Software Architecture Modeling and Evaluation Based on Stochastic Activity Networks

Ali Sedaghatbaf, Mohammad Azgomi

Software Architecture Modeling and Evaluation Based on Stochastic Activity Networks

Ali Sedaghatbaf and Mohammad Abdolahi Azgomi

School of Computer Engneering, University of Science and Technology, Tehran, Iran
ali_sedaghat@comp.iust.ac.ir
azgomi@iust.ac.ir

Abstract. Quantitative and integrated evaluation of software quality attributes at the architectural design stage provides a sound basis for making objective decisions for design trade-offs and developing a high quality software. In this paper we introduce a formal method for modeling software architectures and evaluationg their quality attributes quantitatively and in a unified manner. This method is based on stochastic activity networks (SANs) and the quality attributes considered include security, dependability and performance.

Keywords: Software architecture, quality attributes, quantitative evaluation, stochastic activity networks (SANs), reward structures.

1 Introduction

Dealing with quality attributes is one of the most difficult tasks in software engineering. To know whether a quality attribute is achieved, it has to be quantified by analysis or measured. However, not only quantification of each attribute has its own difficulties, but also they have complex dependencies.

In software systems, quality attributes are principally determined by the system's architecture. Evaluating quality attributes at the architectural design stage not only helps in assuring that stakeholders expectations are met, but also aids in discovering flaws in a shorter time and with lower cost than latter stages.

According to an investigation on different types of quality attributes and their application domains [1], security, dependability and performance are among the top quality attributes important for software systems. The necessity of the integrated evaluation of security and performance has gained much attention in research communities. However, a few have contributed to the quantitative evaluation of security. Dependability is a quality attribute closely related to both security and performance [2]. The necessity of the integrated evaluation of dependability and performance led to the derivation of a new quality attribute called perfomability. On the other hand, many of the methods proposed for quantitative security evaluation are inspired from dependability evaluation techniques. Therefore, despite the significant differences between security and performance, their integrated and quantitative evaluation can be performed regarding their close relation to dependability.

The purpose of this paper is to take a small step in the direction of developing a unified approach for reasoning about multiple quality attributes. The attributes considered include security, dependability and performance (called the SDP attributes in this paper). In this approach hierarchical colored stochastic activity networks (HCSANs) [5,6] are used for architecture modeling and activity-marking oriented reward structures [5] are used for evaluation.

Stochastic activity networks (SANs) [6] are stochastic extensions of Petri Nets, which are more powerful and more flexible than other stochastic extensions such as GSPNs and have been effectively used for performance, dependability, performability and security evaluations. HCSANs are extensions of SANs, whose hierarchical nature facilitates top-down and bottom-up model construction and their support for colored tokens facilitates complex data manipulations.

The remainder of this paper is organized as follows: in section 2 the related work is discussed. An introduction to HCSANs and activity-marking oriented reward struc-tures is provided in section 3. Section 4 presents the proposed approach and finally section 5 provides the concluding remarks and outlines the future work.

2 Related Work

Discrete-time Markov chains (DTMCs) are used in [7] to model software architectures and evaluate their security, performance and reliability. In this approach each component is modeled as a simple state and the arcs between states model the control flow between components. Quality attributes are evaluated by assigning reward functions to the states of the model.

In [8] a framework is proposed for analyzing the performance degradation induced by different security solutions. In this approach UML is used for modeling both the architecture and different security solutions. These models are then composed and converted to GSPN models for performance evaluation.

A methodology is proposed in [9] for combined performance and security risk analysis for border management systems. These systems are good examples of the systems in which both security and performance are critical. On one hand travelers should not linger because of security checks and on the other hand impostors should be distinguished from genuine travelers. In this approach the UML models of systems architecture are annotated with performance requirements. From these models LQN models are extracted for performance analysis. Also, cost curves are used to estimate the risk of misclassifying travelers with different classifiers.

