MODELLING AND SIMULATING A BENCHMARK ON DYNAMIC RELIABILITY AS A STOCHASTIC ACTIVITY NETWORK

Daniele Codetta-Raiteri[(a)]

[(a)]Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria, Italy

[(a)]raiteri@mfn.unipmn.it

ABSTRACT

Several versions of a benchmark on dynamic reliability taken from the literature are examined: each version deals with particular aspects such as state dependent failure rates, failures on demand, and the repair of components. The benchmark was modelled in the past, using two types of Petri Nets; in this paper, we exploit another Petri Net based modelling formalism called Stochastic Activity Network (SAN). This allows a more compact model of the system by exploiting input and output gates, together with the possibility to represent float variables by means of extended places. The SAN model of the system undergoes simulation in order to compute the system unreliability: the results are coherent with those obtained with other methods, and this confirms that Petri Net based models can be a valid approach to dynamic reliability evaluation.

Keywords: dynamic reliability, benchmark, modelling, simulation, Stochastic Activity Network, Petri Net.

1. INTRODUCTION

We talk about safety critical systems when their incorrect behaviour may cause undesirable consequences to the system itself, the operators, the population or the environment. This definition fits categories of systems such as industrial production plants, electric power plants, and transportation systems. *Dependability* is a fundamental requirement for this class of systems. The Dependability level of a system is the degree of confidence that the system will provide its service correctly during its life cycle.

There are two main methods to evaluate the Dependability: the *Measurement-based* method and the *Model-based* method. The first one requires the observation of the behaviour of the physical objects composing the system, in the operational environment. This method is more believable, but it may be impractical or too expensive. Therefore the model-based method is preferable and consists of the construction of a model representing the behaviour of the system in terms of modelling primitives defined in a formalism. The model of the system must be a convenient abstraction of the system; this means that the level of accuracy of the model must be high enough to represent correctly the aspects of the system behaviour which are relevant to Dependability evaluation. The mechanisms that lead to the failure of a technological item are very complex and depend on physical, technical, human and environmental factors which may not obey deterministic laws; so, the model-based method follows the probabilistic approach.

The concept of Dependability is quite general; in order to quantify the Dependability, we need particular measures: the *Reliability R(t)* of an item (component or system) is the probability that the item performs the required function in the time interval *(0, t)*, given the stress and environmental conditions in which the item operates; the *Unreliability U(t)=1-R(t)* is the probability that the item is in the failed state at time *t* (Sahner, Trivedi, and Puliafito 1996).

We talk about dynamic reliability (Marseguerra, Zio, Devooght, and Labeau 1998) when the reliability parameters of the system change according to the current configuration of the system. For instance, the failure rate of a component may be expressed as a function of one or more variables describing the current behaviour or state of the system. In dynamic reliability, considering only the combinations of failure events is not sufficient to evaluate the system (un)reliability, but we actually have to take into account the complete behaviour of the system. This means modelling the normal functioning of the system, the occurrence of failure events and their effect on the system functions. For this reason, combinatorial models such as *Fault Trees* and *Reliability Block Diagrams* (Sahner, Trivedi, and Puliafito 1996) are not suitable to deal with cases of dynamic reliability because such kinds of model can only represent combinations of component failure events. Their extensions such as *Dynamic Fault Trees* (Dugan, Bavuso, and Boyd 1992) and *Dynamic Reliability Block Diagrams* (Distefano and Xing 2006) introduce the possibility to represent dependencies among the events, but they still only focus on the failure propagation ignoring the other aspects of the system behaviour.

Such aspects could be represented instead by means of state space based models, such as *Markov Chains* and *Stochastic Petri Nets* (Sahner, Trivedi, and Puliafito 1996), but their use typically leads to the state space explosion because the complete dynamics of the system has to be modelled. Therefore the model analysis

becomes impractical because of the high computing cost (and time). For these reasons, dynamic reliability cases are typically evaluated by means of simulation.

In this paper, we take into account a benchmark on dynamic reliability taken from the literature (Marseguerra and Zio 1996). The system consists of a tank containing some liquid whose level is influenced by two pumps and one valve managed by a controller, with the aim of avoiding the failure of the system occurring in case of the dry out or the overflow of the liquid. Such events are consequences of the pumps or valve failure because in such condition the components ignore the orders coming from the controller. The dry out or the overflow does not occur as soon as a particular combination of component failures occurs, but such basic failures may influence the liquid level, possibly leading the system to the failure after that some time has elapsed or another event has happened. Because of this, not only the component failure combinations have to be modelled, but also any variation in the liquid level caused by the components action or failure.

In Marseguerra and Zio (1996), several versions of the benchmark are proposed: the initial case of state independent failure rates (that we call Version 1), the case of state dependent failure rates (Version 2), the case with possible failure on demand of the controller (Version 3), the case with repairable components (Version 4), and finally the case with temperature dependent failure rates (Version 5). All the versions are described in Sec. 3. In Sec. 5, each version of the system is modelled as a *Stochastic Activity Network* (SAN) (Sanders and Meyer 2001), a particular form of Stochastic Petri Net; the SAN formalism is described in Sec. 4. The SAN models are designed and simulated by means of the *Möbius* tool (Deavours, Clark, Courtney, Daly, Derisavi, Doyle, Sanders, and Webster 2002); the aim is to compute the system unreliability in each version of the benchmark (Sec. 6). The advantages of SAN with respect to other forms of Petri Nets (Sec. 2) are presented in Sec. 7.

2. RELATED WORK

In Marseguerra and Zio (1996) the unreliability of the system in Versions 1, 2, and 3 is evaluated first in an analytical way by computing the probabilities of the minimal cut sets of component failure events leading to the dry out or the overflow. Then, the system unreliability is evaluated by means of Monte Carlo simulation. The cut set analysis only considers the combinations of events, while the Monte Carlo simulation deals with the complete dynamics of the system: therefore there is a relevant difference between the unreliability values returned by the two approaches, in particular in Versions 2 and 3. Such difference highlights the necessity to take into account the complete behaviour in order to evaluate the system in an accurate way. Versions 4 and 5 are only evaluated by means of Monte Carlo simulation in Marseguerra and Zio (1996).

