), the part passes the inspection and the token is fired to B. As a result, a new age is calculated in accordance with the efficiency of the inspection: a new value is sampled from a distribution corresponding to the maximum undetected flaw size. The token's age is used to determine the new time associated with the failure transitions, and the process is repeated. If the part does not pass inspection, the token is fired from A to D (instead

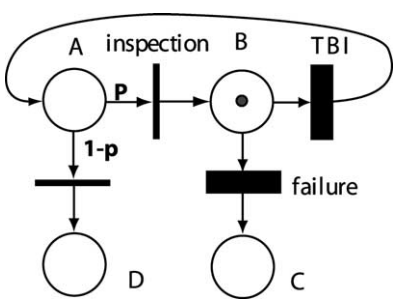


Fig. 9. A simple SPN model for a damage-tolerant component.

of C), which indicates the need for repair or retirement of the part.

5. Implementation example

The emphasis of the paper is on providing a graphical means of modeling system reliability. While striving for implementation independence, the importance of demonstrating the utility of this framework is recognized as well. Therefore, the description of a relatively straightforward implementation of the proposed framework is provided with the understanding that the applications of aging tokens are by no means restricted to this implementation example. In accordance with the classification provided in Ref. [36], the example corresponds to a direct, component-based discrete event simulation. Relevant dependability measures for the system (such as reliability or availability) are calculated by tracking the histories of the appropriate tokens' movements. In what follows, SPNs with n transitions, m places, and only multiple timed transitions are considered (as shown above, both standard and series transitions can be expressed in terms of multiple transitions). Arcs, tokens, and transitions can each have sets of properties that are defined as follows.

5.1. Arc attributes

For each transition α, $(\alpha = 1, \ldots, n)$ a_α, b_α, and s_α represent sets of attributes for input, output, and inhibitor arcs, respectively. The attributes include: the place that terminates the arc, multiplicity, and color preference. The latter property merits a more detailed description. A subset of colors that are compatible with this arc is specified therein (by default this subset coincides with the full set of colors). If a transition changes a token color, a new color is used to determine whether the token is compatible with an output arc. If tokens and arcs are incompatible they do not affect each other's behavior. For example, if a token arrives at a transition and none of the output places is compatible with the color of this token, the token's death occurs (see, for example, Fig. 2, transition from place d_1 to a_1). This property allows the filtering of tokens and their redirection in accordance with their colors (see, for example, T_1 in Fig. 2 and the transitions from p_1 to a_2 and d_2). In addition, a set of attributes that is peculiar to output arcs is specified: an option is provided for the birth of new tokens. If the corresponding flag is turned on, then a new token's color and age are specified. The age can be either deterministic or sampled from a certain distribution (as in modeling *pri* policy or in example G). It is useful to extend the set of utilized distributions to include such 'derived' distributions as provided by Eq. (1).

5.2. Token attributes

At the start of a simulation $t = t_0$, tokens are numerated and provided with the corresponding index, $\gamma = 1, \ldots, k_0$. For each γ, the initial marking M_γ^0 indicates the place where the token (denoted as Y_γ) is located; here k_0 corresponds to the initial number of tokens. As the simulation progresses, the number of tokens present in the model might change. In the instance of the death of Y_γ the corresponding marking is set to zero: $M_\gamma(t) = 0$. Tokens newly born during simulation are assigned consecutive indices: $\gamma = k_0 + 1, \ldots$. The token's colors and age are denoted as $C_\gamma \in \{1, \ldots, q\} \equiv C$ and $A_\gamma \in [0, 1[$, respectively (in the present implementation only one label of each kind is permitted).

5.3. Transition attributes

For each transition α and each subset of colors $c \subseteq C$, the following attributes $Z_{\alpha cj}$ $(j = 1, \ldots, 7)$ are provided:

