IJCTA, 9(27), 2016, pp. 533-542 ISSN: 0974-5572
© International Science Press

An Algorithmic Approach for QoS-based Reusable Service Composition Optimal Selection

A.Florence Deepa[1] and J.G.R. Sathiaseelan[2]

ABSTRACT

In recent years, most of the services are available on the internet and can just utilized anytime and anywhere. To select an appropriate web service from different service providers offering the services, will play a vital role according to web client's need. A developer of web service who may create totally new service or extend an existing developed service. To publish the common features as services and to reuse the already published services is a perfect phenomenon in SOA. Reusability is one of the key objective when developing complex applications based on Web Services. Consequently, service reusability has a high demand and will help to reduce cost and time of a new development. Availability and reliability are the significant QoS parameters that are used to estimate service reusability of a web service efficiently.In this paper, the major contribution is to propose an approach to predict the reusability of a web service by quantifying *availability*and *reliability* of atomic services using the Weibull analysis and Linear Regression. The result of the proposed work is two-fold. First, by calculating the service availability from total uptime and total downtime of the service and the service reliability from mean time to failure of the web service. Second, using the above result, service reusability is estimated which will decrease the complexity during the new development.

Keywords: Availability, Reliability, Reusability, Weibull Analysis, Linear Regression, Kermack McKendrick model

1. INTRODUCTION

Web Service is the basic building block of Service Oriented Architecture (SOA). In web service architecture [1], the service provider offers the web services that provide the tasks or business processes which are defined over the internet that are invoked by the web clients. A web service requester articulates the requirements to find a suitable web service [2]. Publishing, binding, and finding web services are the three major tasks in web service architecture. The service providers offer the web services that satisfy the user's needs, which are available in the internet. The web service requestor is the web client who request for a web service [3]. The web service registry is a centralized directory of services where the web service providers publish their web services information. The specified information is kept in the registry and examined on submission of request by requester. Universal Description, Discovery and Integration (UDDI) is the registry standard for web services [4].

A. QoS Requirements for Web Services

Web Services are self-independent application that exposes modular and distributed concepts [5]. Web services can be developed at any irrespective platform and used by any application. Web service description is provided in WSDL document, it can be accessed from internet using SOAP protocol. The ultimate goal of web services is to interpret and normalize the application interoperability within and across the development of the operational skills and to cherish the partner relationships.

According to W3C [6], Quality of Service (QoS) denotes the quality aspects of a web service such as reliability, execution time, availability, response time, cost, performance, scalability, reusability and so on. Constraints

1 Ph.D. Research Scholar, Bishop Heber College (Autonomous), Tiruchirappalli, Tamil Nadu, India.

2 Associate Professor and Head, Bishop Heber College (Autonomous), Tiruchirappalli, Tamil Nadu, India.

are defined on the QoS, and these constraints can be utilized to select an optimal service for a requester. The faultless combination of web services, business processes and applications over the internet calls an active e-business visualization. Because of its vibrant and changeable nature, it is an essential and also a major challenge to carry out the QoS on the internet. The dynamic e-business idea is to require a perfect arrangement of web services, business procedures, and functions on the web [7]. Implementing quality of service on the web is a vital and main test due to its exciting and variable character.

B. Stipulation for Web Service Selection

In fact, finding out the appropriate web services that the service registries don't provide adequate query elements for clients to clear relevant service queries, is one of the vital challenge that can encounter by the user's requirements [8]. Still, to find the appropriate web services could not be achieved using simple keyword-based search method mainly as web services multiply. For example, service registries let the clients to carry out simple search queries such as searching by service or business name [9]. Since little documented information is often made available in service discovery interfaces, it is not practical to distinguish web services from each other using keyword corresponding techniques. Clients spend hours of searching through possible service resources by themselves in discovering appropriate web services [10]. Consequently, a service broker is able to collect web service information from diverse environments (together with service portals, service registries, and search engines) and offers a central access point for client to clear their search queries in an inventive method.