In Codetta and Bobbio (2005a), Versions from 1 to 4 have been modelled as *Generalized Stochastic Petri Nets* (GSPN) (Ajmone, Balbo, Conte, Donatelli, Franceschinis 1995) by means of the *GreatSPN* tool (Chiola, Franceschinis, Gaeta, and Ribaudo 1995). The GSPN model can undergo analysis, but this requires the liquid level to be discretized in several intermediate integer levels. This is because in GSPN models, only discrete variables can be represented as the number of tokens (*marking*) inside places. The number of such intermediate levels must not be high; otherwise the state space dimensions may explode, with the consequent increase of the computing cost. Moreover, in the GSPN model, some deterministic timed events such as the action of the pumps or the valve on the liquid level, have to be approximated by stochastic events, in order to allow the model analysis. So, despite of the advantages given by the model analysis instead of simulation, the GSPN model suffers from some approximation about the liquid level and its variations during the time.

In Codetta and Bobbio (2005a), Versions from 1 to 4 are modelled also as *Fluid Stochastic Petri Net* (FSPN) (Gribaudo, Sereno, Horvath, and Bobbio 2001), a particular form of Petri Net including fluid places containing a continuous amount of fluid instead of a discrete number of tokens. Fluid places directly represent continuous variables, such as the liquid level or temperature. FSPN models are typically simulated. This can be done by means of the *FSPNedit* tool (Gribaudo 2001).

Version 5 of the benchmark could not be modelled as a GSPN because the temperature would have been approximated in several intermediate integer values leading to an unacceptable approximation of the current temperature, and consequently to the approximation of the current failure rates. Besides this, the expression of the failure rate as a function of the liquid temperature (Eq. 3 in Sec. 3) cannot be represented in the GSPN model. Moreover, the combination of the possible temperature values together with the possible values of the other parameters describing the system state, would have led to the explosion of the underlying state space dimensions. For this reason, in Codetta and Bobbio (2005b), Version 5 of the system has been modelled and simulated only as a FSPN.

In this paper, the benchmark is modelled and simulated using the SAN formalism. The SAN models can undergo analysis as well, but to this aim, the deterministic activities (transitions) have to be replaced by stochastic activities reducing the accuracy of the model with respect to the real behaviour of the system.

3. THE BENCHMARK

The system (Fig. 1.a) is composed by a tank containing some liquid, two pumps (P1 and P2) to fill the tank, one valve (V) to remove liquid from the tank, and a controller (C) monitoring the liquid level (H) and acting on P1, P2 and V.

Initially H is equal to 0, with P1 and V in state ON, and P2 in state OFF; since P1, P2 and V have the same level variation rate (Q=0.6 *m/h*), the liquid level does not change while the initial configuration holds. The cause of a variation of H is the occurrence of a failure of one of the components consisting of turning to the state Stuck-ON or Stuck-OFF. The time to failure is random and obeys the negative exponential distribution; the failure rate (Tab. 1) does not depend on the current state of the component, so the effect of the failure is the stuck condition, while the state transitions toward the Stuck-ON or the Stuck-OFF state are uniformly distributed (Fig. 2.a).

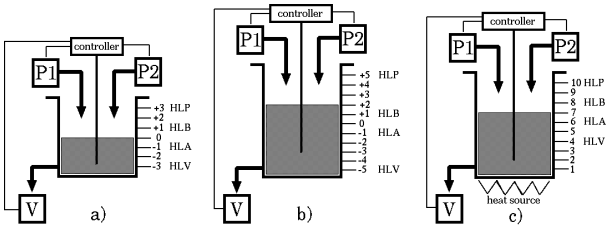

Figure 1. The System Schemes in Versions 1, 2, 3 (a), in Version 4 (b), in Version 5 (c)

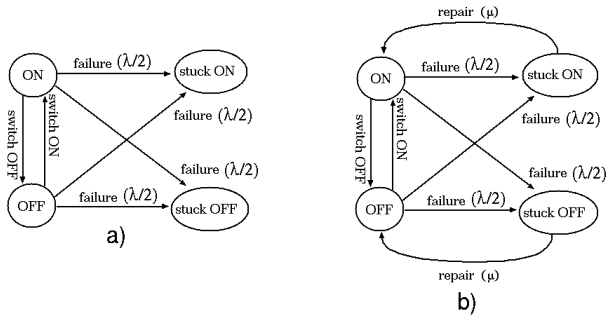

Figure 2: The States of P1, P2 and V in Versions 1, 3, 5 (a). The States of P1, P2 and V in Version 4 (b)

Tab. 2 shows how H changes with respect to the current configuration of the component states; the controller believes that the system is correctly functioning while H is inside the region between the levels HLA (-1 *m*) and HLB (+1 *m*). If H exceeds HLB, the controller orders both pumps to switch OFF, and the valve to switch ON, with the aim of decreasing H (Tab. 3) and avoiding the liquid overflow; this event occurs when H exceeds the level HLP (+3 *m*). If a component is stuck, it does not obey the controller order and maintains its current state.

The other undesired situation is the tank dry out; this happens when H is below HLV (-3 *m*); if H goes below HLA, the controller orders both pumps to switch ON, and the valve to switch OFF, with the aim of increasing H (Tab. 3) and avoiding the dry out.

The failure of the whole system happens when the dry out or the overflow occurs.

We denote such configuration of the system as **Version 1**. In this paper, we deal with several versions of the benchmark, still proposed in Marseguerra and Zio (1996).

Table 1: Failure Rates in Versions 1, 3, 4

component	failure rate (λ)
P1	0.004566 h^{-1}
P2	0.005714 h^{-1}
V	0.003125 h^{-1}

Table 2: The Level Variation in each State Configuration

configuration	P1	P2	V	effect on L
1	ON	OFF	OFF	↑
2	ON	ON	OFF	↑↑
3	ON	OFF	ON	=
4	ON	ON	ON	↑
5	OFF	OFF	OFF	=
6	OFF	ON	OFF	↑
7	OFF	OFF	ON	↓
8	OFF	ON	ON	=

Table 3: Control Boundaries and Laws

boundary	P1	P2	V
H < HLA	ON	ON	OFF
H > HLB	OFF	OFF	ON

3.1. Version 2: state dependent failure rates

In this version, the failure rate of a component changes according to its current state and the state reached as a consequence of the failure (Tab. 4).

