Journal of Theoretical and Applied Information Technology
20th March 2013. Vol. 49 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ENVIRONMENT FOR MODELING AND TESTING STOCHASTIC SYSTEMS

[1] MOKDAD AROUS, [2] KENZA BOUAROUDJ, [3] DJAMEL-EDDINE SAÏDOUNI

[1, 2, 3] MISC Laboratory, Mentouri University, Constantine, 25000, Algeria.

E-mail: [1] arous@misc-umc.org, [2] bouaroudj@misc-umc.org, [2] saidouni@misc-umc.org

ABSTRACT

This paper addresses a formal testing based on Stochastic Refusals Graphs (SRG) in order to test stochastic systems. As a semantic model for these systems, we chose the MLSTS models. MLSTS (for *Maximality-based Labeled Stochastic Transition System*) is a new semantic model for characterizing the stochastic temporal properties of concurrent systems, under the assumption of arbitrarily distributed (i.e. non-Markovian) durations of actions [14]. MLSTS can be easily described by using a Stochastic Process Algebra (SPA) language, namely S-LOTOS [15]. So, a method for testing stochastic system modeled by MLSTS is proposed [23]. We present here a tool for Modeling and testing Stochastic Systems, called MoVeS, which allows generating automatically MLSTS models from S-LOTOS specifications, then constructing automatically Refusal Graphs and Canonical Tester.

Keywords: *Basic-LOTOS, Maximality Semantics, Semantic Models, Labeled Stochastic Transition System, Refusal graph, Canonical tester.*

1. INTRODUCTION

Nowadays, Computer applications have become increasingly involved in real-time and stochastic systems (e.g. automotive, avionic and robotic controllers, mobile phones, communication protocols and multimedia systems). These systems are known by their high complexity. Formal testing allows checking their correctness and helps to ensure their quality.

In order to use a formal testing technique, we need that the systems under study can be expressed in terms of a formal language. Formal languages, which was first developed for functional needs, have become more expressive, by allowing the explicit representation of non-functional aspects of systems, for example, the probability [18, 9] or the time consumed by the system to perform tasks. This time was first considered as fixed [13, 2], and recently defined in stochastic terms [3, 19].

The use of stochastic models (e.g. stochastic extensions of automata (Network) [21], Petri Nets [6], and Process Algebras [1, 8, 11, 20]) allows producing more realistic systems. Among the specification languages, Stochastic Process Algebras (SPAs) take advantage from its compositionality (model a system as the interaction of its components) and abstraction aspects (build up complex models from detailed components but disregarding internal behavior when it is appropriate to do so), whereas providing a formal description context.

Two types of semantics are considered for SPAs and their semantic models. The first one is the interleaving semantics, where the executions of two actions are interpreted by their interleaved executions in time, and which hold only in the case of exponentially distributed durations [1, 11, 16]. The second one consists in the true concurrency semantics, which appear as the solution when considering action durations with general (non-Markovian) distributions.

In our work, systems are represented by a new semantic model, namely MLSTS (for *Maximality-based Labeled Stochastic Transition System*) [14], wherein actions elapse in time and their durations depend on probabilistic distribution functions. MLSTS models are based on maximality semantics [5] and advocates the true concurrency; from this point of view it is well suitable for modeling real time, concurrent and distributed systems. [14] shows that the maximality based semantic models describe the same qualitative and quantitative properties as specified in the ST-Semantic models. The main advantage of the MLSTS is the drastic reduction of the number of states and transitions w.r.t. standard ST-semantic models, which are frequently used in modeling and analyzing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

stochastic systems with non-Markovian Process Algebra, e.g. [1, 11, 20].

In this paper, we are interested in formal testing approaches where the temporal requirements of systems are taken into account. We present testing architecture based on Stochastic Refusals Graph (SRG). SRG results from a new definition of refusals. This graph allows us to generate a canonical tester by several transformations on MLSTS. Moreover, we investigate the automatic extraction of test cases. The proposed architecture is summarized in Fig.1.

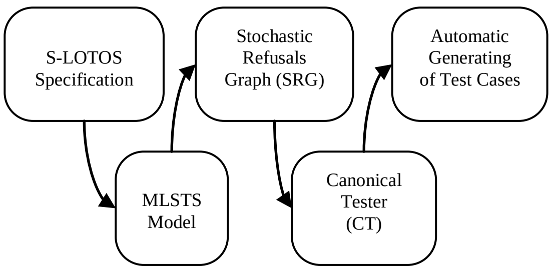


Fig. 1. Test architecture.

