Semi-Markov Models of composite Web services for their performance, reliability and bottlenecks

Zheng Zheng Member, IEEE, Kishor S. Trivedi Fellow, IEEE, Kun Qiu, and Ruofan Xia

Abstract—When combining several services into a composite service, it is non-trivial to determine, prior to service deployment, performance and reliability values of the composite service. Moreover, once the service is deployed, it is often the case that during operation it fails to meet its service-level agreement (SLA) and one needs to detect what has gone wrong (i.e., performance/reliability bottlenecks). To study these issues, we develop a Semi-Markov Process (SMP) formulation of composite services with failures and restarts. By explicitly including failure states into the SMP representation of a service, we can compute both its performance and reliability using a single SMP. We can also detect its performance and reliability bottlenecks by applying the formal sensitivity analysis technique. We demonstrate our approach by choosing a representative example that is validated using experiments on real Web services.

Index Terms—Bottlenecks, Closed-form analysis, Composite Web services, Performance, Reliability, Semi-Markov process.

I. INTRODUCTION

COMPOSITION of multiple Web services is growing in popularity as a convenient way of defining new services within a business process. By combining existing services using a high-level language such as BPEL [1], service providers can quickly develop new services. When deploying these services, service providers often commit service-level agreements (SLAs) with their customers, which include performance and dependability-related metrics. For example, the mean and variance of response times and the service reliability for each incoming request are guaranteed. Since a composite Web service may have complex application logic, it is non-trivial to check whether or not the composed service will meet its SLA. In this paper, we develop an analytical approach to determine the overall performance and reliability of composite Web services.

As an example of such a Web service, we consider a travel plan making process, called TravelPlan. Figure 2 shows a concrete implementation of this process in BPEL. An interesting part of this process is that it tries to select the mode of travel in a unique manner: First, it looks up two different modes of travel in parallel, i.e., railway and airline. When they respond, it chooses one of the modes based on some criterion such as distance to the target, etc. Otherwise, when either of the two travel service fails to respond, it chooses the other one. In the case both fail to respond, it gives up and aborts. Any of the other Web services may also fail to respond, from which we attempt to recover by means of a restart.

Issues we consider here are summarized as follows: (1) Before starting the service, the provider needs to estimate what can be guaranteed to its customers. (2) During operation, it needs to keep its SLA, and in case something goes wrong and the system suffers from degradation, it needs to detect the bottleneck and resolve the problem.

To resolve these issues, we develop a set of semi-Markov models, for computing the performance and the reliability of Web services and detecting bottlenecks. In so doing, we address the following specific challenges: (1) Web services are defined using a rich set of control constructs. These include sequence, parallelism, branch, and loop. Our model can include all the control constructs allowed in BPEL. (2) Restarts in failed activity is allowed in BPEL via fault handlers. We include restarts in our model. (3) We discuss both the mean and variance computation for the performance of composite Web services. (4) We study bottleneck detection based on sensitivity functions and optimization.

The rest of this paper is organized as follows. Section II summarizes the related work. Section III lists the research questions, symbol definitions and assumptions. In Section IV, an SMP model for a BPEL process is constructed, and closed-
form solutions for mean and variance of the response time and reliability are obtained. Section V implements the model on four atomic structures and a case study TravelPlan. Two types of bottleneck detection based on a formal sensitivity analysis are proposed in Section VI. In Section VII, experiments on real web services are conducted to show the effectiveness and properties of the developed approach. The conclusion and future work are presented in the last section.

II. RELATED WORK

In this section, we discuss the previous research in performance and reliability modeling of composite web services. We also give a classification of various known efforts and relate them to our paper.

In the following, we classify the prior efforts according to the problem settings, underlying models, assumptions, analysis techniques, and metrics. Based on this, we design ten dimensions to make the classification as shown in Table I. One dimension of classification is with respect to failure modes: failures of nodes in the web service, failures of underlying h/w/sw resources or both. If a reference considers these failures, it is marked as "yes" in Table I otherwise marked as "no". The second dimension is with respect to recovery: restart/retry after failure of nodes allowed/not, and reboot/repair after the failure of sw/hw resources incorporated or not. The third dimension is with respect to sequential or parallel control structures. The fourth dimension is with respect to structured control constructs (such as conditional, loop etc.) or arbitrary control constructs (such as goto). The fifth dimension is the underlying models employed, such as DTMC, CTMC, SMP, SPNs, directed acyclic Graph and so on. The sixth dimension is about distributional assumption of individual site response times, for example whether exponential distribution (exp in short) is assumed or general distributions (general in short) are allowed. The seventh dimension is about whether the variance analysis of response times is taken in the work. The eighth dimension is the types of solution methods used. In this dimension, closed-form denotes that the solutions are represented by symbolic expressions; numerical solutions are those generated by numerically solving the governing equations, where numerical errors may exist; simulative solutions are arrived at by experiments on a system model to empirically determine its characteristics. The ninth dimension is the specific measures computed: reliability (rel), availability (avail), response time (resp), throughputs (thr) and so on. The tenth dimension is whether sensitivity is considered or not. Now each of the prior effort is classified along the above ten dimensions as shown in Table I.

