Dependability and Rollback Recovery for Composite Web Services

H. Elfawal Mansour and T. Dillon, *Fellow*, *IEEE*

Abstract—In this paper, we propose a service-oriented reliability model that dynamically calculates the reliability of composite web services with rollback recovery based on the real-time reliabilities of the atomic web services of the composition. Our model is a hybrid reliability model based on both path-based and state-based models. Many reliability models assume that failure or error arrival times are exponentially distributed. This is inappropriate for web services as error arrival times are dependent on the operating state including workload of servers where the web service resides. In this manuscript, we modify our previous model (for software based on the Doubly Stochastic Model and Renewal Processes) to evaluate the reliability of atomic web services. In order to fix our idea, we developed the case of one simple web service which contains two states, i.e., idle and active states. In real-world applications, where web services could contain quite a large number of atomic services, the calculus as well as the computing complexity increases greatly. To limit our computing efforts and calculus, we chose the bounded set techniques that we apply using the previously developed stochastic model. As a first type of system combination, we proposed to study a scheme based on combining web services into parallel and serial configurations with centralized coordination. In this case, the broker has an acceptance testing mechanism that examines the results returned from a particular web service. If it was acceptable, then the computation continues to the next web service. Otherwise, it involves rollback and invokes another web service already specified by a checkpoint algorithm. Finally, the acceptance test is conducted using the broker. The broker can be considered as a single point of failure. To increase the reliability of the broker introduced in our systems and mask out errors at the broker level, we suggest a modified general scheme based on Triple modular redundancy and N-version programming. To imitate a real scenario where errors could happen at any stage of our application and improve the quality of Service QoS of the proposed model, we introduce fault-tolerance techniques using an adaption of the recovery block technique.

Index Terms—Bounded set, broker, checkpoint, dependability, doubly stochastic model, rollback, semi-Markov process, series-parallel systems.

1 Introduction

Web services [1], [2], [3], [4] are self-contained component applications that can be described, published, located, and invoked over Internet. Web service systems promote significant decoupling and dynamic binding of component web services. The advantages of the emerging paradigm of secure web services lead to design, development, and quick deployment of the services [3].

The independence of different service publishers and the subscribers can help with selection of the most suitable services. The basic messaging system, *Simple Object Access Protocol* (SOAP) [5], serves as the medium for exchanging the messages among publishers, service brokers, and subscribers. The contemporary web service specification models merely focus on *syntactical* levels, e.g., the *Web Service Description Language* (WSDL) [6], *Web Service Capability Description Language* (SCDL) [7], and *Web Service Choreography Service* (WSCI) [8]. These schemes capture the structural properties of the web components by only using the BPEL and WSCI to weave different web services into meaningful business processes.

In the next generation of systems, web services represent the most important computing paradigms for configuring applications. A lot of work has appeared on developing the middle-ware framework for these web services including the messaging structure, the typical composition of nodes, workflow architectures, orchestration, and choreography of these services. When considering the reliability of these web services, much work has been done on the reliability of the messaging structure [9], [10]. Using software reliability models, some work have also been done on the reliability of "*individual* or *atomic*" web services [11], [12], and [13].

The service composition and selection are central activities in service-oriented computing, and the *Quality of Service* QoS of a *Service-Oriented Architecture* SOA plays a key role to appropriately drive these activities. A key issue for the *QoS* analysis of *SOA* is parameter estimation. The properties of basic services are not easily made available from service providers. Thus, monitoring/estimation techniques are needed to populate *QoS* models with values of basic service characteristics such as the probability of failure (Pf) or the reliability [14]. Concerning the reliability modeling for *SOA*, the architectural models are partitioned as follows: 1) path-based models, where the reliability of an assembly of components is calculated starting from the reliability of architectural paths; 2) state-based models, where the reliability is calculated starting from the reliability of

- *H. Elfawal Mansour is with the College of Computers and Information Technology, University of Tabuk, PO Box 741, Tabuk 71491, KSA. E-mail: mansour@ieee.org.*
- *T. Dillon is with the Digital Ecosystems and Business Intelligence Institute (DEBII), Curtin University of Technology, Enterprise Unit 4, De Laeter Way, Technology Park, Bentley, WA 6102, Australia. E-mail: Tharam.Dillon@cbs.curtin.edu.au.*

Manuscript received 5 May 2009; revised 8 Mar. 2010; accepted 27 Mar. 2010; published online 28 Apr. 2010.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSCSI-2009-05-0109. Digital Object Identifier no. 10.1109/TSC.2010.16.

system states and from the transition probabilities among states. We should mention here that the models in the literature assume the independence among different components of web services, the independence between the web service and the load on the server or as well as the independence of the volume and the nature of traffic wishing to utilize the service over a given time horizon. Application developers simply focus on the arrangement topology of the service within a workflow graph [15].

In real-world applications, the above-mentioned assumptions cannot be fulfilled. In this case, the loading level on the server may change its failure characteristics. Therefore, common mode failures are possible due to a server having multiple services residing on it failing. The service execution could be blocked or experience long delays due to congestion at a particular service or broker arising from traffic. To properly model the reliability of web services, one has to take into account

- the software reliability of the individual web service,
- the reliability of the related server on which it is running,
- the loading levels on the running server, and
- the possibility of execution blocking of a particular service or group of services due to the volume and the nature of the traffic.

In our previous work [16], we proposed a mathematical model for software based on the doubly stochastic and renewal processes. Due to differences between traditional software and web services [12] (nature of failure, reliability definition, and interactions) and based on our previous software model, we develop a new model for web services in order to model the error arrival time as a function of the workload of the server.

Now concerning the requirements of a user, generally it cannot be met by a single web service and users may require the composition of many component web services. This raises two problems.

- The first can be the selection of the required component web services and their composition into the composite.
- The second is the execution coordination of different web services in this composite. Therefore, they are executed in the right sequence and the preconditions for a component service are met before execution. If the coordination is centralized, then orchestration is used for coordination [1]. On the other hand, if the coordination is distributed, then choreography is used for coordination [17], [3].

To improve the overall reliability of composite web service, one can use redundancy [18] and [19]. Here, by redundancy we mean that the same kind of functionality is available in the web service provided by different service providers [20]. So in this case, we can still provide the same functionality even when some of the web services are not available due to a failure or other reasons. Also, the redundancy of web services often has a high diversity. Web services delivered by different service providers are often developed individually. The diversity makes it much less likely that the same failure would hit all redundant web services.

In the case of complex composite configurations, a state explosion problem is likely to arise and has to be managed. So to help in reducing the complexity of required calculations, the reliability calculation is evaluated using our previously proposed bounded set (BS) approach [21], [22], [23] to deal with large state cases.

After the modeling stage, the fault tolerance and execution error at the web service level were introduced by using an adaption of the recovery block technique. The broker is supposed to have a built-in acceptance testing mechanism that examines the results feedback from a particular web service. If the returned results (i.e., the feedback results) can be deemed acceptable, it proceeds with the computation to the next web service. If they are considered to be unacceptable, they can involve rollback and another component web service to carry out the same function previously required from the faulty component web service. This can be done by saving the states and creating checkpoints [24].

Although a broker with such an acceptance testing mechanism increases the trust in the composite web services, it still constitutes a single point of failure. One way of addressing this would be to provide redundancy at the broker level using "Triple modular redundancy (TMR)" and "N-version programming" [25] to mask out errors at the broker level.

This paper is divided into three main parts. In the first part, we develop a reliability model based on the Doubly Stochastic model and Renewal processes. This model shows the dependency among arrival time errors and different operating states including workload states of the server where the web service resides. In the second part, the bounded set approach is briefly introduced to help scale reliability calculations for complex composite configurations. In the third section, a *fault-tolerance* model is developed for composite web services with centralized coordination using recovery block techniques, using the *Doubly Stochastic* reliability model and the *bounded set* approach. Finally, simulation results are presented and discussed.

