Towards the quantitative evaluation of phased maintenance procedures using non-Markovian regenerative analysis

Laura Carnevali[1], Marco Paolieri[1], Kumiko Tadano[2], and Enrico Vicario[1]

[1] Dipartimento di Ingegneria dell'Informazione, Università di Firenze, Italy
{laura.carnevali, marco.paolieri, enrico.vicario}@unifi.it
[2] Service Platforms Research Laboratories - NEC Corporation, Kawasaki, Japan
k-tadano@bq.jp.nec.com

Abstract. The concept of Phased Mission Systems (PMS) can be used to describe maintenance procedures made of sequential actions that use a set of resources and may severely affect them, for instance operations that require outage of hardware and/or software components to recover from a failure or to perform upgrades, tests, and configuration changes. We propose an approach for modeling and evaluation of this class of maintenance procedures, notably addressing the case of actions with non-exponential and firmly bounded duration. This yields stochastic models that underlie a Markov Regenerative Process (MRP) with multiple concurrent timed events having a general (GEN) distribution over a bounded support, which can be effectively analyzed through the method of stochastic state classes. The approach allows evaluation of transient availability measures, which can be exploited to support the selection of a rejuvenation plan of system resources and the choice among different feasible orderings of actions. The experiments were performed through a new release of the Oris tool based on the Sirio framework.

Keywords: Phased mission systems, maintenance-induced failures, transient availability measures, Markov regenerative processes, stochastic state classes.

1 Introduction

Phased Mission Systems (PMS) perform multiple tasks with possibly different requirements during non-overlapping phases of operation, typically achieving the mission success only if each phase is completed without failure [8]. The abstraction of PMS is fit by several critical applications, which notably include: aircrafts flights comprising distinct steps from take-off to landing, command sequences executed by aerospace systems, recovery operations performed to bring back automated systems from the breakdown state to a recovery point, and maintenance procedures requiring the safe shutdown and restart of hardware (HW) and software (SW) subsystems [27, 14, 30].

Reliability analysis of PMS faces challenges concerned with the evaluation of the success probability and the distribution of the completion time, for the

overall mission or for intermediate phases. This often conflicts with the possible failure and recovery of resources supporting the different steps. Steady state analysis is usually applied to evaluate the system availability in the presence of recurrent phased procedures, while transient analysis is more appropriate in the case of one-shot operations where the focus is rather on the probability that the procedure is completed within a given deadline.

Various modeling approaches have been proposed in the literature, and in particular different causes of failure and policies for error detection, rejuvenation, and repair have been addressed. Methods based on state-space analysis address PMS with complex behaviors deriving from fixed or random phase sequence, deterministic or random phase duration, permanent or transient components failures, and dependencies among components. Notable examples include: the approach of [15], which leverages Markovian analysis to support the reliability evaluation of PMS with deterministic sojourn time in each phase; the method proposed in [18], where the Markov Renewal Theory [13, 1, 2] is tailored to the evaluation of PMS with random phase sequence and duration, forcing a repetition or a premature completion of unfinished repair works at each phase end to guarantee that phase completion times are regeneration points for the underlying stochastic process; the methodology of [6] for performance evaluation of composed web services, which derives steady state and transient measures through WebSPN [5] by relying on the approximation of GEN timers with discrete phase type distributions over unbounded supports.

If the events order does not affect the mission outcome, the problem of state-space explosion can be circumvented by applying combinatorial solution techniques, which achieve a lower computational complexity at the expense of reducing the modeling expressivity. In this area, several approaches have been developed on the structure of Binary Decision Diagrams (BDD) [31, 28], also encompassing phase uncovered failures that cause a mission failure [29]. More recently, approaches that integrate methods based on state-space analysis and combinatorial techniques have been proposed. The joint probability method of [19] combines results obtained by independently solving static and dynamic system components through BDD and Markovian analysis, respectively. In [26], a hierarchical approach is presented which addresses PMS with repairable components, modeling their aging process as a Continuous Time Markov Chain (CTMC). The modular technique developed in [21] also encompasses the case of unordered and ordered component states.

When the overall process duration is much shorter than the mean time between failures of used resources, concentrated failure or error probabilities induced by the usage itself become more relevant in the evaluation of the overall reliability. This is for instance the case of system level maintenance procedures, where HW and SW resources are subject to operations exposed to various types of faults, such as disk failures at shutdown and restart, or erroneous restoration of exposed services in infrastructural SW components.

In this paper, we model such classes of phased missions as a sequence of non-concurrent actions that may affect and downgrade a set of resources, evalu-

ating the impact that the operations may have on the system availability in the transient regime. According to a two-mode failure scheme, resources may reach an error state before incurring a breakdown. While quite onerous repair actions are necessary to restore failed resources, lighter rejuvenation operations can be performed to prevent failures of already flawed resources. Actions are subject to a timeout mechanism that limits their repetitions due to subsequent resource failures and to precedence constraints that restrain their feasible orderings.

As a salient trait of the contribution, we face the representation and analysis of steps with non-exponential and firmly bounded duration, which may take relevance in the synchronization of events. To this end, we leverage the method of stochastic state classes [25, 9, 17], which supports the analysis of models with multiple concurrent timed events having a non-Markovian distribution over a possibly bounded support. This enables evaluation of transient availability measures that can be used to support the choice among the possible orderings of actions and the selection of a rejuvenation plan. Computational experience is reported on a case of real complexity to show the potentialities of the approach. The experiments have been performed through a new release of the Oris tool based on the Sirio framework [10, 12, 7].