In recent years, Wang [3], Sharma [4], Ma [5] and Lu [6] presented approaches for modeling and analysis of component-based systems with failures and recovery. After a failure has been detected, recovery strategies are adopted to tolerate

---

**Fig. 1: TravelPlan process**

**Fig. 2: TravelPlan process in BPEL**
### TABLE I: Comparison of related work w.r.t. the 10 dimensions

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Failure</th>
<th>Recovery</th>
<th>Flow patterns</th>
<th>Control structures</th>
<th>Underlying model</th>
<th>Time distribution</th>
<th>Variance analysis</th>
<th>Solution type</th>
<th>Metrics</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>yes</td>
<td>restarts/retries/reboots/repairs</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>D/CTMC</td>
<td>exp</td>
<td>no</td>
<td>numerical</td>
<td>rel</td>
<td>no</td>
</tr>
<tr>
<td>[5]</td>
<td>yes</td>
<td>repairs (one-backup)</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>CTMC</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>rel</td>
<td>no</td>
</tr>
<tr>
<td>[6]</td>
<td>yes</td>
<td>repairs</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>DTMC</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>avail</td>
<td>no</td>
</tr>
<tr>
<td>[7]</td>
<td>yes</td>
<td>repairs</td>
<td>sequential, parallel</td>
<td>none</td>
<td>CTMC</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>rel, resp</td>
<td>yes</td>
</tr>
<tr>
<td>[8]</td>
<td>yes</td>
<td>no</td>
<td>fork-join(^1)</td>
<td>structured</td>
<td>CTMC</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>rel, resp</td>
<td>no</td>
</tr>
<tr>
<td>[9]</td>
<td>yes</td>
<td>no</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>graph</td>
<td>general</td>
<td>no</td>
<td>numerical</td>
<td>rel, resp, cost</td>
<td>no</td>
</tr>
<tr>
<td>[10]</td>
<td>yes</td>
<td>no</td>
<td>sequential, parallel</td>
<td>loop</td>
<td>Probabilistic</td>
<td>general</td>
<td>no</td>
<td>closed-fm</td>
<td>rel, resp, cost</td>
<td>no</td>
</tr>
<tr>
<td>[11]</td>
<td>yes</td>
<td>no</td>
<td>sequential, parallel</td>
<td>All</td>
<td>DTMC</td>
<td>exp</td>
<td>no</td>
<td>numerical</td>
<td>rel</td>
<td>no</td>
</tr>
<tr>
<td>[12]</td>
<td>no</td>
<td>no</td>
<td>fork-join (BPEL)</td>
<td>conditional, loop</td>
<td>SAC(^2)</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>rel</td>
<td>no</td>
</tr>
<tr>
<td>[13]</td>
<td>no</td>
<td>no</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>SPN</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>resp</td>
<td>no</td>
</tr>
<tr>
<td>[14]</td>
<td>no</td>
<td>no</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>SWWF net</td>
<td>exp</td>
<td>no</td>
<td>closed-fm</td>
<td>resp</td>
<td>no</td>
</tr>
<tr>
<td>[15]</td>
<td>no</td>
<td>no</td>
<td>sequential, parallel</td>
<td>conditional, loop</td>
<td>graph</td>
<td>general</td>
<td>no</td>
<td>closed-fm</td>
<td>resp</td>
<td>no</td>
</tr>
<tr>
<td>ours</td>
<td>yes</td>
<td>restarts/retries/reboots/repairs</td>
<td>fork-join (BPEL)</td>
<td>conditional, loop</td>
<td>SMP</td>
<td>general</td>
<td>yes</td>
<td>closed-fm</td>
<td>rel, resp, sens</td>
<td>yes</td>
</tr>
</tbody>
</table>

1. A fork is a transition with more than one output places and a join is a transition with more than one input places.
2. SAC: Structure analysis chart

the failure. In [3]–[4], the authors used an approach similar to ours, which utilized multiple means of failure recovery such as restarts, retries, reboots and repairs. They discussed efficient numerical computation of performance and reliability of hierarchically-nested software systems. In [5], a repair state is considered in the Markov chain modeling of a one-backup service case. The models in complex cases are not elaborated. In [6], the authors mainly focus on the availability analysis. In contrast to the above papers, we focus on finding closed-form results for both mean and variance, while carrying out formal sensitivity analysis at the same time. To the best of our knowledge, there is only a few papers which analyze the sensitivity of composite web services [7]. However, they mainly focus on the sensitivity analysis on services, while this work not only analyzes the sensitivity of services but those of structures as well. In addition, we carry out the analysis without the assumption of exponential distributions, and variance of the response times are computed in this paper. In [16], although the paper also studies the impact of a service on the composite one, it is different since ours is a sensitivity analysis to look for the bottlenecks while they take a correlation analysis to identify services when performance violation happens.

In [8]–[11], although the authors do not consider recovery after a failure in the models, failure states are considered and different fault tolerant schemes are modeled. In the papers, they usually assume that several components or services are available with the same functionalities. The components or services may be considered as backups to each other so that, in case of a failure, the system can dynamically discover and bind to a new component or service as a replacement for the old one. Generally, split/join constructs are used to represent the backup. In contrast with these, we consider recovery after a failure in our models and carry out formal sensitivity analysis.
In [12]–[15], no fault tolerance or recovery strategies are considered in the models. They modeled the performance and reliability for a composite service to capture the internal structural relationships. For example, in [12], the authors map six structural activities into composite services, construct the structure analysis chart and analyze the structures’ reliability. The difference between these papers and ours lies not only in the consideration of fault tolerance and failure recovery but the distributional assumption and sensitivity analysis.

Our reliability model is related to the one that appeared in [17]. The computation of sensitivity functions is discussed in [18] and [19]. Different from these papers, ours is cast in the context of composite web services and sensitivity analysis that we carry out is new.

III. PRELIMINARY

For clarity of exposition, in this section we first present the symbol definitions, assumptions, motivational examples and research questions.

A. Symbol Definitions and Assumptions

We use the following notation throughout the paper.

- \( R_i \): the success probability (or reliability) of the \( i \)th service.
- \( c_i \): the probability of successfully restarting the \( i \)th service.
- \( mrsp_i \): the mean response time of the \( i \)th service.
- \( vrsp_i \): the variance of the response times of the \( i \)th service.
- \( mrsp_{sys} \): the mean response time of the composite web service.
- \( vrsp_{sys} \): the variance of the response time of the composite web service.