The structure of this paper is as follows: in section 2 we present informally the MLSTS models, and we define it formally. In section 3 we present an approach for testing stochastic system modeled by MLSTS. Section 4 presents our tool MoVeS (for ***Mo****deling and* ***Ve****rifying* ***S****tochastic* ***S****ystems*) which consists of an implementation of the theoretical notions and approach presented in this paper. Next, section 5 presents by an example the functionalities of MoVeS. Finally, section 6 gives some conclusions and perspectives.

2. MAXIMALITY-BASED LABELED STOCHASTIC TRANSITION SYSTEMS

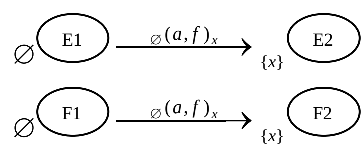
2.1. Informal Presentation of MLSTS

Within the semantic model MLSTS, each transition only represents the start of an action execution. Since actions are not considered as atomic, the concurrent execution of multiple actions can be represented, and distinguishing between sequential and parallel executions is possible.

In MLSTS models, the running actions are represented at the states level. Each instance of running actions is called a maximal event and is identified by a distinct name. In fact, each state of the system is featured by a unique configuration [4]. The configuration of a state s is denoted $_M[E]$ s.t. M is the set of maximal events in s and E is the behavior expression of s. Every transition defined from s is labeled by $_C(a, f)_x$ whenever a is an action that can be activated from E iff. the maximal events of the subset $C\subseteq M$ are terminated. Further C is called the causality set of the transition. x is the name identifying the start event of the new execution of a. The event identification is required to avoid confusion since several instances of running actions can have the same action name.

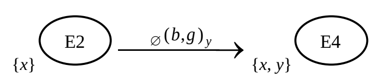
A detailed presentation of the maximality semantics can be found in [4, 5]. In this section we illustrate the principal of maximality semantics of MLSTS models by a simple example, consider tow actions a and b with probability distribution functions f and g respectively, and two systems E and F represented by Petri Nets of Fig. 2, s.t. E executes a in parallel with b, and F executes either a followed by b or b followed by a. The MLSTSs representing the behaviors of E and F, obtained by applying the maximality semantics, are represented in Fig. 3.

Initially, no action has yet been executed, then the set of maximal events is empty, and the initial configurations associated with E and F are, respectively, $_\varnothing[E]$ and $_\varnothing[F]$. By assuming that the action a happens first from E and F, the corresponding transitions are respectively:



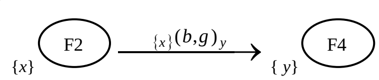
Where x is the event name identifying the starting of a. In both new resulting states, x is said maximal.

From $E2$, the following transition occurs in case b starts:



where y is the maximal event name identifying the start of b, which does not depend on the termination of a because the parallel execution.

From the new state $F2$ and because of the sequential execution of actions a and b, we deduce that the start of b is constrained by the causality dependence against x. Actually, it is submitted to the end of the execution of a. This results in the following transition:



Journal of Theoretical and Applied Information Technology
20th March 2013. Vol. 49 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

