ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Estimating the Reliability of Composite Web Services- Service Consumer Perspective

Dr.L. Arockiam
Associate Professor, Department of Computer Science
St.Joseph's College
Trichy, TN, India
larockiam@yahoo.com

Sasikaladevi*
Research Scholar, Department of Computer Science
St.Joseph's College
Trichy, TN, India
sasikalade@yahoo.com

Abstract: Web services are playing a predominant role in today's business environment. Business process is implemented as composed web services. Different workflow patterns are available for web service composition. In a business environment, web services' availability has been identified as one of the key properties for service-oriented applications. Quality of Service (QoS), including reliability, has been identified by IEEE as a service consumer perceived property. Though numerous web services are evolving today, reliability of the web service is an important issue. In this paper, reliability estimation technique for atomic web service is proposed. And Aggregate reliability is estimated for basic workflow patterns for composite web service.

Keywords: Web Service, Reliability, Workflow Patterns.

I. INTRODUCTION

A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized eXtensible Markup Language (XML) messaging. A web service is characterized by its flexibility to encapsulate discrete business functionalities and its interoperability to support universal application integration. Composition of computational resources and web-based services into integrated solutions is a key activity to enhance offerings and allow for a smooth process from the point of view of customers. Such integration process has been greatly simplified with the advent of web services technology. Qualities of Services (QoS) are defined in the service level objectives part of Service Level Agreement (SLA). QoS for web services are security, performance and availability.

Reliability is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality. 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. Service reliability can be defined as the continuity of correct service delivery. Reliability is a key requirement for building dependable systems. Reliability is the probability that the software will be functioning without failure under a given environmental condition during a specified period of time.

Menascé [1] characterizes QoS as a combination of several properties, including availability, security, response time, and throughput. Cardoso et al [2] develop ontology for the specification of QoS metrics, including task response time, task cost, and reliability as QoS dimensions of interest.

Sumra and Arulazi [3] Suggest that web services QoS requirement include performance, reliability, integrity, accessibility, availability, interoperability, and security. Ran [4] proposes a new web services discovery model using QoS metrics, classifying these metrics into four distinct categories, related to execution, transaction, configuration management, and cost.

Reliability, security, cost, and performance are criteria that are frequently identified as being relevant non-functional requirements when selecting web services. To determine the relative importance of these criteria a survey was conducted among several information technology architects in the Midwest [5]. The relative weights of these criteria were inferred through the analytic hierarchy process [6]

The results were surprisingly consistent, and indicated that security and reliability were among the most important criteria that architects consider when evaluating web services.

Software reliability is typically characterized as the probability that a piece of software will exhibit failure- free operation during a specified period. Xie [7] defines software reliability as "the probability that the software will be functioning without failures under a given environmental condition during a specified period of time". Other characterizations tend to measure reliability in terms of a percentage of failure cases in a given number of attempts to compensate for variations in usage over time [8][9]. The use of multiple concepts of failure and metrics for assessment of failure gives rise to a host of definitions. In the software context, failure is generally indicative of the exposure of underlying design faults by selection of specific inputs or operating conditions. Breaking down the source and nature of failures permits more precise modeling of software artifact reliability. However, most characterizations of reliability of web services prefer to leave the sources of failure unstated [7][10].Some researchers view reliability as a non-functional

characteristic of web services [11]. Still others characterize it as a component in setting service level agreements with web services vendors [10].

In this paper, web service reliability estimation technique for composed web service is proposed. And Aggregate reliability is estimated for basic workflow patterns.

II. RELATED WORK

Abdelkarim erradi and piyush maheshwari described a broker based approach [12] for improving web service reliability. As interest in web services has developed, so has the need to deliver reliable services with high quality of service (QoS) attributes covering functionality, performance and dependability. QoS requirements are more obvious in web services based integration because of the independence of the system involved the cross-administrative domains interactions and the web latency. The services may fall short for various reasons like network fault, overload and resource starvation. Due to this, there is a developing new architectural principle for integrate web services for high accessibility and assured and ordered message delivery even in case of system (or) network failure. To address these requirements is to introduce an extensible mediation layer called web services message bus [WSBUS] to interrupt exchanged messages and transparency enhance the reliability of services interaction.