We make the following assumptions:

- All services fail independently of each other as well as in their successive executions. Thus, the successful probability of each service can be calculated independently.
- If recovery is available, the recoveries for all services are independent. Thus, the successful probability of recovering each service can be calculated independently of each other.
- If recovery is available, the number of attempts for recovery is not limited. That means, if a service is recovered but failed again, the recovery can be invoked.

B. Motivational example and research questions

We start with a simple case which can be described by the BPEL process in Figure 2 where we never encounter failures. In this case, the composite service is constructed by seven services, i.e., initialization, position, airport, railway, hotel, direction and customer notification. The mean response time of the process can easily be computed as follows.

\[
mrsp_{sys} = mrsp_{int} + mrsp_{gc} + mrsp_{ap} + mrsp_{ht} + mrsp_{di} + mrsp_{rep}
\]

in which \( mrsp_{ap} \) denotes the mean response time of the parallel structure with airport service and railway service, i.e., it is equal to the expected value of the maximum of airport service’s response times and railway service’s response times. \( mrsp_{int}, mrsp_{gc}, mrsp_{ht}, mrsp_{di}, \) and \( mrsp_{rep} \) denote the mean response times of other five services respectively. The overall service reliability is clearly 1, i.e., \( R_{sys} = 1 \).

Next, we consider more complex situations. Suppose each step of a BPEL process may fail. Thus, for example, we suppose that the invocations of Position service may result in failures. To consider these possibilities, we should add a single failure state to the model. Furthermore, to achieve high reliability, BPEL processes often specify recovery procedures, called fault-handlers, which are invoked for restarting failed invocations [1]. To capture this, more states should be considered in the model. Thus, three questions arise:

- How to model a composite web service with failures and restarts?
- Given the performance and reliability of each single service, how to compute the performance and reliability of the composite service?
- How to carry out formal sensitivity analysis on a composite Web service so as to find performance and reliability bottlenecks?

In this paper, we explore these research questions. The reliability of a service denotes its success probability [17] and the performance of a service refers to its mean and variance of response times. In a composite service, the bottleneck of it is the particular activities or parameters that usually cause bad behaviors and need to be improved. To detect the bottleneck of its reliability and performance, the parametric sensitivity analysis [18] are computed in this work, calculating the derivative of the measure with respect to a chosen input parameter.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2475957, IEEE Transactions on Services Computing

Consider a BPEL process with its associated directed graph as shown in Figure 3(a). Each vertex in the figure represents a service. $s_1$ is the start vertex. $S$ has no outgoing edge and is thus a stop vertex. The weight of an edge $(s_i, s_j)$ is interpreted as the conditional probability that the SMP will next execute service $s_j$ after completing service $s_i$. We assume that this probability depends only on the current service and not on the previous history of the system. Therefore, the corresponding Markov chain is homogeneous. We imagine a "dummy" edge forming a self-loop on the absorbing state $S$. With this modification, the transition probability matrix of the Markov chain is given by

$$
P = \begin{bmatrix}
  s_1 & s_2 & s_3 & s_4 & S \\
  0 & p_{12} & p_{13} & 0 & 0 \\
  0 & 0 & 0 & p_{24} & p_{25} \\
  0 & p_{32} & 0 & p_{34} & p_{35} \\
  0 & 0 & p_{43} & 0 & p_{45} \\
  0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

In general, we consider a markov chain with $n$ states, $s_1, s_2, \ldots, s_n$, where $s_n$ is the absorbing state, and the remaining states are transient. The transition probability matrix of such a chain may be partitioned so that

$$
P = \begin{bmatrix}
  Q & C \\
  - & - \\
  0 & 1
\end{bmatrix}
$$

where $Q$ is an $(n-1)$ by $(n-1)$ substochastic matrix (with at least one row sum less than 1) describing the probabilities of transition only among the transient states. $C$ is a column vector and 0 is a row vector of $(n-1)$ zeros.

Now the $k$-step transition probability matrix has the form

$$
P^k = \begin{bmatrix}
  Q^k & C' \\
  - & - \\
  0 & 1
\end{bmatrix}
$$

The $(i, j)$ entry of matrix $Q^k$ denotes the probability of arriving in (transient) state $s_j$ after exactly $k$ steps starting from (transient) state $s_i$. It can be shown that $\sum_{k=0}^{\infty} Q^k$ converges as $t$ approaches infinity [20]. This implies that the inverse matrix $(I - Q)^{-1}$, called the fundamental matrix, $M$, exists and is given by

$$
M = (I - Q)^{-1} = I + Q + Q^2 + \ldots = \sum_{k=0}^{\infty} Q^k
$$

(1)

Let $X_{ij}(1 \leq i, j < n)$ be the random variable denoting the number of times the SMP visits state $s_j$ before entering the absorbing state, given that it started in state $s_i$. We know

Fig. 3: Example of DTMC
that $E[X_{ij}] = m_{ij}$, the $(i, j)$th element of the fundamental matrix $M$ \cite{20}.

Let $N_j$ denote the number of times the state $s_j$ is visited in a typical run of the system and $V_j = E[N_j]$, starting from state $s_1$. Then we get $V_j = m_{ij}$, the element in the first row and the $j$th column of the fundamental matrix $M$.

From Eq.\((1)\), we have $M = I + MQ$. Therefore, we have

$$V_j = \delta_{1j} + \sum_{k=1}^{n-1} V_k \cdot p_{kj}, \quad j = 1, 2, ..., n - 1. \tag{2}$$

Thus, the average visit counts, $V_j$, are obtained by solving a system of $(n-1)$ linear equations, not requiring matrix inversion as in Equation\((1)\).