The rest of the paper is organized as follows. In Section 2, we illustrate the proposed modeling framework of maintenance procedures and we introduce a running case study. In Section 3, we present an extension of stochastic Time Petri Nets (sTPNs) that leaves unchanged their analysis complexity while largely enhancing their modeling convenience, and we discuss the structure of the sTPN model of maintenance procedures with reference to the running case study. In Section 4, we briefly recall the main results of the solution technique of [17] for transient analysis of non-Markovian models and we discuss the conditions that must be satisfied to guarantee its applicability, referring the reader to [17] for more details. In Section 5, we present the experimental results. Conclusions are finally sketched in Section 6.

2 Problem formulation

We address sequential maintenance procedures performing potentially critical operations that require usage of system resources and may affect their status (Section 2.1). This fits the class of PMS provided that adequate assumptions are made to guarantee that the executed operations do not overlap. A running example illustrates model specification (Section 2.2).

2.1 A general class of phased maintenance procedures: stylized facts

We consider phased mission procedures intended to perform non-overlapping operations that may severely affect a set of system resources. When the procedure duration is much shorter than the mean time between age-related failures, then the probability of maintenance-induced failures turns out to be prevailing for the purposes of the evaluation of transient availability measures and can be

well accounted by concentrated failures or error probabilities. This in particular fits the case of maintenance procedures requiring outage of HW and/or SW components, for instance to recover from a failure or to perform upgrades, tests, and configuration changes. Figure 1 illustrates involved concepts.

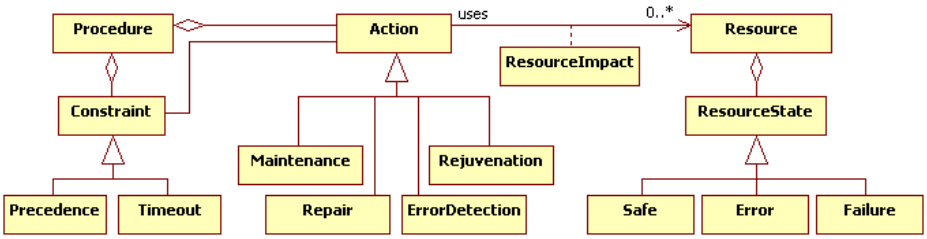


Fig. 1. An UML class diagram representing the maintenance procedure model.

Actions and resources. A procedure is a sequence of non-concurrent *actions* subject to *precedence* constraints that restrict their possible orderings. An action may need to *use* one or more system *resources* to perform its assigned task and its execution may *impact* on some of them, possibly causing errors or failures.

Resource failures. At the beginning of a procedure, all resources are in a *safe* state. During the execution of a maintenance action, a resource may reach an *error* state (from a safe state) or a *failure* state (either from an error state or directly from a safe state). The switch probabilities that represent the state transitions of resources may vary from action to action. While a resource is being used by an action, if the resource reaches an error state then the action is nevertheless successfully completed, whereas if the resource reaches a failure state then the action also fails. The execution time of an action may vary depending on whether the resources in use become flawed or failed. A resource can be recovered from a failure through a *repair* operation.

Failure management. If a resource fails while it is used by an action, then the action is interrupted and a repair operation is started. When the resource has been repaired, the action is restored. Repetitions of the same action due to subsequent failures of its requested resources occur according to the Preemptive Repeat Different (PRD) semantics , i.e., no memory is carried across repetitions. Repetitions are also limited by a *timeout*, which is activated when the action is started for the first time. When the timeout elapses, the action is stopped and the overall procedure is restarted, possibly waiting for ongoing repair operations to be completed. In a variant of this policy, when the timeout of an action expires, the procedure is restarted from that action.

Policy at action completion. At the end of each action, a procedure waits for the completion of ongoing repair operations and possibly performs a *rejuvenation*

of system resources. Specifically, an *error detection* operation can be executed to identify the flawed resources and a rejuvenation activity of failed resources can be started to bring them back to the safe state. This can be performed according to various schemes, e.g., every n executed actions or at the end of selected actions. In a variant of this policy, the rejuvenation of flawed resources that are not used by the remaining actions is avoided and performed only if the procedure is repeated due to the failure of other resources.

2.2 An example

A maintenance procedure can be specified by detailing its actions according to the procedure model shown in Figure 1. As a running example, we consider a procedure made of a sequence of 10 maintenance actions a_1, a_2, ..., a_{10} which may affect a system resource r_1. Table 1 shows a fragment of the specification of such procedure, which pertains to the maintenance action a_1 and to the repair, error detection, and rejuvenation actions performed on r_1. Action a_1 may affect r_1 while performing its requested task. When a_1 is in execution, if r_1 is in a safe state then it remains safe, reaches an error state, or fails with probability 0.75, 0.23, and 0.02, respectively, and the execution time of a_1 is uniformly distributed over $[8, 10]$, $[8, 10]$, and $[2, 5]$ min, respectively. Conversely, if r_1 is in an error state, then it remains flawed or fails with probability 0.75 and 0.25, respectively, and the time spent in the execution of a_1 has a uniform distribution over $[8, 10]$ and $[2, 5]$ min, respectively. The repair of r_1 requires a uniformly distributed time over $[30, 45]$ min. Error detection is performed on r_1 at the completion of a_1 and a_6, triggering a subsequent rejuvenation if r_1 is in an error state. The error detection time and the rejuvenation time have a uniform distribution supported over $[0, 1]$ minutes and $[12, 15]$ minutes, respectively.

