Time Based QoS Modeling and Prediction for Web Services

Leilei Chen[1], Jian Yang[2], and Liang Zhang[1]

[1] School of Computer Science, Fudan University, China
{081024012,lzhang}@fudan.edu.cn
[2] Department of Computing, Macquarie University, Australia
jian.yang@mq.edu.au

Abstract. Quality of Service (QoS) prediction and aggregation for composite services is one of the key issues in service computing. Existing solutions model service QoSs either as deterministic values or probabilistic distributions. However, these works overlooked an important aspect in QoS modeling, *time*. Most QoS metrics, such as response time, availability, are time-dependent. We believe time variation should be explicitly reflected in QoS modeling as well as aggregation. In this paper, we propose a dynamic web service QoS model to capture the time based QoS patterns, based on which QoS of composite services are aggregated.

1 Introduction

The QoS aspect of web services has attracted much attention from the research community [1,2,3,4,5,6,7]. QoS aggregation, i.e., estimating the QoS of a composite service based on the available QoS information of the component services, becomes crutial for QoS-based service selection [1,2,3,4]. Here, two issues need to be addressed: (1) how QoS of web services can be effectively modeled? (2) how the QoS of individual services can be aggregated according to the composition structure.

QoS modeling of web services in existing works can be classified as following: (1) *Deterministic QoS value* [2,3]. (2) *Probability Mass Function* (PMF) [4], which represents the probability of a QoS over its discrete values; (3) *Probability Density Function* (PDF) of some standard statistical distributions [1,5]; (4) PDF of irregular distributions [6]. However, all the above works have neglected a commonly observed phenomenon in reality: service QoSs often exhibit some time related patterns. Taking stock service as an example, usually the service is mostly accessed during trading hours and the usage of the service outside these peak hours is relatively low. As a result, within different time periods, service QoS can vary dramatically. A static QoS model is inadequate to represent service QoS with different characteristics in different time periods, and the aggregated QoS for composite service based on these models can not effectively reflect the impact of time.

Modeling and aggregating time based QoS is challenging. First of all, how can we find the *time-based QoS Change Cycle*? This cycle of QoS change presents

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 532–540, 2011.

repeating patterns on the regular basis. Secondly, how can we capture the individual time spans within the cycle, in which the QoSs exhibit different patterns? Thirdly, aggregating component QoSs need to consider both the composition structure and the distinctive characteristics for each QoS aspect. In this paper, we will tackle this challenge by establishing dynamic QoS models for web services and providing methods for QoS aggregation based on these dynamic models.

The paper is organized as follows: Section 2 discusses the related work. In Section 3, we develop the time based dynamic QoS model for web services. In Section 4, QoS aggregation based on the dynamic model is explained. Section 5 presents the simulation results and Section 6 concludes the paper.

2 Related Work

Several previous studies have addressed the problems of QoS modeling and QoS aggregation. Cardoso *et al* [1] propose a Stochastic Workflow Reduction (SWR) algorithm to compute the QoS of a composition with sequential, parallel, conditional and simple loop blocks. Jaeger *et al* [2] examine more composition patterns and Dumas *et al* [3] propose a QoS aggregation method that can handle unstructured acyclic fragments. Hwang *et al* [4] propose a probability-based QoS model where a QoS measure of an atomic or composite service is quantified as a PMF. Zheng *et al* [6] adopts a Gaussian Kernel Density estimation approach to generate the PDFs for component services, based on which, QoS aggregation is implemented to get more accurate estimations for composite services.

None of the above studies take the timing characteristics of QoS into consideration. Klein *et al* [7] address the inherent variability in the actual values of QoS at runtime. Our research is motivated by the similar concern as Klein's. However, we focus more on the details of setting up dynamic QoS models and computing aggregated QoSs for composite services.

3 Dynamic QoS Modeling for Web Services

Web services can exhibit different quality in different environmental conditions such as usage, network traffic. In many real-world cases, the change of the environmental conditions presents certain periodical patterns, which leads to the changes on the service QoSs. We use *time cycle (TC)* to represent the period when the service has different qualities which will re-appear in the next cycle.

We differentiate two kinds of QoS metrics based on the feature of their value space: continuous values and discrete values, which bring different complexities into QoS aggregation. We use PDF to model continuous QoS metrics such as response time. For discrete QoS metrics including reliability and cost, we use PMF. Moreover, as different QoS metrics always correlate in the same time period, it is preferable to integrate all QoS metrics into one dynamic model in order to capture the impact of time pattern on the service. In this work, we will consider three representative QoS metrics: response time, reliability and cost.

