INTERNATIONAL JOURNAL OF SOFT COMPUTING

MEDWELL PUBLICATIONS SINCE 2000

Consumer Centric Versioning Recommendation Engine for Web Service-Oriented Revisions

[1]M. Paul Arokiadass Jerald and [2]K. Vivekanandan
[1]Manonmaniam Sundaranar University, Tirunelveli, India
[2]Pondicherry Engineering College, Puducherry, India

Key words: Service versioning, service compatibility, pearson coefficient of correlation, cosine similarity, Quality of Service (QoS), consumer rating, Consumer Centric Versioning Recommendation Engine (CCVRE)

Corresponding Author:
M. Paul Arokiadass Jerald
Manonmaniam Sundaranar University, Tirunelveli, India

Page No.: 86-95
Volume: 14, Issue 5, 2019
ISSN: 1816-9503
International Journal of Soft Computing

Abstract: The evolution of web services brought a key challenge to the service oriented paradigm. The QoS expectation of consumer is increasing with every web services. The providers are competing with other web services in the point of business. In order to fulfill the consumer satisfaction, the services or products require frequent improvement by providing more functionalities and modifications. Each change introduces a new version of web service. The service consumers are independent with the service providers. The service providers cannot control or even force the service clients to enforce the particular services. Versioning is an important method to be continued by service developers to yield several versions of the aforementioned service to handle linearly at the concordant time. Since, there is no common versioning mechanism to satisfy the user expectation and provide business aspect expectation, we propose a Consumer Centric Versioning Recommendation Engine (CCVRE) which recommends the service provider to upgrade their QoS of web services. Our proposed method considers QoS parameters and feedback of customers about the versioning needed for the web service. We have applied our approach in research with real-world dataset. Our results show that our versioning recommendation engine supports the service provider in the view of business aspect.

INTRODUCTION

The evolution of web services brought a key challenge to the service oriented paradigm. In order to fulfill the consumer satisfaction, the services or products require frequent improvement by providing more functionalities and modifications. The service provider requires competing with other relevant services by gratifying the consumer's requirements. The modified web services need another name to distinguish with existing web services. This is called as web service versioning. Web service evolution requires good strategies to manage versions of web services. Any functional change or semantic change deployed in the code of web service may create the new version of web service. The modifications could be in the form of adding new functionalities to an existing web service or alteration of existing operations in a web service. The clients are confused with multiple similar services and multiple versions. The provider also suffers in knowing the mindset of client's new requirements and their expectation. They have no opportunity to use all the similar web services for their particular task. There arises a necessity of bridge between the web service end user and service provider. Present Simple Object Access Protocol (SOAP) (Box *et al*., 2000), Web Services Description Language (WSDL) (Curbera *et al*., 2002) and Universal Description, Discovery and Integration (UDDI) (Curbera *et al*., 2002) may not support explicitly for the web service versioning. So, we need an effective service versioning management to balance the user QoS level expectation and provider level issues. A good service versioning management enables service provider to enhance services and manage the deployment of multiple service versions in a cost effective and productive manner.

A good web service versioning management should have the ability to consume the current web service version that satisfies the user's requirements. There should not be more deviation when recommending similar web services and it should meet the user expectations. The multiple versions of the same web service should run all together at the same time and the criticized services should be avoided. Due to the incompatibility issues with web service versioning, the consumed services suffer with non functioning properties and affects future versions. Typically, the traditional works have developed approaches based on traceability changes (Alam *et al*., 2015), time based versioning and checkpoint based versioning (Ghosh *et al*., 2018) methods for addressing this problem which determined the changes done with the web services centered in the viewpoint of service providers. But it has the major problems in understanding critical relationships and semantic dependencies. In our former work (Jerald and Vivekanandan, 2018) we reviewed various versioning guidance of service oriented architecture, lack of interrelation of services, compatibility issues and version maintenance. We assured to develop web services with different versioning standards and make them to interoperate with common efficient manner and to use metrics to quantify the influence of changes with web services. This research work aims to develop a new web service versioning method by considering the valuable feedback values of consumer and web service similarity measures. The input into this research paper contains:

- We propose a new web service versioning approach to the providers that unifies the feedback of service consumer and their demands
- We explicitly propose the concept of "Expectation based versioning" and develop a new versioning recommendation engine in a direct and demanding way
- We formulate our framework as a regularized multiple-rating problem and propose easy way for web service versioning
- We theoretically show that the proposed versioning approach satisfies both provider and the service consumer, overcoming a common drawback of the existing algorithms.
- We implemented broad experiments to assess the recommended versioning approach with controlled learning. The outcomes reveal its uniqueness and loftier performance