P. W. Chan and Michal R Lyu have given solution for dynamic web service composition [13]. They provide a dynamic Web service composition algorithm with verification of Petri-Nets. Each Web service is described by Web Service Definition Language (WSDL) and their interactions with other services are described by Web Service Choreography Interface (WSCI). Their algorithm composes the Web services with the information provided by these two descriptions. After the composition, they verify the Web service to be deadlock free with modeling the Web service as a Petri-Net. They conduct a series of experiments to calculate the precision and performance of the composed Web service.

Jianbin Huang, Heli Sun [14] have described a novel approach for reliable Web service provisioning based on mobile agent and resource discovery technologies is proposed. In this approach, new service can be appropriately instantiated on demand, therefore reducing the risk of service being unavailable. Then a dispatching algorithm for resources selection and a multi-phase resource planning algorithm for composite Web services are presented. They propose techniques for reliable Web services provisioning. A service-transfer is associated with a Web service, which can instantiate a new copy of the service that is on demand too frequently. They introduce a dispatching algorithm for selecting computing resources that meet the requirements of Web services. they also propose a multi-phase resource planning approach where resources are selected for the components of the composite service based on a number of criteria. The concept of mobile Web services was used in our model as the basis for the reliable services provisioning.

Jiang Ma, Hao Peng chen [15] have given a reliability evaluation framework on composite web services. All service providers register their web services on the UDDI which holds functional and non-functional information of these web services in service descriptor. When each client invokes and consumes web service, it will automatically feedback QoS information to UDDI. The repository stores QoS information from data collector which is the feedback interface of UDDI. Based on those QoS information, a composite service could evaluate its reliability and when this composite service applies to other web service or backup services pool, the reliability will be reevaluated. In their model, the feedback information they collect is from client. The common way for QoS collecting is from service provider side which adds extra QoS metrics when data transport between consumers and providers. UDDI acquires this information from service provider and utilize it to recommend. However, the challenge of this model is the trustworthiness and accuracy of the information the provider feedbacks. They believe this pivot data for recommendation and evaluation should be took by a third party UDDI framework.

Duhang zhong and zhichang [16] have given a Petri Net based approaches for reliability prediction of web services. In this paper, they propose a Petri net based approach to predict the reliability of web service composition. First step of the approach involves the transformation of web service composition specification into Stochastic Petri Nets (SPN) model. The proposed transformation is built upon BPEL. From the SPN model, they can derive the reliability and performance measure of web service composition.

Zhang et.all [17] have explored the criteria of reliability of Web services-oriented systems, and discusses how to design and generate test cases to conduct tests over Web services. A prototype system is constructed to test the effectiveness and efficiency of our algorithms.

Hangjung Zo, Derek L. Nazareth, Hemant K. Jain [18] have provides a basis for measuring reliability of an application system that is assembled using web services. It outlines criteria for measuring reliability for individual tasks, specifies combining functions for differently configured tasks.

III. RELIABILITY ESTIMATION MODEL FOR WEB SERVIECS

During the web service invocation, service consumer finds any of the following three statuses on the invoked web service and these statuses are come to most of the web services. The statuses are Continuous success, continuous failure and transitory failure. In the web service world, encountering continuous successes is the most widespread status. When examining the services' invocation records, we find that some services may encounter unbalanced failures. This status is further divided into two categories: first, a service consumer may encounter transitory invocation failures but successfully access the service in subsequent invocations; second, transitory invocation successes and failures happen alternatively. This status has been discovered in several services. We devise this status with less than continuous five failed invocations as transitory failures. Rarely services may encounter continuous failures during our study. Because the invocation records are collected at the service consumer side,

this status presents the failures of either the service or network. These statuses show the runtime characteristics of service invocation successes and failures.

We assume that running web service may fall in any one of the following three statuses based on the invocation records. Invocation records stores the status information about the invoked web service.

Continuous Success (S): This status means that the service is running stably and no invocation failure occurs. Commonly services developed by reputable software enterprises are running in this status.

Transitory Failure (T): This status indicates that encountering failures during invocation is not predictable and it may be recovered by a simple "retry".

Continuous Failure (F): This status means that a service becomes down due to various factors. From the perspective of service consumer, they encounter continuous failures. This status is common for web services.