Definition. A Dynamic QoS Model(DQM) for a service is a tuple (TC, I, DM), where

• $TC = [T_0, T_0 + T)$ is the QoS change time cycle and T_0 is the cycle starting time; T is the length of the time cycle.

• $I = \{I_0, I_1, ..., I_{N-1}\}$ is a segmentation of TC, where $I_i = [t_i, t_{i+1})$, and $\bigcup_i I_i = TC$

• $DM =< SM_0, SM_1, ..., SM_{N-1} >$ is a sequence of QoS models in the time segments, where $SM_i = (f_{resp_i}, P_{rel_i}, P_{cost_i})$ is a vector of probability models corresponding to the segment I_i, and

▷f_{resp_i} is the PDF of response time

▷P_{rel_i} is the PMF of reliability

▷P_{cost_i} is the PMF of cost

Given the DQM of a service, the QoS probability model can be represented as a function of time, i.e., $Q(t) = (f_{resp_i}, P_{rel_i}, P_{cost_i})$, where $t \in [t_i, t_{i+1})$.

The construction of the segment models can be achieved using existing probabilistic methods that have been discussed in previous works. Here, we focus on identifying the time cycle TC of QoS and getting a reasonable segmentation of the cycle for the DQM of a service.

We regard a service combined with its executing environment as a 'QoS-generating' system. We observe the dynamic QoS displayed by the system to find the internal mechanism that generates the QoS. For this purpose, we adopt a Hidden Markov Model (HMM) [8] based method, assuming that the hidden mechanism is a state machine, each state corresponds to a certain environmental condition, within which the system exhibits stable behavior (i.e., the QoS the system generates within each state demonstrates the stable statistical properties), and state transitions have certain regularity.

A HMM is formally defined as $\lambda = (S, A, B, \Pi)$, where

• S is a finite set of states;

• $A = \{a_{ij}\}$ is the transition probability matrix, in which, a_{ij} is the probability of state i transiting to state j in the next step;

• $B = \{b_i(o)\}$ is the emission probability matrix, where o is an observation and $b_i(o)$ represents the probability of observing o when the system is in state i;

• $\Pi = \{\pi_i\}$, in which π_i is the probability of the system beginning in state i.

In this paper, we adopt two traditional HMM algorithms. The first one is Baum-Welch algorithm, i.e., given an observation sequence $O_i(i = 1, 2, ...)$ generated by a system, to get a model λ to best describe the system. And the second one is Viterbi algorithm, i.e., given a model λ and a sequence of observations O_i, to find the hidden state sequence S_i.

To model the system as a HMM, we first partition the time into equal intervals. Next, we get the QoS statistics of the service in each time interval and take them as the interval's observation. As discussed before, some QoS metrics have discrete value domains while others have inherently continuous values. In order to uniformly represent the QoS metrics in HMM, for each time interval, we compute the expectation value of each QoS metric by taking the mean of the observed values, and combine them into a QoS observation vector. The QoS observation vector of

the ith time interval is $O_i(E_{resp_i}, E_{rel_i}, E_{cost_i})$, where E_{resp_i},E_{rel_i},E_{cost_i} represent the expectation values of response time, reliability and cost respectively. In this vector, E_{rel_i} and E_{cost_i} are represented as continuous values with the sample spaces as $Dom(E_{rel_i}) = [0, 1]$ and $Dom(E_{cost_i}) = [min(cost), max(cost)]$, respectively. In this way, all the QoS metrics are represented as continuous variables uniformly. As different QoS features have quite different scales, we normalize all variables into the scale range of $[0, 1]$ according to the following formula:

$$E'_{q_i} = \begin{cases} \frac{E_{q_i} - E_q^{min}}{E_q^{max} - E_q^{min}} & \text{if } E_q^{max} - E_q^{min} \neq 0 \\ 1 & \text{if } E_q^{max} - E_q^{min} = 0 \end{cases}$$

where E_{q_i} represents the element of O_i, $E_q^{max} = max(E_{q_i})$ and $E_q^{min} = min(E_{q_i})$. So, we get a sequence of observation vectors represented as: $O'_i(E'_{resp_i}, E'_{rel_i}, E'_{cost_i})$, $i = 1, 2, ...n$, where n is the total number of the time intervals created. As now the elements of the vector have continuous values, we assume they obey Gaussian Mixture distribution. Therefore, the system is eventually modeled as a Gaussian Mixture HMM with the emission probability density function for each state defined by multivariate Gaussian distribution.