In comparison to the above methods, the approach presented in this paper has the following distinguishing features:

- all the three SDP attributes can be evaluated quantitatively,
- the internal behavior of components can be modeled and analyzed,
- error propagation between components can be modeled,
- in contrast to many evaluation methods, any distribution function can be used for estimating the time spent by each software activity and

– the generality of the activity-marking oriented reward structures makes this approach extensible to other quality attributes.

3 HCSAN-based Reward Models

In addition to the five primitives of ordinary SANs (i.e. place, input gate, output gate, instantaneous activity and timed activity), Colored stochastic activity networks (CSANs) [3,4] have the following two primitives: (1) token type: a non-integer data type specifying the type of each token stored in a colored place and (2) colored place: a place maintaining a list of tokens with a specific token type. A selection policy (e.g. FIFO, LIFO, Priority) may be associated to each colored place specifying the order in which tokens are removed from that place.

HCSANs as an extension of CSANs, have one additional primitive, i.e. macro activity. A macro activity is a sub-model of an HCSAN model with a predefined interface. This interface includes a set of fusion places, which are virtual (colored) places that must be bound to concrete (colored) places in the encompassing model.

As a modification of the SAN-based reward structures, the reward structure of an HCSAN model can be defined formally as follows:

Definition 1. *An activity-marking reward structure of an HCSAN model with places* $P = SP \cup CP$ *and activities* $A = IA \cup TA \cup MA$ *is a pair of functions:*

– $C : A \to \Re$ *where for* $a \in A$, $C(a)$ *is the reward obtained due to the completion of activity* a, *and*
– $R : \wp(P, M) \to \Re$ *where for* $v \in \wp(P, M)$, $R(v)$ *is the rate of reward obtained when for each* $(p, m) \in v$ *the marking of place* p *is* m,

where $\Re$ *is the set of real numbers, and* $\wp(P, M)$ *is the set of all partial functions between* P *and* M.

In order to quantify the total reward associated with an HCSAN model at an instant of time t, variable V_t can be used, which is defined as follows:

$$V_t = \sum_{v \in \wp(P,M)} R(v).I_t^v + \sum_{a \in A} C(a).I_t^a \tag{1}$$

where I_t^v is a random variable indicating that for each $(p, m) \in v$, the marking of place p is m at time instant t, and the random variable I_t^a indicates that activity a is the most recently completed activity with respect to time instant t. If I_t^v and I_t^a converge in distribution for all v and a with non-zero rewards as t approaches ∞, then steady-state reward evaluation is also possible:

$$V_{t\to\infty} = \sum_{v \in \wp(P,M)} R(v).I_{t\to\infty}^v + \sum_{a \in A} C(a).I_{t\to\infty}^a \tag{2}$$

In order to evaluate the total reward accumulated in an interval $[t, t + \tau]$ variable $Y_{[t,t+\tau]}$ can be used, which can be expressed as:

$$Y_{[t,t+\tau]} = \sum_{v \in \wp(P,M)} R(v).J^v_{[t,t+\tau]} + \sum_{a \in A} C(a).N^a_{[t,t+\tau]} \quad (3)$$

where J^v_t is a random variable indicating the total time the model is in a marking such that for each $(p,m) \in v$, the marking of place p is m during $[t,t+\tau]$, and the random variable N^a_t indicates the number of completions of activity a during $[t,t+\tau]$. Variable $W_{[t,t+\tau]} = \frac{Y_{[t,t+\tau]}}{\tau}$ can be used to evaluate time-averaged measures.

4 The Proposed Approach

In this section we explain how to model software architectures and evaluate their SDP attributes with HCSAN-based reward models. We call this approach SAN-based architecture modeling (SANAM). In SANAM, HCSANs are used to define the behavior models of components and connectors and HCSAN-based reward structures are used to define and evaluate quality attributes.

