Dependability Modeling and Evaluation of Multiple Phased Systems using DEEM

Andrea Bondavalli[1], Silvano Chiaradonna[2], Felicita Di Giandomenico[2] and Ivan Mura[3]

[1] DIS, Univ. of Florence, Via Lombroso 6/17 I-50134 Firenze Italy, a.bondavalli@dsi.unifi.it

[2] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy, Silvano.Chiaradonna@guest.cnuce.cnr.it

[2] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy, digiandomenico@iei.pi.cnr.it

[3] Motorola Technology Center Italy, Via P. C. Boggio 65/A, 10138 Torino, Italy, Ivan_Mura@mot.com

Abstract. Multiple-Phased Systems, whose operational life can be partitioned in a set of disjoint periods, called "phases", include several classes of systems such as Phased Mission Systems and Scheduled Maintenance Systems. Because of their deployment in critical applications, the dependability modeling and analysis of Multiple-Phased Systems is a task of primary relevance. However, the phased behavior makes the analysis of Multiple-Phased Systems extremely complex. This paper describes DEEM, a dependability modeling and evaluation tool specifically tailored for Multiple Phased Systems, and its use for the solution of representative MPS problems. DEEM supports the methodology proposed in [28, 29] although not yet completely. When compared to general purpose DSPN tools [17], DEEM offers advantages on the modeling side (PhN and SN sub-models neatly model the phase-dependent behaviors of MPS), and on the evaluation side (a specialized algorithm allows a relevant reduction of the solution cost and time). Thus, DEEM is able to deal with all the scenarios of MPS that have been analytically treated in the literature, at a cost which is comparable with that of the cheapest ones [7, 26, 27, 34], completely solving the issues posed by the phased-behavior of MPS. DEEM is freely available to the academic world, for information see http://bonda.cnuce.cnr.it/DEEM.

Categories and Subject Descriptors: C4 **[Performance of Systems]**: Reliability, availability, survivability; C4 **[Performance of Systems]**: Fault Tolerance; C4 **[Performance of Systems]**: Modeling Techniques; D.2.4 **[Software/Program Verification]**: Reliability, Validation; **D.2.8 [Metrics]:** Performance measures; I.6 **[Simulation and Modeling]**: Model validation and analysis.

1 Introduction

Many embedded systems devoted to the control and management of critical activities have to perform a series of tasks that must be accomplished in sequence. Their operational life consists of a sequence of non-overlapping periods, called *phases*. These systems are often called Multiple-Phased Systems (MPS). MPS are very general, since their phases can be distinguished along a wide variety of differentiating features.

(1) During a specific phase, an MPS is devoted to the execution of a particular set of tasks, which may be different from the activities performed within other phases.

(2) The performance and dependability requirements of an MPS can be completely different from one phase to another.

(3) During some phases the system may be subject to a particularly stressing environment, thus experiencing dramatic increases in the failure rate of its components.

(4) In order to accomplish its mission, an MPS may need to change its configuration over time, to adopt the most suitable one with respect to the performance and dependability requirements of the phase being currently executed, or simply to be more resilient to an hazardous external environment.

(5) The successful completion of a phase, as well as the activities performed therein, may bring a different benefit to the MPS with respect to that obtained with other phases.

Examples of MPS can be found in various application domains (nuclear, aerospace, transportation, electronic, and many other industrial fields). They include systems for the aided-guide of aircraft, whose mission-time is divided into several phases such as take-off, cruise, landing, with completely different requirements. A very important sub-class of MPS is represented by the so-called Scheduled Maintenance Systems encountered in almost all the application domains where an artifact is to be used for long time and is periodically subject to maintenance actions. An SMS is easily formulated as an MPS, for which operational and maintenance phases alternate according to a prefixed schedule.

Because of their deployment in critical applications, the dependability modeling and analysis of MPS have been considered tasks of primary relevance, and many different approaches have appeared in the literature [8, 9, 15, 26, 28, 29, 33-35]. However, modeling a MPS can be a complex task even inside one single phase; when a multiplicity of phases and the dependencies among them are to be taken into account, additional difficulties are encountered.

This paper describes DEEM (DEpendability Evaluation of Multiple-phased systems) [7], the dependability modeling and evaluation tool specifically tailored for the time-dependent analysis of MPS, being currently developed at the University of Florence and CNUCE-CNR. DEEM

supports the methodology proposed in [28, 29] for the dependability modeling and evaluation of MPS. This methodology relies upon Deterministic and Stochastic Petri Nets (DSPN) as a modeling formalism and on Markov Regenerative Processes (MRGP) for the model solution. DEEM is equipped with many modeling features that improve the expressive power of Petri Net modes, the same modeling features as those already available in several general purpose tools such as SPNP [14], UltraSAN [32], DSPNexpress [23, 24], PANDA [3], and SURF-2 [6].

The rich set of DEEM modeling features is made accessible through a X-Window Graphical User Interface (GUI). The GUI supports: a) model construction of MPS, including definition of attributes of model objects; b) definition of the relevant dependability measures for the system under analysis; and c) models executions (which can also be done through a command line).

The DEEM solution algorithm is based on splitting the Markov chains underlying the DSPN for different phases. This way the transient analysis of the overall DSPN model is almost completely reduced to the cheaper problem of the separate solution of each MPS phase.

The paper is organized as follows. Section 2 summarizes the DSPN approach adopted in DEEM to the modeling of MPS and the analytical solution technique highlighting the advantages over previous general MRGP solutions. Section 3 describes the Graphical User Interface of DEEM for modeling, and management of the analyses, describing also the specialized solution algorithm. Then, Section 4 describes the use of DEEM for the solution of representative MPS problems. In particular the DEEM models and model solution of a Phased Mission System [28, 29] and of a Scheduled Maintenance System [9] will be described. These problems have already been investigated and solved by hand in the reference provided; here we wish to point out the advantages of the simple automated support compared to hand made modeling and evaluation. A brief overview of related work is the content of Section 5.

2 The DSPN methodology to model MPS adopted in DEEM

Deterministic and Stochastic Petri Nets (DSPN) have been chosen as the modeling formalism for DEEM, whereas the solution technique finds its ground in the efficient time-dependent analysis of Markov Regenerative Processes (MRGP) presented in [11]. Precisely, DEEM is based on the methodology proposed in [28, 29], which has been explicitly developed to provide a support for the modeling and time-dependent analysis of the MPS dependability related features. The main advance of that methodology is found in the adoption of a highly expressive modeling formalism for concisely represent MPS dynamic behavior, coupled with a powerful analytical solution method of the stochastic processes underlying the models. In its actual version, DEEM does not fully exploit the potentialities of the referred methodology, which is very powerful to include general distributions, but the following restrictions are applied:

1) the duration of the phases is deterministic;

2) each intra-phase process is a time-homogeneous Markov chain (the phase model is restricted to contain only exponential and instantaneous transitions).

2.1 Modeling of MPS using DSPN

DSPN models extend Generalized Stochastic Petri Nets and Stochastic Reward Nets, allowing for the exact modeling of events having deterministic occurrence times. A DSPN model may include immediate transitions, transitions with exponentially distributed firing times, and transitions with deterministic firing times. Due to their high representative power, DSPN models are able to cope with the dynamic structure of MPS, and allow defining very concise models of even quite complex systems, through the use of guards on transitions, immediate transition priorities, halting conditions, multiplicity functions, rate and impulse rewards, etc. Indeed, the treatment of the dependencies among phases is moved from the low level of the Markov chains to the more abstract and easier to handle level of the DSPN.

According to the methodology in [28, 29], models of MPS consist of two logically separate parts: the System Net (SN), which represents the failure/repair behavior of system components, and the Phase Net (PhN), which represents the execution of the various phases. Each net is made dependent on the other one by marking-dependent predicates that modify transition rates, enabling conditions, transition probabilities, multiplicity functions, etc., to model the specific MPS features. Notice that, the predicates expressing enabling conditions are not intended to substitute the classic enabling rules of Petri net model transitions, rather they represent additional constraints that must be fulfilled for a transition to be eligible for firing.