Now, if $T_{ij}$ denotes the execution (or response) time of $s_i$ at the $j$th visit, then the response time of the SMP as a whole is given by

$$T_{sys} = \left(\sum_{i=1}^{n-1} \sum_{j=1}^{N_i} T_{ij}\right) + T_n \tag{3}$$

since the number of visits to the stop vertex $s_n$ is one. Then

$$mrsp_{sys} = E[T_{sys}] = \sum_{i=1}^{n-1} (V_i \cdot mrsp_i) + mrsp_n \tag{4}$$

where $mrsp_i = E[T_{ij}]$.

The fundamental matrix $M$ can also be used to compute the variance of the expected number of visits \cite{21} \cite{22}. Let $\theta^2_{ij}$ denote the variance of the number of visits to state $j$ starting from state $i$. Then, we have

$$\theta^2 = M(2M_{dg} - I) - M_{sq} \tag{5}$$

where $M_{dg}$ represents a diagonal matrix obtained from matrix $M$ by zeroing out all its off-diagonal entries. $M_{sq}$ denotes the Hadamard product \cite{23} of $M$ with itself, i.e. the $(i, j)$ entry of $M_{sq}$ is given by $(M_{sq})_{ij} = (m_{ij})^2$. Hence,

$$Var[X_{ij}] = \theta^2_{ij} \quad \text{and} \quad Var[N_j] = \theta^2_{ij} \tag{6}$$

Furthermore, since $N_j$ and $T_{ij}$ $(i = 1, ..., n$ and $j = 1, ..., N_i)$ are mutually independent random variables, we have

$$vrsp_{sys} = Var[T_{sys}] = \sum_{i=1}^{n} \left(Var[T_{ij}] \cdot E[N_i] + (E[T_{ij}])^2 \cdot Var[N_i]\right) \tag{7}$$

Now we consider the overall success probability (or Reliability) of the system. A failure state needs to be added to the SMP. In this case, there are two absorbing states in the SMP: the $(n-1)$st is the completion state and the $n$th is the failure state. Figure\(3(b)\) illustrates an example, in which $S$ and $F$ are the absorbing states, representing successful completion and failure. In the example, $R_1$, $R_2$, $R_3$, and $R_4$ are the success probabilities of $s_1$, $s_2$, $s_3$, and $s_4$ respectively. Then the transition probability matrix can be partitioned so that (although actual entries are modified, we use the same notation for simplicity)

$$P = \begin{bmatrix} Q & C \\ 0 & I \end{bmatrix}$$

where $I$ is a 2 by 2 identity matrix. Define the matrix $B = [b_{ij}]$ so that $b_{ij}$ denotes the probability that the embedded DTMC starting with a transient state $i$ eventually gets absorbed in an absorbing state $j$. Then it can be shown that \cite{24}

$$B = MC = (I - Q)^{-1}C \tag{8}$$

Recall that, $M$ is the fundamental matrix of the embedded DTMC. Thus, the composite web service will complete successfully with probability $b_{11}$ and fail with probability $b_{12}$, i.e.

$$R_{sys} = b_{11} = 1 - b_{12}. \tag{9}$$

V. ATOMIC STRUCTURE ANALYSIS BASED ON SMP

In this section, we analyze four atomic structures and the TravelPlan process by the approach described in the last section. In each part of the analysis, we focus on the following three cases.

- Case with no failures: In this case, the BPEL process never encounters failures;
- Case with failures but no restarts: In this case, each execution of a BPEL process may fail with a probability;
- Case with restarts: For higher reliability, the BPEL processes often specify recovery procedures, called fault handlers, which are invoked for restarting failed invocations.

A. Sequential structure

Figure\(4(a)-(c)\) show the DTMCs of the sequential structure in the three cases. In the case with restarts, when a failure is detected, the service goes to a recovery state, where it tries to handle the failure. The recovery may not successful...
Once the

with probability \( c_s \). For example, if a failure is caused by a Mandelbug, the service will recover sometimes and not recover some other times due to its dependence on the execution environment [25]. In the case of an unsuccessful recovery, the service fails, leading to a system failure state. This event has probability \( 1-c_s \) for service \( s_i \).

Equation [1] or 2 can be used to compute \( V_s \) for service \( s_i \) in the structure. In Lemma 1, we provide the results in the case with recovery but prove it in an equivalent way.

**Lemma 1.** For the \( i \)th service in a sequential structure,

\[
V_{s_i} = \frac{V_{s_{i-1}} \cdot R_{s_{i-1}}}{1 - c_s \cdot (1 - R_s)} = \frac{V_{s_1} \cdot \prod_{k=1}^{i-1} R_{s_k}}{\prod_{k=2}^{i} \left( 1 - c_s \cdot (1 - R_{s_k}) \right)}
\]

**Proof:** For the \( i \)th service, the restart may be successful with probability \( c_s \), while it fails with probability \( 1 - c_s \). Once the \( i \)th service starts to run, the mean number of times of executing the service is

\[
\lim_{n \to +\infty} \left( \sum_{k=0}^{n} (c_s (1 - R_s))^k \right) = \frac{1}{1 - c_s (1 - R_s)}
\]

Since the \( i \)th service can start if and only if the \((i - 1)\)st service has been successfully completed, the number of times of service \( i \) starts after running the \((i - 1)\)st service is \( V_{s_{i-1}} \cdot R_{s_{i-1}} \). Thus, we obtain:

\[
V_{s_i} = \frac{V_{s_{i-1}} \cdot R_{s_{i-1}}}{1 - c_s (1 - R_s)}
\]

Iteratively, we can get

\[
V_{s_i} = \frac{V_{s_{i-1}} \cdot R_{s_{i-1}}}{1 - c_s (1 - R_s)}
\]

For the variance, we can use Equation 9 to compute the reliability of the structure. Specially, in the case without failures, overall service reliability is 1. In the case with failures but no recovery, we can get

\[
R_{seq} = \prod_{k=1}^{n} R_{s_k}.
\]