A SANAM-based architecture model can be formally defined as a 4-tuple $SANAM = (CM, CN, HD, RS)$, where:

- $CM = \{cm_1, cm_2, \ldots, m_n\}$ is a set of component models such that each component model $cm = (IBM, PS, RS)$ consists of: an internal behavior model IBM specified with HCSANs, a set PS of provided services such that each provided service is modeled by a concrete macro activity and a set RS of required services, each modeled by a virtual macro activity which should be bound to a concrete macro activity providing the service.
- $CN = \{cn_1, cn_2, \ldots, cn_n\}$ is a set of connector models. Connectors are building blocks for modeling interactions among components.
- $HD = \{hd_1, hd_2, \ldots, hd_n\}$ is a set of hardware device models. Each software component or connector may be bound with a set of hardware devices such as processors, disks, links, etc. Speed, capacity, and failure behavior of these devices have significant impacts on the SDP attributes of software and
- RS is a set of HCSAN-based reward structures which can be used for specifying and evaluating the quality measures of interest.

As an illustrative example, consider a Group Communication System (GCS) used to store a set of documents and give users access to them. Several use cases can be defined for a GCS (e.g. subscribe, unsubscribe, submit a document, retrieve a docu-ment and update a document). In this paper we focus on document retrieval. The SANAM model of this system is depicted in Fig. 1. This model includes two software components (i.e. $CApp$ and $Serv$) representing the client application and the communication server respectively. $Serv$ provides one service (i.e. $rDoc$) which facilitates retrieving a document. This component is bound with two hardware resources i.e. the processor $SPrc$ and the disk $SDsk$, and its communication with $Serv$ is handled by the connector $CSPr$, which represents a client-server protocol. The behavior model of $CApp$ is depicted in Fig 2.

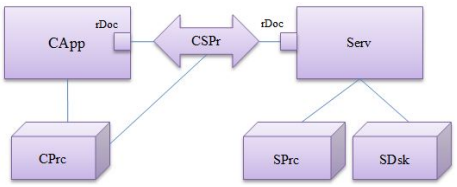
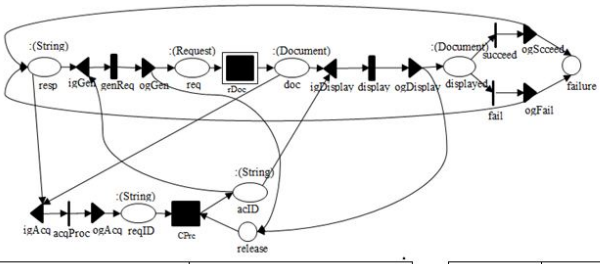


Fig. 1. SANAM model of a GCS system

This component iteratively generates requests, sends them to *Serv* and displays the responses. In this model the timed activities *genReq* and *display* are bound with the processor *CPrc*. The activity *genReq* (*display*) is enabled whenever a token is put in the place *resp* (*doc*) and it has acquired an idle processor i.e. the ID of this activity is put in the place *acID* by *CPrc*. After completion, this activity releases the acquired processor. If *display* fails, an error message will be displayed. Otherwise, the response of the server will be displayed which may be either a valid document or a server-side error message. The virtual macro activity *rDoc* corresponds to the required service of *CApp*. The behavior model



Gate	Enabling Predicate	Function
igGen	"gen" ∈ acID.Mark & resp.Mark != {}	acID.Mark.remove("gen"); resp.Mark.remove();
ogGen	-	req.Mark.add(someReq);
igDisplay	"display" ∈ acID.Mark & doc.Mark != {}	acID.Mark.remove("display"); rdoc = doc.Mark.remove();
ogDisplay	-	displayed.Mark.add(rdoc);
igAcq	doc.Mark != {} || resp.Mark != {}	-
ogAcq	-	if(doc.Mark != {}) acqID.Mark.add("display"); else acqID.Mark.add("gen");
ogFail	-	resp.Mark.add(error); failure.Mark = 1;
ogSucceed	-	resp.Mark.add(rdoc); if (rdoc == error) failure.Mark = 1;

