IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

A Self-adaptive System reconfiguring a Composite Web Service for Emergency Medical Aid

Navinderjit Kaur Kahlon

Department of Computer Science, Guru Nanak Dev University, Amritsar, Punjab

Available online at: www.ijcseonline.org

Accepted: 21/Jan/2019, Published: 31/Jan/2019

Abstract- The web services run in a highly dynamic environment so, the most fundamental challenges in web services based software solutions is to manage QoS changes of their component web services at runtime. In order to make the composite web service adapt to these changes, a self adaptive system is proposed for web service composition. The distributed approach is followed at the client and the server side along with the runtime monitoring and adaptation of the component web services at the provider side. For the self adaptive systems to recover as quickly as possible, a way of performance prediction is proposed in this paper along with the case study and the performance of the system. The prototype is developed using Java, and the JADE platform is used for implementing software agents using hospital lookup case study. The experimental results show that the proposed solution has better performance for supporting self adaptive web service composition.

Keywords- Self-adaptive systems; Web services; Quality of Service (QoS) and Web Service Composition

I. INTRODUCTION

The service composition which integrates different web services provided from various service providers is becoming as one of the major issue in Service Oriented Computing. The service providers and consumers platforms are more and more being used to provide methods that make possible the combination of component web services to create the composed web services. Therefore, the self adaptive web service composition is critical to any emergency situation like in wake of disaster. The medical emergency services are especially vulnerable due to the failure or degradation of component web services in a dynamic environment. This affects the services offered to the patients. Therefore continuous availability of medical services is critical for any hospital or medical care facility. Though various fault tolerant solutions have been introduced in recent years, this paper addresses a self-contained and self-healing composite web service.

The composite web service not only support business processes within organizations but also across several organizations [11]. Thus the companies should provide the services which are of better quality among other web services. The web services which are slow can adversely affect the overall execution of the composite web service. Thus, monitoring and adaptation of its partner web services in a dynamic environment is needed while the execution of the composite web service, and the client should be notified as soon as possible for timely partner replacement decisions. Although, many solutions have been proposed to adapt a CWS to QoS changes of its partner web services [3], but run-time monitoring of partner web services, and then communication of the monitoring data to the clients still needs to improve. Moreover, adaptation solution should be custom made as different clients have different QoS requirements.

A CWS may execute successfully when all its partner web services are available and running at the expected QoS levels. However, if a partner web service fails to deliver; the CWS may have to be reconfigured to avoid its own failure or poor performance. The research challenge is to make this reconfiguration happen as soon as a partner web service fails or slows down. The composite web service should easily switch to a replacement service and should not affect the execution performance of the composite web service. The focus should be on finding a replacement of the faulty web service which may or may not be available in the repository at that time. Self autonomic systems ensure the self-management, self-configuration, self-optimization, self-healing and self-protection of the application if there is a need for replacement of the faulty web service [22]. Many existing solutions explore various mechanisms to ensure successful execution of a CWS when a partner web service fails to deliver. Notably the solutions follow backward recovery using checkpoints or otherwise [17], forward recovery using substitution as a last resort after multiple retries to use the resource fail. Web service substitutes are supposed to be made available by the existing provider, or client can start discovering substitutes when failure occurs. Service replication at the provider side is used as a mechanism to improve reliability of partner web services [16]. The

© 2019, IJCSE All Rights Reserved

combination of these strategies depending upon the context is explored in [5].

With all these problems in the mind, our proposed approach presents a solution for the composite web service to adapt itself. The key to our approach is to provide a self adaptive solution in which web services included in the CWS workflow are monitored using a distributed approach. Whenever, a QoS degradation is detected at the provider side, the client is notified, and then the client may choose to replace the partner service with an alternative from another provider, if its aggregate QoS is compromised.

In our previous work, a preliminary framework to monitor and adapt a CWS when quality of its partner web service(s) degrades [15] is discussed. This paper extends the framework to incorporate more QoS parameters such as throughput, availability and reliability. Also, a more elaborated experimental study to evaluate the framework is presented.
The rest of the paper is organized as follows: Section II decribes a review of the related work, Section III provides the proposed methdology. Section IV repersents the case study whereas, Section V illustrates the experimantal setup with Section VI provides the results and anaysis. Finally, Section VII concludes the paper.