Literature review: In this study, the existing approaches that are used for web service versioning in the perspective point of service providers are surveyed with its advantages and disadvantages. Gebhart (2013) recommended a method to measurethe quality of service interfaceand service component belonging to service layer. This research considered the consumer satisfaction with the service in the web service design developing scenario. It introduced the concept of quality checker according to the user expectation. The metrics were introduced based on customer feedback. This model is only suitable for initial development of alpha version. This work did not provide any further idea for next upcoming web service versioning. Feng *et al*. (2013) recommended service to manage a registry for service evolution. It focused on the service changes based on time associated with it. It also provided an alert management system when the evolution takes place. The minor changes may not influence the alerts to the user and provider. But the major changes, alerts the provider and user to adopt for the next version of web service. This approach lacks with checking the evolution registry whenever the user uses the services. It makes comparison overhead to the system. Zuo *et al*. (2014) introduced a new model to analyze the changes made with web services in a centralized model. This work loaded responsibilities to the service broker by watching the changes occurred in services. The idea behind this concept is good enough which is centralized. But it is very hard to maintain the centric design in the distributed environment. Almalk and Shen (2015) dealt with the compatibility issues with the new service version. It suggested checking the possibility of issues then rectifying and servicing deployment. The message will be transmitted to the provider about the service information. This approach recommended the provider about the standard of the user and their requirements. This work dealt with the compatibility possible issues with the clients and not versioning. Cai and Cui (2014) involved consumer expectation based service conformation associated with granular changes. The user requirements were collected and multiple granular level of changes was implemented with its semantic constraints. The matched services were returned to the customer based on the correlation of services. This work concentrated to provide best suited related services to the consumer. Juric *et al*. (2009) suggested an approach by providing appendage to WSDL and UDDI to help in the state of runtime and progress time versioning of service interfaces. The extension has been done with XML namespaces to maintain version number and service level versioning and WSDL extension for operation level versioning. This method is effective for some web services and not for all services. Potocnik and Juric (2013) suggested CEP to react withcomplex events specified in service interface. CEP manager controlled critical events through centralized supervising. CEA services are connected with adapters and engines. The registry manages the events and connects services. It enabled the definitions of complex event types and complex event sinks in the CEA interface and also supported the key principles of SOA architecture. CEP adoption brought more complexity with SOA architecture. Alam *et al*. (2015) evaluated traceability checking by dependency analysis graph-based techniques. Hierarchical coverage analysis for inner level process and inter level of service changes. It focused graph based horizontal and vertical changes and the impacts beyond different layers. The problem with the work was very difficult to understand critical relationships and semantic dependencies. Mwebaze *et al*. (2013) proposed class-based object versioning. The service related changes are determined and combined with sources. These changes are linked with the created data objects. A centralized repository to oversee the code objects. A connecting mechanism links versions to data objects regarding the changes. This method is not fully proved and might fail in some cases when applying a change that can lead to ambiguity and derive multiple similar results. Teixeira Filho *et al*. (2016) utilized semantic approach for Justification of Service-oriented architectures. It used web service relation based ontologies, formal semantic directions and requests in order to make simpler the passivity endorsement process. Version control policy could be controlled based on service releases and variants named as policy set to define the versioning. But this ontology based process list used in the evaluation could be extended to evaluate other critical areas in SOA.

Sohan *et al*. (2015) worked to handle evolution in Web API. The new changes made on Web API crash the applications which are dependent. But the previous versions of the APIs malfunctioned when they combine with web APIs. The temporal changes are considered with APIs which gives better notification. It mainly focused with limited Web API and not overall Web API. This is considered as the disadvantage of this study. Chiponga *et al*. (2014) suggested proxy centered service conversion model which translate the messages across the service consumer and service provider to match the web service version which was implemented by the user. It keeps the evolution of a service from one version to the next as transparent to the consumer. But the proxied web services suffer with large user traffic. Chavan *et al*. (2015) introduced a version aware query language, capable of querying dataset versions. The different versions were constructed in the graph based on its standard. The query traverses the graph and check the metadata and determines the feasible version by comparing various versions. It enforced high computational cost for more numbers of versions. The data was stored in compressed style which is difficult at the time of query fetching. Ciccarese *et al*. (2013) proposed Provenance Authoring and Versioning (PAV) for tracking the past events of web services, versioning authority through a simple message passing mechanism related to ontology descriptions. PAV worked in contrast with detailed process oriented approach by implementing snapshot based approach. It traced the technical results and the provenance properties which gives more detailed past events of information on the web. The disadvantage of this work is the vocabulary list that should be maintained in efficient manner.

The review deliberated that the traditional techniques are convenient for a specific view point and inconvenient with other point of view, especially with provider's business point. The suggested methods have the following limitations:

- Alteration of services might alter the formed contract of services
- Changes related to Implementation could change the contract which in will alter the clients
- The performance of web service is affected by versioning methods
- Difficult for extension and less adjustable
- The same versioning approach does not always provide the same result with every change

In order to solve these problems, this study aims to develop a new recommendation mechanism with using location based web service clustering and reputation score of consumers with the service.

