Dependability Analysis Using SAM

Tianjun Shi, Xudong He
School of Computer Science
Florida International University, Miami, FL 33199
{tshi01, hex}@cs.fiu.edu

Abstract

Software architecture has been of increasing importance for designers to analyze and verify system properties. Non-functional properties reflect the quality of a software system and are essential for a successful software system, but analysis of non-functional properties is less well studied compared to functional verification in the filed of software architecture. Performance and dependability are most concerned non-functional properties in safety and mission critical systems. SAM is a framework for specifying and analyzing software architecture and has been used to verify a variety of functional properties. In this paper, we extend SAM with stochastic constructs so that it can be used to model and analyze performance and dependability.

1. Introduction

As software systems keep growing in size and complexity, software architecture as a high-level abstraction is of increasing importance for designers to reason about the software systems and make early design decisions. As a result, many architecture description languages (ADLs) have been proposed [10]. Most ADLs, however, focus mainly on the formal notation, and many of them do not offer analytical techniques to verify the properties of their designs. Some ADLs do provide analytical techniques, but tend to focus on a particular property and leave other properties unexplored. For example, Wright [2] allows deadlock detecting; MetaH [18] and UniCorn [13] support schedulability analysis [10]. To contribute to the analysis of software architectures, we extend *Software Architecture Model* (SAM) with stochastic constructs for non-functional property evaluation in addition to its existing capability of functional property verification.

SAM is a formal framework for specifying and analyzing software architectures [19]. It supports hierarchical decomposition and automatic analysis of software architectures. Different techniques and tools have been used to analyze a SAM model for the evaluation of various properties, and the SAM framework has proved to be useful and desirable in specifying and verifying software architectures. For example, in [19], reachability analysis technique was used to analyze the timeliness of a real time system. The symbolic model checker SMV was used in [14] to verify the functional correctness of a communication protocol, and the theorem prover STeP was used to reason about the correctness of an electronic commercial system [20].

To date, the SAM framework mainly focuses on functional property analysis. However, non-functional properties, such as performance and dependability, are of the same importance to the quality of a software system. Dependability, known as the collection of reliability, availability, safety and related measures, is especially concerned in safety and mission critical system. Different modeling and analyzing techniques have been exploited to evaluate various non-functional properties. For example, fault-trees were used for system reliability modeling [11]; queuing networks have been used for performance analysis of computer and communication systems [1, 12]; a variety of *Stochastic Petri Nets* (SPNs), as a convenient high level formalism of Markov chains, have become popular for the analysis of performance, dependability and performability [5, 9]. In most research work, functional properties and non-functional properties are separately modeled and evaluated because of the complexity of integration. The separation eases the work of modeling and analysis, but is prone to inconsistency of models and imprecision of evaluations. With incorporated stochastic constructs, the SAM model can be transformed to a Stochastic Reward Net [5] (SRN) model. While functional property verification is conducted using the techniques mentioned above, performance and dependability can be evaluation by solving the derived SRN model.

The rest of the paper is organized as follows. Section 2 introduces SAM and SRNs. In section 3, a multiprocessor system is given as an running example throughout the paper. Section 4 depicts the stochastic extension on SAM, and section 5 explains how to transform the SAM model into an SRN. The results of reliability analysis for the example system are described in section 6, and conclusions are drawn in section 7.

2. Overview of Formalisms

To give readers a rough idea about the formalisms used in this paper, we briefly introduce SAM and SRN.

2.1. Software Architecture Model

SAM is a formal framework for specifying and analyzing software architectures. A SAM model consists of a set of *compositions* and a *hierarchical mapping* relating the compositions. A composition in turn consists of multiple *components* and *connectors*, and a set of composition *constraints*. Each component (or connector) consists of a behavior model and a set of component (or connector) constraints. A SAM model is said to be correct if the behavior models satisfy each constraint. For a formal definition and description of SAM, refer to [8].

There are two complement formal methods underlying a SAM model. Petri nets are used to define the behavior models of components and connector, while temporal logic is used to specify the constraints. To be flexible, the underlying formal foundation of SAM is not limited to a fixed pair of Petri net and temporal logic. The selection of a particular Petri net and temporal logic is based on the application under consideration. For example, real-time Petri nets and real-time computational tree logic were used to study software architectures of real-time systems [19], *Predicate Transition nets* (PrT nets) and first order linear time temporal logic were used to specify and verify a communication protocol [14].