C. Prerequisite for Web Services Composition

Web Services Composition is the interconnection of certain web services that satisfies the common business process. The interconnection of web services to meet a certain business process is called Web Service Composition [11]. The composition can be viewed as an aggregation of elementary or composite web services. The business process of the client is implemented by several services when these web services are composed as a single service [12]. A client can invoke a composite service itself be exposed as a web service. Composition rules specify the order in which services are invoked and the conditions that some service may or may not be invoked. A main feature of services is the reuse mechanism to build new applications and to compose the composition rules that describe the coherent global services [13].

The remainder of the paper is organized as follows. Section II surveys some related work including QoS Aggregation ontology, Composite patterns, and Aggregated Reliability of composite web service. Section III contains QoS parameter specification such as web service standards, web service composition techniques, and other QoS constraint web service composition research and briefly introduce the concept and notations used in the paper. Section IV describes the model of proposed SISS QoS model, including basic definitions, constraints and their aggregation that provides the basic notions for composite service and QoS based formulation according to the existing literature, which are related to the proposed work. Section V depicts the design of ReUsable QoS-based Service Selection (RUQSS)algorithm for the Composite Web Service and its complexity analysis, and QoS-based Web Service Composition Framework. Section VI presents the evaluation setup, experiment results and analysis. In addition, a comparison and discussion are made with other similar approaches. And finally, the conclusions and future work are given in Section VII.

2. RELATED WORK:

The Composite Web Services technology plays a vital role in services computing that allows business processes by composing elementary services under a shared workflow thereby providing the business process as a single-service. The technical challenges for integrating the services in terms of compatibility and adaption are carried out in the earlier research papers.

A. QoS Aggregation Ontology

According to [1] [2], the composite QoS depends on two determinants: QoS parameters and workflow [11]. The first determinant, QoS parameters are different with regard to number, name, data type, and conceptualization and rarely adhere to any standard. A classification was proposed exclusively with regard to their aggregation, instead of contributing to their harmonization, and it was constructed based on the following principle: If any two parameters allocate the same aggregation function, then they are a member of the same parameter type, in spite of other characteristics. They apply that principle results in five parameter types.

B. Composite Patterns:

The second determinant workflow was analyzed by means of workflow patterns. [12] analyzed workflows effect on parameter aggregation and proposed seven *composition patterns* (CP), i.e., Sequence (CP_1), Loop (CP_2), XOR-XOR (CP_3), AND-AND (CP_4), AND-DISC (CP_5), OR-OR (CP_6), and OR-DISC (CP_7).

A composition model was resulted based on patterns derived from the workflow patterns identified in languages for Web Service composition. The following composite patterns, Sequence (CP_1), Loop (CP_2), XOR-XOR (CP_3), AND-AND (CP_4) and OR-OR (CP_6) patterns that are capable to construct the aggregation function concept for the QoS parameter typesto be proposed are applied.

3. QOS PARAMETER SPECIFICATION

In general, the web service interactions can be categorized as executable processes which models the actual behavior of a participant in a business interaction and as abstract processes. In the Abstract Process (AP) [13], the basic patterns of operations such as sequence, parallel, conditional and loop structure are used to construct the fundamental control flow structures of business workflows and an composite process description can be represented by a directed acyclic graph (DAG).

A. Web service QoS requirements

The important QoS attributes of service to select the optimal one according to the web client's need that are identified in this paper are: Availability and Reliability are used to examine the QoS-based Service Composition Optimal Selection (SCOS) that satisfy the user requirements to build the complex applications.

- **Availability**: Availability is the quality aspect of whether the Web service is present or ready for immediate use [14]. It indicates the readiness of the service that provides the exact one. It also represents the probability that a service is available. Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time [15].
- **Reliability**: Reliability is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality [16]. Reliability denotes the probability that the service will fail after a certain period of time. The number of failures per month or year represents a measure of reliability of a Web service. In another sense, reliability refers to the assured and ordered delivery for messages being sent and received by service requestors and service providers [17].

