Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 160608, 11 pages
http://dx.doi.org/10.1155/2014/160608

Research Article

Reliability Evaluation of Service-Oriented Architecture Systems Considering Fault-Tolerance Designs

Kuan-Li Peng and Chin-Yu Huang

Department of Computer Science, National Tsinghua University, Hsinchu 30013, Taiwan

Correspondence should be addressed to Chin-Yu Huang; cyhuang@cs.nthu.edu.tw

Received 18 September 2013; Accepted 15 November 2013; Published 9 January 2014

Academic Editor: Osamu Mizuno

Service-oriented architecture (SOA) provides an elastic and automatic way to discover, publish, and compose individual services. SOA enables faster integration of existing software components from different parties, makes fault tolerance (FT) feasible, and is also one of the fundamentals of cloud computing. However, the unpredictable nature of SOA systems introduces new challenges for reliability evaluation, while reliability and dependability have become the basic requirements of enterprise systems. This paper proposes an SOA system reliability model which incorporates three common fault-tolerance strategies. Sensitivity analysis of SOA at both coarse and fine grain levels is also studied, which can be used to efficiently identify the critical parts within the system. Two SOA system scenarios based on real industrial practices are studied. Experimental results show that the proposed SOA model can be used to accurately depict the behavior of SOA systems. Additionally, a sensitivity analysis that quantizes the effects of system structure as well as fault tolerance on the overall reliability is also studied. On the whole, the proposed reliability modeling and analysis framework may help the SOA system service provider to evaluate the overall system reliability effectively and also make smarter improvement plans by focusing resources on enhancing reliability-sensitive parts within the system.

1. Introduction

Service-oriented architecture (SOA) has become a major distributed computing framework [1]. With characteristics like standardized interfaces, loosely coupled structure, cross-platform as well as elastic service discovery, deployment, and reuse capabilities, SOA opens a new door to faster integration of existing software components from different parties, especially in the scheme of Web services (WS). Legacy components may still live within the system via service adapters [2], which is good for enterprises which prefer system upgrades in gentle and stable way.

It is noted that SOA also makes fault-tolerance (FT) techniques feasible for building reliable systems. Since it is difficult to build failure-free useful systems under limited development costs and the pressure of time to market, software fault tolerance [3], whose concepts originated from hardware reliability assurance, was proposed as an effective way to utilize redundancy to mask software failures and recover to normal operational states in a long running system. However, the extra costs of bringing out alternative software designs (redundancy) basically limit the applications of software fault tolerance to the fields that require ultrahigh reliabilities such as military, transportation, and aerospace. The emergence of service-oriented computing (SOC) helps lower the costs of making redundant software logic by reusing similar services published by different parties [4].

The unpredictability of open distributed environments where SOA systems are exposed, such as service fail-outs, service modification, linkage failures, and traffic congestion, threatens the dependability of the overall system. To ensure sufficient SOA system dependability, suitable reliability assessment and improvement methodologies are needed. Reliability modeling has been studied extensively in the field of software engineering, and many elegant solutions have emerged [5, 6], among which the component-based or architecture-based models [7, 8] appear to be most conceptually suited to be mapped to SOA systems. However, one of the major distinctions between traditional software system reliability models and SOA system reliability models are that the former models usually assume reliable communication between components and high transparency of system operation information, which generally is not applicable for SOA systems.

Based upon our previous research [9], in this paper, we further investigate the sensitivity analysis of each of the system's subparts, which may be useful in making critical system maintenance plans for resource allocation. Note that the proposed model is an extended version of the work by Wang et al. [10], whereas a large portion of changes and extensions have been made for SOA systems such as the incorporation of service-based concepts and more detailed considerations of fault-tolerance mechanisms. The major contributions of this paper are twofold. (1) The first is elegant SOA system reliability modeling considering unreliable services, unreliable communication links, and internal mechanisms of three typical fault-tolerance strategies. Note that existing reliability modeling methods tend to ignore the unreliabilities of communication links and oversimplify the FT mechanisms, which may deviate from the real situations. This paper is also among the earliest SOA reliability studies that adopt Markov-based analysis rather than popular path-based analysis, and therefore the complexities of tricky path analysis involving various branches and loops are eliminated. (2) The second is sensitivity analysis of SOA at two grain levels, which helps the system service provider to identify reliability-critical subparts and make smarter system improvement plans.

The remainder of the paper is organized as follows. Section 2 presents some work on the reliability modeling and the realization of FT on service-based systems. Section 3 describes the proposed system reliability model for SOA systems and presents the reliability sensitivity analysis. Section 4 is the experiment results and analysis. Then in Section 5, there are discussions of factors that should receive attention when the results of this research are extended to other systems. Finally, Section 6 concludes the paper. Supplements such as RcB reliability formulation, sensitivity analysis derivation, and manual path-based reliability computation are presented in the appendixes.

2. Related Works

SOA systems generally run in open and distributed environments, which introduces new sources of failures against traditional software systems such as interface changes, workflow inconsistency, time-outs, service-level agreement (SLA) constraints, and QoS constraints. Chan et al. [11] presented a fault taxonomy for service-oriented systems. In addition to the basic enabling protocols for SOA such as SOAP [12] and WSDL [13], protocols to enhance reliability on SOA systems have been proposed. WS-ReliableMessaging [14] allows SOAP messages to be delivered between services reliably. WS-coordination [15] describes a framework enabling coordination of transaction and workflow to operate in a heterogeneous environment. However, these protocols have not yet formed a complete SOA reliability solution and sometimes it is inevitable to interact with services without the support of those protocols. As mentioned in the preceding section, fault tolerance is suitable for applications that require high reliability, and various FT implementations under service-oriented environments are available such as [16–18], which makes it a viable option for SOA system engineers.

It is also noted that some models for evaluating the reliability of SOA systems have been proposed. Some focus on the reliability assessment of a single Web service. For example, Zheng et al. [19] designed a framework to retrieve the information of a large number of Web services worldwide, autogenerate testing modules, make testing invocations, and finally analyze the feedback from Web services to retrieve their reliability values. Other work, such as [20], evaluates the reliability of a composite service based on its structural information or considers basic FT behaviors on SOA systems, such as [4]. Existing models are helpful in understanding the QoS of service-oriented systems though improvements in the SOA reliability models are still desired, including more detailed considerations of FT mechanisms as well as uncertainties of communication links. This paper covers three major fault-tolerance designs, namely, *Recovery Block* (RcB), *N-version Programming* (NVP), and *Retry Block* (RtB), without assuming the implementation details. Basic introduction considerations of their exceptional behavior are briefly presented in the following.

Recovery block (RcB) [3] is a backward recovery technique that uses the acceptance test (AT) to check the outputs of a module; the next alternative is invoked once the former alternative fails the AT. The executive in RcB is responsible for checkpoint establishment, checkpoint restoration, invocation of the alternatives, and successful return of RcB. RcB is vulnerable to failure under conditions such as when (1) checkpoint establishment fails, (2) checkpoint restoration or invocation of the next alternative fails, or (3) the AT itself fails or no alternative passes the AT.