Several advantages are offered by the DSPN approach over previous proposals cited in [28, 29]. First, the modeling features of DSPN allow a very concise representation of MPS, compared with a Markov chain that results in huge models, which readily become sources of errors in modeling. On the contrary, the DSPN high-level approach turns out in an overall MPS model that is concise, easy to understand and to modify. The two parts of the DSPN model represent two different abstraction levels of the same MPS. The mission profile is explicitly modeled in the PhN, and can be very easily modified to represent different MPS. Moreover, the whole modeling procedure limits in itself the possibility of introducing errors inside the models. Various structural properties of the separate Petri net sub-models can be checked to increase the confidence that can be put in the modeling itself. Further, the links among the various sub-models are expressed through predicates of the marking, in a clear and unambiguous way. Phase-triggered reconfigurations, which add a significant complexity to the treatment of dependencies among phases, are easily handled through the implicit mapping which is embedded in the model (as in [2, 5, 15, 33]). Indeed, the mapping between successive phases are

completely specified at the level of the DSPN modeling, thus dramatically reducing the amount of user-assistance needed to define the MPS models with respect to Markov chain based techniques.

2.2 The specialized analytical technique to solve MPS

We summarize in this section the transient solution technique presented in [28, 29] to evaluate dependability related measures of MPS at specific time instants. The algorithm implementation will be described later in Section 3.4. The probability of successful mission completion, the relative impact of each single phase on the overall dependability figures, the impact on MPS reliability of a given maintenance schedule, and the amount of useful work that can be carried out within the mission are among the measures assessable through such technique.

The specialized solution finds its ground by observing that the only deterministic transitions in a DSPN model of a MPS are the phase duration, and that these transitions are enabled one at the time. Thus, the marking process $\{M(t), t \geq 0\}$ of the DSPN is a Markov Regenerative Process (MRGP) [11] for which the firing times of the deterministic transitions are indeed regeneration points. Moreover, the following property holds of the DSPN model of a MPS:

Property 1: in every non-absorbing marking of the DSPN there is always one deterministic transition enabled, which corresponds to the phase being currently executed.

The general solution method for MRGP processes considers computing matrix $V(t)$, whose entry $\dot{m}, \dot{m}'$ is the occupation probability of marking $\dot{m}'$ at time $t \geq 0$ given the initial marking $\dot{m}$. Matrix $V(t)$ is the solution to the generalized Markov renewal equation $V(t) = E(t) + K(t) * V(t)$, where $K(t)$ and $E(t)$ are the global and local kernel matrices [11] and "*" is the convolution operator. Instead of directly attacking the solution of the generalized Markov renewal equation by numerical algorithms or Laplace-Stiltjes transform, DEEM computes matrix $V(t)$ according to the following analytical method, proposed in [28, 29].

Let S denote the state space of the MRGP process, let $1, 2, \mathrm{K}, n$ be the set of phases the MPS can perform, and finally let τ_i denote the duration of phase i, $i = 1, 2, \mathrm{K}, n$. Consider the following subsets of S:

$$S_i = \{\dot{m} \in S \mid \textit{phase i is being performed}\},\ i = 1, 2, \mathrm{K}, n$$

$$S_{n+1} = \{\dot{m} \in S \mid \text{no phase is being performed}\}$$

Owing to "Property 1", and because different phases correspond to distinct markings of the DSPN model, sets S_i, $i = 1, 2, \mathrm{K}, n+1$, are a partition of the marking space S. The stochastic

process $\{M_i(t), t \geq 0\}$, $i = 1, 2, \ldots, n$, defined as the restriction of the MRGP within the execution of phase i, is a continuous-time Markov chain with state space S_i and transition rate matrix Q_i. The transient analysis of the MRGP is carried out by separately considering the evolution of the processes $\{M_i(t), t \geq 0\}$.

Consider the block structure that is induced on matrix $V(t)$ as a result of the marking space partitioning. Each block $V_{i,j}(t)$ is separately computed as follows. Consider the unique path $p(i,j)$ that links phase i to phase j according to the structure of the PhN. This path is a set of phases $p(i,j) = \{p_1, p_2, \ldots, p_r\}$, with $p_1 = i$, and $p_r = j$. Block $V_{i,j}(t)$ is given by:

$$V_{i,j}(t) = \left(\prod_{h=1}^{r-1} e^{Q_{ph} \tau_{ph}} \Delta_{p_h, p_{h+1}} \right) e^{Q_j \left(t - \sum_{h=1}^{r-1} \tau_{ph} \right)} \qquad (1)$$

Where $\Delta_{p_h, p_{h+1}}$, $h = 1, 2, \ldots, r-1$ is the branching probability matrix, whose entry $\Delta^{p_h, p_{h+1}}_{\vec{m}, \vec{m}'}$ is defined as the probability that $\vec{m}'$ is the initial marking of phase p_{h+1}, given that $\vec{m}$ is the marking at the end of phase p_h.

3 Description of DEEM

DEEM has been explicily designed and implemented to support (part of) the methodology proposed in [28, 29] for the dependability modeling and evaluation of MPS. The tool is written in C, runs under Solaris (SUN) and Linux (PowerPC and Pentiumclass) workstations and possesses a X-Window Graphical User Interface (GUI), inspired by [3], realized using an X11 installation with Motif runtime Libraries. The main features the DEEM's GUI supports are:

1 MPS model construction based on the DSPN modeling formalism [1], according to the methodology defined in [28, 29];

2 Definition and management of evaluation scenarios (studies) and setting of the parameter values for multiple evaluations;

3 Definition of dependability measures of interest, through the general mechanism of marking-dependent reward functions;

4 Activation of the transient analysis to evaluate the dependability measures;

5 Saving in a file of the state distribution of the model at the end of the transient analysis and loading from a file of the initial state distribution used for the transient analysis;

6 Documentation of the MPS model producing a LATEX file containing all model information.

Other features the DEEM's GUI supports are those tipycal of many similar tool, such as file and editing utilities, help-on-line, etc, as shown in the menu bar of Figure 1.

3.1 Modeling MPS with DEEM

A DEEM model may include immediate transitions, represented by a thin line, transitions with exponentially distributed firing times, represented by empty rectangles, and transitions with deterministic firing times, represented by filled rectangles. Moreover, DEEM makes available a set of modeling features that significantly improve DSPN expressiveness: arbitrary functions of the model marking may be employed to define firing times (rates or deterministic times) of timed transitions, probabilities of immediate transitions, enabling conditions (named guards) of the transitions, arc multiplicities and rewards. This rich set of modeling features, accessible through a Graphical User Interface, provides DEEM with the ability of supporting the bipartite MPS modeling approach described in the previous section.

The working area is split in two fields, as shown in Figure 1:

- System Net (SN), shown in the lower part of the window in Figure 1, which represents the intra-phase evolution of an MPS. For instance, depending on the specific MPS, the SN may include the failure/repair behavior of system components, the operations to be carried out within operational phase, the maintenance activities, and so on. SN-type subnets are only allowed to include exponentially distributed and immediate transitions. This constraint is imposed to ensure the existence of a simple and computationally efficient time-dependent analytical solution. Besides that, any structure of the SN sub-model can be considered.

- Phase Net (PhN), shown in the upper part of the window in Figure 1, which represents the execution of the various phases. PhN contains all the deterministic transitions of the overall DSPN model and may as well contain immediate transitions. A token in a place of the PhN model represents a phase being executed, and the firing of a deterministic transition models a phase change. The DSPN of the PhN must possess distinct markings corresponding to different phases. Notice that quite general structures of the PhN sub-model are allowed. In particular, the PhN is not limited to have a linear structure, but it may take a tree or cyclic structure [8, 9, 28, 29].

