Reliability Analysis of On-Demand Service-Based Software Systems Considering Failure Dependencies

Kuan-Li Peng and Chin-Yu Huang

Abstract—Service-based software systems (SBSSs) are widely deployed due to the growing trend of distributed computing and cloud computing. It is important to ensure high quality of an SBSS, especially in a strongly competitive market. Existing works on SBSS reliability usually assumed independence of service failures. However, the fact that resource sharing exists in different levels of SBSS operations invalidates this assumption. Ignorance of failure dependencies have been discussed as potentially affecting system reliability predictions and lowering the benefits of design diversity, as typically seen in high-reliability systems. In this paper, we propose a reliability framework that incorporates failure dependence modeling, system reliability modeling, as well as reliability analysis for individual services and for failure sources. The framework is also capable of analyzing the internal structures of popular software fault tolerant (FT) schemes. The proposed method is applied to a travel agency system based upon a real-world practice for verifying its accuracy of reliability modeling and effectiveness of varied reliability measures. The results show that failure dependence of the services is an essential factor for analyzing any valuable SBSS system. Further, a set of reliability measures with different capabilities and complexities are available for assisting SBSS engineers with system improvements.

Index Terms—Service-based software systems, service correlation, common cause failures, reliability modeling, reliability analysis, software fault tolerance

1 Introduction

Service-based software systems (SBSSs) [1] have become a major methodology for facilitating agile development of large systems and have served as a backbone of cloud computing. An SBSS is built up from a number of reusable services, which package basic functionality and are accessible through the network with standard protocols. With wide applications of SBSS, it is important to consider not only the functionality but also the quality of services (QoS), especially in a highly competitive information service market. In fact, reliability has been identified in the standard of Open Group Service Integration Maturity Model (OSIMM) [2] as a key factor in the application dimension, and the reliability of the organization's business-critical applications should be ensured for reaching the maturity level of virtualized services (level 6).

However, the current approaches do not guarantee non-functional requirements of a Web service [3]. Since it is difficult for humans to build fault-free systems, software fault tolerance (FT) [4], which utilizes multiple services (called replicas) to implement the same task, is a suitable approach to high reliability, especially for SBSSs [5]. Transaction modeling and verification of Web service systems are also proposed [6], [7] to ensure the alignment with the behavioral specifications, where failures do not corrupt the business transactions.

It is noted that uncertainties of the SBSS environments such as server shut-downs, service changes, and network problems introduce new challenges to the operation of such systems [8], [9], making existing analysis approaches for component-based software systems (CBSSs) less applicable. In addition, failure correlations (or failure dependencies[1]) complicate the situation. It is not unusual for services to share the same providers (such as Amazon, Google, and Yahoo), networks, databases, or external libraries. Common-cause failures (CCFs) may occur when there are problems in these shared resources [10]. In addition to common failure sources, erroneous messages propagated between services as well as failures of services in load-balanced environments could also affect the operation of other services. It is reported that ignoring failure dependencies may cause pessimistic reliability prediction for series structures and optimistic prediction for parallel structures [11]. Unfortunately, many redundant designs to improve performance or reliabilities are intrinsically parallel-structured, and the resulting optimistic reliability predictions are usually undesired by system engineers [12], [13].

Towards quality assurance of service-based software systems, a number of reliability assessment methods have been proposed, but most of the existing methods have simply assumed independent failures. We consider the following travel agency system based on a real-world industrial practice [14], where Fig. 1 presents the basic work-flow in the BPMN diagram [15] including notations for implementation

• *The authors are with the Department of Computer Science, National Tsinghua University, Hsinchu, Taiwan. E-mail: geoybest@hotmail.com, cyhuang@cs.nthu.edu.tw.*

Manuscript received 9 Sept. 2014; revised 8 Aug. 2015; accepted 19 Aug. 2015. Date of publication 27 Aug. 2015; date of current version 9 June 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2473843

1. Failure dependence implies causality, while failure correlation is related to statistical dependence. These two terms are sometimes used interchangeably in this paper.

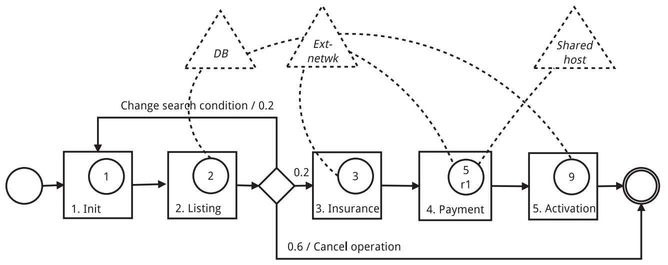

Fig. 1. Scenario-base with service dependencies and branch probabilities.

TABLE 1
Service Reliabilities

Service	Condition	Rel.	Service	Condition	Rel.
Srv. 1		.97	Srv. 2	DB	.97
Srv. 3	$ENet$	.97		$\overline{DB}$	.10
	$\overline{ENet}$	.40	Srv. 9	DB	.97
Srv. 5	$ENet, SHost$	.97		$\overline{DB}$	.10
	$ENet, \overline{SHost}$	.10	DB		.95
	$\overline{ENet}, SHost$	.40	$ENet$		.95
	$\overline{ENet}, \overline{SHost}$	.00	$SHost$		.95

services (circles with service numbers) and service correlations (triangles with correlation names). Further, Table 1 displays the reliabilities of each service in various conditions. Dependencies exist between services so that the independent-failure assumption does not hold true. For example, the listing and the payment services both rely on some internal database (DB). These services work with a reliability of 0.97 when the DB is functioning, but the reliability significantly drops to 0.1 when the DB fails. Correlations also exist between the insurance and the payment services since they are accessed through external networks. Another type of dependency due to utilizing a common service host will be discovered later.

Each time the system receives a request, it invokes service 1 to create a request record and then retrieves the products for the customer with service 2. Given the product list, customers may select among the following operations: 1) to modify their preferences and loop back to service 1 with probability 0.2, 2) to branch out to cancel the operation directly with probability 0.2, or 3) to proceed to order insurance, to make payment, and finally to activate the request record with services 3-9. In this study, we will consider the dependence of failures while making the reliability predictions of serviced-based systems. We will discuss the operations of fault tolerance technologies in the presence of service correlations. The way an individual service reliability or service correlation affects the overall SBSS reliability will also be studied.

To explore the above aspects, this paper is organized as follows. Section 2 reviews the related research. Section 3 proposes a reliability modeling framework for SBSS considering the service correlation and fault tolerant mechanisms, along with an example explaining the framework. In Section 4, we describe a number of sensitivity analysis methods for individual services and also for the service dependencies. Section 5 presents extensive results and analysis for the proposed methods, followed by the discussions of threats to validity in Section 6. Finally, Section 7 summarizes the research.

2 Previous Work

Systematic studies of failure dependencies in software systems might be traced back to Eckhardt and Lee's (E&L's) work [16]. They related input variation and program variation with outputs of each component, and they derived an intensity function to describe the probabilities where multiple components fail together. Later, Littlewood and Miller [17] generalized E&L's model by considering the diversity of software development processes. Tomek et al. [18] also extended E&L's model to analyze the pair-wise correlation between software modules in the context of recovery blocks (RcB). These intensity-based methodologies have explored the properties of multi-versioning "on the average" rather than analyzed a certain selection of program versions.

Some research focused on error propagation between software components. Popic et al. [19] presented a Bayesian reliability model that analyzed the probabilities of observing erroneous states of a component due to message transmissions. Fiondella and Gokhale [20] proposed a Markov model that considered not only error propagation during transmission but also error recoveries (self-healing) of the components.

In addition, Dai et al. [21] analyzed the reliability of dependent n-version programming (NVP) by considering all possible combinations of components and deriving the reliability with fault-tree analysis. Jain and Gupta [22] also analyzed NVP systems with common mode failures with universal generating functions. Zhou et al. [10] extended the traditional path-based approaches to executions of composite Web services. Kristiansen et al. [23] also gave an upper bound for the probabilities of simultaneous component failures through Bayesian hypothesis testing.

Although contemporary studies enable us to analyze failure dependencies in various aspects, it is noted that most methods would suffer from low scalability since an exponential number of parameters are required, making them computation-intensive [11]. More efficient methods are still desired.

On denoting the reliability and dependence factors, fault trees, Bayesian networks, event trees, and reliability block diagrams are commonly used for hardware systems [24]. Recent works extended these notations to analyze CCFs for such systems [25], [26]. For software systems, Scott et al. [27] used event trees to analyze the reliabilities of several FT strategies with joint error events, while Dai et al. [21] used fault trees to aggregate each combination of joint error events. In the work by Vieria and Richardson [28], partial-order multi-sets (POSMSETs) semantics were used to denote the functional dependencies between software components.

As discussed earlier, SBSS failures are quite different in nature from traditional hardware systems or component-based software systems. Rather than extending the classical models, Zhou et al. [10] proposed a hierarchical reliability tree model tailored for SBSS. They identified the failures for Web services in four levels (service, host, provider, and network) and assumed that errors in an upper level are the direct causes of CCFs for all the child nodes.