In this paper, we use PrT nets as the behavior modeling formalism of the SAM framework. PrT nets are a class of high level Petri nets. Using the conventions and definitions in [8], a PrT net is defined as a tuple $(N, Spec, ins)$, where $N = (P, T, F)$ is the net structure; *Spec* defines the used sorts, operations and relations; and $ins = (\varphi, L, R, M_0)$ defines the mapping of places to sorts, the labels, the constraint of each transition, and the initial marking respectively. For a complete definition of a PrT net, refer to [8].

2.2. Stochastic Reward Net

SRN [5] is an extension to SPN. An SRN can be mapped into a Markov Reward model [5]. By definition, an SRN is an 11-tuple consisting of:

1. A finite set of places.
2. A finite set of transitions.
3. A finite set of input arcs from place to transition.
4. A finite set of output arcs from transition to place.
5. A finite set of inhibitor arcs from place to transition.
6. Enabling function for each transition. A transition is disabled if the enabling function is evaluated false.
7. Priorities for each transitions. A transition is disabled if there exists an enabled transition with a higher priority.
8. The initial marking.
9. A positive exponential distributed rate for each timed transition, and an immediate transition fires at no time.
10. A weight for each transition.
11. Reward measures.

SPNP [15] and SMART [4] are two tools for SRNs. By solving the underlying Markov chains of an SRN, performance and dependability of the model can be evaluated.

3. A Multiprocessor Example

We use a multiprocessor system depicted in [3] as a running example to demonstrate the use of extended SAM. As illustrated in Figure 1, a multiprocessor system consists of n subsystems $S_1, \ldots, S_n$. Each subsystem S_i is composed of one processor P_i, one local memory M_i, and m replicated mirrored disk units D_{ij}. A bus B connects the n subsystems and a shared memory M_g, which is shared by all subsystems. The complete system fails when the bus B fails or when k $(1 \leq k \leq n)$ subsystems fail. A subsystem fails when its processor fails or all its disks fail or both the local memory and the shared memory fail.

The reliability (i.e. probability of system down) is the main issue of the multiprocessor system. Figure 2 depicts the behavior model in SAM for the failures of the multiprocessor system. Each net element in the behavior model is described in Table 1. Initially, every element of system is working, The net inscription of the behavior model is as follows. Symbol $\wp$ denotes power set; n is the number of subsystems; m is the number of disks in each subsystem;

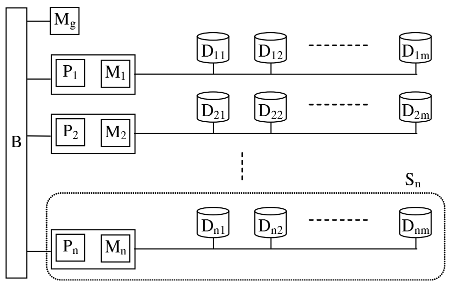

Figure 1. The multiprocessor system.

Table 1. Description of net elements.

Elements	**Description**
P_P/P_{Pf}	Processor P_i is working/not working.
P_D/P_{Df}	Disk D_{ij} is working/not working.
P_{GM}/P_{GMf}	memory M_g is working/not working.
P_{LM}/P_{LMf}	memory M_i is working/not working.
P_B/P_{Bf}	Bus B is working/not working.
P_S/P_{Sf}	The system is working/not working.
P_{sub}	Subsystem S_i is not working.
T_P	Processor P_i fails.
T_D	Disk D_{ij} fails.
T_{GM}	Shared memory M_g fails.
T_{LM}	Local memory M_i fails.
T_B	Bus B fails.
T_1	S_i fails due to processor failure.
T_2	S_i fails due to memory failures.
T_3	S_i fails due to disk failures.
T_4	system fails due to subsys. failures.
T_5	system fails due to bus failure.

and k is the minimum number of subsystems whose failures will result in the failure of the complete system.