In the case with recovery,

\[
R_{seq} = \prod_{k=1}^{n} \frac{R_{s_k}}{1 - c_s (1 - R_s)}.
\]

**B. Parallel structure**

Since SMP cannot directly model parallel structure, we use the following method to reduce the structure to one state in the DTMC. Suppose \( Z \) is the response time of a parallel structure with \( n \) services. Assuming independence, CDF of \( Z \) can be calculated by [20] for the case without failures

\[
F_Z(t) = P(\max\{Z_i\} \leq t) = \prod_i F_{Z_i}(t)
\]

In this paper, we focus on the case of two parallel components. Next, consider the case with failures. Let \( Z_1 \) and \( Z_2 \) denote the completion times of the two services in a parallel structure in the absence of failures, respectively. The response time in the presence of failures of the parallel structure can be conditioned by the random variable \( Y \) as follows

\[
Y = \begin{cases} 
1, & s_1 \text{ completes and } s_2 \text{ completes} \\
2, & s_1 \text{ completes and } s_2 \text{ fails} \\
3, & s_1 \text{ fails and } s_2 \text{ completes} \\
4, & s_1, s_2 \text{ both fail}
\end{cases}
\]

In case 4, the parallel structure as a whole fails. Hence,

\[
F_Z(t) = \sum_{i=1}^{3} (F_{Z_i}(t)\cdot F_{Y=i})
\]

Note that, \( F_{Z}(t) \) is defective [20]. Thus we can use the following function to calculate the mean response time of the parallel structure.

\[
m_{\text{mispal}} = E[Z] = \int_0^\infty \frac{1 - F_Z(t)}{R_1 + R_2 - R_1 R_2} \, dt
\]

For the variance, we can use...
(a) (b) (c)

Fig. 5: DTMC for Branch Structure

\[ \text{Var}[Z] = E[Z^2] - (E[Z])^2 \]
\[ = \int_0^\infty 2t(1 - F_s(t)) \frac{dt}{R_1 + R_2 - R_1R_2} - \left( \int_0^\infty \frac{1 - F_s(t)}{R_1 + R_2 - R_1R_2} dt \right)^2 \]

In this case, the reliability can be calculated by
\[ R_{poa} = 1 - \prod_{i=1}^n (1 - R_i) \]

C. Branch structure

Figure 5 (a)-(c) show the DTMCs of the branch structure in the three cases, i.e., the case with no failures, the case with failures but no recovery, and the case with recovery. Here, \( b_{pi} \) is the probability of executing branch \( i \). We can use Equation 1 or 2 to compute \( V_{s_i} \) for service \( s_i \) in the structure. In Lemma 2, the results are presented for the case with recovery.

Lemma 2. For the \( i \)-th service in a branch structure,
\[ V_{s_i} = \frac{V_{s_{0i}} \cdot b_{s_i}}{1 - c_i \cdot (1 - R_{s_i})} \]

The lemma can also be proven in the same way as Lemma 1. Similarly, we can get \( V_{s_i} \) for the other two cases:
- For the case with no failures, \( V_{s_i} = b_{r_{si}} \).
- For the case with failures but no recovery, \( V_{s_i} = V_{s_{0i}} \cdot b_{r_{si-1}} \).

After getting the mean visit count for each service, the output measures can be obtained using Equation 4 and 7.

In the case without failures, overall reliability of the structure is 1. For the other two cases, we can use Equation 9 to compute the reliability. Specially, in the case with failures but no recovery, we can get
\[ R_{br} = \sum_{i=1}^n (b_{r_{si}} \cdot R_{s_i}) \]

In the case with recovery,
\[ R_{br} = \sum_{i=1}^n \left( b_{r_{si}} \cdot \frac{R_{s_i}}{1 - c_i (1 - R_{s_i})} \right) \]

D. Loop structure

Figure 6 shows the DTMCs of the loop structure. In the case with restarts, the matrix \( Q \) is given by
\[ Q = \begin{bmatrix} R_l \cdot p_l & 1 - R_l \\ c_i & 0 \end{bmatrix} \]

Lemma 3. For the service in a loop structure,
\[ V_l = \frac{1}{1 + c_l \cdot R_l - c_l - R_l \cdot p_l} \]

Proof: From Equation 1, we obtain
\[ M = (I - Q)^{-1} = \begin{bmatrix} 1 - R_l \cdot p_l & R_l - 1 \\ -c_i & 1 \end{bmatrix} \]
\[ = \frac{1}{1 + c_l \cdot R_l - c_l - R_l \cdot p_l} \begin{bmatrix} 1, & 1 - R_l \\ c_i, & 1 - R_l \cdot p_l \end{bmatrix} \]

From Lemma 1, we know \( V_l = m_{11} \).

Similarly, we can get the visit counts of the restart state,
\[ V_r = \frac{1 - R_l}{1 + c_l \cdot R_l - c_l - R_l \cdot p_l} \]

We can utilize the approach in Section IV to compute the mean and variance of response times.

Similarly, we can get \( V_l \) for the other two cases:
- For the case with no failures, \( V_l = \frac{1}{1 - p_l} \).
- For the case with failures but no recovery, \( V_l = \frac{1}{1 - R_l \cdot p_l} \).

For the calculation of reliability, in the case with failures but no recovery, we can derive the reliability of the system by using Equation 9 as follows:
\[ R_{lp} = \frac{R_l \cdot (1 - p_l)}{1 - R_l \cdot p_l} \]

1939-1374 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
In the case with recovery,
\[ R_{tp} = \frac{R_t \cdot (1 - p_t)}{1 + c_l \cdot R_t - c_l - R_t \cdot p_t} \]  

(19)

E. Performance analysis for TravelPlan process

The structure of BPEL process is mainly represented by several structured activities, including sequence activity, flow activity, while activity, pick activity, switch activity. The sequence activity, flow activity and while activity can be modeled by the sequential structure, parallel structure and loop structure proposed above respectively. The pick activity and switch activity can be modeled by the branch structure \[26\].