- *$j = 1$: transition delay type*. Value 0 corresponds to an immediate transition, 1: timed, 2: exponential, 3: Weibull, 4: lognormal, 5: normal, etc.;
- *$j = 2,3$: parameters of the transition*. In the case of an immediate transition, the first parameter specifies the type of the conflict resolution (0: priority, 1: probabilistic weight), while the second parameter provides either the probability or the numerical priority. If priorities are specified for several simultaneous transitions that are enabled by the same token, only the transition with the highest priority is fired; otherwise the firing transition is randomly chosen with probabilities proportional to the weight specified.
- *$j = 4$: policy with respect to the color label*. Values take integers (positive or negative) that are added to the color label. Default zero value keeps the color unchanged.
- *$j = 5$–7 describe interactions between the transition and the token's age*. $j = 5$ Indicates whether the age is affected (1) or not (0). An internal check is implemented to ensure that only one transition affects a given token in a given place. The sixth value conversely indicates whether the firing policy is affected by age (1) or not (0). The seventh value specifies whether upon the firing of the transition the age is reset to any value $\tilde{A}_\gamma \in [0, 1[$; the value 1, on the other hand, indicates that age has not been reset.

A component-based discrete event simulation [36] as applied to SPNs implies that each enabled transition-token pair $\{Z_{\alpha_j}, Y_{\gamma_j}\}$, $j = 1, l$ can potentially change the system state. However, only the event that corresponds to the earliest firing cannot be affected by other events. Therefore, the simulation consists of the following two phases.

5.4. Initialization

- $A_\gamma(t_0)$ are specified; at $t = t_0$ all enabled pairs of transition and tokens $\{Z_{\alpha_j}, Y_{\gamma_j}\}$, $j = 1, l$ are identified

and assembled into a list of following tuples:

$$W_j = \langle Z_{\alpha_j}, Y_{\gamma_j}, t_e, t_s \rangle$$

Here t_e corresponds to the time when the pair was enabled, and $t_s = F^{-1}(A_\gamma)$ is the equivalent starting time for this transition. Only probabilistic conflict resolutions and choices among multiple tokens for immediate transitions are run-dependent; all other tuples can be determined prior to the individual runs, so a simple loop over all transitions is not computationally expensive.

- For each token Y_j with stochastic aging, a random number $A_j^m \in]0,1[$ is sampled that indicates the maximum age of the token, i.e. when token is fired).
- Firing delays τ_j for each W_j are calculated, and W is re-ordered in accordance with τ_j, so that $\tau_1 = \min_j \{\tau_j\}$. For all aging tokens and matching transitions (e.g. if $Z_{\alpha_j c6} = 1$):

$$\tau_j = F_{\alpha_j}^{-1}(A_{\gamma_j}^m) - F_{\alpha_j}^{-1}(A_{\gamma_j}) \tag{2}$$

where F is the corresponding CDF. All other random firing delays are sampled as necessary.

5.5. Transport

First, the conflict resolutions (if any) are resolved for W_1 : the presence of a conflict implies that $t = \tau_1$ and there is at least one more W_j, such that $\tau_j = t$ and either $Z_{\alpha_1} = Z_{\alpha_j}$ or $Y_{\gamma_1} = Y_{\gamma_j}$. In the former case, several tokens simultaneously enable the same immediate transition, and the token to be fired first is chosen at random; the corresponding tuple is placed first within W. In the latter case the same token enables several transitions; then all such W_j are used to resolve the conflict either probabilistically or deterministically in accordance with the transition properties. As a result a single tuple, which after a resorting becomes W_1, is chosen for firing. All other tuples are removed from W. Time is updated $t = \tau_1$ and then W_1 is fired. All the changes in the system take place as follows:

- W_1 is removed from the list of enabled pairs
- Marking $M_\gamma(t)$ is updated, and the age A_{γ_1} is either reset (if $Z_{\alpha_1 c7} \neq 1$) or updated (if $Z_{\alpha_1 c5} = 1$) using formula $A_{\gamma_1} = F(t_s + t - t_e)$. If the age is reset, a new maximum age A_1^m is sampled.

