Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Viale Teresa Michel 11, 15121 Alessandria
http://www.di.unipmn.it

SAN models of a benchmark on dynamic reliability

D. Codetta Raiteri (daniele.codetta_raiteri@mfn.unipmn.it)

TECHNICAL REPORT TR-INF-2011-12-03-UNIPMN
(December 2011)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL `http://www.di.unipmn.it/`. Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2011-02 *A new symbolic approach for network reliability analysis* , M. Beccuti, S. Donatelli, G. Franceschinis, R. Terruggia, June 2011.

2011-01 *Spaced Seeds Design Using Perfect Rulers*, L. Egidi, G. Manzini, June 2011.

2010-04 *ARPHA: an FDIR architecture for Autonomous Spacecrafts based on Dynamic Probabilistic Graphical Models*, D. Codetta Raiteri, L. Portinale, December 2010.

2010-03 *ICCBR 2010 Workshop Proceedings*, C. Marling, June 2010.

2010-02 *Verifying Business Process Compliance by Reasoning about Actions*, D. D'Aprile, L. Giordano, V. Gliozzi, A. Martelli, G. Pozzato, D. Theseider Dupré, May 2010.

2010-01 *A Case-based Approach to Business Process Monitoring*, G. Leonardi, S. Montani, March 2010.

2009-09 *Supporting Human Interaction and Human Resources Coordination in Distributed Clinical Guidelines*, A. Bottrighi, G. Molino, S. Montani, P. Terenziani, M. Torchio, December 2009.

2009-08 *Simulating the communication of commands and signals in a distribution grid*, D. Codetta Raiteri, R. Nai, December 2009.

2009-07 *A temporal relational data model for proposals and evaluations of updates*, L. Anselma, A. Bottrighi, S. Montani, P. Terenziani, September 2009.

2009-06 *Performance analysis of partially symmetric SWNs: efficiency characterization through some case studies*, S. Baarir, M. Beccuti, C. Dutheillet, G. Franceschinis, S. Haddad, July 2009.

2009-05 *SAN models of communication scenarios inside the Electrical Power System*, D. Codetta Raiteri, R. Nai, July 2009.

2009-04 *On-line Product Configuration using Fuzzy Retrieval and J2EE Technology*, M. Galandrino, L. Portinale, May 2009.

2009-03 *A GSPN Semantics for Continuous Time Bayesian Networks with Immediate Nodes*, D. Codetta Raiteri, L. Portinale, March 2009.

2009-02 *The TAAROA Project Specification*, C. Anglano, M. Canonico, M. Guazzone, M. Zola, February 2009.

2009-01 *Knowledge-Free Scheduling Algorithms for Multiple Bag-of-Task Applications on Desktop Grids*, C. Anglano, M. Canonico, February 2009.

2008-09 *Case-based management of exceptions to business processes: an approach exploiting prototypes*, S. Montani, December 2008.

Contents

SAN models of a benchmark on dynamic reliability

Daniele Codetta-Raiteri
Dipartimento di Informatica, Università del Piemonte Orientale, Italy
e-mail: raiteri@mfn.unipmn.it

Abstract

This report provides the detailed description of the Stochastic Activity Network (SAN) models appearing in [1] and concerning a benchmark on dynamic reliability taken from the literature.

1 Introduction

We talk about dynamic reliability [2] 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 behavior or the current state of the system. In dynamic reliability, considering only the combinations of failure events in the system is not enough to evaluate the system (un)reliability, but we actually have to consider the whole system behavior. This means modeling the normal functioning of the system, the occurrence of component failure events and their effect on the system functioning. For these reason, dynamic reliability cases are typically evaluated by means of simulation.

In this report, we take into account several versions of a benchmark on dynamic reliability taken from the literature [2]. Each version of the system is modeled and simulated as a *Stochastic Activity Network* (SAN) [3], a particular form of Stochastic Petri Net. The aim is to compute the system unreliability. The SAN models are designed and simulated by means of the *Möbius* tool [4]. The advantages of this approach with respect to previous works, are explained in [1].

This report is organized as follows: Sec. 2 describes the system behavior in each version of the benchmark. Sec. 3 provides the essential notions about the SAN formalism, in order to understand the description of the SAN models provided in Sec. 4. The results of the simulation of such models are reported in Sec. 5.

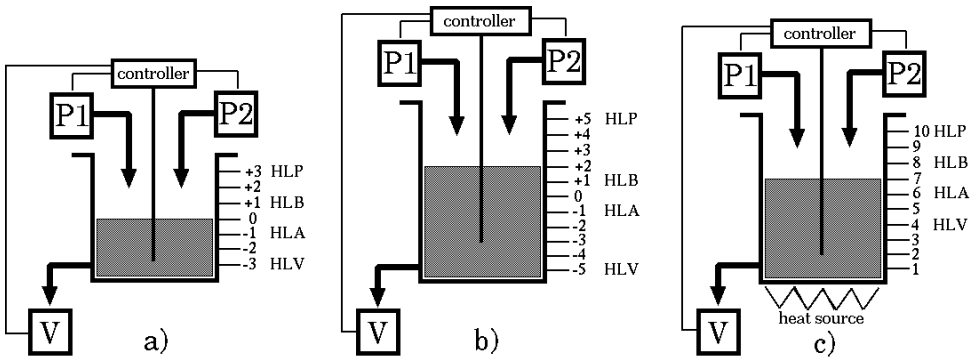

Figure 1: The system schemes in Versions 1, 2, 3 (a), in Version 4 (b), in Version 5 (c).

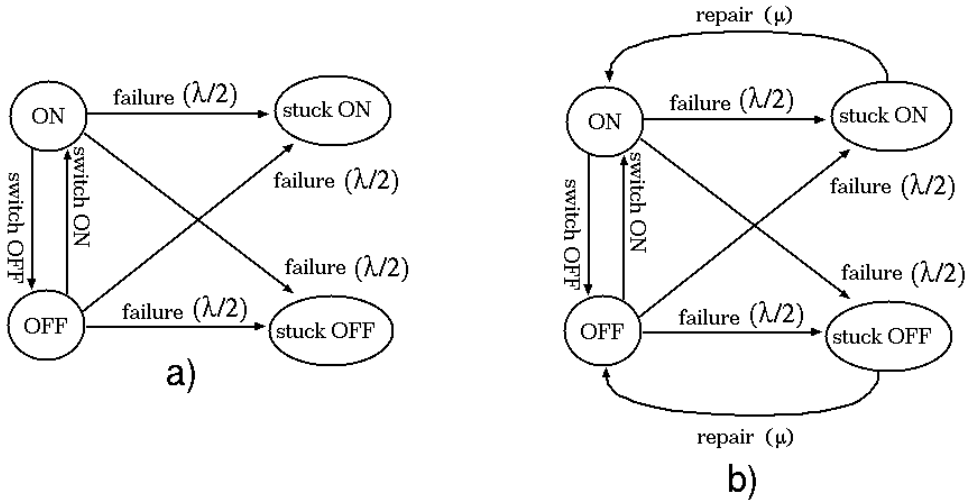

Figure 2: The states of a P1, P2 and V in Versions 1, 3, 5.1, 5.2, 5.3 (a). The states of a P1, P2 and V in Version 4 (b).

2 Benchmark specification

2.1 Version 1: state independent failure rates

The system (Fig. 1.a) is composed by a tank containing some liquid, two pumps (P1 and P2) to fill the tank, a 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 both pumps and the valve have the same rate of level variation (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; a failure consists of turning to the states stuck ON or stuck OFF. The failure probability obeys the negative exponential distribution; the failure rate 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

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

Table 1: Failure rates in Versions 1, 3, 4.

Configuration	P1	P2	V	effect on H
1	ON	OFF	OFF	$\uparrow$
2	ON	ON	OFF	$\uparrow\uparrow$
3	ON	OFF	ON	=
4	ON	ON	ON	$\uparrow$
5	OFF	OFF	OFF	=
6	OFF	ON	OFF	$\uparrow$
7	OFF	OFF	ON	$\downarrow$
8	OFF	ON	ON	=

Table 2: The level variation in each state configuration.

Boundary	P1	P2	V
$H < HLA$	ON	ON	OFF
$H > HLB$	OFF	OFF	ON

Table 3: Control boundaries and laws.

stuck ON or the stuck OFF state, are uniformly distributed. This corresponds to the situation depicted in Fig. 2.a according to the failure rates reported in Tab. 1.

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 denoted by HLA (-1 m) and HLB (+1 m). If H reaches 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 denoted as 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); in order to avoid the dry out, when H reaches HLA, the controller orders to both pumps to switch ON and to the valve to switch OFF, with the aim of increasing H. Tab. 3 shows the control laws with respect to H. The failure of the whole system happens when the dry out or the overflow occurs.

2.2 Version 2: state dependent failure rates

In this version of the system, the failure rate of a component changes according to its current state and to the state reached as a consequence of the failure (Fig. 3); Tab. 4 shows the rates for each state transition due to a component failure.

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}

Table 4: Failure rates for each component in each state, in Version 2.