2 Reliability Model Based on the Doubly Stochastic Model and the Renewal Processes

A main issue for the *QoS* analysis of *SOA* is the parameter estimation of the atomic web services and their combination. Concerning the atomic web services, that we study in this section, many reliability models assume that the failure or error arrival times are exponentially distributed. This is inappropriate for web services as the error arrival times will be dependent on the operating state including workload of the server where the web service resides. Previous work [26] indicates that the probability of errors occurring in a computer system is higher where the workload is higher. Due to differences between traditional software packages and web services [12] (nature of failure, reliability definition, and interactions), we develop a new reliability model for web services based on our previous model for software packages [16]. In fact, for the case of web services, the probability of an error occurring is also greater when the service is in an active state than in an idle state. Hence, in modeling *error arrival times* in web services, it is more

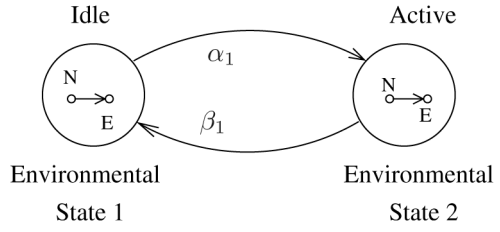

Fig. 1. α_1 stands for the pdf of the transition from environmental $state\ 1$ to environmental $state\ 2$; β_1 is the pdf of the reverse transition.

appropriate to use the *randomly varying environment* notion. Randomly varying environments for the general reliability problem have been introduced in [27].

2.1 Reliability Modeling for a Single Web Service

In this part, we develop the new model for the reliability of a web service in order to include the influence of different parameters of the environment.

In our model, we consider two operating environments, namely

- State 1: The web service is idle.
- State 2: The web service is active and processing.

We associate with each of these states certain probabilities that the web service will remain error free (state N) or erroneous (state E). These probabilities are assumed to be exponentially distributed with hazard rates λ_1 and λ_2. λ_i can be the sample function of a stationary continuous stochastic process $\Lambda(t)$ of the operating environment. In this case, the reliability R_i of the system during a particular type of cycle can be

$$R_i(x) = e^{-\lambda_i x}, \tag{1}$$

where $i = 1, 2$. This process is known as a *doubly stochastic poisson process* and it is illustrated in Fig. 1. Note in this figure that when the web service enters any of the environmental states, it is entering a new operating environment, i.e., the process is renewed constituting an embedded renewal process and the states are regenerative giving an *embedded semi-Markov process* [28]. In contrast of the exponential probability density function (pdf) transitions within a state, it is likely that transitions among environmental states will be characterized by their means and their variances. This will be developed in the following section.

2.2 Distribution of the Environmental States

The variance concerning the transitions among environment states depends on the load of the server where the web service resides and the type of processing being conducted. The distribution of these environmental states may not be exponentially distributed, but assumes the shape shown in Fig. 2. The curve before the mean represents the distribution of an environmental state when the system is lightly loaded and the curve after the mean represents a heavily loaded system. A probability distribution representing this curve is given by

$$f(t) = \frac{\mu^k t^{k-1}}{\Gamma(k)} e^{-\mu t}, \tag{2}$$

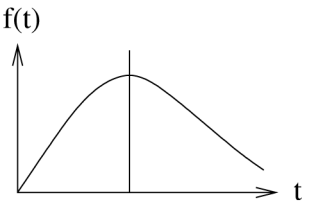

Fig. 2. Pdf of an environmental state.

where $\Gamma(k) = (k-1)!$ is the *Euler* function of first order, k is the order of the distribution, and μ is the number of events per time unit. The Laplace transform £ of (2) becomes

$$\pounds(f(t)) = F(s) = \left[\frac{\mu}{s+\mu}\right]^k. \tag{3}$$

For a particular value of k, the probability distribution is known as kth order $Erlangian$ distribution. The cumulative distribution function (Cdf) corresponding to the probability density function $f(t)$ is

$$F_k(t) = 1 - e^{-\mu t} \sum_{i=0}^{k-1} \frac{(\mu t)^i}{i!}. \tag{4}$$

Fig. 3 shows the pdf for different $Erlangian$ orders. We can see that as k increases, the distribution has a lower dispersion. A measure of dispersion is the coefficient of variation C_v given by

$$C_v = \frac{S_d}{E[t]}, \tag{5}$$

where S_d is the standard deviation, the mean $E[t]$ is equal to $\frac{k}{\mu}$, and C_v is equal to $\frac{1}{\sqrt{k}}$. So, the distribution of the environmental states can be modeled by an appropriate choice of k.

2.3 Mean Time to Error MTTE Formulation

Let us first give the following probability definitions:

- $P_N = \Pr\{No\ error$ occurring in the system, in the time interval $[0, t]\}$
- $P_{N,i} = \Pr\{$the system is in states (N, i), in the time interval $[0, t]\}$
- $P^0_{N,i} = \Pr\{$the system is in states (N, i), in $[0, t] \mid$ the system is in states (N, i) at $t = 0\}$

where i is the corresponding *State* number (1 or 2) and N means that there's *No error*. The development of these

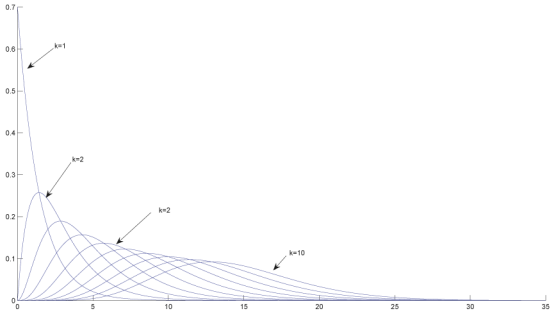

Fig. 3. Pdf for different orders of Erlangian distributions.

probabilities in certain time interval $[0,t]$ mainly depends on two parts; one part depends on the probability that it's a working state in the corresponding time interval $[0,t]$ being sure that no transitions are done before the time t. And the second part depends on the renewal process that consists on the continuity of the transitions between the states.

Under the assumption that we are in *state* 2 at $t=0$, different probabilities will be given by

$$\begin{aligned}
P^0_{N,1}(t) &= R_1(t)\int_t^{+\infty}\alpha_1(x)dx \\
&\quad + \int_0^t R_1(x)\alpha_1(x) * R_2(x)\beta_1(x) * P^0_{N,1}(x)dx, \\
P_{N,1}(t) &= \int_0^t R_2(x)\beta_1(x) * P^0_{N,1}(x)dx, \\
P^0_{N,2}(t) &= R_2(t)\int_t^{+\infty}\beta_1(x)dx \\
&\quad + \int_0^t R_2(x)\beta_1(x) * R_1(x)\alpha_1(x) * P^0_{N,2}(x)dx,
\end{aligned}$$

where $R_i(t)$ is the reliability of state i according to (1), $\alpha_1(t)$ is the probability that the *Active* state (*State 2*) occurs after t seconds in *State 1*, and $\beta_1(t)$ is the probability that the *Idle* state (*State 1*) occurs after t seconds in *State 2*. The *Laplace* transform of the previous equations is given by

$$\begin{aligned}
P_{N,1}(s) &= \frac{1-\alpha_1^*(s+\lambda_1)}{s+\lambda_1} \\
&\quad \times \frac{\beta_1^*(s+\lambda_2)}{1-\alpha_1^*(s+\lambda_1)\beta_1^*(s+\lambda_2)},
\end{aligned} \tag{6}$$

$$\begin{aligned}
P^0_{N,2}(s) &= \frac{1-\beta_1^*(s+\lambda_2)}{s+\lambda_2} \\
&\quad \times \frac{1}{1-\alpha_1^*(s+\lambda_1)\beta_1^*(s+\lambda_2)}.
\end{aligned} \tag{7}$$

Here, the function $f^*(s)$ stands for the *Laplace* transform of $f(t)$.

