THE INFLUENCE OF SERVICE INTERACTIONS ON INDIVIDUAL SERVICE RELIABILITY IN A COMPOSITION SCENARIO

Abhishek Srivastava and Paul G. Sorenson
Department of Computing Science, University of Alberta, Edmonton, Canada

Keywords: Composite service reliability, Service selection.

Abstract: Selecting an optimal service from a group of functionally equivalent ones is non-trivial. This is more so when the service to be selected is part of a composite application. Research in the past has resolved this issue making use of the Quality of Service (QoS) attributes of the services to determine the most optimal from the functionally equivalents. This paper too attempts to tackle this problem using one of the more important QoS attributes, reliability. The novelty of the technique proposed here is due to the fact that while papers in the past have looked upon the reliability of individual services in a service composition in isolation, we take into account the influence that the interaction among services in a composition has on individual reliabilities. The service domain along with the interactions is represented as a continuous time Markov chain, and through appropriate procedure the reliability of individual services is calculated in the form of 'failure distance'. The services selected are the ones with the largest values of failure distance. The results of experiments conducted by us have also been included to validate this technique.

1 INTRODUCTION

The world today is undergoing an economic revolution wherein service dominated economies are fast replacing product based manufacturing economies (Battilani and Fauri, 2007)(Kilburn, 2003). Organizations, in such a scenario, are more inclined towards offering functionalities as services rather than complete end-products, which the clients may harness according to their customized respective requirements. The extensive penetration of the 'internet' in the daily lives of people figures prominently in the acceptance and adoption of the service-oriented model. Service vendors readily make use of this medium, and conveniently offer their expertise as web-based services.

A high-level view of the generic service-oriented architecture (SOA) comprises three main players: the service-requester, the service-provider, and the service-registry, as shown in Figure 1 (Papazoglou, 2003).

The service-requester may be a standalone client or may itself be a service forming part of a mesh of composite services. The service-requester looks up a *service-registry* which is a directory of services where prospective service-providers *publish* their identities, whereabouts, and capabilities with the hope of being 'discovered' by prospective clients. The service-requester, selects a service from this registry which most closely matches its requirements. The requester then "loosely" binds with the corresponding service-provider and makes use of the functionality offered by the selected service. A more detailed exposition on this generic view of SOA may be found in (Papazoglou, 2003).

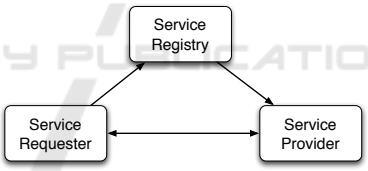

Figure 1: The basic model of an SOA.

Selecting the most optimal service from the service registry is however non-trivial. The criteria for service selection comprise the list of functionalities specified by the service requester. Often, however, more than one service in the registry cater to these requirements. The issue then is picking out *the* best service from these functionally equivalent services.

Effort in the past has been on selecting the 'best' service from the functionally equivalent ones on the basis of various Quality of Service (QoS) parameters

Srivastava A. and G. Sorenson P.
THE INFLUENCE OF SERVICE INTERACTIONS ON INDIVIDUAL SERVICE RELIABILITY IN A COMPOSITION SCENARIO.
DOI: 10.5220/0002792300300039
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

(Ran, 2003) (Kokash, 2005). The service whose QoS parameters are the best among the lot is selected.

In this paper, we propose a technique to select the 'best' among functionally equivalent services on the basis of one of the QoS parameters, reliability. We claim that our technique is novel because it calculates the reliability of individual services taking into account the influence that the presence of other services in the domain, particularly the services that it directly invokes, have on it. By this, we mean the varying degrees of coupling that services have on each other in an interactive environment lends its influence on the reliability figures of individual services. Suppose for example two services *A* and *B* are provided by the same organization and thus *A* invokes *B* much more than another service offering the same functionality as *B* (thus *A* and *B* have a high value of coupling). Further suppose that *B* is a highly unreliable service. As *A* uses *B* so often, it would appear to *A*'s parent that *A* is quite unreliabile. The reliability of services in an environment in which the influence of interacting services is taken into account forms the basis of the technique proposed in this paper. Previous methods tend to ignore this influence and calculate reliability by treating the individual services in isolation (Yu and Lin, 2004) (Tsai et al., 2004) (Wang et al., 2003) (Zeng et al., 2003).

The remainder of this paper is structured as follows. Section 2 is a discussion on relevant related work done in the past. Section 3 is a discussion on how the service domain is represented in our work. Subsequently, the representation of this service domain as a continuous time Markov chain (CTMC) (Norris, 1998)(the CTMC representation is a requirement for the proposed technique) is also discussed in the same section. Section 4 comprises a detailed description of the use of the CTMC representation to express how 'close' or 'far' each individual service is from the 'failure' state taking into account its interactions with other services in the domain. Section 5 describes the experiments conducted by us and the results of the same, that validate the technique, and finally Section 6 concludes the paper with pointers at future work.