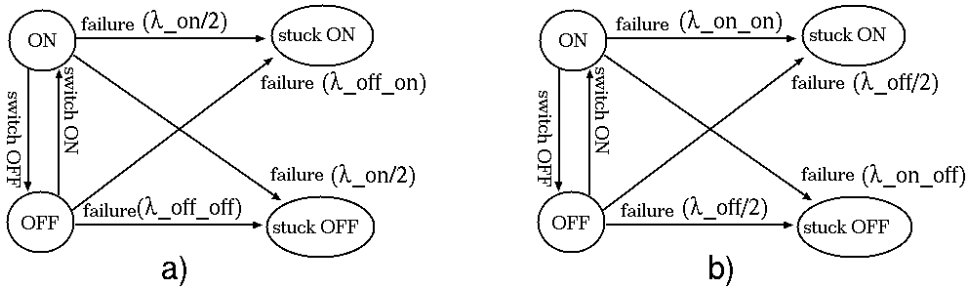

Figure 3: The states of a P1 and V in the Version 2 (a). The states of a P2 in the Version 2 (b).

2.3 Version 3: controller failure on demand

In this version of the system, the controller has a 0.1 probability of failure on demand. This means that when the liquid level exceeds the region of correct functioning (HLA<H<HLB), the controller may fail with a 0.1 probability. If this occurs, no command is sent to P1, P2 and V, so they maintain their current state, although the liquid is outside the region of correct functioning.

2.4 Version 4: repairable components

In this version of the benchmark a stuck component can be repaired during the grace period which begins when the region of correct functioning (HLA<H<HLB) is exceeded for the first time, and lasts until the occurrence of the system failure (dry out or overflow condition). The time to repair of a component is a random variable obeying to 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 the liquid level is currently out of the region of correct functioning (Tab. 3). Moreover, the repaired component

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

Table 5: Failure rates for $T = 20°C$ in Versions 5.1, 5.2, 5.3.

can respond to eventual other 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, the thresholds for the dry out (HLV) and the overflow condition (HLP) are -5 m and +5 m respectively, as shown in Fig. 1.b.

2.5 Version 5: temperature dependent failure rates

In this version, the current temperature of the liquid in the tank is taken into account. A heat source increases the temperature (T) of the liquid inside the tank. The heating power of the heat source is $w = 1m°C/h$; we assume that there is no heat released outside the tank, and that the heat is uniformly distributed on the liquid [2]. 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 we assume that it gets mixed instantaneously with the liquid in the tank [2].

The level variation rate for the components 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 Eq. 2 to provide respectively the liquid level and temperature at time $t > t_0$. In Eq. 1, L_0 is the the liquid level at time t_0, while in Eq. 2, T_0 is the liquid temperature at time t_0.

$$L(t) = L_0 + Q \cdot (t - t_0) \tag{1}$$

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

The failure rates of the components P1, P2 and V are temperature dependent; λ_0 is the failure rate of the component for a temperature equal to $20°C$ (Tab. 5); the failure rate as a function of T, is given by Eq. 3.

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

Besides the dry out and the overflow, in this version of the system, another condition determines

the failure of the system: this occurs if T reaches $100°C$.

Moreover in these versions, the initial level of the liquid in the tank is 7 m, the thresholds HLA and HLB for the control laws are set to 6 m and 8 m respectively, the boundaries for the dry out (HLV) and the overflow condition (HLP) are equal to 4 m and 10 m respectively (Fig. 1.c).

In [2], three versions of the benchmark are characterized by the aspects described above:

- **Version 5.1:** the controller can not fail.
- **Version 5.2:** the controller has a probability of failure on demand equal to 0.2; this means that the probability to fail of C when the liquid reaches a control boundary (Tab. 3), is 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% after the execution of a control law (demand).

3 Basic notions about SAN

SAN can be considered as a particular form of Stochastic Petri Nets. So, a SAN model contains *places*. A standard place contains a certain number of *tokens* (*marking*); therefore the marking of a standard place corresponds to an integer variable. A place graphically appears as a circle. Besides standard places, a SAN model can contain extended places. The marking of such a place corresponds to a variable whose type is not integer, but it can be a float number, a character, an array, etc. A particular condition on the marking of a certain set of places enables the completion (firing) of *activities* (transitions) whose effect is modifying in some way the marking of the places. Activities graphically appear as vertical bars.

In the SAN formalism, the completion of an activity can be immediate or timed. In the second case, the completion time can be a constant value or a random value. If the completion time is random, its value 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. Moreover, in this paper, we call "immediate activity" an activity completing as soon as it is enabled; we call "deterministic activity" an activity whose time to complete is deterministic and not immediate; we call "stochastic activity" an activity whose time to complete is a random variable ruled by the negative exponential distribution.

The completion of an activity of any kind 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 condition on the marking enabling a certain activity 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 connected to the gate; if this condition holds, then the activity connected to the gate is enabled to complete.

- a *function* expresses the effect of the activity completion on the marking of the places connected to the gate.

Besides input gates, a SAN model can contain *output gates* as well. An output gate has to be connected to a certain activity and to a set of standard or extended places. The role of an output gate is specifying only the effect of the activity completion on the marking of the places connected to the output gate. 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.

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 to a place.

The design, the composition, the analysis and the simulation of SAN models is supported by the *Möbius* tool [4]. Further information about the SAN formalism can be found in [3].

4 Benchmark models

4.1 Modelling Version 1

The SAN model of the Version 1 of the benchmark is reported in Fig. 4. The details about the activities and the gates in such model are reported in Tab. 6 and Tab. 7. The initial marking of each place in the

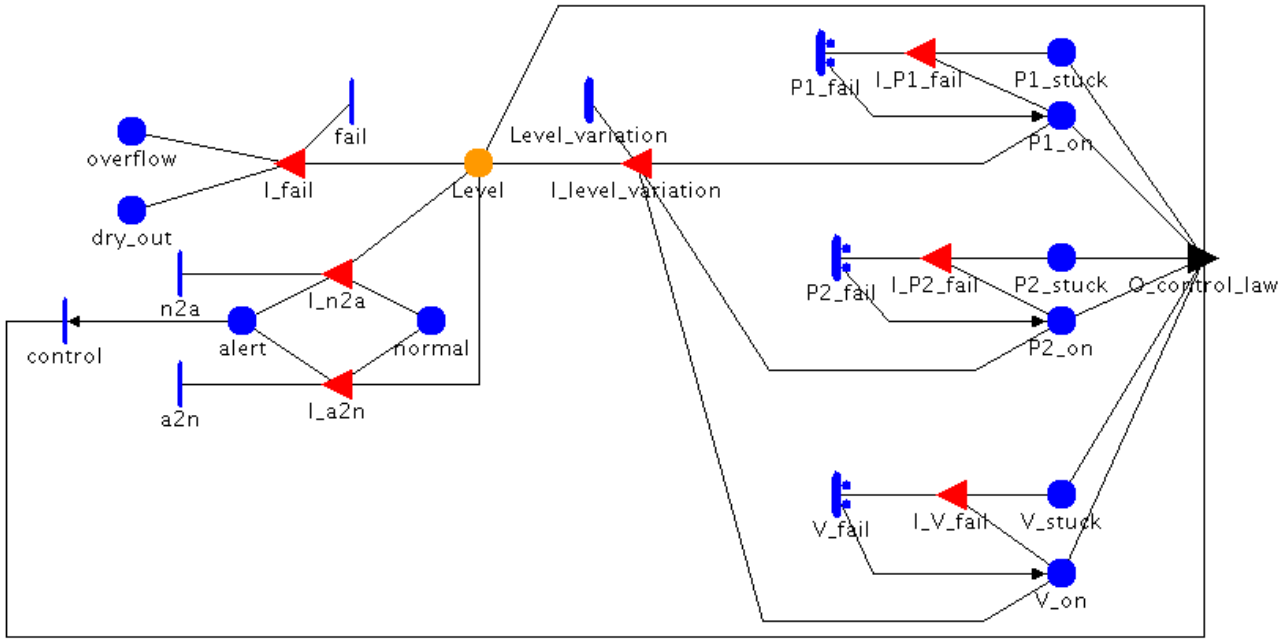

Figure 4: The SAN model of the Version 1.

SAN model in Fig. 4 (and in the following models) is reported in Tab. 15.

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 currently OFF; if instead the place *P1_on* is marked with one token, then P1 is currently 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 [1].

Initially the place *P1_on* is marked with one token, while the place *P1_stuck* is empty, in order to model that P1 is initially ON and is not stuck. The state transitions of P1 caused by its failure are modeled by the stochastic activity *P1_fail* whose completion rate is equal to the failure rate of P1 (0.004566 h^{-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). Besides enabling the completion of the activity *P1_fail*, the input gate *I_P1_fail* partially specifies the effect of the completion of *P1_fail*: this determines the marking of the place *P1_stuck* (P1 becomes stuck), and sets the marking of *P1_on* to 0. The effect of the completion of the activity *P1_fail* is determined also by two completion cases associated with the same activity: 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 is added to the marking of the place *P1_on* in order to represent the state transition toward the state stuck ON. These two completion cases are mutually exclusive and have the same probability to occur: 0.5.