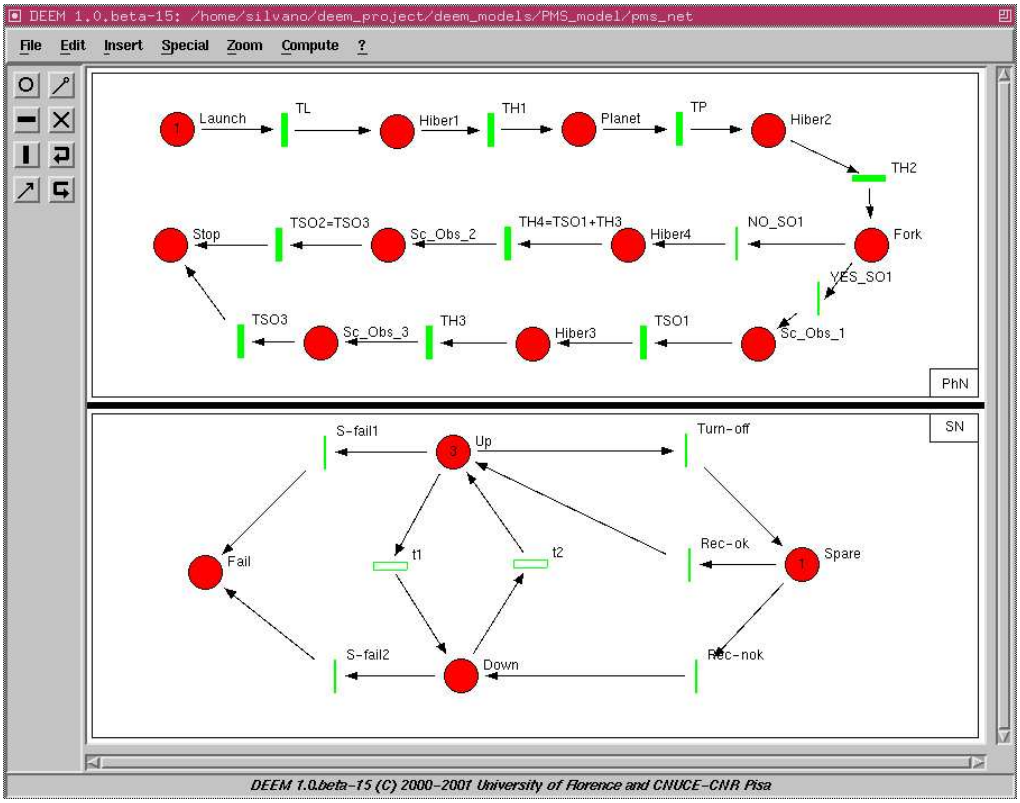

Figure 1: DEEM interface and the DSPN model of the MPS in [29]

Windows associated to each place, such as that in Figure 2 referred to the place Launch, support the definition of the initial marking of the model. The field Capacity specifies the maximum number of tokens for the place, e.g., when there are two tokens in a place with capacity equal to two, then each transition having an arc entering in that place is disabled.

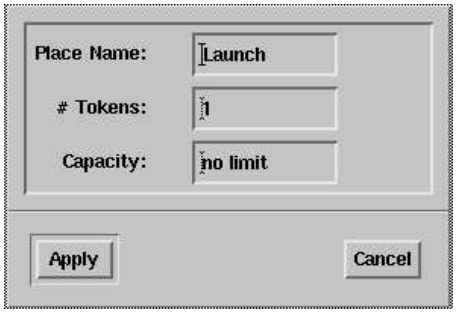

Figure 2: Property window associated to place Launch

Each net can be made dependent on the other one by marking-dependent predicates which modify transition rates, enabling conditions, transition probabilities, multiplicity functions, etc., to model the specific MPS features. A restriction is only imposed on the firing times of the deterministic transitions of the PhN, which are not allowed to be dependent on the marking of the SN.

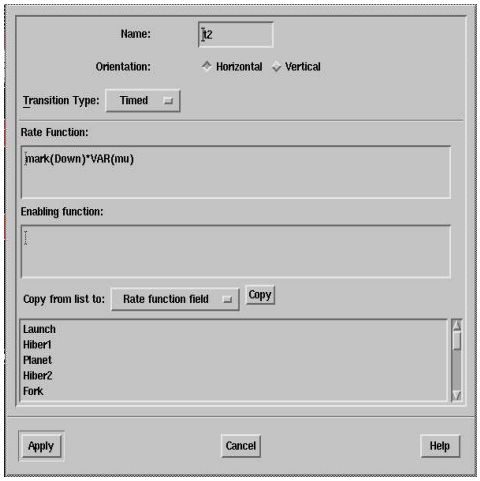

Figure 3: Property window associated to the exponential timed transition t2

Marking dependent attributes of the various objects (transitions and arcs) can be defined through the DEEM property window associated to each object. Figure 3 shows the window associated to transition t2 of the SN of Figure 1, while Figures 4 and 5 show those associated to transition T6 and NO_SO1 respectively of the PhN.

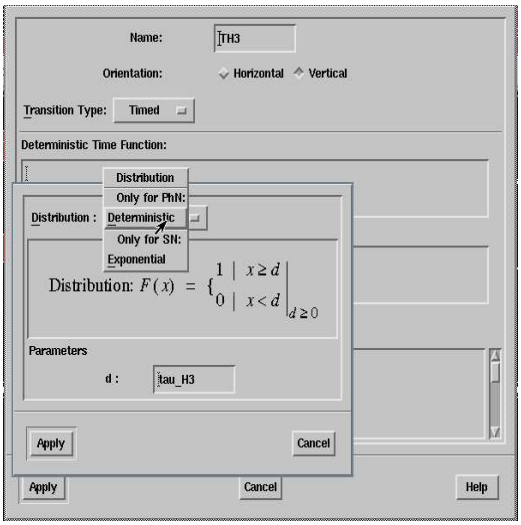

Figure 4: Property window associated to the deterministic timed transition TH3

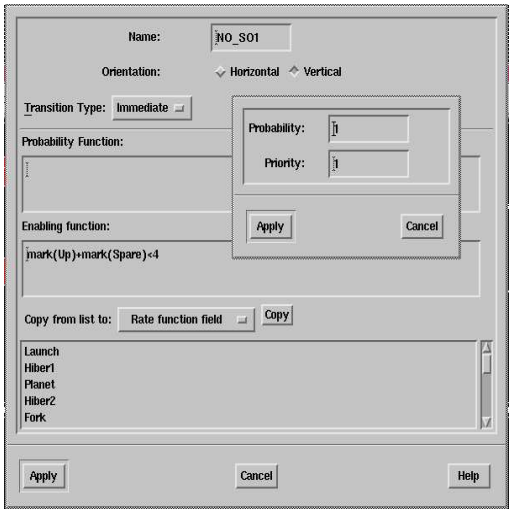

Figure 5: Property window associated to the immediate transition NO_SO1

Notations "MARK(Place_Name)" and "VAR(Parameter_Name)" denote the number of tokens in the place "Place_Name" and the parameter "Parameter_Name", respectively. Other examples of expressions will be shown in Sections 4, when presenting case studies. More information can be obtained from the Help on line of the tool.

Figure 6 shows the window associated to the arc that connects place Up and transition t1.

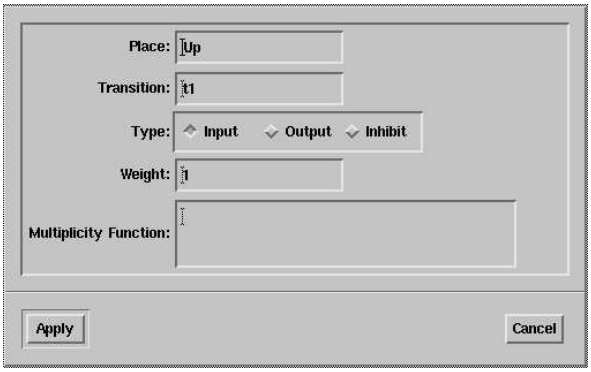

Figure 6: Property window associated to the arc from place Up to transition t1

3.2 Parameters and studies

When building the models, the attributes of objects like times, rates, probabilities or multiplicities, can be expressed through parameters rather than numerical values directly, using the syntax "VAR(Parameter_Name)", as already shown in Figure 3. Then, prior to proceed to the model evaluation, the user has to assign values to the parameters.

