International Journal For Technological Research In Engineering
Volume 1, Issue 4, December - 2013 *ISSN (Online) : 2347 - 4718*

PREDICTING QUALITY MEASURES OF COMPOSITE WEB SERVICES

Karthikeyan. J[1], Suresh Kumar. M[2]
[1]PG Scholar, Computer and Communication Engineering
[2]Associate Professor, Department of Information Technology
Sri Sairam Engineering College, Chennai

***Abstract:** Now-a-days the web services are uprising technology which appeals lots of attention from both academic and industry. The quality of service has been one of the major challenges in the web services. If the web services become operational then the service providers with Service Level Agreement (SLA) need to consider and monitor with respect to the quality of web services and SLA. The most important factor is that to monitor and prevent the SLA intrusion in the composite web services. In the existing work the adaptation is done only at the instance level of the composition, for each level of composition the adaptation is done in each instance. Then the aggregated SLO's are defined over the number of instances which are out of scope and also the adaptation is not considered as permanent. In this PERvent (prediction and prevention based on event monitoring) framework is not supported for large number of adaptation action per the checkpoint because it is fully based on the number of checkpoints, and also it supports limited structural adaptation in the composition. The proposed work is to exclude these drawbacks by introducing the framework called Ranking QoS based dynamic composition (RQoSDC) framework for monitoring the SLA parameter as well as monitoring Service Quality (QOS) Parameter for composite web services. The efficient optimizing algorithm used to reduce the cost for the service provider and also we will discuss the experimental results to prove how these approaches to monitor the Quality of Service (QOS) parameters for the composed web services as well as SLA and to reduce the cost for composite web service providers.*

***Keywords:** Composite web services, Monitoring, Quality of Services (QoS), Service level Agreement (SLA).*

I. INTRODUCTION

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards the Web services architecture is based upon the interactions between three primary roles: service provider, service registry, and service requestor. These roles interact using publish, find, and bind operations. The service provider is the business that provides access to the Web service and publishes the service description in a service registry. The service requestor finds the service description in a service registry and uses the information in the description to bind to a service. A logical view of the Web services architecture is shown in Figure 1. In this view of the Web services architecture, the service registry provides a centralized location for storing service descriptions. A UDDI registry is an example of this type of service registry.

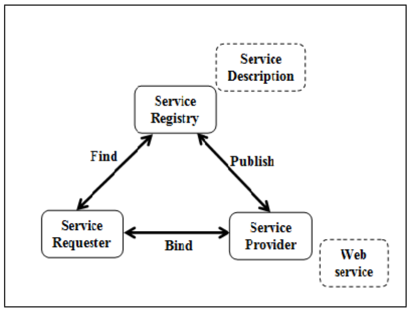


Figure 1: Web services architecture

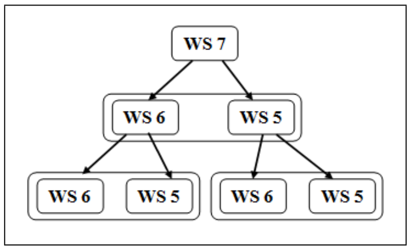


Figure 2: Web service composition

A service-level agreement (SLA) is a contract between a network service provider and a customer that specifies, usually in measurable terms, what services the network service provider will furnish. Many Internet service providers (ISP) s provide their customers with an SLA. More recently, IS (Information Services or Information System) departments in major enterprises have adopted the idea of writing a service level agreement so that services for their customers (users in other departments within the enterprise) can be measured, justified, and perhaps compared with those of outsourcing network providers.

Web service composition is able to compose services into composite services, which in turn can be composed into even bigger composite services. This is illustrated by Figure 2.

Where web services W1 and W2 are composed into a composite service W5, and web services W3 and W4 are composed into a composite service W6. W5 and W6 are in turn composed into W7.

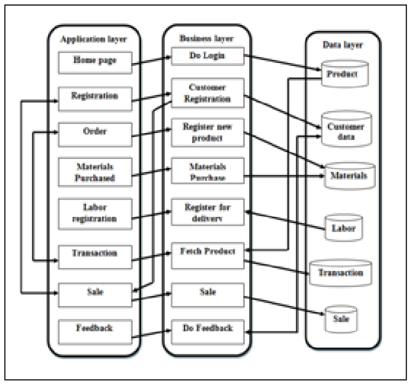


Figure 3: Production planning application

II. MOTIVATION

In this paper, we use the scenario depicted in Fig. 1 to motivate and explain our approach. This scenario considers the case of "IPLAN"- Production Planning application (Figure 1) is a product that processes the allocation of all supplies that the business takes in and ensures that they are taken to the proper place and used at the proper time to manufacture products in order to satisfy the demand of the customer. IPLAN tracks the creation of products from beginning to end. It coordinates the activities such as: It takes in the order placed by the customer, analyzes the materials required to create the product, predicts the number of days required to manufacture the product, cost and maintenance and quality assurance.