N-version programming (NVP) [3] is a forward recovery technique that uses the decision maker (DM) to vote for the outputs of all the involved alternative modules. The executive is responsible for input distribution and successful return of the NVP block. NVP is vulnerable to failure under conditions such as when (1) input distribution to the alternatives fails, (2) less than $\lfloor n/2 + 1 \rfloor$ consistent and correct results successfully reach the DM, or (3) the DM itself fails.

Retry block (RtB) [3] is a backward recovery technique that also uses AT like RcB to check the outputs of a module. Contrary to RcB, RtB retries the same module rather than using another module once the AT evaluation fails. The original design of RtB requires a data reexpression algorithm (DRA) to change the form of inputs before reusing the same module [3], but since in the Web applications or SOA systems failures may be caused by temporary service busy or network congestion (Heisenbug) and it is not always possible to tailor the DRA for the application, some designs eliminate DRA and use the same inputs on retry, and this simplified retry mechanism is widely supported in modern SOA execution environments. This paper considers the latter designs. RtB is vulnerable to failure under conditions such as when (1) checkpoint establishment fails, (2) checkpoint restoration fails, or (3) the AT itself fails or the retry limit is exceeded before the AT passes.

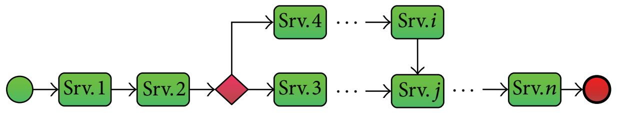

Figure 1: SOA service flow graph.

It should be noted that although the main function is protected by the redundant designs/executions in the fault-tolerance blocks, the executive and the AT/DM are not protected and may be a potential reliability bottleneck. Several modified designs of RcB and NVP such as [21, 22] also introduce redundancy at these points, resulting in even more complex fault-tolerance design and higher costs. Therefore such improvements are not considered in this paper.

3. SOA System Reliability Modeling and Analysis

Before the introduction of the proposed model, several definitions of SOA systems and reliability modeling are listed here.

3.1. Framework of the Model. Without loss of generality, an SOA system may be viewed as a flow of services (called *workflow*) and be depicted by a BPMN diagram [23] as in Figure 1, which contains the single *start event* (the thin-lined circle), *n abstract services* (round rectangles), *message* transmission between services (arrows), *branching points* (diamonds), and the single *end event* (the thick-lined circle). Specifically, this research takes the viewpoint of on-demand business process rather than long running process, where at the request of a client, the system starts at the virtual start event, passes control to the next abstract service along the message path, and finally stops normally and transfers to the *success state* when the control reaches the virtual end point. The workflow may branch and converge at some points.

In the realization of the SOA system, an abstract service is often fulfilled by one or more *physical services*, which could be an atomic Web service or a composite Web service which invokes one or more atomic/composite services internally, and messages travel in the network under the protocols such as SOAP [12]. Services may also be protected by the FT techniques in the form of a composite Web service. It should be noted that in the real world, failures (or exceptions) may originate from any service, control point, or communication path. Unhandled failures interrupt the normal process flow and turn the system into a *failure state*.

A number of notations in this paper are defined as follows.

Reliability. This paper views the operation of SOA systems as an on-demand business process, and the reliability is the probability of successful executions. This kind of time-invariant reliability definition is used in traditional software reliability estimation [7, 24] and later widely adopted in existing research on SOA reliability modeling that also considers on-demand cases, such as [1, 25, 26].

System Reliability (R). It is the probability of a system execution that finally reaches the success state, estimated by $R = 1 - f/t$, where t is the total number of system executions and f is the number of system executions that ended up in failure states.

Service Reliability (r_i). It is the probability of successful executions of physical service i, estimated by $r_i = 1 - f_i/t_i$, where t_i is the total number of service invocations and f_i is the number of service executions that threw unhandled exceptions.

Link Reliability $(l_{i;j})$. It is the probability of successful messages passing from physical service i to physical service j, estimated by $l_{i,j} = 1 - f_{i,j}/t_{i,j}$, where $t_{i,j}$ is the total number of messages passing from service i to service j and $f_{i,j}$ is the total number of unsuccessful messages passing from service i to service j.

System Reliability Model. The SOA system reliability model is denoted by a 5-tuple (Q, δ, s_1, s_k, M) [9, 10], where

(i) Q: a finite set of k states $\{s_1, s_2, \ldots, s_k\}$;

(ii) δ: state transition mapping $Q \times E \to Q$, and E is a set of triggering events;

(iii) s_1: the virtual initial state with reliability 1, which transits to all the real initial states with their corresponding initiating probabilities, and if there is exactly one initial state, s_1 can be replaced by that state for simplicity;

(iv) s_k: the virtual success final state with reliability 1, which is reached from all the real final states with their corresponding terminating probabilities;

(v) M: a $k \times k$ Markov transition matrix for the states in Q, whose entries are defined by

$$m_{i,j} = \begin{cases} p_{i,j}, & \text{if } \exists e \in E \text{ s.t. } \delta(s_i, e) = s_j, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $p_{i,j}$ is the transition probability from state i to state j.

To estimate the reliability of an SOA system, one may apply the following steps (Figure 2).

(1) Identify the workflow of the system and the physical services that realize each abstract service within the workflow. This may be obtained from process specifications, such as WS-BPEL [27] documents.

(2) Determine the reliability of each physical service and message link within the system. This may be estimated from the service access logs if they are available or be collected and estimated from user feedback.

(3) Determine the transition probabilities (or frequency) between abstract services. Existing techniques for building operational profiles [28] may apply here.

(4) Identify the internal states and construct the Markov transition matrix M from the information collected in Steps 1–3. This is explained further in Section 3.2.

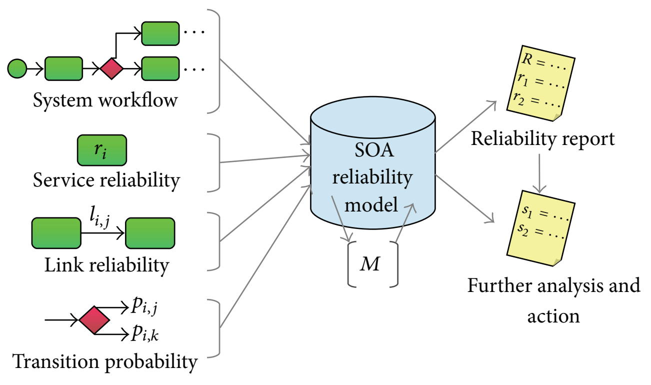

Figure 2: Framework of the model.

(5) Derive the reliability estimation of the whole SOA system.

(6) Optionally, perform further analysis (such as sensitivities) and action based on the results of Step 5.