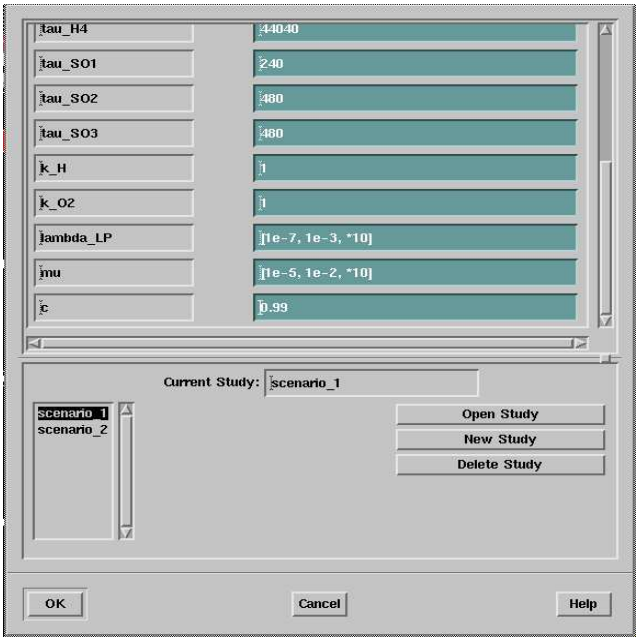

Figure 7: Parameters window with two studies for the example in Figure 1

DEEM automatically builds a parameters table collecting all the symbols defined in the model; this table is made accessible through the command "Parameters" in the Menu "Compute", generating the window illustrated in Figure 7. For each study, represented by a column in the table, parameters are also allowed to take a range or a set of values; in the current version, this feature is restricted to two parameters only. This way, a sensitivity analysis can be performed around the values of impacting parameters, thus generating a family of curves from the evaluation of a single study.

3.3 Dependability measures and transient analysis

Central to the dependability evaluation process is the definition of specific measures of system behavior that are of interest to a user. Useful work to formally categorize measures of system behavior based on "reward models" are [22, 31]. A reward model consists of a stochastic process and a "reward structure". Following the same approach adopted in [31], the specific dependability measures of interest for the MPS evaluation are defined through a reward structure that quantifies behaviors at the DSPN level, instead of the state level. They are based on a general mechanism of marking-dependent reward functions. Informally, a reward structure [22, 31] consists of a rate or impulse reward that are associated with the time spent in a state or with some event of the process, respectively. Rate rewards can be defined as arbitrary function of the model marking. Measures are further distinguished in three categories, in accordance with the time interval they depend on: instant-of-time, interval-of-time and time-averaged interval-of-time [31]. The instant-of-time measure represents the reward that is associated with the status of the modeled system at a particular time. The interval-of-time and time-averaged interval-of-time

measures represent the total and time-averaged reward accumulated during some interval of time, respectively.

In DEEM, dependability measures can be defined by a reward function "IF(<*Predicate*>)THEN(<*Reward*>)ELSE(<*Reward*>)" and an analysis type flag (for instantaneous, cumulated and timed-averaged analysis), as shown at the top of Figure 8. This window is selected through the command "Measures" in the Menu "Compute". <Reward> denotes an expression defining the rate or impulse reward associated to the state (number of tokens in places) or event (firing of a transition), respectively, defined by the Boolean expression <Predicate>. Composed measures can be also defined, as a function of the evaluated reward-based measures, referred with the notation "FUN(<Measure Name>)".

Once measures are defined, the transient evaluation can be launched on the selected study. Results are collected in two output files, in formats compatible with spreadsheet programs and gnuplot respectively, so as to produce plots or tables of the dependability measures.

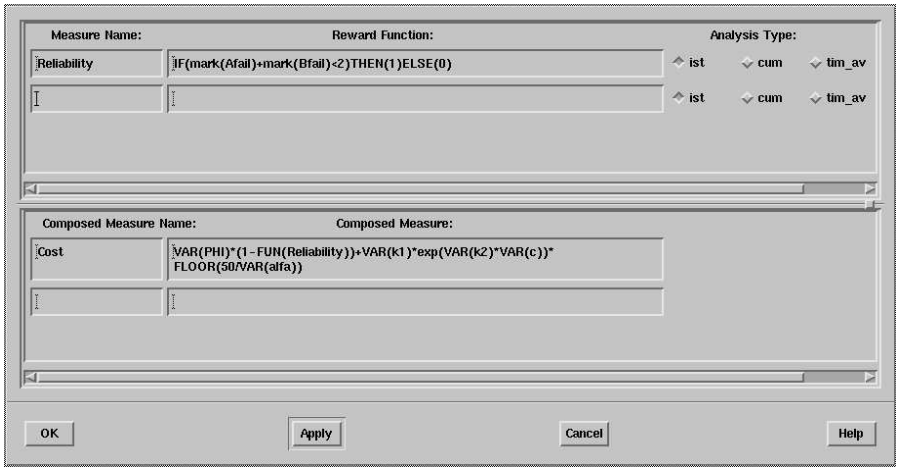

Figure 8: Measures window

Finally, DEEM permits to save in a file the state distribution of the net at the end of the transient analysis. From this file, the initial state distribution used for a next transient analysis may be loaded. In this way, it is possible to separately perform evaluations of the same model in a sequence of different periods of time, where the parameters relative to each period depend on the results of the evaluation at previous periods of time.

3.4 The solution algorithm

This section describes in some details the solution algorithm implemented in DEEM. It is organized in two parts: first, the steps of the algorithm are presented, then a discussion on the efficiency of the proposed algorithm follows.

3.4.1 Algorithm description

To compute the defined dependability figures, DEEM computes the probability vector $P(t)$ of each marking in SN at time t. $P(t)$ is obtainable from the transient probability matrix $V(t)$, through the equation $P(t) = P_0 \cdot V(t)$, where P_0 is the initial probability vector of the DSPN.

Equation (1) allows to evaluate $V(t)$ through the separate analysis of the various alternative paths composing the mission, and only requires the computation of the matrix exponentials $e^{Q_i \tau}$, and the branching probability matrices $\Delta_{i,j}$, $i, j = 1, 2, \mathsf{K}, n$, which can be automatically obtained once the reachability graph is generated. The solution of the DSPN model is thus reduced to the cheaper problem of solving a set of homogeneous, time-continuous smaller Markov chains.

To compute $P(t)$ and then the dependability figures of the system, the solution engine of DEEM starts taking as input the DSPN model and its initial probability vector P_0, and performs the following steps:

I. Builds RGP, the reachability graph of the PhN sub-model. This graph has exactly one stable marking $\dot{m}_i$ for each phase $i = 1, 2, ..., n$ the MPS may perform.

II. Calls *deem_solver(*$1, P_0, 0$*)*.

deem_solver($i, P_i^{init}, t_i^{init}$ *)* implements a recursive algorithm, whose steps are:

1. To build the reachability graph $RGS(\dot{m}_i)$ of the whole DSPN model when marking $\dot{m}_i$ is the only one permitted for the PhN. From $RGS(\dot{m}_i)$ the transition rate matrix Q_i of the continuous-time Markov chain describing the evolution of the DSPN during the execution of phase i is obtained.

2. If phase i is the last one, or $t \le t_i^{init} + \tau_i$ then to compute the transient state probability vector $P_i(t) = P_i^{init} e^{Q_i \left(t - t_i^{init} \right)}$ and return, else to continue at step (3).