By using the above three statuses, we devise the new metric for evaluating the web service reliability. We evaluate how lone the particular service is in the above three states and based on this time value, we calculate the reliability rate for the web service

A. Service Reliability Estimation Metrics:

Service Invocation results are stored in the Service Invocation Registry. The Structure of the Service Invocation Record is given below,

$$R_{1n} = \{ V_{t1}, V_{t2}, V_{t3}, V_{t4} \ldots, V_{tn} \} \quad (1)$$

Where, R_{1n} is the Service Invocation Record for a specific service form time t1 to tn. V_{ti} is the Service Invocation Result at time ti. The Value is either 0 or 1. "0" refers failure and "1" refers Success of the service invocation. Set of service invocation records are stored in the service registry. The structure of the service invocation registry is shown below,

$$R = \{ R_{i1,i1}, R_{i2,i2}, R_{i3,i3} \ldots, R_{in,im} \} \quad (2)$$

Where, R is the registry of the set of services, $R_{i1,i1}$ is the Service Invocation record from time i1 to j1.

$$S = \{ S_1, S_2, S_3 \} \quad (3)$$

Where, S is the web service status result. S_1 is the continuous success status (S), S_2 is the transitory failure status (T), S_3 is the continuous failure status (F).

B. Service Reliability Estimation Method:

The input of the evaluation approach is a sequence of invocation records. The goal of the estimation approach is to calculate the service reliability values under different statuses. We have to identify service running status-based on invocation records and to calculate the reliability value under each different status. The estimation process involves,

a. Segmenting the invocation records
b. Estimating the values of availability metrics under each status.

a) Segmentation Algorithm:

//Given all the historical invocation records L1;n, we first divide them into m small fragments. The parameter m is set manually according to n. Each fragment contains an approximately equal number of records//

Input: $L_{1,n}$, k.

Output: L_v

a. Set number of segments as $\lfloor (n/k) \rfloor$
b. Examine $L_{1,n}$ orderly, put every $\lfloor (n/k) \rfloor$ records into a different sequence $L_{i,t}$. The remaining records are put into the last sequence. Then $L_{1,n}$ is divided into k fragments.
c. Repeat the following procedure until L_v remains unchanged. Examine all sequences in L_v orderly and comparing the records at the boundary of each sequence.
d. Examine all sequences in L_v orderly. Divide sequence $L_{i,t}$ into three sub-sequences. In this step, t is a manually defined threshold value to identify whether the service status is continuous down: when the length of continuous failures is more than t, then the service status can be regarded as continuous down.

b) Setting the Service Status Value:

Status of the individual services is identified by using the following rules.

a. If there is no service invocation failure in $L_{k,i}$, then this sequence is marked as status continuous success (S).
b. If there is no service invocation success in $L_{k,i}$, then this sequence is marked as status continuous failure (F).
c. Others are marked as transitory failure (T).

The Service reliability value is calculated based on the status as shown in the table 1.

Table 1 Service Status and its Reliability Rate

Service Status	Reliability Rate
Continuous Success (S)	1
Continuous Failure (F)	0
Transitory Failure (T)	Based on Weighted Average value

C. Service Reliability Estimation:

The service may have been in the transitory failure status for several times. For every time when the service runs in this status, there are several invocation records being recorded and the reliability value can be calculated as given below,

$$R_T = (N_a - N_f)/N_a \quad (4)$$

where, N_a is the number of invocations and N_f is the number of failures. Weighted average value is calculated based on the given formula.

$$R_T^w = \frac{\sum_{i=1..t \wedge S_L = T} * R_T}{\sum_{i=1..t \wedge S_L = T}} \quad (5)$$

In the fragmented sequences L_v, there are many sequences marked with status transitory failure, and each sequence has a different success rate. To produce an overall evaluation for the success rate in status transitory failure, we devise a weighted average technique to calculate the overall success rate, and the

weighting value is the time elapsed for each sequence. Finally we calculate the particular service is in each of the three states.

The calculation of the time percentage of each status is based on Equation (6). In this equation, the denominator is the total time duration of the invocation and the numerator is the time duration of one of the three statuses where si denotes a status, such as "S", "T" or "F".

$$T = \frac{\sum_{i=1..t \wedge S_{L_{i,i'}} = s_i} |t_{ki'}, t_{ki}|}{\sum_{i=1..t} |t_{ki'}, t_{ki}|} \quad (6)$$