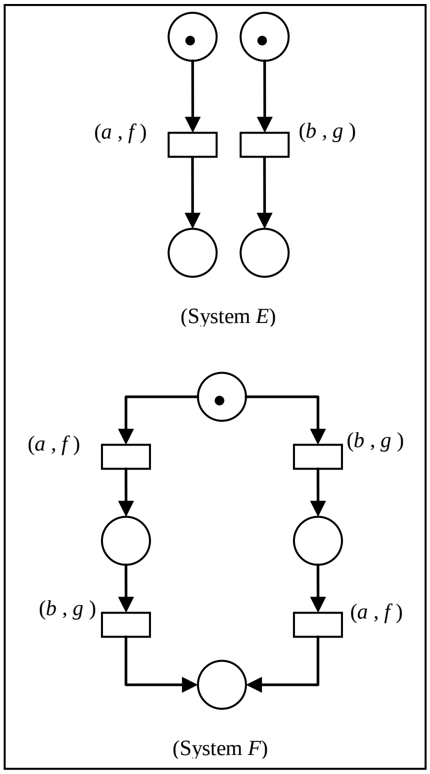


Fig. 2. Example of two systems

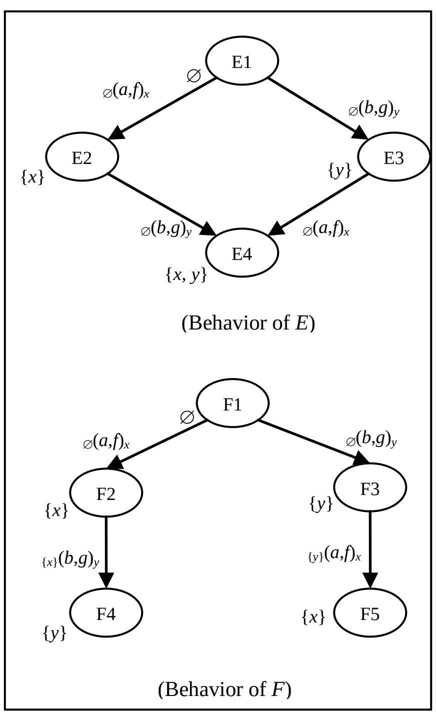


Fig. 3. Behaviors of *E* and *F* according to the maximality semantics

In the resulting state, the only maximal event is the one identified by *y*, representing the start of execution of *b*. which is different from the state E4 whereas two maximal events appear (identified by *x* and *y*). Observe that a symmetric scenario happens when the action *b* happens first as this can be seen in Fig. 3.

2.2. Formal Definition Of MLSTS

An MLSTS is defined as follows:

Definition 1. Maximality-based Labeled Stochastic Transition System (MLSTS).

Let $\mathcal{M}$ be a countable set of event names. An MLSTS is a structure $(\Omega, A, DF, L, \mu, \xi, \psi)$ with:

- $\Omega = (S, s_0, T, \alpha, \beta)$: is a Transition System s.t. *S* is the countable set of states for the system, at least including the initial state s_0 ; *T* is the countable set of transitions specifying the states changes ; α, β are two functions: $T \rightarrow S$. mapping every transition with its source $\alpha(t)$ and its target $\beta(t)$.

- *A* : is a (finite) set of actions

- *DF* : is a finite set of probability distribution functions ($\Re \rightarrow [0, 1]$).

- L: $T \rightarrow (A \times DF)$: this function associates each transition with a pair composed of an action and a probability distribution function, thus $(\Omega, (A \times DF))$ is a transition system labeled by alphabet $(A \times DF)$.

- $\psi : S \rightarrow 2_{fn}^{\mathcal{M}}$: this function associates every state with a finite set of maximal event names in the state.

- $\mu : T \rightarrow 2_{fn}^{\mathcal{M}}$: this function associates every transition with a finite set of maximal event names of actions that have started their execution so that their terminations allow the start of this transition. This set corresponds to the direct causes of the transition.

- $\xi : T \rightarrow \mathcal{M}$: this function associates each transition with an event name identifying an event occurrence such that for any transition $t \in T$:

$\mu(t) \subseteq \psi(\alpha(t))$

$\xi(t) \notin \psi(\alpha(t)) - \mu(t)$

$\psi(\beta(t)) = (\psi(\alpha(t)) - \mu(t)) \cup \{\xi(t)\}$ ♦

MLSTS is able to deal with any kind of probability distribution instead of restricting only to exponential distributions, and without being attacked by the state space explosion problem inherent to the splitting of actions as in standard

Journal of Theoretical and Applied Information Technology
20th March 2013. Vol. 49 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ST-semantic models, which are frequently used in performance specification and modeling under the assumption of general distributed durations. The main advantage of the MLSTS consists in reducing the number of states and transitions w.r.t. standard ST-semantic models [14].

3. AUTOMATIC GENERATION OF MLSTS

For a given system, its MLSTS model can be easily generated from a high level specification language, namely S-LOTOS [15]. S-LOTOS is a Stochastic Process Algebra which deals with general probability distributions instead of restricting to exponential ones. The formal semantics given to different operators of the language allows generating automatically MLSTSs from algebraic specifications with S-LOTOS according to the maximality semantics. The reader is assumed to be familiar with the syntax of Basic LOTOS, a standard process algebra, from which S-LOTOS derives. See [15] for details about the semantics of the different operators.

In the MLSTS models, the set of states is given by the set of configurations. The transition relation between states is defined by the operational rules of the maximality semantics given in [15]. The transition label is given by the tuples of its causality set, the action name, and the event name identifying the execution start of this action.