2 RELATED WORK

Substantial research has been done in the past on the issue of computing the reliability of a composite application on the basis of the individual reliability of its constituent services. The technique discussed by Yu *et al.* is for QoS in general and reliability is one among a number of parameters (Yu and Lin, 2004). Their approach is quite basic. They assume that the reliability values of individual services are given and the reliability of the composite application is simply the product of the reliabilities of individual constituent services. Tsai *et al.* present a method of first calculating the reliability of individual services using a technique called 'group testing' (Tsai et al., 2004). In this technique, the service in question is put through the same test as a number of other functionally equivalent services of known reliabilities. The other services are assigned appropriate weights on the basis of their reliability values. Depending on how close or far the result of the service in question is from that of the other services, the service is assigned a weight which is an approximate reflection of its reliability. Next, using the reliability of individual services so calculated, the reliability of the composite application is calculated using one of various simple formulae depending upon how the individual services are connected in the application: as a 'sequence', 'choice', 'loop' or 'concurrently'. Zeng *et al.*, in their approach represent the various possible composite applications as a directed acyclic graph (DAG) (Zeng et al., 2003). In this DAG, they find a 'critical path'. The reliability of the composite application is a product of $e^{rel(s_i)*z_i}$ where $rel(s_i)$ is the reliability of service i and $z_i = 1$ if service i falls in the critical path, and $z_i = 0$ otherwise. The reliabilities of individual services are calculated using historical data as the ratio of successful service delivery to total number of invocations. Wang *et al.* present an interesting technique for calculating the reliability of the service composition (Wang et al., 2003). Their method assumes that the transition behaviour of the system from one service to another follows the Markov process (Norris, 1998). They express a 'reliable' transfer from each service to every other service in the form of a matrix. The reliability of a service composition is expressed by raising this matrix to a power that is equal to the number of transition steps required to move from the initial service to the final service of the composition. Subsequently, the matrix is raised to all possible powers from 0 to ∞, representing all possible service compositions and the summation of this geometric progression gives the reliability of the composition. More recently Epifani *et al.* present a dynamic modeling technique wherein Bayesian estimation techniques are used to dynamically revise the non-functional attributes of the constituent services of a composite application (Epifani et al., 2009). The posterior distribution of non-functional attributes such as reliability are determined on the basis of prior distribution and a better service composition is achieved.

All these methods, while being relevant suffer from the assumption that the reliability of individual services in a service domain is calculated in isolation. We feel that in an environment like this where services interact in a complementary manner to form composite applications, a certain degree of 'coupling' among services does exist. This coupling may arise out of business relationships, better 'inter usage' capability, or other miscellaneous factors. In our work, we assign a value to the coupling between interacting services. The higher this value, the greater is the coupling. The reliability calculation of individual services is therefore done incorporating these coupling values. Subsequently, the services with the best individual values of reliability, so calculated, collectively contribute to the best reliability value for the composite application.

3 SERVICE DOMAIN REPRESENTATION

In this work, we look at the service domain as a group of services which are assumed to have been already discovered. A set of services from this domain need to be selected and loosely 'chained' together to form a composite application. A common example of a composite application is a 'trip planner' whose goal is to plan a trip for prospective clients. The constituent services in this case may be: a) a flight reservation service, b) a hotel booking service, c) a payment processing service, etc.

The representation of the service domain in our work has been done as a *multi-tier acyclic graph*, where each tier represents a certain functionality and is populated by the set of simple services that possess the capability to perform the respective functionality. The size of this set is dynamic in nature and is characterized by frequent entry of new service instances, as well as upgrade and departure of existing ones. The left panel of Figure 2 shows a simple representation of this. Services B, C, that are at the same level, represent functionally equivalent services. The same holds for services D, E, F, as well as G, H. The work-flow is from top to bottom, never from bottom to top and hence there are no cycles. Thus a service at a certain level 'invokes' one of the services at the level immediately below it, and subsequently this invoked service invokes one of the services from the next level, and so on. In this way, a composite application gets formed. It should be noted that a service invokes one at the lower level only *after* it has completed executing its own task. An example of this service domain representation (the 'trip-planner' example) is shown on the right panel of Figure 2.

A variable called 'coupling', as mentioned earlier, is assigned between each service at a certain level to each and every service at the next immediate level. The coupling C_{ij}, depends upon factors such as business relationships between the providers of services i, and j, degree of ease in inter usage between the two services, and other miscellaneous factors. A high value of C_{ij} indicates a high degree of coupling between i, and j. For example, assuming the services in our trip-planner example, 'TravelPedia' and 'HoteloCity' are provided by the same parent organization. It would be expected therefore that there would be a high value of coupling between the two. The value of coupling between services ranges from 1 to ∞. A value of 1 indicates ordinary interaction with no special bonding between the services, whereas higher values indicate better bonding. The coupling values are assigned when a service enters a domain, on the basis of expert judgement after a thorough investigation of its nature and interactions.