3. To compute the transient state probability vector $P_i = P_i^{init} e^{Q_i \tau_i}$.

4. To build the reachability graph $RGS(\dot{m}_i, next(\dot{m}_i))$, where $next(\dot{m}_i) = \{\dot{m}_{j_1}, ..., \dot{m}_{j_m}\}$, of the whole DSPN model, when the initial marking of the PhN is $\dot{m}_i$, and transition t_i^{Det} is the only deterministic one allowed to fire. Each marking $\dot{m}_{j_h}$ is reachable from $\dot{m}_i$ through the firing of some instantaneous transition next to the firing of t_i^{Det}.

5. For each stable marking $\dot{m}_{j_h}$ (phase j_h), to perform the following steps:

   5.1. From $RGS(\dot{m}_i, next(\dot{m}_i))$, to obtain the branching probability matrix Δ_{i,j_h} for the transition from phase i to phase j_h.

   5.2. To compute the initial state probability vector of the phase j_h: $P_{j_h}^{init} = P_i \Delta_{i,j_h}$.

   5.3. To call *deem_solver*($j_h, P_{j_h}^{init}, t_i^{init} + \tau_i$).

To evaluate the specific dependability measure of interest for the MPS, based on reward structures, DEEM operates on P_i according to the standard computation algorithms [13].

It is worthwhile observing that: i) the state space of the MRGP process does not need to be generated and handled as a whole; ii) transition rate matrices Q_i can be generated separately one from another; iii) to build matrices Δ_{i,j_h}, the generation of the reachability graph for consecutive phases is required.

The realization of all the steps described above only require well-known algorithms [10, 30]; in fact they have been already implemented in most of the tools for the automated evaluation of dependability. The generation of the reachability graphs and their reduction to a continuous-time Markov chains (CTMC) are obtained using the sequential version of the algorithm SRGG proposed in [4] and implemented in the tool PANDA [3], by eliminating on-the-fly all vanishing states (all states with zero sojourn time). This way, CTMC is built directly from the SPN without permanently storing all the vanishing states. Before feeding SRGG with the DSPN model, the transitions of the PhN are modified in exponentially distributed. In particular, to generate the RGP in step I, SRGG takes as input the DSPN model with the enabling predicates of the transitions of the SN modified set to FALSE. To build the $RGS(\dot{m}_i)$ in step 1, SRGG takes as input the DSPN model with the enabling predicate of the transitions of the PhN modified set to FALSE and the initial marking of the PhN modified in the one corresponding to phase i. Finally, to generate the $RGS(\dot{m}_i, next(\dot{m}_i))$ in step 4, the SRGG takes as input the DSPN model modified as follows: the initial marking of the PhN is that of the phase i; the enabling predicate of all the other transitions of PhN is FALSE; and rate 1 is assigned to the transition enabled in phase i. Assuming rate 1, the entries of matrix Δ_{i,j_h} in step 5.1 may be obtained directly from the corresponding values in $RGS(\dot{m}_i, next(\dot{m}_i))$, without normalization. To compute the matrix exponential in steps 2 and 3, the version of the uniformization (or randomization) algorithm in [30] is used.

3.4.2 Evaluation of the algorithm efficiency

For generating the reachability graphs at steps I, 1, 4, the asymptotic computational complexity is given by $O\left(\left(\sum_{h=0}^{m^{\max}} C_{j_h}\right)^2 \log\left(\sum_{h=0}^{m^{\max}} C_{j_h}\right)\right)$, where $C_i = |S_i|$, $j_0 = i$ and $m^{\max} \geq 1$ is the maximum number of phases reachable directly from a phase (n-1 in the worst case), and the asymptotical memory requirements are given by $O\left(\left(\sum_{h=0}^{m^{\max}} C_{j_h}\right)^2\right)$. These costs, dominated by the operations at step 4, can be reduced to $O\left(\sum_{h=1}^{m^{\max}} \left(C_i + C_{j_h}\right)^2 \log\left(C_i + C_{j_h}\right)\right)$ (time complexity) and $O\left(\left(C_i + C_{j_h}\right)^2\right)$ (space complexity) by generating the reachability graph only for the two consecutive phases i and j_h, for each phase j_h reachable from the current phase i. Note that the same memory space can be reused to store the reachability graph relative to different consecutive phases. For the transient solutions at steps 2 and 3, the computational complexity is $O\left(C_i^2 q_i \tau_i\right)$, where q_i is the maximum absolute diagonal entry of Q_i and the memory requirements are $O\left(C_i^2\right)$. For the multiplication operations at steps 2, 3 and 5.2 (repeated at most $m^{\max}$ times) the computational complexity is $O\left(C_i^2 + C_i \sum_{h=1}^{m^{\max}} C_{j_h}\right)$ and the memory requirements are $O\left(C_i^2 + C_i C_{j_h}\right)$. For obtaining the branching probability matrix at step 5.1 (repeated at most $m^{\max}$ times), the computational complexity is $O\left(C_i \sum_{h=1}^{m^{\max}} C_{j_h}\left(C_i + C_{j_h}\right)\right)$ and the memory requirements are $O\left(\left(\sum_{h=0}^{m^{\max}} C_{j_h}\right)^2\right)$.

Therefore, the overall asymptotic computational cost of the DEEM solution algorithm is dominated by the operations at steps 2 and 3 and is given by $O\left(\sum_{i=1}^{n} C_i^2 q_i \tau_i\right)$. The overall asymptotic memory requirements are dominated by the costs at step 4 and is given by $O\left(\left(\sum_{h=0}^{m^{\max}} C_{j_h}\right)^2\right)$, but can be reduced to $O\left(\left(C_i + C_{j_h}\right)^2\right)$.

The complexity orders, for time and space, shown by our algorithm are very good, being comparable to those of the cheapest algorithms in the literature to deal with PMS scenarios [29]. Therefore, the issues posed by the phased-behavior of PMS are completely and efficiently solved by our method. The applicability is only limited by the maximum time required to compute the operations at steps 2 and 3, and by the maximum memory space required for the reachability graph of the consecutive phases computed at step 4.

4 Examples of DEEM applications

In this section we illustrate the use of DEEM in the solution of two representative problems that have been already studied in the literature, but without the support of any automated tool. The purpose of the re-examination here is to show the usage of DEEM on interesting case studies, to better illustrate the advantages the tool offers.

The first problem we deal with is the modeling and evaluation of a Phased Mission System for space applications, which has been studied in [29]. The other consists in the optimization of a Scheduled Maintenance System from dependability and cost viewpoints, a problem which has been previously dealt with in [9, 36]. With respect to PMSs, SMS add a further degree of complexity, in that the long lifetime of a SMS can be partitioned into a set of missions, each of them composed by several phases.

4.1 The case of a Phased Mission System

Consider a space application whose mission alternates operational phases (as launch, planet flyby, scientific observations) with hibernation phases (typically entered to maintain a low level of activity during the navigation). Primarily due to the adverse environmental situations and to the long mission duration, there is a high likelihood that spacecraft components are subject to failures. In order to ensure adequate dependability levels, the system employs a set of N identical processors.

The main characteristics of such PMS, to be properly modeled by exploiting the interaction capabilities between the PhN and SN sub-nets, are the following.

Phase-Triggered Reconfigurations of the SN. The system uses in each phase the number of processors that are actually needed to meet the dependability requirements of the current phase, and keeps the others as cold spares, ready to be employed when performing critical activities. Specifically, hibernation phases employ two active processors, being also able to survive with just one active processor, while operational phases always require three active processors. At the start of each hibernation phase, a reconfiguration takes place and one of the active processor is turned off. Similarly, when a new operational phase starts, one spare must be turned on. A standby processor is fault-free; however, a failure may occur when activating a cold spare with probability 1-c (being c the coverage of the activation procedure).

