

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/301642819>

# A Formal Approach for Maintainability and Availability Assessment Using Probabilistic Model Checking

Chapter · January 2016

DOI: 10.1007/978-3-319-33410-3\_21

CITATION

1

READS

127

4 authors:



**Abdelhakim Baouya**

Université Grenoble Alpes

13 PUBLICATIONS 17 CITATIONS

SEE PROFILE



**Djamal Bennouar**

Bouira University, Algeria

38 PUBLICATIONS 73 CITATIONS

SEE PROFILE



**Otmane Ait Mohamed**

Concordia University Montreal

183 PUBLICATIONS 741 CITATIONS

SEE PROFILE



**Samir Ouchani**

Concordia University Montreal

38 PUBLICATIONS 232 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Design and Verification of Time Sensitive Networks for Safety Critical Cyber Physical Systems [View project](#)



Software Product Line [View project](#)

# A Formal Approach for Maintainability and Availability Assessment Using Probabilistic Model Checking

Abdelhakim Baouya, Djamel Bennouar, Otmame Ait Mohamed and Samir Ouchani

**Abstract** Availability is one of the crucial characteristics that measures the system quality and influences the users and system designers. The aim of this work is to provide an approach to improve the system availability by taking into account different system situations at design step using SysML state machine diagram. We construct a formal model of state machine in the probabilistic calculus which supports modeling of concurrency and uncertainty. In this paper, we consider a single industrial control equipment and a multiprocessing computing platform where its behavior is modeled by SysML state machine diagram and we use logical specification of maintainability and availability properties. The probabilistic model checker PRISM has been used to perform quantitative analyses of these properties.

**Keywords** SysML state machine diagram · Reliability · Availability · Maintainability

## 1 Introduction

Constraints on system design in terms of reliability, availability and maintainability are becoming more stringent. For critical systems, availability constraints are having an increasing influence on the design and maintenance cost. It is necessary today, to

---

A. Baouya (✉)  
LIMPAF Lab, CS Department, University of Blida, Blida 1, Blida, Algeria  
e-mail: baouya.abdelhakim@gmail.com

D. Bennouar  
LIMPAF Lab, CS Department, University of Bouira, Bouira, Algeria  
e-mail: dbennouar@gmail.com

O.A. Mohamed  
ECE Department, Concordia University, Montreal, Canada  
e-mail: otmane.aitmohamed@concordia.ca

S. Ouchani  
SnT Center, University of Luxembourg, Luxembourg City, Luxembourg  
e-mail: samir.ouchani@uni.lu

take efficiently these constraints into consideration during the design process to reach a compliant solution. Thus, the evaluation and the prediction of system’s behavior at early stage of design is beneficial to handle time and effort.

The concept of reliability is quantifiable and suitable to describe the behavior in time of repairable systems according to the historical statistics [10], whereas, availability can be defined as the ratio of delivered service under given conditions at a stated instant of time [4]. Maintainability can be described as the ability of the system to be restored to a specified state following a failure [4]. The reliability functions are discussed in numerous publications [18, 20, 24] and we can state that the most approaches rely on the application of *Markov processes*. However, the quantification becomes complex when the number of system states is large [10]. Recently, the automated approaches such as Model Checking [5, 7, 21, 23] are used frequently for these purposes.

The *Model checking* [13] is a formal technique used to verify systems whose behavior is unpredictable, especially stochastic in nature. The technique consists on the exploration of every possible system behavior to check automatically that the specifications are satisfied. The verification can be focused on either qualitative or quantitative properties [1]. Quantitative properties puts the constraints on a certain event, e.g. the probability of processor failure in the next 3 h is at a least 0.88, while qualitative properties assert that certain event will happen surely (i.e. Probability = 1).

In this paper, we are interested in the formal verification of probabilistic systems modeled as SysML state machine diagram. The overview of our framework is depicted in Fig. 1. It takes State machine diagrams as input and constructs a formal model of state machine in the probabilistic calculus which supports modeling of concurrency and uncertainty. After that, we encode our model in language of the PRISM symbolic probabilistic model checker [15]. Commercial tools for reliability prediction, such as Lambda Predict [26], cannot be used to estimate the performance at the required moment as they do not support reward modeling (i.e. cost). However,

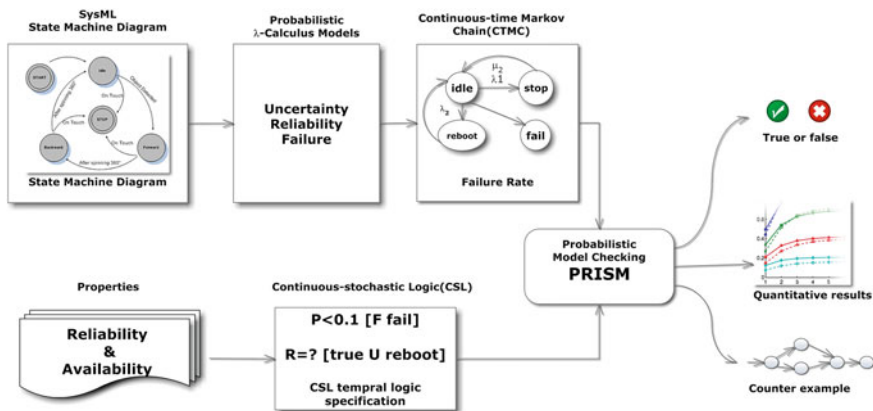


Fig. 1 A SysML State machine diagram verification approach

the probabilistic model checker PRISM overcomes this limitation. We analyze the logical properties expressed in Continuous-stochastic logic (CSL) [28] to check the system availability and maintainability.