MATERIALS AND METHODS

In this study, the detailed description about the proposed methodology is presented. The main motive of this study is to provide a versioning model to the web service provider subjected to the consumer's expectation. For this purpose, a Consumer Centric Versioning Recommendation Engine (CCVRE) is proposed in this research which aims to suggest a new versioning mechanism to the previous employed web services. The architecture of the proposed system is depicted in Fig. 1 which includes the following stages:

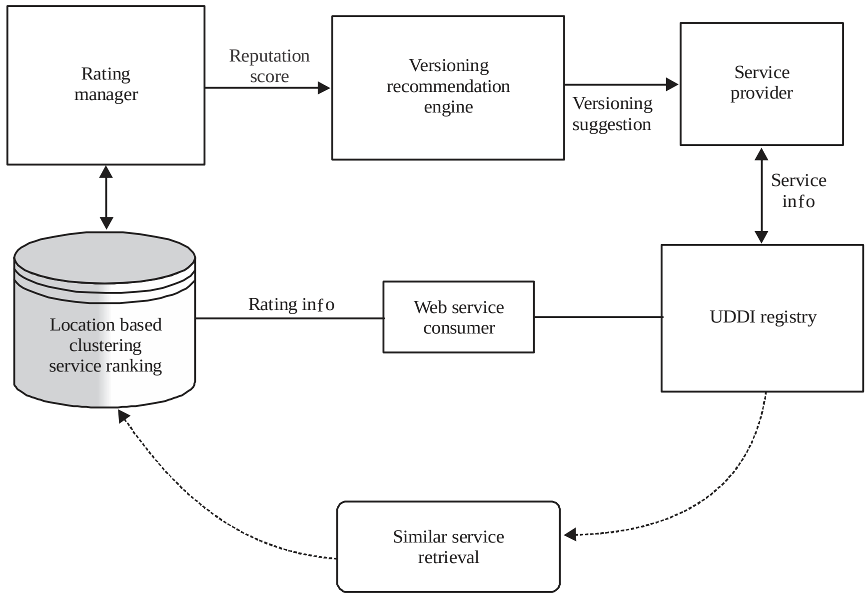

Fig. 1: Proposed architecture

- Similarity computation
- Location based web service clustering
- Ranking on services
- Recommendation for versioning

At first, the relevant web services related to the given web service are determined through cosine formula. This phase returns numerous number of similar web services. There will be many similar services for a specific task. We apply location based available web service grouping which reduce the quantity of similar web services. Among the web services, the similarity score is calculated with the use of Pearson Correlation Coefficient (PCC). We compare the given web service with the other related services and compute the similarity matrix. The users of same location might experience different with same service according to their network topology, Hardware configuration and available resources. These location aware similar services are clustered to mine the effective services. Then based upon the consumer feedback, the rating manager collects the rating information for the related web services. Then rating manager selects the most three web services which has higher QoS values and rating values. Versioning recommendation engine fetch this results and recommends the provider whether they need versioning or not.

Similarity computation: The proposed work initially implicates the cosine law in finding the related web services. The similarity metrics of web services could be extracted effortlessly from UDDI, SOAP, WSDL and XML schema. Every individual web service could be represented in tuple notation with the input metrics, Output value, precondition and effect. Web service = {I, O, P, E}. Where I represents Input metrics, O represents Output values, P represents Precondition and E represents Effects. Service ID and set of inputs of services are the input metrics of the tuple notation I. O represent set of outputs of service. P denotes Set of pre-conditions to trigger a service execution ofparticular service. E denotes set of effects received after the execution of particular service. Web service is represented as vector with having many parameters. In order to find the similarity between these two web service vectors we use cosine law which computes the similarity score in between -1 to 1. The similar web services have positive cosine value depends upon its similar operations. We consider the web service Y as consumer requested query and the web service X represent the other web service to be compared:

$$\text{WebService}_x = \{i_x, o_x, P_x, e_x\} \quad (1)$$

$$\text{Web Service}_y = \{i_{xy}, o_y, P_y, e_y\} \quad (2)$$

The cosine similarity score could be calculated with the use of the following equation:

$$\text{Similarity score}_{x,y} = \frac{I_x I_y + O_x O_y + P_x P_y + e_x e_y}{\sqrt{I_x^2 + o_x^2 + P_x^2 + e_x^2} * \sqrt{I_y^2 + o_y^2 + P_R^2 + e_R^2}} \quad (3)$$

If the cosine similarity score value is -1, then the analyzed web service is straight opposite service and if the value is negative then the compared service has high dissimilarity. If the value is 0 means the compared service has no similar match with the requested service. The matching criteria could be done based on the threshold

Table 1: An example of location based throughput matrix

User IP	Location	S1	S2	S3	S4	S5	Service on validation
12.46.129.15	US(37.79, -122.219)	0.938	15.267	21.978	7.968	7.874	7.025
122.1.115.91	Japan(36.0,-138.0)	2.341	10.928	15.957	5.602	5.586	26.086
128.10.19.52	US(40.424, -86.916)	2.886	17.699	25.751	9.090	9.132	8.72
128.10.19.53	US(40.424, -86.916)	2.309	17.621	25.751	9.090	9.049	8.196
128.111.52.61	US(34.432. -119.837)	1.091	15.936	23.346	7.662	8.368	11.811

positive value. We need high similar web services in order to implement versioning. Here, we consider the threshold value as <0.8 which associate high similar web services.