Maintenance actions								
Act.	**Time Out**	**Res.**	**Safe2Safe Ex. time**	**Safe2Err Ex. time**	**Safe2Fail Ex. time**	**Err2Err Ex. time**	**Err2Fail Ex. time**	**Fail2Fail Ex. time**
a_1	60	r_1	$p = 0.75$ $[8, 10]$, unif	$p = 0.23$ $[8, 10]$, unif	$p = 0.02$ $[2, 5]$, unif	$p = 0.75$ $[8, 10]$, unif	$p = 0.25$ $[2, 5]$, unif	$p = 1$ $[2, 5]$, unif

Repair actions			
Action	**Res.**	**Execution time**	**Triggering condition**
rep_1	r_1	$[30, 45]$, unif	r_1 failed

Error detection actions			
Act.	**Res.**	**Execution time**	**Triggering condition**
$errd_1$	r_1	$[0, 1]$, unif	a_1 completed || a_6 completed

Rejuvenation actions			
Act.	**Res.**	**Execution time**	**Triggering condition**
rej_1	r_1	$[12, 15]$, unif	r_1 error detected

Table 1. A fragment of the specification of a procedure (times expressed in minutes).

3 Modeling

The specification of a maintenance procedure can be translated into a formal model that supports the deployment of a theory of analysis. We formulate the model as an extension of *stochastic Time Petri Nets* (sTPN) [25, 9] with enabling and flush functions, which change the enabling condition of transitions and the rule according to which tokens are moved after each firing (Section 3.1). This augments the modeling convenience by facilitating the representation of dependent actions and decision-making activities, without restricting the model expressivity or impacting on the subsequent analysis (Section 3.2). The proposed sTPN extension is basically equivalent to SRNs [23].

3.1 An extension of stochastic time Petri nets

Syntax. An sTPN is a tuple $\langle P; T; A^-; A^+; A^\cdot; m_0; EFT^s; LFT^s; \mathcal{F}; \mathcal{C}; E; L\rangle$.

The first ten elements are the model of *stochastic Time Petri Nets* (sTPN) [25, 9]. Specifically, P is a set of places; T is a set of transitions disjoint from P; $A^- \subseteq P \times T$, $A^+ \subseteq T \times P$, and $A^\cdot \subseteq P \times T$ are the sets of precondition, postcondition, and inhibitor arcs; $m_0 : P \to \mathbb{N}$ is the initial marking associating each place with an initial non-negative number of tokens; $EFT^s : T \to \mathbb{Q}_0^+$ and $LFT^s : T \to \mathbb{Q}_0^+ \cup \{\infty\}$ associate each transition with a *static Earliest Firing Time* and a (possibly infinite) static *Latest Firing Time*, respectively ($EFT^s(t) \leq LFT^s(t)\ \forall\ t \in T$); $\mathcal{C} : T \to \mathbb{R}^+$ associates each transition with a weight; $\mathcal{F} : T \to F_t^s$ associates each transition with a static Cumulative Distribution Function (CDF) supported over its static firing interval $[EFT^s(t), LFT^s(t)]$.

As usual in Petri Nets, a place p is said to be an *input*, an *output*, or an *inhibitor* place for a transition t if $\langle p, t\rangle \in A^-$, $\langle t, p\rangle \in A^+$, or $\langle p, t\rangle \in A^\cdot$, respectively. As typical in Stochastic Petri Nets, a transition t is called *immediate* (IMM) if $[EFT^s(t), LFT^s(t)] = [0, 0]$ and *timed* otherwise. A timed transition t is called *exponential* (EXP) if $F_t^s(x) = 1 - e^{\lambda x}$ over $[0, \infty]$ for some rate $\lambda \in \mathbb{R}_0^+$ and *general* (GEN) otherwise. A GEN transition t is called *deterministic* (DET) if $EFT^s(t) = LFT^s(t) > 0$ and *distributed* otherwise (i.e., $EFT^s(t) \neq LFT^s(t)$). For each distributed transition t, we assume that F_t^s is absolutely continuous over its support $[EFT^s(t), LFT^s(t)]$ and, thus, that there exists a Probability Density Function (PDF) f_t^s such that $F_t^s(x) = \int_0^x f_t^s(y)dy$.

E and L extend the model of sTPN with enabling and flush functions, respectively. $E : T \to \{true, false\}^{\mathbb{N}^P}$ associates each transition $t \in T$ with an *enabling function* $E(t) : \mathbb{N}^P \to \{true, false\}$ that, in turn, associates each marking $m : P \to \mathbb{N}$ with a boolean value; $L : T \to \mathcal{P}(P)^{\mathbb{N}^P}$ associates each transition $t \in T$ with a *flush function* $L(t) : \mathbb{N}^P \to \mathcal{P}(P)$ that, in turn, associates each marking $m : P \to \mathbb{N}$ with a subset of P, i.e., an element of the power set of P.

Semantics. The *state* of an sTPN is a pair $\langle m, \tau\rangle$, where $m : P \to \mathbb{N}$ is a marking that associates each place with a non-negative number of tokens and $\tau : T \to \mathbb{R}_0^+$ associates each transition with a (dynamic) real-valued time-to-fire.