III. WEB SERVICE COMPSITION

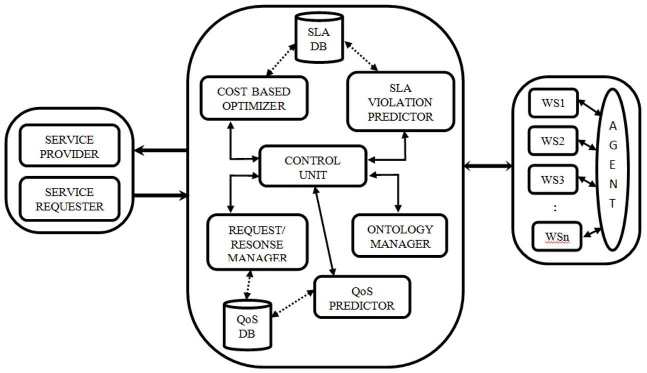


Figure 4: Framework

In this paper proposed a framework called Ranking QoS-based framework. In this framework includes with ontology's for improving the web semantic discovery and also provide the best service according to the customer interest which is based on the parameter cost and time and provide the reliable response to the customer based on the selection. This framework processes the services based on customer specification and providing the best response to the user requirement by the ontology concepts.

In this framework, we used a monitoring mechanism based on dynamic execution of web service and continually we calculate and rank the service based on the customer's response. And analysis the QoS parameter and display as output of the web service utilization; this report comes from the monitoring agent, getting feedback from the user which utilizes the service and service provider. Then the report stores in the database for future use.

In this framework consists of following components: cost based optimizer, SLA violation predictor, control unit, request/response manager, QoS manager and managing services. The architecture of the framework is shown in the figure 4. Those components are discussed in detail in following section.

a) **Control units**
Control unit is responsible for all the functional parts of the framework. Mainly this control unit is responsible for managing the communication between the each component in the framework. It will manage request gets from the client and replies to the customer based on their requirements. And also it manages storage of the final result of customer responses in the database.

b) **Request/Response manager**
This component based on the input and output. The requests are the inputs, it depends upon the nature of the customer and the responses are the output which depends upon the user request with the recognized result and requested services for the customer. The request and response include the following requirements

(a) Requests from providers in order to register in web services and gets a response from the service providers.

(b) Requests sent from the consumer, which based upon requests or the feedback of the used service by the customer and response to the user request.

(c) The request will be sent to the agent side to update the QoS parameters which include the service like response time, accessibility and availability of the service, etc.,

c) **SLA violation predictor**
SLA violation predictor is responsible for predicting the service which is not assigned to that particular customer. SLA agreement services are stored in the database for each customer which may helpful to predict the violation against the customer. It may help not to pay the penalties for the SLA violations. In the SLA violation is used to estimate the historical execution of each service which is used to produce the numerical estimation of SLO parameter at runtime.

d) **QoS predictor**
QoS predicator is mainly used for estimating the QoS parameter for web service such as service response time, performance, reliability, integrity, availability and accessibility of services Which is calculated by the mathematical expressions which is discussed in later sections. It is used to estimate the quality parameter based on cost or time.

And also stores the result in the QoS database in the framework.

e) **Cost based optimizer**
Cost based optimizer is used to adaption and minimize the cost for the service providers. The optimizer make decision depends upon the service generated prediction and the SLA agreement which is also helpful for SLA violation predictor. The generated prediction are present at the runtime adaption which is gets optimal cost for the service provider in dynamic composition of web services.

f) **Ontology manager**
Ontology Manager is responsible for managing common and domain-specific ontology's used by provider and requester. It is extracted domain-specific service related QoS and functional properties of advertising Web service that published by the provider. This component merges these ontology's with general ontology's (i.e. WordNet, Yago, Wikipedia...) and creates new generalized ontology. And also this component will rank and propose the sorted list of semantic web services that provide user preferences by using this ontology's.