The current state of the pump P2, and the state transitions due to a failure of P2, are modeled in a similar way by the places *P2_on* and *P2_stuck*, by the stochastic activity *P2_fail* and the input gate *I_P2_fail*. The current state of the valve V and its state transitions as a consequence of the failure, are modeled by the places *V_on* and *V_stuck*, by the stochastic activity *V_fail* and the input gate *I_V_fail*. Initially both the place *P2_on* and the place *P2_stuck* are empty in order to model that P2 is initially in state OFF. The initial ON state of V is modeled by the presence of one token inside the place *V_on* and no tokens inside the place *V_stuck* (Tab. 15).

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 [1]. In the SAN model, we model any variation of the liquid level by 0.01 m; this is done by increasing or decreasing the marking of the extended place *Level* by 0.01. The action of P1, P2 and V on the liquid level are modeled by the deterministic activity *Level_variation* and in particular by the corresponding input gate *I_Level_variation*. Such gate enables the activity *Level_variation* to complete only when a state configuration leading to the liquid level variation holds (state configurations 1, 2, 4, 6, 7 in Tab. 2). The action of P1, P2 or V on the liquid level in the tank is ruled by a level variation rate equal to 0.6 m/h [1]; this means that the action of a pump (valve) increases (decreases) the liquid level by 0.01 m every 0.016667 h. Since in our SAN model we are interested in representing any variation of the liquid level by 0.01 m, the deterministic activity *Level_variation* completes every 0.016667 h (in state configurations 1, 4, 6, 7 in Tab. 2) or every 0.016667/2 h (in state configuration 2 in Tab. 2). The gate *I_Level_variation* specifies the effect of the completion of the activity *Level_variation* as well: each time such activity completes, the marking of the place *Level* is increased by 0.01 (in the state configurations 1, 2, 4, 6 in Tab. 2) or is decreased by 0.01 (in the state configuration 7 in Tab. 2).

The place *normal* is initially marked with one token in order to represent that the liquid level is inside the region of correct functioning (HLA<H<HLB [1]). 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. 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 the place *alert* enables the immediate activity *control* to complete. The effect of its completion is ruled by the output gate *O_control* incorporating 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 according to the control laws in Tab. 3. So, the immediate activity *control* together with the output gate *O_control*, 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* has no effect on the place *P1_on*, *P2_on* or *V_on* respectively. In this way we model that the controller can not act on the state of a stuck component.

The controller action on the component states may lead the liquid level 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 the activity *a2n* is the presence of one token inside the place *normal* in order to represent that the liquid level in the tank is now correct.

The dry out and overflow conditions 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 system failure due to the dry out. If instead the marking of the place *Level* is greater than HLP, then the effect of the completion of *fail* is the presence of one token inside the place *overflow* modeling the occurrence of the system failure in case of overflow.

4.2 Modelling Version 2

In the Version 2 of the benchmark [1], the failure rates of P1, P2 and V change according to the current state of the component and the state reached as a consequence of the failure (Tab. 4). The other aspects of the system behavior do not change. The SAN model of the Version 2 of the benchmark appears in Fig. 5; such model differs from the one in Fig. 4 (Version 1) only for the part modeling the failure of P1, P2 and V.

In Fig. 5, the current state of P1 is still modeled by the marking of the places *P1_on* and *P1_stuck*. The state transition caused by the failure is now modeled by three stochastic activities: *P1_on_fail*, *P1_off_fail_off* and *P1_off_fail_on*. According to Tab. 4, the failure rate of P1 in state ON does

Activity:	*Level_variation*
type:	deterministic
value:	0.016667/fabs(P1_on->Mark()+P2_on->Mark()-V_on->Mark()) h
input gate:	*I_Level_variation*
input gate predicate:	(P1_on->Mark() + P2_on->Mark() - V_on->Mark() != 0) && (Level->Mark() >= HLV) && (Level->Mark()<=HLP)
input gate function:	if (P1_on->Mark() + P2_on->Mark() - V_on->Mark() > 0) Level->Mark()=Level->Mark()+0.01; if (P1_on->Mark() + P2_on->Mark() - V_on->Mark() < 0) Level->Mark()=Level->Mark()-0.01;
Activity:	*n2a*
type:	immediate
input gate:	*I_n2a*
input gate predicate	(Level->Mark() < HLA || Level->Mark() > HLB) && (normal->Mark()==1)
input gate function:	normal->Mark()=0; alert->Mark()=1
Activity:	*a2n*
type:	immediate
input gate:	*I_a2n*
input gate predicate	(Level->Mark() >= HLA) && (Level->Mark() <= HLB) && (normal->Mark()==0)
input gate function:	normal->Mark()=1; alert->Mark()=0;

Table 6: The activities in the SAN model in Fig. 4 (Version 1).

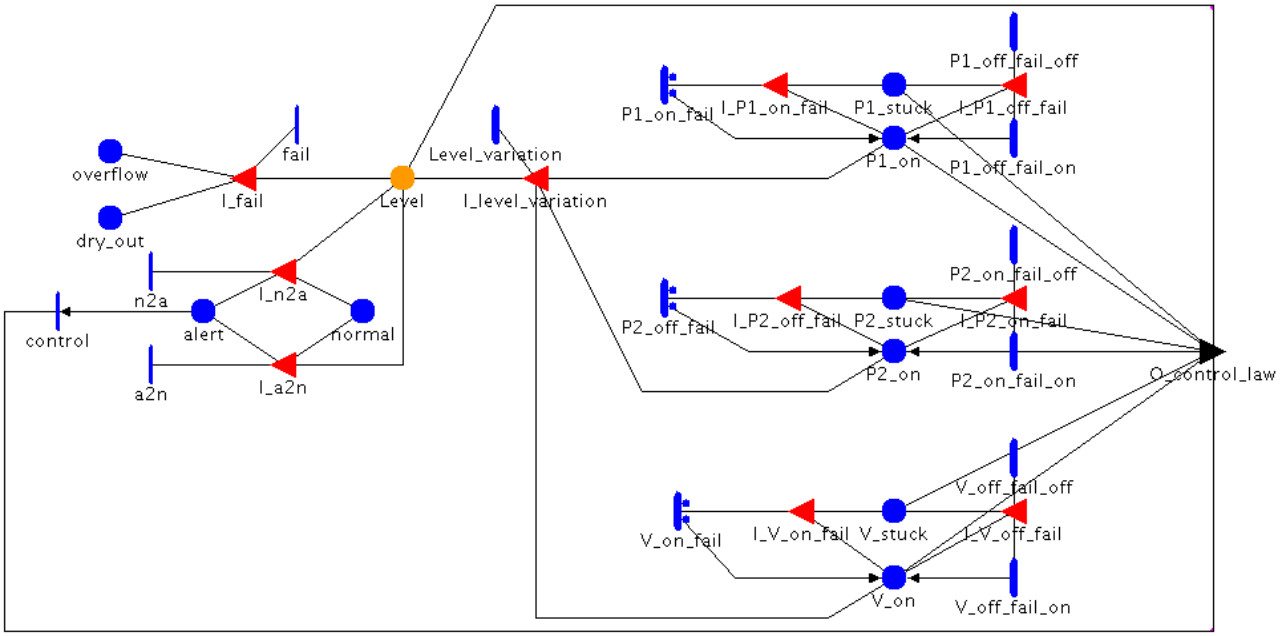

Figure 5: The SAN model of the Version 2.

Activity:	*control*
type:	immediate
output gate:	*O_control*

output gate function:	if (Level->Mark() < HLA) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=1;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=1;
	if (V_stuck->Mark()==0)
	V_on->Mark()=0; }
	if (Level->Mark() > HLB) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=0;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=0;
	if (V_stuck->Mark()==0)
	V_on->Mark()=1; }

Activity:	*fail*
type:	immediate
input gate:	*I_fail*
input gate predicate	(Level->Mark() > HLP || Level->Mark() < HLV) && (overflow->Mark()==0) && (dry_out->Mark()==0)

input gate function:	if (Level->Mark() > HLP)
	overflow->Mark()=1;
	if (Level->Mark() < HLV)
	dry_out->Mark()=1;

Activity:	*P1_fail*
type:	stochastic
rate:	0.004566 h^{-1}
input gate:	*I_P1_fail*
input gate predicate:	P1_stuck->Mark()==0
input gate function:	P1_stuck->Mark()=1; P1_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*P2_fail*
type:	stochastic
rate:	0.005714 h^{-1}
input gate:	*I_P2_fail*
input gate predicate:	P2_stuck->Mark()==0
input gate function:	P2_stuck->Mark()=1; P2_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*V_fail*
type:	stochastic
rate:	0.003125 h^{-1}
input gate:	*I_V_fail*
input gate predicate:	V_stuck->Mark()==0
input gate function:	V_stuck->Mark()=1; V_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5

Table 7: The activities in the SAN model in Fig. 4 (Version 1).

not change according to the state reached as a consequence of the failure. The stochastic activity *P1_on_fail* models the failure of P1 during the state ON: its input gate *I_P1_fail_on* enables its completion only while the place *P1_on* is marked, and in case of completion the gate sets the marking of *P1_on* to 0 and the marking of *P1_stuck* to 1. The effect of the completion is specified also by two completion cases associated with the activity *P1_on_fail*: in one case the marking of *P1_on* remains 0 (state transition to stuck OFF), in the other case the marking of *P1_on* becomes 1 (state transition to stuck ON). The completion rate of *P1_on_fail* is 0.004566 h^{-1}, while the probability of its completion cases is uniformly distributed (Tab. 4).