Phase-Dependent Behavior of the SN. Active processors fail with phase-dependent rates: the failure rate during hibernation phases is less than the failure rate during operational phases. Repair actions are applied to faulty components, based on the nature of the faults affecting the spacecraft. We assume that active processors fail and are repaired independently from each

Table 1: Properties of the timed transitions for the model of Figure 1

A token in a place of the PhN model represents a phase being executed; τ_L, τ_{H1}, ..., τ_{SO2} are the deterministic time duration of the phases; and the firing of a deterministic transition models a phase change. A token in the places *Sc_Obs_2* or *Sc_Obs_3* represent the same Scientific Observation 2, which is replicated along the two possible mission paths. The sequence of phases ends with a token in the *Stop* place, which represents the mission's end. Notice that, without the place *Stop*, the PhN net shows a tree structure, according to the fact that this particular MPS performs a dynamic choice between two distinct mission paths. Transition *t1* in the SN subnet represents the failure of a processor, and transition t2 the repair of a faulty processor. The initial marking of the DSPN is: mark(Launch)=1, mark(Up)=3, mark(Spare)=N-3, and no tokens in the remaining places. The actions performed at each phase

Table 2: Properties of the immediate transitions for the model of Figure 1

4.1.2 The DEEM analytical solution

Estimations of the probability that the system successfully completes its mission, that is the reliability of the PMS at the end of the mission, have been performed by using the transient solver of DEEM. In order to allow direct comparison, the values assumed for the model parameters are the same as those in [29]. The number of system nodes is 4, and fault rates are considered different in the different phases (the fault rate is higher during the most stressing phases such as scientific observations, while it lowers during hibernation phases). After execution, the transient solver of DEEM returns the following file:

```
FILE_NAME: pms_net.study_PMS
NET_NAME: pms_net

STUDY_NAME: study_PMS
```

```
STUDY_variables_setting:
Time    8.853600e+04
tau_L   4.800000e+01
tau_H1  1.752000e+04
tau_P   1.680000e+02
tau_H2  2.628000e+04
tau_H3  4.380000e+04
tau_H4  4.404000e+04
tau_SO1 2.400000e+02
tau_SO2 4.800000e+02
tau_SO3 4.800000e+02
k_H     1.000000e-01
k_O2    1.000000e+01
lambda_LP       1.000000e-05
mu      [1e-5, 1e-2, *10]
c       {0.9, 0.99, 0.999, 0.9999}
```

```
REW_MEASURE_NAME: Reliability
REW_FUNCTION: IF(mark(Fail)=0)THEN(1)ELSE(0)
ANALYSIS_TYPE: INSTANTANEOUS

mu/c            9.000000e-01    9.900000e-01    9.990000e-01    9.999000e-01
1.000000e-05    8.823736e-01    9.628646e-01    9.689930e-01    9.695845e-01
1.000000e-04    9.335368e-01    9.843568e-01    9.874891e-01    9.877822e-01
1.000000e-03    9.434508e-01    9.888337e-01    9.913524e-01    9.915837e-01
1.000000e-02    9.540315e-01    9.946170e-01    9.965105e-01    9.966778e-01
```

The first part of the file summarizes the parameters setting used in the experiment, then the definition of the measure under evaluation follows, and finally the obtained results are listed (in the format directly usable by a spreadsheet program). Note that parameters *mu* and *c* are variable in an interval and a set of values, respectively.

4.2 The case of a Scheduled Maintenance System

Consider a system equipped with two components. Component A is a primary unit providing some functionality to the system, and component B acts as a backup unit for component A, ready to take the role of primary upon A's failure. The system cyclically executes two types of mission: 1) M_1 consisting of a single phase of fixed duration τ_{11}, and 2) M_2 consisting of two phases, having duration τ_{21} and τ_{22}, respectively. The time to failure of component A is exponentially distributed, with parameter λ_{1A} during M_1, and parameter λ_{2A} during M_2, whereas the time to failure of component B is constantly λ_B.

Maintenance actions are undertaken during the SMS lifetime, according to the following schedule. i) a complete maintenance check on the whole system is operated every 100 missions, which restores the system to the initial condition; ii) primary unit A is replaced at the end of each mission, if failed; after its repair, A takes again the role of primary unit; and iii) backup unit B is subject to a partial check at the end of each α pairs of missions. Since after complete checks the system is as a new one, the reliability R needs to be studied in the time interval between two complete checks (i.e., the duration of 100 missions). System failure is defined as the failure of A and B, a possibly catastrophic event leading to relevant economic loss. The probability of system failure is denoted by F, and can be computed as 1-R.

Here, we are interested in optimizing maintenance actions. It is therefore relevant to determine whether it is economically convenient to improve the maintenance (by increasing the number of intermediate checks, or increasing the coverage c of such checks), or to accept the costs associated to a higher system failure. Suppose a cost Φ is paid in case F occurs during the one hundred missions, and a cost φ is paid to perform the partial check of B each α pairs of missions. The cost φ of the check and the coverage c that it provides obviously depend from each other: the higher the target coverage, and the more expensive the check required. We use $\varphi=\varphi(c)$ to denote such dependency. Then, to optimize the SMS with respect to the schedule of the maintenance, one should minimize the following overall expected cost function:

$$Cost = \Phi \cdot Prob[F] + \varphi(c) \cdot \left\lfloor {}^{50}\!/_{\alpha} \right\rfloor = \Phi \cdot (1 - R) + \varphi(c) \cdot \left\lfloor {}^{50}\!/_{\alpha} \right\rfloor \qquad (2)$$

4.2.1 The DEEM model

The DEEM model of the SMS is shown in Figure 9. The defined model is slightly different from that proposed in [9]; however, the modifications do not impact on the results, which therefore remain comparable.

The PhN concisely represents the two types of mission executed by the SMS. A token in place *P1* represents the execution of the single phase of M_1, and the firing of transition *T1* represents the completion of the phase. A token in places *P2* and *P3* enables transitions *T2* and *T3*, modeling the execution of phases 1 and 2 respectively of M_2, and firings of *T1* and *T2* represent completions of those phases. The place *Count* keeps track of the number of missions performed by the SMS. Definitions of the timed and immediate transitions are listed in Tables 3 and 4 respectively. As in the previous case, from the tables it can be observed the dependency of the properties of the transitions from the marking of the two sub-nets. Moreover, it is worthwhile

Table 3: Properties of the timed transitions for the model of Figure 10

The discrete repairs induced by the maintenance actions are modeled by the immediate transitions *Yes_repair_A*, *Yes_repair_B*, *ok_repair* and *nok_repair*. These immediate

Table 4: Properties of the immediate transitions for the model of Figure 10

To explain the evolution and synchronization of the PhN and SN models, suppose that component A correctly works, component B is failed (one token is in Bfail) and the α-th Mission 2 is not yet ended (one token in P2 or P3 and α-1 tokens in Count). As soon as α-th Mission 2 ends, one token reaches places Stop2 enabling transition Tcount and one token reaches Count, which now contains α tokens. At this point, component B is to be repaired. First the transition Yes_repair_B fires (now its enabling function is true and its priority is higher than that of Tcount) and one token reaches place repair_B. Then transitions ok_repair and nok_repair are allowed to fire. Similar is the evolution and synchronization of the submodels in the case that component A fails. Notice that when both components A and B are failed (one token in Afail and Bfail) the Yes_repair_A and Yes_repair_B are permanently disabled and the failure of the mission is modeled.