In our model, the environmental states, assumed to be *Erlangian* distributed, as discussed in the previous section, will be expressed as set out previously in (2). This gives

$$\alpha_1(t) = \frac{a_1^k t^{k-1}}{\Gamma(k)} e^{-a_1 t}, \tag{8}$$

$$\beta_1(t) = \frac{b_1^k t^{k-1}}{\Gamma(k)} e^{-b_1 t}, \tag{9}$$

where $\alpha_1(t)$ is the pdf of the transition from the $Idle$ state to the $Active$ state and $\beta_1(t)$ is the pdf of the transition from the $Active$ state to the $Idle$ state. The corresponding Laplace transform becomes

$$\alpha_1^*(p) = \left[\frac{a_1}{s+a_1}\right]^k, \tag{10}$$

$$\beta_1^*(p) = \left[\frac{b_1}{s+b_1}\right]^k. \tag{11}$$

The probability of *no error* occurring in the system will be given by

$$P_N(s) = P_{N,1}(s) + P^0_{N,2}(s). \tag{12}$$

In general, depending on the probability density function of the environment states, the inverse Laplace transforms may be difficult to obtain. In many cases, however, it is sufficient to obtain the steady-state solutions from the Laplace transform domain which is expressed as

$$E_X[t] = P_X(s)|_{s=0}, \tag{13}$$

where $E_X[t]$ is the expected value of t for the random variable X, and $P_X(s)$ is the Laplace transform of the cumulative distribution function of X. So, the *MTTE* occurrence can be given by

$$MTTE = \lim_{s\to 0} P_N(s). \tag{14}$$

Under the three randomly varying environments, the *MTTE* of the system will be

$$MTTE = P_{N,1}(s=0) + P^0_{N,2}(s=0). \tag{15}$$

Substituting in the values and assuming $s=0$ yields

$$\begin{aligned}
MTTE &= \frac{1}{1-\alpha_1^*(\lambda_1)\beta_1^*(\lambda_2)} \\
&\quad \times \left[\frac{[1-\alpha_1^*(\lambda_1)]\beta_1^*(\lambda_2)}{\lambda_1} + \frac{1-\beta_1^*(\lambda_2)}{\lambda_2}\right],
\end{aligned} \tag{16}$$

where $\alpha_1^*(p)$ is the *pdf Laplace* transform of the transition from the $Idle$ state to the $Active$ state as given in (10) and $\beta_1^*(p)$ is the *pdf Laplace* transform of the transition from the $Active$ state to the $Idle$ state as given in (11). Some basic probability functions and examples could be found in [29].

2.4 Experimental Results for the Doubly Stochastic Model

It is clear that the MTTE given by (16) depends on various parameters, as the workload on the system relied on a_1 (a_1 is the number of events per time unit for α_1), the error rates λ_i, the *Erlangian* order k, and the speed of the processing in the active state relies on b_1 (b_1 the number of events per time unit for β_1). We assume that when the web service is idle, the error rate λ_1 will be much smaller than λ_2 when it is *active*. Herein after, the effects of these parameters on the MTTE evolution are considered.

- **Error rates variation.** Many experiments have shown that errors tend to occur more frequently as the workload on a system increases. On the proposed model presented in Fig. 4, the load can be represented by increasing the rate of transition between the *idle* state and the *active* state, i.e., increasing the value of a_1. We can see that when the error rates are equal to λ, the MTTE remains constant at $\frac{1}{\lambda}$. The environments have no longer any effects on the process. For a high ratio between λ_1 and λ_2, we see that as a_1 increases, the MTTE decreases from $\frac{1}{\lambda_1}$ to $\frac{1}{\lambda_2}$. The system enters the *active* state more frequently and the MTTE becomes dependent on the error rates λ_1 and λ_2.
- **Erlangian order k.** Based on the definition in the *Laplace* domain of α_1 and β_1 shown in (3), we can notice that when s tends to zero, $F(s)$ tends to 1 with

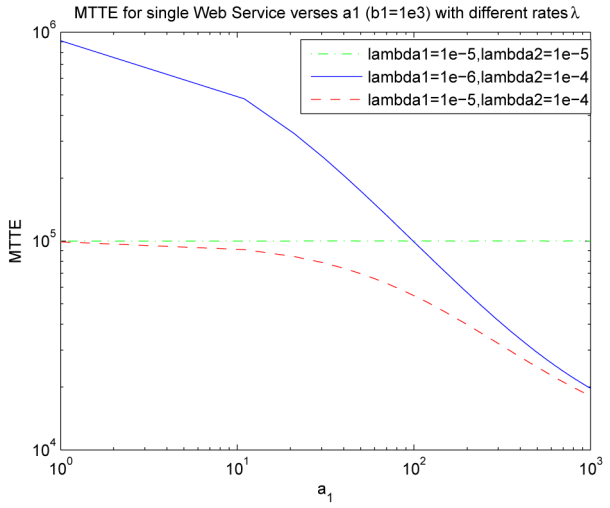

Fig. 4. MTTE versus a_1 with different error rates λ_1 and λ_2.

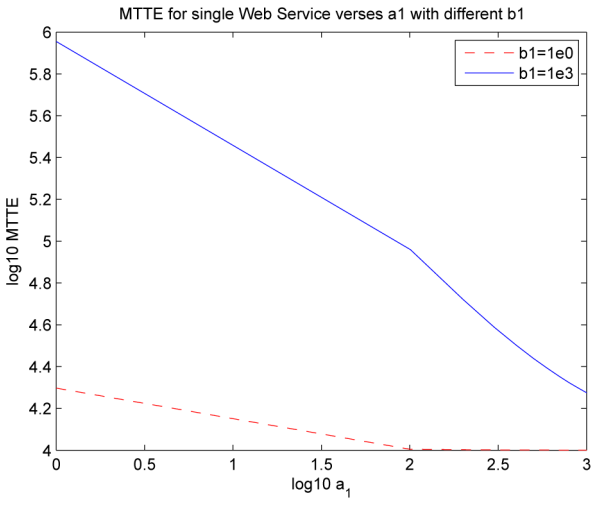

Fig. 6. MTTE versus a_1 with different b_1.

any *Erlangian* order k. Fig. 5 shows the MTTE with different Erlangian orders $k = 2$ and $k = 8$. We can see that the MTTE for $k = 1$ and the MTTE for $k = 8$ are almost the same with a *relative error* equal to 0.0149 which shows that the *Erlangian order* has little influence on the results.

- **Speed of the processing depending on b_1.** For small b_1, the *active* state is slow in processing before finishing it's action and going back to the *idle* state. In this case, we can see in Fig. 6 that the MTTE depends more on the reliability of the system in the *active* state ($\frac{1}{\lambda_2} = 10^4$).

To have a better overview, we show in Fig. 7 the $3D$ plot of MTTE variation with a_1 and b_1 with different error rates.

To conclude this section, we find that the effect of the *Erlangian* order k is negligible and that more the workload on a system increases, the errors tend to occur more frequently (*MTTE* decreases). Also, the evolution of the *MTTE* is bounded by the inverse of the error rates of the system states. This time variable *MTTE* must help us to approximate the time variable reliability for each web service in order to evaluate the reliability of a complex scheme. As $MTTE = 1/\lambda$ [29], [30], where λ is the error rate, having the $MTTE$ for each time t, we approximate the reliability using the *Exponentially Distributed* formula as following:

$$R(t) \simeq e^{-\frac{1}{MTTE}t}. \tag{17}$$

In this way, we can have the time-dependent reliability depending on the loading on the server for different web services and these models will be used for the composite system reliability calculations.

3 Failure Probability Using the Bounded Set Technique

The requirements of a user will often not be met by a single web service, but require the composition of many component web services.

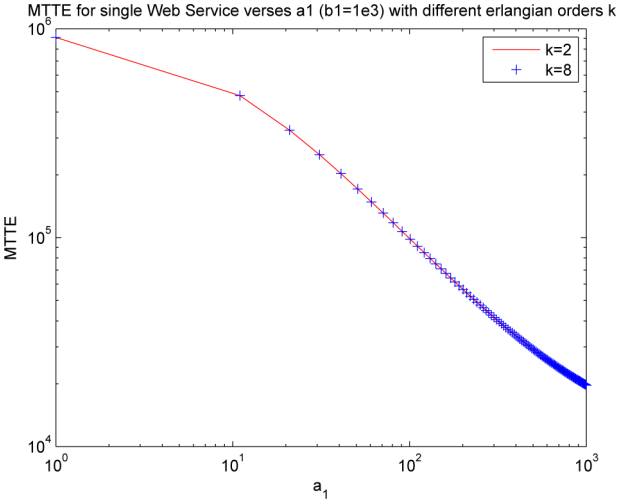

Fig. 5. MTTE versus a_1 with different Erlangian orders k.