Given the observation vector sequence $O =< O'_1, O'_2, ..., O'_n >$, we use the Baum-Welch algorithm to get the HMM λ that best describes the system. Then, with λ and O, utilize the Viterbi algorithm to get the hidden state sequence. Next, self-transitive states are gradually merged, from which we get the segmentation of the state sequence. The time cycle TC will be eventually recognized from the segmentation.

With the time cycle pattern recognized, we can set up segment QoS models for each time segment and integrate them into the final DQM.

4 QoS Aggregation Based on DQM

In this section, we shall address the issue of estimating the QoS of a composite service based on the DQMs of its component services. We focus on composite services with four structured composition patterns, including sequential pattern, parallel pattern, conditional pattern and loop pattern. Three challenge issues remain: (1) determining the time cycle of a composite service; (2) estimating the QoS of the composite service if it is invoked at a certain time point; (3) obtaining the DQM of the composite service.

4.1 Estimating the Time Cycle Length of Composite Services

The cycle length of a composite service (denoted as T) is computed as the least common multiple (LCM) of all the cycle lengths of its component services. We then standardize the DQMs of all the component services. That is, expanding the cycles of all the component services to the standard T. By doing this, all services under consideration have the same cycle length, and the time can be represented uniformly in all the DQMs.

4.2 Estimating QoS of Composite Services for Single Invocation

Problem Statement: Suppose composite service CS is recursively constructed using the following basic composition patterns: sequential, parallel, conditional, and loop, its component services are $ws_1, ..., ws_M$, and the respective standardized DQMs are $DQM_1, ..., DQM_M$, given an invocation time t_0, ($T_0 \leq t_0 < T_0+T$, where T_0 is the standardized cycle starting time and T is the standardized cycle length), estimate the QoS of CS for this invocation.

The impact of a single component service ws_i on the overall QoS of CS correlates with the time when ws_i is invoked. So, we need to estimate the invocation time for every ws_i. Furthermore, as the response time of ws_i affects the invocation time of all services that are invoked after it, the estimation of invocation time have to be conducted in one direction, i.e., traversing the composition process step by step and from the beginning to the end. We identify two set of parameters in relation to the reduction of a composition pattern: input QoS parameters and output QoS parameters. We use $SubCS$ to represent a sub composition structure of CS. Given the input QoS parameters of $SubCS$, we compute its output parameters according to its composition pattern, which will then be transferred to its direct successor as the inputs. The reduction algorithm will be repeatedly executed from the very beginning and the effect of each component service is gradually reduced and merged till the end of CS's process.

Input QoS parameters of a $SubCS$ include: (1) $startTime$, a continuous variable that represents the time when $SubCS$ is invoked, with its PDF denoted as f_{stime}; (2) $startRel$, a discrete variable whose value domain is $\{0, 1\}$, where $P(startRel = 1)$ represents the probability of $SubCS$ can be successfully invoked, with its PMF denoted as P_{srel}; (3) $startCost$, a discrete variable that represents the cumulated cost of CS before $SubCS$ is invoked, with its PMF denoted as P_{scost}. And output parameters of a $SubCS$ include: (1) $endTime$, a continuous variable that represents the time when the executing of $SubCS$ is finished, with its PDF denoted as f_{etime}; (2) $endRel$, a discrete variable whose value domain is $\{0, 1\}$, where $P(endRel = 1)$ represents the probability that $SubCS$ is just successfully executed, with its PMF denoted as P_{erel}; (3) $endCost$, a discrete variable that represents the cumulated cost of CS just after $SubCS$ is executed, with its PMF denoted as P_{ecost}.

For the first $SubCS$ that should be invoked in CS, the initial input parameter models are as follows: $f_{stime}(x) = \delta(x - t_0)$, $P_{srel}(1) = 1$, $P_{srel}(0) = 0$, $P_{scost}(0) = 1$. Below, we will introduce the method for calculating output parameters for different structured $SubCS$, including a single service, parallel composition, conditional composition and loop composition. Sequential composition will not be discussed here since it is just the simple repeating of single services.