B. Service Reusability

The most challenging elementary principle of Service Oriented Architecture (SOA) is Reusability which represents the capability of using the service again. Service reusability is the measure of easiness in which one can use already existing developed services into an innovative applications. Erl [18] defined as, "Services contain and express agnostic logic and can be positioned as reusable enterprise resources". Designing a reusable service will increase Return On Investment (ROI) and also reduces the cost associated with design, development, software testing and maintenance. In [19], the author defined as, "Reusability of service is the degree to which the service can be used in more than one business process or service applications without having much overhead to discover configure and invoke it."

4. PROPOSED WORK

This section expresses the QoS model on the basis of theRUQSS algorithm. Here the aggregation function for the QoS parameters and the concept definitions for the parameter aggregation are described.

A. The ReUsable QoS-based Service Selection (RUQSS) QoS Model

In this section the QoS model articulates the ReUsable QoS-based Service Selection (RUQSS) algorithm for the Composite Web Service structure as *Nodes* – basic unit of a Composite Service and as *Operations* – relationship between nodes that determines the behavior of the Composite service (directed acyclic graph, DAG). The Operation relationship adopted from Jorge Cardoso's Stochastic Workflow Reduction (SWR) algorithm. The description of QoS aggregation ontology and the aggregation function that utilize this ontology, demonstrate the usefulness of the proposed algorithm for the developers of composite services and assess its computational efficiency.

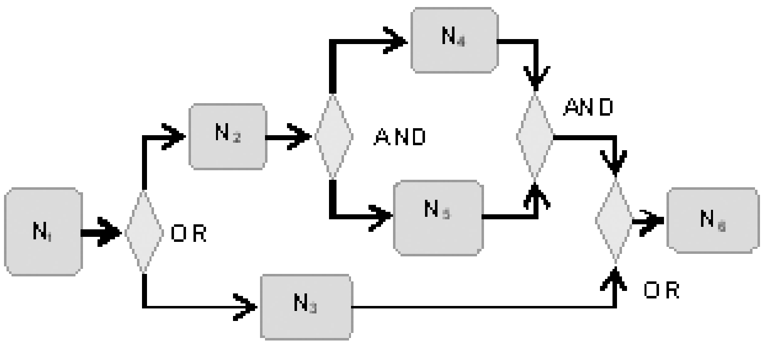

Figure 1. Example Workflow as a Business Process

Consider a composite service according to the figure 1 shows an example, which contains six nodes and four operation relations (represented as diamond shape). To instantiate the workflow as a business process for different services, the *nodes* are separated from actual services by means of a binding (*operationrelations*).

Since the formalization of the aggregation is a combination of parameter and composition patterns, the parameter types *(Cost, Execution Time, Reliability and Availability)*, which are defined in section I, and composition patterns *(Sequence, Loop, AND-AND, OR-OR and XOR-XOR)* that are proficient for the parameter types and the generic aggregation functions, with $x_1,\ldots,x_n$ denoting the parameter to be aggregated, which are related to the parameter types that are used in Table 1 to assign a function to pair of parameter type P and the composite patterns CP.

The QoS Aggregation relates the parameter types areidentified with generic aggregation formula and composition patterns by means of description logic (DL). Based on the ontology, reasoning determines the parameter aggregation for any given (annotated) QoS parameter and workflow pattern. Table 1 defines the aggregation function that can be implemented to each pair of parameter type P and the composite patterns CP.

The concept Aggregation Function *AF* and its disjoint sub-concepts represent all generic aggregation functions identified for the parameter types. Two functional roles Parameter Type *forP* and Composite Patterns *forCP* exist, concerning their dependence.