II. RELATED WORK

This section describes the work done on the web service composition and self-adaptive system. The self adaptive systems ensure the quality of an application through analysis of runtime events at the time of execution. Run-time monitoring generates alerts whenever an unexpected event occurs. In run-time monitoring, events can be analyzed online - during execution of an application, or offline - after the execution has been terminated. Online monitoring is preferred over offline monitoring as it analyzes small and sufficient amount of data. It enables to handle the quality violations as soon as they happen. It also gives a chance of recovery once a problem has been detected, e.g., by terminating execution or returning to a stable state [15].

Monitoring results in triggering adaptation events. A monitoring module sounds an alert when it identifies an erroneous situation (after it has happened), and in reaction the system adapts to accommodate the change (e.g. replaces an unavailable service). The adaptation mechanisms (or reconfiguration) of a CWS can be reactive or proactive. In reactive adaptation, service manager changes configuration of the composition when an undesirable event has already occurred (an unavailable web service), whereas a proactive adaptation is based on prediction of occurrence of an undesirable event [18]. In this work, a proactive predictive approach is suggested that prevents invocation of partner web services with degraded QoS values.

A cost effective monitoring for monitoring and evaluating different monitoring strategies for service - based systems (SBS) is presented [12]. A novel approach, CriMon, calculates the criticalities of the different execution paths and component services in service - based systems to produce optimal monitoring strategy to be followed for cost of monitoring. A QoS monitoring framework (called SALMON) is designed to support entire service-based system's lifecycle [21].

After detecting violations in QoS values, the adaptation mechanism gets triggered. The effective runtime adaptation of service needs real changes in QoS of web services for timely and accurate decisions about -When to trigger adaptation action?, Which web service to replace in execution?, and Which candidate web service to select? [28]. The composite service should adapt itself as quickly as possible by performance prediction based on Semi Markov Model [10]. A proactive adaptation framework is proposed for service composition based upon prediction of the response time of a service using exponentially weighted moving average [8]. The framework adapts the composition if a service become unavailable or its response time degrades.

Various recovery strategies are analysed and to define a model to choose the best recovery strategy that is dynamically based on context information of the execution state is given in [5]. The recovery techniques are backward, forward, and replication based on detecting the impact on QoS if failure occurs.

The analysis of self-adaptive web service composition by reactive adaptation [11] classifies on use of variability models, context awareness, and multi agent approaches with detailed discussion on their limitations.

III. THE PROPOSED METHODOLOGY

This section presents an approach to address the self adaptive system working in a service oriented dynamic execution environment.

A workflow manager receives a client request; gets the corresponding abstract composition; selects corresponding concrete web services; dispatches agents to service providers to monitor the QoS behavior of the chosen web services for execution in composition; invokes the partner web services for preparing the results and responds back to the client.

Basically, a web service is defined by its operations and QoS attributes. Operations describe the functional capabilities of a service, and QoS attributes describe the quality with which functionality is delivered to the clients. There are a number of criteria to describe a web service's QoS e.g. execution time, throughput, reliability, availability, accuracy, and so on. The mathematical formulae to calculate the value of a QoS

© 2019, IJCSE All Rights Reserved

parameter for a given web service are specified below (as given in Table 1) are embedded in the framework and are calculated automatically at runtime. The observed values of QoS parameters (taken at runtime execution of framework) are stored in a log file which are used concurrently for comparing the real observed values and expected values (refer definition 2) of QoS parameters. The four QoS parameters used for dynamic monitoring of web services are shown in Table 1 below.

Table1. Formulae to calculate QoS Parameters

Execution Time	Response Time − Request Time
Throughput	$\frac{\text{No of completed requests}}{\text{unit time}}$
Reliability	$\frac{\text{Number of failures}}{\text{total requests per unit time}}$
Availability	$\frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$

MTBF in Table 1 represents the mean time between failures. Larger are the values of MTBF, better it is. MTTR represents the mean time to repair i.e. average time taken for repair of a faulty web service. The less the values of MTTR, better it is.

Majority of the QoS attributes, discussed here, have positive dimension i.e., higher the value, better the quality. Examples are throughput and availability. The execution time has negative dimension i.e. lower the value, better the quality.

Once the deployed web services are invoked upon client request in the composition workflow, continuous monitoring and analysis of web services is done by examining their execution log. Every web service that gets executed, when invoked by a client, maintains its log of QoS values on the provider side. In the proposed work, a service monitor agent monitors execution logs of the web services maintained at the service provider side. The web service execution log is maintained for monitoring and prediction based analysis. The QoS monitoring of the partner web services is done for the following four QoS parameters - execution time, throughput, reliability and availability.