For each service in the domain, there is a service completion rate which is the rate at which the respective service completes the requests. The service completion rate of service i, μ_i may be calculated as follows:

$$\mu_i = \frac{\textit{no. of service requests completed by service } i}{\textit{total time}} \tag{1}$$

Subsequently, a transition rate (λ_{ij}) between a calling service i, and the called service j may be calculated as:

$$\lambda_{ij} = C_{ij} \cdot \mu_i \tag{2}$$

where C_{ij} is the coupling between services i, and j, and μ_i is the service completion rate of service i. The transition rates are indicated by dotted arrows between services in the left panel of Figure 2.

Besides these transition rate values, there is a 'failure rate' ($\lambda_{i \to fail}$) from each service i in the domain to the 'fail' state. The independent failure rate of each service i may be expressed as follows,

$$\lambda_{i \to fail} = \frac{\textit{no. of times service } i \textit{ fails}}{\textit{total up-time}} \tag{3}$$

There is correspondingly a 'recovery rate' ($\lambda_{fail \to i}$) from the fail state to each service i in the domain. The recovery rate may be calculated as follows,

$$\lambda_{fail \to i} = \frac{\textit{no. of times service } i \textit{ recovers from failure}}{\textit{total down-time}} \tag{4}$$

The motive behind representing the service domain as a CTMC is to take advantage of the various analysis procedures meant for the latter, and in effect

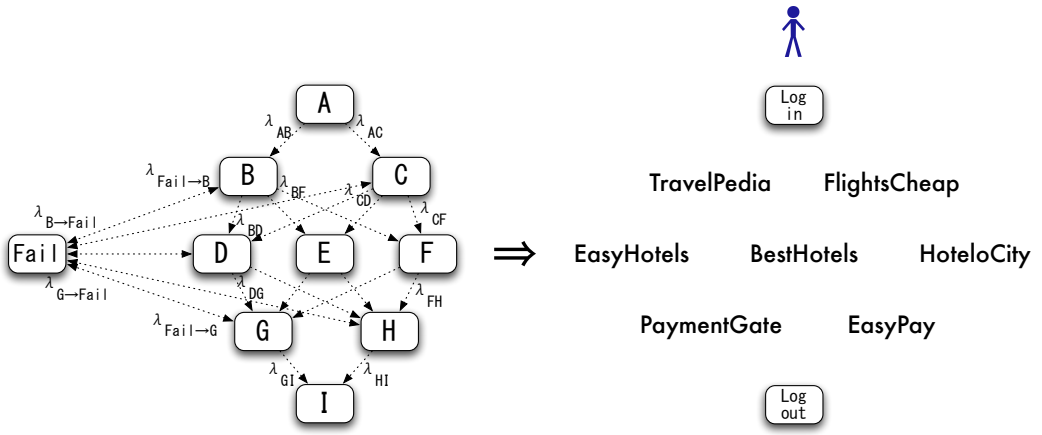

Figure 2: The service domain.

to come up with useful results regarding the reliability of the system.

	Log in/out	TravelPedia	FlightsCheap	EasyHotels	BestHotels	HoteloCity	PaymentGate	EasyPay
Log in/out	--	1.0	1.0	--	--	--	--	--
TravelPedia	--	--	--	1.4	1.7	1.5	--	--
FlightsCheap	--	--	--	1.1	2.1	1.9	--	--
EasyHotels	--	--	--	--	--	--	1.4	2.7
BestHotels	--	--	--	--	--	--	2.2	1.9
HoteloCity	--	--	--	--	--	--	1.2	1.7
PaymentGate	1.0	--	--	--	--	--	--	--
EasyPay	1.0	--	--	--	--	--	--	--

Figure 3: Coupling values.

	Failure rate	Recovery rate	Completion rate
Log in/Log out	--	--	1.0
TravelPedia	0.2	0.7	1.7
FlightsCheap	0.08	0.5	2.4
EasyHotels	0.05	0.009	1.4
BestHotels	0.3	0.002	3.2
HoteloCity	0.08	1.1	1.8
PaymentGate	0.2	0.06	0.8
EasyPay	0.3	0.7	1.4

Figure 4: Initial failure, recovery, and service completion rate values.

The transition rates between states in the CTMC may be expressed in the form of a matrix called the *infinitesimal generator matrix* (IGM). The value of the ij^{th} element of this matrix is equal to the value of the rate of transition from the i^{th} state to the j^{th} state of the CTMC. The sum of the elements in each row of this matrix is always zero. Therefore the ii^{th} element in each row is the negative of the summation of the other elements in the row.

The transition rates in the case of our service domain are the transition-rate values between services calculated using equation (2), and the transition-rate values between the services and the 'fail' state are calculated using equations (3) and (4). The infinitesimal generator matrix for our trip-planner example whose coupling values between services and service completion rate values are shown in Figures 3 and 4 is shown in Figure 5.

$$= \begin{bmatrix} - & . & . & . & . & . & . & . & . \\ . & - & . & . & . & . & . & . & . \\ . & . & - & . & . & . & . & . & . \\ . & . & . & - & . & . & . & . & . \\ . & . & . & . & - & . & . & \mathbf{6.08} & . \\ . & . & . & . & . & - & . & . & . \\ . & . & . & . & . & . & - & . & . \\ . & . & . & . & . & . & . & - & . \\ . & . & . & . & . & . & . & . & - \end{bmatrix}$$

Figure 5: Infinitesimal Generator Matrix.