In the following, we will propose a reliability synthesis and analysis model for SBSS. The proposed model has very

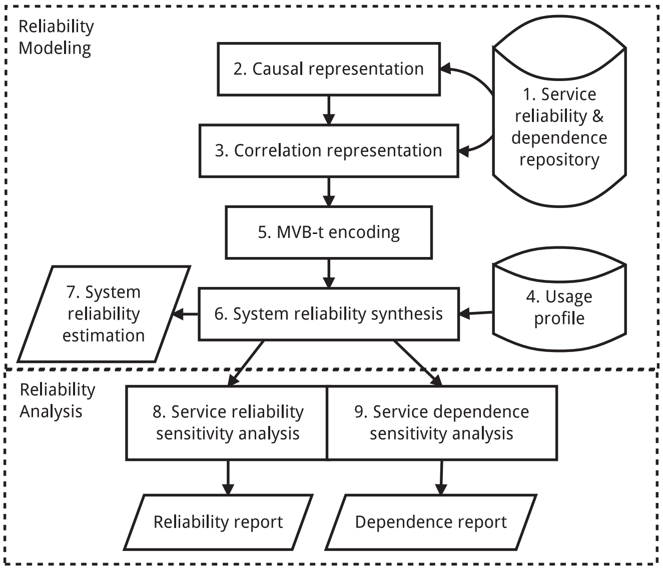
Reliability Modeling
2. Causal representation
1. Service reliability & dependence repository
3. Correlation representation
5. MVB-t encoding
7. System reliability estimation
6. System reliability synthesis
4. Usage profile
Reliability Analysis
8. Service reliability sensitivity analysis
9. Service dependence sensitivity analysis
Reliability report
Dependence report

Fig. 2. SBSS reliability framework diagram.

high flexibility since it considers the potential failure sources and correlations either explicitly in early stages or implicitly at later stages.[2] Note that the failure conditions will not be restricted to the single-parent assumption as in the tree-based counterparts. Key attributes of SBSS, such as the service reliabilities, transmission reliabilities, system structure, as well as FT internal mechanisms, are also embedded into our Markov-based model. The encoding scheme has high correlation analysis efficiency, making it suitable for further sensitivity analysis.

3 SBSS Reliability Modeling

To evaluate the system reliability, an SBSS reliability framework is presented in this section. For simplicity, the assumptions are listed below:

1) The reliabilities of services with common dependencies are positively correlated. That is, failures of a service increase the probabilities of failures of other correlated services [10], [11], [23], [29].
2) Services are used on-demand rather than long-running, and the state (success or failure) of each constituent service does not vary during a system execution [5], [9], [10], [23], [30].

Fig. 2 illustrates the framework of this study. Each component is briefly explained as follows:

1) *Service reliability and dependence repository* is a database for the reliability of each service and reliability dependence between services. Such information may be collected from middleware [5].
2) *Causal representation* (see Section 3.1) expresses the dependence between services by conditional reliabilities in the form of Bayesian networks. This is the preferred way for dependence representation since it is more flexible and enables more reliable analysis techniques.
3) *Correlation representation* (see Section 3.1) expresses the dependence between services by their marginal reliabilities and the correlation matrix. It can be converted from causal representation by belief propagation [31] or be provided directly from the service reliability and dependence repository.
4) *Usage profile* is a database for the system usage pattern, which contains the system structure and the transition probabilities of each service for each user.
5) *MVB-t encoding* (see Section 3.2) converts the service reliability and dependence to a set of multivariate Bernoulli (MVB) variables for further analysis.
6) *System reliability synthesis* (see Section 3.3) estimates the overall SBSS reliability by combining the MVB variable and the system reliability modeling method which considers system structure and fault tolerance.
7) *System reliability estimate* is the result of system reliability synthesis.
8) *Service reliability sensitivity analysis* (see Section 4.1) provides various metrics to quantify the impacts of individual service reliabilities on the system.
9) *Service dependence sensitivity analysis* (see Section 4.2) provides various metrics to quantify the impacts of service failure correlations on the system.

3.1 Representations of Service Reliability and Dependence

Service dependencies are initially expressed in causal form. The state of each service is modeled by a binary random variable X, and the values $1/0$ indicate operational and failure states, respectively. In causal form (such as in Table 1), the s-expected reliability of service i is conditioned by the states of its dependent sources DL_i, defined by

$$DL_i = \{dl_{i,j} : 1 \le j \le n\}, \tag{1}$$

where $dl_{i,j}$ is a binary random variable for the state of the jth dependent source.

Service dependencies are then converted to correlation form, which consists of a marginal reliability vector R and a correlation matrix C. The ith entry of R is the marginal reliability of service i, derived from

$$r_i = \Pr[X_i = 1] = \sum_{Y \in \{DL_i\}} \Pr[X_i = 1 \mid Y] \cdot \Pr[Y], \tag{2}$$

where $\{DL_i\}$ is the set of all possible joint conditions of the dependent sources. When the conditional reliability on a certain Y is not immediately available, one may use belief propagation [31] to derive it. For simplicity, we only consider the cases of acyclic dependencies [10] in this study.

The states of a service can be viewed as a sequence of Bernoulli random variables [11], [12], [32] whose probability of success varies for dependent sources. The correlation of the states between services i and j is formulated by

$$c_{i,j} = \frac{\Pr\big[X_i = 1,\, X_j = 1\big] - r_i r_j}{\sqrt{r_i(1 - r_i) r_j \big(1 - r_j\big)}} \tag{3}$$

as the entry (i, j) of the correlation matrix C. $\Pr[X_i = 1,\, X_j = 1]$ is also denoted by $r_{i,j}$ in the following sections.

2. Identifying and modeling all correlations formally may be infeasible in some cases.

3.2 MVB-t Encoding

When the service marginal reliabilities R and correlation C are ready, they are MVB-t-encoded to produce s-independent Poisson random variables (RVs). The outputs consist of a Poisson lambda vector $\Lambda = [\lambda_i]$ and a membership vector $O = [o_i]$. The MVB-t encoding technique presented here originates from Fiondella et al. [11] but with modifications to handle the problems in practice.

The basic idea is to view the reliability of service i as an aggregated Poisson RV Y^i_{aggr} with mean

$$\lambda^i_{aggr} = \sum_{j=1}^{m} \lambda_j \big[i \in o_j\big],$$

where o_j is a set of all associated services for the jth Poisson RV. An invocation of service i is reliable iff the aggregated Poisson RV experiences zero events. In this case,

$$r_i = \Pr[Y^i_{aggr} = 0] = e^{-\lambda^i_{aggr}}. \tag{4}$$

Thus, we have

$$\lambda^i_{aggr} = \ln(1/r_i). \tag{5}$$

The correlation between services i and j can also be viewed as aggregated Poisson RVs $Y^{i,j}_{aggr}$, Y^{i-j}_{aggr}, and Y^{j-i}_{aggr} with the means

$$\lambda^{i,j}_{aggr} = \sum_{k=1}^{m} \lambda_j [\{i, j\} \subseteq o_k],$$

$$\lambda^{i-j}_{aggr} = \sum_{k=1}^{m} \lambda_i [i \in o_k \wedge j \notin o_k] = \lambda^i_{aggr} - \lambda^{i,j}_{aggr},$$

and

$$\lambda^{j-i}_{aggr} = \sum_{k=1}^{m} \lambda_j [j \in o_k \wedge i \notin o_k] = \lambda^j_{aggr} - \lambda^{i,j}_{aggr}.$$

Then,

$$\begin{aligned}\Pr[X_i = 1,\ X_j = 1] &= \Pr[Y^{i-j}_{aggr} = 0,\ Y^{j-i}_{aggr} = 0,\ Y^{i,j}_{aggr} = 0] \\ &= e^{-(\lambda^i_{aggr} + \lambda^j_{aggr} + \lambda^{i,j}_{aggr})} \\ &= r_i r_j e^{\lambda^{i,j}_{aggr}}.\end{aligned} \tag{6}$$

By combining (3) and (6), we have

$$\lambda^{i,j}_{aggr} = \ln\left(1 + c_{i,j}\sqrt{\frac{(1 - r_i)(1 - r_j)}{r_i r_j}}\right). \tag{7}$$

It could be found that (5) is a special case of (7), and $\lambda^i_{aggr} = \lambda^{i,i}_{aggr} = \ln(1/r_i)$.