Note that, as reported by Zheng and Lyu [1], different users may actually experience different performance from the same service, since the service performance is influenced by the communication links substantially, and some SOA system may provide different localized physical services depending on the users' preferences or location. Therefore, in Steps 2 and 3, the information may be collected together (from the viewpoint of the system operator) or be classified in groups of users individually (from the viewpoint of specific user groups) depending on the purpose of the reliability analysis.

3.2. Construction of Markov Transition Matrix M. In the system reliability model, the internal states are derived from abstract services. Each abstract service is mapped to a distinct macro-state. A macro-state consists of only one microstate unless its corresponding abstract service is guarded by fault tolerance, in which more than one tightly coupled microstate may be generated for that service. It should be noted that each microstate belongs to exactly one macro-state; each macro-state has exactly one entry micro-state and one exit micro-state and the two may overlap. Also remember that macro-state 1 is the initial state while the last macro-state is the final state as stated in Section 3.1. The transition matrix is obtained by

$$M = \left[m_{i,j}\right] = \begin{cases} \mathrm{sr}_i \ \mathrm{sp}_{i,j} \ \mathrm{sl}_{i,j}, & \text{if } \exists A, B \text{ s.t. } i = \omega_A \wedge j = \alpha_B, \\ f(i,j), & \text{else if } \exists A \text{ s.t. } i \in A \wedge j \in A, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In (2), sr_i, $\mathrm{sp}_{i,j}$, and $\mathrm{sl}_{i,j}$ denote state reliability, state transition probability, and state message link reliability, respectively; the entry and exit micro-states of a macro-state K are denoted by α_K and ω_K; and the inclusion relation of microstate i to macro-state K is denoted by $i \in K$.

In the first condition of (2), assuming that the corresponding physical services to micro-states i and j are services m and n respectively, then $\mathrm{sr}_i = r_m$ (It is the default case unless redefined later in this section.), $\mathrm{sp}_{i,j} = p_{m,n}$, and $\mathrm{sl}_{i,j} = l_{m,n}$. The method for retrieving the $p_{i,j}$ and $l_{i,j}$ values has been briefly explained earlier, and [9] has the interpretation of $l_{i,j}$ in detail under two major service composition schemes in SOA systems. The major problem left is to define $f(i,j)$ for the macro-states whose corresponding abstract services are realized in more complicated way, especially in various FT blocks.

Generally, for nonfault-tolerant services, either atomic or composite, it is intuitive to define a macro-state with a single micro-state, which reduces to the first condition and it is therefore not required to define $f(i,j)$ for them.

For any abstract service implemented in RcB, two approaches may be used. One way is to define micro-states separately for the executive, the AT, and each alternative. Assume that there are n alternatives in total, and both the microstate numbers as well as the corresponding physical service numbers for (executive, alternatives from the first to last, AT) are $(d, d+1 \cdots d+n, d+n+1)$. Then within the domain $d \le i, j \le d+n+1$, $f(i,j)$ is defined as follows:

$$f(i,j) = \begin{cases} r_d, & i = d, \quad j = d+1, \\ \left(1 - S_{d,i,d+n+1}\right) r_d r_{d+n+1}, & d+1 \le i < d+n, \\ & j = i+1, \\ S_{d,i,d+n+1}, & d+1 \le i \le d+n, \\ & j = d+n+1, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here a convenient substitution $S_{a,b,c} = l_{a,b} r_b l_{b,c}$ is used to simplify the formula. Further note that $\mathrm{sr}_{d+n+1} = r_{d+n+1}$. The second approach is to equivalently define a single microstate k with the state reliability computed by

$$\mathrm{sr}_k = \sum_{i=1}^{n} \left(S_{d,d+i,d+n+1} r_d^i r_{d+n+1}^i \prod_{j=1}^{i-1} \left(1 - S_{d,d+j,d+n+1}\right) \right). \tag{4}$$

The first approach is more intuitive while the second one results in more compact transition matrix. Refer to Appendix A for more on the RcB reliability formulation.

For any abstract service implemented in NVP, assuming that there are n alternatives in total and the corresponding physical service numbers for (executive, alternatives from the first to last, DM) are $(e, e+1 \cdots e+n, e+n+1)$, then one can also define a single microstate k with the state reliability computed by

$$\mathrm{sr}_k = r_e r_{e+n+1} \sum_{F_c \in 2^n} \prod_{1 \le i \le n} S_{e,e+i,e+n+1}^{F_{c,i}} \times \left(1 - S_{e,e+i,e+n+1}\right)^{1-F_{c,i}} \left[\left|F_c\right| > \frac{n}{2}\right], \tag{5}$$

where $[X]$ is an indicator function such that for condition X,

$$[X] = \begin{cases} 1, & \text{if } X \text{ is true,} \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

and F_c denotes a configuration set c of n binary values representing the outcomes $F_{c,i}$ of each alternative i defined as follows:

$$F_{c,i} = \begin{cases} 1, & \text{if alternative } i \text{ successfully passes DM,} \\ 0, & \text{otherwise;} \end{cases}$$
$$|F_c| = \sum_i [F_{c,i} = 1]. \tag{7}$$

For any abstract service implemented in RtB, assuming that the retry limit is γ and the service numbers for (executive, the retry alternative, AT) are numbered $(h, h+1, h+2)$, then its operation is equivalent to an RcB where there are γ duplicates of the alternative h. One may create $\gamma + 2$ micro-states and define $f(i, j)$ as in (3) or otherwise define a single microstate k with the state reliability computed by

$$\mathrm{sr}_k = \sum_{i=1}^{\gamma} \left(S_{h,h+1,h+2} \left(1 - S_{h,h+1,h+2}\right)^{i-1} r_h^i r_{h+2}^i \right). \tag{8}$$

Finally, once the transition matrix M is completed, the system reliability R is computed by

$$R = (-1)^{k+1} \frac{\left|(I - M)_{k,1}\right|}{|I - M|}, \tag{9}$$

according to [7], where I is the identity matrix of dimension $k \times k$ and $(I - M)_{k,1}$ is the minor matrix eliminating the last row and the leading column of $I - M$.

3.3. Sensitivity Analysis. Based on the previous reliability modeling, this section tries to explore the system more deeply and identifies the critical services with respect to their impact on the overall reliability of an SOA system.

Here the sensitivity analysis is made at two levels [10]: a coarse-grained level (L1, denoted by sc) and a fine-grained level (L2, denoted by sf). L1 involves the sensitivity of each macro-state within the reliability model, while L2 involves the sensitivity of each service within the SOA system. Substantial modifications of the original sensitivity analysis have been made for SOA systems, and link reliabilities are also included in this study. For convenience of analysis, here it is assumed that each macro-state consists of exactly one microstate and no physical services are shared among different abstract services.

A number of notations are defined as follows:

(i) $W = [w_{i,j}] = I - M$,

(ii) $E = [e_{i,j}] = (I - M)_{k,1}$,

(iii) $\alpha_{i,j}$: cofactor of $w_{i,j}$,

(iv) $\beta_{i,j}$: cofactor of $e_{i,j}$.