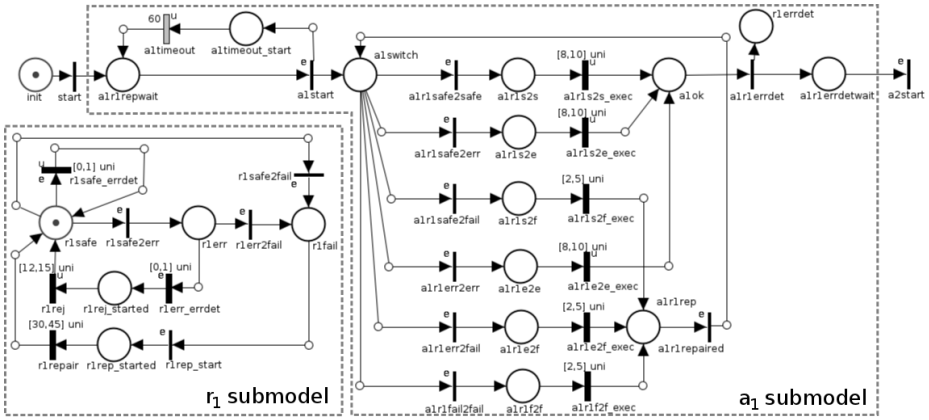
A transition t is *enabled* by marking m if: *i)* each of its input places contains at least one token (i.e., $m(p) \geq 1 \ \forall \ \langle p,t \rangle \in A^-$), *ii)* none of its inhibitor places contains any token (i.e., $m(p) = 0 \ \forall \ \langle p,t \rangle \in A^\cdot$), and *iii)* its enabling function evaluates to true in marking m (i.e., $E(t)(m) = true$). An enabled transition t is *firable* in state $s = \langle m, \tau \rangle$ if its time-to-fire is not higher than that of any other transition enabled by marking m (i.e., $\tau(t) \leq \tau(t') \ \forall \ t' \in T^e(m)$, where $T^e(m)$ is the set of transitions that are enabled by m). When multiple transitions are firable, one of them is selected as the firing transition according to the random switch determined by $\mathcal{C}$. Specifically, $Prob\{t \text{ is selected}\} = \mathcal{C}(t) / \sum_{t_i \in T^f(s)} \mathcal{C}(t_i)$, where $T^f(s)$ is the set of transitions that are firable in s.

The state of an sTPN evolves depending on the times-to-fire sampled by transitions and the resolution of random switches according to the weights of transitions. Specifically, when a transition t fires, the state $s = \langle m, \tau \rangle$ is replaced by a new state $s' = \langle m', \tau' \rangle$. Marking m' is derived from marking m by: *i)* removing a token from each input place of t and assigning zero tokens to the places belonging to the subset $L(t)(m)$ of P (identified by the value of the flush function of t in m), which yields an intermediate marking m_{tmp}, *ii)* adding a token to each output place of t, which finally yields m'. Transitions that are enabled both by m_{tmp} and by m' are said *persistent*, while those that are enabled by m' but not by m_{tmp} or m are said *newly-enabled*. If the fired transition t is still enabled after its own firing, it is always regarded as newly enabled [4, 24]. For any transition t_p that is persistent after the firing of t, the time-to-fire is reduced by the time elapsed in the previous state s (which is equal to the time-to-fire of t measured at the entrance in s), i.e., $\tau'(t_p) = \tau(t_p) - \tau(t)$. For any transition t_n that is newly-enabled after the firing of t, the time-to-fire takes a random value sampled in the static firing interval according to the static CDF $F^s_{t_n}$, i.e., $EFT^s(t_n) \leq \tau'(t_n) \leq LFT^s(t_n)$, with $Prob\{\tau'(t_n) \leq x\} = F^s_{t_n}(x)$.

3.2 Deriving an sTPN model of maintenance procedures

The specification of a maintenance procedure can be translated into a corresponding sTPN model, which is made of a submodel for each action and a submodel for each resource that the actions may affect. The sTPN shown in Figure 2 is a model fragment of the procedure introduced in Section 2.2, specifically corresponding to action a_1 and resource r_1 specified in Table 1.

The IMM transition *start* models the outset of the overall procedure and its output place is chained with the IMM transition $a1start$ representing the beginning of action a_1. When $a1start$ fires, a token arrives in $a1timeout\_start$, enabling the DET transition $a1timeout$ which models the timeout of 60 minutes associated with a_1. The firing of $a1start$ also deposits a token in $a1switch$, which is an input place for the 6 IMM transitions that model the concentrated probabilities of error/failure affecting r_1 as a consequence of the usage by a_1. Specifically, if r_1 is safe, then it may remain safe or become either flawed or failed with probability 0.75, 0.23, and 0.02, respectively. This is modeled by the random switch among the IMM transitions $a1r1safe2safe$, $a1r1safe2err$, and



(a)

Transition	Enabling function	Flush function	Weight
a1start	r1rep_started==0	-	1
a1r1safe2safe	r1safe>0	-	75
a1r1safe2err	r1safe>0	-	23
a1r1safe2fail	r1safe>0	-	2
a1r1err2err	r1err>0	-	75
a1r1err2fail	r1err>0	-	25
a1r1fail2fail	r1fail>0	-	1
a1r1s2s_exec	-	{a1_timeout}	1
a1r1s2e_exec	-	{a1_timeout}	1
a1r1e2e_exec	-	{a1_timeout}	1
a1r1_repaired	r1safe>0	-	1
a1timeout	-	{a1_switch, safe2safe, safe2err, safe2fail, err2err, err2fail, a1_r1repaired}	1
a2start	r1errdet==0	-	1
r1safe2err	safe2err>0	-	1
r1err2fail	err2fail>0	-	1
r1safe2fail	safe2fail>0	-	1
r1repstart	a1r1rep>0	-	1
r1safe_errdet	r1_errdet==1	{r1_errdet}	1
r1err_errdet	r1_errdet==1	-	1
r1rej	-	{r1_errdet}	1

(b)

Fig. 2. An sTPN fragment of the example procedure of Section 2.2, representing the action a_1 and the resource r_1 specified in Table 1 **(a)**. A table that details the enabling functions, the flush functions, and the weights of the transitions that appear in the model fragment **(b)**. The entire model of the procedure integrates the r_1 submodel and 10 action submodels like the a_1 submodel.