Fig. 3 lists the main encoding algorithm *mvbenc*, which accepts the inputs of service reliabilities R, service correlations C, and an extra tolerance threshold tol. It eventually returns the lambda values Λ, the associated services O as well as the zero event probabilities Z of the encoded RVs. Note that inputs R and C may not be perfectly encodable by the original method [11] when they are obtained statistically from sources such as log files rather than generated analytically. The problem is caused by the effects of uncertainties that could result in underestimation of $\lambda^{i,i}_{aggr}$ and overestimation of $\lambda^{i,j}_{aggr}$. Different from the original version, *mvbenc* uses *tol* to control the error tolerance level, which serves as the maximum tolerable $\lambda^{i,j}_{aggr}$ whenever a constraint violation

$$\lambda^{i,j}_{aggr} > \min(\lambda^{i,i}_{aggr}, \lambda^{j,j}_{aggr}) \tag{8}$$

occurs during the encoding process.

```
1:  procedure MVBENC($R, C, tol$)
2:                                          ▷ 1. lambda conversion
3:      $\Lambda_{aggr} \leftarrow [\,]$
4:      for all $1 \le i \le j \le |R|$ do
5:          $\Lambda^{i,j}_{aggr} \leftarrow \ln(1 + C_{i,j}\sqrt{\frac{(1-R_i)(1-R_j)}{R_i R_j}})$
6:      end for
7:                                          ▷ 2. MVB-t encoding loop
8:      $k \leftarrow 0$, $\Lambda \leftarrow [\,]$, $O \leftarrow [\,]$, $Z \leftarrow [\,]$
9:      while $\Lambda_{aggr}$ has nonzero entries do
10:         $k \leftarrow k + 1$
11:         $\Lambda_k \leftarrow \texttt{nonzero\_min}(\Lambda_{aggr})$
12:         $i, j \leftarrow \arg\texttt{nonzero\_min}(\Lambda_{aggr})$
13:         $Z_k \leftarrow e^{-\Lambda_k}$
14:         if $\min(\Lambda^{i,i}_{aggr}, \Lambda^{j,j}_{aggr}) < \Lambda_k$ then
15:             if $\Lambda_k <= tol$ then              ▷ tolerance test
16:                 $\Lambda^{i,j}_{aggr} \leftarrow 0$, $k \leftarrow k - 1$
17:                 continue
18:             else
19:                 exception "Encoding Error!"
20:             end if
21:         end if
22:         $O_k \leftarrow \{i, j\}$                  ▷ associated-service set
23:         for all $1 \le t \le |R| \wedge t \notin O_k$ do
24:             if $\forall s \in O_k$, $\Lambda^{\min(t,s),\max(t,s)}_{aggr} \ge \Lambda_k$ then
25:                 $O_k \leftarrow O_k \cup \{t\}$           ▷ expand $O_k$
26:             end if
27:         end for
28:         for all $x, y \in O_k \wedge x \le y$ do
29:             $\Lambda^{x,y}_{aggr} \leftarrow \Lambda^{x,y}_{aggr} - \Lambda_k$       ▷ update $\Lambda_{aggr}$
30:         end for
31:     end while
32:     return $\Lambda$, $O$, $Z$
33: end procedure
```

Fig. 3. MVB-t encoding algorithm.

In the beginning of *mvbenc*, a matrix Λ_{aggr} containing the means of the aggregated Poisson RVs is generated by applying (7) in lines 2-6. Lines 7-31 are the main encoding loop, which terminates successfully when all the entries in the aggregation matrix Λ_{aggr} become zero. Lines 11-13 compute the next (kth) encoding RV, whose mean is the minimal non-zero value of the matrix. The zero-event probability is also computed by applying (4). Then, line 14 checks the potential conflicts of encoding according to (8). When a conflict occurs, *mvbenc* manages to reset the problematic entry and restart the encoding process when its value is within the tolerance, or otherwise it terminates the process exceptionally. After the kth encoding RV is ready, lines 22-27

```
1: procedure CALCJOINTPROB($S, O, Z$)
2:     $failset \leftarrow \{i : S_i = 0\}$
3:     $rv\_rel \leftarrow \{i : fail\_set \not\supseteq O_i\}$
4:     $rv\_unknown \leftarrow \{i : 1 \le i \le |Z| \wedge i \notin rv\_rel\}$
5:     $p \leftarrow \prod_{i \in rv\_rel} Z_i$                    ▷ init joint prob
6:     return $p *$ cjp($rv\_unknown, \emptyset, failset, O, Z$)
7: end procedure
```

Fig. 4. Joint probability algorithm.

determine the set of associated services O_k, where the aggregated Poisson RVs between each pair of the set members can be encoded by this encoding RV. Finally, the aggregation matrix Λ_{aggr} is updated in lines 28-30.

3.3 System Reliability Synthesis

After the MVB-t encoding, the overall reliability of an SBSS can be calculated from the joint probability and conditional system reliability for each possible service state, formulated by

$$r_S = \sum_{S \in \mathrm{BIN}(|R|)} \Pr\left[\bigcap_{i=1}^{|R|} X_i = S_i\right] \cdot \Pr\left[SBSS \middle| \bigcap_{i=1}^{|R|} X_i = S_i\right]. \quad (9)$$

In (9), S_i denotes the concerned state of service i, whose values can be either 1 (successful invocation) or 0 (unsuccessful invocation), and $\mathrm{BIN}(|R|)$ denotes the state space of all the services within in the SBSS.

To compute the conditional system reliability for a certain set of service states $\Pr[SBSS| \bigcap_{i=1}^{|R|} X_i = S_i]$, one should take into account the system structure and user behavior. This can be obtained from our formerly proposed service-based system reliability model [30] by substituting each input of service reliability $r_i^{'}$ for the corresponding binary outcome of service state X_i. The original model [30] is capable of analyzing various system structures such as sequences and branching and well-known fault tolerant designs including recovery block [4] and n-version programming [4]. Therefore, the proposed SBSS reliability model naturally inherits these capabilities.

The joint probability for a certain set of service states $\Pr[\bigcap_{i=1}^{|R|} X_i = S_i]$ can be calculated from the joint zero-event probabilities of all matching outcome sets of MVB-t-encoded RVs. Since the encoded RVs are s-independent, the joint zero-event probability of an outcome set T can be calculated independently. Define a new binary RV Z_i that evaluates to 1 iff the corresponding encoded RV Y_i equals zero. Then,

$$\begin{aligned}\Pr[\bigcap_{i=1}^{|\Lambda|} Z_i = T_i] &= \prod_{i=1}^{|\Lambda|} \Pr[Z_i = T_i] \\ &= \prod_{i=1}^{|\Lambda|} \left(e^{-\lambda_i}\right)^{T_i} \left(1 - e^{-\lambda_i}\right)^{1-T_i}.\end{aligned} \quad (10)$$

The final stretch of evaluating the SBSS reliability involves matching the encoded RV configurations (T_i) in (10) to the service states (S_i) in (9). A naive exhaustive-search implementation requires $2^{|R|+|\Lambda|}$ times the joint RV outcome probability computation, which is undesirable for a large SBSS. The algorithms *calcJointProb* and *cjp* presented

```
1: procedure CJP($rv\_curr, fs\_curr, fs, O, Z$)
2:     if $fs\_curr = fs$ then                    ▷ satisfy fail-set
3:         return 1.0
4:     else if $|rv\_curr| = 0$ then              ▷ not satisfy fail-set
5:         return 0
6:     end if
7:     $rv \leftarrow$ pop_one($rv\_curr$)
8:     $p1 \leftarrow Z_{rv} *$ cjp($rv\_curr, fs\_curr, fs, O, Z$)  ▷ zero events
9:     $p2 \leftarrow (1 - Z_{rv}) *$ cjp($rv\_curr, fs\_curr \cup O_{rv}, fs, O, Z$)   ▷ non-zero events
10:    return $p1 + p2$
11: end procedure
```

Fig. 5. Internal joint probability algorithm.

in Figs. 4 and 5 use a *pruning* technique and the concept of *fail-set* to lower the computation efforts.

The *calcJointProb* procedure is given the service outcomes S, the ownership matrix O, and the zero-event probabilities Z for each MVB-t encoded RV. The fail-set is the collection of the failure services (line 2). If an encoded RV is not associated with any member within the fail-set (line 3), it is then constantly considered reliable[3] and exempt from the subsequent searches. Otherwise, its exact outcome still depends upon other RVs (line 4) and will be decided later by *cjp* (line 6).

The *cjp* procedure implements exact searching. It is given the remaining RVs under searches *rv_curr*, the current fail-set *fs_curr*, the original fail-set *fs*, and other parameters passed by *calcJointProb*. During searching, it selects an RV from *rv_curr* and computes the joint probability for each different outcome of the selected RV (lines 7-10). A search terminates successfully when the current RV settings satisfy the fail-set (lines 2-3) or unsuccessfully when there are no more RVs to search yet the fail-set has not been satisfied (lines 4-5).

Finally, Fig. 6 presents the *calcSysRel* procedure for system reliability synthesis. It accepts the arguments *arch* for the SBSS specification, *profile* for the usage profile, *service* for the set of services, and others that have been explained. Computation efforts are reduced further by calling *calcJointProb* only when the corresponding conditional probability is non-zero (lines 6-9). The algorithm also involves generating the reliability matrix (line 4) and deriving the SBSS reliability (line 5). These processes are explained in our previous study [30] and will be demonstrated in the following section.

3.4 An Example: Travel Agency System

This section illustrates the process of deriving the SBSS reliability for the travel agency system in Fig. 1. The reliability specification in causal form is in Table 1.

