Stochastic Modeling of Composite Web Services for Closed-Form Analysis of Their Performance and Reliability Bottlenecks

N. Sato[1] and K.S. Trivedi[2]

[1] IBM Research
[2] Duke University

Abstract. Web services providers often commit service-level agreements (SLAs) with their customers for guaranteeing the quality of the services. These SLAs are related not just to functional attributes of the services but to performance and reliability attributes as well. 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 appropriately. Moreover, once the service is deployed, it is often the case that during operation it fails to meet its SLA and needs to detect what has gone wrong (i.e., performance/reliabilty bottlenecks).

To resolve these, we develop a continuous-time Markov chain (CTMC) formulation of composite services with failures. By explicitly including failure states into the CTMC representation of a service, we can compute accurately both its performance and reliability using the single CTMC. 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 of composite Web services and providing a set of closed-form formulas for its bottleneck detection.

1 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 [13], 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 response time 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 determining the overall performance and reliability of composed Web services.

As an example of such a Web service, we consider a business process, called TravelAgent (Figure 1). Figure 2 shows a concrete implementation of this process in BPEL. An interesting part of this process is that it tries to make the airline reservation in a unique manner: First, it looks up two different airlines for vacancy in parallel. When they respond, it chooses one of the airlines based on some criterion such as fare, schedule, etc.

B. Krämer, K.-J. Lin, and P. Narasimhan (Eds.): ICSOC 2007, LNCS 4749, pp. 107–118, 2007.
© Springer-Verlag Berlin Heidelberg 2007

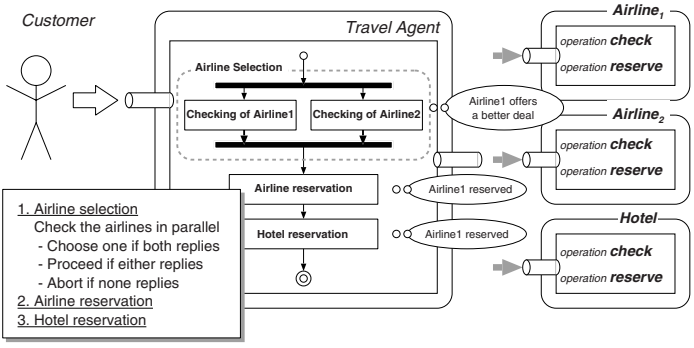


Fig. 1. TravelAgent process

Otherwise, when either of the two airlines fails to respond, it chooses the other airline. In case *both* fail to respond, then it gives up and aborts. Any other Web service may fail to respond, from which we attempt to recover by means of a restart.

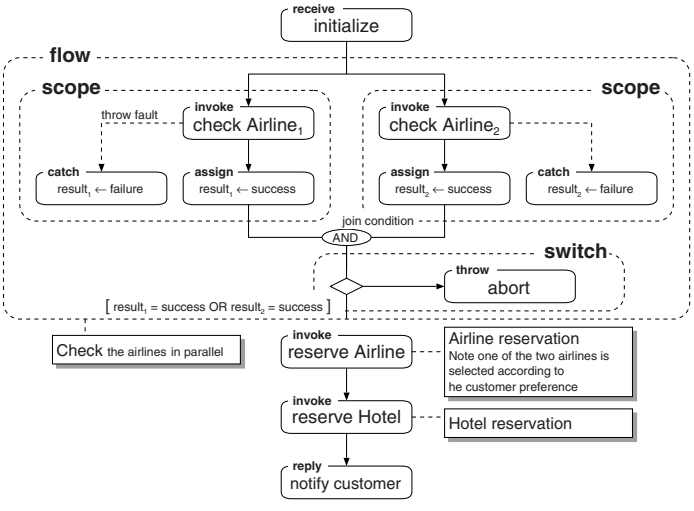


Fig. 2. TravelAgent process (BPEL)

Issues we observe 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 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 switch, while, flow, and scope. Our model will

include all the control constructs allowed in BPEL. (2) Restarts in failed activity is allowed in BPEL via fault handlers. We will include restarts in our model. (3) We will discuss parameterization based on experiments and monitoring. (4) We will primarily be concerned with bottleneck detection, based on sensitivity functions and optimization.

Our contributions are four fold: First, we provide a *continuous-time Markov chain* (CTMC) formulation of composite Web services *with failures*. Then, closed form expressions of the mean response time and the reliability of TravelAgent are derived. Thirdly, bottleneck detection using the formal sensitivity analysis is carried out. Lastly, outline of a solution in the general case is also given.