Location based web service clustering: Table 1 location is represented by longitude and latitude where the available web services are limited for the users. The service consumers belong to the same location could access some sort of web services with high QoS parameters. Contrast to it, the same location users could not able to access some sort of web of services with effective QoS parameters. This could be due to its network based QoS, geographical issues and legal aspects. We derive the location based sample users with respect to the parameter availability. Availability is the QoS parameter which represents the obtainability of a web service for current access. This concerned value should be maximum in order to access the web service effetely. If the value of availability is low, then the particular web service lacks with reliability. The appended Time-To-Repair (TTR) value indicates the additional time required for the self-healing. Generally, TTR value should be low for the proper accessing of web service. We here discover the optimal web services with the identified set of similar web services resulting from the previous phase. The determined web services will be subset of the identified similar web services which is shown in Fig. 2. We here implement location kindliness rate to determine location based best services:

$$\text{Location kindliness rate} = \frac{\alpha}{\beta} \tag{4}$$

Where:
α = No. of best web service availled on that location
β = No. of similar web services for the perticular task

The location kindliness rate to a particular location has similar QoS usage experience. Every user belong to exactly one particular location. The location bound could be estimated based on its kindliness rate. If the kindliness rate deviates more for the same web service, then that location user aggregates with other location bound. Figure 3 explains the work flow of the proposed approach. At each iteration, the newly observed QoS data are collected to update the model. The QoS record could be checked in the time interval of 10 min. The updated QoS values are send to the PCC and if there is no change

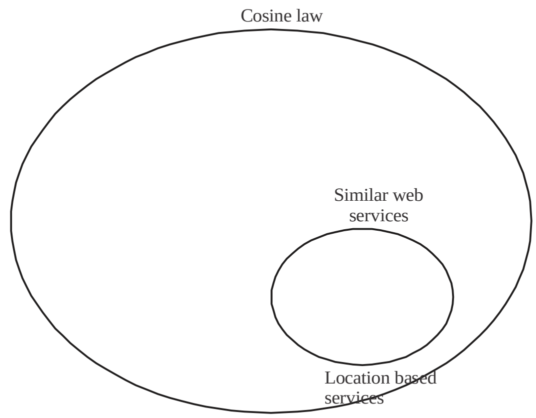

Fig. 2: Subset determination in similar web services

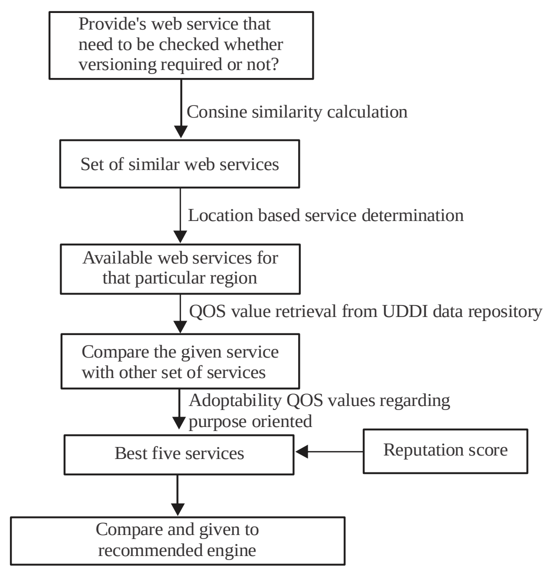

Fig. 3: Flow of the proposed approach

we use the old record for that service. The following algorithm explain the relation between location user service, QoS values changes in Time T.

Algorithm 1: User service matrix

Input: Sequentially observed QOS stream: $\{(t, U_i, s_j, R_{ij}(t)\}$
Output: Online QoS prediction results: $\left\{\hat{R}_{ij}(t_c)\middle|I_{ij}(t_c)=0\right\}$

Step 1: Repeat
Step 2: Collect newly observed QoS data;
Step 3: if a new data sample $\{(t, U_i, S_j, R_{ij}, (t_c))\}$
Step 4: Set $I_{ij}(t_c) \leftarrow 1$;
Step 5: $R_{ij}(t_c)$ observed at current time slice t_c

Step 6: If U_i is a new user or sj is a new service then
Step 7: Randomly initialize $U_i \varepsilon R^d$, εR^d
Step 8: Initialize $e_{ui} \leftarrow 1$, or $e_{sj} \leftarrow 1$
Step 8: Update the user and service matrix
Step 9: else
Step 10: Randomly pick a historical data sample and update if the time interval exceeds to limit.
Step 11: Update the matrix
Step 12: eles
Step 13: Set $I_{ij}(t_c) \leftarrow 0$
Step 14: Matrix update

$$L_k = \frac{1}{2}\sum_{i=1}^{mk}\sum_{j=1}^{nk} I_{ij}^k \left(R_{ij}^k - \left(U_i^k\right)^T S_j^k\right)^2 + \frac{\lambda_u^k}{2}\left\|U^k\right\|2_F + \frac{\lambda_S^k}{2}\left\|S^k\right\|2_F \tag{5}$$