While P1 is in state OFF instead, its failure rate changes according to the state determined by the failure, stuck ON or stuck OFF (Tab. 4). For this reason, we use the stochastic activity *P1_off_fail_off* modeling the failure during the state OFF and leading to the state stuck OFF, and we use the stochastic activity *P1_off_fail_on* modeling the failure during the state OFF and leading to the state stuck ON. Both activities are enabled to complete by the input gate *P1_off_fail* while the place *P1_on* is empty (P1 is OFF). The effect of both activities is the presence of one token inside the place *P1_stuck* (P1 is stuck), but the activity *P1_off_fail_off* does not change the marking of *P1_on* (P1 is stuck OFF), while the activity *P1_off_fail_on* sets the marking of *P1_on* to 1 (P1 is stuck ON). The completion rate of *P1_off_fail_off* is 0.045662 h^{-1}, while the completion rate of *P1_off_fail_off* is 0.456621 h^{-1} (Tab. 4).

In the case of P2, during the state OFF, the failure rate is the same for the state transition toward the state stuck ON and the transition toward stuck OFF (Tab. 4). This is modeled by the stochastic activity *P2_off_fail* enabled by the input gate *I_P2_off_fail* while the place *P2_off* contains one token. The effect of its completion is the presence of one token inside the place *P2_stuck* (stuck condition), and the presence of one token inside the place *P2_on* (state stuck ON) or in the place *P2_off* (state stuck OFF). The completion rate of the stochastic activity *P2_off_fail* is 0.005714 h^{-1} (Tab. 4).

The failure rate of P2 in state ON changes according to the state reached as a consequence of the failure (Tab. 4). The failure of P2 in state ON leading to the state stuck ON is modeled by the stochastic activity *P2_on_fail_on* whose completion rate is 0.057142 h^{-1}, while the failure in state ON leading to the state stuck OFF is represented by the stochastic activity *P2_on_fail_off* whose failure rate is 0.571429 h^{-1}.

In the case of the valve V, the failure in state ON leading to the state stuck ON or stuck OFF

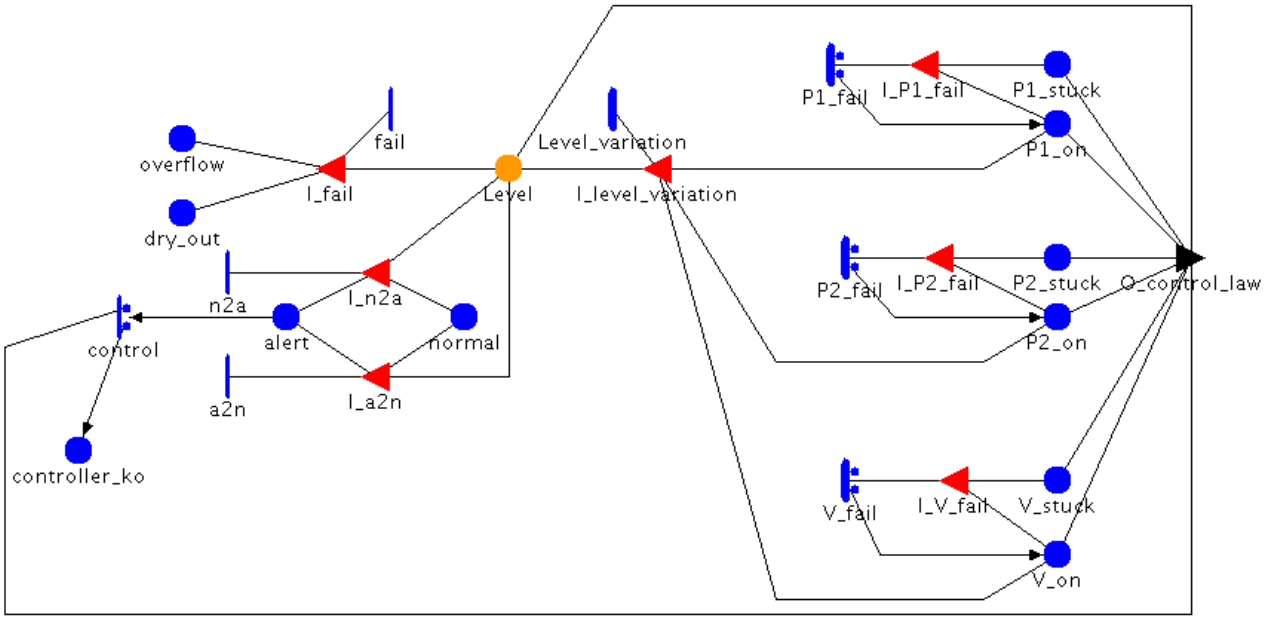

Figure 6: The SAN model of the Version 3.

according to the same failure rate, is modeled by the stochastic activity *V_on_fail* whose completion rate is 0.003125 h^{-1} (Tab. 4). The failure of V during the state OFF leading to the state stuck ON is modeled by the stochastic activity *V_off_fail_on* whose failure rate is 0.03125 h^{-1}; the failure of V during the state OFF leading instead to the state stuck OFF is represented by the stochastic activity *V_off_fail_off* whose failure rate is 0.3125 h^{-1}.

The activities and the gates introduced in the SAN model in Fig. 5 are detailed in Tab. 8 and Tab. 9. The other parts of the SAN model in Fig. 5 are the same as in the model in Fig. 4 concerning the Version 1 of the benchmark.

4.3 Modelling Version 3

The Version 3 of the benchmark is modeled by the SAN shown in Fig. 6. In such model the controller failure on demand is introduced: if one token appears in the place *alert* (the liquid level is outside the region of correct functioning), then the immediate activity *control* is enabled to complete. Such activity represents the action of the controller as in Fig. 4, but in Fig. 6 the activity has two completion cases: in one case, the effect of the completion of the activity *control* is ruled by the output gate *O_control* incorporating the control laws (Tab. 2) and acting on the state of the components attempting to avoid the dry out or the overflow of the liquid in the tank. In the other case, the failure on demand of the controller occurs and one token is added into the place *controller_ko*: no command is sent from

Activity:	*P1_on_fail*
type:	stochastic
rate:	0.004566 h^{-1}
input gate:	*I_P1_on_fail*
input gate predicate:	(P1_stuck->Mark()==0) && (P1_on->Mark()==1)
input gate function:	P1_stuck->Mark()=1; P1_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*P1_off_fail_off*
type:	stochastic
rate:	0.045662 h^{-1}
input gate:	*I_P1_off_fail*
input gate predicate:	(P1_stuck->Mark()==0) && (P1_on->Mark()==0)
input gate function:	P1_stuck->Mark()=1; P1_on->Mark()=0;
Activity:	*P1_off_fail_on*
type:	stochastic
rate:	0.456621 h^{-1}
input gate:	*I_P1_off_fail*
input gate predicate:	(P1_stuck->Mark()==0) && (P1_on->Mark()==0)
input gate function:	P1_stuck->Mark()=1; P1_on->Mark()=0;
Activity:	*P2_off_fail*
type:	stochastic
rate:	0.057142 h^{-1}
input gate:	*I_P2_off_fail*
input gate predicate:	(P2_stuck->Mark()==0) && (P2_on->Mark()==0)
input gate function:	P2_stuck->Mark()=1; P2_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*P2_on_fail_off*
type:	stochastic
rate:	0.571429 h^{-1}
input gate:	*I_P2_on_fail*
input gate predicate:	(P2_stuck->Mark()==0) && (P2_on->Mark()==1)
input gate function:	P2_stuck->Mark()=1; P2_on->Mark()=0;
Activity:	*P2_on_fail_on*
type:	stochastic
rate:	0.005714 h^{-1}
input gate:	*I_P2_on_fail*
input gate predicate:	(P2_stuck->Mark()==0) && (P2_on->Mark()==1)
input gate function:	P2_stuck->Mark()=1; P2_on->Mark()=0;

Table 8: The activities in the SAN model in Fig. 5 (Version 2).

Activity:	*V_on_fail*
type:	stochastic
rate:	0.003125 h^{-1}
input gate:	*I_V_on_fail*
input gate predicate:	(V_stuck->Mark()==0) && (V_on->Mark()==1)
input gate function:	V_stuck->Mark()=1;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*V_off_fail_off*
type:	stochastic
rate:	0.03125 h^{-1}
input gate:	*I_V_off_fail*
input gate predicate:	(V_stuck->Mark()==0) && (V_on->Mark()==0)
input gate function:	V_stuck->Mark()=1;
	V_on->Mark()=0;
Activity:	*V_off_fail_on*
type:	stochastic
rate:	0.3125 h^{-1}
input gate:	*I_V_off_fail*
input gate predicate:	(V_stuck->Mark()==0) && (V_on->Mark()==0)
input gate function:	V_stuck->Mark()=1;
	V_on->Mark()=0;

Table 9: The activities in the SAN model in Fig. 5 (Version 2).

the controller to P1, P2 and V, even though the liquid level is outside the region of correct functioning.