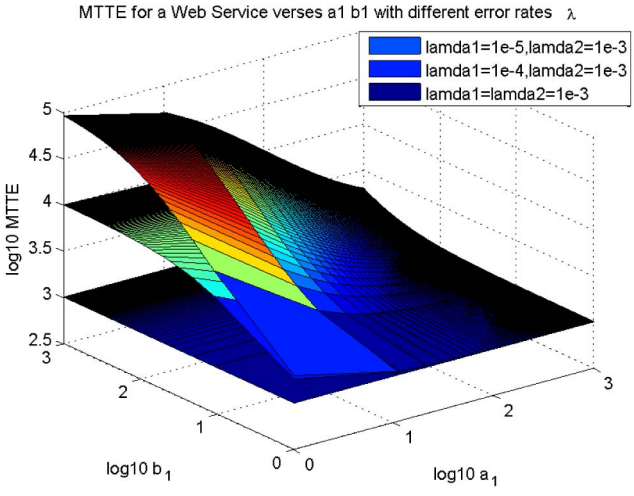

Fig. 7. MTTE versus a_1 and b_1 with different error rates λ_1 and λ_2.

A main issue for the *QoS* analysis of *SOA* is the parameter estimation. Concerning the atomic web services, we will use the reliability model developed in the previous section. Here, we will study the reliability modeling of the composite schemes. Concerning the reliability modeling for *SOA* [14], the architectural models are partitioned as follows: 1) path-based models, where the reliability of an assembly of components is calculated starting from the reliability of architectural paths; 2) state-based models, where the reliability is calculated starting from the reliability of system states and from the transition probabilities among states.

But once we start to examine a complex set of configurations, the number of states can increase sharply. To simplify the analysis and the required calculus, we previously developed a technique known as the bounded set method [21] for managing a large number of states.

So, our model will be a hybrid reliability model based on both path-based models (with the bounded set) and state-based models (doubly stochastic and renewal processes).

For the configuration of web services, we start by studying the *Serial-Parallel* (SP) configurations that focus on the redundancy technique which enhance the reliability value [18], [19], and [20]. The next section will explain the *bounded set* technique and after that we will develop the *SP* configurations with their corresponding failure probabilities based on the bounded set.

3.1 The Bounded Set Technique

The bounded set method involves a systematic decomposition into mutually exclusive sets of states and the development of upper and lower bounds for these sets. A classification of these sets of states into operational and nonoperational sets is carried out by the examination of the bounds of these sets.

According to the bounded set method, the general system state X can be written as a b-tuple min-term with x_j being a binary variable associated with the state of the component j in the system state

$$X = X_{A1}, X_{A2}, X_{A3}, \ldots, X_{AN}, \tag{18}$$

where X_A is an active component state with N components.

As we mentioned before, each of the system states X can be classified into mutually exclusive sets of system states. The A subsets consist of nonoverlapping sets of working states while the B subsets consist of nonoverlapping sets of states corresponding to system failure. In order to ensure mutual exclusiveness in this division, all states belonging to a given set must have a selected number of component states set to a fixed value, being 0 or 1. These selected components are not allowed to have any other value, within the given set. The remaining components may be either in 0 or 1 states and they are designated u denoting unselected state or "*don't care*" state.

All the system states in the set S can be defined as

$$\underline{X}^S \leq \overline{X}^S, \tag{19}$$

where $\underline{X}^S$ and $\overline{X}^S$ are the upper and lower *limiting states* of the set S, respectively. By *limiting states*, we mean the limit values of a state depending on its dimension. For example, in the case of two-state models (0: failure or 1: working), the upper limits will be 1 and the lower limits will be 0.

Let us define the following notations:

$$\begin{cases} J_S^1 & \text{set of all indices } j \text{ where } \overline{X}_j^S = \underline{X}_j^S = 1, \\ J_S^0 & \text{set of all indices } j \text{ where } \overline{X}_j^S = \underline{X}_j^S = 0, \\ J_S & \text{set of all indices } j \text{ where } \overline{X}_j^S = 1,\ \underline{X}_j^S = 0. \end{cases} \tag{20}$$

If we consider a system of n, two states devices were

$$\overline{p}_j = Pr\big(\text{j is in an operational state, i.e., } X_j^S = 1\big),$$
$$\overline{q}_j = Pr\big(\text{j is in a failure state, i.e., } X_j^S = 0\big).$$

Then, the probability of encountering any one of the system states in set S, or simply the probability of S occurring is

$$Pr(S) = \prod \overline{p}_i \prod \overline{q}_j, \tag{21}$$

where $i \in J_S^1$ and $j \in J_S^0$. Therefore, if the failure characteristics, $\overline{p}$ and $\overline{q}$, of every module are known, the probability of the system in any state can be easily evaluated using the equation shown for $Pr(S)$.

For a systematic decomposition, we can start decomposing from the most significant bit of the system state, i.e., we observe what happens when the first module is definitely working and $x_1 = 0$, when it is definitely failed. These two decomposed sets are mutually exclusive and together; they contain all the system states in the original set S_0, which is destroyed in the process of the decomposition. After the decomposition, the two subsets, S_1 and S_2, are examined. Examination of each subset S_1 involves examining the upper bounding state $\overline{X}_1^S$ with all the u states set to 1 and the lower bounding state $\underline{X}_1^S$ with all the u states set to 0. If $\overline{X}_1^S$ is a failed state, then the subset S_1 falls into the A subset. However, if $\overline{X}_1^S$ is a working state and $\underline{X}_1^S$ is a failed state, then S_1 is an unclassified set designated by U. In the above decomposition, if $\overline{X}_1^S$ (equal to $[011\ldots1]$) is a failed state, then S_1 falls into a B subset and it is not decomposed any further. For S_2, if $\overline{X}_2^S$ (equal to $[111\ldots\ 1]$) is a working state and $\underline{X}_2^S$ is a failed state, then S_2 is a U subset and it has to be decomposed further by elaborating the next significant bit in S_2 to give S_3 and S_4. After each decomposition, the two resulting subsets are examined as before. The decomposition process stops, once all subsets have been classified as A or B subsets. Note that in the assumed decomposition, the subsets A_1, A_2, B_1, and B_2 are mutually exclusive. Hence, the decomposition process results in the destruction of all unclassified or U subsets and their replacement with a mutually exclusive set of A and B subsets. By collecting all the sets that fall into the B subsets, we can determine the failure probability P_F of the systems as follows:

$$P_F = \sum Pr(B_i), \tag{22}$$

where B_i are the failed subsets.

Therefore, decomposing in this manner effectively contains the number of states which would, otherwise,

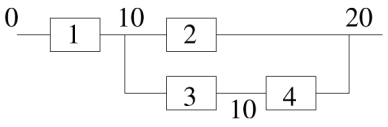

Fig. 8. Example of composite web services workflow graph.

become unmanageable as the number of modules increases. But with the bounded set techniques, we can fix a number corresponding to the maximum of failed components that we would like to reach and then consider each states subset containing more than this maximum value as *failed state B*. By doing that, we can limit the states explosion problem and obtain an approximation of the failure probability of the composite system.

In our previous work, we have extended this approach when examining fault-tolerant computer systems to deal with hot and cold start, components with different fault coverage and different failure distributions [22].

3.2 Serial-Parallel Systems Using the Bounded Set Technique

As their names indicate, the *Series-Parallel* web services' compositions are basically made by series configurations and parallel configurations. In the case of *Series* composite, the components are presented by a flow graph where each node of the graph models a stage of the execution that must be completed before transition to the next node. The other case of parallel composite is presented by a flow graph where only one of the web services must be completed before a transition to the next node can take place and this is the case of redundant components.

To fix the idea for the *bounded set* approach, we explain how to calculate the probability of failure in the case of *Serial-Parallel* system assuming that the failure probabilities of the different components are known.