$a1r1safe2fail$, which are actually enabled if $r1safe$ contains a token and have a weight equal to 75, 23, and 2, respectively. In a similar manner, if r_1 is flawed, then it may remain flawed or become failed, which is represented by the random switch between $a1r1err2err$ and $a1r1err2fail$. Conversely, if r_1 is failed, then the only enabled transition of the 6 mentioned IMM transitions is $a1r1fail2fail$, which models the fact that r_1 remains failed until a repair operation is started.

When the outcome of the random switch in the a_1 submodel corresponds to an r_1 state transition, then the corresponding IMM transition in the r_1 submodel becomes firable. Specifically, if $a1r1safe2err$, $a1r1safe2fail$, or $a1r1err2fail$ fires, then a token arrives in $a1r1s2e$, $a1r1s2f$, or $a1r1e2f$, respectively, thus making $r1safe2err$, $r1safe2fail$, or $r1err2fail$ fireable, respectively.

The firing of $a1r1safe2safe$, $a1r1safe2err$, and $a1r1err2err$ enables $a1r1s2s\_exec$, $a1r1s2e\_exec$, and $a1r1e2e\_exec$, respectively, which model the execution time of a_1 in the cases of successful completion. When one of the latter transitions fires, a token is removed from $a1timeout\_start$ (i.e., the timeout is stopped) and a token is added to $a1ok$. This enables $a1r1errdet$ which triggers an error detection operation on r_1. Specifically, when a token arrives in $r1errdet$, either $r1safe\_errdet$ or $r1err\_errdet$ (in the r_1 submodel) becomes enabled depending on whether r_1 is safe or flawed, respectively: *i)* If r_1 is safe, the firing of $r1safe\_errdet$ simultaneously removes and adds a token to $r1safe$ (in the r_1 submodel) and removes a token from $r1errdet$ (in the a_1 submodel), thus enabling $a1start$ which models the beginning of the subsequent action. *ii)* If r_1 is flawed, the firing of $r1err\_errdet$ removes a token from $r1err$ and deposits a token in $r1rej\_started$ (in the r_1 submodel), enabling $r1rej$ which models a rejuvenation. The firing of $r1rej$ adds a token to $r1safe$ (in the r_1 submodel) and removes a token from $r1errdet$ (in the a_1 submodel), thus enabling $a2start$.

The firing of $a1r1safe2fail$, $a1r1err2fail$, and $a1r1fail2fail$ enables $a1r1s2f\_exec$, $a1r1e2f\_exec$, and $a1r1f2f\_exec$, respectively, which model the execution time of a_1 in the cases of unsuccessful completion. The firing of one of the latter transitions adds a token to $a1r1rep$ enabling $r1rep\_start$ (in the r_1 submodel), whose firing in turn enables $r1repair$ which models a repair operation performed on r_1. When $r1repair$ fires, a token is moved in $r1safe$, enabling $a1r1repaired$ (in the a_1 submodel) whose firing brings a token back to $a1switch$ so that a_1 is repeated.

If $a1timeout$ fires before a token arrives in $a1ok$, then all the tokens in the a_1 submodel are removed, a token is deposited in $a1repwait$, and $a1start$ becomes fireable as soon as $r1repstarted$ contains no tokens (i.e., if the timeout elapses before the successful completion of a_1, then a_1 is restarted, possibly waiting for an on-going repair of r_1 to be completed).

4 Quantitative analysis

We discuss the conditions that guarantee the applicability of the solution technique of [17] to the analysis of procedure models that underlie a Generalized Semi-Markov Process (GSMP) [20, 1] (Section 4.1) or a Markov Regenerative Process (MRP) [13, 1, 2] (Section 4.2). The approach is efficiently implemented in the new release of the Oris tool based on the Sirio framework [10, 12, 7] under the assumption that all timed transitions have expolynomial PDF. For space limitations, the reader is referred to [17] for the details on the analysis technique.

4.1 Transient analysis

The sTPN model derived in Section 3.2 includes multiple concurrent GEN transitions with bounded support, e.g., a repair action enabled together with the timeout associated with a maintenance action. According to this, the model underlies a GSMP [20, 1] with equal-speed timers, for which a viable approach to transient analysis within any given time bound is the solution technique of [17]. The approach samples the state of the underlying GSMP after each transition firing, maintaining a timer τ_{age} that accounts for the absolute time elapsed since the entrance in the initial state. This identifies an embedded Discrete Time Markov Chain (DTMC) called *transient stochastic graph*. A state in the embedded DTMC is named *transient stochastic state class* (transient class for short) and provides the marking of the sTPN plus the joint support and PDF of τ_{age} and the times-to-fire of the enabled transitions. The marginal PDF of τ_{age} permits to derive the PDF of the absolute time at which the transient class can be entered. This enables evaluation of continuous-time transient probabilities of reachable markings within a given time horizon, provided that either the number of transient classes that can be reached within that time interval is bounded (*exact analysis*) or it can be truncated under the assumption of some approximation threshold on the total unallocated probability (*approximated analysis*).