The complete specification of the activity *control* in the SAN model in Fig. 6 is reported in Tab. 10. The other parts of the model in Fig. 6 are the same as in Fig. 4 (Version 1).

4.4 Modelling Version 4

This version of the benchmark is modeled in Fig. 7 where the place *grace* is used to indicate that the grace period has begun or not: as soon as the liquid level exceeds the correct region (input gate *I_n2a*), the immediate activity *n2a* completes. This determines the appearance of one token inside the place *grace*. From now on, the stochastic activities *P1_repair*, *P2_repair* and *V_repair* are enabled to complete by the input gates *I_P1_repair*, *I_P2_repair* and *I_V_repair* respectively.

The repair of the pump P1 is modeled by the stochastic activity *P1_repair* and in particular by the corresponding input gate *I_P1_repair*. The effect of its completion is removing the token inside the place *P1_stuck* if any (if the stuck condition is holding for P1). In this way, we model that P1 is not stuck any more and it can obey again to the orders coming from the controller. In particular, if the liquid level is outside the correct region when the repair of P1 ends, then P1 may have to change immediately its state according to the control laws in Tab. 3. This is specified inside the input gate *I_P1_repair*.

Activity:	*control*
type:	immediate
case 1 probability:	0.9
case 2 probability:	0.1
output gate (case 1):	*O_control*

output gate function:	if (Level->Mark() < HLA) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=1;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=1;
	if (V_stuck->Mark()==0)
	V_on->Mark()=0; }
	if (Level->Mark() > HLB) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=0;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=0;
	if (V_stuck->Mark()==0)
	V_on->Mark()=1; }

Table 10: The activities in the SAN model in Fig. 6 (Version 3).

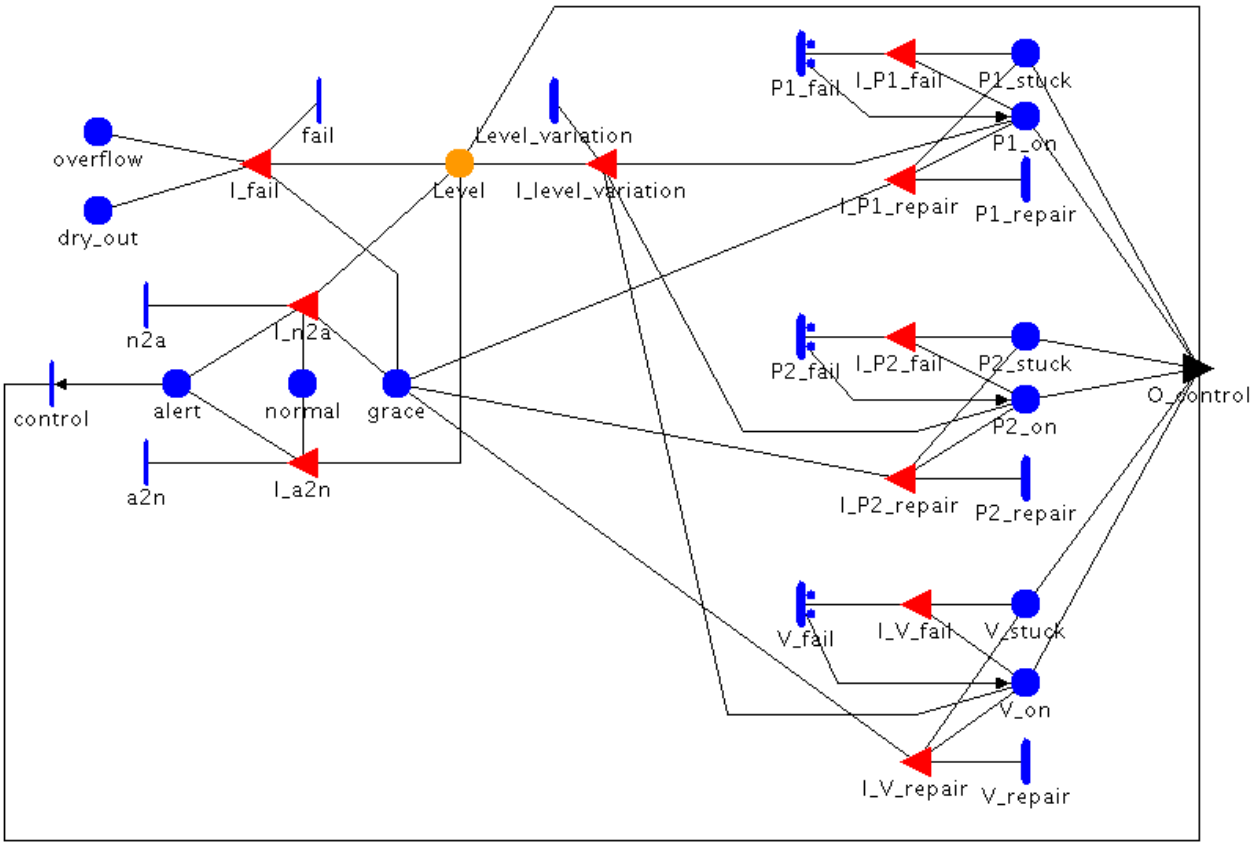

Figure 7: The SAN model of the Version 4.

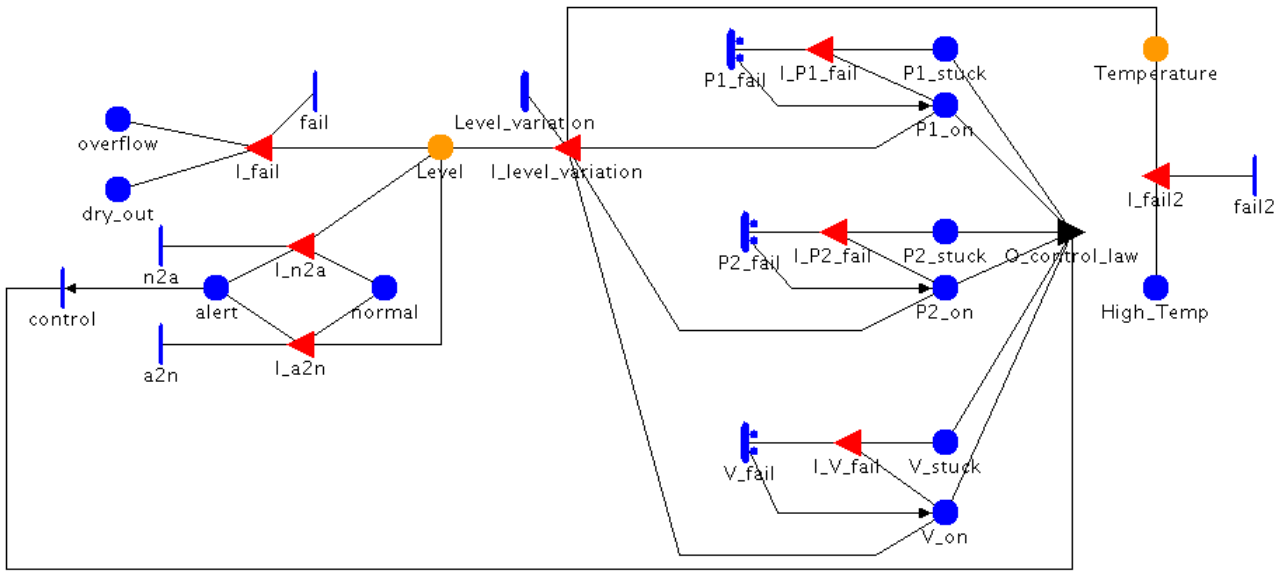

Figure 8: The SAN model of the Version 5.1.

The repair of the components P2 and V is modeled in a similar way by the stochastic activity *P2_repair* and the input gate *I_P2_repair* in the case of P2, and by the stochastic activity *V_repair* and the input gate *I_V_repair* in the case of V. The stochastic activities *P1_repair*, *P2_repair* and *V_repair* have the same completion rate corresponding to the repair rate: 0.2 h^{-1}.

In the SAN model in Fig. 7, the system failure due to such conditions is still modeled by the immediate activity *fail* as in the SAN model of the Version 1 (Fig. 4). The effect of its completion includes now the remotion of the token inside the place *grace* in order to model that the grace period ends when the system failure occurs.

The other parts of the model in Fig. 7 are the same as in Fig. 4 (Version 1). The details about the new activities introduced in the SAN model in Fig. 7 are reported in in Tab. 11.

4.5 Modelling Version 5

Version 5.1. The SAN model representing the Version 5.1 of the benchmark is depicted in Fig. 8. With respect to the SAN model of the Version 1 (Fig. 4), a new extended place called *Temperature* is now used in order to represent the current value of the liquid temperature. Such place is initially set to 15.66667, corresponding to the initial temperature of the liquid. The extended place *Level* is initially set to 7 because of the different level thresholds in the Version 5.1 of the benchmark.