In [17], we define a syntax for complete specification S-LOTOS of systems, as show the example in Fig. 4, a system specification S-LOTOS contains:

✓ A header containing the name and parameters (actions with their distribution function) of the system. For example :

```
system test [(incard,f) , (valide,g),
(codenotok,g), (rejectcard,f),
(keepcard,g), (codeok,g), (outcard,g),
(takemoney,f)]
```

✓ A behavioral specification according the syntax of Basic LOTOS [7]. For example :

```
incard; valide;
((codenotok; rejectcard; stop) []
(codenotok; keepcard; stop) [] (codeok;
outcard; takemoney; stop))
```

✓ Definitions of eventual processes composing the system. For example :

```
process
wrongcode[codenotok,rejectcard,keepcard] :=
(codenotok; rejectcard; stop) []
(codenotok; keepcard; stop)
endproc
```

✓ Definition, for each distribution function of actions cited in the system header, of the type of probability distribution function with effective parameters. For example: `"pdf g := exppdf(1.4) endpdf"` is the definition of an exponential probability distribution function *g*.

4. AN APPROACH FOR TESTING STOCHASTIC SYSTEM MODELED BY MLSTS

In [23] we proposed a new testing architecture based on Stochastic Refusals Graph (SRG). SRG results from a new definition of refusals. This graph allows us to generate a canonical tester by several transformations on it. Moreover, we investigated the automatic extraction of test cases. A detailed presentation of this approach is in [23], We can summarize the architecture used as follow:

- Computation of refusals sets from a deterministic MLSTS; therefore the SRG graph is constructed.
- Generation of canonical tester.
- Automatic extraction of test cases.

In the first point, Stochastic Refusals Graphs (SRGs) are generated from MLSTS model. We calculate sets of refusals. Those sets decorate each location of SRG. An important aspect considered at this level is the non-determinism which is captured by permanent refusals, thus refusals are calculated in the same time of determinization. Temporary refusals are induced by the fact that actions elapse in time. Therefore, the proposed model integrates both permanent and temporary refusals.

In the second point, we generate a canonical tester over the SRG. In the canonical tester, refusals are found associated with transitions that are leading to *"Fail"* location. So, these transitions are labeled by actions which are prohibited by the specification. When the tester is in the location "*Fail*", this means that the test failed. Finally, a test case is a possible path in the tester.

5. MoVeS Tool

In this section, we present the implementation of the theoretical notions and approach outlined in previous section. We have developed a tool named MoVeS (for ***Mo**deling and **Ve**rifying **S**tochastic **S**ystems*) that allows us to editing S-LOTOS system specifications, compiling them, generating the underlying semantic models in terms of MLSTS models, and its corresponding SRG, Canonical

Journal of Theoretical and Applied Information Technology
20th March 2013. Vol. 49 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Tester and Test Cases. The graphic visualization of these different graphs is in fact supported by using the rich graphic module GraphEdit of TORSCHE Scheduling toolbox [24].

To present the functionalities of MoVeS, let as consider the example of ATM system (*Automatic Teller Machine*). This machine allows withdrawing money from account. Its behavior is as follow. Customer has to insert card in machine, after he has to type a code, if the code is correct the machine delivers money and card, if the code is wrong, the machine can keep the card or reject it. The S-LOTOS specification of this machine is given in Fig. 4. Fig.5 presents a screen of the MoVeS tool and the generated MLSTS model corresponding to the ATM system.

```
system test [(incard,f), (valide,g),
(codenotok,g), (rejectcard,f),
(keepcard,g), (codeok,g), (outcard,g),
(takemoney,f)]:=
incard ; valide ;
(wrongcode[codenotok,rejectcard,keepcard]
[] codeok;outcard;takemoney;stop )
where
process
wrongcode[codenotok,rejectcard,keepcard] :=
(codenotok; rejectcard; stop) []
(codenotok; keepcard; stop)
endproc
pdf g := exppdf(1.4) endpdf
pdf f := wblpdf(4,4) endpdf
endsys
```

Fig. 4. S-LOTOS specification of the ATM machine

Moreover, MoVeS allow us to :

- Construct the SRG over the MLSTS model as defined in [23].

- Create the canonical tester based on the SRG.

- Derive test cases from the canonical tester.

Considering our example of Fig 4, Fig 6, 7 and 8 show respectively the Stochastic Refusals Graph, its corresponding canonical tester and a test cases for the ATM system.