g) **QoS Database**
QoS Database is used to store the QoS parameters of web services which are maintained by two kinds of table. One is used for store the QoS related information published by web service providers and the other one is used for stored the dynamic modified QoS parameter of the Web service, which is based on the monitoring agent and the user feedback. The each QoS parameters of the web service are identified by the unique identification number called UIDs. Each UID have the details about the service response time, performance, reliability, integrity, availability and accessibility of services

IV. Estimating QoS parameter

The QoS based selection of best services for simple and complex task requires continuously updating the published QoS information. QoS information can be updated only by monitoring the QoS parameters performed by trusted third party brokers. To perform monitoring and evaluation of QoS parameters associated with Web services, access rate are measured in this section that can help service consumer and provider to pay little attention towards the QoS parameters during service discovery and registration. These measured parameters will be discussed in the following subsection.

A. Access Rate Parameter

Access rate is an accessibility parameter that requires continuous monitoring of Web services by broker to provide updated QoS information. Access rate is directly related with the availability from the host location. Before publishing a new service on the server, the broker monitors the particular service by invoking it in own environment for a specific time period. The purpose of monitoring the Web service is to evaluate the access rate parameter value and other QoS parameter values, so that the consumer can always access the Web service with updated QoS information.

Access rate can be defined as the rate of total number of Web service request requested by the service consumer through broker interface. It is the sum of successfully invoked Web services, failed Web services and bounced Web services without invocation. These three types of request can be represented by success access rate i.e., (S(R)), failure access rate i.e., (F(R)) and bounce access rate i.e., (B(R)). Access rate is denoted by R and it can be calculated as follows:

$$R = S(R) + F(R) + B(R) \tag{1}$$

1. **Bounce Access Rate**
It is the percentage of visiting and leaving without executing a particular web service by service consumer through broker interface.
To accurately measure the bounce access rate i.e., B(R), the total number of bounces for a Web service (b) will be divided by the number of times the Web service was called for execution (N) in any reason. It is denoted by B(A_r) and it can be calculated as follows:

$$B(Ar) = \frac{total\ \ number\ \ of\ \ bounced\ \ Web\ \ services}{failed\ \ Web\ \ services + successfully\ \ invoked\ \ Web\ \ services + bounced\ \ Web\ \ services}$$

$$B(R) = b/N \tag{2}$$

Therefore, the percentage of bounce access rate B(Ar) can be calculated as:

$$B(R) = b/N * 100 \tag{3}$$

2. **Failure Access Rate**
It is the ratio between the number of times a web service request failed (f) to perform its operation for any reason and the number of times the Web service was called for execution (N), i.e. unsuccessful executions/called for execution. It is the relationship between the number of times the Web service failed after the execution and the number of times the web service is successfully invoked. It is denoted by F(R) and can be calculated using the following formula:

$$F(Ar) = \frac{failed\ \ Web\ \ services}{failed\ \ Web\ \ services + successfully\ \ invoked\ \ Web\ \ services + bounced\ \ Web\ \ services}$$

Therefore, the percentage of failure access rate F(Ar) can be calculated as:

$$F(R) = f/N * 100 \tag{4}$$

3. **Successful Access Rate**
It is the ratio between the numbers of times a web service is successfully (s) invoked to perform its operation and the number of times the Web service was called for execution

(N), i.e. successful executions/called for execution. It is the relationship between the number of times the web service is successfully invoked and the number of times the Web service was called for execution. It is denoted by S(R) and can be calculated using the following formula:

$$S(R) = s/N * 100 \tag{5}$$

B. Reliability

Reliability is the probability in which the provider correctly answers a request within a maximum expected time. It is measured as the number of success request divided by the number of request. It is denoted by W_r and can be expressed as follows:

$$W_r = S(R) \tag{6}$$

C. Availability

The probability that the web service is in its expected functional condition and therefore capable of being used in a stated environment. Availability deals with the duration of uptime for Web service operations. It is often expressed in terms of up-time and down-time of web service. Up-time refers to a capability to perform the task and downtime refers to not being able to perform the task. It is dented by W_A and can be expressed as follows:

$$W_A = S(R)/N \tag{7}$$

D. Response Time

Response time is the total time duration spent between the request (Q_{sc}, Q_{sb}) and response (P_{sbr}, P_{sp}) for a particular Web service from the side of service consumer broker and service provider. It is denoted by W_{RT} and can be calculated as:

$$W_{RT} = (Q_{sc} + Q_{sb}) - (P_{sb} + P_{sp}) \tag{8}$$

E. Effective Service Access Time

The total time required to process the consumer request for particular service through broker. It is denoted by T_{ESA}. The broker access time (T_{BA}) and service access time (T_{SA}) can be evaluated through broker for T_{ESA}. T_{BA} and T_{SA} can be calculated as:
T_{BA} = Q_{sc}– P_{sb}
T_{SA} = (Q_{sc}+ Q_{sb}) - (P_{sb} + P_{sp})
Therefore,

$$T_{ESA} = (F(R) * T_{BA}) + (F(R) * T_{SA}) \tag{9}$$

V. Algorithm for evaluating QoS Parameters

An algorithm for monitoring and evaluating of different non-functional parameters and overall quality score at broker's operating environment is presented in this section. With the help of proposed algorithm, the broker can monitor the non-functional parameters of each Web service before publishing on host location and the consumers of service can retrieve up to date Web services with QoS information. In this algorithm, the invoke and monitor procedure invokes the particular web service for a specified time period for finding the number of successful request (success_request), failed request(failed_request) and bounced request (bounced_request). The obtained values of success_request, failed_request and bounced_request are further be used to construct the values of other non- functional parameters. total_request is the count of total number of request arrived for a particular Web service. success_rate, failed_rate and bounced_rate are the rate of percentage of successful, failed and bounced request. The update_old_QoS_dataset updates the old QoS data with the new QoS data after every invocation.

The algorithm for the monitoring and evaluation of proposed QoS parameters performed at brokerâĂŹs operating environment is as follows:

Procedure: MONITORING_WEB_SERVICE (WebServiceName, WebServiceUrl, TotalService).

Given WebServiceName is the name of Web service to be invoked and WebServiceUrl is the destination URL where the service is actually located. TotalService is the total number of QoS parameters used for the evaluation of average score of all QoS parameters.

Input: WebServiceName, WebServiceUrl, TotalService
Output: successful_request,failed_request,bounced_request, total_request, success_rate, fail_rate, bounced_rate, reliability, availability, response_time.

```
[Initialize]
successful_request = 0
failed_request = 0
bounced_request = 0
[Retrieve old successful request, failed request and bounced
request from Web service QoS dataset]
Read&Store(WebServiceName,old_succ_request,
old_failed_request,old_bounced_request)
[Invoke the specified Web service]
If invoked_success = true then
successful_request = successful_request + 1
Return
If invoked_failed = true then
failed_request = failed_request + 1
Return
If invoked_bounced = true then
bounced_request = bounced_request + 1
Return
[Evaluate other parameter]
tot_request =
success_request + failed_request + bounced_request
s_rate = success_req / tot_req
f_rate = failed_req / tot_req
b_rate = bounced_req / tot_req
reliability = s_rate
```

```
availability = s_rate / (s_rate + f_rate)
response_time = avg (resp_time)
update_old_QoS_Dataset (success_request, failed_request, bounced_request, tot_request, success_rate, failed_rate, bounced_rate, reliability, availability, response_time)
Print availability, reliability, response_time
[Finished]
Return
```

Procedure: EVALUATE_WS(WSName)

Given WSName is the name of Web service, WSUrl is the destination url of Web service and N is the number of QoS parameters. Procedure Evaluate_WS shows the list of similar Web services that fulfills the required functionality. This procedure evaluates the QoS parameters for each service s in service list service_list.

Input: Web service name
Output: Web service with QoS Read N.

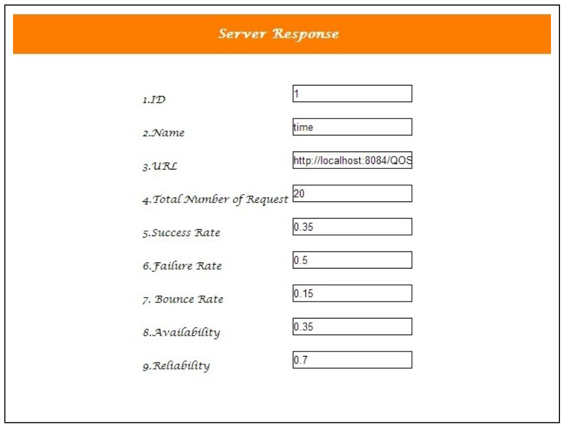
```
[Select Web service from service_list for QoS evaluation]
For each service s in service_list
Begin
CALL MONITOR_WEB_SERVICE(s)
End
```

The proposed algorithm can be helpful in evaluating the different non-functional parameter values through access rate and provide the aggregated QoS score of each Web service during service discovery and publish. The obtained quality score can be used for ranking and selecting the appropriate service for composition which is having highest score. In spite of the advantage, the proposed algorithm is unable to consider the non-functional parameter values of different units.