The deterministic activity *Level_variation* still models the variation of the liquid level as in the SAN model of the Version 1 (Fig. 4), but in the SAN model of the Version 5.1 (Fig. 8) such activity

Activity:	*P1_repair*
type:	stochastic
rate:	0.2 h^{-1}
input gate:	*I_P1_repair*
input gate predicate:	(grace->Mark()==1) && (P1_stuck->Mark()==1)

input gate function:	P1_stuck->Mark()=0;
	if (Level->Mark() < HLA)
	  P1_on->Mark()=1;
	if (Level->Mark() > HLB)
	  P1_on->Mark()=0;

Activity:	*P2_repair*
type:	stochastic
rate:	0.2 h^{-1}
input gate:	*I_P2_repair*
input gate predicate:	(grace->Mark()==1) && (P2_stuck->Mark()==1)

input gate function:	P2_stuck->Mark()=0;
	if (Level->Mark() < HLA)
	  P2_on->Mark()=1;
	if (Level->Mark() > HLB)
	  P2_on->Mark()=0;

Activity:	*V_repair*
type:	stochastic
rate:	0.2 h^{-1}
input gate:	*I_V_repair*
input gate predicate:	(grace->Mark()==1) && (V_stuck->Mark()==1)

input gate function:	V_stuck->Mark()=0;
	if (Level->Mark() < HLA)
	  V_on->Mark()=0;
	if (Level->Mark() > HLB)
	  V_on->Mark()=1;

Activity:	*n2a*
type:	immediate
input gate:	*I_n2a*
input gate predicate:	(Level->Mark() < HLA || Level->Mark() > HLB) && (normal->Mark()==1)

input gate function:	normal->Mark()=0;
	alert->Mark()=1;
	grace->Mark()=1;

Activity:	*fail*
type:	immediate
input gate:	*I_fail*
input gate predicate:	(Level->Mark() > HLP || Level->Mark() < HLV) && (overflow->Mark()==0) && (dry_out->Mark()==0)

input gate function:	if (Level->Mark() > HLP)
	  overflow->Mark()=1;
	if (Level->Mark() < HLV)
	  dry_out->Mark()=1;
	grace->Mark()=0

Table 11: The activities in the SAN model in Fig. 7 (Version 4).

models also the temperature variation as a consequence of the heat source and of the injection of some new liquid in the tank by the pumps. For this reason, with respect to the SAN model of the Version 1, the activity *Level_variation* is always enabled to complete by the input gate *I_Level_variation*. This leads to a relevant increase of the computational cost of the simulation. In order to reduce such cost, in the SAN model of the Version 5.1, the liquid level is increased or decreased by 0.1 m instead of 0.01 m as in the SAN model of the Version 1. Since in the Version 5.1 of the benchmark the level variation rate due to the action of P1, P2 or V is 1.5 m/h, the liquid level is increased (decreased) by 0.1 m every 0.066667 h (Eq. 1). So, the completion time of the activity *Level_variation* is set to 0.066667 h.

The effect of the completion of the activity *Level_variation* is ruled by the corresponding input gate *I_Level_variation*. Since the power of the heat source is 1 $m°C/h$ and the heat is uniformly distributed on the liquid, every 0.066667 h the heat source causes an increase of the temperature by 0.066667°/H, where H is the current level of the liquid in the tank. To model this, every time the activity *Level_variation* completes (every 0.066667 h), the marking of the extended place *Temperature* is increased by 0.066667/*Level->Mark()*, where *Level->Mark()* is the current marking of the extended place *Level*.

The completion of the activity *Level_variation* models the action of P1, P2 and V on the liquid level: if the place *P1_on* contains one token (P1 is currently ON), the marking of the extended place *Level* is increased by 0.1 (the liquid level is increased by 0.1 m); at the same time, the marking of the extended place *Temperature* is updated according to Eq. 2. In this way, we model the increase of the liquid level caused by P1, together with the temperature variation due to the injection by P1 of some new liquid in the tank. In a similar way, if the place *P2_on* is marked, the marking of the place *Level* is increased by 0.1, and the value of the marking of the place *Temperature* is updated still according to Eq. 2. The valve V only removes liquid from the tank, so if the place *V_on* is marked, then the marking of the place *Level* is decreased by 0.1.

Another difference between the SAN model of the Version 5.1 (Fig. 8) and the SAN model of the Version 1 (Fig. 4) concerns the failure rate of the components P1, P2 and V: in the SAN model of the Version 5.1, the failure rate of the stochastic activities *P1_fail*, *P2_fail* and *V_fail* modeling the failure of P1, P2 and V respectively, is expressed as a function of the marking of the extended place *Temperature* according to Eq. 3.

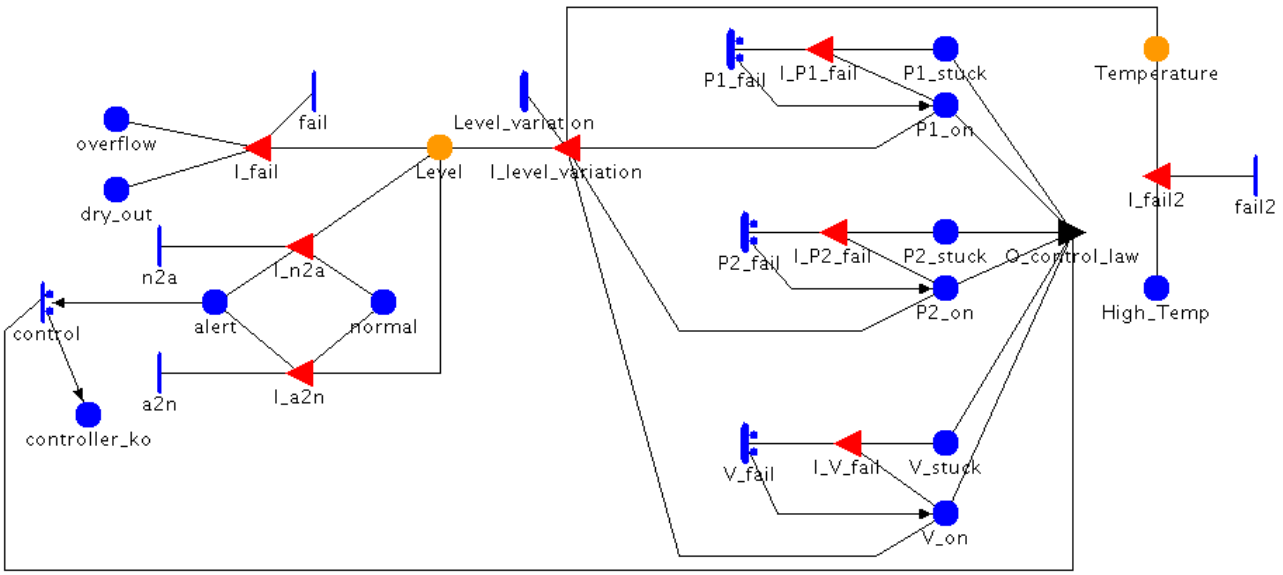

Figure 9: The SAN model of the Version 5.2.

In the Version 5.1 of the benchmark, the system failure condition due to the high temperature of the liquid is introduced. In the SAN model in Fig. 8, such condition is detected by the immediate activity *fail2* ruled by the corresponding input gate *I_fail2*: when the marking of the extended place *Temperature* reaches the value of 100, one token appears inside the place *High_Temp*. The other failure conditions (dry out and overflow) are detected in the same way as in the SAN model of the Version 1, but according to different thresholds for the liquid level.

The details about the activities and the gates characterizing the SAN model in Fig. 8 are reported in in Tab. 12.

Version 5.2. The Version 5.2 of the benchmark is obtained by extending the Version 5.1 with the introduction of the failure on demand of the controller. The SAN model of Version 5.2 is shown in Fig. 9 and differs from the SAN model in Fig. 8 (Version 5.1) for the immediate activity *control* having now two completion cases modeling the correct functioning of the controller or its failure on demand respectively. Such activity has the same behavior as in the SAN model in Fig. 6 (Version 3), but the probabilities of the completion cases are now 0.8 (case of the correct functioning of the controller) and 0.2 (case of the controller failure on demand).