Table 1. Aggregation Functions for QoS parameters

Parameter Type	*Composite Patterns CP*				
P	*Sequence*		*Parallel*		
		Loop	*AND-AND*	*OR-OR*	*XOR-XOR*
Cost	χ_{sum}	χ_{linear}	χ_{sum}	χ_{max}	χ_{max}
Execution Time	χ_{sum}	χ_{linear}	χ_{sum}	χ_{max}	χ_{max}
Availability	$\chi_{product}$	χ_{min}	$\chi_{product}$	-	χ_{sum}
Reliability	$\chi_{product}$	χ_{min}	$\chi_{product}$	-	χ_{sum}

Table 1 represents the valid parameter types and composition patterns, for an example, say that a composite pattern *Loop* is aggregated with a sequence of parameter types *(Reliability and Availability)* by the aggregation function *Min*; consequently, the concept *Min is* identical to the concept definition as follows.

Table 2. Concept Definitions for Parameter Aggregations

Concept	*Inclusion*
Sum	$\exists$ *for CP.Sequence* $\cap$ (*for P.Cost* $\cup$ *for P.Execution Time*) $\cap$ *for CP.ANAND* $\cap$ (*for P.Cost* $\cup$ *for P.Execution Time*) $\cap$ *for CP.XORXOR* (*for P.Reliability* $\cup$ *for P.Availability*)
Product	*for CP.Sequence* $\cap$ (*for P.Reliability* $\cup$ *for P.Availability*) $\cap$ *for CP.ANAND* $\cap$ (*for P.Reliability* $\cup$ *for P.Availability*)
Max	*for CP.OROR* $\cap$ (*for P.Cost* $\cup$ *for P.Execution Time*) $\cap$ *for CP.XORXOR* $\cap$ (*for P.Cost* $\cup$ *for P.Execution Time*)
Min	*for CP.Loop* $\cap$ (*for P.Reliability* $\cup$ *for P.Availability*)
Linear	*for CP.Loop* $\cap$ (*for P.Cost* $\cup$ *for P.Execution Time*)

As concept *Min* is valid for more than one combination of parameter type and composite pattern, the concept definition consists actually of several pairs of universal restrictions being concatenated by a logical OR. Table 2 contains the full definitions of all aggregation formula concepts.

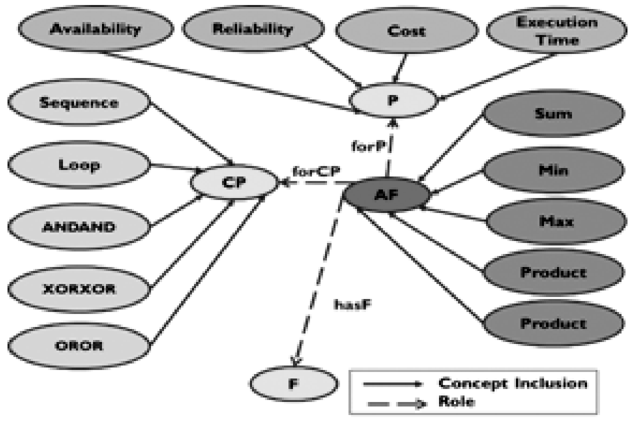

Figure 2. An Overview of QoS aggregation ontology

At last, to verify the concept not only the aggregation type, but also the aggregation formula to be used are identified. To define this principle, the ontology provides a concept *F* and a functional role defined as *(AF, F): hasF* as shown in the Figure 2, depicted in the algorithms *AggrQoS()* and *AggrValues()*.

5. REUSABLE SERVICE SELECTION ALGORITHM

This section articulates the design of the RUQSS algorithm. Here describe the QoS criteria, aggregation functions,and constraints are described.

A. Design of the RUQSS Algorithm

The service selection process of a SCOS proceeds the abstract processes as inputs, with the composite patterns that are supported by composite web services. That is, for an abstract process AP containing *n*abstract services AS_{AP}, where AS_{AP}= {as_1, ..., as_n}, there will be a set of composite patterns CP = {cp_1, ..., cp_m} associated with each as_i, where, each abstract service as_i(i = 1, ..., n) has $m_i$composite patterns. CP = {cp_{i1}, ..., cp_{im}} is a composite pattern of as_i. Each composite pattern cp_{ij} (j = 1, ..., m) ∈ $P_i$is associated with a QoS parameter (AV(cp_{ij}), RE(cp_{ij}), CT(cp_{ij}), ET(cp_{ij})).The QoS constraints of the AP defined by users are QC = {qc_{AV}, qc_{RE}, qc_{CT}, qc_{ET}}.