Table 4: Failure Rates for each Component in each State, in Version 2

component	from	to	failure rate (λ)
P1	ON	Stuck-ON	0.004566/2 h^{-1}
P1	ON	Stuck-OFF	0.004566/2 h^{-1}
P1	OFF	Stuck-ON	0.045662 h^{-1}
P1	OFF	Stuck-OFF	0.456621 h^{-1}
P2	ON	Stuck-ON	0.057142 h^{-1}
P2	ON	Stuck-OFF	0.571429 h^{-1}
P2	OFF	Stuck-ON	0.005714/2 h^{-1}
P2	OFF	Stuck-OFF	0.005714/2 h^{-1}
V	ON	Stuck-ON	0.003125/2 h^{-1}
V	ON	Stuck-OFF	0.003125/2 h^{-1}
V	OFF	Stuck-ON	0.031250 h^{-1}
V	OFF	Stuck-OFF	0.312500 h^{-1}

3.2. Version 3: controller failure on demand

Here, the controller has a probability of failure on demand equal to 0.1. This means that each time H exceeds the region of correct functioning (HLA<H<HLB), the controller may not send the command to P1, P2 and V, so they maintain their current state.

3.3. Version 4: repairable components

In this version, a stuck (failed) component can be repaired during the grace period which begins when the region of correct functioning is exceeded for the first time, and ends when the dry out or the overflow occurs. The time to repair of a component is a random variable obeying the negative exponential distribution with the repair rate equal to 0.2 h^{-1}. The effect of the repair

consists of removing the stuck condition of a component (Fig. 2.b). As soon as the repair is completed, the component is set to the state ON or OFF if H is currently out of the region of correct functioning (Tab. 3). Moreover, the repaired component can respond to future orders from the controller, changing its state again if necessary. After the repair, a component may fail and undergo repair again.

In this version of the benchmark, HLV and HLP are set to -5 *m* and +5 *m* respectively (Fig. 1.b).

3.4. Version 5: temperature dependent failure rates

The current temperature (T) of the liquid in the tank is taken into account, and T influences the failure rate of the components P1, P2 and V. A heat source increases T according to the heating power $w = 1 m°C/h$. There is no heat released outside the tank, and the heat is uniformly distributed on the liquid. The initial temperature of the liquid inside the tank is 15.6667°*C*; the temperature of the liquid introduced in the tank by the pumps is $T_{in} = 15°C$, and it gets mixed instantaneously with the liquid in the tank.

The level variation rate for P1, P2 and V is now Q=1.5 *m/h*. Assuming that a pump is activated at time t_0 and is still active at time $t > t_0$, we use the Eq. 1 and 2 to provide the liquid level and temperature respectively, at time $t > t_0$. In Eq. 1, L_0 is the liquid level at time t_0; in Eq. 2, T_0 is the liquid temperature at time t_0.

The failure rates of P1, P2 and V are temperature dependent according to Eq. 3 where λ_0 is the failure rate of the component for a temperature equal to 20°*C* (Tab. 5). Besides the dry out and the overflow, another condition determines the failure of the system: T reaches 100°*C*.

The initial level of the liquid in the tank is 7 *m*; HLA and HLB are set to 6 *m* and 8 *m* respectively; HLV and HLP are equal to 4 *m* and 10 *m* respectively (Fig. 1.c).

Table 5: Failure Rates for T = 20°*C* in Version 5

component	λ_0
P1	0.004566 h^{-1}
P2	0.005714 h^{-1}
V	0.003125 h^{-1}

$$L(t) = L_0 + Q \cdot (t - t_0) \quad (1)$$

$$T(t) = T_0 \cdot L_0/L(t) + T_{in} \cdot Q \cdot (t - t_0) / L(t) \quad (2)$$

$$\lambda(T) = \lambda_0 \cdot (0.2e^{0.005756(T-20)} + 0.8e^{-0.2301(T-20)}) \quad (3)$$

Three versions of the benchmark are characterized by the aspects described above:

- **Version 5.1**: the controller cannot fail.
- **Version 5.2**: the controller has a probability of failure on demand equal to 0.2.
- **Version 5.3**: initially the controller has a probability of failure on demand equal to 0.2; due to the wear of the controller, such probability is increased of 50% every time that the controller has to act (at each demand).

4. BASIC NOTIONS ABOUT SAN

A SAN model can contain two kinds of *places*. A *standard* place contains a certain number of *tokens* (*marking*) corresponding to an integer variable. The marking of an *extended* place corresponds to a variable whose type is not integer, but it can be a float number, a character, an array, etc. A place graphically appears as a circle, while *activities* (transitions) graphically appear as vertical bars.

The completion (firing) of an activity is enabled by a particular condition on the marking of a set of places. This marking can be expressed by connecting the activity to the standard places by means of oriented arcs, as it is possible in Petri Nets. The effect of the activity completion on the standard places can be specified in the same way. Another way to express the enabling condition consists of using *input gates*. An input gate is connected to an activity and to a set of standard or extended places; the input gate is characterized by two expressions:

- a *predicate* consists of a Boolean condition expressed in terms of the marking of the places; if this condition holds, then the activity is enabled to complete.
- A *function* expresses the effect of the activity completion on the marking of the places.

A SAN model can contain also *output gates*. The role of an output gate is specifying only the effect of the activity completion on the marking of the places. Therefore an output gate is characterized only by a function. The marking enabling the same activity can be expressed by means of oriented arcs, or by means of an input gate. Gates graphically appear as triangles (input gate: ◄ - output gate: ►).

In a SAN model, it is possible to set several completion cases for an activity; each case corresponds to a certain effect of the completion and has a certain probability: when the activity completes, one of the cases happens. A case graphically appears as a small circle close to the activity; from the case an arc is directed to a gate or a place.

The completion of an activity can be immediate or timed. In the second case, the completion time can be constant or random. A random completion time has to be ruled by a probability distribution; in this paper, we always resort to the negative exponential one, but several other distributions are available in the SAN formalism. In this paper (and in Codetta (2011)), we call "immediate activity" an activity completing as soon as it is enabled; we call "deterministic activity" an activity whose completion time is deterministic and not immediate; finally, we call "stochastic activity" an activity whose completion time is random.

5. MODELING THE SYSTEM

Each version of the benchmark (Sec. 3) has been modelled as a SAN where each aspect of the system behaviour is represented: the state of components, the

failure events, the current liquid level and its variations, the orders by the controller, the liquid dry out or overflow, etc.