In this section, we consider the TravelPlan process. In this case, the BPEL process of Figure 1 can be captured by the embedded DTMC in Figure 7. Figure 7 (a) and (b) show the case, the BPEL process of Figure 1 can be captured by the DTMCs in the case with failures but no restarts and the case with restarts respectively.

The parallel invocation in Figure 2 is reduced to one state. We note that the model here assumes no contention for hardware or software resources. In the future, we will introduce contention by using a product-form queueing network \[20\] or a non-product-form network \[27, 28\].

Using Equation 4, the mean response time of the TravelPlan process can be written down as:
\[ mrsp_{otr} = \text{int} \cdot mrsp_{int} + \text{gc} \cdot mrsp_{gc} + V_{ap} \cdot mrsp_{ap} + V_{ht} \cdot mrsp_{ht} + V_{dir} \cdot mrsp_{dir} + V_{rep} \cdot mrsp_{rep} \] 

(20)

where the mean number of visits to the states are computed using Equation 2

\[ V_{int} = 1 \] 
\[ V_{gc} = \frac{R_{int} \cdot V_{int}}{1 - c_l \cdot (1 - R_{int})}; V_{gc} = (1 - R_{gc}) \cdot V_{gc}; \]  
\[ V_{ap} = (1 - R_{ap}) \cdot V_{ap}; \]  
\[ V_{rep} = (1 - R_{rep}) \cdot V_{rep}; \]  
\[ V_{ht} = \frac{(1 - (1 - R_{ht})(1 - R_{ap})) \cdot V_{ap}}{1 \cdot c_l \cdot (1 - R_{ht})}; \]  
\[ V_{dir} = \frac{R_{ht} \cdot V_{ht}}{1 - c_l \cdot (1 - R_{dir})}; V_{dir} = (1 - R_{dir}) \cdot V_{dir}; \]  
\[ V_{rep} = \frac{R_{ht} \cdot V_{ht}}{1 - c_l \cdot (1 - R_{rep})}; V_{rep} = (1 - R_{rep}) \cdot V_{rep}. \]  

(21)

The mean response time in this case tends to be larger due to:
(1) multiple executions of the same activity; and (2) overheads of restarts.

In the case without failures, overall service reliability is 1. In the case with failures but no recovery (DTMC of Figure 7 (a)), the overall service reliability can be shown to be:
\[ R_{sys} = R_{int} \cdot R_{gc} \cdot (1 - (1 - R_{a1})(1 - R_{a2})) \cdot R_{ht} \cdot R_{dir} \cdot R_{rep} \]  

(22)

in which \(1 - (1 - R_{a1})(1 - R_{a2})\) is the reliability of the parallel structure.

In the case with recovery (DTMC of Figure 7 (b)), the overall service reliability can be calculated by Equation 5:
\[ R_{sys} = \frac{R_{int}}{1 - c_{int} \cdot (1 - R_{int})}; \frac{R_{gc}}{1 - (1 - R_{a1})(1 - R_{a2})}; \frac{R_{ht}}{1 - c_{ht} \cdot (1 - R_{ht})}; \frac{R_{rep}}{1 - c_{rep} \cdot (1 - R_{rep})} \]  

(23)

If all \(c_j\) (\(j = \text{int}, \text{gc}, \text{ap}, \text{ht}, \text{dir}, \text{or rep}\)) are set equal to 0 and 1, the respective lower and upper bounds are obtained respectively.

F. Parameter Value Determination

To compute performance/reliability metrics of TravelPlan, we need to specify the parameters of the DTMC(s) for TravelPlan. Specifically, these parameters are computed from the following types of primitive values.

Executive time of an activity: From a collected sample of \(n\) values, the sample mean and sample variance are computed. We then use the Student t distribution to compute the interval estimate of the mean response time of each activity. We either use the expression for this together with critical values of the t-distribution from a text such as \[20, 29, 30\], or use a statistical analysis package such as R \[31\].

Successful completion probability (Reliability): Since we are concerned only with software failures, the service reliability can also be measured through execution. Actual measurements give us counts of the number of successful tries \(n_s\) out of a total of given number of trials \(n\). The ratio \(n_s/n\) is the sample mean. We can also determine confidence intervals, using formulas based on the Bernoulli sampling \[20\] or using a statistical analysis package.

Overhead time for restarts: The same method as in execution time of each activity.
Success probabilities for restarts: Same method as in the reliability above.

Branching Probabilities: Since BPEL process definitions often include conditional branches (switch) and loops (while), we need to transform these parts of the definitions into probabilistic forms. Same method as in the reliability above can be applied to obtain point and interval estimates.

VI. BOTTLENECK DETECTION

In addition to computing traditional measures of performance and reliability, it is often interesting to determine the performance or reliability "bottleneck" of a system or to optimize system architectures. Towards this end, we discuss parametric sensitivity analysis in this section that involves the computation of derivatives of system measures with respect to various model input parameters. Parameters with large sensitivities usually deserve close attention in the quest to improve system characteristics. These parameters may also indicate elements of a model that are particularly prone to error.

The basic idea is to compute the derivatives of the measure of interest with respect to all the input parameters. These derivatives can then be used to pinpoint the bottleneck [18]. We divide the bottleneck detection into two categories: bottleneck on services and bottleneck on model branching probabilities.