Based on the above analysis, a service selection process could be formulated as Eq. (1). Find X = (x_1, ..., x_n),]* x_i ∈ {1, ..., m_i}, i = (1, ..., n) such that,

$$\begin{aligned} &AV\ (\{as_1,\ af_{1x_1}),\ ...,\ (as_1,\ af_{1x_1})\},\ AP) \geq qc_{AV} \\ &RE\ (\{as_1,\ af_{1x_1}),\ ...,\ (as_1,\ af_{1x_1})\},\ AP) \geq qc_{RE} \\ &CT\ (\{as_1,\ af_{1x_1}),\ ...,\ (as_1,\ af_{1x_1})\},\ AP) \leq qc_{CT} \\ &ET\ (\{as_1,\ af_{1x_1}),\ ...,\ (as_1,\ af_{1x_1})\},\ AP) \leq qc_{CT} \end{aligned} \tag{1}$$

X is a vector representing the feasible solution that satisfies the reusability QoS constraints containing the indices of selected composite patterns for every abstract service in AP. It can be viewed as a selection plan of AP. According to Eq. (1), the service selection problem can be used to model for a SCOS solution.

The details of the RUQSS algorithm are shown in the algorithms *AggrQoS(n)*, *AggrValues(cp,N')*and *RUQSS(AS_{AP},n).*There are three major steps in the RUQSS algorithm. The first step is used to categorize the aggregated QoS constraints that are associated with the composite patterns of the workflow. The aggregation is executed on workflow *W.* Algorithm*AggrQoS(n)* depicts theexecution for *W*'s starts from the node *n*. If it begins with an AND_{split}/XOR_{split}/OR_{split}, the recursive process takes place for all branches. Then the aggregated nodes are collected in the set *N'*.The actual aggregation is performed based on the respective pattern using the algorithm *AggrValues(cp,N')* and the result is stored in the join nodes.

The next process is to detect subsequent *Sequence* and *Loop* patterns. From the *DAG* results stored in the join nodes are used to perform the task by the function *getSeqNodes(n)* works as follows: for task nodes, non-nested XOR_{split}, AND_{split}, OR_{split}, and subsequent task nodes are collected, whereas XOR_{split}/AND_{split}/OR_{split}, nodes are recursively aggregated. For the detection of nested *Sequence* patterns in *XORXOR*, *ANDAND*, as well as *XORXOR* patterns, the *DAG* is traversed until the corresponding join node is reached for each branch of XOR_{split}/AND_{split}/OR_{split} nodes. *Loop* nodes specify the multiple executions of the preceding node.

In the second step, the concrete aggregation is performed, which uses the aggregation ontology as depicted in the algorithm *AggrValues(cp,N').* To infer the right aggregation function, first it creates the individuals for parameter, composition pattern and aggregation function. By relating these three individuals by the roles *forP, forCP andforF,* the knowledge base is queried for all concept memberships of *hasF,* which returns the DL reasoning returns three membershipsand specific one is denoted as *f* usedto calculate the parameter value by considering all the nodes from the input set *N'*. The QoS parameter aggregation is calculated separately for each parameter.

In the final step of RUQSS, an empirical is proposed to find a feasible solution that satisfies the QoS constraints in the Eq. (1) as depicted in the algorithm *RUQSS(AS_{AP},n).* The original motivation of RUQSS is to find out a feasible solution which categories the existing web services that are could be reuse according with the user-defined requirements.