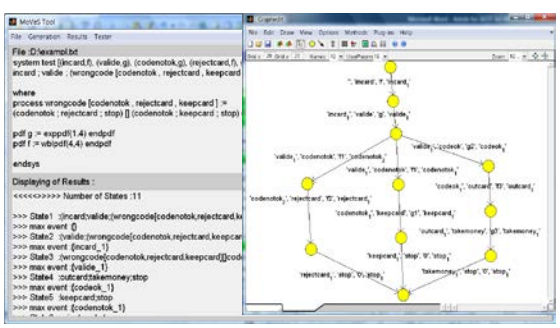


Fig 5 . Generation of MLSTS model corresponding to the ATM machine from S-LOTOS specification

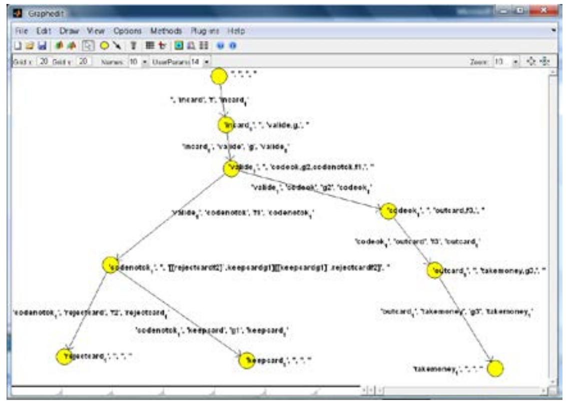


Fig 6. Stochastic Refusal Graph

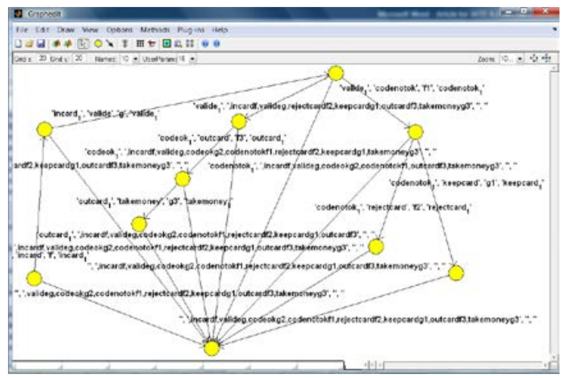


Fig. 7. Canonical tester.

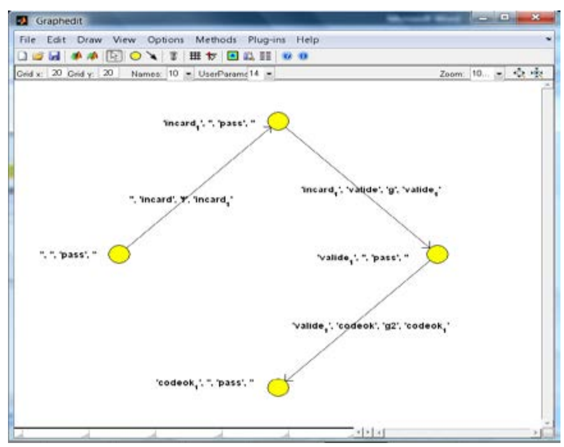


Fig. 8. Test cases.

6. CONCLUSION AND PERSPECTIVE

In this paper, we present our tool MoVeS (for ***Mo***deling and ***Ve***rifying ***S***tochastic ***S***ystems), which consist of an implementation of our approach for modeling and testing stochastic concurrent systems. These systems can be easily specified, under the assumption of generally distributed durations of actions, using a stochastic process algebra called S-LOTOS. The MoVeS tool allows the automatic generation of MLSTS models from S-LOTOS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

specification of systems. Moreover, following our testing architecture based on Stochastic Refusals Graph (SRG), the MoVeS tool allows the construction of canonical tester and the derivation of test cases for the considered systems.

As perspective, we plan to complete this work by strategy for choosing which of test cases are sufficient for insuring some completeness guarantees. Moreover, we can improve our tool to investigate more general problems of verification like performance evaluation.

REFRENCES:

[1] Clark, S. Gilmore, J. Hillston, and M. Tribastone, "Stochastic Process Algebras", In. M. Bernardo and J. Hillston (Eds.): *Formal Methods for Performance Evaluation*, LNCS 4486, Springer-Verlag, 2007, pp. 132–179.

[2] Bravetti, M., Gorrieri R.: The theory of interactive generalized semi-Markov processes. Theoretical Computer Science, 282(1):5-32, (2002).

[3] Cazorla, D., Cuartero, F., Valero, V., Pelayo, F.L., Pardo, and J.J.: Algebraic theory of probabilistic and non-deterministic processes. Journal of Logic and Algebraic Programming, 55(1-2):57-103, (2003).