5.1. Modelling Version 1

The SAN model of Version 1 is depicted in Fig. 3 where the current state of the pump P1 is represented by means of the places *P1_on* and *P1_stuck*: if *P1_on* is empty, this means that P1 is off; if instead the place *P1_on* is marked with one token, then P1 is on. The place *P1_stuck* is used to represent the stuck condition of P1: if such place is empty, then P1 is not stuck; if instead the place *P1_stuck* contains one token, this means that P1 is currently stuck. According to the marking combinations of the places *P1_on* and *P1_stuck*, we can model all the possible states of P1: ON, OFF, Stuck-ON, Stuck-OFF (Sec. 3).

Initially the place *P1_on* is marked with one token, and the place *P1_stuck* is empty, in order to model that P1 is initially in state ON. The state transitions of P1 caused by its failure are modelled by the stochastic activity *P1_fail* whose completion rate is equal to the failure rate of P1 (Tab. 1). The completion of such activity is ruled by the input gate *I_P1_fail*: *P1_fail* may complete only if the place *P1_stuck* is empty (P1 is not stuck), while there is no condition about the place *P1_on* (the failure may occur during both the ON state and the OFF state). The same gate partially specifies the effect of the completion of *P1_fail*: the gate sets the marking of the place *P1_stuck* to 1 (P1 becomes stuck), and sets the marking of *P1_on* to 0. The effect of the activity *P1_fail* is ruled also by two completion cases: in one case the marking of the place *P1_on* is not changed, and in this way we model the state transition toward the state Stuck-OFF; in the other case, one token appears in *P1_on*, in order to represent the state transition toward the state Stuck-ON. These two completion cases have the same probability to occur: 0.5.

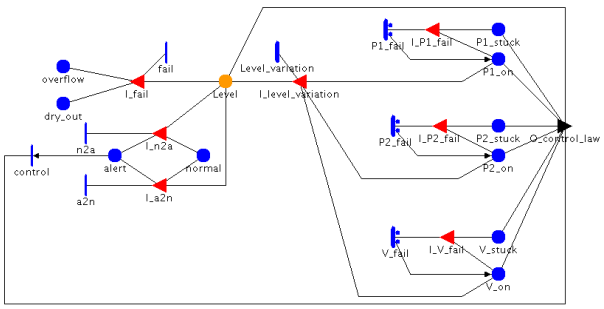

Figure 3: The SAN Model of Version 1

The current state of the pump P2 and the state transitions due to a failure of P2 are modelled in the same way by the places *P2_on* and *P2_stuck*, the stochastic activity *P2_fail* and the input gate *I_P2_fail*. Initially both *P2_on* and *P2_stuck* are empty in order to model that P2 is initially in state OFF. The state evolution of the valve V is modelled by the places *V_on* and *V_stuck*, the stochastic activity *V_fail* and the input gate *I_V_fail*. The initial ON state of V is modelled by the presence of one token inside *V_on* and no tokens inside *V_stuck*.

The current level (H) of the liquid in the tank expressed in meters, is represented by the extended place *Level* whose marking is a float variable initially set to 0 corresponding to the initial level of the liquid (Sec. 3). In the SAN model, we model any variation of H by 0.01 *m*; this is done by increasing or decreasing the marking of *Level* by 0.01. The action of P1, P2 and V on H are modelled by the deterministic activity *Level_variation* and in particular by the corresponding input gate *I_Level_variation*. Such gate enables *Level_variation* to complete only when the state configuration 1, 2, 4, 6, or 7 holds (Tab. 2). The action of P1, P2 or V on H is ruled by a level variation rate equal to 0.6 *m/h* (Sec. 3); this means that the action of a pump (valve) increases (decreases) the liquid level by 0.01 *m* every 0.016667 *h*. Since we are interested in representing any variation of H by 0.01 *m*, *Level_variation* completes every 0.016667 *h* in state configurations 1, 4, 6, 7, or every 0.016667/2 *h* in state configuration 2 (Tab. 2). The gate *I_Level_variation* specifies also the effect of the completion of *Level_variation*: each time such activity completes, the marking of the place *Level* is increased by 0.01 in the state configurations 1, 2, 4, 6, or is decreased by 0.01 in the state configuration 7 (Tab. 2).

The place *normal* is initially marked with one token in order to represent that H is inside the region of correct functioning (Sec. 3). The completion of the immediate activity *n2a* removes the token inside the place *normal*; according to the input gate *I_n2a*, this happens if the marking of the extended place *Level* is less than HLA or more than HLB (Tab. 3). The same gate sets the marking of the place *alert* to 1. In this way, we model that the liquid level in the tank is outside the correct region. The presence of one token inside *alert* enables the immediate activity *control* to complete. The effect of its completion is ruled by the output gate *O_control_law* executing the control laws in Tab. 3: such gate acts on the marking of the places *P1_on*, *P2_on* and *V_on*, and consequently on the state of P1, P2 and V. So, *control* together with *O_control_law*, models the orders given by the controller. If the place *P1_stuck*, *P2_stuck* or *V_stuck* is marked, then the output gate *O_control_law* has no effect on the place *P1_on*, *P2_on* or *V_on* respectively. In this way we model that the controller cannot act on the state of a stuck component.

The controller action on the component states may lead H back to the region of correct functioning. In this case, the immediate activity *a2n* is enabled to complete by the input gate *I_a2n* checking that the marking of the extended place *Level* is equal or greater than HLA and less or equal to HLB. The effect of the completion of *a2n* is the presence of one token inside the place *normal* in order to represent that H in inside the region of correct functioning.

The dry out and the overflow condition determining the system failure, are detected by the immediate activity *fail* and in particular by the corresponding input gate *I_fail*: if the marking of the extended place *Level* is less than HLV, then one token appears in the place *dry_out* in order to model the occurrence of the dry out. If instead the marking of *Level* is greater than HLP, then the effect of the completion of *fail* is the presence of one token inside the place *overflow* modelling the occurrence of the overflow.

Further details of the SAN model in Fig. 3 are reported in Codetta (2011).