There are several research efforts related to ours. The IBM BPM engine [4] supports performance *simulation* of BPEL processes. In contrast, we take an analytic approach and we introduce failures and recoveries from failures. In addition, we also consider sensitivity analysis. Our reliability model is related to a paper by Laprie [7]. But ours is cast in the BPEL context and sensitivity analysis that we carry out is new. The paper by Sharma and Trivedi [14] is the closest to current effort. But we find closed form results and carry out formal sensitivity analysis. The computation method for the mean response and reliability we use is described in the paper by Wang [21] and in the book by Trivedi [18]. The computation of sensitivity functions is discussed in [1,8].

2 CTMC Formulation of Composite WS

We assume throughout that times to complete all individual Web services are exponentially distributed. Similarly we assume that the the overhead time to conduct a restart is also exponentially distributed. If desired, these restrictions can be removed, as presented in [19].

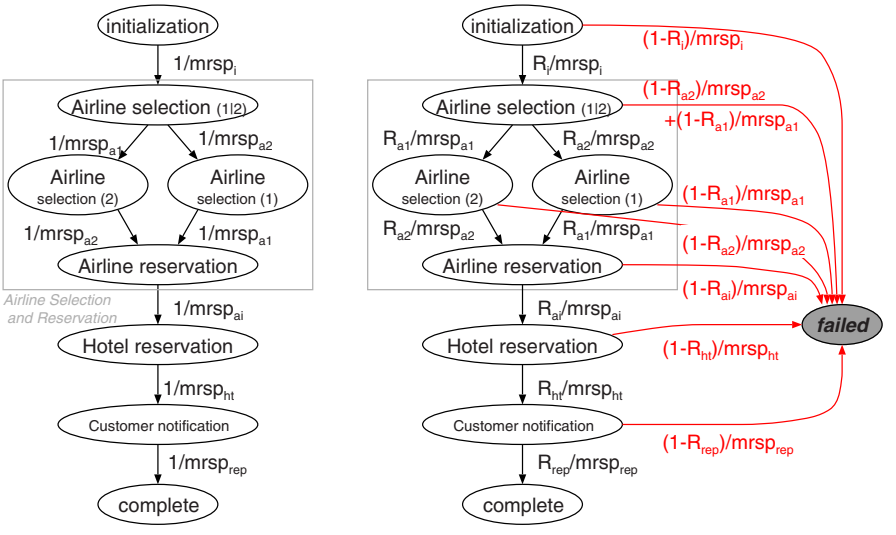
2.1 CTMC for a Process with Concurrency

We start with a simple case where we never encounter failures. In such a case, the BPEL process in Figure 1 can be encoded to the CTMC in Figure 3(a). The parallel invocation in Figure 1 gets translated into 3 states [9]. In the state labeled Airline selection (1,2), both activities are ongoing. After one of them finishes, only the other one is active. Finally, when both finish, we proceed to make the reservation.

We note here that the model here assumes no contention for hardware or software resources. In future, we will introduce contention for resources using a product-form queueing network [18] or a non-product-form network [2,22].

2.2 CTMC with Failures

Each execution of a BPEL process may *fail*. Thus, for example, we suppose that the invocations of the airlines may result in failures. To take account of these possibilities, we add a single failure state to the CTMC. When failure states are added to a CTMC, we need to modify the transition rates of the CTMC in the following manner. Suppose an operation q_1 takes λ^{-1} on average and it has a probability of R for successful completion (i.e., $(1 - R)$ for failure). Then, the successful transition now has a rate $\lambda \cdot R$, while the



(a) With no failure (b) With a single failure state

Fig. 3. CTMCs for the TravelAgent process

other transition (to the failure state) has a rate $\lambda \cdot (1 - R)$. Figure 3(b) is the revised CTMC with failures introduced in the CTMC of Figure 3(a). Note that only if both airline invocations return successfully then we continue, otherwise we abort.

2.3 CTMC with Restarts

For high reliability, BPEL processes often specify recovery procedures, called *fault-handler*s, which are invoked for restarting failed invocations [13]. Figure 4 shows the CTMC with failures and restarts. We have assumed that restart may be successful with probability C while it fails with probability $1 - C$. We also allow for an overhead time for restart. Thus, for instance, upon the failure of the hotel invocation, a restart attempt is made with the mean overhead time of $mrsp_{ht}$ and probability of success as C_{ht}. We assume that there is no restart for the airline invocation. Further that if either one or both airlines invocation is successful, we proceed further in the flow. Upon the failure of both invocations, we abort.

2.4 Response Time and Service Reliability

Now, we are ready to compute the mean response time and the service reliability based on the CTMCs we have developed in the preceding sections. We derive closed form expressions for the mean response time and service reliability based on the CTMCs we have developed in Figure 3(a), 3(b), and 4.