Consider the composite web services presented in Fig. 8. Each node n of the graph corresponds to a web service. For each node n, we correspond the numbers i and j to fix the connections with other web services.

To calculate the failure probability of the whole system, the failure probabilities of each web service should be known. For simplicity, all the web services have the same failure probability $q_i = 0,2$. If we apply the bounded set decomposition developed in this section, the A_i and B_i sets are obtained as shown in Fig. 9. The initial set $[u\ u\ u\ u]$ corresponds to unclassified states for the web services $[1\ 2\ 3\ 4]$ of the graph.

The B subsets, corresponding to nonoverlapping sets of system failure, are (B_1, B_2, B_3):

$$\begin{aligned} P_F &= \sum Pr(B_i), i = 1, 2, 3 \\ &= Pr(B_1) + Pr(B_2) + Pr(B_3) \\ &= q_1 + p_1 q_2 q_3 + p_1 q_2 p_3 q_4 \\ &= 0.2576. \end{aligned} \tag{23}$$

As a first step, we consider the orchestration model for centralized coordination. So, the broker is introduced as a new component in the bounded set approach. We propose introducing fault tolerance for a fault or error at the web service level by using an adaption of the recovery block technique developed in the following section.

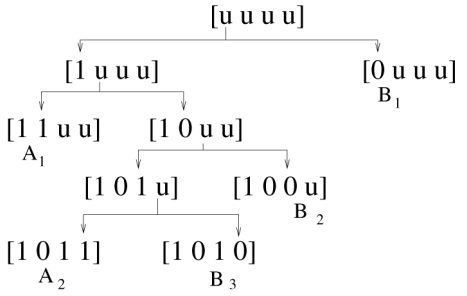

Fig. 9. Bsets for the example of composite web services graph.

4 Fault-Tolerance Study for Composite Web Services with Centralized Coordination Using Recovery Block Techniques

The QoS of an SOA is very important and by increasing the trust in the composite schemes, one may apply some tests on the results coming from different services to be sure of the validity of the data flow until the end.

In the case of central coordination, the broker is considered to have an acceptance testing mechanism that examines the results returned from a particular web service and if they are deemed to be acceptable proceeds with the computation to the next web service. If, however, it is unacceptable, it will involve rollback and invoke another component web service to carry out the same function as was required from the faulty component web service. In some senses, saving the state and creating checkpoints [24] is more straightforward, as they could be associated with initiating a web service and completion of a web service. It should be noted that the recovery block approach can address both design errors, hardware server errors and channel transmission errors. If it is anticipated that the second two causes which could involve transient errors were likely to be prevalent, then rollback and re-executing the component web service which gave the erroneous result would also be an option.

Our actual study will be focused on the orchestration model of centralized coordination and the reliability calculation will be done using the bounded set techniques for the *SP* systems. For the fault-tolerance study, we use an adaption of the recovery block technique that we previously used for computer programs in [31]. Reliability modeling of the above-mentioned schemes and techniques is an important step to understand the improvements in the reliability of the whole system. We should also be aware that the broker constitutes a single point of failure. One way of addressing this would be to provide redundancy at the broker using Triple Modular redundancy and N-Version Programming [25] to mask out errors at the broker level. The N-Version programming is introduced in Section 4.2 and the model that is used in our simulations is elucidated.

4.1 Recovery Block

Rollback recovery [32], [33] is a backward error recovery scheme to recover from transitory faults in computer systems. In real-time systems, it is crucial that recovery be possible within a specified time at any point in a program.

In previous work, Chandy and Ramamoorthy [32] gave a model for inserting rollback points in process control type programs. However, the checkpointing technique proposed in [32] is not suitable for systems where the parameters of the program are time dependent during the mission. Shambhu [34], [31], proposed a checkpointing scheme to handle dynamic variations in the parameters of a program.

4.1.1 Preliminaries and Problem Statement

In this model, the workflow is represented by a directed graph where node i represents service i and a directed edge (i, j) exists if service i is followed by service j with nonzero probability. By suitable preanalysis, an arbitrary workflow is transformed into a graph with no cycles leading to an acyclic graph.

For each node i of the workflow graph, a real number t_i is specified; t_i is the expected time of completion of service i. With each edge (i, j) of the workflow graph, a quantity $S(i, j)$—save time is associated, which is the time taken to save the state of the system (possibly in a secondary store) after service i is completed and when it is known that service j follows service i. Load time $L(i, j)$ associated with edge (i, j) is the time taken to load the state of the system at edge (i, j) to the main memory. The recovery time R, at node i, is the time required to load the saved state at the most recent rollback point and to rerun the program from this point to node i. If the first task executed after placing the most recent rollback point is denoted by service 1 and the next service by service 2 and so on, then $R_i = L(1, 2) + \sum_{s=1}^{i} r_s$ where r_s is the actual execution time of service s, $r_s \leq t_s$. Let service i be just completed and service j is to be processed next. At each edge (i, j), an interrogation is made to determine whether a rollback point is required in that edge. If a rollback point is required, then a copy of the system state is made. If a transient fault is detected before the creation of the next rollback point, recovery is initiated by restarting the program from the most recent rollback point.

The algorithm has the following inputs, constraints, and objectives:

- Inputs: t_i associated with each service i, $L(i, j)$, $S(i, j)$ associated with each edge (i, j) and a specified maximum recovery time M.
- Constraints: rollback points are inserted such that at every point in the program, the maximum possible recovery time does not exceed the specified maximum recovery time M.
- Objectives: minimize the maximum time that may be spent in saving the states of the system.

The problem of rollback point insertion in a dynamic environment is defined as follows: obtain an algorithm for rollback point insertion with the constraint that recovery should be possible within a specified time M, where M, t_i, $S(i, j)$, and $L(i, j)$ are all time-dependent quantities during the mission.

4.1.2 Shambhu J. Rollback Model [31]

Shambhu in his work dealt with program graphs which captured the properties of program execution at a relatively low level. In web services, the paths we are modeling essentially workflows. Thus, we will use elements of Shambhu's algorithm for workflow graphs to model these higher level workflows. The reason why we have chosen Shambhu's algorithm is that it allows for dynamic variation of the parameters which is the case in web services' compositions.

In order to present a simple rollback point insertion strategy for real-time systems with time-dependent parameters, we set one of the parameters $L(i, j)$ to a constant for all (i, j). This simplification might introduce some errors. However, these errors can be negligible.

For each pair of adjacent edges (i, j) and (j, k) joined at vertex j, a function $g(i, j, k)$ is defined as

$$g(i, j, k) = S(j, k) - S(i, j), i < j < k.$$

For an edge (i, j), a quantity $S(i, j)_{max}$ is defined as the maximum of all the save times associated with the edges in the paths from node j to n where n is the exit node. $S(i, j)_{max}$ is 0 if $j = n$. For an edge (i, j) of the workflow graph, a function $E(i, j)$ is defined as

$$E(i, j) = M - R_i - t_j,$$

where M is the maximum specified recovery time. The relevance of $E(i, j)$ in rollback point insertion can be intuitively explained as follows. Let service i be just completed and service j is to be processed next. Let (i, j) and (j, k) be adjacent. If $E(i, j)$ is found negative after the execution of service i, a rollback point must be inserted in edge (i, j) before the execution of service j so that the recovery time constraint is not violated. On the other hand, if $E(i, j)$ is positive, then based on the expected time of completion of service k, it will be possible to predict the requirement of a rollback point in edge (j, k) even before the execution of service j. This look-ahead feature constitutes the main part of the rollback point insertion algorithm leading to the minimization of system save times. A new function $D(i, k)$ should be defined as follows:

$$D(i, k) = E(i, j) - t_k.$$

Clearly, if $D(i, k) \geq 0$, then there is no need to insert rollback points at edge (i, j) or (j, k) which are two adjacent edges in a given workflow graph. At edge (i, j), the cumulative gain $c\_gain(j)$ is defined recursively as

$$c\_gain(j) = \begin{cases} c\_gain(i) - S(j, k)_{max} + g(i, j, k), & \text{if } A \geq 0, \\ c\_gain(i) - S(i, j)_{max} + g(i, j, k), & \text{if } A < 0, \end{cases}$$

where $A = D(i, k) + R_i$. When a workflow graph is entered (at the entry node), the cumulative gain is 0. In the above definition, $A \geq 0$ implies that no rollback point will be required in edge (j, k) if a rollback point is inserted in edge (i, j). On the other hand, if $A < 0$, insertion of a rollback point in edge (i, j) may not necessarily mean that a rollback point will not be required in edge (j, k). Accordingly, $S(i, j)_{max}$ should be used in computing $c - gain(j)$ instead of $S(j, k)_{max}$. In dynamic checkpointing, optimal insertion of