We first consider services 2 and 9, which are dependent upon a DB. Conditioning on the states of this DB, the marginal reliabilities are evaluated as $r_2 = r_9 = .97 \times .95 + .10 \times .05 = .9265$, and the probability that both the services are operational is $r_{2,9} = .97 \times .97 \times .95 + .10 \times .10 \times .05 \approx .8944$. The correlation between the service states is evaluated as $c_{2,9} = \left(r_{2,9} - r_2 r_9\right) / \sqrt{r_2(1 - r_2) r_9 (1 - r_9)} \approx .5280$. Likewise,

3. It encounters zero events.

```
procedure CALCSYSREL($arch, profile, service, O, Z$)
    $r \leftarrow 0$                                    ▷ system reliability
    for all $S \in \{0,1\}^{|\text{service}|}$ do                ▷ outcome sets
        $M \leftarrow$ genRelMatrix($arch, profile, S$)
        $cp \leftarrow$ sysRel($M$)
        if $cp > 0$ then                          ▷ non-zero cond prob
            $jp \leftarrow$ calcJointProb($S, O, Z$)
            $r \leftarrow r + cp * jp$
        end if
    end for
    return $r$
end procedure
```

Fig. 6. SBSS reliability algorithm.

by applying (2) and (3) repeatedly, we get the marginal reliability vector

$$R = [\,.97 \quad .9265 \quad .9415 \quad .8992 \quad .9265\,]^T$$

and the correlation matrix

$$C = \begin{bmatrix} 1 & & & & \\ & 1 & & & .5280 \\ & & 1 & .1287 & \\ & & & 1 & \\ & & & & 1 \end{bmatrix},$$

where only non-zero rounded values are presented. The rows and columns correspond to services 1, 2, 3, 5, and 9, in order.

Then, in the *mvbenc* algorithm (Fig. 3), the initial aggregated lambda matrix is obtained by applying (7)

$$\Lambda_{aggr}^{(0)} = \begin{bmatrix} .0305 & & & & \\ & .0763 & & & .0410 \\ & & .0603 & .0107 & \\ & & & .1063 & \\ & & & & .0763 \end{bmatrix}.$$

The first MVB-encoded RV has a mean of .0107, which is the minimal non-zero value in the above matrix. Since no other services besides 3 and 5 could be added in the first iteration, only entries (3,3), (3,5), and (5,5) are updated by subtracting this mean. The MVB-t encoding repeats the above process k' times until all the entries in $\Lambda_{aggr}^{(k')}$ are zero; the results are illustrated in Table 2. It may require 31 variables to model the correlation between five services, but only seven encoded RVs are used here.

To derive the system reliability (9), the conditional probabilities under various service states will be computed. We consider the case where all of the services except nine are functioning. That is, $S = [\,1 \quad 1 \quad 1 \quad 1 \quad 0\,]^T$. According to Figs. 4 and 5, $failset$ contains service {9}, and rv_rel contains encoded RVs $\{1, 2, 4, 6, 7\}$, which correspond to the rows in Table 2 without service 9 in the last column. To satisfy this fail-set, all encoded RVs except the fifth should be reliable, so the only matching configuration is $(1, 1, 1, 1, 0, 1, 1)$. In this case, the joint probability is evaluated as $\Pr[\bigcap_i X_i = S_i] = \left(1 - e^{-\lambda_5}\right) \cdot \prod_{1 \le k \le 7, k \ne 5} e^{-\lambda_k} \approx .0267,$ and the corresponding transition matrix by calling the *genRelMatrix* procedure [30] in Fig. 6 would be

TABLE 2
MVB-t Encoding Results

k	λ_k	o_k	k	λ_k	o_k
1	.0107	{3,5}	5	.0353	{9}
2	.0305	{1}	6	.0496	{3}
3	.0410	{2,9}	7	.0956	{5}
4	.0353	{2}			

$$M = \begin{bmatrix} & 1 & & & & \\ .2 & & .2 & & & .6 \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 0 \\ & & & & & \end{bmatrix},$$

where the last row and column represent the implicit terminating service, and the conditional system reliability $\Pr[SBSS| \bigcap_i X_i = S_i] = .750$. Finally, we get the SBSS reliability of .861. It is noted that a classical model which ignores service correlation would underestimate the system reliability as .850. More service correlation analysis will be presented in the following sections.

4 SBSS Reliability Analysis

When a method to evaluating the overall SBSS reliability is available, this section will combine it with various metrics to quantify the impacts of individual services in Section 4.1 and the impacts of service correlations in Section 4.2 to the overall system.

4.1 Sensitivity Analysis for Individual Services

The following metrics for analyzing each service are used here:

1) Failure-occurrence-based (FOB) measure,
2) Birnbaum's reliability importance (Birnbaum) measure, and
3) design of experiments / main effects (DOE/ME) measure.

4.1.1 FOB Measure

FOB measures the impact of a service by considering its error-proneness in terms of failure frequencies. Such measures can be evaluated efficiently from the error logs of service invocations. There are variants of FOB measures depending on the context of failure frequencies.

In *iFOB*, the data are collected separately from the system's execution. The *iFOB* measure for service i (I_i^{iFOB}) is estimated by the ratio of its failure counts over its total observations. When the observation for each service is made independently, I_i^{iFOB} is approximated by $1 - r_i$.

In *boFOB*, the data are collected during the system's execution. Such a measure for service i (I_i^{boFOB}) is estimated by the ratio of its failure counts over its total execution counts during the system's execution.

Finally, in *brFOB*, the data are collected during the system's execution. The measure for service i (I_i^{brFOB}) is estimated by the ratio of its failure counts over the total number of system executions. Different from *boFOB*, the measure of *brFOB* reflects not only the failure intensities but also the execution probabilities of the service under consideration.

4.1.2 Birnbaum Measure

Birnbaum's reliability importance measure was originally proposed by Birnbaum [33] and then extended by Kristiansen et al. [34] for multi-component software. Birnbaum measure for service *i* is defined as

$$I_i^B = \frac{\partial r_S}{\partial r_i}. \tag{11}$$

Since the state of service *i* is represented by a binary RV, the reliability of the system can be expanded by

$$\begin{aligned} r_S =& r_i r_{S[X_i=1]} + (1 - r_i) r_{S[X_i=0]} \\ =& r_i \left(r_{S[X_i=1]} - r_{S[X_i=0]} \right) + r_{S[X_i=0]}, \end{aligned} \tag{12}$$

where $[X_i = 0]$ and $[X_i = 1]$ specify the conditions of service i. Therefore, the Birnbaum measure for service i can be expressed by

$$I_i^B = r_{S[X_i=1]} - r_{S[X_i=0]}, \tag{13}$$

and is independent of r_i.

The form presented here (13) is still applicable when the closed form of system reliabilities are unavailable.

4.1.3 DOE/ME Measure

All the previous service analysis metrics discussed can be computed efficiently. FOB and the Birnbaum measures require only $O(|R|)$*calcSysRel* calls for all constituent services within an SBSS. However, each time the former considers a single property (failure occurrence) and the latter considers a property change (success or failure states) of a single service; both may be blind to uncertainties of properties (such as measurement inaccuracies and variances with time) or correlations between the services.

On the other hand, design of experiments [35] explores a wider parameter space by considering various settings of all variables (services). For example, Li et al. [36] used this technique to assist in the decision of software release time. Here we utilize two-level designs, where each service *i* in a configuration set *j* may be labeled either positive ($x_{ji} = 1$, meaning $\alpha\%$ increase of its observed reliability) or negative ($x_{ji} = -1$, meaning $\alpha\%$ decrease of its observed reliability), and the result y_j (the calculated system reliability) is collected during the experiments. The complete two-level full factorial (FF) design requires $2^{|R|}$ *calcSysRel* calls, which will become very costly as the system size grows. Therefore, fractional factorial experimental designs are sometimes used instead to reduce the computation efforts while some important features of the problem are studied. For example, two-level resolution III (Res-III) design requires $2^{\lceil \lg(|R|+1) \rceil}$ calls and ensures an equal amount of combinations for any two services.[4] Two-level resolution IV (Res-IV) design requires $2^{\lceil \lg(|R|) \rceil + 1}$ calls and ensures an equal amount of combinations for any three services.[5]

4. Res-III designs ensure an even distribution of (+,+), (+,-), (-,+), and (-,-) for any two variables.

5. Res-IV designs ensure an even distribution of (+,+,+), (+,+,-), (+,-,+), (+,-,-), (-,+,+), (-,+,-), (-,-,+), and (-,-,-) for any three variables.

When the results of the designed experiments Y are ready, the main effect (ME) for service i is obtained to measure the effect of that service, evaluated by

$$I_i^{ME}(Y) = \frac{1}{|Y|} \sum_{j=1}^{|Y|} x_{ji} y_j, \tag{14}$$

and may be considered a sensitivity measure based upon design of experiments (DOE). It can also be shown that the variance in the results explained by service *i* alone is related to its ME [35] by

$$\mathrm{Var}[FF_i] = [I_i^{ME}(FF)]^2. \tag{15}$$

4.2 Sensitivity Analysis for Service Correlations

Service correlations are studied systematically in the form of *dependence sets* (DSs). A DS is a collection of dependence descriptions that are considered in the model. For any SBSS, there is a full DS, denoted by DS, which is the collection of all known dependence factors in the system. Other DSs are constructed by excluding one or more dependence factors. A DS is denoted by DS_{-x} when exactly one dependence factor x is excluded from the full DS. A proper subset of DS is *non-trivial* when it includes only the dependence factors related to the dependent services in the system. The complement of the non-trivial DSs that perform most differently to DS under a certain measurement is called a *most significant dependence set* (MS-DS). On the contrary, the complement of the non-trivial DSs that perform most closely to DS is called a *least significant dependence set* (LS-DS).