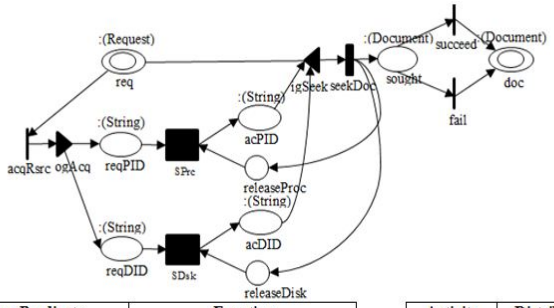
Activity	Distribution Function	Probability
getReq	exp(λ)	-
display	exp(μ)	-
succeed	-	p
fail	-	1 – p

Fig. 2. HCSAN model of *CApp*

of the service *Serv.rDoc* is presented in Fig. 3. This activity first requests access to the local disk and processor. If it acquires these resources, it will seek for the

requested document. In case of success, the content of the found document is put in the place *doc*, and a token representing an error message otherwise.

The behavior model of $CSPr$ includes two timed activities for transferring requests and documents between $Serv$ and $CApp$ (see Fig. 4). The activity *send* is enabled whenever a request is received from $CApp$ and an idle processor is available. After completion, if this activity succeeds in sending the request, the activity $Serv.rDoc$ will be enabled. Otherwise, the request token will be put back in req to try again. The behavior of $recv$ is similar to *send*. The only difference is the type of token they process. The two activities $intercept$ and $modify$ are



Gate	Enabling Predicate	Function
igSeek	"seek" ∈ acPID.Mark & req.Mark != {}	rqst = req.Mark.remove(); acPID.Mark.remove("seek"); acDID.Mark.remove("seek");
igAcq	req.Mark != {}	-
ogAcq		reqPID.Mark.add("seek"); reqDID.Mark.add("seek");

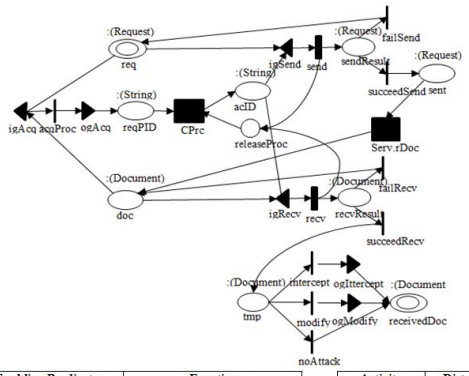
Activity	Distribution Function	Probability
seekDoc	exp(α)	-
succeed	-	q
fail	-	1 − q

Fig. 3. HCSAN model of $Serv.rDoc$

added to the behavior model of $CSPr$ to represent Man-in-the-Middle (MitM) attacks. In MitM attacks, an attacker establishes independent connections with the communicating parties and relays messages between them such that they believe that they are communicating directly over a private connection. But in fact, the connection is controlled by the attacker. As such, attacker will be able to intercept and modify the messages transferred between them.

For simplicity, the HCSAN models of the hardware resources are considered identical. As depicted in Fig. 5, the incoming requests which include the ID of the requesting activity are put in a queue and if at least one idle resource exists, one of the requests is approved probabilistically and its ID is put in the place resp. If the resource fails, it will be repaired. The order of processing requests is determined by the selection policy associated with the place queue i.e. FIFO. Whenever a timed activity releases a resource or the repair process completes, a token will be put in the place *idle*.

Now, to evaluate *reliability* as a dependability measure, the notion of system failure should be defined first. The GCS system fails when a token is put in the



Gate	Enabling Predicate	Function
igSend	"send" ∈ acPID.Mark & req.Mark != {}	rqst = req.Mark.remove(); acPID.Mark.remove("send");
igAcq	req.Mark != {} | doc.Mark != {}	-
ogAcq	-	If (req.Mark != {}) acPID.Mark.add("send"); else acDID.Mark.add("recv");
igRecv	"recv" ∈ acPID.Mark & doc.Mark != {}	rdoc = doc.Mark.remove(); acPID.Mark.remove("recv");
ogIntercept	-	receivedDoc.Mark.add(idoc);
ogModify	-	receivedDoc.Mark.add(mdoc);