For example, the transition rate value from service *BestHotels* to *EasyPay* is calculated using equation (2) as follows (shown in bold in the matrix):

$$\lambda = Coupling_{BestHotels\text{-}EasyPay} * Completion\text{-}rate_{BestHotels}$$
$$= 1.9 * 3.2 = 6.08$$

The last column and row ($9th$ from left and top) of the matrix correspond to the fail state, and its elements are simply obtained from the table in Figure 4 as the initial failure and recovery rate values respectively of each service.

A number of analysis techniques require that the transition values between the different states of a CTMC be expressed as a probability. To take advantage of these techniques, an approximate transition

probability matrix may be obtained from the infinitesimal generator matrix, which is called the *embedded Markov chain matrix*. Each ij^{th} element of this matrix may be obtained as the ratio of the corresponding element of the infinitesimal generator matrix to the sum of all the elements except the ii^{th} element in the corresponding row of the infinitesimal generator matrix. The ii^{th} element of the embedded Markov chain matrix is always zero. The embedded Markov chain matrix P for our trip-planner example corresponding to the infinitesimal generator matrix is shown in Figure 6.

Figure 6: Embedded Markov Chain Matrix.

As an example, the element in the embedded Markov chain matrix corresponding to the element in bold in the IGM is calculated as follows:

$$P = \frac{6.08}{7.04 + 6.08 + 0.3} = 0.453$$

4 RELIABILITY IN TERMS OF 'FAILURE DISTANCE'

The CTMC representation of the service domain is utilized to calculate the reliability values. More precisely, the CTMC representation is used to find the 'failure distance' of individual services, and this failure distance is regarded as an expression of reliability. The larger the failure distance of a service, the higher is its reliability.

It would be worthwhile to mention at this point that the failure distance of a service is measured as the 'number of transitions' that the system would need to go through before converging to the 'fail' state, given that it starts at the service in question. The larger the number of transitions, the larger the failure distance. This would be further elaborated upon, in the subsequent portion.

The CTMC representation of the service domain which is used to calculate the failure distance includes the infinitesimal generator matrix (shown in Figure 5 for our example) which consists of the transition rate values between every pair of 'invoking' and 'invoked' services in the domain. Another important component of the CTMC representation is the *probability vector*. The elements in the probability vector correspond to the various states of the CTMC representation. Each element in the vector represents the probability that the system exists in the corresponding state at a certain stage. Thus initially, when number of transitions, $t = 0$, the probability vector may be represented as π_0, where

$$\pi_0 = \{p_1^0, p_2^0, p_3^0, \ldots, p_n^0\} \quad (5)$$

$p_i^0 (i = 1, 2, \ldots, n)$ are the values representing the probability that the system initially is in state i. Furthermore, after the first transition, $t = 1$,

$$\pi_1 = \pi_0 \cdot P \quad (6)$$

where P is the embedded Markov chain matrix (shown in Figure 6 for our example). As mentioned earlier, the embedded Markov chain matrix consists of the transition probability values from each state in the system to every other state. It is analogous to the transition probability matrix of a discrete time Markov chain (Norris, 1998).

Thus, if we continue multiplying the probability vector with the embedded Markov chain matrix P, we will eventually arrive at a probability vector, π_s, which remains constant, *i.e.* it does not change with further multiplication with P.

$$\pi_1 = \pi_0 \cdot P \Rightarrow \pi_2 = \pi_1 \cdot P \Rightarrow \cdots \pi_s = \pi_s \cdot P \quad (7)$$

π_s is called the *equilibrium probability vector*. Each element in this vector represents the probability of the system being in the corresponding state at equilibrium.

We will now show that the equilibrium probability vector is the same as the left-hand eigenvector of matrix P corresponding to the unit eigenvalue (Greub, 1981). If e_i^T is the left-hand eigenvector of a matrix P corresponding to the eigenvalue λ_i, then we know that

$$\lambda_i \cdot e_i^T = e_i^T \cdot P \quad (8)$$

when $\lambda_i = 1$, *i.e.* the unit eigenvalue, then

$$e_i^T = e_i^T \cdot P \quad (9)$$

Observing equations (7) and (9) together,

$$e_i^T = e_i^T \cdot P \Leftrightarrow \pi_s = \pi_s \cdot P \Rightarrow e_i^T = \pi_s$$

The equilibrium probability vector may thus be easily calculated as the left hand eigenvector (normalized to sum to 1) corresponding to the unit eigenvalue, of the embedded Markov chain matrix of the CTMC representation of any service domain. The equilibrium probability vector for our trip-planner example is shown below:

$$[0.2092, 0.1173, 0.1137, 0.0593, 0.0888, 0.0991, \ldots$$
$$\ldots 0.1083, 0.1488, \mathbf{0.0555}] \quad (10)$$

Of the elements of this equilibrium probability vector, the one corresponding to the 'fail' state (shown in bold) of the CTMC representation, gives the probability of the system being in the fail state at equilibrium. If the probability of the system being in the fail state at equilibrium is high, we may conclude that the service that is far from equilibrium is also far from the fail state, and vice-versa. Therefore, if we somehow find a way of calculating the distance of a service from equilibrium, it would also give us an estimate of its distance from failure. Thus the problem of finding the distance of a service from failure may be translated to that of finding the distance of the service from equilibrium.