IV. AGGREGATE RELIABILITY OF WORKFLOW PATTERNS

A workflow is a Combination of tasks and transitions. Tasks are represented by a circle, and transitions are represented using an arrow. Transitions state dependencies between tasks and are related with an enabling probability. If a task has only one outgoing transition, then the probability is 1. A task with more than one outgoing transition can be clustered as an AND-split or XOR-split. AND-split tasks facilitate all their outgoing transitions after completing their execution. XOR-split tasks permit only one outgoing transition after completing their execution. AND-split tasks are represented with a '*' and XOR-split tasks are represented with a '+'. A task with more than one incoming transition can be classified as an AND-join or XOR-join. AND-join tasks start their execution when all their incoming transitions are enabled. XOR-join tasks are executed as soon as one of the incoming transitions is enabled. As with AND-split and XOR-split tasks, AND-join tasks and XOR-join tasks are represented with the symbol '*' and '+', respectively. When no symbol is present to indicate the input or output logic of a task, then it is assumed to be an XOR.

The stochastic workflow reduction method consists of applying a set of reduction rules to a workflow until only one atomic task exists. Each time a reduction rule is applied, the workflow structure changes. After several iterations only one task will remain. When this state is reached, the remaining task contains the reliability metrics corresponding to the workflow under analysis. The set of reduction rules that can be applied to a given workflow corresponds to the set of inverse operations that can be used to construct a workflow. We have decided to only allow the construction of workflows which are based on a set of predefined construction systems; this protects users from designing invalid workflows. To compute reliability metrics, we used five reduction rules: (1) sequential, (2) parallel, (3) conditional, (4) simple loop and (5) dual loop.

A. Sequential Workflow Pattern:

Several tasks of a process are executed after each other. A task is enabled after the preceding task has finished and before the proceeding task has been started [19]. Fig. 1 illustrates how two sequential workflow tasks *ti* and *tj* can be reduced to a single task *tij*. In system, the incoming transitions of *ti* and outgoing transition of tasks *tj* are transferred to task *tij*.

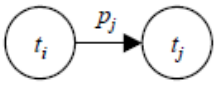

Figure 1 Sequential Workflow Pattern

In a sequential system, $pj = 1$. This reduction can only be applied if the following two conditions are satisfied: a) *ti* is not a XOR/AND split and b) *tj* is not a XOR/AND join. These conditions prevent this reduction from being applied to parallel, conditional, and loop systems. To compute the reliability of the reduction, the following formula is applied:

$$R(t_{ij}) = R(t_i) * R(t_j) \quad (7)$$

B. Parallel Work flow Pattern:

A Parallel Split is a divergent point in a business process where a single branch is alienated into two or more parallel branches which are executed in parallel. Synchronization is a distinct point in a business process where two or more different branches are combined into one single branch. It is called Synchronization because it expects all merged branches to be finished before going ahead with the process [19].

Fig. 2 illustrates how a system of parallel tasks *t1, t2, ..., tn*, an and split task *ta*, and an and join task *tb* can be condensed to a sequence of three tasks *ta*, *t1n*, and *tb*. In this reduction, the incoming transitions of *ta* and the outgoing transition of tasks *tb* remain the same. The only outgoing transitions from task *ta* and the only incoming transitions from task *tb* are the ones shown in the figure below. The $p1n = pb = 1$. To compute the reliability of the above reduction, the following formula is applied:

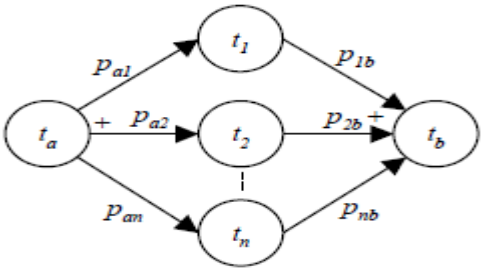

Figure 2 Parallel Workflow Pattern

$$R(t_{1n}) = \Pi R(t_i) \quad (8)$$

C. Conditional Work flow Pattern:

A Conditional is a distinct point in a business process where a branch is divided into two or more branches enabling just one of these several links [19].