Correlation calculation of web services: The correlation calculation can be determined with our modified formula that was derived from pearson correlation. This approach uses the QOS values of the services which can be easily retrieved from UDDI. Thus we have the QOS values of the given provider's query service and the comparable set of location based services. Every service's QOS values are compared with the PCC based derivative formula. generally, pearson correlation measurement is one of the best statistical measures of association between the two variables (X, Y) of interest. It provides the information regarding the degree of the association, association and direction of the relationship (P x, y) based on the covariance (cov). But here we derive a formula that represent a QOS related magnitude value. Then, the detailed formulation of mean and standard deviation, the Pearson Correlation Coefficient (PCC) is modeled and its derivative formulation used as the reference for this proposed work to compute the correlation between the given web service and location based availed web services based on the QoS parameter. Generally Quality of Service (QoS) changes which depends upon the web services. The customer satisfaction factor may differ for different services. But we can cluster similar web services and metrics could be applied to find the correlation analysis. We apply performance, reliability, scalability, capacity, robustness, exception handling, accuracy, integrity, accessibility, availability interoperability and security as QoS requirements for web services. The performance of a web service characterizes the fast in means of throughput, response time, latency, execution time and transaction time. The reliability is defined as stability to continue the service connection. Accuracy means that the functions of web service is working properly or not. It could be estimated based on the error rate caused by the web service. The integrity is one of the security related QoS parameter which is associated with unauthorized access or data or program modification.

Inter-operability is one of the essential QOS value which consider dependable operations related to the web service. Web services should be interoperable between the different developments environments used to implement services, so that, developers using those services do not have to think about which programming language or operating system the services are hosted on. Web services should be provided with the required security by providing authentication, authorization, confidentiality, traceability/auditability, data encryption and non-repudiation. The correlation between the given web service and availed web services can be formulated based on the QoS parameteras follows:

$$\text{Response\_time}_{PCC} = \frac{\sum_{i=1}^{n}(\text{Rest1}_i - \overline{\text{Rest}})(\text{Rest2}_i - \overline{\text{Rest}})}{\sqrt{\sum_{i=1}^{n}(\text{Rest1}_i - \overline{\text{Rest}})^2}\sqrt{\sum_{i=1}^{n}(\text{Rest2}_i - \overline{\text{Rest}})^2}} \tag{6}$$

Where, Rest1, Rest2-Response time values for given and availed comparable web service. n-number of similar availed web services resultant from cosine similarity measure and having high location kindliness rate. $\overline{\text{Rest}}$ represents the average response time value of that category of web service submitted by all users:

$$\text{Throughput}_{PCC} = \frac{\sum_{i=1}^{n}(\text{Thres1}_i - \overline{\text{Thres}})(\text{Thres2}_i - \overline{\text{Thres}})}{\sqrt{\sum_{i=1}^{n}(\text{Thres1}_i - \overline{\text{Thres}})^2}\sqrt{\sum_{i=1}^{n}(\text{Thres2}_i - \overline{\text{Thres}})^2}} \tag{7}$$

Thres1, Thres2-Throughput values for given and location based availed comparable web services. n-number of similar location based availed web services resultant from cosinesimilarity measure and having high location kindliness rate. $\overline{\text{Thres}}$ represents the average throughput time value of that category of Web service submitted by all users:

$$\text{Availability}_{PCC} = \frac{\sum_{i=1}^{n}(\text{Avail1}_i - \overline{\text{Avail}})(\text{Avail2}_i - \overline{\text{Avail}})}{\sqrt{\sum_{i=1}^{n}(\text{Avail1}_i - \overline{\text{Avail}})^2}\sqrt{\sum_{i=1}^{n}(\text{Avail2}_i - \overline{\text{Avail}})^2}} \tag{8}$$

Avail1, Avail2-Availability values for given and location based availed comparable web service. $\overline{\text{Thres}}$ represents the average availability time value of that category of web service submitted by all users:

$$\text{Accuracy}_{PCC} = \frac{\sum_{i=1}^{n}(\text{Accur1}_i - \overline{\text{Accur}})(\text{Accur2}_i - \overline{\text{Accur}})}{\sqrt{\sum_{i=1}^{n}(\text{Accur1}_i - \overline{\text{Accur}})^2}\sqrt{\sum_{i=1}^{n}(\text{Accur2}_i - \overline{\text{Accur}})^2}} \tag{9}$$

Accur1, Accure2-Accuracy values for given and availed comparable web service. $\overline{\text{Accur}}$ represents the average accuracy value of that category of web service

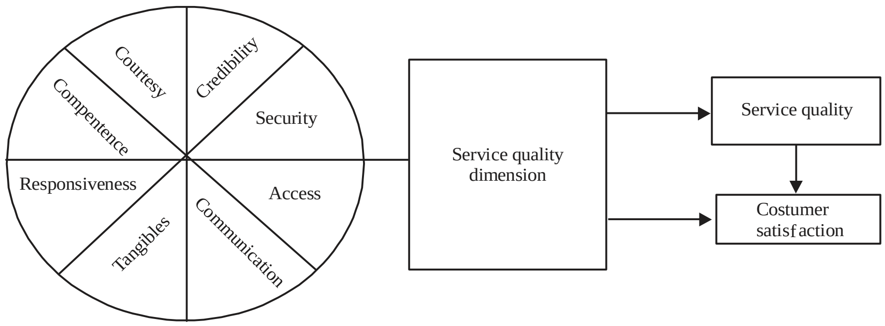

Fig. 4: Service quality dimension in SERVQUAL

Table 2: Service's QoS information

Services	Response time (sec)	Availability (%)	Throughput (m sec)
S1	0.05	90	500
S2	0.02	97	500
S3	0.05	99	800
S4	0.05	98	600