Response Time. We start with the simple CTMC in Figure 3(a). In this case, the mean response time can easily be computed as follows.

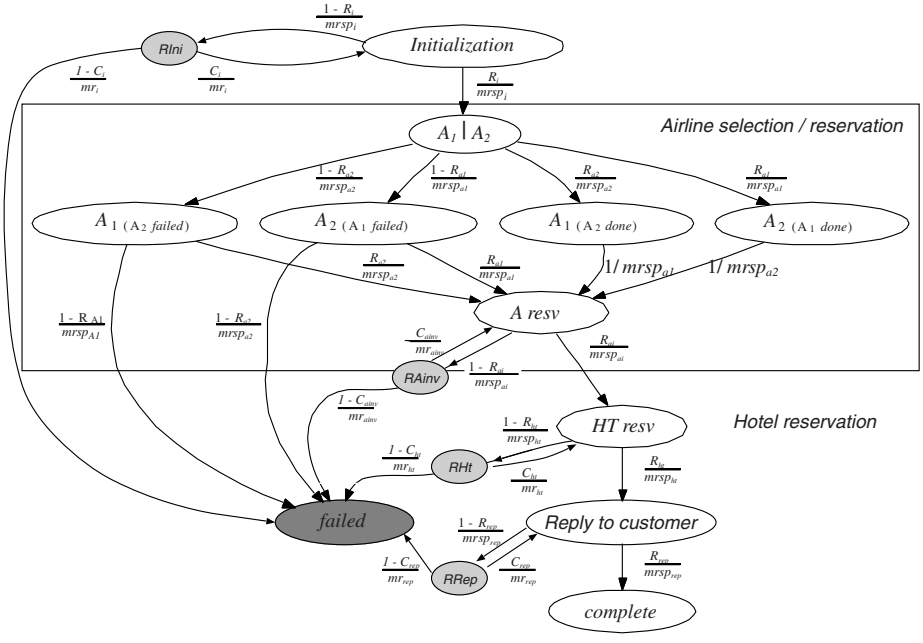


Fig. 4. CTMC with restarts

$$mrsp_{sys} = mrsp_i + \left(mrsp_{a1} + mrsp_{a2} - \frac{1}{\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}}} \right) + mrsp_{ai} + mrsp_{ht} + mrsp_{rep} \tag{1}$$

The expression in the parentheses above is a well-known one for the parallel construct [16].

For the second case (CTMC of Figure 3(b)), the system mean response time can be shown to be:

$$\begin{aligned} mrsp_{sys} = mrsp_i \\ + R_i \cdot \Bigg(& \frac{1}{\left(\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}}\right)} \cdot \left(1 + R_{a2} \cdot \frac{mrsp_{a1}}{mrsp_{a2}} + R_{a1} \cdot \frac{mrsp_{a2}}{mrsp_{a1}}\right) \\ & + R_{a1} \cdot R_{a2} \cdot \left(mrsp_{ai} + Rai \cdot (mrsp_{ht} + R_{ht} \cdot mrsp_{rep})\right) \Bigg) \end{aligned} \tag{2}$$

Note that the mean response time will reduce due to failures since some fraction of requests will not traverse the graph to completion. Also, notice that the above expression (2) reduces to the expression (1) when all reliability values are set to 1.

For the third case (CTMC of Figure 4), the system mean response time can be shown to be:

$$\begin{aligned} mrsp_{sys} = {} & v_i \cdot mrsp_i + v_{1|2} \cdot \frac{1}{\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}}} \\ & + v_{a2} \cdot mrsp_{a2} + v_{f1} \cdot mrsp_{a2} + v_{a1} \cdot mrsp_{a1} + v_{f2} \cdot mrsp_{a1} \\ & + v_{ainv} \cdot mrsp_{ai} + v_{ht} \cdot mrsp_{ht} + v_{rep} \cdot mrsp_{rep} \\ & + v_{rinit} \cdot mr_i + v_{rainv} \cdot mr_{ai} + v_{rht} \cdot mr_{ht} + v_{rrep} \cdot mr_{rep} \end{aligned} \tag{3}$$

where the average number of visits to the states are:

$$\begin{aligned} v_i &= \frac{1}{1 - C_i(1 - R_i)} & v_{ainv} &= \frac{v_{a1} + R_{a2} \cdot v_{F1} + v_{a2} + R_{a1} \cdot v_{F2}}{1 - C_{ai} \cdot (1 - R_{ai})} \\ v_{Rinit} &= (1 - R_i) \cdot v_i & v_{Rainv} &= (1 - R_{ai}) \cdot v_{ainv} \\ v_{1|2} &= R_i \cdot v_i & v_{ht} &= \frac{R_{ai} \cdot v_{ainv}}{1 - C_{ht} \cdot (1 - R_{ht})} \\ v_{a1} &= \frac{R_{a2} \cdot v_{1|2}}{mrsp_{a2} \cdot (\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}})} & v_{Rht} &= (1 - R_{ht}) \cdot v_{ht} \\ v_{a2} &= \frac{R_{a1} \cdot v_{1|2}}{mrsp_{a1} \cdot (\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}})} & v_{rep} &= \frac{R_{ht} \cdot v_{ht}}{1 - C_{rep} \cdot (1 - R_{rep})} \\ v_{F1} &= \frac{(1 - R_{a1}) \cdot v_{1|2}}{mrsp_{a1} \cdot \left(\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}}\right)} & v_{Rrep} &= (1 - R_{rep}) \cdot v_{rep} \\ v_{F2} &= \frac{(1 - R_{a2}) \cdot v_{1|2}}{mrsp_{a2} \cdot \left(\frac{1}{mrsp_{a1}} + \frac{1}{mrsp_{a2}}\right)} \end{aligned}$$

Check again that when all R_k's are set equal to 1, the above expression reduces to Expression 1. Note also that the mean response time in this case will tend to be larger due to: (1) multiple executions of the same activity and (2) overheads of restarts.

For the general case, it will be impossible to find closed-form answers. After first generating CTMC, we can numerically solve for the overall mean response time using a package such as SHARPE [10]. Alternatively, we can first construct a stochastic Petri net from the BPEL description and then automatically generate and solve the underlying CTMC using a software package such as SPNP [17] or SHARPE. Equations to compute the mean time to absorption in a CTMC can be found in [18,21].

Service Reliability. The service reliability is computed in closed form using the equations provided in [21,18]. Refer also to [11] for the reliability computation. In the case without failures (CTMC of Figure 3(a)), overall service reliability is 1.

In the case with failures (CTMC of Figure 3), the overall service reliability can be easily written down as:

$$R_{\text{sys}} = R_i \cdot R_{a1} \cdot R_{a2} \cdot R_{ai} \cdot R_{ht} \cdot R_{rep} \tag{4}$$

Finally, in the case with failures and restarts (CTMC of Figure 4), the overall service reliability in closed-form can be shown to be:

$$\begin{aligned} R_{\text{sys}} = {} & \frac{R_i}{(1 - C_i \cdot (1 - R_i))} \cdot \frac{(R_{a1} + R_{a2} - R_{a1} \cdot R_{a2}) \cdot R_{ai}}{(1 - C_{ai} \cdot (1 - R_{ai}))} \\ & \cdot \frac{R_{ht}}{(1 - C_{ht} \cdot (1 - R_{ht}))} \cdot \frac{R_{rep}}{(1 - C_{rep} \cdot (1 - R_{rep}))} \end{aligned} \tag{5}$$

We note that Expression (5) above does not reduce to Expression (4) if we set each of the coverage probability to 0. In fact, in that case we obtain the lower bound of R_{sys} as follows:

$$R_{sys} = (R_{a1} + R_{a2} - R_{a1} \cdot R_{a2}) \cdot R_{ai} \cdot R_{ht} \cdot R_{rep} \quad (C_k = 0 \text{ for all } k) \tag{6}$$

The reason is that our fault handling procedure says that if either airline succeeds we proceed. Also note that if all C_j are set equal to 1, the upper bound turns out as follows:

$$R_{sys} = R_{a1} + R_{a2} - R_{a1} \cdot R_{a2} \quad (C_k = 1 \text{ for all } k) \tag{7}$$

This is the best case reliability we can obtain.

2.5 Parameterization

To compute performance/reliability metrics of TravelAgent, we need to specify the rate parameters of the CTMC for TravelAgent. Specifically, these rate parameters are computed from the following types of primitive values:

1. Execution time of each activity (i.e., mean response time of each activity)
2. Reliability (i.e., success probability) of each activity
3. Overhead time for restart of each activity
4. Success probabilities for each restart
5. Branching probabilities in the original BPEL graph (if any)

Note that in our particular example, there are no branches in the original BPEL graph.

Execution time of an activity. From a collected sample of n values, the sample mean and sample variance can be computed. We can then use the *Student t* distribution to compute the interval estimate of the mean response time of each activity. We can either use the expression for this together with critical values of the t-distribution from a text such as [12,15,18], or use a statistical analysis package such as R [5].