We utilize the method put forward by William J. Stewart to calculate the distance of the service from equilibrium at the various stages of service selection (Stewart, 1991). The method is explained as follows.

Let x_1^0 represent the probability vector that models a system that has a 100% probability of being in state 1 initially when the number of transitions $t = 0$,

$$x_1^0 = \{1, 0, 0, \ldots, 0\} \quad (11)$$

similarly,

$$x_2^0 = \{0, 1, 0, \ldots, 0\} \quad (12)$$

and in general,

$$x_i^0 = \{0, 0, 0, \ldots, 0, 1, 0, \ldots, 0\} \quad (13)$$

Let the left-hand eigenvectors of matrix P be

$$\{e_1^T, e_2^T, \ldots, e_n^T\}$$

corresponding respectively to the eigenvalues,

$$\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

Since x_i^0 in equation (13) is a row vector, it may be expressed as a linear combination of other row vectors. Thus,

$$x_i^0 = c_{i1} \cdot e_1^T + c_{i2} \cdot e_2^T + \ldots + c_{in} \cdot e_n^T \quad (14)$$

where $c_{ij} (j = 1, 2, \ldots, n)$ are currently unknown constants whose values and significance will subsequently be discussed.

Just like π_1 was computed in equation (6), x_i^1 which is the probability vector after the first transition $t = 1$ given that the system starts (at $t = 0$) with a 100% probability of being in state i, may also be computed as,

$$\text{equation (6): } \pi_1 = \pi_0 \cdot P \Rightarrow x_i^1 = x_i^0 \cdot P$$

Thus, from equation (14), it follows that,

$$x_i^1 = x_i^0 \cdot P = c_{i1} \cdot e_1^T \cdot P + c_{i2} \cdot e_2^T \cdot P + \ldots + c_{in} \cdot e_n^T \cdot P \quad (15)$$

Using equation (8): $\lambda_i \cdot e_i^T = e_i^T \cdot P$, we get,

$$x_i^1 = c_{i1} \cdot \lambda_1 \cdot e_1^T + c_{i2} \cdot \lambda_2 \cdot e_2^T + \ldots + c_{in} \cdot \lambda_n \cdot e_n^T \quad (16)$$

Similarly, we may get x_i^2 as,

$$\begin{aligned} x_i^2 &= x_i^1 \cdot P \\ &= c_{i1} \cdot \lambda_1 \cdot e_1^T \cdot P + c_{i2} \cdot \lambda_2 \cdot e_2^T \cdot P + \ldots + c_{in} \cdot \lambda_n \cdot e_n^T \cdot P \\ &= c_{i1} \cdot \lambda_1^2 \cdot e_1^T + c_{i2} \cdot \lambda_2^2 \cdot e_2^T + \ldots + c_{in} \cdot \lambda_n^2 \cdot e_n^T \end{aligned}$$

Thus, after a certain number of transitions, the vector x_i will eventually converge to the equilibrium probability vector. Say after k steps,

$$x_i^k = c_{i1} \cdot \lambda_1^k \cdot e_1^T + c_{i2} \cdot \lambda_2^k \cdot e_2^T + \ldots + c_{in} \cdot \lambda_n^k \cdot e_n^T \quad (17)$$

Now similarly, let x_j^0 represent the probability vector such that the system is 100% surely in state j initially when the number of transitions $t = 0$. Thus,

$$x_j^0 = \{0, 0, 0, \ldots, 0, 1, 0, \ldots, 0\}$$

Just like x_i^0 in equation (14), x_j^0 may also be expressed as a linear combination of the left hand eigenvectors of P,

$$x_j^0 = c_{j1} \cdot e_1^T + c_{j2} \cdot e_2^T + \ldots + c_{jn} \cdot e_n^T$$

Similarly,

$$\begin{aligned} x_j^1 &= c_{j1} \cdot e_1^T \cdot P + c_{j2} \cdot e_2^T \cdot P + \ldots + c_{jn} \cdot e_n^T \cdot P \\ &= c_{j1} \cdot \lambda_1 \cdot e_1^T + c_{j2} \cdot \lambda_2 \cdot e_2^T + \ldots + c_{jn} \cdot \lambda_n \cdot e_n^T \end{aligned}$$

and after k steps,

$$x_j^k = c_{j1} \cdot \lambda_1^k \cdot e_1^T + c_{j2} \cdot \lambda_2^k \cdot e_2^T + \ldots + c_{jn} \cdot \lambda_n^k \cdot e_n^T \quad (18)$$

Observing equations (17) and (18) together, the only difference between the two equations are the constants c_{il}, and c_{jl} where $l = 1, 2, \ldots, n$. In other words, the difference in the state of the system after k transition steps, when the starting state was state i, and when the starting state was state j, is represented by the difference in the values of the constants c_{il}, and c_{jl}, where $l = 1, 2, \ldots, n$. The values of c_{il}, and c_{jl}, $(l = 1, 2, \ldots, n)$ therefore hold the key to finding the difference in the distance (in terms of number of transition steps) of state i, and state j from any other state of the system. Thus, these constants would also reflect the difference in distance of the two states from the equilibrium state.