A. Bottleneck on services

For the overall response time mrsp\textsubscript{sys}, we can argue that scaled sensitivities are the relevant quantities in this case so that bottleneck service $I$ is obtained, using the sensitivities $S_k$ ($k$ ranges over the activities), as follows.

$$\text{Bottleneck } I = \arg\max_k |S_k| \quad (i.e. |S_I| = \max_k \{|S_k|\})$$

Denote by $mrsp_i = E[T_i]$ ($i = 1, ..., n$), then we have

$$\text{Sensitivity } S_k = \frac{mrsp_k}{mrsp_{sys}} \cdot \frac{\partial mrsp_{sys}}{\partial mrsp_k}$$

According to Equation 2, i.e.

$$mrsp_{sys} = \sum_{i=1}^{n-1} (V_i \cdot mrsp_i) + mrsp_n$$

Thus, the scaled sensitivity values are derived as follows:

$$S_i = \frac{mrsp_i}{mrsp_{sys}} \cdot \frac{\partial mrsp_{sys}}{\partial mrsp_i} = \frac{mrsp_i \cdot V_i}{mrsp_{sys}} \quad (i = 1, ..., n - 1)$$

$$S_n = \frac{mrsp_n}{mrsp_{sys}}$$

(24)

For the overall reliability $R_{sys}$, we can argue that unscaled derivatives can be used to pinpoint the bottleneck. The bottleneck $J$ should be determined as follows.

$$\text{Bottleneck } J = \arg\max_k |S_k|$$

$$\text{Sensitivity } S_k = \frac{\partial R_{sys}}{\partial R_k}$$

where $R_{sys}$ is the reliability of the system.
Applying this to the second case of TravelPlan (DTMC of Figure 7(a)), we obtain the following formula according to Equation 22:

$$\frac{\partial R_{sys}}{\partial R_k} = \frac{R_{sys}}{R_k}$$

For the third case (DTMC of Figure 7(b)), we show some of its sensitivity values as follows derived from Equation 23:

$$\frac{\partial R_{sys}}{\partial R_{ge}} = \alpha \cdot \frac{1 - e_{ge}}{(1 - e_{ge})^2}$$

$$\frac{\partial R_{sys}}{\partial R_{a1}} = \beta \cdot \frac{(1 - c_{ap}) \cdot (1 - R_{a2})}{(1 - c_{ap} \cdot (1 - R_{a1}) \cdot (1 - R_{a2}))^2}$$

where

$$\alpha = \frac{R_{int}}{1 - e_{int} \cdot (1 - R_{int})}, \ \frac{1 - (1 - R_{a1}) \cdot (1 - R_{a2})}{R_{dir} \cdot (1 - R_{dir}) \cdot R_{rep}}$$

$$\frac{c_{ap}}{c_{ap} \cdot (1 - R_{a1}) \cdot (1 - R_{a2})}$$

$$\beta = \frac{R_{int}}{1 - e_{int} \cdot (1 - R_{int})}, \ \frac{1 - e_{ge} \cdot (1 - R_{ge})}{R_{dir} \cdot (1 - R_{dir}) \cdot R_{rep}}$$

$$\frac{c_{ap}}{c_{ap} \cdot (1 - R_{a1}) \cdot (1 - R_{a2})}$$

B. Bottleneck on model branching probabilities

For the bottleneck on model branching probabilities, the bottleneck \( J \) should be determined as follows:

$$\text{Bottleneck } J = \text{argmax}_k |S_{i \rightarrow J}|$$

$$\text{Sensitivity } S_{i \rightarrow J} = \frac{p_{ij}}{mrsp_{sys}} \cdot \frac{\partial mrsp_{sys}}{\partial p_{ij}}$$

in which \( S_{i \rightarrow j} \) denotes the sensitivity of the connection from state \( i \) to state \( j \).

According to Equation 23, i.e.

$$mrsp_{sys} = \left( \sum_{j=1}^{n-1} V_j \cdot mrsp_j \right) + mrsp_n$$

Thus, the scaled sensitivity values are derived as follows:

$$S_{i \rightarrow j} = \frac{p_{ij}}{mrsp_{sys}} \cdot \frac{\partial mrsp_{sys}}{\partial p_{ij}} = \frac{p_{ij}}{mrsp_{sys}} \cdot \sum_{k=1}^{n-1} \frac{\partial V_k}{\partial p_{ij}} \cdot mrsp_j$$

\((i = 1, ..., n-1)\)

Finally, we can get

$$S_{i \rightarrow J} = \frac{p_{ij}}{mrsp_{sys}} \cdot \sum_{k=1}^{n-1} \left( \frac{\partial V_k}{\partial p_{ij}} \cdot mrsp_j \right)$$

$$= \frac{p_{ij}}{mrsp_{sys}} \cdot \sum_{k=1}^{n-1} \frac{\text{det}(A_k)}{\text{det}(A)} \cdot mrsp_j$$

\((i = 1, ..., n-1)\)

By definition, the sensitivity metric for an activity or a connection tells us about the potential contribution of its improvement to the overall improvement. Thus, it is natural to identify the activity or the connection with the largest absolute value of sensitivity as the bottleneck. Since the derivatives of the same expression with respect to different parameters are taken, the two types of bottlenecks share much of the expression and hence are implicit dependency. The model assumptions could have an impact on the bottleneck detection, especially the independence assumption. How to analyze the bottlenecks after relaxing the assumptions is our future work.
In this section, we evaluate the effectiveness of our approach, using the example in Figure 1. In the experiments, the web services are provided by Baidu via a suite of server-side web APIs. Based on them, we defined a BPEL process for the example shown in Figure 1 and implemented it in Java language. As for the reliability parameters, we have artificially injected failures in the 5 service invocations, namely Position, Railway, Airport, Hotel, and Direction. We have assumed no failures for the other activities, i.e., \( R_{int} = R_{rep} = 1 \). Since Baidu does not provide APIs to restart their services, in our experiments we emulate the restarts by setting the failure handling time as 0.3s in the cases with restarts. Note that, there is no restriction on the distribution of the restart overhead time in our approach.