The remainder of this paper is structured as follows: Sect. 2 discusses the related work. We recall the system life cycle states in Sect. 3. Section 4 describes SysML state machine diagram and its operational semantics. PRISM Model Checker is presented in Sect. 6. Section 7 illustrates the application of our mapping rules on case study for availability and maintainability assessment. Section 8 draws conclusions and lays out the future works.

## 2 Related Work

There are a numerous attempts to use formal methods to address the problems of design behavior prediction especially in automotive systems [6], industrial process control [10] and avionic navigation [17, 29]. Dhouibi et al. [6] presented a safety-based approach for system architecture optimization. The approach is based on Genetic Algorithm [27] for best components allocations. However the specification is not explained and the algorithm is time consuming. Huang et al. [12] proposed a verification of SysML State Machine Diagram by extending the model with MARTE [19] features to express the execution time. The tool has as input the State Machine Diagram and as output timed automata expressed in UPPAAL syntax [3]. UPPAAL uses Computational Tree Logic (CTL) properties to check if the model is satisfied with respect to liveness and safety properties. Morant et al. [20] proposed a Markov representation of availability and failure according to the statistical observations. However, the analysis did not refer to any relation between reliability and availability for safety interpretation. Nevertheless, Lu et al. [18] constructed a formal model of GNSS based positioning system directly in the probabilistic Pi-calculus that supports concurrency and uncertainty which is directly mapped to PRISM language for availability interpretation. Qiu et al. [25] used UML state chart with failure rates to evaluate the availability of systems where the approach is based on simulation method. However, the mapping rules and simulation tool are not clearly described in the paper. Liu et al. [17] used Architecture Analysis and Design Language (AADL) to describe the system architecture and used Error Model Annex (i.e. textual representation of state transitions) with the Risk-based Failure Mode Effect Analysis (RFMEA) property to express error effects. The developed plug-in extracts a set of measures for quantitative assessments. However, the authors restricted themselves to the generation of a set of failure rates only.

Compared to the existing works, our work maps a standard behavioral diagram called SysML State Machine into PRISM code. In addition, We construct a formal model of state machine in the probabilistic calculus which supports modeling of concurrency and uncertainty. Our goal is to adopt probabilistic model checking for system availability and maintainability assessment on the basis of the system reliability and the failure rate of its components. Table 1 highlights the comparison of our work with the existing approaches.

**Table 1** Comparison with the existing approaches

Paper	SysML	Approach	Formalization	Automation
[6]		Genetic algorithm		√
[12]	√	Model checking		√
[10]		Analytic		
[20]		Analytic		
[18]		Model checking	√	√
Our	√	Model checking	√	√

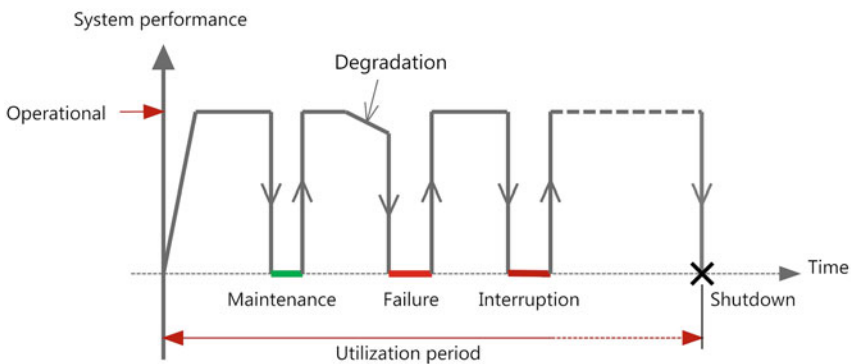
### 3 System Life-Cycle

As shown in Fig. 2, system life-cycle is a final loop that begins from execution to the shutdown and includes four states [10]; operational, failure, interruption (i.e. Accident) and maintenance. Three types of stops are considered: stop 1 after preventive maintenance, stop 2 after failure and stop 3 after an interruption:

- Stop 1 or maintenance time is the necessary time to perform the preventive maintenance represented by mean time to maintenance (MTTM).
- Stop 2 or repairing time is the time required to repair the system after breakdown represented by mean time to repair (MTTR).
- Stop 3 or preparing time is the time to restore the system to the operating state after an interruption represented by mean time to preparing (MTTP).

Availability therefore, is the probability for the system to function correctly at the required moment [16], the basic definition is:

$$Availability = \frac{Operational\ time}{Total\ utilization\ period}$$



**Fig. 2** System life-cycle [10]

In our case, availability is computed using CSL temporal logic in Sect. 7. Moreover, we adopt the following indicators cited in [10] to assess availability:

- MTBF: Mean Time Between Failures is the mean time between two consecutive failures.
- MTBM: Mean Time Between Maintenance is the mean time between two preventive maintenance.
- MTBI: Mean Time Between Interruptions (i.e. Accident) is the mean time between two interruptions.