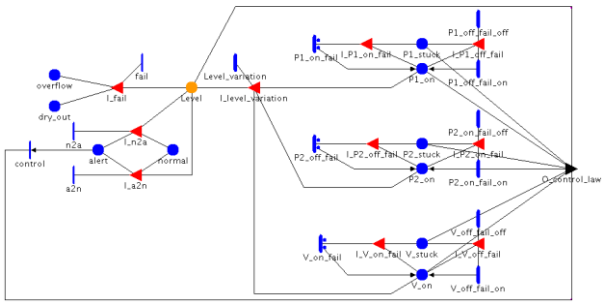

Figure 4: The SAN Model of Version 2

5.2. Modelling Version 2

In Version 2 (Sec 3.1), the failure rates of P1, P2 and V are state dependent (Tab. 4). The SAN model of Version 2 appears in Fig. 4 where the current state of P1 is still modelled by the marking of the places *P1_on* and *P1_stuck*, but the state transitions caused by the failure are now modelled by three stochastic activities: *P1_on_fail* models the failure of P1 during the state ON; *P1_off_fail_off* represents the failure during the state OFF and leading to the state Stuck-OFF; *P1_off_fail_on* models the failure during the state OFF, but leading to the state Stuck-ON. The state evolution of P2 and V is modelled in a similar way. The other parts of the SAN model in Fig. 5 are the same as in the model of Version 1 (Fig. 3).

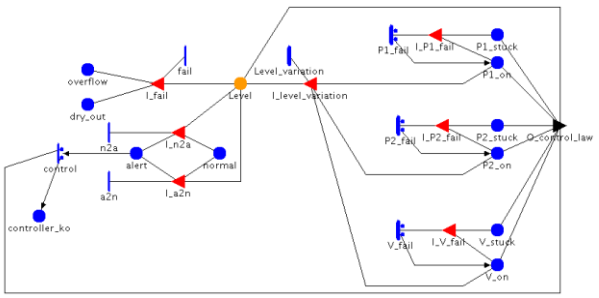

Figure 5: The SAN Model of Version 3

5.3. Modelling Version 3

In Version 3, the controller failure on demand is introduced (Sec. 3.2); this aspect is represented in the SAN model in Fig. 5 by the presence of two completion cases for the immediate activity *control* modelling the action of the controller. In one case, the effect of the completion of *control* is ruled by the output gate *O_control* executing the control laws (Tab. 3). In the other case, the failure on demand occurs and the only effect is the addition of one token to the marking of the new place *controller_ko* counting the number of failures of the controller.

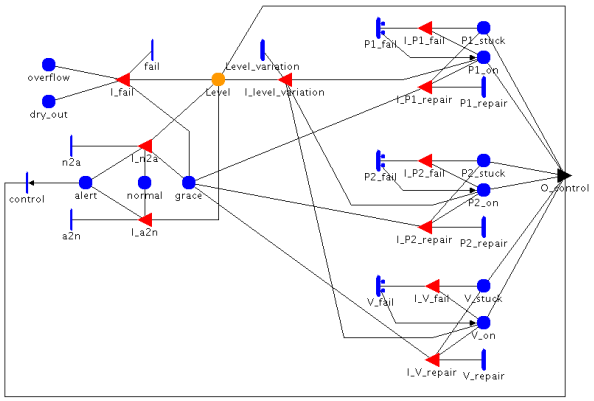

Figure 6: The SAN Model of Version 4

5.4. Modelling Version 4

Version 4 is modelled in Fig. 6 where the immediate activity *n2a* still completes when H exceeds the region of correct functioning, but now *n2a* inserts also one token inside the new place *grace* in order to represent that the grace period (Sec. 3.3) has begun. Such marking enables the new stochastic activities *P1_repair*, *P2_repair* and *V_repair* ruled by the input gates *I_P1_repair*, *I_P2_repair* and *I_V_repair*, and modelling the repair of P1, P2 and V respectively. The effect of the completion of such activities is the removal of the token inside the place representing the stuck condition of the component (*P1_stuck*, *P2_stuck* and *V_stuck* respectively). The immediate activity *fail* still models the system failure, but now it also removes the token inside the place *grace*, in order to represent the end of the grace period.

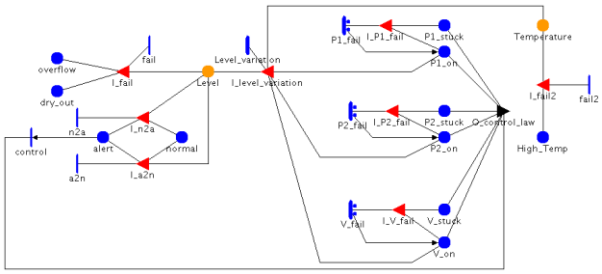

Figure 7: The SAN Model of Version 5.1

5.5. Modelling Version 5

The SAN models representing Versions 5.1, 5.2, 5.3 are depicted in Figures 7, 8, 9, respectively. Such models are characterized by the presence of a new extended place called *Temperature* representing the current liquid temperature (T). The deterministic activity *Level_variation*, together with the input gate *I_level_variation*, models the variation of H and T, as a consequence of the heat source (Sec. 3.4) and the injection of new liquid in the tank by the pumps (Eq. 2). The rates of the stochastic activities *P1_fail*, *P2_fail*

and *V_fail* modelling the failure of P1, P2 and V respectively, are expressed as a function of the marking of *Temperature* according to Eq. 3.

In Versions 5.1, 5.2, 5.3, the system failure condition due to the high temperature of the liquid is introduced (Sec. 3.4). In the SAN models in Fig. 7, 8, 9, such condition is detected by the new immediate activity *fail2* ruled by the input gate *I_fail2*: when the marking of *Temperature* reaches the value of 100, one token appears inside the new place *High_Temp*.

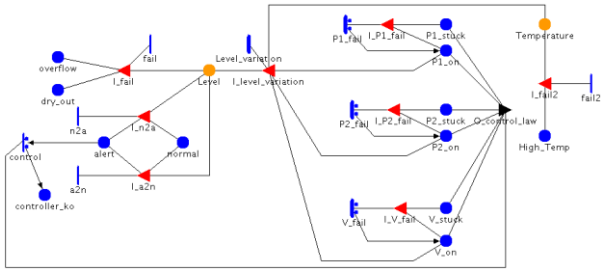

Figure 8: The SAN Model of Version 5.2

The Version 5.2 is characterized by the possible failure on demand of the controller (Sec. 3.4). In the SAN model in Fig. 8, such aspect is represented in the same way as in the SAN model of Version 3 (Fig. 5).