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 [18] 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), it turns out that we need to transform these parts of the definitions into probabilistic forms. Same method as in the execution time above.

3 Bottleneck Detection

In order to detect bottlenecks to pinpoint the particular activity or parameter that is the cause of bad behavior, we carry out a formal sensitivity analysis. This can be used at design time to point out the activity/parameter that needs to be improved. We can also use this in a realtime setting during the operational phase.

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 [1].

For the Overall Response Time $mrsp_{sys}$. We can argue that scaled sensitivities are the relevant quantities in this case so that bottleneck device I is obtained, using the sensitivity S_k (k ranges over the activities), as follows.

$$\textit{Bottleneck } I = \text{argmax}_k \, |S_k| \quad (i.e.\ |S_I| = \max_k \{|S_k|\})$$

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

For the first case (CTMC of Figure 3(a)), the scaled sensitivity values are derived as follows:

$$S_{a1} = \frac{mrsp_{\text{a1}}}{mrsp_{\text{sys}}} \cdot \frac{\partial mrsp_{\text{sys}}}{\partial mrsp_{\text{a1}}} = \frac{mrsp_{\text{a1}}}{mrsp_{\text{sys}}} \cdot \left(1 - \left(\frac{mrsp_{a2}}{mrsp_{a1} + mrsp_{a2}}\right)^2\right)$$

$$S_{ht} = \frac{mrsp_{\text{ht}}}{mrsp_{\text{sys}}}$$

For the second case (CTMC of Figure 3(b)), the scaled sensitivity values are derived as follows:

$$S_{a1} = \frac{mrsp_{\text{a1}}}{mrsp_{\text{sys}}} \cdot R_i \cdot \left(\frac{{mrsp_{a2}}^2}{(mrsp_{al} + mrsp_{a2})^2} + R_{a2} \cdot \frac{{mrsp_{al}}^2 \cdot mrsp_{a2} + 2 \cdot mrsp_{al} \cdot {mrsp_{a2}}^2}{(mrsp_{al} + mrsp_{a2})^2} - R_{a1} \cdot \frac{mrsp_{a2}}{(mrsp_{al} + mrsp_{a2})^2} \right)$$

$$S_{ht} = \frac{mrsp_{\text{ht}}}{mrsp_{\text{sys}}} \cdot (R_i \cdot R_{a1} \cdot R_{a2} \cdot R_{ai})$$

For the third case (CTMC of Figure 4), the scaled sensitivity values are derived as follows:

$$\begin{aligned} S_{a1} = \frac{mrsp_{\text{a1}}}{mrsp_{\text{sys}}} \cdot \Bigg(& v_{1|2} \cdot \left(\frac{mrsp_{a2}}{mrsp_{al} + mrsp_{a2}}\right)^2 + \left(\frac{\partial v_{a2}}{\partial mrsp_{al}} + \frac{\partial v_{f1}}{\partial mrsp_{al}}\right) \cdot mrsp_{a2} \\ & + \left(\frac{\partial v_{a1}}{\partial mrsp_{al}} \cdot mrsp_{al} + v_{a1}\right) + \left(\frac{\partial v_{f2}}{\partial mrsp_{al}} \cdot mrsp_{al} + v_{f2}\right) \\ & + \frac{\partial v_{ainv}}{\partial mrsp_{al}} \cdot mrsp_{ai} + \frac{\partial v_{ht}}{\partial mrsp_{al}} \cdot mrsp_{ht} + \frac{\partial v_{rep}}{\partial mrsp_{al}} \cdot mrsp_{rep} \\ & + \frac{\partial v_{Rainv}}{\partial mrsp_{al}} \cdot mr_{ai} + \frac{\partial v_{Rht}}{\partial mrsp_{al}} \cdot mr_{ht} + \frac{\partial v_{Rrep}}{\partial mrsp_{al}} \cdot mr_{rep} \Bigg) \end{aligned}$$

$$S_{ht} = \frac{mrsp_{\text{ht}}}{mrsp_{\text{sys}}} \cdot v_{ht}$$

where

$$\frac{\partial v_{a1}}{\partial mrsp_{al}} = R_{a2} \cdot v_{1|2} \cdot \frac{mrsp_{a2}}{(mrsp_{al} + mrsp_{a2})^2} \qquad \frac{\partial v_{a2}}{\partial mrsp_{al}} = R_{a1} \cdot v_{1|2} \cdot \frac{-mrsp_{a2}}{(mrsp_{al} + mrsp_{a2})^2}$$