## 4 SysML State Machine Diagram

SysML State Machine diagram (SMD) is a graph-based diagram to describe state-dependent behavior [22]. Table 2 shows the sub set of interesting artifacts used for verification in this paper and its corresponding algebraic expression and PRISM commands. A behavior starts (resp. stops) executing when its initial (resp. final) pseudo-state becomes active. We note that state behavior (i.e. do, entry and exit) are ignored in this paper. Transitions are defined by triggers and guards. The trigger (i.e. events) causes a transition from the source state when the guard is valid. In addition, the control node supports junction, choice, join, fork and terminate. A junction splits an incoming transition into multiple outgoing transitions and realizes a static conditional branch, as opposed to a choice pseudo-state which realizes a dynamic conditional branch. To illustrate how a rate value is specified, the transition leaving choice nodes are annotated with the  $\ll\textit{rate}\gg$  stereotype. We present in Definition 1, the formal definition of SysML state machine diagrams that embed the rate function.

**Definition 1** State machine diagrams is a tuple  $S = (i, \textit{fin}, \mathcal{N}, E, \textit{Rate})$ , where:

- $i$  is the initial node,
- $\textit{fin} = \{\odot, \times\}$  is the set of final nodes,
- $\mathcal{N}$  is a finite set of state machine nodes,
- $E$  is a set of events (i.e. triggers),
- $\textit{Rate}: (\{i\} \cup \mathcal{N}) \times E \times (\textit{fin} \cup \mathcal{N}) \rightarrow \mathbb{R}_{\geq 0}$  is a rate function assigning for each transition from  $(\{i\} \cup \mathcal{N})$  to  $(\textit{fin} \cup \mathcal{N})$  and  $\alpha \in E$  a positive real value  $\lambda$ .

## 5 State Machine Syntax

We formalize state machine diagrams by developing a calculus where its terms are presented in Table 2 according to the graphical notation defined in the standard. Using this calculus, a marked term is typically used to denote a reachable configuration, whereas, the unmarked term corresponds to the static structure of the diagram.

**Table 2** Formal notation of SysML state machine diagram artifacts



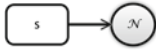
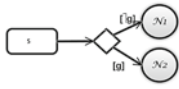

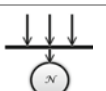
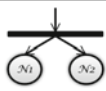
Artifacts	Formal notation	PRISM representation
	$l : i \mapsto \mathcal{N}$	$\square Initial \rightarrow (Initial' = false) \& (\mathcal{N}' = true);$
	$l : \odot$	$\square Final \rightarrow (Final' = false) \& \dots \& (\mathcal{N}'_i = true);$
	$l : s \mapsto \mathcal{N}$	$\square S \rightarrow (S' = false) \& (\mathcal{N}' = true);$
	$l : D(\lambda_1, g, \mathcal{N}_1, \lambda_2, \mathcal{N}_2)$	$[l_g] g_1 \rightarrow \lambda_1 : (l'_g = false) \& (\mathcal{N}'_1 = true);$ $[l_{!g}] g_2 \rightarrow \lambda_2 : (l'_{!g} = false) \& (\mathcal{N}'_2 = true);$
	$l : M(x) \mapsto \mathcal{N}$	$\square l_{x1} \rightarrow (l'_{x1} = false) \& (M'_x = true);$ $\square l_{x2} \rightarrow (l'_{x2} = false) \& (M'_x = true);$ $\square M'_x \rightarrow (M'_x = false) \& (\mathcal{N}' = true);$
	$l : J(x) \mapsto \mathcal{N}$	$\square l_{x1} \& l_{x2} \rightarrow (l'_{x1} = false) \& (l'_{x2} = false) \& (\mathcal{N}' = true);$
	$l : F(\mathcal{N}_1, \mathcal{N}_2)$	$\square Fork \rightarrow (Fork' = false) \& (\mathcal{N}'_1 = true) \& (\mathcal{N}'_2 = true);$

Figure 3 shows the formal operational semantic based on our state machine calculus terms presented in Table 2 that are part of [2]. To support tokens we augment the “Over bar” operator with integer value  $n$  such that the  $\overline{\mathcal{N}}^n$  denotes the term  $\mathcal{N}$  marked with  $n$  tokens. Furthermore, we use a prefix label  $l$  for each node to uniquely reference it in the case of a backward flow connection. Let  $\mathcal{L}$  be a collection of labels ranged over by  $l; l0; l1, \dots$  and  $\mathcal{N}$  be any node (except initial) in the state machine.

$$\text{INIT } l : i \mapsto \mathcal{N} \xrightarrow{l_1} l : \bar{i} \mapsto \mathcal{N}, \text{ PRG-1 } l : \bar{i} \mapsto \mathcal{N} \xrightarrow{l_1} l : i \mapsto \overline{\mathcal{N}}$$

$$\text{FORK } l : \overline{F(\mathcal{N}_1, \mathcal{N}_2)}^n \xrightarrow{l_1} l : \overline{F(\overline{\mathcal{N}}_1, \overline{\mathcal{N}}_2)}^{n-1}, \text{ JOIN } l : \overline{\mathcal{N}} \mapsto l' : J(x, y) \xrightarrow{l_1} l : \overline{\mathcal{N}} \mapsto l' : J(\overline{x}, \overline{y})^n$$

$$\text{CHOICE } l : \overline{D(\lambda_1, g, \mathcal{N}_1, \lambda_2, \mathcal{N}_2)}^n \xrightarrow{g}_{\lambda_1} l : \overline{D(g, \overline{\mathcal{N}}_1, \mathcal{N}_2)}^{n-1}, \text{ FINAL } S[l : \odot] \xrightarrow{l} |S|$$

$$\text{JUNCTION } l' : \overline{\mathcal{N}} \mapsto M(x, g_1, \mathcal{N}_1, y, g_2, \mathcal{N}_2) \xrightarrow{l} l : \overline{\mathcal{N}} \mapsto l' : M(\overline{x}, g_1, \overline{\mathcal{N}}_1, y, g_2, \overline{\mathcal{N}}_2)^n$$