A. Response times

To validate the proposed model, we run the experiments under different parameter settings. Table II shows the mean and variance of response times computed by the proposed approach. To compare with them, the values measured by our clients are also provided.

For example, in the fourth experiment, we set \( R_{gc} = R_{a1} = R_{a2} = R_{ht} = R_{dir} = 0.9 \), \( c_{gc} = c_{ht} = c_{dir} = 0.9 \) and \( c_{a1} = c_{a2} = 0 \), we invoke the composite web service 500 times, and measure the response times of each single service. The mean and variance of the services are presented in the fourth and fifth columns, i.e., \( mrsp_{gc} = 0.8509s, mrsp_{a1} = 0.7133s, mrsp_{a2} = 0.7580s, mrsp_{ht} = 0.5161s, mrsp_{dir} = 0.5513s \).

Using Equation 13 and 14 we get \( mrsp_{ap} = 0.8336662 \) and \( var_{ap} = 0.27352 \) for the parallel structure with railway service and airport service. Then, we can calculate the mean and variance of response times for the composite web service using Equation 4 and 7 and the result is \( mrsp_{sys} = 2.9673 \) and \( var_{sys} = 0.8871 \).

To validate the results, we measure the response times of the composite service at the same time. Finally, we can get the mean and variance of the measured response times, i.e., \( mrsp_{sys} = 2.9247 \) and \( var_{sys} = 1.0938 \), which are close to what is predicted by the model.

B. Two Types of Bottleneck

Table III shows the two kinds of bottlenecks calculated. We use the fourth experiment as an example. For the bottleneck of services, Equation 24 is used to calculate the scaled sensitivity for each service:

\[
S_{gc} = 0.3151; S_{ap} = 0.2673; S_{ht} = 0.1871; S_{dir} = 0.1977; \\
S_{gc} = 0.0111; S_{ap} = 0; S_{ht} = 0.0109; S_{dir} = 0.0108;
\]

In this case, we observe the Position Service is the performance bottleneck of the system.

For the bottleneck on model branching probabilities, the scaled sensitivity for each connection can be calculated by Equation 30 and the results are as follows:

\[
S_{gc} \rightarrow gc = 0.0111, S_{gc} \rightarrow gc = 0.0284, S_{gc} \rightarrow ap = 0.2673, \\
S_{ap} \rightarrow ap = 0, S_{ap} \rightarrow ap = 0, S_{ap} \rightarrow ht = 0.1703, \\
S_{ht} \rightarrow rht = 0.0109, S_{rht} \rightarrow ht = 0.0168, S_{ht} \rightarrow dir = 0.1799, \\
S_{dir} \rightarrow dir = 0.0108, S_{dir} \rightarrow dir = 0.0178
\]

Based on these, we know the connection from Position Service to the Parallel Searching Service is the bottleneck.

Note that, the setting of the overhead time to conduct a restart has influence on the results. In previous experiments, we set it as 0.3s. If we increase the overhead time of Hotel Service from 0.3s to 2s. Then the performance bottleneck on services will shift from the Position service to Direction service \( (S_{gc} = 0.1781, S_{ap} = 0.1730, S_{ht} = 0.1926, S_{dir} = 0.2039, S_{rgc} = 0.0343, S_{rht} = 0.0288, S_{rdir} = 0.0265) \).

From the experiments, we observe that the composite service’s performance and reliability can be effectively predicted using our approach. We can also detect its two types of bottlenecks by applying the formal sensitivity analysis technique. The merits of our model include: no specific distribution assumption is necessary for the response time of each service, so it is easy to be used in real scenarios; it is not necessary to gather the complete historical data, only mean and variance of the data are necessary; the variance of the overall response time is calculated by the model. Another advantage to use the model is that we calculate reliability of the composed service by including any restarts after failure and imperfection in restarts. We also compute derivatives of the two measures so as to determine bottlenecks. Finally, the complexity of the analysis is relatively low.

VIII. CONCLUSIONS

In this paper, an approach based on Semi-Markov Chains is developed to compute the mean and variance of the overall
response time and the reliability of composite Web services. Closed-form expressions are derived in the general case. The approach is discussed for four types of structures and a typical example under three cases. Furthermore, we show how sensitivity functions can be used to detect bottlenecks. Experimental results on real Web services are used to parameterize and validate our theoretical expressions.

Due to the complexity of real cases, the failures and recoveries of the services could be dependent. In these cases, failure dependence can be introduced in the model by inserting additional states that keep track of memory pertaining to success/failure of previous recovery attempts on this service. The study of these dependencies and the closed form derivation based on them is left to future work.

**REFERENCES**


Zheng Zheng received Ph.D. degree in computer software and theory in Chinese Academy of Science. In 2014 he was with Department of Electrical and Computer Engineering at Duke University, working as a research scholar. He is currently an Associate Professor at Beihang University of China. His research interests include software dependability modeling, and software fault localization.

Kishor S. Trivedi holds the Hudson Chair in the Department of Electrical and Computer Engineering at Duke University. He is an IEEE Fellow and a Golden Core Member of IEEE Computer Society. He is the recipient of IEEE Computer Society Technical Achievement Award. His research interests include reliability, availability, performance and survivability of computer and software systems.

Kun Qiu received B.S. in Automation from HeFei University of Technology. He is currently a PhD candidate in Control Science and Engineering at BeiHang University. His research interests include software performance modeling and software testing.

Ruofan Xia received B.S. in Electronic Engineering from Tsinghua University, China, and M.S. in Electrical and Computer Engineering from Duke University, US. He is currently a PhD candidate in Computer Science at Duke University. His research interests include computer system reliability, availability, performance modeling and optimization with a focus on distributed systems and replicated storage.