To calculate the values of $c_{1l}, c_{2l}, c_{3l}, \ldots, c_{nl}$, where $l = 1, 2, \ldots, n$, we make the following observations. We know that,

$$x_1^0 = \{1,0,0,\ldots,0\}$$
$$x_2^0 = \{0,1,0,\ldots,0\}$$
$$\vdots$$
$$x_n^0 = \{0,0,0,\ldots,1\}$$

Writing this in matrix form,

$$X = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} = \text{Identity Matrix} \quad (19)$$

We also know that,

$$x_1^0 = c_{11} \cdot e_1^T + c_{12} \cdot e_2^T + \ldots + c_{1n} \cdot e_n^T$$
$$x_2^0 = c_{21} \cdot e_1^T + c_{22} \cdot e_2^T + \ldots + c_{2n} \cdot e_n^T$$
$$\vdots$$
$$x_n^0 = c_{n1} \cdot e_1^T + c_{n2} \cdot e_2^T + \ldots + c_{nn} \cdot e_n^T$$

Writing this in matrix form as well,

$$\begin{bmatrix} x_1^0 \\ x_2^0 \\ x_3^0 \\ \vdots \\ x_n^0 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \ldots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \ldots & c_{2n} \\ c_{31} & c_{32} & c_{33} & \ldots & c_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \ldots & c_{nn} \end{bmatrix} \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \\ \vdots \\ e_n^T \end{bmatrix} \quad (20)$$

Therefore,

$$X = CE^T$$

From equation (19), we know that X is the identity matrix (I), therefore,

$$I = CE^T \Rightarrow C = \frac{I}{E^T} = (E^T)^{-1}$$

$(E^T)^{-1}$ is the matrix of the right-hand eigenvectors of matrix P. The values of the constants c_{ij}, which are capable of determining the distance of the various states from equilibrium, can thus be easily computed as the right-hand eigenvectors of matrix P. Of these right-hand eigenvectors, the sub-dominant eigenvector gives the *best* estimate of the distance of each state from equilibrium (Stewart, 1991). The sub-dominant eigenvector for our trip-planner example is shown below:

$$[0.3749, 0.3566, 0.3606, 0.3513, \mathbf{0.3435}, 0.3476, \ldots$$
$$\ldots 0.3352, 0.3420, 0.1038] \quad (21)$$

Translating these results to the service domain, we are in a position to calculate via the CTMC representation of the domain, the 'distance' of each service from equilibrium.

The important point here, however, is that the distance that needs to be calculated is the failure distance and not necessarily the equilibrium distance. A clear understanding on the relationship between the failure state and the equilibrium state needs to be established. We will attempt to do this through a simple example.

Suppose, the failure probability at equilibrium is very high (say 0.9). This means that the probability that the system at equilibrium is in the fail state is 0.9. This would be represented by the element corresponding to the fail state in the equilibrium probability vector π_s of equation (7). In such a scenario, the larger the distance of a service from equilibrium, the larger the failure distance and hence higher its reliability. Conversely, suppose the failure probability at equilibrium is a small value (*e.g.* 0.2). In this case, the smaller the distance of a service from the equilibrium, higher its reliability. In general, if the failure probability at equilibrium is greater than 0.5, a larger distance from equilibrium reflects higher reliability and vice-versa for a failure probability smaller than 0.5. The failure distance of a service may be calculated using the formula in equation (22).

$$\textit{Failure Distance} = \frac{1}{\lambda_{i \to Fail} * c_{i2}^{10 \cdot (0.5 - \pi_s(Fail))}} \quad (22)$$

c_{i2} is the magnitude of the i^{th} element of the subdominant right-hand eigenvector of the embedded Markov chain matrix (P) of the CTMC representation of the service domain. The magnitude of the sub-dominant eigenvector gives the *best* estimate of the distance of the i^{th} state from equilibrium, as mentioned earlier (Stewart, 1991). The larger the value of c_{i2}, the larger the distance of the i^{th} state from equilibrium. Therefore in equation (22), as long as the failure probability is less than 0.5, the factor $c_{i2}^{10 \cdot (0.5 - \pi_s(Fail)}$ is in the denominator of the expression (since $(0.5 - \pi_s(Fail))$ would be a positive value). Thus, a larger value of c_{i2} (which expresses a larger distance from equilibrium) results in a smaller value of failure distance. Also, the smaller the value of $\pi_s(Fail)$, the higher the power to which the c_{i2} gets raised and hence the failure distance becomes smaller. The opposite holds whenever $\pi_s(Fail)$ is greater than 0.5. When the failure probability at equilibrium is exacly 0.5, the factor $c_{i2}^{10 \cdot (0.5 - \pi_s(Fail)}$ becomes equal to 1 and the failure distance would depend only on the current value of the factor $\lambda_{i \to Fail}$. The failure distance values calculated for the services in our trip-planner example are: *TravelPedia* $\rightarrow$ 489.24 ; *FlightsCheap* $\rightarrow$ 1163.94 ; *EasyHotels* $\rightarrow$ 2091.66 ; *BestHotels* $\rightarrow$ 385.19 ; *HoteloCity* $\rightarrow$ 1370.28 ; *PaymentGate* $\rightarrow$ 644.16 ; and *EasyPay* $\rightarrow$ 392.76 .