$$\varphi(P_P) = \varphi(P_{Pf}) = \wp(\{i|1 \leq i \leq n\})$$
$$\varphi(P_{LM}) = \varphi(P_{LMf}) = \varphi(P_{sub}) = \varphi(P_P)$$
$$\varphi(P_B) = \varphi(P_{Bf}) = \wp(\{\text{“}notempty\text{”}\})$$
$$\varphi(P_{GM}) = \varphi(P_S) = \varphi(P_{GMf}) = \varphi(P_{Sf}) = \varphi(P_B)$$
$$\varphi(P_D) = \varphi(P_{Df}) = \wp(\{\langle i,j\rangle|1 \leq i \leq n, 1 \leq j \leq m\})$$
$$R(T_P) = R(T_{LM}) = R(T_B) = R(T_{GM}) = true$$
$$R(T_D) = R(T_1) = R(T_2) = R(T_5) = true$$
$$R(T_3) = (\exists Y \subseteq X.\ Y = \{\langle i,j\rangle|1 \leq j \leq M\} \wedge X' = X - Y)$$
$$R(T_4) = (\exists Y \subseteq X.\ |Y| = K \wedge X' = X - Y \wedge r = \text{“}notempty\text{”})$$
$$M(P_P) = M(P_{LM}) = \{i \mid 1 \leq i \leq n\}$$
$$M(P_D) = \{\langle i,j\rangle|\ 1 \leq i \leq n\ \wedge\ 1 \leq j \leq m\}$$
$$M(P_B) = M(P_{GM}) = M(P_S) = \{\text{“}notempty\text{”}\}$$
$$M(P_{Pf}) = M(P_{Df}) = M(P_{Bf}) = M(P_Sf) = \{\}$$
$$M(P_{LMf}) = M(P_{GMf}) = M(P_{sub}) = \{\}$$

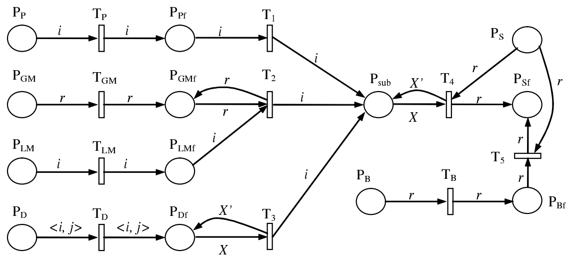

Figure 2. The SAM model for the example.

In this behavior model, the failure rate of each element of the multiprocessor system is not addressed yet. Therefore, the reliability of the system cannot be analyzed based on this model. The next section shows how SAM is extended to express stochastic information.

4. Stochastic Extension on SAM

In SAM, each behavior model is represented by a PrT net, and each requirement constraint is specified in temporal logic. The SAM framework is extended in two ways for non-functional property analysis. First, a stochastic construct is add to the behavior model (PrT net model) to express stochastic information. Second, a formalism is introduced to specify the non-functional requirements.

A special variable $RATE$ is added into the constraint of a transition (i.e. the mapping R in the net inscription) in a PrT net to denote the firing rate of the transition. A predicate $RATE = \lambda$ in the constraint of transition T denotes that T fires at rate λ, and $RATE = 0$ denotes that T is an immediate transition and takes zero time to fire. For example, if the bus B in the multiprocessor system fails at rate λ_B, the constraint of transition T_{LM} in Figure 2 can be expressed as $R(T_B) = (RATE = \lambda_B)$. The firing rate of a transition is not necessary constant, it can be expressed as marking dependent by making more than one occurrence of variable $RATE$. For example, if local memory M_1 in subsystem S_1 is more dependable than all other local memories, and M_1 fails at rate λ_1 while others fail at rate λ_2 ($\lambda_1 < \lambda_2$), then the constraint of T_{LM} in Figure 2 can be specified as $R(T_{LM}) = ((RATE = \lambda_1 \wedge i = 1) \vee (RATE = \lambda_2 \wedge 2 \leq i \leq n))$. The extra stochastic construct in the constraint of a transition does not affect the enabling and firing rules of this transition. That is, the PrT net model with the special variable $RATE$ has the same semantics with a original PrT net regarding the enabling and firing of a transition.