Activity	Distribution Function	Probability
send	exp(β)	-
succeedSend	-	r
failSend	-	1 − r
recv	exp(γ)	-
succeedRecv	-	s
failRecv	-	1 − s
intercept	-	i
modify	-	m
noAttack	-	1 − i - m

Fig. 4. HCSAN model of $CSPr$

place $CApp.failure$. Therefore, the reliability of this system can be specified as:

$$C(a) = 0, \forall a \in A,$$
$$R(v) = \begin{cases} 1 & \text{if } v = \{(CApp.failure, 0)\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

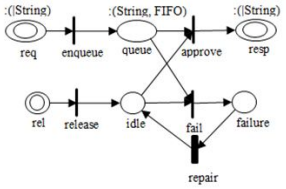
Performance measures can be evaluated in a similar way. For example, if we define the throughput of the GCS system during some interval $[t, t+\tau]$ as the number of documents successfully displayed for users in this interval, then the following reward structure can be used to specify throughput:

$$C(a) = \begin{cases} p & \text{if } a = CApp.display \text{ and } rdoc! = error \\ 0 & \text{otherwise} \end{cases} \tag{5}$$
$$R(v) = 0, \forall v \in \wp P(P, M)$$

where p is the success probability of the activity display and $rdoc$ is the token that this activity has removed from the place $CApp.doc$ (see Fig. 2).

To evaluate *confidentiality* as a security measure, we should determine in which states this attribute is compromised. The confidentiality of the GCS system is compromised whenever the content of a document is intercepted during transfer i.e. place $CApp.doc$ is marked with a token whose value is $idoc$. Therefore, the confidentiality attribute can be specified using the reward structure

$$C(a) = 0, \forall a \in A,$$
$$R(v) = \begin{cases} 0 & \text{if } v = \{(CApp.doc, idoc)\} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$



Activity	Distribution Function	Probability
approve	-	t
fail	-	1 - t
repair	exp (θ)	-

Fig. 5. HCSAN model of the hardware resources

5 Conclusions and Future Work

Regarding the necessity of integrated and quantitative evaluation of software quality attributes, we proposed SANAM as a formal method for modeling software architectures and evaluating their quality attributes in a unified manner. As future work we intend to define transformation rules to extract SANAM models from software modeling notations (e.g. UML, PCM, etc.) and develop a software tool for automating the transformation and evaluation procedures.

References

1. Mairiza, D., Zowghi, D. and Nurmuliani, N.: An investigation into the notion of non-functional requirements. In: ACM Sym. Applied Computing (SAC 2010), Sierra, Switzerland, pp. 311–317 (2010)
2. Bernardi, S., Merseguer, J. and Petriu, D.C.: Model-driven dependability assessment of software systems. Springer Berlin Heidelberg (2013)
3. Azgomi, M.A., Movaghar, A.: Coloured stochastic activity networks: preliminary definitions and behavior. In: 20th Annual UK Perf. Eng. Wksp., Bradford, UK, pp. 297–308 (2004)
4. Sedaghatbaf A. and Azgomi, M.A.: Attack modelling and security evaluation based on stochastic activity networks. J. Secur. Commu. Networks 7(4), 714–737 (2014)
5. Sanders, W.H., Meyer, J.F. and Arbor, A.: A unified approach for specifying measures of performance, dependability, and performability. J. Computing 4, 215–237 (1991)
6. Movaghar, A.: Stochastic Activity Networks: a new definition and some properties. Sci. Iran. 8, 303–311 (2001)
7. Sharma, V.S. and Trivedi, K.S.: Quantifying software performance, reliability and security: An architecture-based approach. J Syst. Softw. 80, 493–509 (2007)
8. Cortellessa, V., Trubiani, C., Mostarda, L. and Dulay, N.: An architectural framework for analyzing tradeoffs between software security and performance. First Int. Sym. Architecting Critical Syst. Prague, Czech Republic, pp. 1–18 (2010)
9. Sacanamboy, M. and Cukic, M.: Combined performance and risk analysis for border management applications. In Int. Conf. Dependable Syst. Networks, pp. 403–412 (2010)