The number of transient classes enumerated within a given time bound is guaranteed to be finite by Lemma 3.4 of [17] provided that the state class graph of the underlying TPN model is finite and does not include a cycle that can be executed in zero time. The state class graph of the underlying TPN model can be regarded as a non-deterministic projection of the transient stochastic class graph and its finiteness is assured under fairly general conditions by Lemma 3.2 of [17], which is not addressed here for the shortness of discussion.

If the state class graph of the underlying TPN model includes cycles that can be executed in zero time, then the number of transient classes enumerated within a given time bound is not finite. In this case, termination can be guaranteed in probability by Lemma 3.5 of [17] if cycles that must be executed in zero time are not allowed. This permits to stop the enumeration when the total probability of reaching one of the discarded successor transient classes within a given time bound is lower than a predefined threshold. In particular, approximated analysis can be leveraged also when the number of transient classes enumerated within a given time bound is theoretically finite but practically too large to afford the enumeration within a reasonable computation time.

The complexity of the analysis actually grows with the number of enumerated transient classes and, thus, with the time horizon. In the experiments performed in this paper, approximated analysis turned out to be feasible with a computation time lower than 5 minutes up to procedures made of 4-5 maintenance actions.

4.2 Transient analysis of Markov regenerative processes

The issue of complexity can be overcome for models that underlie an MRP that within a finite number of steps always reaches a regeneration point, which is a

state where the future behavior of the stochastic process is independent from the past behavior through which the state has been reached. In the approach of [17], regeneration points can be identified as the transient classes where all times-to-fire are either *i)* newly-enabled, or *ii)* exponentially distributed, or *iii)* deterministic, or *iv)* bounded to take a deterministic delay with respect to a time-to-fire satisfying any of the previous conditions. According to this, the sTPN model derived in Section 3.2 is guaranteed to reach a regeneration point at the completion of each action, since the subsequent action is not started until any ongoing repair is completed. When the underlying stochastic process satisfies this condition, the solution technique of [17] can be limited to the first regeneration epoch and repeated from every regenerative point. This supports the derivation of the local and global kernels that characterize the behavior of the MRP [13, 1, 2] and enables the evaluation of the transient probabilities of reachable markings at any time through the numerical integration of generalized Markov renewal equations (*regenerative exact analysis*). Termination is guaranteed if the number of transient classes reached within the first regeneration epoch is bounded. This is assured by Lemma 4.1 of [17] if the state class graph of the underlying TPN model is finite and every cycle that it contains visits at least one state class that is a (non-deterministic) projection of a regenerative transient class. If the number of transient classes reached within the first regeneration epoch is not finite or practically too large, termination can be guaranteed in probability by Lemma 3.5 of [17] under the assumption of a time bound and an approximation threshold on the total unallocated probability (*approximated regenerative analysis*).

In the experiments performed in this paper, regenerative analysis permitted to afford cases of real complexity concerning procedure made of 10 or more maintenance actions, and it seems to be a solid ground for future developments.

5 Computational Experience

We consider the procedure introduced in Section 2.2, which is made of 10 maintenance actions a_1, a_2, ..., a_{10} that use and may affect a resource r_1. For the simplicity of interpretation of experimental results, we adopt for each maintenance action the specification given in Table 1. According to this, the sTPN model of such procedure is a composition of the r_1 submodel shown in Figure 2 with 10 action submodels equal to the a_1 submodel shown in Figure 2. The switch probabilities and the supports of the temporal parameters appearing in Table 1 were selected according to general experience at NEC Corporation [22], with the aim of experimenting the approach on plausible data. The way how experimental data are acquired and interpreted to derive not only the expected min-max duration of temporal parameters but also their distribution is still matter of study. In such cases, the principle of insufficient reason, or maximum entropy can be advocated to motivate the assumption of a uniform distribution [3].

To illustrate the potentialities of the approach, the experimentation is finalized to evaluate, for each maintenance action a_i, the transient probability that a_i is completed and the subsequent action a_{i+i} (if any) is still ongoing. Specifically,

this measure of interest is derived as the sum of the transient probabilities of any marking such that: *i)* the submodel of any action that precedes a_i or follows a_{i+1} (if any) contains no tokens, and *ii)* *aiok* in the a_i submodel contains a token, or *aiok* and $air1errdet$ in the a_{i+1} submodel contain no tokens. This permits to derive the time at which a given action or the overall procedure has been successfully completed with an assigned probability or, vice-versa, the probability that a given action or the overall procedure has been successfully completed within a given time bound. The considered performance measures also permit to evaluate different strategies for error detection and rejuvenation, also with respect to different feasible orderings of the maintenance actions.

The experiments were performed through the new release of the Oris tool based on the Sirio framework [10, 12, 7]. Regenerative analysis with approximation threshold equal to 0.01 and time bound equal to 300 minutes was repeated for four different policies of error detection and rejuvenation. In all the cases, the analysis took 1 s to enumerate transient classes and less than 4 minutes to solve Markov renewal equations.

Error detection & rejuvenation never performed. Figure 3a shows that the probability of successful completion of the overall procedure within 125, 150, and 175 minutes is nearly 0.29, 0.35, and 0.74, respectively, while the time by which the procedure has been successfully completed with probability higher than 0.99 is 293 minutes.