In order to predict the values of a QoS parameter, say Q_k, EWMA [20] calculates the next expected value, which is exponentially weighted moving average of the past data. The EWMA is a statistical analysis which provides less weight to data as they get older and can forecasts the next observation one step ahead of the most recent observation.

The EWMA calculation at time t (for t=1, 2,…,n) is calculated recursively from the individual data readings for n observations, where the first EWMA value, $EWMA_0$, is arithmetic average of historical data.

$$\text{Exp}(\text{Q}_\text{k}(\text{t}+1)) = \alpha * \text{Obs}(\text{Q}_\text{k}(\text{t})) + (1-\alpha)\text{Exp}(\text{Q}_\text{k}(\text{t})) + \text{k} * \text{SD} \quad (1)$$

Where $Exp(Q_k(t+1))$ is the expected value of QoS parameter $Q_k$at time t+1 of execution;

$Obs(Q_k(t))$is the last observed value QoS parameter $Q_k$at time t of execution;

$Exp\ (Q_k(t))$is the expected value of QoS parameter $Q_k$at time t of execution;

The weighting factor,α, is a smoothening factor that decides the rate at which past expected values are considered in the calculation $(0 < \alpha < 1)$;

$k * SD$ is the set threshold value where SD is the standard deviation calculated from the past observed values and k is a constant parameter. The weighting factor, α, is a smoothening factor that decides the rate at which past expected values are considered in calculation$(0 < \alpha < 1)$. It acts as a balancing factor between the older, and the most recent observations. Depending upon α, the upper and lower control limits are set. The upper control limit (as in case of the execution time), or the lower control limit (as in case of throughput) is added to the threshold value in equation 1.

Adaptation of a partner web service is based on the comparison of observed QoS values with expected QoS values. If SMA notices three consecutive degradations (beyond tolerance) in any of QoS values, it notifies the CMAs, and sends them the normalized values (refer Definition 3) of the QoS parameters. A CMA then uses these normalized values to calculate a quality score for the web service as per the client's quality requirements (refer Definition 4). The normalized value of a QoS parameter Q_i is represented in the following equations (2) & (3), where Q_i^t is the i^{th} attribute of a particular web service in a given transaction t and $Q_i^{max}$and Q_i^{min} are the maximum and minimum values respectively for all transactions recorded for the same web service in a particular session.

$$\text{Norm}(\text{Q}_\text{i}) = \begin{cases} \frac{\text{Q}_\text{i}^\text{t} - \text{Q}_\text{i}^\text{min}}{\text{Q}_\text{i}^\text{max} - \text{Q}_\text{i}^\text{min}} & \text{if } \text{Q}_\text{i}^\text{max} - \text{Q}_\text{i}^\text{min} \neq 0 \\ 1 & \text{if } \text{Q}_\text{i}^\text{max} - \text{Q}_\text{i}^\text{min} = 0 \end{cases}$$

for a positive QoS parameter (2)

© 2019, IJCSE All Rights Reserved

$$\text{Norm}(Q_i) = \begin{cases} \frac{Q_i^{max} - Q_i^t}{Q_i^{max} - Q_i^{min}} & \text{if } Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & \text{if } Q_i^{max} - Q_i^{min} = 0 \end{cases}$$

for a negative QoS parameter (3)

The proposed approach works by proactively taking adaptive actions to handle the QoS degradation of web services that are chosen for execution at runtime. It uses an agent based approach. A mobile agent is a light weight program that can migrate to a remote machine on the network. A mobile agent is dispatched to the service provider side. QoS requirements of different clients are not the same. A client may prefer execution time over reliability. The CMA calculates the aggregate QoS values for the affected service using the normalized values it received from the SMA, and the client's QoS preferences (on the basis of the weights assigned to different QoS parameters). It compares the QoS aggregate value of the affected web service with the threshold value as specified by the client. If the aggregated value is within the permissible limits, then the CMA itself may defer the plan to notify the service and continue with the web service without notifying the client about the degradation. Otherwise, the CMA will notify the client. Then the client may run the adaptation logic to replace the web service.

The aggregated QoS of a particular web service is calculated (by the CMA) using the Simple Additive Weighting Technique [14]. It uses the normalized values of the QoS parameters (refer equations 2 & 3) along with their weights as per client preferences.

$$\text{AggScore} = \sum(w_i \times \text{Norm}(Q_i)) \quad (4)$$

w_i is the weighting factor of Q_i. The weights are assigned in such a way that $\sum w_i = 1$.