The following metrics for analyzing service correlations are used for determining MS-DSs and LS-DSs for an SBSS [34]:

1) Direct calculation (DC),
2) The Birnbaum-based measure, and
3) design of experiments / principal component analysis (DOE/PCA) measure.

4.2.1 Direct Calculation

The direct calculation is the difference between the SBSSs actual reliability (by considering all known dependencies) and the system reliability where the examined dependence factor is ignored. A service dependence is probably of high importance when such a difference is significant. For dependence x, it is evaluated by

$$D_x^{DC} = r_{S[DS_{-x}]} - r_s, \tag{16}$$

where $[DS_{-x}]$ represents the situation that excludes dependence x. This measure requires two *calcSysRel* calls.

4.2.2 Birnbaum-Based Measure

The Birnbaum-based measure considers the amount of change the existence of a certain dependence factor causes to the Birnbaum value of each service within the SBSS. This measure relies on a basic assumption that the existence of a critical service dependence changes the importance measures of various services in the system significantly [34]. For dependence x, it is expressed by

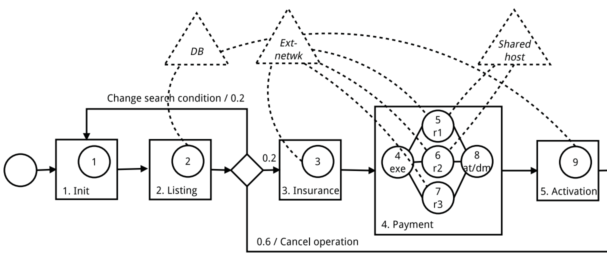

Fig. 7. Scenario-RcB/NVP.

$$D_x^B = \sqrt{\frac{\sum_{j=1}^{|R|}\left(I_j^B - I_{j[DS_{-x}]}^B\right)^2}{|R|}} \tag{17}$$

and requires $4 \times |R|$ *calcSysRel* calls.

4.2.3 DOE/PCA Measure

DOE is applied here by considering dependence sets as variables and the calculated system reliabilities as results in the experiments. The experimental results then go through principal component analysis [37]. PCA is a popular unsupervised multivariate analysis method, and it reveals the internal structure of a set of variables through linear combinations of the principal components sorted by their contributions to maximize the variance of data [31], [38]. PCA enables visual analysis of data through *score plots* and *loading plots*. Score plots represent the variation in data by projecting data to the space span by the first two principal components (PC1 and PC2); loading plots represent the contributions of the original variables to describe such variation. Kristiansen et al. [34] suggest that we focus on the variables that fall close to the variable of full DS on the loading plots, since the most important service dependencies may explain the variation of data in similar manners. In this study, we will also examine this idea through extensive experiments and analysis for service-based systems.

5 Experiments and Discussion

In addition to the basic Travel Agency System that is discussed in previous sections (scenario-base), an extended version with fault tolerance [4] mechanisms (Fig. 7) is also considered in the experiments. In the extended work-flow, the payment task is implemented with three functionally equivalent services 5-7 (also known as *replicas*), and they work either sequentially in the recovery block design or in parallel in the n-version programming design. We will refer to these two enhanced systems as scenario-RcB and scenario-NVP, respectively. More correlations are introduced, as the replica services are all dependent upon external networks, and we study a case of correlated replicas, where services 5 and 6 share a Web host. The scenarios here incorporate failure correlations in host, network, and database levels, which are common in real-world applications.

TABLE 3
FT Service Reliabilities

Service	Condition	Rel.	Service	Condition	Rel.
Srv. 4		.97	Srv. 6	$ENet, SHost$	.88
Srv. 7	$ENet$	.85		$ENet, \overline{SHost}$	.10
	$\overline{ENet}$	.30		$\overline{ENet}, SHost$	.40
Srv. 8		.97		$\overline{ENet}, \overline{SHost}$	.00

TABLE 4
Dependence Sets

DS#	Description
0	Ignore all dependencies.
1	Preserve full dependencies.
2	Preserve all dependencies except DB.
3	Preserve all dependencies except ENet.
4	Preserve all dependencies except SHost.
5-13	Preserve all dependencies except services 1-9.

Scenario-RcB and scenario-NVP share the same reliability settings with scenario-base, and the additional settings for the FT services are listed in Table 3. Table 4 shows the DSs that will be studied in Section 5.3. In a non-full DS, when dependence related to a service is excluded, the service reliability is determined by transforming the conditional reliabilities into unconditional ones. Taking DS 6 as an example, the reliability of service 2 would be $r_2 = \Pr[X_2 = 1] = \sum_{x\in\{0,1\}} \Pr[X_2 = 1|X_{DB} = x]\cdot \Pr[X_{DB} = x] = 0.9265$.

In the remainder of this section, we will first compare the model results with the simulation results to validate the proposed SBSS reliability model in Section 5.1. Then, Section 5.2 will present the sensitivity analysis of individual services, followed by the sensitivity analysis of various service dependencies in Section 5.3.

5.1 Model versus Simulation Results

Extensive simulations are performed to validate the proposed SBSS reliability model. The proposed SBSS reliability synthesis and analysis methods in Sections 3-4 are implemented with Python, and the experiments are executed on an Intel core i3 machine with four 2.3 GHz cores and 4 GB of memory, running a Linux 3.8 kernel.

For each scenario, zero dependencies (dep0) and full dependencies (dep1) are selected as experimental subjects. In simulation, we first perform a topological sort on the services according to their dependence relations. The service states for each system invocation are then sampled in this order and follow the reliability settings in Tables 1 and 3. Each subject is tested by 10 rounds of simulations, and each round contains 100 K calls of the dedicated SBSS. Ten model computations are made for each subject. Repeated model computations are needed because the correlation representation is not derived analytically but statistically from the causal representation, which is an alternative approach without the Bayesian propagation logic. As shown in Table 5, the average values from the proposed model and those from the simulations are very close in scenarios with various FT and dependence settings; the standard deviations are below 0.0014, and the relative errors are within $\pm 0.25\%$, showing that the proposed method is workable to synthesize SBSS reliabilities.

5.2 Sensitivity Analysis for Individual Services

Based upon the proposed SBSS reliability model, the seven metrics introduced in Section 4.1 are measured to quantify

TABLE 5
Model versus Simulation

Scenario	Model Mean	Simulation Mean	Simulation Standard Deviation	Relative Error
Base-dep0	0.850	0.850	0.001	0.02%
Base-dep1	0.861	0.861	0.001	−0.02%
RcB-dep0	0.858	0.858	0.001	0.02%
RcB-dep1	0.862	0.865	0.001	−0.24%
NVP-dep0	0.848	0.848	0.001	0.00%
NVP-dep1	0.850	0.851	0.001	−0.05%

the importance of each service to the overall system. The error tolerance level *tol* in MVB-t encoding ranges from 0 to 0.026, depending on the encoding process. The reliability increase ratio in DOE is 10 percent. Data normalization is made by dividing the raw value of each service by the largest one among all services; the final results for scenario-base, scenario-RcB, and scenario-NVP are compiled in Tables 6-8, respectively. The metrics are aligned in columns, while the services are aligned in rows. We only analyze the cases of dep0 and dep1 as in Section 5.1, and highlight the services with the largest (solid triangles) and the lowest (hollow triangles) values of each metric.

It can be seen that the FOB-family measures of the replica services (services 5-7) are the greatest among all the services (except that they fall behind service 1 in the brFOB measures), since the replicas have the highest failure rates. On the contrary, the Birnbaum measures and the DOE/ME-family measures of services 1 and 2 are the greatest; this is reasonable because they are on the trunk of the execution paths. For all FT scenarios (Tables 7 and 8), the Birnbaum and the DOE/ME-family measures of the replica services drop to the lowest among all the services, reflecting the fault masking capabilities of the FT designs.

While service 9 is only on a branch of the execution paths, the raw values of I_9^B under dep1 cases increase significantly (209-303 percent) when compared with their counterparts under dep0, placing it third among all the services. For the dep1 cases where correlation exists between services 9 and 2, it can be derived by Bayes propagation that $\Pr[X_9 = 1] \approx 0.93$, $\Pr[X_9 = 0] \approx 0.07$, $\Pr[X_2 = 1|X_9 = 1] \approx 0.97$, and $\Pr[X_2 = 1|X_9 = 0] \approx 0.44$; the last two imply a strong positive correlation. Since the whole SBSS is very sensitive to service 2 and the Birnbaum definition treats the cases of $X_9 = 1$ and $X_9 = 0$ equally, regardless of the fact that $\Pr[X_9 = 1] \gg \Pr[X_9 = 0]$, I_9^B is lifted up with the help of service 2 thereafter. We argue that such growth in

TABLE 6
Service Reliability Analysis (Scenario-Base)

DS	Service\Metric	iFOB	boFOB	brFOB	Birnbaum	FF/ME	Res3/ME	Res4/ME
dep0	1: init	△0.027	△0.317	0.427	0.956	0.749	0.757	0.749
	2: listing	0.033	0.767	▲1.000	▲1.000	▲1.000	▲1.000	▲1.000
	3: insurance	0.030	0.604	△0.181	△0.204	△0.185	0.249	△0.185
	5: payment	▲1.000	▲1.000	0.282	0.214	0.224	0.219	0.226
	9: activation	0.713	0.742	0.189	0.207	0.203	△0.202	0.203
dep1	1: init	△0.303	△0.361	0.426	0.955	0.998	0.999	▲1.000
	2: listing	0.733	0.874	▲1.000	▲1.000	▲1.000	▲1.000	0.999
	3: insurance	0.578	0.677	0.181	△0.213	△0.147	0.211	△0.149
	5: payment	▲1.000	▲1.000	0.252	0.223	0.198	0.196	0.198
	9: activation	0.735	0.427	△0.098	0.624	0.196	△0.195	0.196

Legend: ▲ Most sensitive services, △ Least sensitive services.