After extending the behavior model in SAM with the stochastic construct, a formalism is desirable to precisely specify non-functional requirements. Although some non-functional properties (e.g. maintainability) cannot be quantified and can only be specified informally, performance, dependability and performability can usually be precisely specified and verified at runtime. Temporal logic is sufficient to specify functional properties. However, it is incapable of specifying dependability and performability since those properties usually involve probability and precise time. To specify dependability and performability in SAM, we introduce a logic called *Probabilistic real time Computation Tree Logic* (PCTL), proposed by Hansson and Jonsson [7]. PCTL extends *Computation Tree Logic*, a branching-time temporal logic, and is capable of expressing time and probability in systems. For example, the property "after a request for service there is at least a 98% probability that the service will be successfully finished within 5 seconds" can be expressed in PCTL as $AG[(p \rightarrow F^{\leq 5}_{\geq 0.98} q)]$, where proposition p represents that a request is issued and proposition q represents that the service is successfully finished. Since performance, dependability and performability can usually be expressed in terms of time and probability, it is sufficient to express most indices of dependability and performability to be verified using PCTL. The advantage of specifying properties formally in PCTL is that it allows flexible analytical techniques. For example, we can also apply model checking over the underlying Markov chain of an SRN to verify properties specified in PCTL [7]. However, there do exist some indices of performance and dependability that cannot be expressed in PCTL. For example, PCTL cannot express the average time it takes to finish a service. For those properties that cannot be expressed by PCTL, we specify them informally in natural language.

By assuming that each element of the same type in the multiprocessor system fails at the same rate, the constraint mapping of the behavior model in Figure 2 is modified as follow to reflect the stochastic information.

$$
\begin{aligned}
&R(T_P) = (RATE = \lambda_P)\\
&R(T_{LM}) = (RATE = \lambda_{LM})\\
&R(T_{GM}) = (RATE = \lambda_{GM})\\
&R(T_D) = (RATE = \lambda_D)\\
&R(T_B) = (RATE = \lambda_B)\\
&R(T_1) = R(T_2) = R(T_5) = (RATE = 0)\\
&R(T_3) = (RATE = 0 \wedge (\exists Y \subseteq X.\\
&\qquad Y = \{\langle i, j\rangle | 1 \leq j \leq M\} \wedge X' = X - Y))\\
&R(T_4) = (RATE = 0 \wedge (\exists Y \subseteq X.\\
&\qquad |Y| = K \wedge X' = X - Y \wedge\ r = \text{“}notempty\text{”}))
\end{aligned}
$$

For the reliability requirement of the multiprocessor system, we assume that there is at least 95% probability that the system is working continuously for 4000 hours. This reliability requirement can be specified in PCTL as $AG[F^{\leq 4000}_{\geq 0.95} M(P_{Sf}) = \phi]$, where $M(P_{Sf}) = \phi$ denotes that place P_{Sf} in Figure 2 holds no token, which means the system is working.

5. Transformation from SAM to SRN

In order to analyze non-functional properties, the behavior model in SAM is transformed to an SRN. The procedure of the transformation is as follows.

1. The PrT net model in SAM is first unfolded to a low level Petri net. A PrT net can be unfolded using the method proposed in [6]. First, each transition is unfolded into a set of transitions according to the set of constant substitutions that satisfy constraint of the transition. Second, places are connected to the transitions according to the substitutions. Finally, remove the dead transitions and combine equivalent elements if any. It is not necessary to unfold the whole PrT net if we are only interested in part of the PrT net model.
2. The SRN model is derived from the unfolded Petri net based on the stochastic information. After the PrT net model is unfolded, each transition is designated either as an immediate transition or as a timed transition with a proper firing rate based on the stochastic information. By making use of the features of SRNs such as enabling function, marking dependent arc cardinality and marking dependent firing rate, the SRN model can be more concise and readable.
3. After the SRN model is derived, existing solution techniques and tools can be applied to obtain the non-functional property measures, and then the numerical results are checked against the property specifications in PCTL to see whether the model satisfies the specifications. The SRN model may need to be revised for the analysis of some non-functional properties. For example, to analyze the mean time to failure (MTTF) of a system, all the failure states should be made absorbing (outgoing arcs from those states are removed) [17].