The calculation of failure-distance for *BestHotels* is shown below:

$$\textit{Failure Distance}_{BestHotels} = \frac{1}{0.3 * 0.3435^{10\cdot(0.5-0.0555)}} = 385.19$$

The failure distance values so calculated take into account the interactions between the services. This is achieved owing to the fact that the transition rate values between the services in the CTMC representation have been considered in the calculation.

5 EXPERIMENTAL VALIDATION

In this section, we attempt to experimentally validate the technique proposed in this paper to calculate the individual reliabilities of services in a domain taking into account the influence of interacting services. To do this, experiments were conducted on a service domain with 29 services (excluding the first and the last) spread over 6 levels of functionality. Seven different sets of initial fail-rate and coupling values were experimented with. The experimental domain is shown in Figure 7.

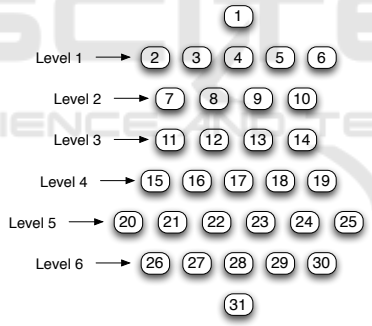

Figure 7: Service domain used in the experiments.

Simulations were run on the experimental domain, wherein services were allowed to fail randomly at their respective fail-rate values. 10,000 simulation runs were conducted, where each run comprised a failure being forced at each functionality level. Every time that a service failed, its own fail-rate was increased marginally (we could have recomputed the fail-rate value following equation (3) with the same effect; in the interest of simplicity we increased the fail-rate marginally), and so was the fail-rate of the services in the functionality level immediately above it. The increase in fail-rate of the parent services was proportional to the value of coupling between them and the failing service. Therefore, a service that had a stronger coupling with the failing service suffered a larger increase in its fail rate. This was done in conformance with the general belief that the reliability value of a service influences the reliability of the interacting services, particularly the ones that invoke it.

A number of simulation scenarios were observed. However, only the two main ones have been discussed here. The first simulation scenario was the static scenario, wherein services were allowed to fail, their fail-rates on failing were increased, and they were allowed to fail again. The fail-rate of services were allowed to rise indefinitely. The second scenario was a more dynamic scenario, wherein services were allowed to fail, and their fail-rates were increased on failing as in the static case. However, once the fail-rate of a service exceeded a certain maximum, the service was taken-out for 'repair' and brought in after a few runs with its fail-rate reinstated to its original value.

The motive behind this simulation exercise was to get an idea of the possible system behaviour if it were observed for a very long period of time (one in which 10,000 failures occurred at each functionality level). The number of failures for each service returned by the simulation formed the basis for the validation of our proposed technique against the existing techniques for reliability assessment in an interactive environment.

The proposed technique, described in previous sections, involved the representation of the service domain in its equivalent CTMC form, putting together the 'infinitesimal generator matrix' (shown in Figure 5 for the trip-planner example) of this CTMC, calculating the 'embedded Markov chain matrix' (shown in Figure 6 for the trip-planner example), and subsequently finding the left-hand eigenvector (corresponding to the unit eigenvalue) which is the equilibrium probability vector of the system (shown in equation (10)) and the subdominant right-hand eigenvector of this matrix. Finally the equilibrium probability vector and the subdominant right-hand eigenvector were used to calculate the 'failure distance' of each service in the domain using equation (22). The service selected at each functionality level was the one with the *largest* value of failure distance at that level. This service was considered to be furthest from the 'fail' state and thus the most reliable.

The existing techniques as discussed in the 'related work' section (section 2) were varied. However, in terms of calculating the individual reliabilities of services, all of them were similar in that they all calculated the individual reliabilities of the services in isolation. That is to say that they assume that the reliabilities of individual services in the domain would be unaffected by their interaction with other services in the domain. Therefore, to replicate the existing tech-

niques in this domain, the initial fail-rate values of the services were utilized, and at each level the service which had the *smallest* fail-rate value at that level was selected.

Static scenario				
	Existing techniques		Proposed technique	
	Failures	Reliability	Failures	Reliability
Domain 1	6,050/10,000	39.5%	1,408/10,000	85.92%
Domain 2	8,765/10,000	12.35%	712/10,000	92.88%
Domain 3	9,008/10,000	9.92%	16/10,000	99.84%
Domain 4	9,844/10,000	1.56%	526/10,000	94.74%
Domain 5	9,778/10,000	2.22%	521/10,000	94.79%
Domain 6	6,490/10,000	35.1%	4,236/10,000	57.64%
Domain 7	9,983/10,000	1.7%	170/10,000	98.3%

Figure 8: Results of experiments in a static scenario (no repair of services).