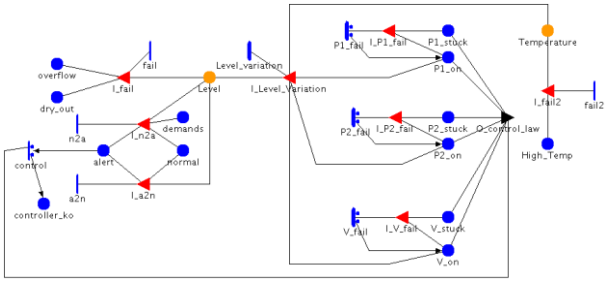

Figure 9: The SAN Model of Version 5.3

In Version 5.3, the probability of failure on demand of the controller is increased by 50% after each demand (Sec. 3.4). In Fig. 9, the marking of the new place *demands* indicates the number of demands: every time that the immediate activity *n2a* completes (H reaches the control boundaries), the marking of *demands* is increased by one. The immediate activity *control* still has two completion cases, but now their probabilities are a function of the marking of *demands*.

The full details of all the SAN models can be found in Codetta (2011).

6. SIMULATION RESULTS

The SAN models presented in the previous section have been simulated. In particular, for each model, 100'000 simulation batches have been performed by means of the *Möbius* tool, requiring a confidence level equal to 0.95, and a relative confidence interval equal to 0.1. The measures computed by the simulation are the *cumulative distribution function* (cdf) of the probability of each system failure condition (Sec. 3) for a mission time varying between 0 and 1000 *h* (or between 0 and 500 *h* in Version 4, as in Marseguerra and Zio (2006)). The cdf provides the system unreliability (Sec. 1) according to a specific failure condition. For instance, the value of the dry out cdf at time $t > 0$ is the probability that the system has failed because of the dry out, during the time period $(0, t)$.

The cdf of the dry out probability is computed as the mean value over the 100'000 simulation batches, of the marking of the place *dry_out* present in all the SAN models (Sec. 5). In each simulation batch and at a certain time, the number of tokens inside the place *dry_out* is equal to 0 if the dry out has not occurred, or it is equal to 1 if the dry out condition holds (Sec. 5). So, the mean value of its marking at a certain time, over the 100'000 simulation batches, provides the probability that the dry out condition holds at that time.

The cdf of the overflow probability is computed in the same way, but with reference to the place *overflow* present in all the SAN models (Sec. 5). In Versions 5.1, 5.2 and 5.3, another system failure condition is taken into account: the temperature of the liquid reaching 100°*C* (Sec. 3.4). The cdf of such condition is computed as the mean number of tokens inside the place *High_Temp* present in the SAN models in Fig. 7, 8 and 9 (Sec. 5.5).

6.1. Results for Versions 1, 2, 3

The values of the cdf of the dry out in Versions 1, 2 and 3 (SAN model in Fig. 4, 5 and 6 respectively) are reported in Tab. 6 and are graphically compared in Fig. 10. The values of the cdf of the overflow are reported in Tab. 7 and are graphically compared in Fig. 11. The results returned by the SAN model simulation are similar to the values returned by Monte Carlo simulation, GSPN analysis and FSPN simulation.

Table 6: The cdf of the Dry Out in Versions 1, 2, 3

time	Version 1	Version 2	Version 3
100 *h*	4.5900E-03	2.0240E-02	4.9090E-02
200 *h*	2.2390E-02	4.0400E-02	8.6710E-02
300 *h*	4.4890E-02	5.4090E-02	1.0952E-01
400 *h*	6.5990E-02	6.3360E-02	1.2664E-01
500 *h*	8.2600E-02	6.9870E-02	1.3844E-01
600 *h*	9.5290E-02	7.3750E-02	1.4707E-01
700 *h*	1.0393E-01	7.6650E-02	1.5313E-01
800 *h*	1.1003E-01	7.8340E-02	1.5739E-01
900 *h*	1.1435E-01	7.9440E-02	1.6024E-01
1000 *h*	1.1747E-01	8.0240E-02	1.6220E-01

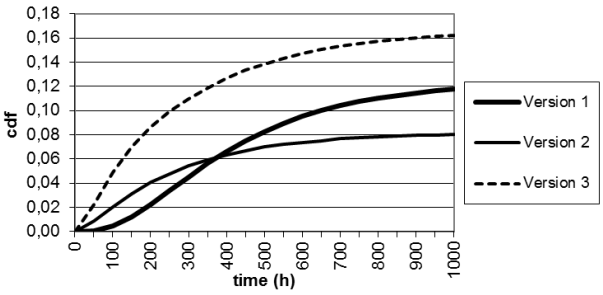

Figure 10: The cdf of the Dry Out in Versions 1, 2, 3

Table 7: The cdf of the Overflow in Versions 1, 2, 3

time	Version 1	Version 2	Version 3
100 *h*	7.8880E-02	7.9370E-02	1.3517E-01
200 *h*	1.9914E-01	1.6852E-01	2.7244E-01
300 *h*	2.9386E-01	2.3411E-01	3.6541E-01
400 *h*	3.6207E-01	2.7882E-01	4.2492E-01
500 *h*	4.0667E-01	3.0943E-01	4.6332E-01
600 *h*	4.3665E-01	3.2938E-01	4.8808E-01
700 *h*	4.5683E-01	3.4310E-01	5.0444E-01
800 *h*	4.7063E-01	3.5284E-01	5.1537E-01
900 *h*	4.7929E-01	3.6009E-01	5.2298E-01
1000 *h*	4.8572E-01	3.6500E-01	5.2797E-01

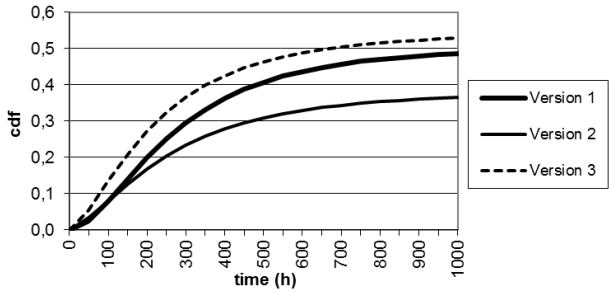

Figure 11: The cdf of the Overflow in Versions 1, 2, 3

6.2. Results for Version 4

The results obtained by simulating the model in Fig. 6, are reported in Tab. 8, in Fig. 12 (dry out) and in Fig. 13 (overflow). They differ from the results returned by Monte Carlo simulation in Marseguerra and Zio (1996), even though they are in the same order of magnitude. Moreover, they differ from the results obtained by means of GSPN analysis and FSPN simulation in Codetta and Bobbio (2005a) where the repair is erroneously assumed to be allowed only while the level is outside the region of correct functioning, instead of during the grace period (Sec. 3.3).