**Fig. 3** A symbolic semantic of State Machine Calculus

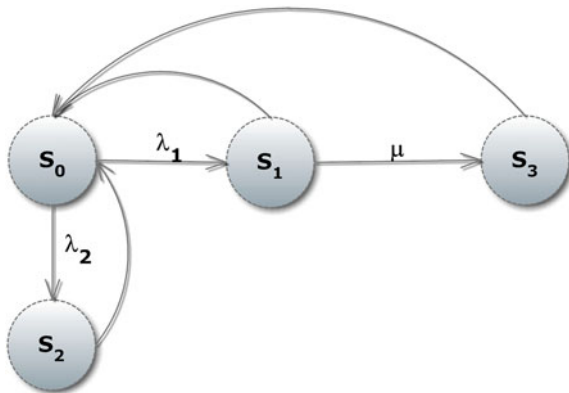
## 6 The PRISM Model Checker

In this paper, we use PRISM probabilistic model checker [14]. It supports the analysis type of probabilistic models: Discrete-time Markov chains (DTMC), Continuous-time Markov chains (CTMC), Markov decision process (MDP). Moreover, PRISM allows us to verify properties specified in PCTL [9] and CSL [28] temporal logic.

In this paper, we focus on Continuous-Time Markov chains (CTMCs) in reliability and availability analysis, such that [11, 15]. A CTMC involves a set of states  $S$  and a transition rate matrix  $R: S \times S \rightarrow \mathbb{R}_{\geq 0}$ . The delay before which a transition between states  $s$  and  $s'$  is specified by the rate  $R(s, s')$ . The probability that a transition between state  $s$  and  $s'$  might take place within time  $t$  is given by  $1 - e^{-R(s,s') \times t}$  which matches the SysML state machine diagram semantics (Sect. 4). We define each transition from  $s$  to  $s'$  in PRISM as one command as shown in Fig. 6. The global state of model is derived from the state of individual value of command variables. The guard  $s0 = \text{true}$  indicates that command is only executed when variable  $s0$  is true. The update  $(s1' = \text{true}) \ \& \ (s0' = \text{false})$  and their associated rate indicate that the value of  $s0$  will change to false and  $s1$  will change to true with rate “ $\lambda_1$ ”. In CTMC, when multiple possible transitions are available in a state, a race condition occurs [13] and this can arise in several ways. Firstly, within a module, multiple transitions can be specified either as several different updates in a command, or as multiple commands with overlapping guards.

Lets assume that we have a system for satellite error detection capability. The Markov model of such a system as shown in Fig. 4, can be build with four states ( $s_0$ : Operational,  $s_1$ : Fault detected,  $s_2$ : Interruption detected,  $s_3$ : Satellite replacement and Launch) representing the satellite status. The failure rates  $\lambda_1, \lambda_2$  are constant between states where  $\mu = 1/24$ , 24h is the time to decide to build a new satellite (i.e. Mean Time To Repair). This system can be described using PRISM modeling language as shown in Fig. 6.

**Fig. 4** A simple Markov chain to illustrate the failure occurrence





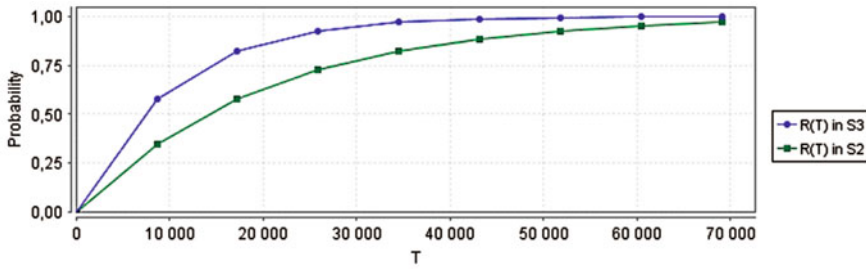


Fig. 5 Reliability versus time

Fig. 6 Sample PRISM code

```

module satellite
s0 :bool init true;
s1 :bool init false;
s2 :bool init false;
s3 :bool init false;
[] (s0=true) → lambda_1:(s1'=true) & (s0'=false);
[] (s1=true) → mu:(s3'=true)&(s1'=false);
[] (s0=true) → lambda_2: (s0'=false) &(s2'=true);
endmodule
    
```

In order to perform model-checking, a property should be specified in Continuous stochastic logic (CSL). For example, CSL allows the expression of logical states such as “what is the expected reward cumulated up to time-instant  $t$  for a computer to reboot”  $R_{\Rightarrow}[C^{\leq t}]$ , where “reboot” labels the reward construct in PRISM or “what is the probability of an error occurring within  $T$  time steps”:  $P_{<0.1}[F \leq 10(s3 = true)]$ ; the property is true in a state  $s3$  of the Markov chain if the set of paths that start from  $s0$  and reach the state  $s3$  in the first 10 times units has a probability of at least 0.1.