Error detection & rejuvenation performed every 5 actions. Figure 3b shows that rejuvenation improves the probability of successful completion of the overall procedure within 150 and 175 minutes to nearly 0.48 and 0.78, respectively, while decreasing the probability of successful completion within 125 minutes to 0.136134. The time by which the procedure has been successfully completed with probability higher than 0.99 is 269 minutes.

Error detection & rejuvenation performed every 3 actions. Figure 3c shows that the probability of successful completion of the overall procedure within 150 minutes is increased to nearly 0.54, while the probability of successful completion within 125 minutes is further decreased to 0.11 and the probability of successful completion within 175 minutes is slightly reduced to nearly 0.77. The time by which the procedure has been successfully completed with probability higher than 0.99 is 267 minutes.

Error detection & rejuvenation performed after each action. Figure 3d shows that the probability of successful completion of the overall procedure within 125 minutes is nearly halved and it is equal to 0.06; the probability of successful completion within 150 minutes is decreased to nearly 0.47, while the probability of successful completion within 175 minutes is increased to 0.79. The time by which the procedure has been successfully completed with probability higher than 0.99 is 258 minutes.

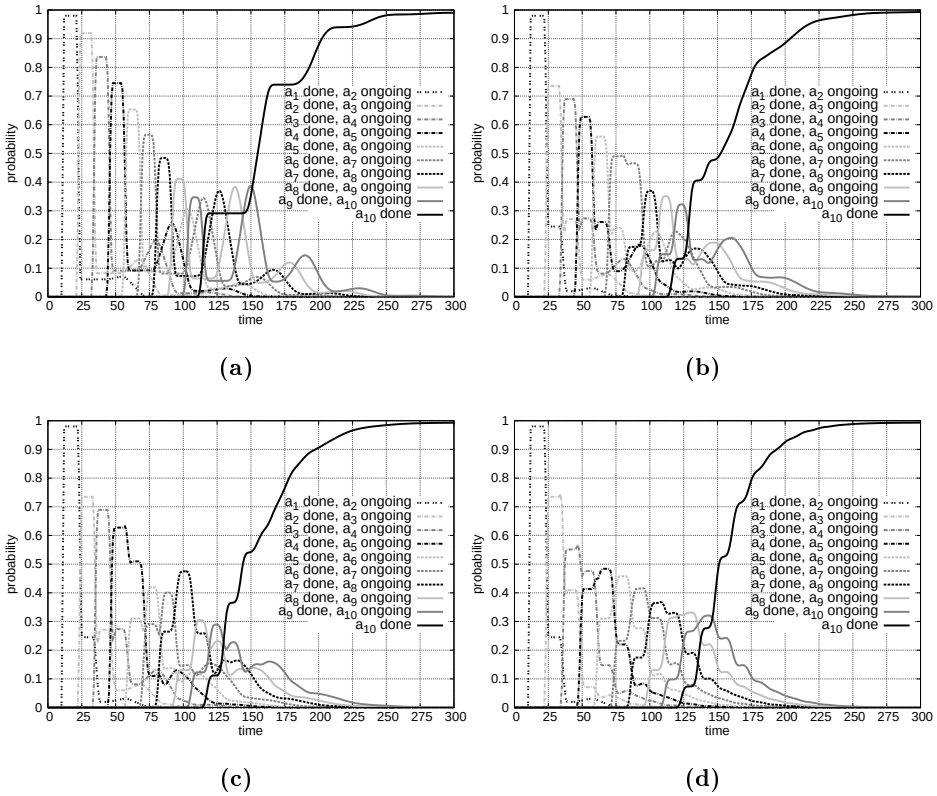


Fig. 3. Transient probability that a_i is completed and a_{i+i} (if any) is ongoing for each action a_i of the procedure specified in Section 2.2, under the assumption that error detection and rejuvenation are: never performed **a)**, performed every 5 actions **b)**, performed every 3 actions **c)**, and performed after each action (except a_{10}) **d)**.

6 Conclusions

We experimented with the approach of stochastic state classes [17] in quantitative evaluation of maintenance procedures that may induce errors or failures of system resources. To this end, we considered a general modeling framework, including precedence and timeout constraints on maintenance actions, repair operations of system resources, and strategies for error detection and rejuvenation of resources. As a relevant aspect, the execution times of such actions may have a non-Markovian distribution over a bounded support. This yields models that underlie an MRP with multiple concurrently enabled GEN timers, which can be effectively solved through the regenerative approach to transient analysis developed in [17]. The method permits to derive transient availability measures, which can be used to support the selection of a rejuvenation plan and the choice among feasible orderings of actions. Computational experience addresses a relatively challenging case study that is able to fit the complexities of reality beyond

the enabling restriction. In so doing, a major contribution of this paper consists in a proof of the applicability of the solution technique of [17], which seems to be extremely promising for the analysis of models of higher complexity and size. While applied to the solution of a specific problem pertaining to the evaluation of critical maintenance procedures, the proposed approach is formulated as a method for modeling and analysis of a more general class of critical applications referred to the abstraction of PMS.

The proposed approach is amenable to integration within a model driven development process where the results of quantitative analysis can be used to support iterative feedback cycles [11, 16]. Specifically, the temporal parameters with unknown distribution are initially associated with a uniform distribution or with a distribution guessed by analogy with previous implementations. Then, they are progressively refined on the basis of quantitative measures and the results of a profiling technique for the estimation of execution times.

Acknowledgments

We kindly thank Stefano Ballerini for his support in the experimentation stage.