Table 8: The cdf of the Dry Out and the Overflow in Version 4

time	dry out	overflow
50 *h*	0.000E+00	8.000E-04
100 *h*	6.000E-05	2.430E-03
150 *h*	1.500E-04	4.230E-03
200 *h*	2.200E-04	6.090E-03
250 *h*	3.300E-04	7.920E-03
300 *h*	3.700E-04	9.460E-03
350 *h*	4.500E-04	1.069E-02
400 *h*	4.500E-04	1.197E-02
450 *h*	4.700E-04	1.298E-02
500 *h*	5.100E-04	1.363E-02

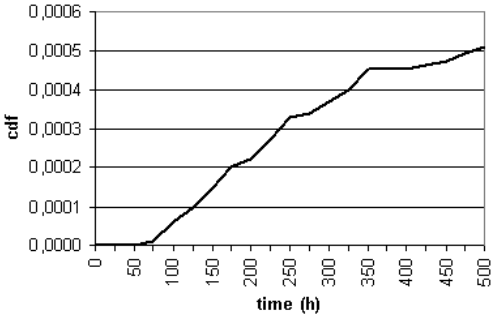

Figure 12: The cdf of the Dry Out in Version 4

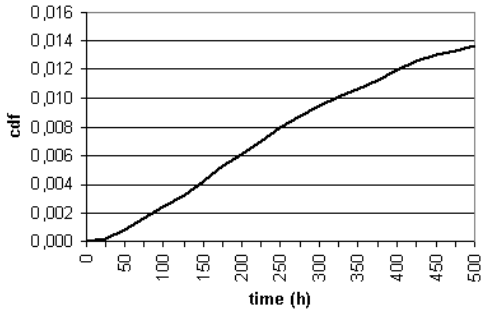

Figure 13: The cdf of the Overflow in Version 4

6.3. Results for Version 5

The results for Versions 5.1, 5.2 and 5.3 are reported in Tables 9, 10 and 11 respectively. In particular, the results in Version 5.1 (Fig. 14) and Version 5.2 (Fig. 15) are similar to the values returned by Monte Carlo simulation in Marseguerra and Zio (1996) and FSPN simulation in Codetta and Bobbio (2005b). Version 5.3 was not modelled as a FSPN in the past. According to the results for such version (Fig. 16), the wear of the controller (Sec. 3.4) does not seem to have a relevant impact on the cdf values, with respect to Version 5.2. In Marseguerra and Zio (1996) instead, the controller wear determines a slight increase of the dry out cdf values.

Table 9: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.1

time	dry out	overflow	high temp.
100 *h*	3.1650E-02	2.4007E-01	0.0000E+00
200 *h*	7.9330E-02	3.9531E-01	0.0000E+00
300 *h*	1.0517E-01	4.5631E-01	0.0000E+00
400 *h*	1.1706E-01	4.8133E-01	0.0000E+00
500 *h*	1.2200E-01	4.9161E-01	1.3000E-04
600 *h*	1.2376E-01	4.9588E-01	3.8200E-02
700 *h*	1.2424E-01	4.9750E-01	7.3850E-02
800 *h*	1.2436E-01	4.9826E-01	1.1855E-01
900 *h*	1.2438E-01	4.9864E-01	1.2614E-01
1000 *h*	1.2438E-01	4.9884E-01	1.2724E-01

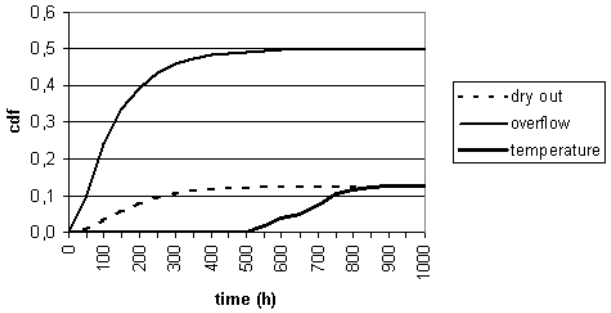

Figure 14: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.1

Table 10: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.2

time	dry out	overflow	high temp.
100 *h*	1.1102E-01	3.3740E-01	0.0000E+00
200 *h*	1.4934E-01	4.7526E-01	0.0000E+00
300 *h*	1.6601E-01	5.2228E-01	0.0000E+00
400 *h*	1.7271E-01	5.4072E-01	2.0000E-05
500 *h*	1.7559E-01	5.4828E-01	2.2000E-04
600 *h*	1.7650E-01	5.5178E-01	2.0370E-02
700 *h*	1.7677E-01	5.5325E-01	4.0250E-02
800 *h*	1.7685E-01	5.5387E-01	6.7460E-02
900 *h*	1.7687E-01	5.5416E-01	7.1930E-02
1000 *h*	1.7687E-01	5.5428E-01	7.2550E-02

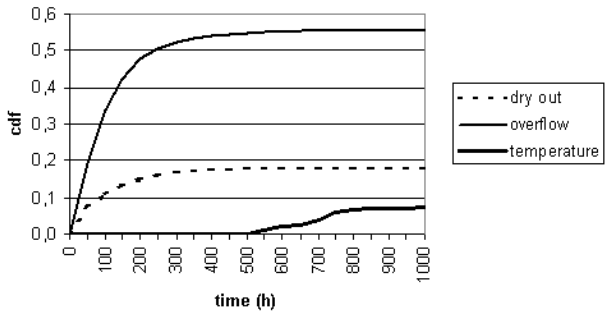

Figure 15: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.2

Table 11: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.3

time	dry out	overflow	high temp.
100 *h*	1.1427E-01	3.4165E-01	0.0000E+00
200 *h*	1.5201E-01	4.7762E-01	0.0000E+00
300 *h*	1.6799E-01	5.2503E-01	0.0000E00
400 *h*	1.7470E-01	5.4360E-01	1.0000E-05
500 *h*	1.7748E-01	5.5109E-01	2.0000E-04
600 *h*	1.7849E-01	5.5462E-01	1.9230E-02
700 *h*	1.7878E-01	5.5604E-01	3.7920E-02
800 *h*	1.7886E-01	5.5663E-01	6.5320E-02
900 *h*	1.7889E-01	5.5694E-01	6.9780E-02
1000 *h*	1.7889E-01	5.5708E-01	7.0460E-02