To depict the process of the RUQSS algorithm, first calculate the quality values for both the QoS parameter types and composite patterns using the Eq. (2) and Eq. (3) respectively. The quality constraint is then verified using the Eq. (4), and stored in X_{old} initially.

$$qv\ (f \text{ or } P_i) = \tag{1}$$

$$qv\ (f \text{ or } CP_i) = \frac{qv(f \text{ or } P_i)}{hasF_i} \tag{1}$$

$$qc\ (f \text{ or } P_i) = \begin{cases} qv(f \text{ or } CP_i) \geq thersold & +ve \quad criteria \\ qv(f \text{ or } CP_i) \leq thersold & -ve \quad criteria \end{cases} \tag{1}$$

Algorithm RUQSS(AS_{AP},n)

```
1 **for** i ← 1 **to** n **do**
2 Calculate qv(forP$_i$), qv(forCP$_i$) using Eq. (2) and Eq. (3)
3 $X_{old}$ ← qc(forP$_i$) using Eq. (4)
4 **next** i
5 iteration ← 1, stop ← true
6 **repeat**
7 **for** i ← 1 **to** n **do**
8 **for** j ← 1 **to** m$_i$ **do**
9 $X_{cur}$ ← AggrValues ($X_{cur_{ij}}$)
10 **if** ($X_{old} \neq X_{cur}$) **then**
11 $X_{new}$ ← AggrValues($X_{new_{ij}}$)
12 **if** is Feasible(X$_{new}$) **then**
13 $X_{old}$ ← $X_{new}$
14 **else**
1 $X_{old}$ ← $X_{cur}$
16 stop ← false
17 **return** the feasible solution $X_{old}$
18 **next** j
19 **next** i
20 iteration ← iteration + 1
21 **until** iteration > iteration$_{max}$ or stop
22 **return** the feasible solution X$_{old}$
```

$$\frac{\text{or } P_i * (f \text{ or } P_i + 1)}{f \text{ or } P_i + e^{(f \text{ or } P_i + 1)}}$$

The final step is implemented as an iterative process until the maximum *iteration* is reached or the *stop* is set as false. For this, initialize the *iteration* as one and set *stop* as true. Now the results of the algorithm *AggrValues(*$X_{cur_{ij}}$*)* are stored in X_{cur}. If the X_{cur} is not defined in the existing one then the algorithm *AggrValues(*$X_{new_{ij}}$*)* is executed with the new requirements that are preferred by the clients and the results are stored in X_{new}. The function *isFeasible*(X_{new}) is used to check whether the solution is a feasible one or not. The feasible solutions are loaded in X_{old} and returned. Once all the operations are examined for their feasibility, if there found an infeasible solution, *stop* is set as false and the *RUQSS* procedure is terminated.

B. Complexity analysis of RUQSS:

To simplify the complexity analysis, it is assumed that each abstract service has *c* composite patterns. Let *n*be the total number of abstract services in an abstract process and *m*be the number of QoS parameter type. For the first step of RUQSS, the complexity of the procedure *AggrQoS* is *O(c.n)*. Similarly in the second step, the complexity for the procedure *AggrValues* is *O(c.n.m)*. Finally, there are *n. (c - 1)* composite patterns to be explored in the iteration process to evaluate whether a feasible solution exists or not. If the maximum number of iterations is *t*, then the total complexity of the third step is *O(t.m.n.c)*. According to above analysis, the complexity of the RUQSS algorithm is *O(t.m.n.c)*.

6. EXPERIMENTAL RESULTS

The proposed algorithms are implemented on Windows 7 platform using Microsoft Visual Studio .NET development environment and Microsoft Visual C# as a programming language. Simple XML structures are used to create the service ontologies, and executed them on an Intel (R) Core™ i3 CPU, M 350 @ 2.27 GHz, 3 GB RAM Laptop with 100 MB/s Ethernet card. To generate the test instances, the abstract processes are randomly generated first, each contains two or more control flow patterns. These abstract processes are organized with the composite patterns that are associated with four QoS parameters namely execution time, cost, availability and reliability. The QoS constraint on each QoS criteria is generated using the Eq. (1).