submitted by all users. The QoS information matrix for three parameters are shown in Table 2. Similar to the way the other QoS based formulation can be calculated to estimate the requirements needed to update the given web service with the existing other best related services to that particular location. Here, it is necessary to deal with the QoS parameters in two sections. The value to be increased and the value to be decreased. The throughput, security, reliability, response time, exception handling and accuracy values should be maximum for an efficient web service and some parameter value like bugs should be minimum for an effective web service. We will consider the above formulated four QoS parameters to find the effectiveness of web service:

$$\begin{aligned}\text{Ews} = \text{Response\_time}_{\text{PCC}} + \\ \text{Throughput}_{\text{PCC}} + \text{Accuracyp}_{\text{PCC}}\end{aligned} \tag{10}$$

The PCC of positive QoS parameters have to be aggregated to evaluate the web service. And the negative QoS parameter values have be considered separately by aggregating the values.

Consumer feedback rating: Trust and reputation mechanisms works universally with open minded framework giving preference to the consumer grievances. Trustworthiness is generally judged based on the characters such as reliability, proficiency, steadfastness of a web service. It is independent and signifies a collective estimation of a group of people/agents while trust is personalized and subjective reflecting an individual's opinion. In order to collect reputation and consumer feedbacks, a central QoS registry (Mehdi *et al*., 2015) is maintained in the UDDI. It provide QoS related information about the web service and consumer preference with it.

There is a relationship between customer satisfactions and service quality. In order to evaluate the service quality various stratified sampling methods (Hill and Brierley, 2017) are used by considering dimension of services. One of the popular sampling approach is (Iihan *et al*., 2015) SERVQUAL which consider five dimension of service (Fig. 4).

Service quality dimension has significant relationship with service quality and customer satisfaction. For our research, we use throughput and response time matrix for the calculation of reputation score. Generally the star based ranking method is applied for all services that inherits all the dimension quality of corresponding services. Here, we consider throughput and response time for the representation of reputation score which is available in the wsdream dataset. Final reputation score (Zhou *et al*., 2016) is calculated for the sort of computed web services in the QoS registry. The same user may rate a service differently according to their different necessities. Totally we get PCC correlation analysis data and overall reputation score for the given and availed web services.

Versioning recommendation: From the former phase we get two matrix results which is QoS value matrix for location based availed web services for the given web service. And user's service rating matrix for that web services. From this two matrices, three possibilities can be inferred for versioning recommendation.

Case 1; The given web service is new application that never exist before: In this case we may not get the similar web services to compare. And there will not be average QoS value for that type of web service. The given new web service is considered as best and the version could be done according to the wish and needs of service provider. Based on the QoS values and rating, the provider can improve their service. They can analyse the weakness of the service and strengthen it by new versioning. This could be major or minor versioning. The major versioning dealt with additional functionalities which need to check compatibility issues and minor versioning dealt with small changes in the services that need not check compatibility issues.

Case 2; The given web service has very low QoS values: In this case the given web service self-estimate, additional functions and other modification are required related to QoS values and consumer rating analysis. This self-estimation supports the provider to enhance the QoS of web service and versioning. Here, the comparison could be done from the analysed best three similar web services which has higher QoS values and rating. For example, if the given web service response time value seems lower to other best web services means we can recommend the provider to improve the response time to that location. Thus the major and minor versioning could be recommended.

Case 3; The given web service has very high QoS values: In this case the given web service is considered as best and the versioning could be done according to the alterations. The versioning is recommended here based on the additional functionalities and customer requirements.

RESULTS AND DISCUSSION

Experiments: We directed testing for our proposed method in the measure of computation time, reliability and performance. There is no universal versioning recommendation engine which tries to combine the service consumer and service provider. We here studied QoS values for the web services and customer feedback rating information from the UDDI central registry.

Experiment setup: In order to estimate our method we select a real-world web service QoS dataset named WSD ream (Zheng and Lyu, 2008), A Distributed Reliability Assessment Mechanism for Web Services (WS-DREAM), permitting users to accomplish web services reliability assessment in a concerted fashion. The different users from different locations can support the experiment and they can also participate their side by providing test cases to the centralized server. It is very difficult to design the model for the web service assessment from various part of the world in real time. This dataset has >2 million real world QoS web service records. But this dataset is limited only with two QoS entities response time and throughput. This data were collected from 340 consumers with 6000 web services. The dataset has the user's location and provider's information. The experiments were conducted to assess web services with effective QOS values. Each experiment consisted of a composition request with service classes, service candidates per class and global QoS constraints. We varied these parameters and each unique combination of them represented one experiment. The number of QoS attributes was set to two.

Service selection: This experiment exhibits the availed services which best meets a customer's requirements. By implementing cosine similarity measure the similar services are identified. With our data set we determined 27 similar services. The standard deviation is justified by the given web service. If we increase the standard deviation we get unrelated web services as a result. This degrade our service data set. So, the value of standard deviation set to minimum value to get only relevant services.