In our study, we use rewards (i.e. state Cost) to calculate the expected time ‘ $T$ ’ for maintenance with respect to the CSL property. PRISM can be augmented with rewards: real values associated with states and transitions of the model [14]. For example, the cost of visiting the state “ $s2 = true$ ” is equal to 1. Rewards are associated with models using the rewards construct:

**rewards** “maintainability”

$S2 = true: 1;$

**endrewards**

For the Markov chain in Fig. 4, if  $\lambda_1 = 0.0001$  and  $\lambda_2 = 0.00005$ , then the reliability function of that Markov model can be generated using PRISM as displayed in Fig. 5. The reliability is obtained by checking the model against the property: “the probability that a satellite will need to be replaced by a new one due to complete failure in 8 years” and expressed using CSL as  $P = ?[F^{\leq T} s3]$ ;  $T = 0:70656:8640$  (Figs. 6 and 7).

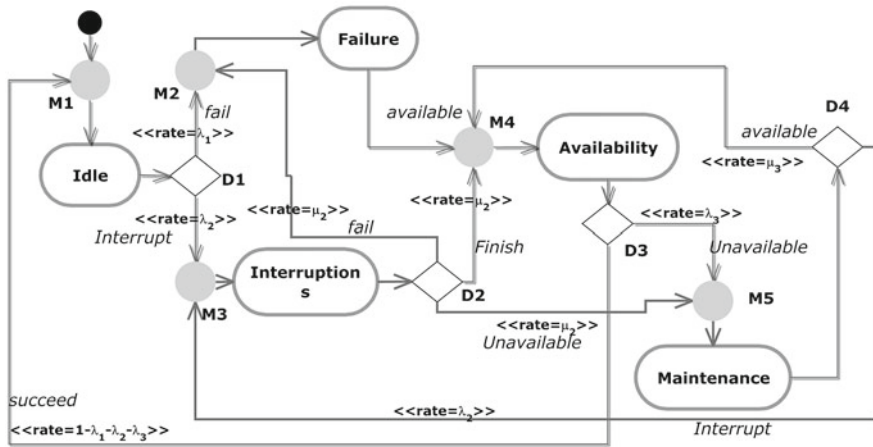


Fig. 7 A failure model and maintainability plan [10]

## 7 Case Study for Availability Assessment

### 7.1 Case Study 1: Industrial Process Control

In the following, we present a case study [10] (Fig. 7) describing an industrial control equipment consisting of printing system which is a complex line including 11 machines which are: an uncurlor, a debtor, 4 printing units, a wipe-catcher system, a dryer, a cooler, a paper passage an folding machine. From the designer’s point of view, the high degree of automation machines makes it possible to have a reduced human intervention on the machine utilization with availability equal to 100 %. The essential role of users is to supervise the working of the machine, except the beginning settings of impression and charging row material. However, the observations at user site show that system is rarely in nominal mode of operation. It is available for about 60 % of its use time. We have observed some events as failures, interruptions, preventive maintenances. Taking into account the production type (process shop), it is very important to note that if any machine of the printing system is stopped, all production will stop. So, the company applied a preventive maintenance. Three types of stop are observed: (1) Stops caused by the preventive maintenance which is the most happening and it is represented by state “maintenance”, (2) Stops caused by failure and it is represented by state “Failure”, in spite of preventive maintenance, we observed some stops caused by failure due to raw material and consumable types (ink, solvent, etc.) used in the process, (3) Stops caused by interruptions (i.e. Accidents), in particularly, at the system setting up and it is represented by state “Interruption”. To assure production continuity user intervenes some times on operating system to replace a failing component which is an undesirable behavior (Figs. 8 and 9).

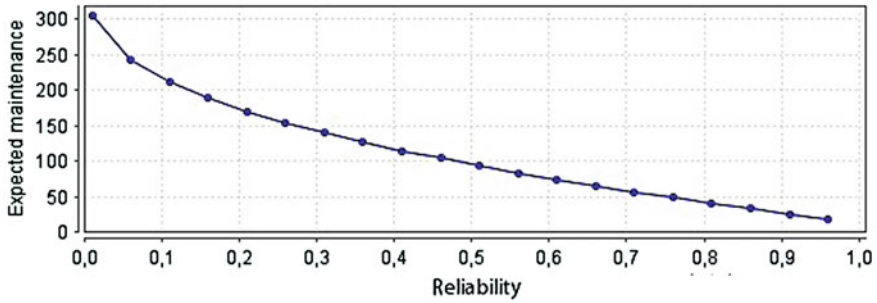


Fig. 8 Maintenance versus reliability

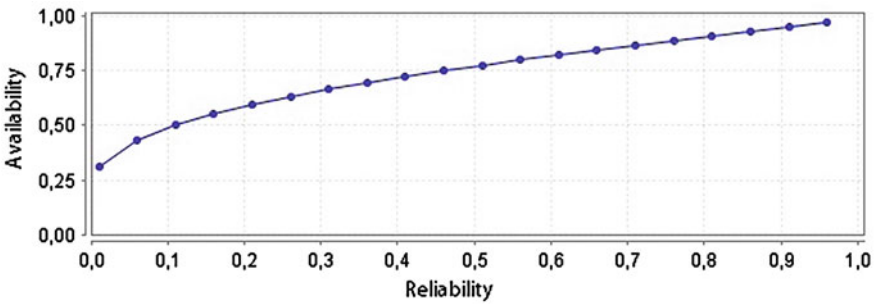


Fig. 9 Availability versus reliability

Table 3 Parameters for the CTMC model and analyses—case study 1

R	$\frac{R}{\lambda_1} = MTBF$	$\lambda_3 = \frac{1}{MTBM}$	$\lambda_2 = \frac{1}{MTBA}$	$\mu_1 = \frac{1}{MTR}$	$\mu_3 = \frac{1}{MTM}$	$\mu_2 = \frac{1}{MTP}$
90 %	3600 s	40 %	5 %	90 %	90 %	90 %

For system analysis, operations related to failures, maintenances, interruptions (i.e. Accidents), repairing and preparing were collected and presented in Table 3. we use R to express the reliability of the designed system. If the system fails, we say that the system moves from normal state (i.e. Idle) to failure state. Taking into account system life cycle aspects, the indicators required to assess availability and maintainability at design stage are described in Sect. 3.