In L1, (9) can be rewritten as

$$R = (-1)^{k+1} \frac{|E|}{|W|}, \tag{10}$$

$$|W| = \theta_{1,i} + \left(\theta_{2,i} + \theta_{3,i}\right) \mathrm{sr}_i, \tag{11}$$

where $\theta_{1,i} = \alpha_{i,i}$, $\theta_{2,i} = -\alpha_{i,i}\mathrm{sp}_{i,i}\mathrm{sl}_{i,i}$, and $\theta_{3,i} = -\sum_{1\le j\le k, j\ne i} \alpha_{i,j}\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}$. It is noted that none of $\theta_{1,i}$, $\theta_{2,i}$, or $\theta_{3,i}$ is a function of sr_i. In addition,

$$|E| = \sigma_{1,i} + \left(\sigma_{2,i} + \sigma_{3,i}\right) \mathrm{sr}_i, \tag{12}$$

where $\sigma_{1,i} = \beta_{i,i-1}$, $\sigma_{2,i} = -\beta_{i,i-1}\mathrm{sp}_{i,i}\mathrm{sl}_{i,i}$, and $\sigma_{3,i} = -\sum_{2\le j\le k, j\ne i} \beta_{i,j-1}\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}$ for $1 < i < k$. It is also noted that none of $\sigma_{1,i}$, $\sigma_{2,i}$, or $\sigma_{3,i}$ is a function of sr_i.

Then,

$$R = (-1)^{k+1} \mathrm{sr}_k \frac{\sigma_{1,i} + \left(\sigma_{2,i} + \sigma_{3,i}\right) \mathrm{sr}_i}{\theta_{1,i} + \left(\theta_{2,i} + \theta_{3,i}\right) \mathrm{sr}_i}, \quad \text{for } 1 < i < k, \tag{13}$$

and the sensitivity for microstate i is

$$\mathrm{sc}_i = \frac{\partial R}{\partial \mathrm{sr}_i} = (-1)^{k+1} \mathrm{sr}_k \frac{\theta_{1,i}\left(\sigma_{2,i} + \sigma_{3,i}\right) - \left(\theta_{2,i} + \theta_{3,i}\right)\sigma_{1,i}}{\left(\theta_{1,i} + \left(\theta_{2,i} + \theta_{3,i}\right)\mathrm{sr}_i\right)^2}, \quad \text{for } 1 < i < k. \tag{14}$$

sc_i is only defined over $1 < i < k$ because macro-states 1 and k are virtual states. Related derivations of $|W|$ and $|E|$ are given in Appendix B.

Next in L2, the sensitivity analysis is made by different categories of the abstract services connected to each macro-state.

For a nonfault-tolerant service k, since there is only one physical service p associated with the macro-state, the state sensitivity is consistent with the service sensitivity. That is,

$$\mathrm{sf}_p = \mathrm{sc}_k. \tag{15}$$

For a service k implemented in RcB, assuming the same service number settings in Section 3.2 and recalling the state reliability for RcB in (4), then the sensitivity of the executive d is

$$\mathrm{sf}_d = \frac{\partial R}{\partial r_d} = \frac{\partial R}{\partial \mathrm{sr}_k}\frac{\partial \mathrm{sr}_k}{\partial r_d} = \mathrm{sc}_k \sum_{i=1}^{n} \left(i S_{d,d+i,d+n+1} r_d^{i-1} r_{d+n+1}^i \prod_{j=1}^{i-1} \left(1 - S_{d,d+j,d+n+1}\right) \right), \tag{16}$$

the sensitivities of the alternatives $d + 1 \cdots d + n$ are

$$\mathrm{sf}_{d+a} = \frac{\partial R}{\partial r_{d+a}} = \frac{\partial R}{\partial \mathrm{sr}_k}\frac{\partial \mathrm{sr}_k}{\partial r_{d+a}} = \mathrm{sc}_k l_{d,d+a} l_{d+a,d+n+1} (A + B), \tag{17}$$

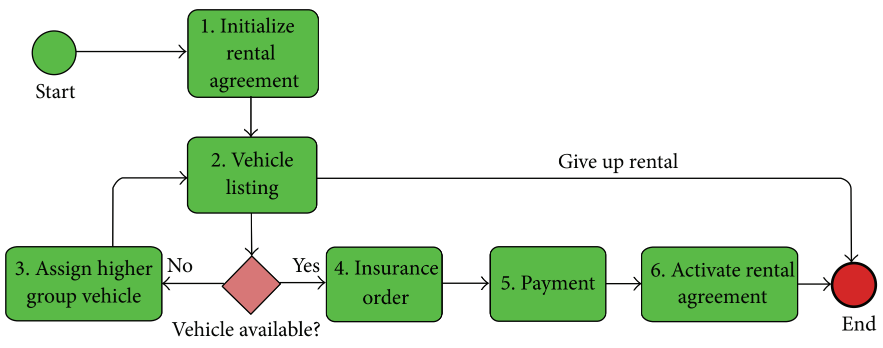

FIGURE 3: BPMN diagram.

where

$$A = r_d^a r_{d+n+1}^a \prod_{j=1}^{a-1} \left(1 - S_{d,d+j,d+n+1}\right)$$

$$B = -\sum_{a<i\leq n} \left(S_{d,d+i,d+n+1} r_d^i r_{d+n+1}^i \prod_{1\leq j<i, j\neq a} \left(1 - S_{d,d+j,d+n+1}\right)\right), \tag{18}$$

and the sensitivity of the AT $d + n + 1$ is

$$\begin{aligned} \mathrm{sf}_{d+n+1} &= \frac{\partial R}{\partial r_{d+n+1}} = \frac{\partial R}{\partial \mathrm{sr}_k} \frac{\partial \mathrm{sr}_k}{\partial r_{d+n+1}} \\ &= \mathrm{sc}_k \sum_{i=1}^{n} \left(i S_{d,d+i,d+n+1} r_d^i r_{d+n+1}^{i-1} \prod_{j=1}^{i-1} \left(1 - S_{d,d+j,d+n+1}\right)\right). \end{aligned} \tag{19}$$

For a service k implemented in NVP, note that (5) is not in a simple closed form. To simplify the discussion, here it is assumed that 3 alternatives in total are in the NVP block. Equation (5) then reduces to

$$\begin{aligned} \mathrm{sr}_k = r_e r_{e+4} [& S_{e,e+1,e+4} S_{e,e+2,e+4} \left(1 - S_{e,e+3,e+4}\right) \\ & + S_{e,e+1,e+4} \left(1 - S_{e,e+2,e+4}\right) S_{e,e+3,e+4} \\ & + \left(1 - S_{e,e+1,e+4}\right) S_{e,e+2,e+4} S_{e,e+3,e+4} \\ & + S_{e,e+1,e+4} S_{e,e+2,e+4} S_{e,e+3,e+4}] . \end{aligned} \tag{20}$$