The aggregated value, *AggScore*, is then used as the basis whether the substitution is needed in that particular case or not. If the calculated *AggScore*, does not match with the given quality threshold that is specified by the client, the client is notified. Then that particular web service is removed from the execution plan and is replaced with an alternative (by the Service Repair Module on the client side). The threshold limit can be specified by executing the framework for multiple requests in a best case scenario.

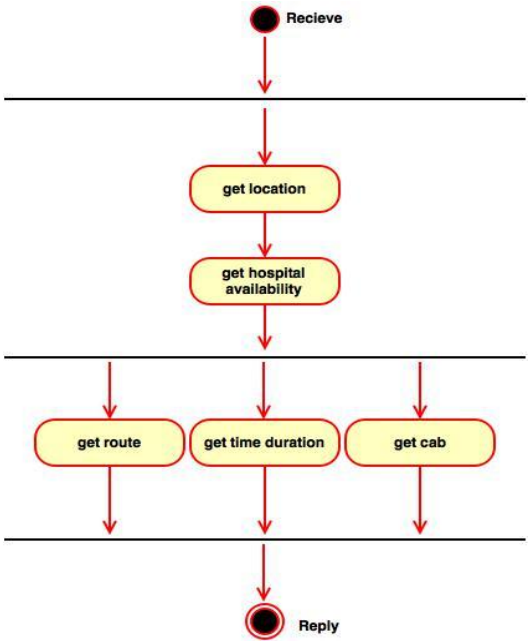

Figure 1. The Case Study

IV. THE CASE STUDY

For implementation and analysis of the framework, a hospital look up service in emergency situations is used as a case study. A primary motivation of creating this service is to reduce patient waiting time in a crowded emergency ward of a hospital. In the worst case, when a patient arrives at a crowded emergency ward, he or she may be diverted to another hospital. This delay in the treatment may put the patient's life at risk. The service can be used in handling patients in wake of emergencies during a natural disaster such as an earthquake.

Figure 1 demonstrates the whole process. A patient or the attendants can use the service to find out the nearby hospitals to handle the patient in emergency. The hospital look-up service interacts with five partner services – 1) location locater (to find the current location of the user with the help of Global Positioning System; 2) a hospital registry (to find a nearby hospital with special facilities to manage the emergency, and availability of a bed; 3) a map service to provide the available routes; 4)a service to provide the best route (depending upon the time of day, weather, and traffic); 5) a transport management service to let the patient request an ambulance or a normal cab. The best route is passed onto the cab driver.

V. EXPERIMENTAL SETUP

The prototype tool of framework is implemented in Java EE. The Abstract Composition, Workflow Manager, Service Manager and Service Repair are implemented as single

© 2019, IJCSE All Rights Reserved

components for simplicity. The Workflow Manager and Service Manager are exposed as RESTFUL web services. Service Repair module is implemented as an agent. The SMA and CMA are implemented as mobile agents. The client request is taken from an Android mobile phone.

The configuration of experiment environment for client and provider is done on an i5 processor with 4 GB RAM using Tomcat server and JADE 7. The external service registry and web service database is implemented using SQL server. The web services used in the experiment are described and selected with quality parameters. In order to save time, an abstract service composition is available before execution of the workflow starts. We also assume that each service is *atomic*. That is, behaviour of a service is independent of other services. The network connection between services is error free, even though individual atomic services may be problematic.

VI. RESULTS AND ANALYSIS

In this section, we analyze the performance of the proposed system as changes happen in the service environment at run-time. The proposed framework is validated by analyzing aggregate QoS behaviour of the CWS. We assume that a degraded web service is replaced with a one-to-one mapping. QoS attributes of the individual partner web services impact the QoS of the CWS. The QoS parameters include execution time, and throughput of the composite web service. Figures 2 to 3 show the QoS values of the CWS for 100 requests. Execution time of the CWS remains stable with QoS degradation happening in the partner web services. Throughput improves over the period of time. Therefore, we can say that the framework contributes in maintaining good levels of the different QoS parameters of the CWS.