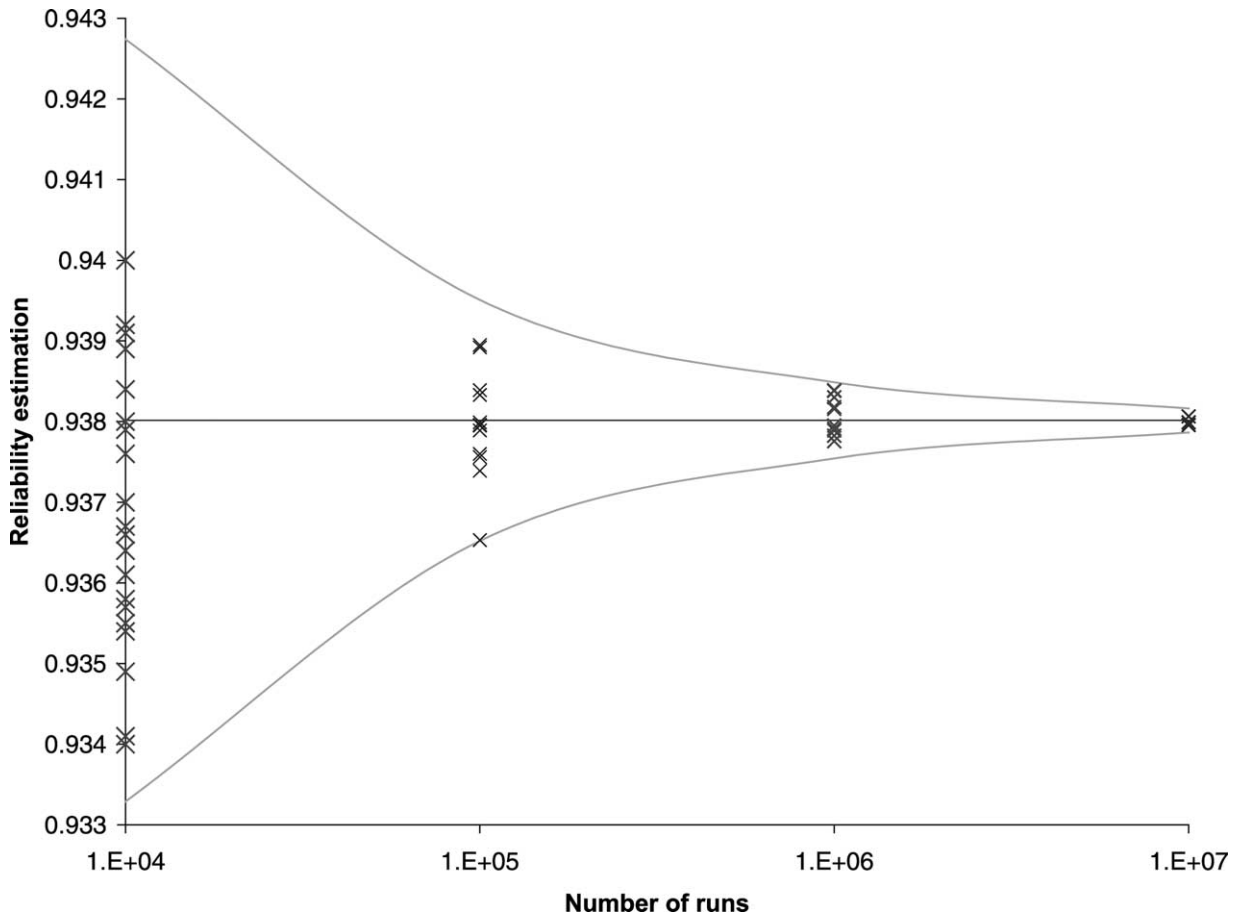


Fig. 10. Reliability estimation for the shared load example using SPNs with aging tokens (discrete event simulation).

- If the firing of transition Z_{α_1} caused the birth of a token Y_{k_0+r} to occur, then a new token age is assigned or sampled.
- All other transitions that were enabled by token Y_{γ_1} in the old place are disabled; the age is updated if appropriate. Similar actions are performed if either Y_{k_0+r} or Y_1 in their respective new places (if any) inhibit some previously enabled transition.
- Conversely, with respect to the previous step, a test is conducted to check whether Y_{γ_1} or $Y_{\gamma_{k_0+r}}$ have enabled any previously disabled transitions; if so, $t_e = T$ are recorded along with t_s, new firing times are calculated based on Eq. (2), and new entries are inserted into W, observing the ordering of W with respect to firing times.

5.6. Verification of simulation results

Example C provides an obvious choice for verification due to the availability of analytical formulae that can be numerically evaluated with high precision [31]. As an example, let us consider two identical components, so that when both components are operating, the failure obeys a two-parametric Weibull distribution with the shape parameter $\kappa = 1.4$, and the scale parameter $\theta = 150$. Upon failure of a component, the failure parameters of the still-operating component change to $\kappa' = 1.7$, and $\theta' = 90$. A numerical integration for time $t_{\max} = 50$ yields reliability $R = 0.938015$. Fig. 10 demonstrates how the simulation results $\hat{R}$ converge to this 'exact' solution. The standard deviation for the reliability prediction can be estimated as

$$\hat{d} = \sqrt{(\hat{R}(1-\hat{R}))/(n-1)},$$

where n is the number of histories. In Fig. 10 the bounds $[R - 1.95997\hat{d}, R - 1.95997\hat{d}]$ are shown corresponding to 95% confidence interval, assuming the applicability of the central limit theorem.