Location based service availability: Among the similar 27 services, the retrieved similar services were reduced when implementing location based user and services. This varies from location to location. We selected the location US which has 5 similar services. This is due to the restriction and bugs of web service.

PCC analysis: The pearson coefficient analysis derivative formulation is used to evaluate the QoS of the availed services. We consider Throughput, response time and availability.

Throughput: The QoS parameter throughput is calculated for the 5 similar web services. The values of throughput is analysed from Table 1 for the determined web services. These five web services has high throughput compared to other web services. Among these five web services the best three web services are selected to compare the given version needed web service. If the given web service is lack with the identified web service then the provider is recommended to upgrade the throughput of that web service I that region. Similar to this, the QoS value of response time for the five best similar services sre retrieved from UDDI. These response time of services are compared with the given service and if the given web service has low response time means, it is recommended to the provider to update the service response time. This also could be the cause of server placement. The effective analysis should be done for the versioning upgradation. The modification should be analysed whether it is major or minor versioning. Both forward and backward compatibility issues have to be analysed before versioning.

Comparison of computation time results: In this experiment, we compared the computation time of our CCVRE approach with those of the other approaches VGMIP, VMIP (Wang *et al*., 2015). As shown in Fig. 5, we found that the computation time consumed by CCVRE

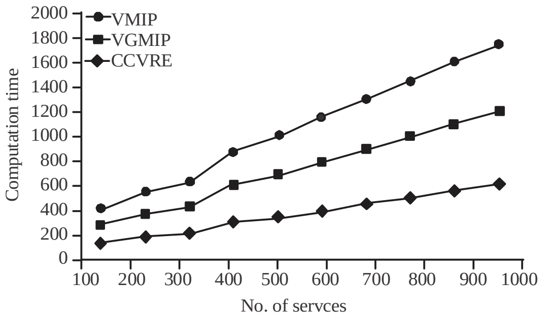

Fig. 5: QOS values of computation time

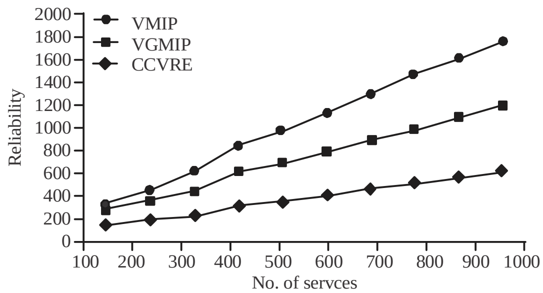

Fig. 6: QoS values of reliability

was always lower than that of the other two approaches with increasing numbers of service candidates. This means that CCVRE can significantly reduce the time cost of service selection because its service search space is the smallest of the approaches. This is because of the location based reduction.

Comparison of reliability results: In this experiment, we evaluated QoS value of reliability. It is an important QoS parameter which signifies the stability of a web service. As shown in Fig. 6, we found that no matter how many service candidates were there, the reliability of CCVRE was always higher than VMIP and VGMIP. These experimental results indicate that VMP can effectively avoid QoS uncertainty from service selection because the reliability of such services is very high. Thus, by using variance to monitor a service's historical QoS transactions, CCVRE effectively identifies which services have large variances in QoS and can then prune them.

CONCLUSION

In this research, a new versioning recommendation engine has been proposed to the web service providers. It briefly analyses the various issues with versioning mechanism. It concentrates on the reputation score for the versioning recommendation. This research assures enhanced QoS level web service to the consumer. Our work is experimented with the WS-DREAM dataset which has real world web services and users. The cosine law identifies the similar web services from the large dataset. Then the location based web service clustering has been done with the use of kindliness rate. The availed web services are identified which are the subset of similar web services. The QoS correlation analysis is implemented with the use of pearson coefficient analysis derivative formula. The various QOS parameters are calculated through a matrix. The customer based feedback rating is considered for that web services. These two metrics are used to determine whether the versioning is needed or not. The prediction of Consumer Centric Versioning Recommendation Engine (CCVRE) provides accurate versioning strategy to the providers. CCVRE is very useful in getting best services in the consumer point of view as well as best business level protocol in the point of provider side.

RECOMMENDATION

In future research we can implement this versioning mechanism to the real time web world.

NOTATION

t	=	Time specific
U_i	=	Service user
S_j	=	Set of service
R_{ij} (t)	=	QoS value observed by U_i user when invoking Service S_j

REFERENCES

Alam, K.A., R. Ahmad, A. Akhunzada, M.H.N.M. Nasir and S.U. Khan, 2015. Impact analysis and change propagation in service-oriented enterprises: A systematic review. Inf. Syst., 54: 43-73.

Almalki, J. and H. Shen, 2015. A lightweight solution to version incompatibility in service-oriented revision control systems. Proceedings of the 24th Internatinal Australasian Conference on Software Engineering (ASWEC 2015) Vol. 2, September 28-October 01, 2015, ACM, Adelaide, Australia, ISBN: 978-1-4503-3796-0, pp: 59-63.