A Single Service: $endTime$ is the sum of $startTime$ and ws's response time; The probability of ws being successfully executed is the product of the probability of it being successfully invoked and the reliability of ws; $endCost$ is the sum of $startCost$ and ws's cost.

$$f_{etime}(x) = f_{stime}(x) \circledast \sum_{i=0}^{N-1} (\sum_{m} \int_{t_i+mT}^{t_{i+1}+mT} f_{stime}(x)dx) \cdot f_{resp_i}(x) \tag{1}$$

$$\begin{aligned} P_{erel}(1) &= P_{srel}(1) \cdot \sum_{i=0}^{N-1} (\sum_{m} \int_{t_i+mT}^{t_{i+1}+mT} f_{stime}(x)dx) \cdot P_{rel_i}(1) \\ P_{erel}(0) &= 1 - P_{erel}(1) \end{aligned} \tag{2}$$

$$P_{ecost}(z) = \sum_{y_1+y_2=z} P_{scost}(y_1) \cdot \sum_{i=0}^{N-1} (\sum_{m} \int_{t_i+mT}^{t_{i+1}+mT} f_{stime}(x)dx) \cdot P_{cost_i}(y_2) \tag{3}$$

Parallel Composition: We first compute the output models for each branch. The models for the jth branch are calculated according to Formula (1), (2) and (3), and we denote them as f^j_{etime}, P^j_{erel} and P^j_{ecost}. The *endTime* of the composition is the maximum *endTime* of each branch; the composition is successfully ended if and only if all the branches are successfully ended; the *endCost* of the composition is the sum of each branch's *endCost*.

$$\begin{aligned} F^j_{etime}(x) &= \int f^j_{etime}(x)dx \\ f_{etime}(x) &= \sum_{j=1}^{k} f^j_{etime}(x) \cdot \prod_{l=1,\ldots,k \& l \neq j} F^l_{etime}(x) \end{aligned} \tag{4}$$

$$P_{erel}(1) = \prod_{j=1}^{k} P^j_{erel}(1) \qquad P_{erel}(0) = 1 - P_{erel}(1) \tag{5}$$

$$P_{ecost}(z) = \sum_{y_1+\ldots+y_k=z} \prod_{j=1}^{k} P^j_{ecost}(y_j) \tag{6}$$

Conditional Composition: We first compute the output models for each branch. The output parameters are the probability weighted sum of each branch.

$$f_{etime}(x) = \sum_{j=1}^{k} p_j f^j_{etime}(x) \tag{7}$$

$$P_{erel}(1) = \sum_{j=1}^{k} p_j P^j_{erel}(1) \qquad P_{erel}(0) = 1 - P_{erel}(1) \tag{8}$$

$$P_{ecost}(z) = \sum_{j=1}^{k} p_j P^j_{ecost}(z) \tag{9}$$

Loop Composition: We model the iteration number as a PMF. So a loop composition can be transformed into a conditional composition with a sequential composition in each path. The output models are then calculated according to the corresponding formulas.

By applying the above calculation methods, we get the probabilistic model for a composite service's *endTime*, *endRel* and *endCost* when it is invoked at a certain time t_0. The definitions of *endRel* and *endCost* remain the same with reliability and cost respectively. However, *endTime* is not the response time of the composite service. We get the response time model as $f_{resp}(x) = f_{etime}(x + t_0)$.

4.3 Establishing DQM for a Composite Service

With the problem of QoS estimation for single invocation solved, now we discuss how DQM is established for a composite service to reveal its QoS change patterns in a time cycle. If two invocation time points (to the same service) are very close, it is very likely that the QoS of these two invocations are very similar. Based on this observation, we first select a sequence of time points t_0, t_1, ..., t_l ($T_0 \leq t_0 < t_1 < \ldots < t_l \leq T_0 + T$, where T_0 is the cycle starting time and T is the cycle length). Then we compute the corresponding QoS models and get a model sequence M_{t_0},...,M_{t_l}. Next, we gradually merge adjacent models which are most similar. A merged model is computed as the average of two adjacent models. The merge procedure will continue until the similarities of current resultant models are all below a certain threshold. When the merge is finished, a new model sequence $M_{t_0 \ldots t_a}, M_{t_{a+1} \ldots t_b}, \ldots\ldots, M_{t_x \ldots t_l}$ is generated. Then, we segment the time cycle in the following way: if two adjacent time point t_y and t_{y+1} are assigned to two different models, then we set the middle of them as a split time point: $st = \frac{t_y + t_{y+1}}{2}$. And $M_{t_p \ldots t_q}$ is the corresponding segment model for time span $[\frac{t_{p-1}+t_p}{2}, \frac{t_q+t_{q+1}}{2})$.

In the merging process, we use Kullback-Leibler divergence to measure the difference of two probability distributions. Two distributions are similar if the KL divergence of them is small. There are three elements in our QoS model and we need to compute the KL divergence for every element and compute the sum of them as the final distance of two models.