A. Experimental Results

RQSS [13] is a well-known heuristic algorithms for QWSC, it is implemented and compared with the proposed work RUQSS here. From the proposed approach and algorithms mentioned, the following aspects related to the will be illustrated in the experiments, as the failure rate is reduced in finding a feasible solution while the abstract service increased.The strength of QoS constraints is increased while the time complexity is decreased and the number of iteration value will improve the better performance of the algorithm.

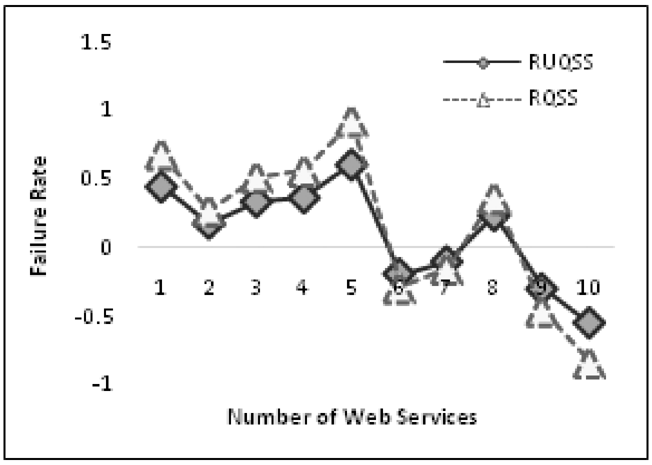

Figure 3. Comparison of Failure Rate

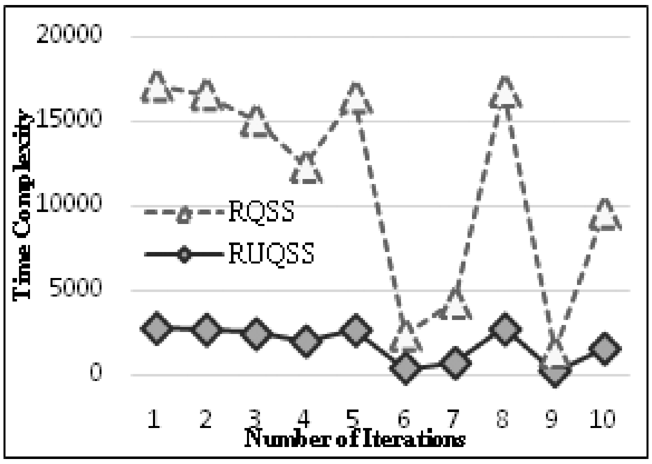

Figure 4. Comparison of Time Complexity

Figure 3 shows the experiment results of examining the failure rate between RUQSS and RQSS. The analysis of the failure rate is focused on the number of abstract services in an abstract process that are associated with the composite patterns and the strength of the QoS constraints. Figure 4 depicts the analysis of the time complexity between RQSS and

RUQSS. Here, it is observed that the time complexity is decreased while the strength of the QoS constraints are increased.

Table 3. Comparison of RQSS and RUQSS

	RQSS	*RUQSS*
Time Complexity	$O(t.m.n^2.c)$	$O(t.m.n.c)$
Availability	Middle	High
Reliability	Middle	High
Execution Time	High	Low
Cost	High	Low

Table 3 depicts the comparison of the existing approach with the proposed one that is observed from the QoS constraint on each QoS criteria by generating using the Eq. (1).Based on the comparison, the failure rate and the availability has been proved that the RUQSS is ultimate superior to the existing approach from the simulation result. In addition, it is observed that the RUQSS is also an optimal solution for Service Composition Optimal Selection (SCOS) with lower failure rate and high availability.

7. CONCLUSION

Since, it is a challenging process to implement a development environment for users to reduce the complexity application building based on the web service composition technology. Most researches or studies do not deal with the situation of no suitable or feasible solution can be found during the service composition process.

The ReUsable QoS-based Service Selection (RUQSS) algorithm, is mainly proposed to get higher system reliability and availability. The experiment results reveal that the failure rate of finding a feasible solution in RUQSS is much lower than RQSS approach. In future work, the RUQSS algorithm can be extended for the dynamic service composition.