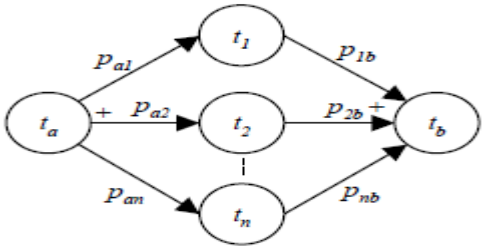

Figure 3 Conditional Workflow Pattern

Fig. 3 illustrates how a system of conditional tasks *t1, t2, …, tn*, a XOR split, and a XOR join can be reduced to a sequence of three tasks *ta, t1n*, and *tb*. Task *ta* and task *tb* do not have any other outgoing transitions and incoming transitions, respectively, other than the ones shown in the figure. In this reduction the incoming transitions of *ta* and outgoing transition of tasks *tb* remain the same. The availability of tasks *ta* and *tb* remain unchanged, and $p1n = pb = 1$. To compute the availability of the above reduction, the following formula is applied:

$$R(t_{1n}) = \sum_{1<i<n} p_{ai} * R(t_i) \quad (9)$$

D. Simple Loop Worklflow Pattern:

The Structured Loop pattern describes the possibility of executing an activity or sub-process repeatedly. This loop has either a pre-test or a post-test condition which means that the condition is either tested at the beginning or the end of a loop. The loop itself has a single entry point and a single exit point. The pattern itself consists of three possibilities: the normal one, the pre-test one and the posttest one [19].Loop systems can be characterized by simple and dual loop systems. Fig. 4 illustrates how a simple loop system can be reduced. A simple loop system in task ti can be reduced to a task *tli*. Probability is $p_i + \sum_{i=1}^{n} p_{oi} = 1$.

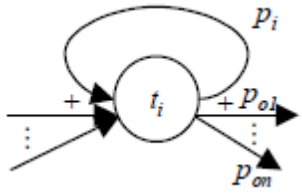

Figure 4 Simple Loop Workflow Pattern

To compute the reliability of the reduction, the following formula is applied:

$$R(t_{1i}) = \frac{(1-p_i)R(t_i)}{1-p_iR(t_i)} \quad (10)$$

E. Dual Loop Workflow Pattern:

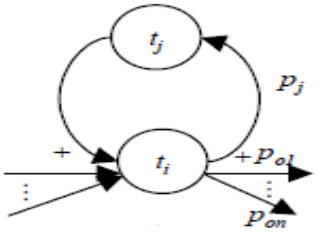

Figure 5 Dual Loop Workflow Pattern

Fig. 5 illustrates how a dual loop system can be abridged. A dual loop system consists of two tasks *ti* and *tj* can be condensed to a single task *tij*. Probability is $p_i + \sum_{i=1}^{n} p_j = 1$. To compute the reliability of the system, the above formula is applied:

$$R(t_{1i}) = \frac{(1-p_i)R(t_i)}{1-p_iR(t_i)R(t_j)} \quad (11)$$

V. EXPERIMENTAL RESULT

Web services which are relevant to students information processing are collected from web. Among these web services, the 1000 most relevant web services are identified. Reliability value is calculated for each of these web services. Invocation history for these web services is collected and invocation records are constructed. Totally 10000 invocation records are created for each web service and it is divided into 100 fragments of size 100. Reliability status is calculated on each fragment and time percentage is calculated using the equation (6). Part of the service invocation registry is shown below (status value in first 10 time period is included).

A. Aggregate Availability of Composed Web Service-Case Study:

The template is designed so that author affiliations are not The Composed web service consists of set of services connected using workflow patterns. Reliability of composed web service is based node/service reliability and link reliability. In this paper, we consider only the service reliability. To compute the reliability of the composed web service, we identify the workflow patterns which are involved in the composed web service. Compute the reliability value based on the derived workflow pattern, finally, aggregate the value.