We assume that the time delay is a random variable selected from an exponential distribution, which is an assumption used in PRISM. according to the system reliability theory, the reliability of a system from  $R(t)$  can be defined as

$$R(t) = Pr\{T > t\} = e^{-\lambda t} \tag{1}$$

and, then we can obtain

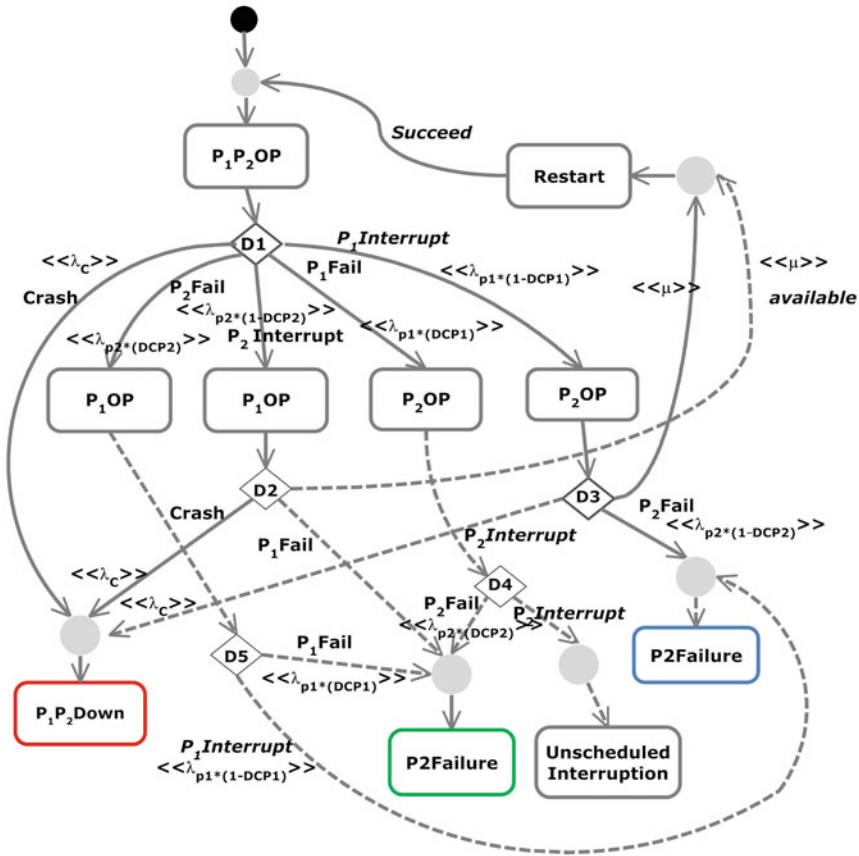


Fig. 10 A failure model of computing platform

$$\lambda(t) = \frac{-\ln R(t)}{MTBF} \tag{2}$$

Failures typically occur at some constant failure rate  $\lambda$ , failure probability depends on the rate  $\lambda$  and the exposure time  $t$ . Typically failure rate are carefully derived from substantiated historical data [10].

PRISM provides support for automated analysis of quantitative properties. In case of our system two properties are analyzed for (1) maintainability and (2) availability assessment:

1. The system maintenance times when the reliability ranges from 0.01 to 0.99 in 3600 s:  $R\{maintenance\}_{=?}[C \leq T]; T = 3600, R = 0.01 : 0.99 : 0.01$ ;
2. The availability of system in 3600 s when the reliability ranges from 0.01 to 0.99:  $R\{availability\}_{=?}[C \leq T]/T; T = 3600, R = 0.01 : 0.99 : 0.01$ ;