REFERENCES

[1] P. Karaenke, J. Leukel and S. Vijayan, "Ontology-based QoS Aggregation for Composite Web Services", *11th International Conference on Wirschatinformatik*, Germany, 1343-1357, 2013.

[2] P. Karaenke and J. Leukel, "Towards Ontology-based QoS Aggregation for Composite Web Services", *Informatik,* GI, Bonn, **175**, 120-125, 2010.

[3] N. Saikaladevi and L. Arockiam, "Reliability Evaluation Model for Composite Web services", *International Journal of Web & semantic Technology*, **1(2)**, 16-21, 2010.

[4] M. Alrifai, D. Skoutas and T. Risse, "Selecting Skyline Services for QoS based Web Service Composition", *World Wide Web Conference*, Raleigh, North Carolina, 2010.

[5] Jiang Ma and Hao-peng Chen, "A Reliability Evaluation Framework on Composite Web Service", *International Symposium on Service-Oriented System Engineering*, 123-128, 2008.

[6] Hangjung Zo, D.L. Nazareth and H.K. Jain, "Measuring Reliability of Applications Composed of Web Services", *40th Annual Hawaii International Conference System Sciences*, 2007.

[7] F. Baader, I. Horrocks and U. Sattler, "Description Logic", *Handbook of Knowledge Representation, Elsevier*, Amsterdam, 2007.

[8] Zhenyu Liu, Ning Gu and Genxing Yang, "A Reliability Evaluation Framework on Service Oriented Architecture", *2nd International Conference on Pervasive Computing and Applications*, 466- 471, 2007.

[9] M.C. Jaeger, G. Rojec-Goldmann and G. Mühl, "QoS Aggregation for Web Service Composition using Workflow Patterns", *8th IEEE Enterprise Distributed Object Computing Conference , IEEE Press*, New York, 149–159, 2004.

[10] J. Cardoso, A.P. Sheth, J.A. Miller, J. Arnold, K. Kochut, "Quality of service for work-flows and web service processes", *Journal of Web Semantics*, **1**, 281–308, 2004.

[11] S. Dustdar, "Web Services Workflows - Composition, Co-Ordination, and Transactions in Service-Oriented Computing", *Concurrent Engineering,* **12(3)**, 2004, 237-245, 2004.

[12] J. Cardoso, A.P. Sheth, J.A. Miller and J. Arnold, “Modeling Quality of Service for Workflows and Web Service Processes”, *Technical Report* #02-002, LSDIS Lab, Computer Science, University of Georgia, 2002.

[13] Chia-Feng Lin, Ruey-Kai Sheu, Yue-Shan Chang and Shyan-Ming Yuan, “A relaxable service selection algorithm for QoS-based web service composition”, *Journal of Information and Software Technology*, **53**, 1370 - 1381, 2011.

[14] “Application guide to the specification of dependability requirements”. *IEC standard 60300-3-4*, Sep. 2007.

[15] P. Axer, M. Sebastian, and R. Ernst. “Reliability analysis for MPSoCs with mixed-critical, hard real-time constraints”, *9th International Conference on Hardware/Software Codesign and System Synthesis*, 149–158, 2011.

[16] H. Mcheick and Y. Qi, “Quality attributes and design decisions in service-oriented computing,” *Proc. Innovations in Information Technology (IIT) Conference*, Abu Dhabi, 283-287, 2012.

[17] K. C. Lee, J. H. Jeon, W. S. Lee, D. Min, M. N. An, S. Kim, S. H. Jeong, and S. W. Park, “QoS for web services: Requirements and possible approaches”, *W3C Working Group Note*, 2003.

[18] Thomas Erl, “SOA Principles of service Design”, *Pearson Education*, 2009.

[19] Si Won Choi and Soo Dong Kim, “A Quality Model for Evaluating Reusability of Services in SOA”, *10th IEEE Conference on E-Commerce Technology and the fifth IEEE conference on Enterprise Computing, E-commerce and E-services*.