$$\frac{\partial v_{f1}}{\partial mrsp_{al}} = (1 - R_{a1}) \cdot v_{1|2} \cdot \frac{-mrsp_{a2}}{(mrsp_{al} + mrsp_{a2})^2} \qquad \frac{\partial v_{f2}}{\partial mrsp_{al}} = (1 - R_{a2}) \cdot v_{1|2} \cdot \frac{mrsp_{a2}}{(mrsp_{al} + mrsp_{a2})^2}$$

$$\frac{\partial v_{ainv}}{\partial mrsp_{al}} = \frac{1}{1 - C_{ai} \cdot (1 - R_{ai})} \cdot \left(\frac{\partial v_{al}}{\partial mrsp_{al}} + R_{a1} \cdot \frac{\partial v_{f1}}{\partial mrsp_{al}} + \frac{\partial v_{a2}}{\partial mrsp_{al}} + R_{a2} \cdot \frac{\partial v_{f2}}{\partial mrsp_{al}} \right)$$

$$\frac{\partial v_{rainv}}{\partial mrsp_{al}} = (1 - R_{ai}) \cdot \frac{\partial v_{ainv}}{\partial mrsp_{al}} \qquad \frac{\partial v_{ht}}{\partial mrsp_{al}} = \frac{R_{ai}}{1 - C_{ht} \cdot (1 - R_{ht})} \cdot \frac{\partial v_{ainv}}{\partial mrsp_{al}}$$

$$\frac{\partial v_{Rht}}{\partial mrsp_{al}} = (1 - R_{ht}) \cdot \frac{\partial v_{ht}}{\partial mrsp_{al}} \qquad \frac{\partial v_{rep}}{\partial mrsp_{al}} = \frac{R_{ht}}{1 - C_{rep} \cdot (1 - R_{rep})} \cdot \frac{\partial v_{ht}}{\partial mrsp_{al}}$$

$$\frac{\partial v_{Rrep}}{\partial mrsp_{al}} = (1 - R_{rep}) \cdot \frac{\partial v_{rep}}{\partial mrsp_{al}}$$

For the Overall Reliability R_{sys}. For this case, we can argue that unscaled derivatives can be used to pinpoint the bottleneck: The bottleneck J should be determined as follows.

$$\textit{Bottleneck } J = \text{argmax}_k \, |S_k|$$

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

Applying this to the second case (CTMC of Figure 3), we obtain the following formula:

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

For the third case (CTMC of Figure 4), we show some of its sensitivity values as follows.

$$\frac{\partial R_{\text{sys}}}{\partial R_{\text{al}}} = \alpha \cdot \frac{(1 - R_{a2}) \cdot R_{ai}}{(1 - C_{ai} \cdot (1 - R_{ai}))}$$

$$\frac{\partial R_{\text{sys}}}{\partial R_{\text{ht}}} = \beta \cdot \frac{1 - C_{ht}}{(C_{ht} \cdot R_{ht} + (1 - C_{ht}))^2}$$

where

$$\alpha = \frac{R_i}{(1 - C_i \cdot (1 - R_i))} \cdot \frac{R_{ht}}{(1 - C_{ht} \cdot (1 - R_{ht}))} \cdot \frac{R_{rep}}{(1 - C_{rep} \cdot (1 - R_{rep}))}$$

$$\beta = \frac{R_i}{(1 - C_i \cdot (1 - R_i))} \cdot \frac{(R_{a1} + R_{a2} - R_{a1} \cdot R_{a2}) \cdot R_{ai}}{(1 - C_{ai} \cdot (1 - R_{ai}))} \cdot \frac{R_{rep}}{(1 - C_{rep} \cdot (1 - R_{rep}))}$$

By definition, the sensitivity metric for an activity tells us about the potential contribution of *its* improvement to the *overall* improvement. Thus, it is natural to identify the activity with the highest sensitivity as the bottleneck.

4 Evaluation

We have evaluated the effectiveness of our approach, using the example in Figure 1: We have defined a BPEL process for the example and run it on IBM WebSphere Process