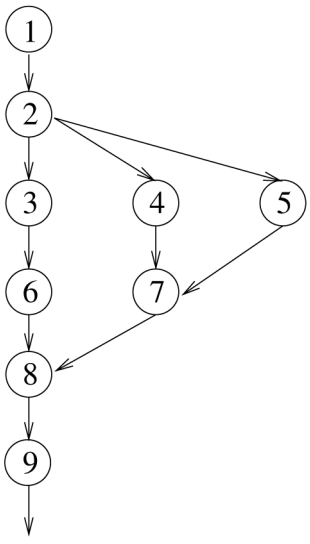

Fig. 10. Simulation example for the checkpoints choices.

rollback points is accomplished by judiciously placing the rollback points on the edges of the workflow graph based on certain criteria. The quantity $c\_gain(j)$ is an indicator of the gain in save time by shifting a rollback point from an edge to its adjacent edge. The criterion for the insertion of rollback points and the associated algorithm could be found in [31]. In Fig. 10, we present a simulation example for a tree structure graph which consists of nine nodes.

If we run the simulation for M specified as 25 and taking the values shown in the table in (24) for the running and saving times, the rollback points should be inserted in edges $(2,4)$ and $(7,8)$.

$$\begin{pmatrix} (i,j) & t_i & Sij & SijMax & r_i \\ (1,2) & 10 & 1 & 15 & 5 \\ (2,3) & 15 & 5 & 15 & 5 \\ (2,4) & 15 & 4 & 15 & 5 \\ (2,5) & 15 & 5 & 15 & 5 \\ (3,6) & 10 & 15 & 8 & 5 \\ (4,7) & 10 & 15 & 8 & 5 \\ (5,7) & 10 & 15 & 8 & 5 \\ (6,8) & 10 & 4 & 8 & 5 \\ (7,8) & 10 & 4 & 8 & 5 \\ (8,9) & 20 & 8 & 8 & 5 \\ (9,-) & 10 & 7 & 0 & 5 \end{pmatrix}. \quad (24)$$

The cumulative gain is 2 and the time spent in saving the states of the system is 9 units. In our case, this algorithm helps us choose the optimal checkpoint insertion for the brokers in the case of central coordination.

In order to increase the trust in the case of web services with dynamic parameters variation, we can use the Shambhu algorithm in an optimal way with respect to time boundary for rollbacks. We note that it is useful to use an acceptance test (AT) after the execution of each service which could be specified as examining a post-condition or invariant association with the service. However, it is not necessary to have a checkpoint and a saving of states after the execution. The insertion of checkpoints is determined by the above algorithm. If any acceptance test fails, one can rollback to the nearest previous checkpoint and re-execute the services. In this manner, it is possible to bound the time required by the fault-tolerant rollback mechanism. An alternative would be to do the acceptance tests at the checkpoints.

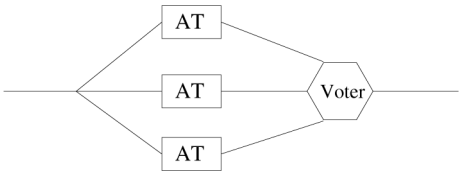

Fig. 11. Triple Modular Redundancy for the Acceptance test.

4.2 N-Version Programming and Modelization for the Broker

A component element in the bounded set approach introduced in Section 3 could be an individual web service or a broker. Recovery block techniques are introduced in Section 4.1 for faults or errors at the web service level. The broker (orchestration model) could have "built in" an acceptance testing mechanism to examine the results returned from a particular web service. This broker constitutes a single point of failure. One way of addressing this would be to provide redundancy at the broker using *Triple Modular Redundancy* and N-Version Programming [25] to mask out errors at the broker level.

Introducing more N-version points may increase the reliability; the problems related to the costs for additional versions will in turn limit the number of N-version points used. In the literature, many N-version structures can be found [16] with different complexity levels. In order to simplify the considered model, our study is limited to the simplest N-version type which is developed in the following.

For a primary module P, there are n versions P_i, $i = 1 \ldots n$, plus a module V to vote on their results. The probability of failure of each of the versions assuming no common mode error is given by P_{f_i}. The reliability of the system is

$$R = 1 - \sum_{k}^{f} C_n^k P_{f_i}^k (1 - P_{f_i})^{n-k}, \quad (25)$$

where $f(n) = integer(n/2 + 1)$. In deriving the reliability expressions, it has been assumed that

1. there are no common mode errors;
2. the N-versions have the same failure probability; and
3. the reliability of the voter is 100 percent.

The complexity study of the N-version programming versus the number of versions could be found in [16]. The version in our case of modeling consists of both broker software as well as the server hardware. Hence, each version of the broker software has to be run on a different server hardware to avoid server breakdown causing a single point of failure.

In our study, we chose for the *AT* the case of *Triple Modular Redundancy* which is the special case of N-version taking $n = 3$ as shown in Fig. 11.

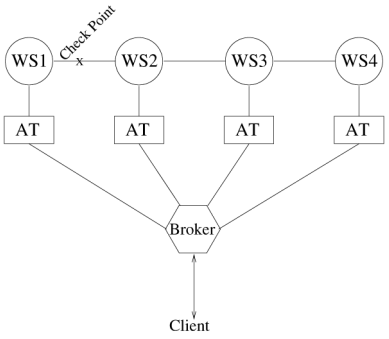

Fig. 12. Physical flow representation for web service.

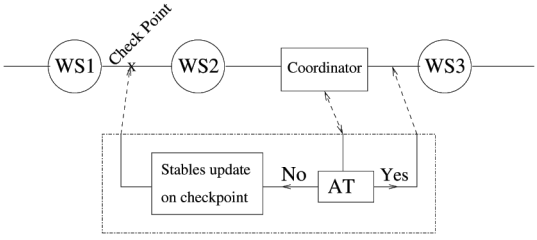

Fig. 13. Logical flow representation for web service.

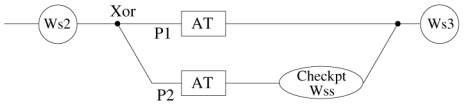

Fig. 14. Flow representation for the broker in a composite system.

Using (25) with $n = 3$, reliability of the *AT-TMR* system will be given by

$$R_{AT\,TMR} = 3R_{AT}^2 - 2R_{AT}^3, \tag{26}$$

where R_{AT} is the reliability of the *AT* ($R = 1 - P_f$).

In composite web services, different web services are not connected to each other; it is the broker that does the connection. The broker takes the results of a web service, applies the AT, and if it is valid, the broker transfers the results to the next web service. Otherwise, it does a rollback to the latest checkpoint, updates the saved states, and re-execute. This physical flow is represented in Fig. 12.

In Fig. 13, we present the corresponding logical flow where we insert the broker as a series component with the corresponding reliability.

In order to model the reliability of the broker, we assume that it is a sort of Xor structure [35] and [12] with two choices of probabilities as shown in Fig. 14.

These probabilities p_1 and p_2 should be proportional to the trust degree concerning the latest web service in the workflow preceding the *AT*. The reliability of the broker at this point will be given by

$$R_{broker} = p_1 R_{AT} + p_2 R_{AT} R_{WSpath}, \tag{27}$$

where R_{AT} is the reliability of the *AT* and R_{WSpath} is the reliability of the path beginning from the latest checkpoint until the current web service. In the case of *TMR*, R_{AT} should be replaced by $R_{AT\,TMR}$ given in (26).