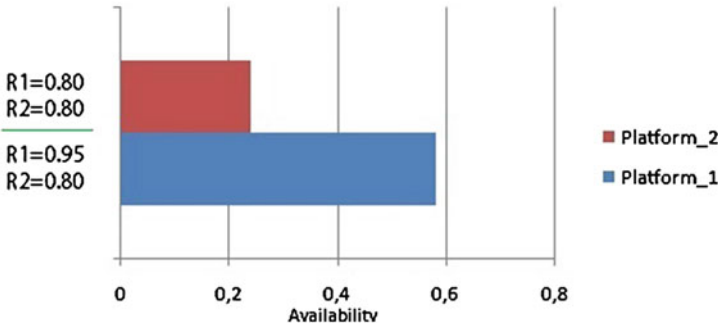


Fig. 11 Availability versus platform reliability

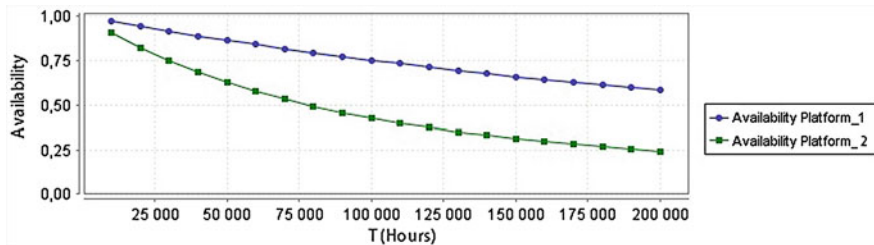


Fig. 12 Availability versus time

As shown in Fig. 11, when the reliability of of the dual core is 0.95 in *platform*<sub>1</sub> the availability is equal to 0.528 however, for the *platform*<sub>2</sub> when reliability is equal to 0.80 the availability is very low and equal to 0.247. As shown in Fig. 12, if the time increases the availability decreases from 0.97 to 0.58 for the first platform and decreases from 0.9 to 0.24 for the second platform. So, it is obvious that the availability decreases in time due to the processors performance degradation but it is clear that the first platform offers more performance than the second one due to the high reliability. For high availability, designer could assume the reliability  $\geq 95\%$ . Nevertheless, the verification of the first property is sufficient to attest that the architecture of the first platform offers more performance than the second one.

### 7.2 Case Study 2: Computing Platform

In this case study we expect a better architectural platform based on availability assessment. The platform consists on two parallel processors P1 and P2. Two kind of configuration platform are checked where the firsts one; the processor P1 is dual-core and the second architecture; the processor is single core. The case study is well detailed in [8] but in our paper it is modeled using SysML. We try to summarize the failure observations data in Table 4 to enrich our state machine diagram in Fig. 10

**Table 4** Parameters for the CTMC model and analyses—case study 2

Platform	$R_{P1}$	$R_{P2}$	$DC_{p1}$ (%)	$DC_{p2}$ (%)	$\lambda_{crash}$	$\mu_{repair}$
1	0.95	0.80	99	60	$200 \times 10^{-9}$	1/24
2	0.80	0.80	99	60	$100 \times 10^{-9}$	1/24

where  $\mu_{repair}$  is a repair rate and  $\lambda_{p1}, \lambda_{p2}$  is a failure rate of the first and second processor calculated from the reliability of both processors and  $DC_{p1}, DC_{p2}$  are diagnostic coverage of the first and second processors. We assume that the MTBF of the processors is 24 years. In addition, red state refers to the crash of two processors, green state is a state when processor 1 and 2 fails, blue state is a state when interruption occurs at P1 and P2 fails.

For best platform configurations, two properties are analyzed for availability assessment:

1. The system availability when the reliability of the dual core is 0.95 in *platform*<sub>1</sub> and 0.80 in *platform*<sub>2</sub> at 200000 h:  $R\{availability\}_{=?}[C \leq T]/T; T = 200000;$
2. The availability of system at different instants T when the reliability of the dual core is 0.95 in *platform*<sub>1</sub> and 0.80 such as  $R\{availability\}_{=?}[C \leq T]/T; T = 1 : 200000 : 10000;$

As shown in Fig. 11, when the reliability of of the dual core is 0.95 in *platform*<sub>1</sub> the availability is equal to 0.528 however, for the *platform*<sub>2</sub> when reliability is equal to 0.80 the availability is very low and equal to 0.247. As shown in Fig. 12, if the time increases the availability decreases from 0.97 to 0.58 for the first platform and decreases from 0.9 to 0.24 for the second platform. So, it is obvious that the availability decreases in time due to the processors performance degradation but it is clear that the first platform offers more performance than the second one due to the high reliability. For high availability, designer could assume the reliability  $\geq 95\%$ . Nevertheless, the verification of the first property is sufficient to attest that the architecture of the first platform offers more performance than the second one.

## 8 Conclusion

In this paper, we presented a formal approach for maintainability and availability assessment of probabilistic systems. PRISM language requires considerable expertise, and engineers do not have the necessary level of training to master its use. For this purpose, we propose a mapping mechanism of the SysML state machine into the input language of the probabilistic model checker PRISM. Moreover, we proposed a calculus dedicated to this diagram that captures precisely their underlying semantics.

For quantitative assessment, we have shown the effectiveness of our approach by applying it on a case study where availability and maintainability are evaluated using

CSL properties. The results are close to [10] and more accurate since the assessments are correlated to reliability. The presented work can be extended in the following two directions. First, we will look for an approximation of other kind of distributions in probabilistic model since our approach is based on exponentiation distribution. Second, we plan to document more interruptions and failure states for a precise interpretation.

**Acknowledgments** This research was performed as part of the LIMPAF/CNEPRU/C00L07UN10 0120110008 project supported by the Algerian Ministry of Higher Education and Scientific Research and the LIMPAF Lab at the University of Bouira, Algeria.