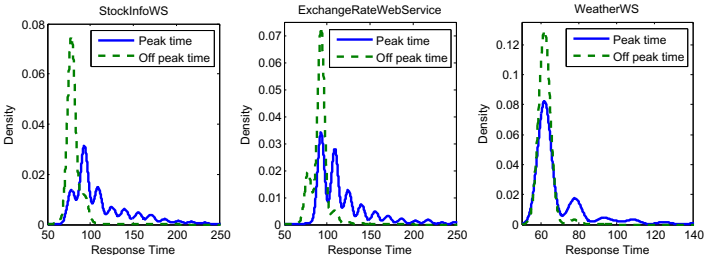

Fig. 1. PDFs of Real World Services

5 Experiment and Evaluation

To justify the effectiveness of the proposed dynamic QoS model and approach, we collect the response time from three real world web services: *StockInfoWS*[1], *ExchangeRateWebService*[2], and *WeatherWS*[3]. We first identify the QoS change

[1] http://www.webxml.com.cn/WebServices/StockInfoWS.asmx?wsdl
[2] http://webservice.webxml.com.cn/WebServices/ExchangeRateWebService.asmx?wsdl
[3] http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl

pattern for these services based on the HMM method. The results show that all three services possess a circle length of 24 hours. *StockInfoWS* exhibits higher values for response time during work hours from approximately 9:30 to 15:00, medium values during lunch time and lower values for other time periods; The QoS pattern of *ExchangeRateWebService* is similar to *StockInfoWS* except that it does not present medium values during lunch time; For *WeatherWS*, the response time is high only within the peak hours between 9:15 and 10:45, and relatively low outside this time range. The PDFs of the services during peak hours and off peak hours are shown in Figure 1.

Next, we will show the accuracy of the QoS aggregation method proposed in Section IV. We create three web services $ws1$, $ws2$ and $ws3$, and their response time values are generated randomly from the collected samples of *StockInfoWS*, *ExchangeRateWebService* and *WeatherWS*, respectively. Then, these services are composed according to the composition structure shown in Figure 2. The probability of each conditional branch is set to be 0.5. And the probabilities that the loop structure will be executed 2 and 3 times are all set to be 0.5.

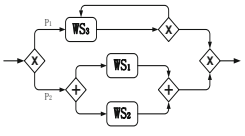

Fig. 2. A Composition

Monte Carlo simulation method is then used to simulate the QoS of the composite service. Based on the DQMs of the component services, we apply our estimation method to compute the PDF for the composite service. In addition, we also generate static QoS models for the component services when the time cycle related QoS changes are not taken into consideration, and compute the PDF of the composition accordingly. Figure 3 illustrates the experiment results when the invocation time is set that all component services are within peak hours. It shows that the PDF computed based on DQMs is very close to the simulation results, while the static models generate more deviations.

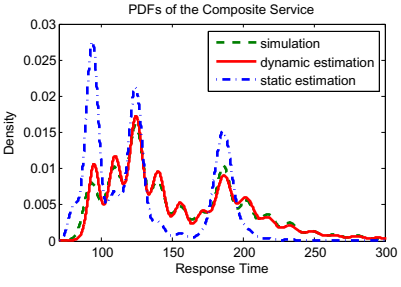

Fig. 3. PDFs of the Composition

6 Conclusion

In this paper we propose a dynamic QoS model to represent the time related characteristics of QoS. Various techniques are employed to develop the DQMs for component services as well as composite services. Experiments show that the proposed solution achieves high accuracy in QoS modeling and prediction.

Acknowledgment. This work is supported in part by NSFC grant 60873115.

References

1. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics 1, 281–308 (2004)
2. Jaeger, M.C., Rojec-Goldmann, G., Muhl, G.: QoS aggregation for Web service composition using workflow patterns. In: EDOC 2004, pp. 149–159 (2004)
3. Dumas, M., García-Bañuelos, L., Polyvyanyy, A., Yang, Y., Zhang, L.: Aggregate Quality of Service Computation for Composite Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 213–227. Springer, Heidelberg (2010)
4. Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workfows. Information Sciences 177(23), 5484–5503 (2007)
5. Rosario, S., Benveniste, A., Haar, S., Jard, C.: Probabilistic qos and soft contracts for transaction-based web services orchestrations. IEEE Transactions on Services Computing 1(4), 187–200 (2008)
6. Zheng, H., Yang, J., Zhao, W.: Qos Probability Distribution Estimation for Web Services and Service Compositions. In: SOCA 2010 (2010)
7. Klein, A., Ishikawa, F., Bauer, B.: A Probabilistic Approach to Service Selection with Conditional Contracts and Usage Patterns. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 253–268. Springer, Heidelberg (2009)
8. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition, pp. 267–296 (1990)