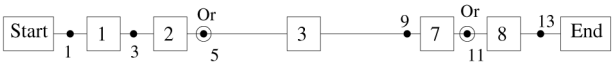

Fig. 15. Flowgraph of Series composite system.

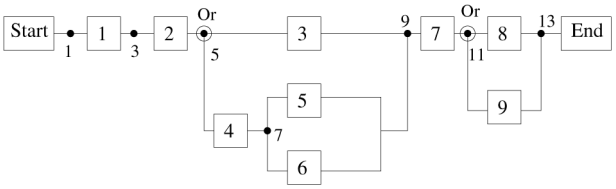

Fig. 16. Flowgraph of Series-Parallel composite system.

TABLE 1
Loading Variation with Time

time	500	3000	4000	7000	9000	12000	20000
a1-1	100	50	10	50	100	10	30
a1-2	100	10	10	50	10	100	30

5 Simulation Results

For the simulations, we consider the flow graphs presented in Fig. 15. At first, we compare the QoS between Series and Series-Parallel flowgraphs based on the exponential distribution for reliability defined previously in (1). If we take $\lambda = 2 \cdot 10^{-5}$ at time $t = 3 \cdot 10^3$s, P_f for the Series model is equal to 0.4984. In the case of Series-Parallel model, where we added some alternative services that can take the relay in the case of failure (redundancy) as shown in Fig. 16, P_f for the Series model is equal to 0.2157. We can see that the P_f is about the half value of the Series scheme.

Now concerning the bounded set technique, it gives the five following failure states (see Section 3.1) in the case of Fig. 15:

$$Bsets = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 0 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & -1 \end{pmatrix}. \tag{28}$$

In the case of Series-Parallel scheme in Fig. 16, the number of the Bsets is 26 and a part of these sets are given in the following:

$$Bsets = \begin{pmatrix} . & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & . \\ . & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & . \\ . & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & . \\ . & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & . \\ . & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & . \\ . & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & . \\ . & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & . \\ . & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & . \\ . & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & . \end{pmatrix}. \tag{29}$$

In the case of large systems, the number of states increases sharply and the BS techniques can be used to limit the computing complexity.

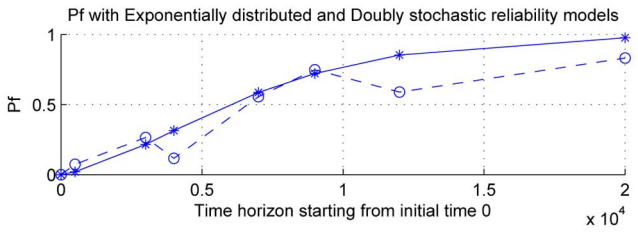

Fig. 17. Pf variation with time depending on Exponential distribution (-) and Doubly stochastics (- -).

To dynamically study composite schemes, we fix a time horizon for the study $T = 10^4$ s. We also take two cases of loading variations as a function of time following Table 1 where a_{1-1} and a_{1-2} are the number of events per time unit following (8) for two different servers. Different cases of mutually high and low loadings will be studied in the following in order to show their influence on the total reliability of the composition.

By running the Shambhu algorithm explained in Section 4.1, we obtain the optimal placements of the checkpoints where we should save the parameters for future rollback use. The chosen checkpoints taking $M = 2, 4 \cdot 10^2$ are $[5, 9]$ in Fig. 16.

In Fig. 17, we show the Pf variation with time depending on Exponential distribution and Doubly stochastics. We can see that it is not exponential and that it is more realistic and depends on different loading variations during the time horizon of the study following Table 1.

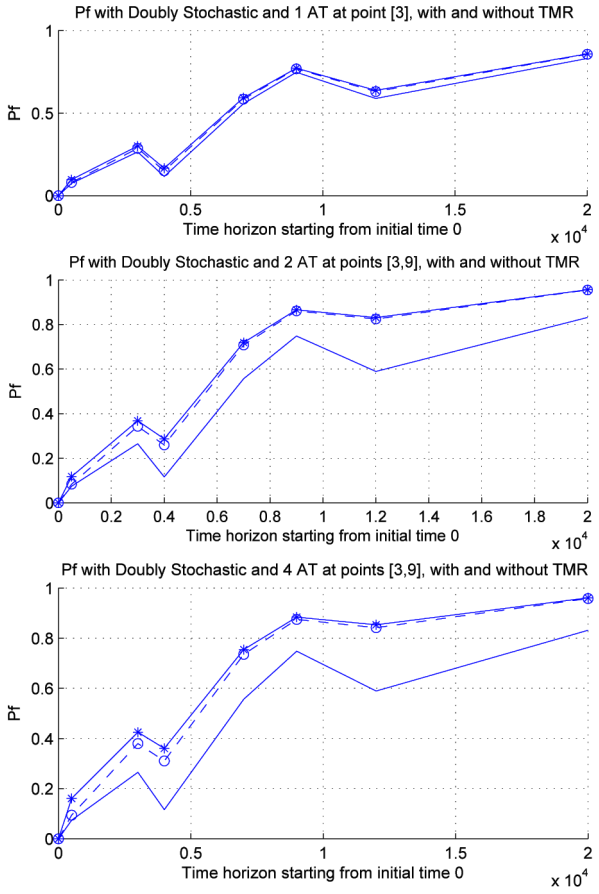

Fig. 18. Variation of the Pf with different schemes: (-) no AT, (- *) with n AT, (- - o) with n AT and TMR.

TABLE 2
Number of Bounded Set Failure States

Number of AT	0	1	2	4
Bounded Set States	26	27	38	50

The Reliability of the acceptance test is equal to 0.98. In Fig. 18, we show the Pf variation with different AT positions and Triple Modular Redundancy for the AT. We considered three different cases:

- Case 1: One AT at point 3 in Fig. 16.
- Case 2: Two AT at points 3 and 9 in Fig. 16.
- Case 4: Four AT at points 3, 5, 9, and 11 in Fig. 16.

We can see that if more AT are added, the Pf is increased. So, one should choose wisely the number and the positions of the different AT in a certain composition depending on the QoS of the atomic web services. We also see that the TMR decreases the Pf making the system more reliable. In Table 2, we show the increasing number of states for different schemes of AT.

6 Conclusion

In this paper, we developed a Service-Oriented reliability model to enhance the QoS of web services' compositions. The model dynamically calculates the reliability of composite web services with rollback recovery based on the real-time reliabilities (depending on the servers loading) of the atomic web services of the composition.

Our model is a hybrid reliability model based on both path-based and state-based models. By using the web services estimations of the failure rates based on the doubly stochastic and renewal processes, we determine the reliability of composite web services using the bounded set approach to manage the states explosion in the case of complex systems. The architecture was mainly based on Series-Parallel models to focus on the redundancy technique. Next, we developed an approach to fault tolerance to provide dependability in the event of component web service failure through rollback and recovery algorithm for web services. This will increase the QoS by enhancing the trust in the results (due to the AT) given by the composite. Different cases were studied and compared using the failure probabilities, the cost, the execution time in different cases: web services, web services with AT and recovery block (with and without TMR), and different AT insertion schemes. In future work, we intend to extend our model to more complicated cases including the AND, Xor, and loop structures dealing with three-state models for the bounded set due to the presence of the Xor structures.

Acknowledgments

The authors would like to express their thanks to the associate editor and the three unknown reviewers for their constructive and useful comments. This work was done while Dr. H. Elfawal Mansour was working with Professor Dillon at the DEBII Institute, Australia.

REFERENCES

[1] H.K.G. Alonso, F. Casati, and V. Machiraju, *Web Services: Concepts, Architectures and Applications.* Springer-Verlag, 2004.

[2] H.K.N. Shadbolt, W. Halt, and T. Berners-Lee, "Web Services: Concepts, Architectures and Applications," *IEEE Intelligent Systems,* vol. 21, no. 3, pp. 96-101, 2006.

[3] T.S. Dillon, W. Chen, and E. Chang, "Grid Space: Semantic Grid Services on the Web—Evolution towards a Soft Grid," *Proc. IEEE Semantics, Knowledge and Grid Conf.,* 2007.