Version 5.3. Fig. 10 shows the SAN model for the Version 5.3 of the benchmark. Here the failure on demand probability is a function of the number of demands. The marking of the place *demands* indicates the number of demands: each time the immediate activity *n2a* completes (the liquid reaches

Activity:	*P1_fail*
type:	stochastic
rate:	0.004566 $*$ (0.2 $*$ exp(0.005756$*$(Temperature->Mark()-20)) + 0.8 $*$ exp(-0.2301$*$(Temperature->Mark()-20))) h^{-1}
input gate:	*I_P1_fail*
input gate predicate:	P1_stuck->Mark()==0
input gate function:	P1_stuck->Mark()=1; P1_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*P2_fail*
type:	stochastic
rate:	0.005714 $*$ (0.2 $*$ exp(0.005756$*$(Temperature->Mark()-20)) + 0.8 $*$ exp(-0.2301$*$(Temperature->Mark()-20))) h^{-1}
input gate:	*I_P2_fail*
input gate predicate:	P2_stuck->Mark()==0
input gate function:	P2_stuck->Mark()=1; P2_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*V_fail*
type:	stochastic
rate:	0.003125 $*$ (0.2 $*$ exp(0.005756$*$(Temperature->Mark()-20)) + 0.8 $*$ exp(-0.2301$*$(Temperature->Mark()-20))) h^{-1}
input gate:	*I_V_fail*
input gate predicate:	V_stuck->Mark()==0
input gate function:	V_stuck->Mark()=1; V_on->Mark()=0;
case 1 probability:	0.5
case 2 probability:	0.5
Activity:	*Level_variation*
type:	deterministic
value:	0.066667 h
input gate:	*I_Level_variation*
input gate predicate:	(Level->Mark() >= HLA) && (Level->Mark() <= HLB) && (Temperature->Mark() < 100)

input gate function:	Temperature->Mark()=Temperature->Mark() + 0.066667/Level->Mark();
	if (P1_on->Mark() > 0) {
	Temperature->Mark()=(Temperature->Mark() $*$ Level->Mark() + T_in $*$ 0.1) / (Level->Mark()+0.1);
	Level->Mark()=Level->Mark()+0.1; }
	if (P2_on->Mark() > 0) {
	Temperature->Mark()=(Temperature->Mark() $*$ Level->Mark() + T_in $*$ 0.1) / (Level->Mark()+0.1);
	Level->Mark()=Level->Mark()+0.1; }
	if (V_on->Mark() > 0)
	Level->Mark()=Level->Mark()-0.1;

Activity:	*fail2*
type:	immediate
input gate:	*I_fail2*
input gate predicate	(Temperature->Mark() >= 100) && (High_Temp->Mark()==0)
input gate function:	High_Temp->Mark()=1;

Table 12: The activities in the SAN model in Fig. 8 (Version 5.1).

Activity:	*control*
type:	immediate
case 1 probability:	0.8
case 2 probability:	0.2
output gate (case 1):	*O_control*

output gate function:	if (Level->Mark() < HLA) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=1;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=1;
	if (V_stuck->Mark()==0)
	V_on->Mark()=0; }
	if (Level->Mark() > HLB) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=0;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=0;
	if (V_stuck->Mark()==0)
	V_on->Mark()=1; }

Table 13: The activities in the SAN model in Fig. 9 (Version 5.2).

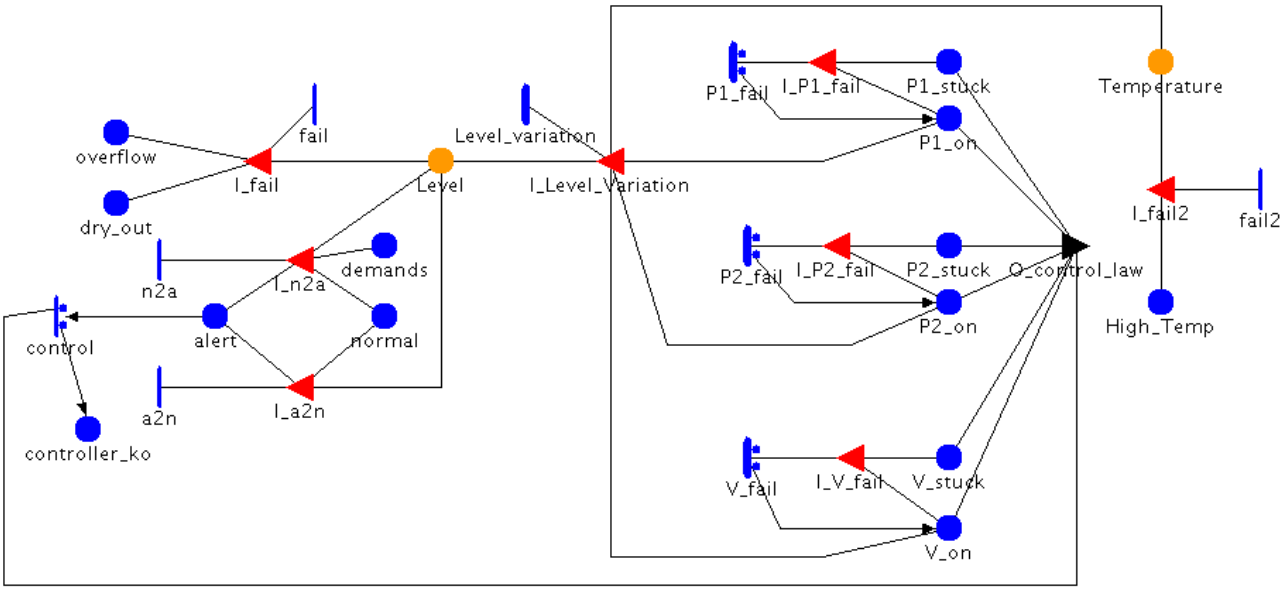

Figure 10: The SAN model of the Version 5.3.

Activity:	*n2a*
type:	immediate
input gate:	*I_n2a*
input gate predicate	(Level->Mark() < HLA || Level->Mark() > HLB) && (normal->Mark()==1)

input gate function:	normal->Mark()=0;
	alert->Mark()=1;
	demands->Mark()=demands->Mark()+1;

Activity:	*control*
type:	immediate
case 1 probability:	1-0.2∗pow(1.5, demands->Mark()-1)
case 2 probability:	0.2∗pow(1.5, demands->Mark()-1)
output gate (case 1):	*O_control*

output gate function:	if (Level->Mark() < HLA) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=1;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=1;
	if (V_stuck->Mark()==0)
	V_on->Mark()=0; }
	if (Level->Mark() > HLB) {
	if (P1_stuck->Mark()==0)
	P1_on->Mark()=0;
	if (P2_stuck->Mark()==0)
	P2_on->Mark()=0;
	if (V_stuck->Mark()==0)
	V_on->Mark()=1; }

Table 14: The activities in the SAN model in Fig. 10 (Version 5.3).

the control boundaries), the marking of such place is increased by one. The immediate activity *control* still has two completion cases: the correct functioning of the controller, and its failure on demand; the probabilities of such cases are set to $1 - 0.2 \cdot 1.5^{demands->Mark()-1}$ and to $0.2 \cdot 1.5^{demands->Mark()-1}$ respectively. In this way, the probability of failure on demand is increased by 50% after each demand.

The details about the activity *control* in the SAN model in Fig. 9 (Version 5.2) and in Fig. 10 (Version 5.3) are reported in Tab. 13 and in Tab. 14 respectively.

5 Model simulation

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 dry out probability, and the cdf of the overflow probability, for a mission time varying between 0 and 1000 h. The cdf provides the system unreliability (Sec. 1) as a consequence of the dry out or of the overflow. For instance, the value of the

place	Fig. 4	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 9	Fig. 10
P1_on	1	1	1	1	1	1	1
P1_stuck	0	0	0	0	0	0	0
P2_on	0	0	0	0	0	0	0
P2_stuck	0	0	0	0	0	0	0
V_on	1	1	1	1	1	1	1
V_stuck	0	0	0	0	0	0	0
Level	0	0	0	0	7	7	7
normal	1	1	1	1	1	1	1
alert	0	0	0	0	0	0	0
dry_out	0	0	0	0	0	0	0
overflow	0	0	0	0	0	0	0
controller_ko	-	-	0	-	-	0	0
grace	-	-	-	0	-	-	-
Temperature	-	-	-	-	15.666667	15.666667	15.666667
demands	-	-	-	-	-	-	0

Table 15: The initial marking of each place in every SAN model. *Level* and *Temperature* are extended places.

dry out cdf at time $t > 0$ is the probability that the system has failed because of the dry out, during the time period $(0, t)$.

Dry out condition. 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. 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. 4). 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 a certain time.

Overflow condition. The cdf of the overflow probability is computed in the same way as the cdf of the dry out probability, but with reference to the place *overflow* present in all the SAN models: the marking of the place *overflow* is equal to 0 if the overflow has not occurred, or it is equal to 1 if the overflow condition holds (Sec. 4).

5.1 Results for Versions 1, 2, 3

The values of the cdf of the dry out in Versions 1, 2 and 3, for a mission time varying between 0 and 1000 h, are reported in Tab. 16 and are graphically compared in Fig. 11.a. The values of the cdf of the overflow in Versions 1, 2 and 3, for a mission time varying between 0 and 1000 h, are reported in Tab. 17 and are graphically compared in Fig. 11.b. The results for Versions 1, 2 and 3, returned by

time	Version 1	Version 2	Version 3
200 h	2,2390E-02	4,0400E-02	8,6710E-02
400 h	6,5990E-02	6,3360E-02	1,2664E-01
600 h	9,5290E-02	7,3750E-02	1,4707E-01
800 h	1,1003E-01	7,8340E-02	1,5739E-01
1000 h	1,1747E-01	8,0240E-02	1,6220E-01

Table 16: The cdf values for the dry out condition in Versions 1, 2, 3.

time	Version 1	Version 2	Version 3
200 h	1,9914E-01	1,6852E-01	2,7244E-01
400 h	3,6207E-01	2,7882E-01	4,2492E-01
600 h	4,3665E-01	3,2938E-01	4,8808E-01
800 h	4,7063E-01	3,5284E-01	5,1537E-01
1000 h	4,8572E-01	3,6500E-01	5,2797E-01

Table 17: The cdf values for the overflow condition in Versions 1, 2, 3.