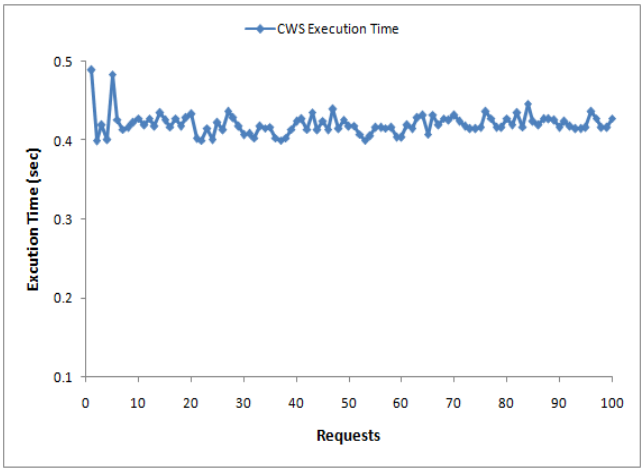

Figure 2. Execution time of the CWS

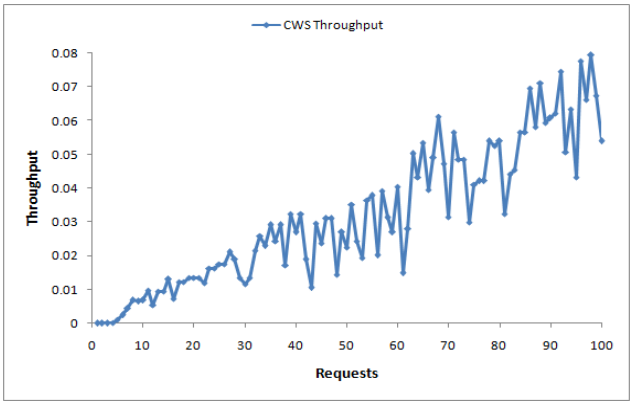

Figure 3. Throughput of the CWS

VI. CONCLUSIONS

The self-adaptation of composite web services based upon prediction and proactive monitoring at runtime has drawn a lot of attention in the web services based solutions. This paper proposes a distributed monitoring and adaptation framework to keep track of the QoS degradation of partner web services for hospital lookup service. This paper follows a preventive approach in invocation of partner web service with degraded QoS. Additionally, the applicability of the self-adaptive system features on the various quality dimensions, like reliability, availability and throughput. The adaptation decision is flexible enough as it takes into consideration different QoS requirements of different clients.

A prototype of the framework is implemented using J2EE and JADE environment. The experimental results show that the proposed approach results is acceptable performance in light of the QoS degradation simulated at different intervals and of different scales. There is, however, a limit on the number of requests that can be handled when concurrent requests are submitted to the system. In the present setup, the system is unstable after 2000 requests. In future, we plan to extend the framework for global optimization of composite web service.

REFERENCES

[1] V. Agarwal, & P. Jalote, "Enabling end-to-end support for non-functional properties in web services", *2009 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)* , pp 1 – 8, 2009.

[2] A. Amin, A. Colman, & L. Grunske,"Statistical detection of qos violations based on cusum control charts". *In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ACM.*, pp. 97-108, 2012.

[3] R. Angarita, Y. Cardinale, & M. Rukoz, "Reliable Composite Web Services Execution: Towards a Dynamic Recovery Decision", *Electronic Notes in Theoretical Computer Science Volume 302, Proceedings of the XXXIX Latin American Computing Conference (CLEI 2013)* , pp 5-28, 2014.

© 2019, IJCSE All Rights Reserved

[4] R. Angarita, "Responsible objects: Towards self-healing internet of things applications",*2015 IEEE International Conference on Autonomic Computing (ICAC). IEEE.*, pp. 307-312, 2015, July.

[5] R. Angarita, M. Rukoz, & Y. Cardinale, "Modeling dynamic recovery strategy for composite web services execution", *World Wide Web, 19(1)*, pp 89-109, 2016.

[6] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, & P. Plebani, "PAWS: A Framework for Executing Adaptive Web-Service Processes", *IEEE software, 24(6)* , 39, 2007.

[7] S. Asadollah, & T. Chiew, "Web Service Response Time Monitoring: Architecture and Validation", *Theoretical and Mathematical Foundations of Computer Science Volume 164 of the series Communications in Computer and Information Science* , pp 276-282, 2011.

[8] R. Aschoff, & A. Zisman, "QoS-Driven proactive adaptation of service composition", *Service-Oriented Computing, Lecture Notes in Computer Science, Volume 7084 2011* , pp 421-435, 2011.

[9] L. Baresi, & S. Guinea, "Towards Dynamic Monitoring of WS-BPEL Processes". *Service-Oriented Computing - ICSOC 2005 Volume 3826 of the series Lecture Notes in Computer Science* , pp 269-282, 2005.