Table 1. MRSP Results

	1a	1b	1c	2a	2b	2c	3a	3b	3c
Computed, using the closed-form expressions ($mrsp_i = mrsp_{rep} = 1$, $R_i = R_{rep} = 1$, $C_{ai} = 1, C_{ht} = 0$, $mr_{ai} = 0.15$)									
MRSP									
$mrsp_{a1}$	2.000	1.000	2.000	2.000	1.000	2.000	2.000	1.000	2.000
$mrsp_{a2}$	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
$mrsp_{ai}$	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
$mrsp_{ht}$	2.000	2.000	1.000	2.000	2.000	1.000	2.000	2.000	1.000
Reliability $R_{a1/a2/ai/ht}$	0.9/0.9/0.9/0.9			0.1/0.9/0.9/0.9			0.1/0.1/0.9/0.9		
$mrsp_{sys}$	8.002	7.336	7.012	7.679	7.012	6.769	4.768	4.101	4.578
S_{a1}	0.187	–	–	0.196	–	–	0.315	–	–
S_{ht}	0.247	–	–	0.237	–	–	0.080	–	–
Measured on WPS									
MRSP									
$mrsp_i$	1.000	1.066	0.976	0.975	0.938	0.932	0.930	1.075	1.054
$mrsp_{a1}$	1.855	0.954	1.886	2.061	1.159	1.936	1.866	0.959	1.826
$mrsp_{a2}$	2.072	1.964	2.177	1.947	1.857	1.999	2.028	1.890	1.960
$mrsp_{ai}$	1.028	1.029	1.032	1.031	1.029	1.028	1.029	1.030	1.028
$mrsp_{ht}$	2.045	2.044	1.053	2.043	2.042	1.043	2.047	2.046	1.044
$mrsp_{rep}$	1.029	1.028	1.032	1.029	1.027	1.026	1.035	1.033	1.030
$mrsp_{sys}$	8.081	7.425	7.171	7.751	6.991	6.825	4.670	4.083	4.530

† *Between (a) and (b) / (c), only the colored parameters have intentionally been changed*

Table 2. Reliability Results

	1a	1b	1c	2a	2b	2c	3a	3b	3c
Computed, using the closed-form expressions ($R_i = R_{rep} = 1$, $C_{ai} = C_{ht} = 1$)									
R_{a1}	0.800	0.900	0.800	0.100	0.200	0.100	0.100	0.200	0.100
R_{a2}	0.800	0.800	0.800	0.800	0.800	0.800	0.100	0.100	0.100
R_{ai}	0.900	0.900	0.900	0.900	0.900	0.900	0.900	0.900	0.900
R_{ht}	0.800	0.800	0.900	0.800	0.800	0.900	0.800	0.800	0.900
R_{sys}	0.768	0.784	0.864	0.656	0.672	0.738	0.152	0.224	0.171
S_{a1}	0.160	–	–	0.160	–	–	0.720	–	–
S_{ht}	0.960	–	–	0.820	–	–	0.190	–	–
Measured on WPS									
R_{a1}	0.804	0.903	0.807	0.103	0.199	0.102	0.099	0.200	0.101
R_{a2}	0.803	0.801	0.797	0.804	0.801	0.797	0.099	0.101	0.101
R_{ai}	0.805	0.800	0.804	0.800	0.805	0.809	0.809	0.803	0.812
R_{ht}	0.800	0.802	0.905	0.800	0.806	0.901	0.796	0.799	0.895
R_{sys}	0.768	0.786	0.868	0.660	0.679	0.736	0.150	0.224	0.171

Server (v6.0). As for the reliability parameters, we have artificially caused failures in the 4 service invocations, namely $\mathsf{Airline}_{1/2}$ (for selection), Airline (for reservation), and Hotel. We have assumed perfect reliability for the other activities ($R_i = R_{rep} = 1$), and

chosen $C_{\text{ai}} = C_{\text{ht}} = 1$ for the coverage parameters. Our evaluation is divided into two parts, and the results are summarized in Table 1 and 2.

First, we focused on performance bottlenecks / improvement in 3 different cases, in each of which we changed either $mrsp_{a1}$ or $mrsp_{ht}$ and evaluated its effect on $mrsp_{sys}$ (Table 1). For example, in Case 1a, *mrsp*s are set to $mrsp_{a1} = mrsp_{a2} = mrsp_{ht} = 2.0, mrsp_{ai} = 1.0$, and R_i, R_{a1}, R_{ht} are all set to 0.9. Then, in Case 1b (1c), $mrsp_{a1}$ ($mrsp_{ht}$) are improved to 1.0. As its result, $mrsp_{sys}$ is improved from 8.002 to 7.336 (7.012). Notice that the higher contribution of the improvement of $mrsp_{ht}$ parallels the fact that S_{ht} is larger than S_{a1} ($0.247 > 0.187$). This applies to the other two cases as well.

Subsequently, we evaluated effects of improvements of reliability values. Since the service reliability does not depend of the *mrsp* values, we do not mention their values. Again, as shown in Table 2, the sensitivity values S_{a1} and S_{ht} successfully suggest which service should be chosen for improving the overall service reliability.