TABLE 7
Service Reliability Analysis (Scenario-RcB)

DS	Service\Metric	iFOB	boFOB	brFOB	Birnbaum	FF/ME	Res3/ME	Res4/ME
dep0	1: init	△0.163	0.164	0.418	0.955	0.749	0.773	0.751
	2: listing	0.403	0.404	▲1.000	▲1.000	▲1.000	▲1.000	▲1.000
	3: insurance	0.318	0.317	0.183	0.211	0.180	0.248	0.179
	4: payment-exe	0.167	△0.162	0.088	0.205	0.149	0.151	0.150
	5: payment-r1	0.554	0.561	0.305	0.006	0.006	0.022	0.008
	6: payment-r2	▲1.000	▲1.000	0.084	△0.004	0.003	0.018	△0.005
	7: payment-r3	0.972	0.996	△0.042	0.004	△0.003	△0.013	0.006
	8: payment-AT	0.164	0.170	0.093	0.205	0.148	0.160	0.150
	9: activation	0.402	0.399	0.203	0.215	0.198	0.209	0.198
dep1	1: init	0.167	0.074	0.433	0.956	0.998	▲1.000	▲1.000
	2: listing	0.398	0.176	▲1.000	▲1.000	▲1.000	0.988	1.000
	3: insurance	0.326	0.143	0.188	0.215	0.145	0.210	0.146
	4: payment-exe	△0.162	0.070	0.087	0.208	0.191	0.189	0.189
	5: payment-r1	0.549	0.204	0.253	0.085	0.004	0.021	0.004
	6: payment-r2	▲1.000	▲1.000	0.170	△0.045	△0.002	△0.016	△0.003
	7: payment-r3	0.968	0.469	△0.062	0.050	0.010	0.021	0.011
	8: payment-AT	0.162	△0.070	0.087	0.208	0.192	0.202	0.194
	9: activation	0.400	0.084	0.097	0.626	0.191	0.201	0.190

TABLE 8
Service Reliability Analysis (Scenario-NVP)

DS	Service\Metric	iFOB	boFOB	brFOB	Birnbaum	FF/ME	Res3/ME	Res4/ME
dep0	1: init	0.168	0.169	0.423	0.956	0.748	0.765	0.747
	2: listing	0.410	0.412	▲1.000	▲1.000	▲1.000	▲1.000	▲1.000
	3: insurance	0.326	0.330	0.185	0.201	0.172	0.229	0.171
	4: payment-exe	△0.167	△0.154	△0.081	0.195	0.141	0.152	0.140
	5: payment-r1	0.559	0.564	0.298	0.059	0.058	0.077	0.062
	6: payment-r2	▲1.000	▲1.000	0.527	△0.049	△0.044	△0.064	△0.046
	7: payment-r3	0.994	0.999	0.527	0.049	0.045	0.066	0.047
	8: payment-AT	0.172	0.178	0.094	0.195	0.141	0.153	0.139
	9: activation	0.414	0.398	0.186	0.205	0.189	0.200	0.191
dep1	1: init	△0.165	△0.182	0.416	0.955	0.998	▲1.000	▲1.000
	2: listing	0.409	0.451	▲1.000	▲1.000	▲1.000	0.996	0.996
	3: insurance	0.324	0.359	0.185	0.205	0.137	0.193	0.140
	4: payment-exe	0.168	0.204	0.099	0.199	0.181	0.186	0.179
	5: payment-r1	0.559	0.519	0.252	0.159	0.046	0.061	0.047
	6: payment-r2	▲1.000	▲1.000	0.485	0.104	0.048	0.064	0.049
	7: payment-r3	0.976	0.991	0.481	△0.064	△0.041	△0.057	△0.043
	8: payment-AT	0.169	0.185	△0.090	0.199	0.181	0.192	0.176
	9: activation	0.409	0.214	△0.090	0.628	0.181	0.191	0.179

sensitivity values caused by a positive correlation with other services may be misleading to the system engineer; this is because without a careful check, the engineer may put too many resources into improving a service that is actually not very "reliability-critical" at all. On the other hand, in all the observed cases, DOE/ME-family measures do not show such a problem.

Fig. 8 depicts the scaled metric values[6] of the payment services for dep-0. Here we can see that I_{r1}^{iFOB} and I_{r1}^{brFOB} do not change under different scenarios, since the failure rate and execution probability of the first payment service do not change in the FT blocks. Additionally, I_{r2}^{brFOB} and I_{r3}^{brFOB} are lower than I_{r1}^{brFOB} under scenario-RcB, since the secondary replicas are only executed when all the preceding replicas have all failed. There are no such drops of the brFOB values under scenario-NVP, since all the replicas are always executed whenever the NVP block runs.

On the other hand, the Birnbaum and the DOE/ME-family measures of the replicas are significantly reduced under scenario-RcB and scenario-NVP, successfully reflecting the fault masking capability of the FT designs. We also found that the Birnbaum and the DOE/ME values of the replicas under scenario-NVP are generally greater than the scenario-RcB counterparts, which is not surprising since RcB only fails when all replicas fail, while NVP fails when two or more fail. That is, failures of the replicas have a higher impact on the NVP.

5.3 Sensitivity Analysis for Service Correlations

The three metrics introduced in Section 4.2 (DC, Birnbaum, and FF/PCA) are measured to analyze the importance of each service correlation to the overall system. The MVB-t tol values also range from 0 to 0.026. Normalization is applied to compare the results of each metric by dividing the raw value of each nontrivial DS by the largest one, and the final results are listed in Table 9, which will be discussed in the end.

6. For each metric, the raw values are divided by that of the first payment service in dep-0 to be displayed balancedly in the figure.

The original results of FF/PCA are generated visually in the loading and score plots. Taking the scenario-RcB as an example, the first two PCs totally explain 64 percent variance in Fig. 9, and the experimental instances are well separated when they were grouped into four different quartiles ordered by the resulting system reliabilities in Fig. 10. The numerical value for dependence i is its weighted distance to the point of full dependence set on the loading plot, obtained by

$$D_i^{PCA} = \sqrt{(\lambda_1 x_i - \lambda_1 x')^2 + (\lambda_2 y_i - \lambda_2 y')^2}, \tag{18}$$

where (x_i, y_i) and (x', y') are the locations of the points for DS_{-i} and DS, respectively; λ_1 and λ_2 are the proportion of variance for PC1 and PC2, respectively.

Fig. 11 depicts the DC results for each scenario. Positive DC measures are high-lighted because they indicate optimistic reliability estimations and are generally undesirable. It can be seen that for all scenarios, I_2^{DC}, I_6^{DC}, and I_{13}^{DC} are the most negative and their DSs all exclude data-serial dependencies. On the other hand, all DC measures whose dependence sets exclude data-parallel dependencies are positive in all scenarios, echoing the concerns in the introduction and the findings of Kristiansen et al. [34]. Dependence set 11 excludes only a data-serial dependence (for

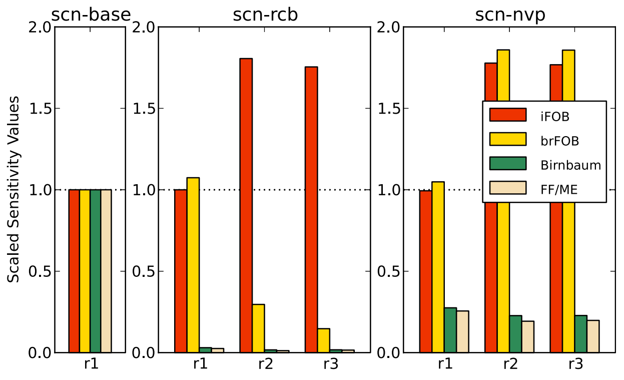

Fig. 8. Service reliability analysis (FT services).