[4] D. E. Saïdouni and J. P. Courtiat, "Prise en compte des durées d'action dans les algèbres de processus par l'utilisation de la sémantique de maximalité", *Proceedings of CFIP'2003. Hermes, France*, 2003.

[5] D. E. Saidouni, "Sémantique de maximalité : Application au raffinement d'actions en LOTOS", PhD thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France, 1996.

[6] F. Bause, P. S. Kritzinger, "Stochastic Petri Nets - An Introduction to the Theory -", 2nd Edition, Vieweg Verlag, Germany, 2002, ISBN: 3-528-15535-3

[7] Bolognesi T. and Brinksma E., "Introduction to the ISO specification language LOTOS", Computer Networks and ISDN Systems, vol. 14, pp. 25-59, 1987.

[8] H. Hermannsa, U. Herzogb, and J.-P. Katoen, "Process algebra for performance evaluation", *Theoretical Computer Science* 274, Elsevier Science B.V, 2002, pp. 43–87.

[9] Hennessy, E., Regan, T.: process algebra for timed systems. Information and Computation, 117(2):221-239, (1995).

[10] Hessel, A. , Larsen, K . G. , Mikucionis, M. , Nielsen, B. , Pettersson, P., Skou, A. : Testing Real-Time Systems Using UPPAAL. Formal Methods and Testing, pp 77-117 (2008)

[11] J. Hillston, "Process Algebras for Quantitative Analysis", *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, 2005, pp. 239-248.

[12] J. Tretmans: Testing Concurrent Systems: A Formal Approach. Jos C.M. Baeten, Sjouke Mauw (Eds.): CONCUR'99, LNCS 1664, pp. 46_65, 1999. Springer-Verlag.

[13] Lopez, N., Núñez M.: A testing theory for generally distributed stochastic processes. In 12th Int. Conf. on Concurrency Theory, CONCUR'01, LNCS 2154, pages 321-335. Springer, (2001).

[14] M. Arous, D. E. Saidouni and J. M. Ilié, "Addressing State Space Explosion Problem in Performance Evaluation Using Maximality-based Labeled Stochastic Transition Systems", 2nd International Conference on Computer and Software Modeling (ICCSM 2012) October 20-21, 2012, Cochin, India, to appear in IPCSIT vol. 54 (2012) © (2012) IACSIT Press, Singapore.

[15] M. Arous, D. E. Saidouni and J. M. Ilié, "Maximality Semantics based Stochastic Process Algebra for Performance Evaluation". 1*st IEEE International Conference on Communications, Computing and Control Applications (CCCA'11)* March 3-5, 2011 at Hammamet, Tunisia. IEEE Catalog Number: CFP1154M-ART, ISBN: 978-1-4244-9796-6.

[16] M. Bernardo, "Theory and Application of Extended Markovian Process Algebra", Ph.D. Thesis, University of Bologna (Italy), 1999.

[17] Mokdad Arous, "Using the maximality semantics for specifying temporal properties of stochastic concurrent systems", *Technical report*, MISC laboratory, Mentouri University, Constantine, Algeria, September 2012.

[18] Nicollin, X., Sifakis J.: An overview and synthesis on timed process algebras. In 3rd Int. Conf. on Computer Aided Verification, CAV'91, LNCS 575, pages 376-398. Springer, (1991).

[19] Nuñez, M.: Algebraic theory of probabilistic processes. Journal of Logic and Algebraic Programming, 56(1-2):117-177, (2003).

[20] P. R. D'Argenio and J.-P. Katoen, "A theory of stochastic systems. Part II: Process algebra", In. *Information and Computation* 203, Elsevier Inc, 2005, pp. 39–74.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[21] P. R. D'Argenio and J.-P. Katoen, "A theory of stochastic systems. Part I: Stochastic automata", In. *Information and Computation* 203, Elsevier Inc, 2005, pp. 1–38.

[22] K. Drira, P. Azema, and P. de Saqui Sannes. "Testability analysis in communicating systems". Elsevier science B. V. computer networks 36, pages 671—693, 2001.

[23] Kenza Bouaroudj, Ilham Kitouni, Hiba Hachichi and Djamel-Eddine Saidouni : "Extending Refusal Testing by Stochastic Refusals for Testing Non-deterministic Systems". In. IJCSI Volume 9, Issue 5, September 2012.(in press).

[24] P. Šůcha, M. Kutil, M. Sojka, and Z. Hanzálek, "TORSCHE Scheduling Toolbox for Matlab". In. *IEEE International Symposium on Computer-Aided Control Systems Design* (CACSD'06), Munich, Germany. 2006.