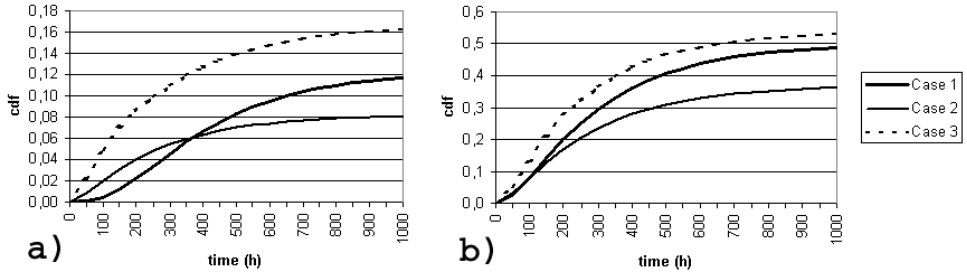

Figure 11: The cdf curves for the dry out condition (a) and the overflow condition (b), in Versions 1, 2, 3.

the SAN models simulation for both the dry out and overflow condition, are similar to those obtained by the Monte Carlo simulation in [2].

5.2 Results for Version 4

The results obtained by simulating the model in Fig. 7, for a mission time varying between 0 and 500 h, are reported in Tab. 18, in Fig. 12.a (dry out) and in Fig. 12.b (overflow). Such results differ from those obtained by Monte Carlo simulation in [2], even though they are of the same order of magnitude: 10^{-4} for the dry out at 500 h, and 10^{-2} for the overflow at the same time. Because of the different values of HLV and HLP in Version 4, the unreliability values can not be compared with those obtained in the previous versions.

time	dry out	overflow
100 h	6,000E-05	2,430E-03
200 h	2,200E-04	6,090E-03
300 h	3,700E-04	9,460E-03
400 h	4,500E-04	1,197E-02
500 h	5,100E-04	1,363E-02

Table 18: The cdf values for the dry out and overflow conditions in Version 4.

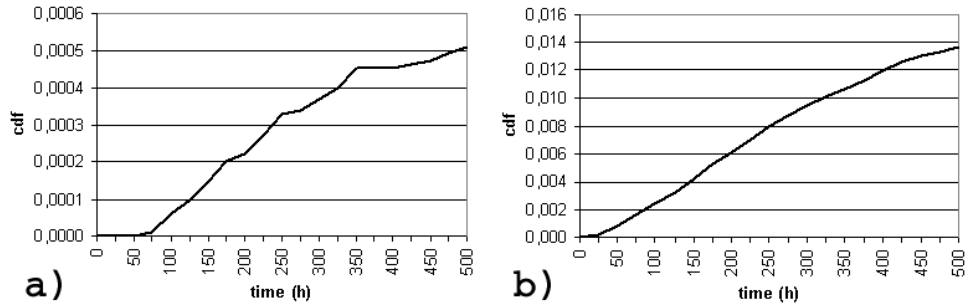

Figure 12: The cdf curves for the dry out condition (a) and the overflow condition (b), in Version 4.

5.3 Results for Version 5

The cdf of the dry out and the cdf of the overflow have been computed as in the previous versions of the benchmark. In Versions 5.1, 5.2 and 5.3, another system failure condition is taken into account: the temperature of the liquid reaching 100°. The cdf of such condition at time t is computed as the mean number of tokens inside the place *High_Temp* at time t, over 100'000 simulation batches. Such place is present in the SAN models in Figures 8 (Version 5.1), 9 (Version 5.2), 10 (Version 5.3). Its marking is initially equal to 0 and turns to 1 if the high temperature condition is reached (Sec. 4.5). So the mean number of tokens at time t inside the place *High_Temp* varies between 0 and 1 and provides the cdf of the high temperature failure condition. The results obtained in Versions 5.1, 5.2 and 5.3 are reported in Tables 19 (dry out), 20 (overflow), 21 (high temperature). The results obtained for Version 5.1 (Fig. 13.a) and Version 5.2 (Fig. 13.b) are similar to those obtained by Monte Carlo simulation in [2]. According to the results of Version 5.3 (Fig. 13.c), the wear of the controller does not seem to have a relevant impact on the cdf values, with respect to Version 5.2. In [2] instead the controller wear determines a slight increase of the dry out cdf values.

time	Version 5.1	Version 5.2	Version 5.3
200 h	7,9330E-02	1,4934E-01	1,5201E-01
400 h	1,1706E-01	1,7271E-01	1,7470E-01
600 h	1,2376E-01	1,7650E-01	1,7849E-01
800 h	1,2436E-01	1,7685E-01	1,7886E-01
1000 h	1,2438E-01	1,7687E-01	1,7889E-01

Table 19: The cdf values for the dry out condition in Versions 5.1, 5.2, 5.3.

time	Version 5.1	Version 5.2	Version 5.3
200 h	3,9531E-01	4,7526E-01	4,7762E-01
400 h	4,8133E-01	5,4072E-01	5,4360E-01
600 h	4,9588E-01	5,5178E-01	5,5462E-01
800 h	4,9826E-01	5,5387E-01	5,5663E-01
1000 h	4,9884E-01	5,5428E-01	5,5708E-01

Table 20: The cdf values for the overflow condition in Versions 5.1, 5.2, 5.3.

time	Version 5.1	Version 5.2	Version 5.3
200 h	0,0000E+00	0,0000E+00	0,0000E00
400 h	0,0000E+00	2,0000E-05	1,0000E-05
600 h	3,8200E-02	2,0370E-02	1,9230E-02
800 h	1,1855E-01	6,7460E-02	6,5320E-02
1000 h	1,2724E-01	7,2550E-02	7,0460E-02

Table 21: The cdf values for the high temperature condition in Versions 5.1, 5.2, 5.3.

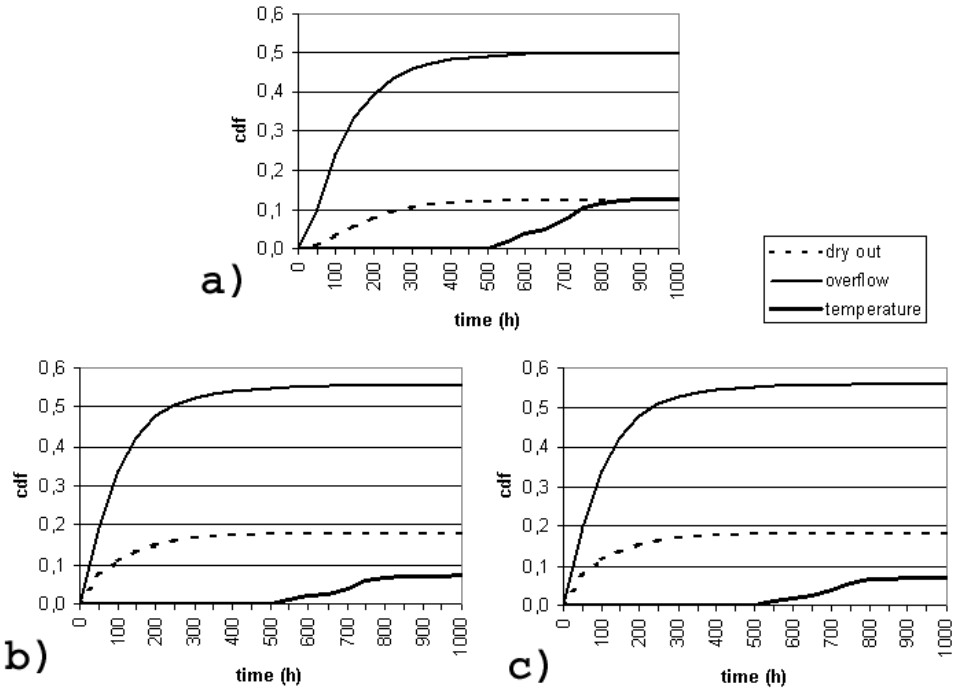

Figure 13: The cdf curves for the failure conditions in Versions 5.1 (a), 5.2 (b) and 5.3 (c).

6 Conclusions

A benchmark on dynamic reliability taken from the literature [2] has been evaluated. Several versions of such benchmark have been modeled and simulated in form of SAN. Each version focused on a particular aspect of the system behavior, such as state or temperature dependent failure rates, repairable components, failures on demand. The results we obtained are in general quite similar to those originally reported in [2] and computed by means of Monte Carlo simulation. Moreover, such results reflect those obtained with other approaches, as discussed in [1]; the advantages given by the use of SAN are described still in [1].

References

[1] D. Codetta-Raiteri. Modeling and simulating a benchmark on dynamic reliability as a Stochastic Activity Network. In *European Modeling & Simulation Symposium*, pages 545–554, Rome, Italy, September 2011.

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

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

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