The sensitivity of the executive e is

$$\mathrm{sf}_e = \frac{\partial R}{\partial r_e} = \frac{\partial R}{\partial \mathrm{sr}_k} \frac{\partial \mathrm{sr}_k}{\partial r_e} = \frac{\mathrm{sc}_k \mathrm{sr}_k}{r_e}. \tag{21}$$

The sensitivities of the alternatives $e + 1 \cdots e + 3$ are

$$\mathrm{sf}_{e+a} = \frac{\partial R}{\partial r_{e+a}} = \frac{\partial R}{\partial \mathrm{sr}_k} \frac{\partial \mathrm{sr}_k}{\partial r_{e+a}}, \tag{22}$$

where

$$\begin{aligned} \frac{\partial \mathrm{sr}_k}{\partial r_{e+1}} = r_e r_{e+4} [& l_{e,e+1} l_{e+1,e+4} S_{e,e+2,e+4} \left(1 - S_{e,e+3,e+4}\right) \\ & - l_{e,e+1} l_{e+1,e+4} \left(1 - S_{e,e+2,e+4}\right) S_{e,e+3,e+4} \\ & + l_{e,e+1} l_{e+1,e+4} S_{e,e+2,e+4} S_{e,e+3,e+4} \\ & + l_{e,e+1} l_{e+1,e+4} S_{e,e+2,e+4} S_{e,e+3,e+4}] \end{aligned} \tag{23}$$

and $\partial \mathrm{sr}_k / \partial r_{e+2}$ as well as $\partial \mathrm{sr}_k / \partial r_{e+3}$ can be derived similarly.

The sensitivity of the DM $e + 4$ is

$$\mathrm{sf}_{e+4} = \frac{\partial R}{\partial r_{e+4}} = \frac{\partial R}{\partial \mathrm{sr}_k} \frac{\partial \mathrm{sr}_k}{\partial r_{e+4}} = \mathrm{sc}_k \frac{\mathrm{sr}_k}{r_{e+4}}. \tag{24}$$

Equations for the sensitivities of the RtB services are similar to those for the RcB services.

4. Experiments and Discussion

In this paper, results and discussions for a vehicle rental system adapted from industrial practices [29] are presented. The workflow diagram (Figure 3) for the system has sequences, branching, and looping, common in real-world projects. According to the diagram, when a customer uses the service, a rental agreement is prepared in the system, and then, based on the customer's preferences, the system searches for vehicle choices from the database. The customer may either accept a choice, call for more choices, or drop the service directly. Upon accepting the rental choice, the system then assists the user in contacting the insurance company and paying the security deposit through a third-party payment service. Finally, the system activates the rental agreement and the customer is guided to take his car.

Two scenarios based upon this workflow specification are incorporated: scenario no. 1 is the fault-tolerance-free version where each abstract service is fulfilled by one physical (either internal or external) service, while scenario no. 2 uses RcB to ensure higher reliability for the payment operation. For each scenario, Table 1 displays the physical services involved and their reliabilities, Table 2 displays the transition (branching) probabilities between the abstract services, and Table 3 displays the link reliabilities between the physical services.

Table 1: Service configuration.

Abstract srv.	Scenario no. 1		Scenario no. 2	
	Physical srv.	Srv rel.	Physical srv.	Srv rel.
1. RA init.	Int srv (1)	0.99	Int srv (1)	0.99
2. Veh. listing	Int DB srv (2)	0.97	Int DB srv (2)	0.97
3. Higher-grp.-veh. Asgmnt.	Int DB srv (3)	0.97	Int DB srv (3)	0.97
4. Insurance order	Ext srv (4)	0.91	Ext srv (4)	0.91
5. Payment	Ext srv (6)	0.91	Int exec (5)	0.99
			Ext alt I (6)	0.91
			Ext alt II (7)	0.88
			Ext alt III (8)	0.85
			Int AT (9)	0.99
6. RA activation	Int DB srv (10)	0.97	Int DB srv (10)	0.97

Table 2: Abstract service transition probability.

Abstract srv trans.	Prob
2. → 3.	0.25
2. → 4.	0.40
2. → End	0.35

Table 3: Physical service link reliability.

Physical srv lnk	Rel
Start → (1)	1.00
(1) → (2)	1.00
(2) → (3)	1.00
(2) → (4)	0.98
(2) → End	1.00
(3) → (2)	1.00
(4) → (5)	1.00
(4) → (6)	0.98
(6) → (10)	0.98
(5) → (6)	0.98
(5) → (7)	0.98
(5) → (8)	0.96
(6) → (9)	0.98
(7) → (9)	0.98
(8) → (9)	0.96
(9) → (10)	1.00
(10) → End	1.00

4.1. Reliability Modeling. Applying the Steps in Section 3.2, one can derive the transition matrix for scenario no. 1

$$M_{\text{scn1}} = \begin{bmatrix} 0 & 0.99 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.242 & 0.38 & 0 & 0 & 0.339 \\ 0 & 0.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.892 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.892 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.97 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (25)$$

and the transition matrix for scenario no. 2

$$M_{\text{scn2}} = \begin{bmatrix} 0 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.242 & 0.38 & 0 & 0 & 0 & 0 & 0 & 0 & 0.339 \\ 0 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.99 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.124 & 0.874 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.845 & 0.152 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.783 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.97 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (26)$$

Ten simulations of the system where each run contained 100,000 system calls were executed for each scenario. Comparing the experimental results and the values from the proposed model in Table 4. In Table 4, the 1st row is the reliability estimation by the proposed model, the 2nd row is from the manual probability computation as explained in Appendix C, the 3rd row contains the statistics of simulation, and the last row is the relative error between the simulation results and the theoretical values in the 3rd row. It can be seen that the proposed SOA reliability model has identical values to those computed from probability theory, and the average simulation results are very close to the theoretical values. Comparing the two scenarios, it can also be seen that the introduction of RcB in the payment service (abstract service 5) improves the local service reliability by 11.36% (Considering both the service invocation and the service execution, the payment service reliability is 0.874 in scenario no. 1 and is 0.973 in scenario no. 2.) and improves the overall system reliability by 5.2%.

It is also clarified that the 0.00% simulation relative error in Table 4 is only the statistical value for certain scenario in certain precision level and should not be otherwise interpreted as 100% simulation accuracy.

4.2. Sensitivity Analysis. Results of the sensitivity analysis for both scenarios are displayed in Tables 5 and 6, respectively. The sensitivity value of a state/service indicates its impact on the overall system reliability. It could also be observed from Figure 4 that abstract services with higher execution probabilities (such as abstract services 1 and 2) have higher sensitivity values. In abstract service 5 in scenario no. 2, it

Table 4: System reliability model validation results.

	Scenario no. 1	Scenario no. 2
SOA reliability model	0.819	0.862
Probability computation	0.819	0.862
Simulation results	0.819	0.865
Simulation relative error	0.00%	0.35%

Table 5: Sensitivity analysis for scn. no. 1.