Next, let us consider a case with warm spares (Example B). Identical components with Weibull distributions with parameters $\kappa_s = 1.4$, and $\theta_s = 150$ for spares and $\kappa_a = 1.7$, and $\theta_s = 90$ for active components are studied. If a system with no repairs is considered, an analytical solution again can be obtained, and the correlation with the simulation results are as good as in the case of the shared load. Similarly, when the distributions for both failure and repair firing times are replaced with exponential firing times, Markov chains were used to validate the procedure. If a parallel repair that obeys a lognormal distribution with parameters $\mu =$ and $\sigma =$ is introduced, then reliability for $t_{\max} = 100$ is estimated as $\hat{R} = 0.712795$ for 1mln histories, with the corresponding estimate for the standard deviation as $\hat{d} = 0.000453$.

6. Conclusions

In previously existing PN formulations, memory was associated solely with transitions, which resulted in certain difficulties in modeling the changes in the system configuration while preserving the memory. In such a setting, the changes in the damage accumulation (such as in multiphase missions or due to load-sharing) must be accommodated by the so-called marking-dependent firing policies. This implies that a firing policy can change if the marking changed elsewhere in the model. However, there is no accepted way to express such dependence in a simple graphical way, and it must be described using an external logic; as a result, the clarity of the modeling is compromised. It is important to note that such difficulties are fundamental: if an event occurs that changes the properties of a given component, then in accordance with the spirit of Petri-net modeling, such an event would best be described by a corresponding token moving from one place to another. However, such a move would result in memory loss, since memory is associated with transitions, and once the token moves to a new place there is no mechanism to associate any aging with the component that it represents.

In contrast to such existing means of modeling aging in SPNs, the concept of aging tokens is introduced. It is demonstrated that aging tokens significantly improve the dependability modeling flexibility and clarity of SPNs when aging systems are considered. Aging tokens can be viewed as a natural extension of colored Petri nets since they are effectively token labels that are allowed to change not only discretely upon the firing of the token, but continuously in the process of enabling a certain transition that has a matching policy. By extending the properties of token labels and assigning memory to the tokens, not only can different firing policies be implemented, but the resulting models minimize the use of marking dependence and provide models that more closely resemble standard SPN models without aging. In addition, the hierarchical procedures for constructing Petri nets (such as described in Ref. [23]) are fully applicable for the proposed technique. The advantages of new constructions are demonstrated in several examples, including load-sharing, shared pools of identical imperfectly repaired components, multiphase missions, and damage-tolerant components.

References

[1] Movaghar A, Meyer JF. Performability modeling with stochastic activity networks. Proceedings of Real-time Systems Symposium, December 4–6, 1984, Hyatt Regency Hotel, Austin, TX, IEEE Computer Society Press; 1984.
[2] Adamyan A, He D. Analysis of sequential failures for assessment of reliability and safety of manufacturing systems. Reliab Engng Syst Safety 2002;76:227–36.
[3] Schneeweiss W. Tutorial: Petri nets as a graphical description medium for many reliability scenarios. IEEE Transactions on Reliability 2001;50:159–64.
[4] Liu T, Chiou S. Application of Petri nets to failure analysis. Reliab Engng Syst Safety 1997;57:129–42.