## References

1. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
2. Baouya, A., Bennouar, D., Ait Mohamed, O., Ouchani, S.: On the probabilistic verification of time constrained sysml state machines. In: Fujita, H., Guizzi, G. (eds.) Intelligent Software Methodologies, Tools and Techniques, Communications in Computer and Information Science, vol. 532, pp. 425–441. Springer International Publishing (2015)
3. Behrmann, G., David, A., Larsen, K.G.: A tutorial on uppaal. In: Formal Methods for the Design of Real-Time Systems, pp. 200–236 (2004)
4. Birolini, A.: Reliability engineering: theory and practice. Basic Concepts, Quality and Reliability (RAMS) Assurance of Complex Equipment and Systems, pp. 1–24. Springer, Berlin (2014)
5. Calinescu, R., Ghezzi, C., Johnson, K., Pezze, M., Rafiq, Y., Tamburrelli, G.: Formal verification with confidence intervals to establish quality of service properties of software systems. *IEEE Trans. Reliab.* **99**, 1–19 (2015)
6. Dhouibi, M., Saintis, L., Barreau, M., Perquis, J.M.: Safety driven optimization approach for automotive systems. In: Reliability and Maintainability Symposium (RAMS), 2015 Annual, pp. 1–7 (2015)
7. Franco, J., Barbosa, R., Zenha-Rela, M.: Reliability analysis of software architecture evolution. In: 2013 Sixth Latin-American Symposium on Dependable Computing (LADC), pp. 11–20 (2013)
8. Ghadhab, M., Kuntz, M., D.K., Fetzer, C.: Formal techniques for safety-critical systems. In: Fourth International Workshop, FTSCS 2015, Paris, France, November 6 and 7, 2015. Springer International Publishing (2016)
9. Hahn, E.M., Han, T., Zhang, L.: Synthesis for PCTL in parametric Markov decision processes. In: Proceedings of 3rd NASA Formal Methods Symposium (NFM'11). LNCS, vol. 6617. Springer (2011)
10. Houssin, R., Coulibaly, A.: Safety-based availability assessment at design stage. *Comput. Ind. Eng.* **70**, 107–115 (2014)
11. Hoque, K., Ait Mohamed, O., Savaria, Y., Thibeault, C.: Early analysis of soft error effects for aerospace applications using probabilistic model checking. In: Artho, C., Ölveczky, P.C. (eds.) Formal Techniques for Safety-Critical Systems, Communications in Computer and Information Science, vol. 419, pp. 54–70. Springer International Publishing (2014)
12. Huang, X., Sun, Q., Li, J., Pan, M., Zhang, T.: An mde-based approach to the verification of sysml state machine diagram. In: Proceedings of the Fourth Asia-Pacific Symposium on Internetware. *Internetware'12*, pp. 9:1–9:7. ACM, New York (2012)

13. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*. LNCS (Tutorial Volume), vol. 4486, pp. 220–270. Springer (2007)
14. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: *Computer Aided Verification—23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14–20, 2011*. Proceedings, pp. 585–591 (2011)
15. Kwiatkowska, M., Norman, G., Parker, D.: Prism: Probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.* **36**(4), 40–45 (2009)
16. Lazzaroni, M., Cristaldi, L., Peretto, L., Rinaldi, P., Catelani, M.: *Reliability engineering: basic concepts and applications in ICT. Repairable Systems and Availability*, pp. 85–92. Springer, Berlin (2011)
17. Liu, Y., Shen, G., Huang, Z., Yang, Z.: Quantitative risk analysis of safety-critical embedded systems. *Softw. Qual. J.* 1–25 (2016)
18. Lu, Y., Peng, Z., Miller, A.A., Zhao, T., Johnson, C.W.: How reliable is satellite navigation for aviation? Checking availability properties with probabilistic verification. *Reliab. Eng. Syst. Saf.* **144**, 95–116 (2015)
19. Mallet, F., de Simone, R.: MARTE: a profile for RT/E systems modeling, analysis and simulation. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops, SimuTools 2008, Marseille, France, March 3–7, 2008*, p. 43 (2008)
20. Morant, A., Gustafson, A., Söderholm, P.: Safety and availability evaluation of railway signalling systems. In: Kumar, U., Ahmadi, A., Verma, A.K., Varde, P. (eds.) *Current Trends in Reliability, Availability, Maintainability and Safety, Lecture Notes in Mechanical Engineering*, pp. 303–316. Springer International Publishing (2016)
21. Norman, G., Parker, D.: *Quantitative verification: Formal guarantees for timeliness, reliability and performance*. Technical Report. The London Mathematical Society and the Smith Institute (2014)
22. O.M. Group (ed.): *OMG Systems Modeling Language (Object Management Group SysML)* (2012)
23. Ouchani, S., Ait Mohamed, O., Debbabi, M.: A property-based abstraction framework for sysml activity diagrams. *Knowl. Based Syst.* **56**, 328–343 (2014)
24. Peng, Z., Lu, Y., Miller, A., Johnson, C., Zhao, T.: A probabilistic model checking approach to analysing reliability, availability, and maintainability of a single satellite system. In: *Modelling Symposium (EMS), 2013 European*, pp. 611–616 (2013)
25. Qiu, S., Sallak, M., Schön, W., Cherfi-Boulanger, Z.: Availability assessment of railway signalling systems with uncertainty analysis using statecharts. *Simul. Model. Pract. Theory* **47**, 1–18 (2014)
26. Reliasoft: Lambda Predict. <http://www.reliasoft.com/predict/>
27. Sivanandam, S.N., Deepa, S.N.: *Introduction to Genetic Algorithms*, 1st edn. Springer Publishing Company (2010) (Incorporated)
28. Song, L., Zhang, L., Godskesen, J.: Bisimulations and logical characterizations on continuous-time markov decision processes. In: McMillan, K., Rival, X. (eds.) *Verification, Model Checking, and Abstract Interpretation. Lecture Notes in Computer Science*, vol. 8318, pp. 98–117. Springer, Berlin (2014)
29. Tian, Y., Wan, L., hung Chen, C., Yang, Y.: Safety assessment method of performance-based navigation airspace planning. *J. Traffic Transp. Eng. (English Edition)* **2**(5), 338–345 (2015)