5 Conclusion

We have developed an approach to computing the overall mean response time and the overall reliability of composite Web services. We find closed-form expressions in a typical example. We show how sensitivity functions can be used to detect bottlenecks. Experimental results are used to validate our theoretical expressions. We have also developed an availability model (not shown in this paper) of the system under consideration. We plan to extend our work by providing a tool to carry out such an analysis in the general case. We plan to remove several assumptions made here such as: no contention for resources. We could also remove several distributional assumptions. We plan to use the sensitivity function in a formal optimization setting. We will consider our scheme in a realtime control theoretic setting. We propose to also extend the availability model to include hardware redundancy and software replication as in [6] and consider interactions between the availability model and performance model as in [14],[20], or [3].

References

1. Blake, J., Reibman, A., Trivedi, K.: Sensitivity analysis of reliability and performability measures for multiprocessor systems. In: ACM SIGMETRICS, pp. 177–186. ACM Press, New York (1988)
2. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.: Queueing networks and Markov chains: modeling and performance evaluation with computer science applications, 2nd edn. Wiley-Interscience, New York, NY, USA (2006)
3. Chimento, P., Trivedi, K.: The completion time of programs on processors subject to failure and repair. IEEE Trans. Comput. 42(10), 1184–1194 (1993)
4. Chowdhary, P., Bhaskaran, K., Caswell, N.S., Chang, H., Chao, T., Chen, S., Dikun, M., Lei, H., Jeng, J., Kapoor, S., Lang, C.A., Mihaila, G., Stanoi, I., Zeng, L.: Model driven development for business performance management. IBM Systems Journal 45(3) (2006)
5. Fox, J.: An R and S-Plus Companion to Applied Regression. Sage Publications, Thousand Oaks (2002)

6. Garg, S., Kintala, C., Yajnik, S., Huang, Y., Trivedi, K.: Performance and reliability evaluation of passive replication schemes in application level fault tolerance. In: the 29th Annual International Symposium on Fault-Tolerant Computing, p. 322 (1999)
7. Goseva-Popstojanova, K., Trivedi, K.: Architecture-based Approach to Reliability Assessment of Software Systems. Performance Evaluation 45(2/3), 179–204 (2001)
8. Goyal, A., Lavenberg, S., Trivedi, K.: Probabilistic Modeling of Computer System Availability. Annals of Operations Research 8, 285–306 (1987)
9. Heidelberger, P., Trivedi, K.: Analytic Queueing Models for Programs with Internal Concurrency. IEEE Transactions on Computers 32(1), 73–82 (1983)
10. Hirel, C., Sahner, R., Zang, X., Trivedi, K.: Reliability and Performability Modeling using SHARPE 2000. In: Haverkort, B., Bohnenkamp, H.C., Smith, C.U. (eds.) TOOLS 2000. LNCS, vol. 1786, Springer, Heidelberg (2000)
11. Littlewood, B.: A reliability model for systems with markov structure. Applied Statistics 24(2), 172–177 (1975)
12. Meeker, W., Escobar, L.: Statistical Methods for Reliability Data. John Wiley & Sons, West Sussex, England (1998)
13. OASIS: Specification: Business Process Execution Language for Web Services (1.1) (2004)
14. Sharma, V., Trivedi, K.: Reliability and performance of component based software systems with restarts, retries, reboots and repairs. In: International Symposium on Software Reliability Engineering (2006)
15. Tobias, P., Trindade, D.: Applied Reliability, 2nd edn. Kluwer, Dordrecht (1995)
16. Towsley, D., Browne, J., Chandy, K.: Models for Parallel Processing within Programs: Application to CPU:I/O and I/O:I/O Overlap. CACM 21(10), 821–831 (1978)
17. Trivedi, K.: SPNP User's Manual Version 6.0. Duke University (September 1999)
18. Trivedi, K.: Probability and Statistics with Reliability, Queuing, and Computer Science Applications. John Wiley & Sons, West Sussex, England (2001)
19. Wang, D., Fricks, R., Trivedi, K.: Dealing with Non-Exponential Distributions in Dependability Models. In: Performance Evaluation and Perspectives, pp. 273–302 (2003)
20. Wang, D., Trivedi, K.: Modeling User-Perceived Service Availability. In: Malek, M., Nett, E., Suri, N. (eds.) ISAS 2005. LNCS, vol. 3694, pp. 107–122. Springer, Heidelberg (2005)
21. Wang, W., Choi, H., Trivedi, K.: Analysis of Conditional MTTF of Fault-Tolerant Systems. Microelectronics and Reliability 38(3), 393–401 (1998)
22. Whitt, W.: The queueing network analyzer. Bell System Technical Journal 62(9), 2779–2815 (1983)