Level 1		Level 2	
State no.	Val.	Physical srv no.	Val.
1.	0.827	(1)	0.827
2.	1.104	(2)	1.104
3.	0.260	(3)	0.260
4.	0.417	(4)	0.417
5.	0.417	(6)	0.417
6.	0.391	(10)	0.391

Table 6: Sensitivity analysis for scn. no. 2.

Level 1		Level 2	
State no.	Val.	Physical srv no.	Val.
1.	0.871	(1)	0.871
2.	1.162	(2)	1.162
3.	0.273	(3)	0.273
4.	0.465	(4)	0.465
		(5)	0.485
		(6)	0.023
5.	0.434	(7)	0.012
		(8)	0.007
		(9)	0.485
6.	0.436	(10)	0.436

is also noted that the executive service and the AT service actually dominate the reliability of RcB, as one observes the sub-graph within Figure 4. Such result is reasonable since both services are always on the execution paths for all the alternatives. On the other hand, the sensitivity values of the alternatives are significantly reduced (94% for physical service 6) in the RcB because their failures are largely masked by invoking the succeeding alternatives. Sensitivity analysis effectively quantifies the importance of each part within the system, which may be very useful for system engineers in determining reliability bottlenecks of the system and making further system improvement plans.

5. Threats to Validity

Users who adopt the proposed SOA reliability model and sensitivity analysis technique should be aware of the potential threats to validity. Factors limiting the generalizability of the results include the following.

Reliability Model. The proposed reliability model assumes independent failure occurrences between services. The Markovian property [30] also implies history independent system behavior. Possible bias should be taken into account when these assumptions do not hold in application.

Sensitivity Analysis. The derivative-based sensitivity analysis adopted in this paper is efficient to calculate sensitivity values at the current computing point. The major limitation of such technique lies on the narrow parameter input space, and higher-order sensitivity indices are also not explored [31]. Care must be taken when there are uncertainties of model inputs or when there is strong interaction/dependency between the constituent services within the SOA system.

Subject Systems. The experiments are performed on a set of scenarios based on real-world project presented in [29], and the settings of service reliability, link reliability, and transmission probability are constructed in this research. To the best of our knowledge, no public reliability data source for SOA systems is available. Researches on SOA reliability modeling such as [4, 8, 20, 25] generated their own scenarios for validation either by simulation or by operating their own example systems. Therefore, care must be taken in applying the model to other subject systems or system configuration.

Performance Evaluation. The restrictions of the proposed model also apply in the experiments. Additionally, failures are generated binomially in the simulation. The evaluation results are valid only with respect to the conditions and system scales in the experiments.

6. Conclusions

This paper presents a Markov-based system reliability model for SOA systems. Starting from the workflow specification of

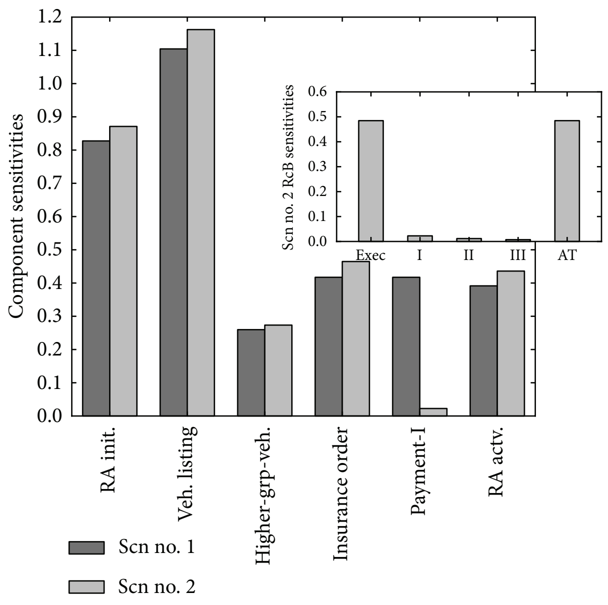

Figure 4: Component sensitivities. The main figure shows component sensitivity comparisons of scenarios no. 1 and no. 2. The subfigure shows the RcB internal sensitivities of the payment service in scenario no. 2.

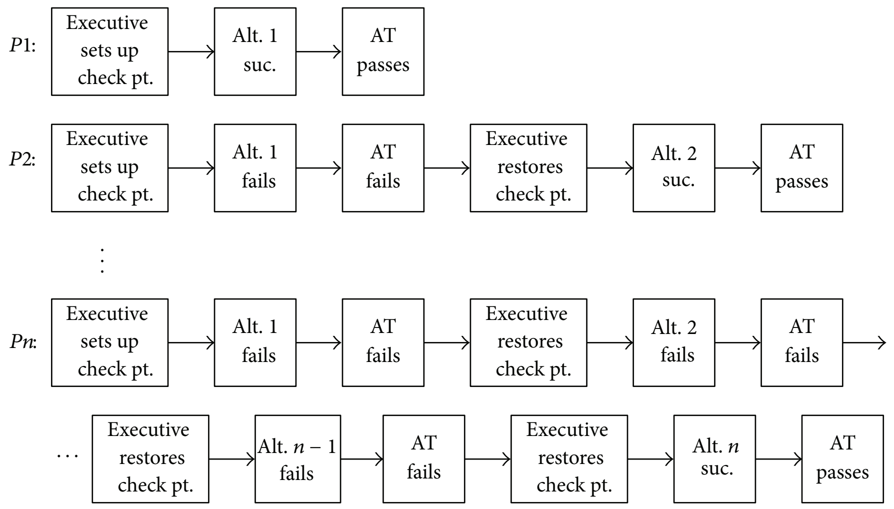
P1:
Executive sets up check pt.
Alt. 1 suc.
AT passes
P2:
Executive sets up check pt.
Alt. 1 fails
AT fails
Executive restores check pt.
Alt. 2 suc.
AT passes
Pn:
Executive sets up check pt.
Alt. 1 fails
AT fails
Executive restores check pt.
Alt. 2 fails
AT fails
Executive restores check pt.
Alt. n − 1 fails
AT fails
Executive restores check pt.
Alt. n suc.
AT passes

FIGURE 5: RcB Path Set.

an SOA system, we have shown how to map each part in the workflow to the model. The proposed reliability model also considers the internal mechanisms of three well-known fault-tolerance strategies such that the execute node, the AT/DM node, the alternatives, and the interactions between them are well reflected in the reliability model. Sensitivity analysis at two grain levels is also incorporated in this paper, which enables the SOA system engineers to identify the reliability critical blocks or internal services effectively and efficiently. Experimental results show that the proposed SOA model and methods give very close results to theoretical and simulation values and how the sensitivity analysis quantizes the effects of system structure as well as fault tolerance on the overall reliability.