[10] Y. Dai, L. Yang, & B. Zhang, "QoS-driven self-healing web service composition based on performance prediction", *Journal of Computer Science and Technology, 24*(2), pp 250-261, 2009.

[11] D. H. Elsayed, E. S. Nasr, M. Alaa El Din, & M. H. Gheith, "Appraisal and Analysis of Various Self-Adaptive Web Service Composition Approaches", *In Requirements Engineering for Service and Cloud Computing, Springer International Pub* , pp. 229-246, 2017.

[12] Q. He, J. Han, Y. Yang, H. Jin, J.G. Schneider, & S. Versteeg, "Formulating cost-effective monitoring strategies for service-based systems", IEEE Transactions on Software Engineering, 40(5). *IEEE Transactions on Software Engineering, 40(5),* pp 461-482, 2014.

[13] J. S. Hunter, "The Exponentially Weighted Moving Average", *Journal of Quality Technology, Vol. 18, No. 4* , pp 203-210, 1986.

[14] C. L. Hwang, & K. Yoon, "*Multiple attribute decision making Methods and applications",* CRC press, 1981.

[15] N.K. Kahlon, K.K. Chahal, & S.B. Narang, "Managing QoS degradation of partner web services: A proactive and preventive approach", *Journal of Service Science Research, 8(2)* , pp 131-159, 2016.

[16] Q. Liang, B. Lee, & P.Hung, "A rule-based approach for availability of service by automated service substitution". *Softw., Pract. Exper. 44(1)* , pp 47-76, 2014.

[17] H. Mansour, & T. Dillon, "Dependability and Rollback Recovery for Composite Web Services", *IEEE Transactions on Services Computing, Volume: 4, Issue: 4* , pp 328-339, Oct.-Dec. 2011.

[18] A. Metzger, C. H. Chi, Y. Engel, & A. Marconi, "Research challenges on online service quality prediction for proactive adaptation", *2012 Workshop on EuropeanSoftware Services and Systems Research - Results and Challenges (S-Cube),* pp 51 – 57, 2012.

[19] A. Michlmayr, F. Rosenberg, P. Leitner, & S. Dustdar, "Comprehensive QoS monitoring of Web services and event-based SLA violation detection", *Proceeding MWSOC '09 Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing, ACM New York, NY, USA* , pp 1-6, 2009.

[20] M. Natrella, *NIST/SEMATECH e-Handbook of Statistical Methods*, 2010.

[21] M. Oriol, X. Franch, & J. Marco, "Monitoring the service-based system lifecycle with SALMon", *Expert Systems with Applications, 42(19)* , pp 6507-6521, 2015.

[22] P. Plebani, & B. Pernici, "URBE: Web service retrieval based on similarity evaluation", *IEEE Transactions on Knowledge and Data Engineering, 21*(11), pp 1629-1642, 2009.

[23] K. Ren, J. Song, M. Zhu, & N. Xiao, "A bargaining-driven global QoS adjustment approach for optimizing service composition execution path", *The Journal of Supercomputing, Volume 63* (1), pp 126–149, 2013.

[24] F. Rosenberg, C. Platzer, & S. Dustdar, "Bootstrapping Performance and Dependability Attributes of Web Services", *In Proceedings of the IEEE International Conference on Web Services (ICWS'06),*pp. 205–212, 2006.

[25] F. Rosenberg, C. Platzer, & S. Dustdar, "QUATSCH–A QoS Evaluation and Monitoring Tool for Web Services" *Journal on Web services Research*, 2007.

[26] J. Ruiz, & C. Rubira, "Quality of Service Conflict During Web Service Monitoring: A Case Study", *Electronic Notes in Theoretical Computer Science, 321* , pp 113-127, 2016.

[27] Z. Zheng, & M. Lyu, "A runtime dependability evaluation framework for fault tolerant web services", In The International Workshop on Proactive Failure Avoidance, Recovery and Maintenance (PFARM'09), co-located with DSN2009, 2009.

[28] J. Zhu, P. He, Z. Zheng, & M. Lyu, "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization", *IEEE Transactions on Parallel and Distributed Systems*, IEEE, 2017.

Author Profile

Navinderjit Kaur Kahlon received Master Degree in Computer Applications and Ph.D. degree in Computer Science and Engineering from Guru Nanak Dev University, Amritsar, India. The research interests include Service Oriented Computing, dynamic monitoring of web services, agent based systems and machine learning approaches.

© 2019, IJCSE All Rights Reserved