VI. EXPERIMENTAL RESULTS

The performance of the QoS parameter can be analyzed by the set similar web service function stored in the database. These web services are collected from the different services providers. The databases for the services are created in MySQL, which includes several services with their ID, name or keyword, and URLs. These web services are several time for a period of time. All the parameters of the web services are tested initially to determine the total rate, success rate, bounce rate, availability, reliability and response time based upon the request and response of the each web services.

An interface for web services is developed as show in figure 5. This is implementation for monitoring the composed web services and evaluates this service for the estimation of quality parameters using the evolution mechanism. Based on the customer needs, it estimates quality of web service. If the customer chooses the time or cost as important, then the mechanism fetches the time or cost as keyword for those particular services which calculate the parameter based on customer needs time.


Server Response

1.ID	1
2.Name	time
3.URL	http://localhost:8084/QOS
4.Total Number of Request	20
5.Success Rate	0.35
6.Failure Rate	0.5
7. Bounce Rate	0.15
8.Availability	0.35
9.Reliability	0.7

Figure 5: Interface for web services

VII. CONCLUSION

Evaluation of QoS parameters of a composed web service can be achieved by fair computation of QoS of every component service for providing trustable and best Web service. The overall performance of the composition can be affected by invoking a service with low quality. In this paper, access rate is related to accessibility and reputation that can help in assuring the quality of the selected Web services for composition. An algorithm is also presented for monitoring and evaluation of proposed parameters which is performed by trusted third party broker after registering a Web services. With the help of these parameters, the broker is able to evaluate the services according to their quality score which used for the service consumers always access the best Web services with updated QoS information. As future work, it would be ideal to make our monitoring system work at runtime with the Web service execution engine. In this way, the provider could take statistical results and correct any violations very quickly and without any mediators. Another interesting direction is to focus on the corrective actions after a violation is detected. This can be performed by another management service that would take as input the condition evaluation results and then, if appropriate, would try to make the service compliant with the SLA document. All this procedure would be ideal to be distributed. And also included a new framework based on ontology to enhance and optimize semantic web service discovery without changing their existing specification and implementation of standard UDDI interface using a broker. It is allowed to the client-side software can transparently connect. By using ontology concepts, the scope of QoS property is extended and can make more accurate and recallable results. We intend to solve the problem of optimization of cost for the service provider based on the SLA agreement.

References

[1] P. Leitner, W. Hummer S. Dustdar. Cost-based optimization of service compositions. *IEEE Transactions On Services Computing*, 6(2), 2013.

[2] P. Leitner, S. Dustdar, B. Wetzstein, F. Leymann. Cost-based prevention of violations of service level agreements in composed services using self-adaptation. *IEEE Transactions on Services Computing (TSC)*, 2012.

[3] Maya Rathore and Ugrasen Suman. Evaluating qos parameters for ranking web service. *IEEE International Advance Computing Conference (IACC)*, 7(8):1267–1274, 2012.

[4] L. Bodenstaff, A. Wombacher, M. Reichert, and M.C. Jaeger. Analyzing impact factors on composite services. In *Proceding IEEE International Conference Services Computing*, 2009.

[5] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann. Runtime prediction of service level agreement violations for composite services. In *Proceding Third Workshop Non- Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC '09)*, 2009.

[6] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar. Monitoring, prediction and prevention of sla violations in composite services. In *Proceding IEEE International Conference Web Services (ICWS '10))*, 2010.

[7] L. Juszczyk and S. Dustdar. Script-based generation of dynamic testbeds for soa. In *Proceding IEEE International Conference Web Services (ICWS '10)*, 2010.

[8] R. Jurca, B. Faltings, and W. Binder. Reliable qos monitoring based on client feedback. In *Proceding 16th International Conference World Wide Web (WWW '07)*, 2007.

[9] Y. Zhang, M. Panahi, and K. J. Lin. Service process composition with qos and monitoring agent cost parameters. In *Proceding IEEE 10th Conference E-Commerce Technology and the Fifth IEEE Conference Enterprise Computing, E-Commerce and E-Services*, 2008.