Appendix

A. RcB Reliability Formulation

It is possible that the reliability formulation depends on different design variations. Since the implementation is not the focus of this paper, we assume the basic RcB scheme as described in Section 2, where an RcB with n alternatives will successfully execute in one of the paths in Figure 5 ("AT passes" blocks in Figure 5 mean that the AT itself is functional and returns true positive result. On the other hand, "AT fails" blocks mean that the AT itself is still functional but returns true negative or false results.). Equation (4) can then be derived from summing the probabilities of all the above paths together.

On the other hand, in (3), the first condition is for the checkpoint establishment, which is made by the executive (d). The second one is for shifting the execution to the next alternative ($i + 1$), conditioning on the AT ($d + n + 1$) successfully identifying the failure of the current alternative (i), and successful checkpoint restoration by the executive (d). The third one is for successful execution of the current alternative (i) and passes of AT ($d + n + 1$). The last one is for all remaining exceptional operations that cannot be guarded by the RcB.

An example is presented here to show the equivalence of (3) and (4). For a system with exactly one 3-RcB, where the executor, the three alternatives, and the AT are numbered from 1 to 5, respectively, by applying (3), the transition matrix M_1 is

$$M_1 = \begin{bmatrix} 0 & r_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_1 r_5 \left(1 - S_{1,2,5}\right) & 0 & S_{1,2,5} & 0 \\ 0 & 0 & 0 & r_1 r_5 \left(1 - S_{1,3,5}\right) & S_{1,3,5} & 0 \\ 0 & 0 & 0 & 0 & S_{1,4,5} & 0 \\ 0 & 0 & 0 & 0 & 0 & r_5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{A.1}$$

By applying (4), the transition matrix M_2 is

$$M_2 = \begin{bmatrix} 0 & \begin{matrix} S_{1,2,5} r_1 r_5 + S_{1,3,5} r_1^2 r_5^2 \left(1 - S_{1,2,5}\right) \\ + S_{1,4,5} r_1^3 r_5^3 \left(1 - S_{1,2,5}\right)\left(1 - S_{1,3,5}\right) \end{matrix} \\ 0 & 0 \end{bmatrix}. \tag{A.2}$$

It can be verified that $R_1 = (-1)^{k+1}(|(I - M_1)_{6,1}|/|I - M_1|)$ and $R_2 = (-1)^{k+1}(|(I - M_2)_{2,1}|/|I - M_2|)$ are equal.

B. Derivation of Sensitivity Analysis

Let $M = [m_{i,j}] = (-1)^{k+1}|E|/|W|$ be the transition matrix in the reliability modeling of an SOA system. Then,

$$\begin{aligned} |W| &= \sum_{j=1}^{n} \alpha_{i,j} w_{i,j} \\ &= \alpha_{i,i} w_{i,i} + \sum_{1 \le j \le n, j \ne i} \alpha_{i,j} w_{i,j} \\ &= \alpha_{i,i} \left(1 - m_{i,i}\right) + \sum_{1 \le j \le n, j \ne i} \alpha_{i,j} \left(-m_{i,j}\right) \end{aligned}$$

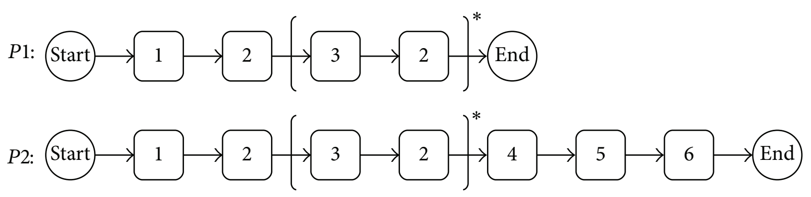

FIGURE 6: Example system path set.

$$= \alpha_{i,i} - \alpha_{i,i}\mathrm{sr}_i\mathrm{sp}_{i,i}\mathrm{sl}_{i,i} - \sum_{1\le j\le k, j\ne i} \alpha_{i,j}\mathrm{sr}_i\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}$$
$$= \alpha_{i,i} + \left(-\alpha_{i,i}\mathrm{sp}_{i,i}\mathrm{sl}_{i,i} - \sum_{1\le j\le k, j\ne i} \alpha_{i,j}\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}\right)\mathrm{sr}_i$$
$$= \theta_{1,i} + (\theta_{2,i} + \theta_{3,i})\,\mathrm{sr}_i. \tag{B.1}$$

The derivation of $|E|$ is similar, and therefore some of the steps are eliminated. For $1 < i < k$,

$$|E| = \sum_{j=1}^{n} \beta_{i,j}e_{i,j}$$
$$= \beta_{i,i-1}\left(1 - \mathrm{sr}_i\mathrm{sp}_{i,i}\mathrm{sl}_{i,i}\right) + \sum_{2\le j\le k, j\ne i}^{n} \beta_{i,j-1}\left(-\mathrm{sr}_i\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}\right)$$
$$= \beta_{i,i-1} + \left(-\beta_{i,i-1}\mathrm{sp}_{i,i}\mathrm{sl}_{i,i} - \sum_{2\le j\le k, j\ne i}^{n} \beta_{i,j-1}\mathrm{sp}_{i,j}\mathrm{sl}_{i,j}\right)\mathrm{sr}_i$$
$$= \sigma_{1,i} + (\sigma_{2,i} + \sigma_{3,i})\,\mathrm{sr}_i. \tag{B.2}$$

C. Reliability Analysis by Probability Computation

The execution of the example system in Section 4 (Figure 3) can be broken down into a set of execution paths as follows (Figure 6).

Suppose that the physical services corresponding to abstract services 1–6 are numbered $s1$–$s6$, respectively. For scenario no. 1, $r_{s1} = r_1, r_{s2} = r_2, r_{s3} = r_3, r_{s4} = r_4, r_{s5} = r_6$, and $r_{s6} = r_{10}$ as in Table 1. For scenario no. 2, r_{s6} is replaced by the reliability of the RcB composite service consisting of physical services 5–9, whose value can be derived from (4).

Then, the following probabilities are computed:

$$\Pr[P1 \text{ successful} \mid P1 \text{ taken}]$$
$$= r_{s1}l_{s1,s2}r_{s2} \times \sum_{n=0}^{\infty}(p_{2,3}l_{s2,s3}r_{s3}l_{s3,s2}r_{s2})^n(1 - p_{2,3})$$
$$= r_{s1}l_{s1,s2}r_{s2}\,\frac{1 - p_{2,3}}{1 - p_{2,3}l_{s2,s3}r_{s3}l_{s3,s2}r_{s2}},$$
$$\Pr[P2 \text{ successful} \mid P2 \text{ taken}]$$
$$= r_{s1}l_{s1,s2}r_{s2} \times \sum_{n=0}^{\infty}(p_{2,3}l_{s2,s3}r_{s3}l_{s3,s2}r_{s2})^n \times (1 - p_{2,3})\,l_{s2,s4}r_{s4}l_{s4,s5}r_{s5}l_{s5,s6}r_{s6}$$
$$= r_{s1}l_{s1,s2}r_{s2}\,\frac{1 - p_{2,3}}{1 - p_{2,3}l_{s2,s3}r_{s3}l_{s3,s2}r_{s2}} \times l_{s2,s4}r_{s4}l_{s4,s5}r_{s5}l_{s5,s6}r_{s6},$$
$$\Pr[P1 \text{ taken}] = \Pr[2 \longrightarrow \text{End} \mid 2 \nrightarrow 3] = \frac{p_{2,\text{End}}}{1 - p_{2,3}},$$
$$\Pr[P2 \text{ taken}] = \Pr[2 \longrightarrow 4 \mid 2 \nrightarrow 3] = \frac{p_{2,4}}{1 - p_{2,3}}. \tag{C.1}$$

In the previous equations, $x \rightarrow y$ denotes transferring to abstract service y upon completion of abstract service x, and $x \nrightarrow y$ denotes not transferring to abstract service y upon completion of abstract service x.