TABLE 9
Service Correlation Analysis

	scenario-base			scenario-RcB			scenario-NVP		
DS\Metric	DC	Birnbaum	FF/PCA	DC	Birnbaum	FF/PCA	DC	Birnbaum	FF/PCA
dep2 (DS_{-DB})	0.929	◆1.000	◆0.991	◆1.000	◆1.000	◆0.989	◆1.000	◆1.000	◆0.991
dep3 (DS_{-ENet})	◇0.317	0.138	0.084	0.361	0.200	0.066	0.112	0.139	0.054
dep4 (DS_{-SHost})	◇0.074	◇0.134	◇0.006	0.408	◇0.105	◇0.005	◇0.071	0.147	◇0.001
dep6 (DS_{-2})	◆0.992	◆1.000	◆1.000	0.907	0.999	◆1.000	0.904	0.998	◆1.000
dep7 (DS_{-3})	0.383	◇0.074	0.062	◇0.095	◇0.101	0.061	0.253	◇0.074	0.054
dep9 (DS_{-5})	0.354	0.165	0.072	0.446	0.195	0.023	0.364	0.269	0.024
dep10 (DS_{-6})	—	—	—	0.369	0.139	0.022	0.403	0.197	0.020
dep11 (DS_{-7})	—	—	—	◇0.276	0.114	0.024	◇0.105	◇0.086	◇0.012
dep13 (DS_{-9})	◆1.000	◆1.000	◇0.009	◆0.979	◆1.000	◇0.004	◆0.974	◆0.999	0.014

Legend: ◆ Top 20% most significant dependence sets, ◇ Bottom 20% least significant dependence sets.

service 7) but I_{11}^{DC} is still positive in scenario-RcB, which is caused by the enhanced recovery ability of the third replica in the RcB mechanism.

Table 9 lists the results of three different measures under various scenarios. For each measure, normalization is performed by dividing the raw values by the largest one; the top 20 percent values (MS-DSs) are marked with solid diamonds, and the bottom 20 percent values (LS-DSs) are marked with hollow diamonds.

It can be seen that all LS-DSs are related to the services along the some branch of the execution paths. For the trunk service (service 2), its associated dependence sets (DSs 2 and 6) are MS-DSs in the table. Generally, DC, Birnbaum, and FF/PCA measures selected similar sets of MS-DSs and LS-DSs except for this conflict: in scenario-base and scenario-RcB, DS 13 (related to the activation service) is selected as an MS-DS under the DC and the Birnbaum measures; however, it is an LS-DS under the FF/PCA measure. In fact, DSs 2, 6, and 13 are indistinguishable under the DC or the Birnbaum measures, but the FF/PCA separates DS 13 from the others by exploring a wider window of service reliability settings.

6 Threats to Validity

To evaluate and highlight some important aspects of this study, a number of threats to validity [39] are reported in this section.

Regarding *conclusion validity*, the process of transforming causal representation to correlation representation (Section 3.1) has been currently implemented in the experiments by statistical sampling and analysis, as presented in (3); however, small variations between each sampling result threatened the reliability of the measures. Averaging of multiple modeling results was used, and Section 5.1 showed that this method provided reasonably reliable results; however, the measuring of correlations between service failures could be improved by more sophisticated techniques such as Bayesian belief propagation [31].

As for *internal validity*, although three different scenarios (base, RcB, and NVP) were used in the experiments, the latter two were FT-enhanced versions of the first one; therefore, they had similar workflow structures. As far as we know, there is no publicly available and widely adopted SBSS implementation for research today. The cases considered in the experiments were based upon a real-world project [14], which covered various control structures (such as sequences, parallelization, branches, and loops), FT mechanisms (such as backward and forward recovery), and dependence conditions that are common in real-world situations; the experimental results did reflect the diversities in such scenarios. The lack of system size diversities is also a research challenge; however, no solution such as the topology generation method [5] was used to widen subject selections; this is because we needed to control the system sizes to meaningfully compare the sensitivity analysis measures, which is related to the following validity concern.

For *construct validity*, sensitivity analysis studies still lack proper criteria to evaluate each analysis method. The concept of sensitivity is not easily explicable unless it is constrained by a specific application goal; thus, its quantification may not be justified objectively. Various sensitivity

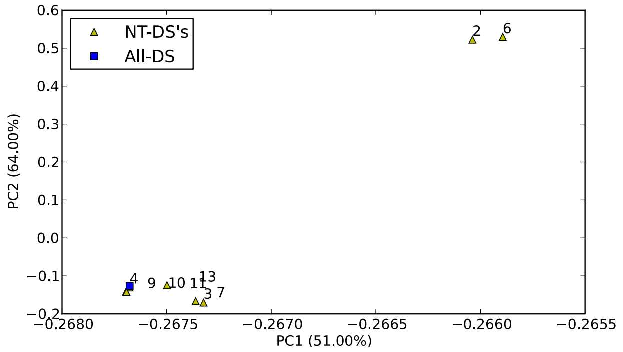

Fig. 9. Loading plot (scenario-RcB).

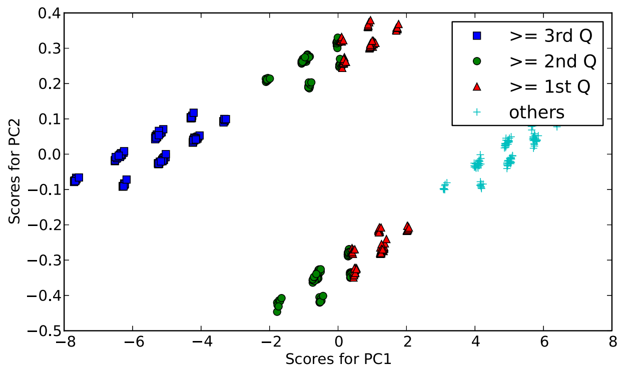

Fig. 10. Score plot (scenario-RcB).

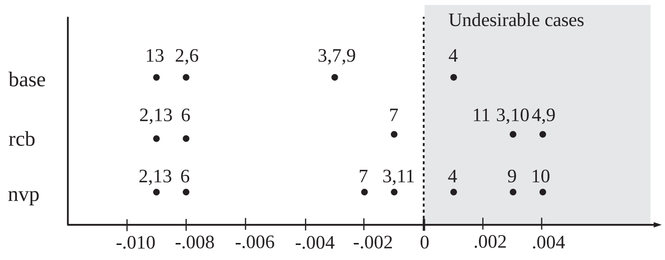

Fig. 11. Line plots for DC results.

TABLE 10
Comparisons of the Service Reliability Measures

Measures	Branch Structure	Failure Recoveries	Failure Correlation	Comput. Efforts
iFOB	–	–	–	very low
brFOB	○	–	–	very low
Birnbaum	●	●	○	low
DOE/ME	●	●	●	higher

Legend: ● high, ○ medium/fair, – low.

measures may be compared more directly in terms of a specific software development cost model or reliability growth model, but the conclusions would therefore not be generalizable to other models. This study therefore took similar approaches to prior studies [34], [36] by regarding the sensitivity measures as ordinal scales and explored some general characteristics of the methods.

Finally, for *external validity*, presumptions of this study such as positive correlation and acyclic dependencies may not apply in rare cases [12]. Research on the analysis of negative reliability correlation has been limited thus far, with few theoretical works [12], [40]. Acyclic dependencies not only are difficult to model but also are undesirable for software engineers. From the transactional point of view, this study modeled only a subset of the transactional relations between services [41], namely *activation*, *abortion*, and *alternative*. Full coverage of the transactional relations would require a more sophisticated model. Another threat would be the inability of the proposed model to make time-variant analysis or predictions. Inputs of the model such as the service reliabilities and correlations would need to be updated periodically to analyze a fast evolving service-based system.

7 Conclusions

This paper presents an SBSS reliability framework that is capable of modeling service reliabilities, service composition structures, and dependent failures. A number of sensitivity analysis measures for individual services and for service correlations are also discussed.

Extensive experimental results for the scenarios, based on a real project, show that the proposed method could accurately model the reliability of SBSS with dependent failures. By comparing the service sensitivity measures (Table 10), it could be seen that the DOE/ME family measures are robust for situations like branches, failure recoveries in FT, and failure correlations, with greater computation. Fortunately, each experimental design is independent, making DOE/ME highly parallelizable, and the reduced experimental designs such as Res-III and Res-IV could be used to enhance the scalability for larger systems. The Birnbaum measure is basically robust and scalable except for its blindness to failure correlations, which may be considered a fast reliability analysis method. On the other hand, failure-occurrence-based measures are intuitive but are vulnerable in many situations.

It is also confirmed that service dependencies (especially data-parallel dependencies that are present in some redundant designs) should be carefully considered, since they tend to increase the probabilities of making optimistic reliability predictions. The proposed dependence sensitivity measures basically identify similar selections of the most significant and least significant dependence sets; however, DOE/PCA measures could differentiate the importance of failure dependencies better than the other measures by exploring a wider space of service reliability configurations.

In all, the results of this study provide a set of algorithmic and statistical tools to help SBSS engineers analyze the dependability attributes and improve the overall system reliability in a more efficient way. The proposed reliability synthesis method may be used in the Goal Question Metric (GQM) [42] approach to investigate the current reliability status of the system. The proposed reliability analysis methods could also be combined with decision mechanisms, such as decision trees [43], and with Web system reconfiguration and optimization approaches [44], [45], to reach the SBSS operation goal.

Acknowledgments

The work described in this paper was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 104-2220-E-007-007.

References

[1] S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan, and M. Muqsith, “Toward development of adaptive service-based software systems,” *IEEE Trans. Serv. Comput.*, vol. 2, no. 3, pp. 247–260, Jul. 2009.

[2] ISO, *The Open Group Service Integration Maturity Model (OSIMM)*, International Organization for Standardization Std. ISO/IEC 16 680:2012, May 2012.

[3] S. Yau and Y. Yin, “QoS-based service ranking and selection for service-based systems,” in *Proc. IEEE Int. Conf. Serv. Comput.*, Jul. 2011, pp. 56–63.