Box, D., D. Ehnebuske, G. Kakivaya, A. Layman and N. Mendelsohn *et al*., 2000. Simple object access protocol (SOAP), version 1.1. W3C Note 08 May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

Cai, H. and L. Cui, 2014. MultiGranular: An effective service composition infrastructure for multi-tenant service composition. Intl. J. Multimedia Ubiquitous Eng., 9: 171-182.

Chavan, A., S. Huang, A. Deshpande, A. Elmore and S. Madden *et al*., 2015. Towards a unified query language for provenance and versioning. Proceedings of the 7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15), July 8-9, 2015, Edinburgh, Scotland, pp: 5-5.

Chiponga, K., P. Tarwireyi and M.O. Adigun, 2014. A version-based transformation proxy for service evolution. Proceedings of the IEEE 6th International Conference on Adaptive Science and Technology (ICAST), October 29-31, 2014, Ota, pp: 1-5.

Ciccarese, P., S. Soiland-Reyes, K. Belhajjame, A.J. Gray and C. Goble *et al*., 2013. PAV ontology: Provenance, authoring and versioning. J. Biomed. Semant., 4: 1-22.

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, 2002. Unraveling the web services web: An introduction to SOAP, WSDL and UDDI. IEEE Internet Comput., 6: 86-93.

Feng, Z., D.K. Chiu and K. He, 2013. A service evolution registry with alert-based management. Proceedings of the 2013 5th International Conference on Service Science and Innovation, May 29-31, 2013, IEEE, Kaohsiung, Taiwan, pp: 123-130.

Gebhart, M., 2013. Measuring design quality of service-oriented architectures based on web services. Proceedings of the 8th International Conference on Software Engineering Advances (ICSEA 2013), October 27-November 01, 2013, Venice, Italy, ISBN:978-1-61208-304-9, pp: 504-509.

Ghosh, A., R. Chaki and N. Chaki, 2018. CMS: Checkpoint-Based Multi-Versioning System for Software Transactional Memory. In: Progress in Intelligent Computing Techniques: Theory, Practice and Applications, Sa, P.K., M.N. Sahoo, M. Murugappan, Y. Wu and B. Majhi (Eds.). Springer, Singapore, ISBN:978-981-10-3372-8, pp: 471-482.

Hill, N. and J. Brierley, 2017. How to Measure Customer Satisfaction. 2nd Edn., Routledge, USA., ISBN: 9781138407855, Pages: 160.

Iihan, A., R. Mohlmann and W.G. Stock, 2015. Customer Value Research and Servqual Surveys as Methods for Information Need Analysis. In: Re-Inventing Information Science in the Networked Society, Pehar, F., C. Schlogl and C. Wolff (Eds.). VWH Verlag Werner Hulsbusch, Gluckstadt, Germany, ISBN:9783864880810, pp: 457-468.

Jerald, M.P.A. and K. Vivekanandan, 2018. A comprehensive review of versioning methods of service oriented architecture. Intl. J. Comput. Eng. Technol., 9: 83-95.

Juric, M.B., A. Sasa, B. Brumen and I. Rozman, 2009. WSDL and UDDI extensions for version support in web services. J. Syst. Software, 82: 1326-1343.

Mehdi, M., N. Bouguila and J. Bentahar, 2015. Trust and reputation of web services through QoS correlation lens. IEEE. Trans. Serv. Comput., 9: 968-981.

Mwebaze, J., D. Boxhoorn, I. Rai and E.A. Valentijn, 2013. Supporting dynamic pipeline changes using Class-Based Object Versioning in Astro-WISE. Exp. Astron., 35: 157-186.

Potocnik, M. and M.B. Juric, 2013. Towards complex event aware services as part of SOA. IEEE. Trans. Serv. Comput., 7: 486-500.

Sohan, S.M., C. Anslow and F. Maurer, 2015. A case study of web API evolution. Proceedings of the 2015 IEEE World Congress on Services, June 27-July 2, 2015, IEEE, New York, USA., pp: 245-252.

Teixeira Filho, H.M., L.G. Azevedo and S.W.M. Siqueira, 2016. IntelliGOV-A semantic approach for compliance validation of service-oriented architectures. J. UCS., 22: 1048-1071.

Wang, S., L. Sun, Q. Sun, X. Li and F. Yang, 2015. Efficient service selection in mobile information systems. Mob. Inf. Syst., 2015: 1-10.

Zheng, Z. and M.R. Lyu, 2008. Ws-dream: A distributed reliability assessment mechanism for web services. Proceedings of the 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), June 24-27, 2008, IEEE, Anchorage, Alaska, USA., ISBN: 978-1-4244-2397-2, pp: 392-397.

Zhou, X., D. Lin and T. Ishida, 2016. Evaluating reputation of web services under rating scarcity. Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), June 27-July 2, 2016, IEEE, San Francisco, California, USA., ISBN:978-1-5090-2629-6, pp: 211-218.

Zuo, W., A.N. Benharkat and Y. Amghar, 2014. Holistic and change-centric model for web service evolution. Proceedings of the IEEE World Congress on Services (SERVICES), June 27-July 2, 2014, Anchorage, AK., pp: 250-253.