Following the procedure above, the behavior model in Figure 2 is transformed to the SRN model in Figure 3, by assuming that $n = 3$, $m = 2$, and $k = 2$. That is we assume that there are three subsystems in the multiprocessor system, each subsystem has two disks and the whole system fails if at least two subsystems fail. In the SRN model, a bar represents an immediate transition and a box represents a timed transition. Figure 3(a) is the SRN model for a subsystem S_i, and

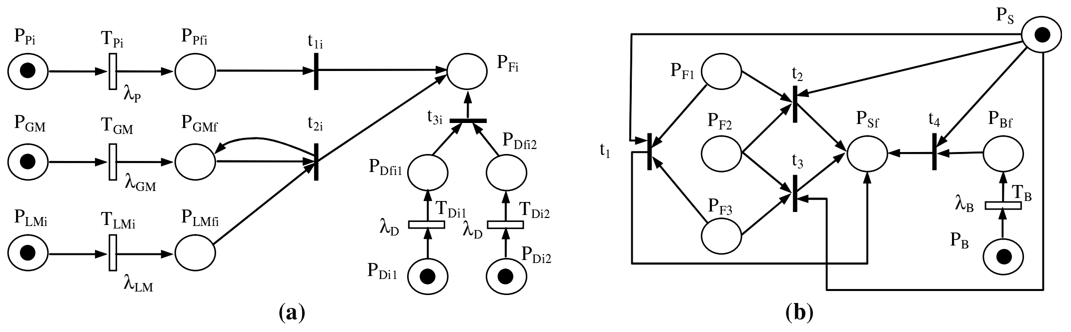

Figure 3. (a) The SRN model for subsystem P_i. (b) The SRN model for the example system.

Figure 3(b) illustrates the SRN model for the multiprocessor system with the detailed structure of each subsystem being hidden. By calculating the number of expected tokens in place P_{Sf} at time t, the probability that the multiprocess system fails at time t is obtained.

In recent research work, SRNs have been applied to evaluate a variety of non-functional properties like performance, dependability and performability. After extending SAM as mentioned above, the SAM framework is capable of modeling and analyzing all the non-functional properties that can be evaluated in SRNs. However, this methodology also inherits the limitations of SRN models. There are three main difficulties in SRN model—*largeness*, *stiffness*, and the assumption of exponentially distributed firing rates of timed transitions [16]. Largeness means the large size of the underlying Markov model of an SRN and stiffness represents the large disparity between the firing rates of an SRN. Largeness and stiffness cause difficulties in solving the underlying Markov model of an SRN. Fortunately, a number of approaches have been proposed to avoid and/or tolerate largeness and stiffness [16]. The "exponential assumption" limits the modeling power and precision of SRNs. While exponentially distributed rates are reasonable in some situations such as failure rates of a system, they are invalid in other situations like deadlines of a system. This problem, however, can be reduced either by phase approximations [16] (approximating a non-exponential distribution by a set of states and transitions such that the holding time in each state is exponentially distributed) or by allowing non-exponential distributions in SRNs.

6. Analysis Results

For the reliability analysis of the multiprocessor system, we assign the failure rates of each elements as listed in Table 2[1]. Based on the SRN model in Figure 3, we evaluate the probability of the system failure at a function of the time. The probability of system failure is obtained by using the tool SMART [4] to calculate the probability that place P_{Sf} in Figure 3 is not empty at time t. Figure 4 shows the distribution of the probability of system down. The numerical results show that there is 96.9% probability that the system keeps working at the $4000th$ hour, which means that the reliability requirement specified in section 5 is satisfied in the modeled system.

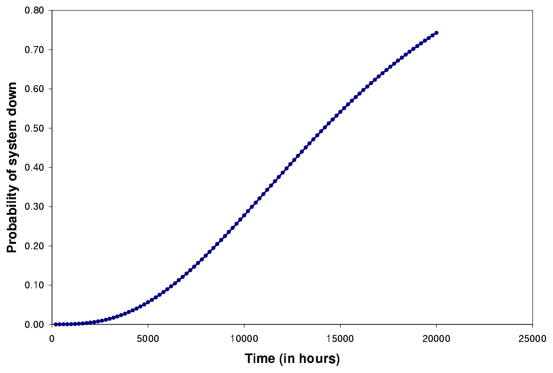

Figure 4. Reliability of the example system.

7. Conclusions

A method to extend SAM for performance and dependability analysis has been presented. After incorporating stochastic information into a SAM model, an SRN model is derived. By solving the SRN model using existing tools, a variety of non-functional properties such as performance, dependability and performability

[1]The failure rates are adopted from [3]

Table 2. Failure rates for the example system.