[4] T.S. Dillon, W. Chen, and E. Chang, "Reference Architectural Styles for Service-Oriented Computing," *Proc. IFIP Network and Parallel Conf.,* 2008.

[5] W3C, SOAP Specifications, http://www.w3.org/tr/soap, 2011.

[6] Business Process Execution Language for Web Services Version 1.1, http://www-128.ibm.com/developerworks/library/specification/ws-bpel, 2011.

[7] Semantic Web Services Language Requirements Version 1, http://www.daml.org/services/swsl/requirements/swsl-requirements.shtml, 2011.

[8] W3C, Web Services Choreography Working Group, http://www.w3.org/2002/ws/chor, 2011.

[9] T. Mikalsen, S. Tai, and E. Wholstadter, "Transaction Policies for Service-Oriented Computing," *J. Data and Knowledge Eng.,* vol. 51, pp. 59-79, Oct. 2004.

[10] Hitachi, WS-Reliability, http://www.hitachi.co.jp/Prod/comp/soft1/wsrm, 2011.

[11] A.S.J. Cardoso, J. Miller, and J. Arnold, "Quality for Service for Workflows and Web Services Processes," technical report, LSDIS Lab, Computer Science Dept., Univ. of Georgia, 2002.

[12] D. Zhang, W.T. Tsai, and Y. Chen, "A Software Reliability Model for Web Services," *Proc. Eighth Latest Conf. and Applications,* pp. 144-149, 2004.

[13] L. Zhang and J. Zhang, "Criteria Analysis and Validation of the Reliability of Web Services Oriented Systems," *Proc. IEEE Conf. Web Services (ICWS '05),* pp. 621-628, 2005.

[14] V. Cortellessa and V. Grassi, *Reliability Modeling and Analysis of Service-Oriented Architectures.* Springer, 2007.

[15] B. Kiepuszewski, A.H.M. ter Hofstede, and W. van der Aalst, "Fundamentals of Control Flow in Workflows," *Acta Informatica,* vol. 39, pp. 143-209, 2003.

[16] K. Lew, "Software Design and Architectural Aspects of Reliable Computer Systems," PhD dissertation, Monach Univ., 1986.

[17] F. Leyman, "Workflow-Based Cooperation and Coordination in a Service World," CoopIS '06 keynote speech, Montpellier, France, http://www.cs.rmit.edu.au/fedconf/2006/index.html, 2006.

[18] Reliability of Series-Parallel Systems, http://www.mathpages.com/home/kmath560/kmath560.htm, 2011.

[19] W. Abramowicz, M. Kaczmarek, and D. Zyskowski, "Duality in Web Services Reliability," *Proc. Advanced Int'l Conf. Telecomm. and Int'l Conf. Internet and Web Applications and Services (AICT/ICIW),* 2006.

[20] Y. Pan, "Will Reliability Kill the Web Service Composition?" technical report, Dept. of Computer Science, Rutgers Univ., 2009.

[21] Y. Yak, "Reliability Modeling of Fault Tolerant Computer Systems," PhD dissertation, Univ. of Sydney, 1985.

[22] T. Dillon, Y.W. Yak, and K. Forward, "The Effect of Incomplete and Deleterious Periodic Maintenance on Fault-Tolerant Computing Systems," *IEEE Trans. Reliability,* vol. 35, no. 1, pp. 85-90, Apr. 1986.

[23] T. Dillon, Y.W. Yak, and K. Forward, "Incorporation of Reconfiguration and Repair Time in the Reliability Modelling of Fault Tolerant Systems," *Proc. IEE/IFAC Int'l Conf. Computer Safety, Reliability, and Security (Safecomp '83),* 1983.

[24] J. Cao and T. Dillon, "Checkpointing and Rollback of Wide Area Distributed Applications Using Mobile Agents," *Proc. 15th Int'l Parallel and Distributed Processing Symp. (IPDPS '01),* 2001.

[25] L. Chen and A. Avizieni, "N-Version Programming: A Fault Tolerant Approach to Reliability of Software Operation," *Proc. IEEE Int'l Symp. Fault-Tolerant Computing (FTCS-8),* pp. 3-9, 1978.

[26] R. Iyer and P. Velardi, "A Statistical Study of Hardware Related Software Errors in MVS," *Proc. IEEE Int'l Symp. Fault-Tolerant Computing (FTCS '84),* pp. 192-197, 1984.

[27] D. Gaver, "Random Hazards in Reliability Problems," *Technometrics,* vol. 5, pp. 211-226, 1963.

[28] A. Birolini, "Some Applications of Regenerative Stochastic Processes to Reliability Theory Part 1," *IEEE Trans. Reliability,* vol. 23, no. 3, pp. 186-194, Aug. 1974.

[29] A. Mansour, *Probabilites et Statistiques Pour Les Ingenieurs.* Hermes, Lavoisier, 2007.

[30] J.L. Romeu, "Understanding Series and Parallel Systems Reliability," Reliability Information Analysis Center (RIAC), http://www.theriac.org/pdfs/startsheets/s&psysrel.pdf, 2011.

[31] J.S. Uphadyaya, "A Study of Rollback Recovery Techniques: Hardware, Models, Algorithms and Evaluation," PhD dissertation, Newcastle Univ., 1985.

[32] K. Chandy and C. Ramamoorthy, "Rollback and Recovery Strategies for Computer Programs," *IEEE Trans. Computers,* vol. 21, no. 6, pp. 546-556, June 2007.

[33] S. Upadawa and K. Saluja, "A Watchdog Processor Based General Rollback Technique with Multiple Retries," *IEEE Trans. Software Eng.,* vol. 12, no. 1, pp. 87-95, Jan. 1986.

[34] J. Shambhu, "Rollback Recovery in Real-Time Systems with Dynamic Constraints," *Proc. IEEE Ann. Int'l Computer Software and Applications Conf.,* 1990.

[35] QCWS: QoS Web Service Composition, http://link.ece.uci.edu/QCWS/qcws_main.html, 2011.

H. Elfawal Mansour received the MS degree in industrial computer sciences in September 1995 from the University of Joseph Fourier, France. She received the MSc and the PhD degrees in control theory from the Institute National Politechnique de Grenoble (INPG), France, in 1996 and 1999, respectively. In 2000, she was a postdoctoral researcher at Bio Memitic Research Centre, Nagoya, Japan. She holds the post of research fellow at the Digital Ecosystems and Business Intelligence Institute (DEBII), which is a research center of the Curtin University of Technology. Her main research expertise includes control theory, computer engineering, optimal control, reliability, and web servers.

T. Dillon is internationally recognized for his research on semantic web, web services, knowledge discovery and data mining, neural networks, intelligent systems, object-oriented systems, communications, fault-tolerant systems, and distributed protocol engineering. He is the head of the IFIP International Task Force WG2.12/24 on Semantic Web and Web Semantics and the IEEE/IES Technical Committee on Industrial Informatics. He has published 12 books and 650 research papers in book chapters, journals, and international conferences. His research has received more than 2,500 citations with a Hurst index of 24 (Google scholar). His research interests include semantic web, ontologies, XML systems, bioinformatics, expert and intelligent systems; knowledge discovery and data mining including association rules for complex data structures and XML documents; automated knowledge acquisition using neural networks and decision tree methods with the Zhou/Dillon Symmetrical Tau for feature selection; validation and verification of knowledge base; timed intelligent systems; hybrid expert and neural systems; fuzzy reasoning including case based approaches; object-oriented computing including conceptual modeling with object-oriented approaches; theoretical foundations of object-oriented systems; object-oriented databases and software engineering; neural networks including adaptive neural networks; addition of domain knowledge to nets and proof of convergence of both supervised and unsupervised nets; computer communications such as formal description, verification, specification, and conformance testing of OSI protocols; fault-tolerant computing including the development of mathematical modeling of tolerant computing methods; development of new architectures for fault-tolerant computing; and the development of strategies for detection and modeling of computer viruses. His research has made significant contributions to a number of application areas including logistics, banking and finance, electrical power systems, and telecommunications and management. He is a fellow of the IEEE.