Dynamic scenario (repair time: 1 iteration)				
	Existing techniques		Proposed technique	
	Failures	Reliability	Failures	Reliability
Domain 1	8,231/10,000	17.69%	100/10,000	99%
Domain 2	1,168/10,000	88.32%	244/10,000	97.56%
Domain 3	1,969/10,000	80.31%	157/10,000	98.43%
Domain 4	1,420/10,000	85.8%	161/10,000	98.39%
Domain 5	3,705/10,000	62.95%	669/10,000	99.31%
Domain 6	688/10,000	93.12%	181/10,000	98.19%
Domain 7	7,199/10,000	28.01%	4,492/10,000	55.08%

Figure 9: Results of experiments in a dynamic scenario (with repair of services).

The sequence followed in the experiments for the static scenario is now described. The proposed technique and existing techniques were first applied on the experimental domain. These gave the set of services selected at each level. The simulation, was next run for $10,000$ iterations each. Each iteration comprised the occurrence of 1 failure at each functionality level. The failing services on each iteration were compared with the services selected by the two techniques (proposed and existing). If any of the selected services by a selection technique happened to fail in an iteration of the simulation, the selected technique was said to have failed in that iteration. In other words, a selection was said to be a success on a particular iteration if and only if none of the services selected (at any level) was one of the failing services in the iteration.

The experimental procedure for the dynamic scenario was similar with the only difference being that the selection techniques (both proposed and existing) were applied after every simulation run rather than only once in the beginning. This was because, with services being taken out for repair, the complexion of the domain was possibly changing on every run.

Subsequently the reliability of the composite application selected by either selection technique was calculated as shown in equation (23).

$$Reliability = \frac{total\ no.\ of\ iterations - total\ no.\ of\ failures}{total\ no.\ of\ iterations} \quad (23)$$

The results for the two scenarios have been shown in the tables in Figures 8, and 9. The results show that in both cases, the proposed technique outperform the existing techniques. The results therefore indicate that the proposed technique does manage to compose a more reliable application in a scenario where the service components are interacting with each other and the reliability of a service that invokes another is affected by that of the one invoked.

6 CONCLUSIONS

The technique proposed in this paper is significant in the following respects: a) It focuses on reliability as a central factor in service selection that aligns well with the concerns of most customers. b) The reliability of individual services is calculated taking into account the interaction of the service with others in the domain which is the novel feature in this technique.

There is however, substantial scope for future work. The model presented in this paper is, in fact, still incomplete as it only takes into consideration the reliability of the prospective services. A number of crucial factors influencing service selection in a composition scenario are yet to be incorporated in this model. These include, analysis of how the prospective services compare in terms of the customer perception of the QoS attributes, how factors such as 'improvement trends' of the services on offer could affect their respective selection potential, the impact that the average waiting time of a service would have on its potential as a prospective service *etc.*

REFERENCES

Balbo, G. (2002). Introducation to stochastic petri nets. In *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*, pages 84–155.

Battilani, P. and Fauri, F. (2007). The rise of a service-based economy and its transformation: the case of rimini. In

Rimini Centre for Economic Analysis, Working Paper Series.

Epifani, I., Ghezzi, C., and Mirandola, R. (2009). Model evolution by run-time parameter adaptation. In *Proceedings of the International Conference on Software Engineering*.

Greub, W. H. (1981). *Linear Algebra*. Springer.

Kilburn, J. (2003). The decline of the mills and the rise of social services in a northeastern mill town. In *Annual meeting of the American Sociological Association*.

Kokash, N. (2005). Web service dicovery with implicit qos filtering. In *Proceedings of the IBM PhD Student Symposium, in conjunction with the International Conference on Service Oriented Computing (ICSOC)*, pages 61–66.

Norris, J. R. (1998). *Markov Chains*. Cambridge University Press.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, pages 3–12.

Ran, S. (2003). A model for web services discovery with qos. In *ACM SIGecom Exchanges*, pages 1–10.

Stewart, W. J. (1991). *Introduction to the numerical solution of markov chains*. CRC press.

Tsai, W., Zhang, D., Chen, Y., Huang, H., Paul, R., and Liao, N. (2004). A software reliability model for web services. In *Proceedings on Software Engineering and Applications (SEA)*.

Wang, W.-L., Wu, Y., and Chen, M.-H. (2003). An architecture-based software reliability model. In *Proceedings of the 12th international conference on World Wide Web*.

Yu, T. and Lin, K.-J. (2004). Service selection algorithms for web services with end-to-end qos constraints. In *Proceedings of the International Conference on e-Commerce Technology*.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*.

Zhovtobryukh, D. (2007). A petri net-based approach for automated goal-driven web service composition. In *Society for Computer Simulation International*, volume 83, pages 33–63.