Finally, the system probability is obtained by summing the probabilities of successful execution of each path as follows:

$$R = \sum_{p\in\text{path set}} \Pr[p \text{ successful} \mid p \text{ taken}] \cdot \Pr[p \text{ taken}]. \tag{C.2}$$

Through manual path-based reliability computation, one potential advantage of our Markov-based method would be discovered that it avoids the complexities of path analysis involving branches and loops, which may be tricky to automate in some cases.

Conflict of Interests

The authors have no financial relations with the commercial identities related to the technologies or standards covered in this paper.

Acknowledgments

The work described in this paper was supported by the National Science Council, Taiwan, under Grants NSC 101-2221-E-007-034-MY2 and NSC 101-2220-E-007-005. Further, the authors would like to thank the anonymous referees for their critical review and valuable comments.

References

[1] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10)*, pp. 35–44, May 2010.

[2] H. M. Sneed, "Integrating legacy software into a service oriented architecture," in *Proceedings of the 10th European Conference on Software Maintenance and Reengineering (CSMR '06)*, pp. 11–14, March 2006.

[3] M. Lyu, *Software Fault Tolerance. Trends in Software*, Wiley, 1995.

[4] Z. Zheng and M. R. Lyu, "An adaptive QoS-aware fault tolerance strategy for web services," *Empirical Software Engineering*, vol. 15, no. 4, pp. 323–345, 2010.

[5] K. Goševa-Popstojanova, A. P. Mathur, and K. S. Trivedi, "Comparison of architecture-based software reliability models," in *Proceedings of the 12th International Symposium on Software Reliability Engineering*, pp. 22–31, IEEE, November 2001.

[6] S. S. Gokhale and K. S. Trivedi, "Analytical models for architecture-based software reliability prediction: a unification framework," *IEEE Transactions on Reliability*, vol. 55, no. 4, pp. 578–590, 2006.

[7] R. C. Cheung, "A user-oriented software reliability model," *IEEE Transactions on Software Engineering*, no. 2, pp. 118–125, 1980.

[8] V. Grassi, "Architecture-based reliability prediction for service-oriented computing," in *Architecting Dependable Systems III*, pp. 279–299, Springer, 2005.

[9] K.-L. Peng and C.-Y. Huang, "Reliability assessment and analysis of incorporating fault tolerance into service-oriented architectural systems," in *Proceedings of the IEEE Internation Conference Industrial Engineering and Engineering Management (IEEM '12)*, 2012.

[10] W.-L. Wang, D. Pan, and M.-H. Chen, "Architecture-based software reliability modeling," *Journal of Systems and Software*, vol. 79, no. 1, pp. 132–146, 2006.

[11] K. M. Chan, J. Bishop, J. Steyn, L. Baresi, and S. Guinea, "A fault taxonomy for web service composition," in *Proceedings of the Workshops of Service-Oriented Computing (ICSOC 707)*, pp. 363–375, Springer, 2009.

[12] *Soap Version 1.2—part 1: Messaging Framework*, 2nd edition, 2007.

[13] *Web Services Description Language (Wsdl) Version 2.0—part 1: Core Language*, 2007.

[14] *Web Services Reliable Messaging (Ws-ReliableMessaging) Version 1.2*, 2009.

[15] *Web Services Coordination (Ws-Coordination) Version 1.2*, 2009.

[16] C.-L. Fang, D. Liang, F. Lin, and C.-C. Lin, "Fault tolerant web services," *Journal of Systems Architecture*, vol. 53, no. 1, pp. 21–38, 2007.

[17] P. P. W. Chan, M. R. Lyu, and M. Malek, "Reliable web services: methodology, experiment and modeling," in *Proceedings of the IEEE International Conference on Web Services (ICWS '07)*, pp. 679–686, July 2007.

[18] S. Subramanian, P. Thiran, N. C. Narendra, G. K. Mostefaoui, and Z. Maamar, "On the enhancement of BPEL engines for self-healing composite Web services," in *Proceedings of the International Symposium on Applications and the Internet (SAINT '08)*, pp. 33–39, August 2008.

[19] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," in *Proceedings of the IEEE 8th International Conference on Web Services (ICWS '10)*, pp. 83–90, July 2010.

[20] B. Li, X. Fan, Y. Zhou, and Z. Su, "Evaluating the reliability of web services based on BPEL code structure analysis and run-time information capture," in *Proceedings of the 17th Asia Pacific Software Engineering Conference: Software for Improving Quality of Life (APSEC '10)*, pp. 206–215, December 2010.

[21] K. H. Kim and H. O. Welch, "Distributed execution of recovery blocks: an approach for uniform treatment of hardware and software faults in real-time applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 626–636, 1989.

[22] R. K. Scott, J. W. Gault, and D. F. McAllister, "Fault-tolerant software reliability modeling," *IEEE Transactions on Software Engineering*, vol. 13, no. 5, pp. 582–592, 1987.

[23] *Business Process Model and Notation (Bpmn)*, 2011.

[24] E. Nelson, "Estimating software reliability from test data," *Microelectronics Reliability*, vol. 17, no. 1, pp. 67–73, 1978.

[25] W. T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul, and N. Liao, "A software reliability model for web services," in *Proceedings of the 8th IASTED International Conference on Software Engineering and Applications*, pp. 144–149, November 2004.

[26] V. Cortellessa and V. Grassi, "Reliability modeling and analysis of service-oriented architectures," in *In Test and Analysis of Web Services*, pp. 339–362, Springer, 2007.

[27] *Web Services Business Process Execution Language Version 2.0*, 2007.

[28] J. Musa, "Operational profiles in software-reliability engineering," *IEEE Software*, vol. 10, no. 2, pp. 14–32, 1993.

[29] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer, *Applied SOA: Service-Oriented Architecture and Design Strategies*, Wiley, 2008.

[30] D. Gross, *Fundamentals of Queueing Theory*, John Wiley & Sons, Hoboken, NJ, USA, 3rd edition, 2008.

[31] A. Saltelli, M. Ratto, T. Andres et al., *Global Sensitivity Analysis. The Primer*, John Wiley & Sons, Chichester, UK, 2008.