4.2.2 The DEEM analytical solution

The SMS model just described has been solved through the DEEM transient solution, in order to study the cost function defined by Equation (2). The study has been developed at varying the coverage c of the check on the backup unit B, for a fixed value $\lambda_B = \lambda_{1A}$ of component B failure rate. We assume as unitary the cost Φ of the system failure, and proportionally define the cost function $\varphi(c) = k_1 \cdot e^{k_2 c}$ in a way that a check providing coverage 0.6 requires a cost of $10^{-6}\Phi$, whereas a check with coverage 0.99 costs $10^{-2}\Phi$. This setting is the same as in [9]. The

following file, returned by the transient solver of DEEM, includes the parameters setting, the definition of the evaluated measures and the results of the evaluation.

```
FILE_NAME: sms_net.Cost_Study
NET_NAME: sms_net

STUDY_NAME: Cost_Study

STUDY_variables_setting:
Time     1.500000e+03
tau_11   1.500000e+01
tau_21   5.000000e+00
tau_22   1.000000e+01
lambda_1A        1.000000e-03
lambda_2A        2.000000e-03
lambda_B 1.000000e-03
alfa     {1, 2, 5}
c        {0.6, 0.7, 0.8, 0.9, 0.95, 0.99}
PHI      1.000000e+00
k1       7.017040e-13
k2       2.361630e+01
--------------------------------------------------------------------------------
REW_MEASURE_NAME: Reliability
REW_FUNCTION: IF(mark(Afail)+mark(Bfail)<2)THEN(1)ELSE(0)
ANALYSIS_TYPE: ISTANTANEOUS

alfa/c   6.000000e-01     7.000000e-01     8.000000e-01     9.000000e-01     9.500000e-01     9.900000e-01
1.00e+00 9.118943e-01     9.238274e-01     9.332045e-01     9.407681e-01     9.440268e-01     9.464250e-01
2.00e+00 8.599788e-01     8.790535e-01     8.946035e-01     9.075257e-01     9.132017e-01     9.174220e-01
5.00e+00 7.536677e-01     7.807849e-01     8.045963e-01     8.256808e-01     8.353453e-01     8.426997e-01
--------------------------------------------------------------------------------
COMP_MEASURE_NAME: Cost
COMP_FUNCTION: VAR(PHI)*(1-FUN(Reliability))+VAR(k1)*exp(VAR(k2)*VAR(c))*FLOOR(50/VAR(alfa))

alfa/c   6.000000e-01     7.000000e-01     8.000000e-01     9.000000e-01     9.500000e-01     9.900000e-01
1.00e+00 8.815566e-02     7.670307e-02     7.242233e-02     1.189230e-01     2.503889e-01     5.535962e-01
2.00e+00 1.400462e-01     1.212117e-01     1.082099e-01     1.223198e-01     1.840061e-01     3.325887e-01
5.00e+00 2.463423e-01     2.193212e-01     1.965291e-01     1.862574e-01     2.035378e-01     2.573045e-01
```

Note the inclusion in the file of the definition and evaluation of the reliability measure, since it is necessary to determine the cost function.

4.3 Remarks on the case studies

Comparing the results obtained through DEEM for both case studies with the corresponding ones previously determined in [29] and [29], where the same systems have been modeled and solved by hand, it can be observed that the results are very similar. This reinforces the confidence on the correct definition and implementation of the DEEM analytical solution.

Now, a few comments on the efficiency of DEEM in solving the above examples. The size of the overall model of the PMS example is given by 35 states and 60 non-zero entries of the matrices Q_i, but thanks to the separation of the solution of the various phases the biggest model solved by DEEM was of 5 states and 10 entries of Q_i. The CPU time needed to perform the evaluation task did not exceed the order of few seconds and the total amount of physical memory used did not exceed 2100 Kilobytes, on a Pentium II 350 MHz, 192Mb RAM PC. In the SMS case, the overall model size is given by 600 states and 600 non-zero entries of the matrices Q_i, but again, thanks to the separation of the solution of the various phases operated by DEEM, the biggest model solved was of 4 states and 4 entries of Q_i. The CPU times needed to perform the evaluation of the studies Study_1 and Study_2 were respectively 66 seconds and 100 seconds and the total amount of physical memory used did not exceed 2100 Kilobytes, on a Pentium II 350 MHz, 192Mb RAM PC. From these data, it can be appreciated the efficiency of the DEEM's solution algorithm with respect to non specialized solutions, which approach the model solution in its entireness.

Of course, DEEM's benefits grow with the growing dimension of the analyzed problem, for which the usage of a general-purpose evaluation tool may be easily defeated by the well-known problem of state explosion. Among the others, a problem we have dealt with and where the efficiency of DEEM has been fundamental in allowing the target evaluation activities has been that of the Reactor Protection System (RPS) in use at Westinghouse nuclear plants [16]. For this very critical system, we have modeled and analyzed scheduled maintenance actions, which have to be executed on-line without interrupting the service provided. The RPS entire model is of the order of one million of states. However, the biggest model solved by DEEM was of 4096 states and the time needed to perform a single study did not exceed, on average, the order of few tens of minutes on a Pentium III 500 MHz, 128 Mb RAM PC. The dimension of this problem would have been hardly tractable by other evaluation tools, unless performing simplifications on the system representation, which of course would have turned out in less accurate evaluation results.

5 Related Work

Some considerations on related work are here drawn. Concerning the methodology for modeling MPS using DSPN which is followed in DEEM, a wide treatment of the related literature has been already carried on in [27, 28], where such methodology has been originally developed.

Among the existing tools for dependability modeling and evaluation, the EHARP tool [34] (an extension of the HARP tool) implements the methodology proposed by Somani *et al.*, which is specifically designed for SMS scenarios. Some further extensions of EHARP for the SMS problem were introduced by Twigg *et al.* in [36]. The EHARP tool is based on a separate Markov chain modeling of the SMS inside the various phases, an approach that is able to effectively master the complexity and the computational cost of the analysis. However, as carefully explained in [28, 29], this separate Markov based modeling approach requires a relevant amount of user-assistance to correctly model the dependencies among successive phases.

Concerning the solution algorithm implemented in DEEM, we mention that, to numerically evaluate the transient state probability vector $P(t)$ of the DSPN model, different approaches, algorithms and tools could be considered [11, 12, 17-19, 21, 24, 25, 37]. A general-purpose transient solver for DSPNs, such as TimeNET, can be used for this purpose [20]. TimeNET provides many of the modeling features available under the SRN paradigm, and is able to support our modeling methodology. Actually, this tool implements three different transient solution techniques: a general solution and two variants which optimize on time and memory consumption but restricted to specific (and therefore restricted) DSPN structures [20, 21]. Only PMS models belonging to the class of DSPN in which a sequence of deterministic transitions fire periodically at previously known instants of time, are solvable by TimeNET using an efficient method having the same computational complexity as the DEEM's algorithm and memory requirements given by $O\left(\sum_{i=1}^{n} C_i^{\,2}\right)$ [19, 20]. The PMS models dealt with in Section 4, do not belong to such class of DSPN; using TimeNET for their modeling and solution imply applying the general transient solution algorithm (adopted for deterministic transitions firing at random time instants). In such method, the technique of the discretization of the continuous variables is used for the numerical solution of a system, consisting of as many differential equations as the number of non-vanishing markings. Of course, in this case no advantage is obtained from the particular structure of the PMS model and the asymptotic computational cost needed for the solution is higher, being $O\left(\sum_{i=1}^{n} C_i^{\,2} q_i \tau_i j_{\max}^{i}\right)$, where $j_{\max}^{i}$ denotes the number of steps required for the discretization of the CDF of the firing time of the non-exponential transitions.

References

[1] M. Ajmone Marsan and G. Chiola, "On Petri Nets with Deterministic and Exponentially Distributed Firing Times," in "Lecture Notes in Computer Science 266", Ed., Springer-Verlag, 1987, pp. 132-145.

[2] M. Alam and U. M. Al-Saggaf, "Quantitative Reliability Evaluation of Reparaible Phased-Mission Systems Using Markov Approach," IEEE Transactions on Reliability, Vol. R-35, pp. 498-503, 1986.

[3] S. Allmaier and S. Dalibor, "PANDA - Petri Net Analysis and Design Assistant," in Proc. Performance TOOLS'97, Saint Malo, France, 1997.

[4] S.C. Allmaier, M. Kowarschik and G. Horton, "State Space Construction and Steady-State Solution of GSPNs on a Shared-Memory Multiprocessor," in Proc. Seventh International Workshop on Petri Nets and Performance Models, Saint Malo, France, 1997, pp. 112-121.