[5] Dugan JB, Bavuso S, Boyd M. Dynamic fault tree models for fault tolerant computer systems. IEEE Trans Reliab 41.
[6] Meshkat L, Dugan JB, Andrews J. Dependability analysis with on-demand and active failure modes, using dynamic fault trees. IEEE Trans Reliab 2002;51:240–51.
[7] Čepin M, Mavko B. A dynamic fault tree. Reliab Engng Syst Safety 2002;75:83–91.
[8] Ramesh AV, Twigg DW, Sandadi UR, Sharma TC. Reliability analysis of systems with operation-time management. IEEE Trans Reliab 2002;51(1):39–48.
[9] Rouvroye JL, Brombacher A. New quantitative safety standards: different techniques, different results? Reliab Engng Syst Safety 1999; 66:121–5.
[10] Rouvroye JL, Van Den Bliek EG. Comparing safety analysis techniques. Reliab Engng Syst Safety 2002;75:289–94.
[11] Peterson JL. Petri net theory and the modeling of systems. Englewoods Cliffs, NJ: Prentice-Hall; 1981.
[12] Haas PJ. Stochastic Petri nets. Modelling, stability, simulation. New York: Springer; 2002.
[13] Jensen K, Coloured Petri nets: basic concepts, analysis methods and practical use, vol. 3. Berlin: Springer; 1997.
[14] Zhou M, Venkatesh K. Modeling, simulation, and control of flexible manufacturing systems: a Petri net approach. Singapore: World Scientific; 1999.
[15] Schneeweiss WG. Better graphs for dependability modeling. Microelectron Reliab 1998;38(5):807–20.
[16] David R, Alla H. Petri nets for modeling of dynamic systems: a Survey. Automatica 1994;30(2):175–202.
[17] Signoret JP, Gabariaud G, Leroy A. Modélisation et évaluation probabiliste de la production d'hydrocarbures d'un ensemble depuits sousmarins. 5e Colloque International de Fiabilité et de Maintenabilité, Biarritz, France; 1986.
[18] Dutuit Y, Châtelet E, Signoret J-P, Thomas P. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases. Reliab Engng Syst Safety 1997;55:117–24.
[19] Nicola VF, Shahabuddin P, Nakayama MK. Techniques for fast simulation of models of highly dependable systems. IEEE Trans Reliab 2001;50(3):246–64.
[20] Marseguerra M, Zio E, Devooght J, Labeau PE. A concept paper on dynamic reliability via Monte Carlo simulation. Math Comput Simul 1998;47:371–82.
[21] German R. Performance analysis of communication systems: modeling with non-Markovian stochastic Petri nets. New York: Wiley; 2000.
[22] Petri A. Kommunikation mit automaten. PhD Thesis. Institut für Instrumentelle Mathematik, Schriften des IIM; 1962.
[23] Jensen K, Rozenberg G, editors. High-level Petri nets: theory and application. Berlin: Springer; 1991.
[24] Volovoi VV. Frameworks for system-dependability modeling. Submitted for publication.
[25] Bobbio A, Puliafito A, Telek M. A modeling framework to implement preemption policies in non-Markovian SPNs. IEEE Trans Software Engng 2000;26(1):36–53.
[26] Heller S, Greiner S, Horton G. A Petri net simulator for fast safety and quality analysis and cost prediction. Proceedings of the 16th European Simulation Multiconference, Darmstadt, Germany; 2002.
[27] Horváath A, Telek M. Time domain analysis of non-Markovian stochastic Petri nets with PRI transitions. IEEE Trans Software Engng 2002;28(10):933–43.
[28] Elerath J. Implied service strategies in availability assessments. Proceedings of Annual Reliability and Maintainability Symposium, IEEE; 2003. p. 96–100.
[29] Bowles J. Commentary-caution: constant failure-rate models may be hazardous to your design. IEEE Trans Reliab 2002;51(3): 375–7.
[30] Jones J, Hayes J. Estimation of system reliability using a non-constant failure rate model. IEEE Trans Reliab 2001;50(3):286–8.
[31] Kececioglu D, Reliability engineering handbook, vol. 2. Englewood Cliffs, NJ: Prentice-Hall; 1991.
[32] Dubi A. Monte Carlo applications in systems engineering. Chichester: Wiley; 2000.
[33] Henley EJ, Kumamoto H. Probabilistic risk assessment: reliability engineering, design, and analysis. New York: IEEE Press; 1992.
[34] Volovoi V. Modeling multiphased missions using stochastic petri nets with aging tokens. Proceedings of Annual Reliability and Maintainability Symposium, IEEE; 2004.
[35] Bondavalli A, Mura I. High-level Petri net modelling of phased mission systems. Proceedings of the 10th European Workshop on Dependable Computing; 1999.
[36] Labeau PE, Zio E. Procedures of Monte Carlo transport simulation for applications in system engineering. Reliab Engng Syst Safety 2002; 77:217–28.