[4] M. Lyu, *Software Fault Tolerance* (Series Trends in Software). Hoboken, NJ, USA: Wiley, 1995.

[5] Z. Zheng and M. Lyu, “A QoS-aware fault tolerant middleware for dependable service composition,” in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2009, pp. 239–248.

[6] W. Gaaloul, K. Gaaloul, S. Bhiri, A. Haller, and M. Hauswirth, “Log-based transactional workflow mining,” *Distrib. Parallel Databases*, vol. 25, no. 3, pp. 193–240, 2009.

[7] W. Gaaloul, S. Bhiri, and M. Rouached, “Event-based design and runtime verification of composite service transactional behavior,” *IEEE Trans. Serv. Comput.*, vol. 3, no. 1, pp. 32–45, Jan.-Mar. 2010.

[8] K. M. Chan, J. Bishop, J. Steyn, L. Baresi, and S. Guinea, “A fault taxonomy for web service composition,” in *Proc. Workshops Serv.-Oriented Comput.*, 2009, pp. 363–375.

[9] C. Xie, B. Li, and H. Leung, “SRM: A staged reliability model for web service,” *Innovations Syst. Softw. Eng.*, vol. 10, no. 2, pp. 137–154, 2014.

[10] B. Zhou, K. Yin, S. Zhang, H. Jiang, and A. Kavs, “A Tree-based reliability model for composite web service with common-cause failures,” in *Proc. 5th Int. Conf. Adv. Grid Pervasive Comput.*, 2010, vol. 6104, pp. 418–429.

[11] L. Fiondella, S. Rajasekaran, and S. Gokhale, "Efficient software reliability analysis with correlated component failures," *IEEE Trans. Rel.*, vol. 62, no. 1, pp. 244–255, Mar. 2013.
[12] B. Littlewood, "The impact of diversity upon common mode failures," *Rel. Eng. Syst. Safety*, vol. 51, no. 1, pp. 101–113, 1996.
[13] M. J. van der Meulen and M. A. Revilla, "The effectiveness of software diversity in a large population of programs," *IEEE Trans. Softw. Eng.*, vol. 34, no. 6, pp. 753–764, Nov.-Dec. 2008.
[14] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer, "Case Study Travel Insurance," in *Applied SOA: Service-Oriented Architecture and Design Strategies*. Hoboken, NJ, USA: Wiley, 2012, ch. 13, pp. 495–539.
[15] Business Process Model and Notation (BPMN) version 2.0. (2011, Jan.). Object Management Group (OMG) Std. [Online]. Available: http://www.omg.org/spec/BPMN/2.0
[16] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors," *IEEE Trans. Softw. Eng.*, vol. SE-11, no. 12, pp. 1511–1517, Dec. 1985.
[17] B. Littlewood and D. R. Miller, "Conceptual modeling of coincident failures in multiversion software," *IEEE Trans. Softw. Eng.*, vol. 15, no. 12, pp. 1596–1614, Dec. 1989.
[18] L. Tomek, J. Muppala, and K. Trivedi, "Modeling correlation in software recovery blocks," *IEEE Trans. Softw. Eng.*, vol. 19, no. 11, pp. 1071–1086, Nov. 1993.
[19] P. Popic, D. Desovski, W. Abdelmoez, and B. Cukic, "Error propagation in the reliability analysis of component based systems," in *Proc. 16th IEEE Int. Symp. Softw. Rel. Eng.*, Nov. 2005, pp. 10–62.
[20] L. Fiondella and S. Gokhale, "Architecture-based software reliability with error propagation and recovery," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst.*, Jul. 2013, pp. 38–45.
[21] Y. Dai, M. Xie, K. Poh, and S. Ng, "A model for correlated failures in N-version programming," *IIE Trans.*, vol. 36, no. 12, pp. 1183–1192, 2004.
[22] M. Jain and R. Gupta, "Reliability assessment of N-version software fault tolerant system with common mode failures," *OPSEARCH*, vol. 51, no. 4, pp. 533–544, 2014.
[23] M. Kristiansen, R. Winther, and B. Natvig, "A Bayesian hypothesis testing approach for finding upper bounds for probabilities that pairs of software components fail simultaneously," *Int. J. Rel., Quality Safety Eng.*, vol. 18, no. 03, pp. 209–236, 2011.
[24] A. Høyland and M. Rausand, "Qualitative system analysis," in *System Reliability Theory: Models and Statistical Methods*. Hoboken, NJ, USA: Wiley, 2009, vol. 420, pp. 73–146.
[25] Z. Tang and J. B. Dugan, "An integrated method for incorporating common cause failures in system analysis," in *Proc. Annu. Symp. Rel. Maintainability*, Jan. 2004, pp. 610–614.
[26] L. Xing and W. Wang, "Probabilistic common-cause failures analysis," in *Proc. Annu. Rel. Maintainability Symp.*, Jan. 2008, pp. 354–358.
[27] R. K. Scott, J. W. Gault, and D. F. McAllister, "Fault-tolerant software reliability modeling," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 5, pp. 582–592, May 1987.
[28] M. Vieira and D. Richardson, "The role of dependencies in component-based systems evolution," in *Proc. Int. Workshop Principles Softw. Evol*, 2002, pp. 62–65.
[29] R.-T. Wang, "A dependent model for fault tolerant software systems during debugging," *IEEE Trans. Rel.*, vol. 61, no. 2, pp. 504–515, Jun. 2012.
[30] K.-L. Peng and C.-Y. Huang, "Reliability evaluation of service-oriented architecture systems considering fault-tolerance designs," *J. Appl. Math.*, vol. 2014, pp. 1–11, 2014.
[31] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2010.
[32] H. Pham, "Reliability engineering measures," *Software Reliability*. Washington, DC, USA: GPO, 2000, ch. 2, pp. 13–36.
[33] Z. W. Birnbaum, "On the importance of different components in a multicomponent system," DTIC Document, Laboratory of Statistical Research, Department Mathematics, University of Washington, Seattle, Washington, Tech. Rep. 54, 1968.
[34] M. Kristiansen, R. Winther, and B. Natvig, "On component dependencies in compound software," *Int. J. Rel., Quality Safety Eng.*, vol. 17, no. 05, pp. 465–493, 2010.
[35] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis: The Primer*.Hoboken, NJ, USA: Wiley, 2008.
[36] X. Li, M. Xie, and S. H. Ng, "Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points," *Appl. Math. Model.*, vol. 34, no. 11, pp. 3560–3570, 2010.
[37] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, Dublin Philosoph. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.
[38] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*.Upper Saddle River, NJ, USA: Prentice-Hall , 2002, vol. 5, no. 8.
[39] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. New York, NY, USA: Springer, 2012, ch. 8, pp. 89–116.
[40] L. Fiondella, "Reliability and sensitivity analysis of coherent systems with negatively correlated component failures," *Int. J. Rel., Quality Safety Eng.*, vol. 17, no. 05, pp. 505–529, 2010.
[41] W. Gaaloul, M. Zaremba, and W. Derguech, "Ensuring customised transactional reliability of composite services," *Innovations Database Des., Web Appl. Inf. Syst. Manage.*, p. 203, 2012.
[42] V. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia Softw. Eng.*, vol. 2, no. 1994, pp. 528–532, 1994.
[43] R. Pressman, "Managing software projects," in *Software Engineering: A Practitioner's Approach* (series McGraw-Hill series in computer science). New York, NY, USA: McGraw-Hill, 2010, pp. 647–784.
[44] J. Li, D. Ma, X. Mei, H. Sun, and Z. Zheng, "Adaptive QoS-aware service process reconfiguration," in *Proc. IEEE Int. Conf. Serv. Comput.*, Jul. 2011, pp. 282–289.
[45] H. Al-Helal and R. Gamble, "Introducing replaceability into web service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 2, pp. 198–209, Apr. 2014.

Kuan-Li Peng received the BE degree and the ME degree in computer science from National Tsing Hua University, Hsinchu, Taiwan in 2007 and 2009, respectively. He is currently working toward the PhD degree in the Department of Computer Science, National Tsing Hua University. He was nominated for Outstanding Paper Award at the 2012 International Conference on Industrial Engineering and Engineering Management (IEEM 2012). His research interests include services computing, software testing, software metrics, software reliability engineering, and wireless networking.

Chin-Yu Huang (M'05) received the MS and the PhD degrees in electrical engineering from National Taiwan University, Taipei in 1994 and 2000, respectively. He is currently a professor in the Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. He was with the Bank of Taiwan from 1994 to 1999, and was a senior software engineer at Taiwan Semiconductor Manufacturing Company (TSMC) from 1999 to 2000. Before joining NTHU in 2003, he was a division chief of the Central Bank of China, Taipei. He received the Ta-You Wu Memorial Award from the National Science Council of Taiwan in 2008. In 2009, he was ranked as the top 15 scholars in systems and software engineering worldwide between 2002 and 2006 by the *Journal of Systems and Software* based on his research on software reliability, software testing, and software metrics. He received an Honorable Mention Best Paper Award at the 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2010). His research interests are software reliability engineering, software testing, software metrics, software testability, and fault tree analysis.