[5] B.E. Aupperle, J.F. Meyer and L. Wei, "Evaluation of Fault-Tolerant Systems with Non-Homogeneous Workloads," in Proc. 19th IEEE Fault Tolerant Computing Symposium (FTCS-19), 1989, pp. 159-166.

[6] C. Beounes, M. Aguera, J. Arlat, C. Bourdeau, J.-E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell and P. Spiesser, "SURF-2: a Program for Dependability Evaluation of Complex Hardware and Software Systems," in Proc. IEEE FTCS'23, Fault-Tolerant Computing Symposium, Toulouse, France, 1993, pp. 668-673.

[7] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli and F. Sandrini, "DEEM: a Tool for the Dependability Modeling and Evaluation of Multiple Phased Systems," in Proc. DSN200 Int. Conference on Dependable Systems and Networks (FTCS-30 and DCCA-8), New York, USA, 2000, pp. 231 - 236.

[8] A. Bondavalli, I. Mura and M. Nelli, "Analytical Modelling and Evaluation of Phased-Mission Systems for Space Applications," in Proc. IEEE High Assurance System Engineering Workshop (HASE'97), Bethesda Maryland, USA, 1997, pp. 85 - 91.

[9] A. Bondavalli, I. Mura and K. S. Trivedi, "Dependability Modelling and Sensitivity Analysis of Scheduled Maintenance Systems," in Proc. EDCC-3 European Dependable Computing Conference, Prague, Czech Republic - September 15-17, 1999, 1999, pp. 7 - 23.

[10] G. Chiola, "Compiling Techniques for the Analysis of Stochastic Petri Nets," in Proc. Fourth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Palma de Mallorca, Spain, 1988, pp. 11-24.

[11] H. Choi, V.G. Kulkarni and K.S. Trivedi, "Transient Analysis of Deterministic and Stochastic Petri Nets.," in Proc. 14th International Conference on Application and Theory of Petri Nets, Chicago Illinois, USA, 1993, pp. 166-185.

[12] G. Ciardo and G. Li, "Efficient Approximate Transient Analysis for a Class of Deterministic and Stochastic Petri Nets," in Proc. IEEE International Computer Performance and Dependability Symposium (IPDS'98), Durham, NC, USA, 1998, pp. 34-43.

[13] G. Ciardo, J. Muppala and K. S. Trivedi, "On the Solution of GSPN Reward Models," Performance Evaluation, Vol. 12, pp. 237-254, 1991.

[14] G. Ciardo, J. Muppala and K.S. Trivedi, "SPNP: Stochastic Petri Net Package," in Proc. International Conference on Petri Nets and Performance Models, Kyoto, Japan, 1989, pp. 142-151.

[15] J. B. Dugan, "Automated Analysis of Phased-Mission Reliability," IEEE Transaction on Reliability, Vol. 40, pp. 45-52, 1991.

[16] R. Filippini and A. Bondavalli, "Modeling and Analysis of a Scheduled Maintenace System: a DSPN Approach," CNUCE-CNR Technical Report B4-2001-017, November 2001.

[17] R. German, "Transient Analysis of Deterministic and Stochastic Petri Nets by Method of Supplementary Variables.," in Proc. IEEE Int. Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '95), Durham, NC, USA, 1995, pp. 394-398.

[18] R. German, C. Kelling, A. Zimmermann and G. Hommel, "TimeNET: a Toolkit for Evaluating Non-Markovian Stochastic Petri Nets.," Performance Evaluation, Vol. 24, pp. 1995.

[19] R. German, D. Logothetis and K.S. Trivedi, "Transient Analysis of Markov Regenerative Stochastic Petri Nets: a Comparison of Approaches," in Proc. Sixth International Workshop on Petri Nets and Performance Models, Durham, North Carolina, USA, 1995, pp. 103-112.

[20] R. German and J. Mitzlaff, "Transient Analysis of Deterministic and Stochastic Petri Nets with TimeNET," in Proc. Joint Conferences 8th Int. Conf. on Modelling Techniques and Tools for Performance Evaluation and 8th GI/ITG Conf. on Measuring, Modelling and Evaluating Computing and Communication Systems, 1995, pp. 209-223.

[21] A. Heindl and R. German, "Fourth-Order Algorithm with Automatic Stepsize Control for the Transient Analysis of DSPNs," IEEE Transactions on Software Engineering (TSE), Vol. 25, pp. 194-206A, 1999.

[22] R. A. Howard, "Dynamic Probabilistic Systems," New York, Wiley, 1971.

[23] C. Lindemann, "Performance Modeling Using Deterministic and Stochastic Petri Nets," John Wiley & Sons, 1998.

[24] C. Lindemann, A. Ruys and A. Thümmler, "The DSPNexpress 2.000 Performace and Dependability Modeling Environment," in Proc. FTCS-29 Fault-Tolerant Computing Symposium, Madison, Wisconsin, USA, 1999, pp. 228-231.

[25] C. Lindemann and A. Thümmler, "Transient Analysis of Deterministic and Stochastic Petri Nets with Concurrent Deterministic Transitions," Performance Evaluation, Vol. 36 &37, pp. 35-54, 1999.

[26] J.F. Meyer, D.G. Furchgott and L.T. Wu, "Performability Evaluation of the SIFT Computer," in Proc. IEEE FTCS'79 Fault-Tolerant Computing Symposium, June20-22, Madison, Wisconsin, USA, 1979, pp. 43-50.

[27] I. Mura and A. Bondavalli, "Hierarchical Modelling and Evaluation of Phased-Mission Systems," IEEE Transactions on Reliability, Vol. 48, pp. 360-368, 1999.

[28] I. Mura and A. Bondavalli, "Markov Regenerative Stochastic Petri Nets to Model and Evaluate the Dependability of Phased Missions," IEEE Transactions on Computers, Vol. 50, pp. 1337-1351, 2001.

[29] I. Mura, A. Bondavalli, X. Zang and K.S. Trivedi, "Dependability Modelling and Evaluation of Phased Mission Systems: a DSPN Approach," in Proc. DCCA-7 - 7th IFIP Int. Conference on Dependable Computing for Critical Applications, San Jose, CA, USA, 1999, pp. 299-318.

[30] A. Reibman and K.S. Trivedi, "Numerical Transient Analysis of Markov Models," Computers and Operations Research, Vol. 15, pp. 9-36, 1988.

[31] W. H. Sanders and J. F. Meyer, "A unified Apporach for Specifying Measures of Performance, Dependability, and Performability," in "Dependable Computing for Critical Applications", Ed., Springer-Verlag, 1991.

[32] W. H. Sanders, W. D. Obal II, M. A. Qureshi and F. K. Widjanarko, "The UltraSAN Modeling Environment," Performance Evaluation, Vol. 21, pp. 1995.

[33] M. Smotherman and K. Zemoudeh, "A Non-Homogeneous Markov Model for Phased-Mission Reliability Analysis," IEEE Transactions on Reliability, Vol. 38, pp. 585-590, 1989.

[34] A. K. Somani, J. A. Ritcey and S. H. L. Au, "Computationally-Efficent Phased-Mission Reliability Analysis for Systems with Variable Configurations," IEEE Transactions on Reliability, Vol. 41, pp. 504-511, 1992.

[35] A. K. Somani and K. S. Trivedi, "Phased-Mission Systems Using Boolean Algebraic Methods," Performance Evaluation Review, Vol. pp. 98-107, 1994.

[36] D. W. Twigg, A. V. Ramesh, U. R. Sandadi, A. Ananda and T. Sharma, "Reliability Computation of Systems that Operate in Multiple Phases and Multiple Missions," Boeing Commercial Airplane Group

[37] A. Zimmermann, R. German, J. Freiheit and G. Hommel, "TimeNET 3.0 Tool Description," in Proc. Petri Nets and Performance Models (PNPM'99), Zaragoza, Spanien, 1999.