References

1. G. Ciardo and R. German and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. *IEEE Trans. SW Eng.*, 20(7):506–515, 1994.
2. A. Bobbio and M. Telek. Markov regenerative SPN with non-overlapping activity cycles. *Int. Comp. Perf. and Dependability Symp. - IPDS95*, pages 124–133, 1995.
3. S. Bernardi, J. Campos, and J. Merseguer. Timing-failure risk assessment of UML design using time Petri net bound techniques. *IEEE Transactions on Industrial Informatics*, 7(1):90–104, February 2011.
4. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on SW Eng.*, 17(3):259–273, March 1991.
5. A. Bobbio, A. Puliafito, M. Scarpa, and M. Telek. WebSPN: a web-accessible Petri net tool. In *Proc. Conf on Web-based Modeling and Simulation*, 1998.
6. D. Bruneo, S. Distefano, F. Longo, and M. Scarpa. Stochastic evaluation of QoS in service-based systems. *IEEE Trans.on Parallel and Distributed Systems*, 2012.
7. G. Bucci, L. Carnevali, L. Ridi, and E. Vicario. Oris: a tool for modeling, verification and evaluation of real-time systems. *Int. Journal of SW Tools for Technology Transfer*, 12(5):391 – 403, 2010.
8. L. Burdick, J.B.Fussell, D. Rasmuson, and J. Wilson. Phased mission analysis: a review of new developments and an application. *IEEE Trans. on Rel.*, R-26(1):43–49, April 1977.
9. L. Carnevali, L. Grassi, and E. Vicario. State-density functions over DBM domains in the analysis of non-Markovian models. *IEEE Trans. on SW Eng.*, 35(2):178–194, 2009.
10. L. Carnevali, L. Ridi, and E. Vicario. A framework for simulation and symbolic state space analysis of non-Markovian models. In *SAFECOMP*, pages 409–422, 2011.

11. L. Carnevali, L. Ridi, and E. Vicario. Putting preemptive time Petri nets to work in a V-model SW life cycle. *IEEE Trans. on SW Eng.*, 37(6), Nov./Dec. 2011.
12. L. Carnevali, L. Ridi, and E. Vicario. Sirio: A framework for simulation and symbolic state space analysis of non-Markovian models. In 8^{st} *Int. Conf. on Quantitative Evaluation of Systems (QEST '11)*, pages 153–154, 2011.
13. H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov regenerative stochastic Petri nets. *Perform. Eval.*, 20(1-3):337–357, 1994.
14. M. Combacau, P. Berruet, E. Zamai, P. Charbonnaud, and A. Khatab. Supervision and monitoring of production systems. In *Proc. IFAC Conf. on Management and Control of Production and Logistics*, 2000.
15. J. B. Dugan. Automated analysis of phased-mission reliability. *IEEE Trans. on Reliability*, 40(1):45–52, 1991.
16. J. B. Dugan. Galileo: A tool for dynamic fault tree analysis. In *Computer Performance Evaluation / TOOLS*, pages 328–331, 2000.
17. A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Perf. Eval.*, 69(7-8):315–335, 2012.
18. I. Mura and A. Bondavalli. Markov regenerative stochastic Petri nets to model and evaluate phased mission systems dependability. *IEEE Trans. Comput.*, 50(12):1337–1351, Dec. 2001.
19. Y. Ou and J. B. Dugan. Modular solution of dynamic multi-phase systems. *Reliability, IEEE Transactions on*, 53(4):499–508, 2004.
20. P. W. Glynn. A GSMP formalism for discrete-event systems. *Proceedings of the IEEE*, 77:14–23, 1989.
21. A. Shrestha, L. Xing, and Y. Dai. Reliability analysis of multistate phased-mission systems with unordered and ordered states. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4):625–636, 2011.
22. K. Tadano, J. Xiang, M. Kawato, and Y. Maeno. Automatic synthesis of SRN models from system operation templates for availability analysis. In *SAFECOMP*, pages 296–309, 2011.
23. K. S. Trivedi. *Probability and statistics with reliability, queuing, and computer science applications*. John Wiley and Sons, New York, 2001.
24. E. Vicario. Static analysis and dynamic steering of time dependent systems using time Petri nets. *IEEE Trans. on SW Eng.*, 27(1):728–748, August 2001.
25. E. Vicario, L. Sassoli, and L. Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. on SW Eng.*, 35(5):703–719, 2009.
26. D. Wang and K. S. Trivedi. Reliability analysis of phased-mission system with independent component repairs. *IEEE Trans. on Reliability*, 56(3):540–551, 2007.
27. K. A. Weiss, N. Leveson, K. Lundqvist, N. Farid, and M. Stringfellow. An analysis of causation in aerospace accidents. In *Digital Avionics Systems Conference (DASC)*, volume 1, pages 4A3–1. IEEE, 2001.
28. L. Xing and J. B. Dugan. Analysis of generalized phased-mission system reliability, performance, and sensitivity. *IEEE Trans. on Reliability*, 51(2):199–211, 2002.
29. L. Xing and J. B. Dugan. A separable ternary decision diagram based analysis of generalized phased-mission reliability. *IEEE Trans. on Rel.*, 53(2):174–184, 2004.
30. R. Yam, P. Tse, L. Li, and P. Tu. Intelligent predictive decision support system for condition-based maintenance. *Int. Journal of Advanced Manufacturing Technology*, 17(5):383–391, 2001.
31. X. Zang, N. Sun, and K. S. Trivedi. A BDD-based algorithm for reliability analysis of phased-mission systems. *Reliability, IEEE Transactions on*, 48(1):50–60, 1999.