Elements	Failure rates (fault/hour)
processor	$\lambda_P = 5 \times 10^{-7}$
disk	$\lambda_D = 8 \times 10^{-5}$
shared memory	$\lambda_{GM} = 3 \times 10^{-8}$
local memory	$\lambda_{LM} = 3 \times 10^{-8}$
bus	$\lambda_B = 2 \times 10^{-9}$

can be evaluated. After the extension with stochastic construct, the SAM framework is capable of the analysis of both functional properties and the common non-functional properties. To illustrate the method, the multiprocessor system was used as a running example, and its reliability was evaluated. The presentation of the example and its numerical results proved the viability of the proposed method, and allowed significant insight to the system qualities to be gained.

The analysis of performance and dependability using SRNs is not new. Our contribution lies in incorporating stochastic techniques into the SAM framework so that both functional properties and non-functional properties like dependability can be analyzed under one unified framework. Currently, the transformation from SAM to SRN model is done by hand, we are interested in developing tools to automate the transformation. We also plan to do the tradeoff analysis of certain conflicting non-functional properties.

References

[1] I. F. Akyildiz. On the exact and approximate throughput analysis of closed queuing networks with blocking. *IEEE Transactions on Software Engineering.*, 14(1):62–70, 1988.

[2] R. Allen and D. Garlan. Formalizing architectural connection. In *Proc. the Sixteenth International Conference on Software Engineering*, pages 71–80, 1994.

[3] A. Bobbio, G. Franceschinis, R. Gaeta, and L. Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Transactions on Software Engineering*, 29(3):270–287, 2003.

[4] G. Ciardo, R. L. Jones, R. M. Marmorstein, A. S. Miner, and R. Siminiceanu. SMART: Stochastic model-checking analyzer for reliability and timing. In *Proc. International Conference on Dependable Systems and Networks (DSN'02)*, Washington, D.C., June 1992.

[5] G. Ciardo, J. Muppala, and K. S. Trivedi. Analyzing concurrent and fault-tolerant software using stochastic reward nets. *Journal of Parallel and Distributed Computing*, 15(3):255–269, 1992.

[6] H. J. Genrich. Predicate/transition nets. *Lecture Notes in Computer Science*, 254:255–269, 1987.

[7] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(4):512–535, 1994.

[8] X. He and Y. Deng. A framework for developing and analyzing software architecture specifications in SAM. *The Computer Journal*, 45(1):111–128, 2002.

[9] A. R. McSpadden and N. Lopez-Benitez. Stochastic petri nets applied to the performance evaluation of static task allocations in heterogeneous computing environments. In *Proc. 6th Heterogeneous Computing Workshop (HPC'97)*, pages 185–194, 1997.

[10] N. Medvidovic and R. Taylor. A framework for classifying and comparing architecture description languages. In *Proc. the 6th European conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 60–76, 1997.

[11] Y. Ren and J. B. Dugan. Design of reliable systems using static and dynamic fault trees. *IEEE Transactions on Reliability*, 47(3):234–244, May 1998.

[12] T. G. Robertazzi. *Computer Networks and Systems: Queuing Theory and Performance Evaluation.* Springer-Verlag, second edition, 1994.

[13] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik. Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering*, 21(4):314–355, 1995.

[14] T. Shi and X. He. Modeling and analyzing the software architecture of a communication protocol using SAM. In *Proc. IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture (WICSA'02)*, pages 63–77, 2002.

[15] K. S. Trivedi. *SPNP User's Manuel, version 6.0.* Department of Electronic and Computer Engineering, Duke University, 1999.

[16] K. S. Trivedi, G. Ciardo, M. Malhotra, and R. A. Sahner. Dependability and performability analysis. *Performance Evaluation of Computer and Communication Systems, Lecture Notes in Computer Science*, 729:587–612, 1993.

[17] K. S. Trivedi, M. Malhotra, and R. M. Fricks. Markov reward approach to performability and reliability analysis. In *Proc. the Second International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems (MASCOTS'94)*, pages 7–11, 1994.

[18] S. Vestal. MetaH programmer's manual, version 1.09. Technical report, Honeywell Technology Center, 1996.

[19] J. Wang, X. He, and Y. Deng. Introducing software architecture specification and analysis in SAM through an example. *Information and Software Technology*, 41(7):451–467, 1999.

[20] H. Yu, X. He, Y. Deng, and L. Mo. A formal method for analyzing software architecture models in SAM. In *Proc. the 26th Annual International Computer Software and Applications Conference (COMPSAC'02)*, pages 645–652, 2002.