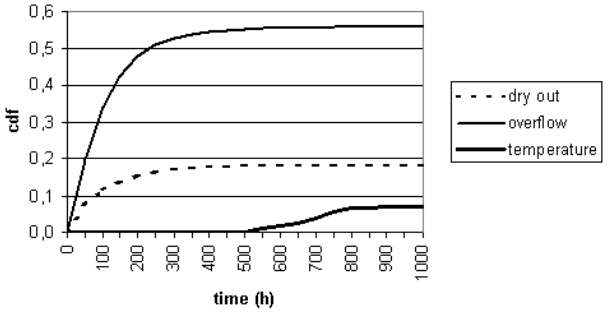

Figure 16: The cdf of the Dry Out, the Overflow and the High Temperature in Version 5.3

7. CONCLUSIONS

A benchmark on dynamic reliability taken from the literature has been examined. Each version focuses on a particular aspect of the dynamic behaviour of the system, such as state or temperature dependent failure rates, repairable components, failures on demand. The benchmark was originally evaluated in terms of system unreliability by means of Monte Carlo simulation. In this paper, the benchmark versions have been modelled and simulated using SAN, a particular form of Petri Net. The results in this paper are in general coherent to the original ones and those obtained by means of other Petri Net based formalisms such as GSPN and FSPN. This confirms that Petri Net models are a valid approach to deal with dynamic reliability cases because of the possibility to model the stochastic, timed or immediate events characterizing the complete behaviour of the system.

In particular, the use of SAN has several advantages. Gates make the SAN model more compact: many predicates and functions (Sec. 4) that are incorporated into the input or output gates, would have required more transitions (activities) and arcs in order to be represented in a GSPN or FSPN. For instance, in the GSPN models, the orders by the controller are represented by 6 transitions, while in the SAN model, only the activity (transition) *control*, together with its output gate, is necessary. In the GSPN model, the variations to the liquid level are represented by 5 transitions, while the activity *Level_variation* is enough in the SAN. The failure of a pump or valve is represented by four transitions in the GSPN; in the SAN instead, one activity is necessary (Sec. 5).

The SAN formalism can represent float variables, as in FSPN, by means of extended places. An example is the place *Level* modelling the liquid level. This avoids the discretization into integer values of float variables, required in GSPN. The negative values of variables can be directly mapped into the marking of SAN places. For instance, the liquid level in Versions 1, 2, 3 varies between -3 *m* and +3 *m* (Sec. 3), just like the marking of the place *Level* (Sec. 5). In the GSPN and FSPN model instead, the liquid thresholds had to be redefined in order to avoid negative values.

AUTHOR BIOGRAPHY

Daniele Codetta-Raiteri received the Ph.D. in Computer Science from the University of Turin, Italy, in 2006. Now he is a researcher at the University of Eastern Piedmont, Italy. His research focuses on stochastic models for Reliability evaluation, with a particular experience in Fault Trees, Petri Nets and Bayesian Networks. He is the (co-)author of more than thirty papers published in proceedings or journals.

REFERENCES

Ajmone-Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G., 1995. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing.

Chiola, G., Franceschinis, G.,Gaeta, R., Ribaudo, M., 1995. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation, special issue on Performance Modeling Tools*, 24(1&2):47–68.

Codetta-Raiteri, D., 2011, *SAN models of a benchmark on dynamic reliability*, Università del Piemonte Orientale. Available from: http://people.unipmn.it/dcr

Codetta-Raiteri, D., Bobbio, A., 2005a. Solving Dynamic Reliability Problems by means of Ordinary and Fluid Stochastic Petri Nets. *Proceedings of the European Safety and Reliability Conference (ESREL)*, pp. 381–389. June, Gdansk (Poland).

Codetta-Raiteri, D., Bobbio, A., 2005b. Evaluation of a benchmark on dynamic reliability via Fluid Stochastic Petri Nets. *Proceedings of the International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*, pp. 52–55. September, Turin (Italy).

Deavours, D., Clark, G., Courtney, T., Daly, D., Derisavi, S., Doyle, J., Sanders, W., Webster, P.G., 2002. The Möbius Framework and its Implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969.

Distefano, S., Xing, L., 2006. A New Approach to Model the System Reliability: Dynamic Reliability Block Diagrams. *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS)*, pp. 189–195. January, Newport Beach (California, USA).

Dugan, J.B., Bavuso, S.J., Boyd, M.A., 1992. Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems. *IEEE Transactions on Reliability*, 41:363–377.

Gribaudo, M., 2001. FSPNEdit: A fluid stochastic Petri net modeling and analysis tool. *Proceedings of Tools of International Multiconfernce on Measurements Modelling and Evaluation of computer Communication Systems*, pp. 24–28. September, Aachen (Germany).

Gribaudo, M., Sereno, M., Horvath, A., Bobbio, A., 2001. Fluid Stochastic Petri Nets augmented with flush-out arcs: Modelling and analysis. *Discrete Event Dynamic Systems*, 11(1&2):97–117.

Marseguerra, M., Zio, E., 1996. Monte Carlo Approach to PSA for dynamic process system. *Reliability Engineering and System Safety*, 52:227–241.

Marseguerra, M., Zio, E., Devooght, J., Labeau, P.E., 1998. A concept paper on dynamic reliability via Monte Carlo simulation. *Mathematics and Computers in Simulation*, 47:371-382.

Sahner, R.A., Trivedi, K.S., Puliafito, A. 1996. *Performance and Reliability Analysis of Computer Systems; An Example based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher.

Sanders, W.H., Meyer, J.F., 2001. Stochastic activity networks: Formal definitions and concepts. *Lecture Notes in Computer Science*, 2090:315–343.