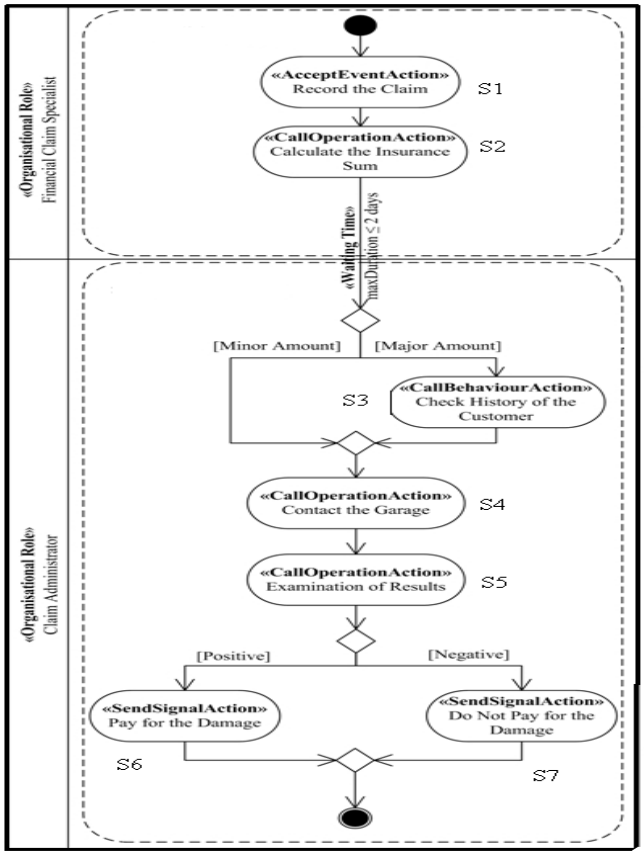

Figure 6 UML Profile of Farm Equipment Claim

$$R_{CWS} = R_1 R_2 (p_a R_3) R_4 R_5 (p_c R_6 + p_d R_7)$$

$$p_a + p_b = 1, \qquad p_c + p_d = 1,$$

$$p_a = 0.6, \qquad p_c = 0.6,$$

$$Finally, R_{CWS} = 0.442$$

Table 2 Reliability Status of Sample Web Service

	Services									
Time Period	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
1	S	S	S	S	S	S	S	S	S	S
2	S	T	S	F	S	S	T	S	T	S
3	S	S	S	S	F	S	S	F	S	T
4	S	S	F	S	S	S	S	S	S	S
5	S	S	S	S	S	S	S	S	S	S
6	T	S	S	S	S	F	S	S	S	S
7	S	S	S	S	S	S	F	S	S	F
8	S	F	S	T	S	S	S	S	S	S
9	S	S	S	S	S	S	S	S	F	S
10	F	S	T	S	T	S	S	T	S	S

Table 3 Reliability Values of Sample Web Service 1

Service	Reliability Value
S_1	0.956
S_2	0.937
S_3	0.946
S_4	0.957
S_5	0.956
S_6	0.946
S_7	0.958

The example business process of an UML 2 AD is presented in figure 6. This is the Farm equipment insurance claim process. The business process begins with an initial node, to activate the first action, Record the Claim. Record the claim passes the token to the next action to Calculate the Insurance Sum. These two actions are part of the activity partition Financial Claim Specialist. After estimating the insurance sum the path is split up by a verdict node into two alternative flows, depending if the insurance sum has a minor amount or a major amount. If the insurance sum has a minor amount, then the action contacting the Person starts. If the insurance sum has a major amount, then the flow is split up into parallel pathes by a fork node. That means that the actions Contacting the Person and Checking History of the Customer are executed in parallel. A merge node combines the different flows, and accepts the token as well as of one path or of both paths. The action Examination of Results decides that the claim is handled either positive or negative. Therefore a decision node splits up the path in two alternative flows, with the actions Pay for the Damage or Do Not Pay for the Damage. After that conclusion the business process ends with a flow final node.

In this basic workflow patterns are involved. The composite Aggregate reliability is,

B. Analysis of Aggregate Reliability of Workflow Patterns:

Several tasks of a process are executed one after each other. A task is enabled after the previous task has finished and before the successive task has been started. Sequential workflow pattern is common in web service composition. For example, in student leave application process, after student submits leave request, then leave account check is performed. This "After-Then" scenario is the representative case that fits in Sequence pattern. Sample web services are selected based on sequential workflow patterns. The aggregate reliability value is calculated with samples. The Table 4 shows the results.

Table 4 Sequential Workflow Patterns- Aggregate Reliability

X	Y	Aggregate Reliability
0.956	0.937	0.896
0.946	0.957	0.905
0.956	0.946	0.904
0.958	0.996	0.954
0.936	0.926	0.867

Parallel Spilt involves parallel tasks; For example, in student fee payment process through bank credit card process, after student submits request, credit check including id check and existing account check are performed by different owners in parallel. This is a typical case which needs Parallel Split pattern. Two tasks are finished concurrently, credit card result task runs immediately. Credit card result task is Synchronization block. Table 5 shows the aggregate reliability of sample parallel patterns.

Table 5 Parallel Workflow Patterns- Aggregate Reliability

X	Y	Z	W	Aggregate Reliability
0.956	0.956	0.926	0.937	0.793
0.946	0.937	0.936	0.957	0.794
0.956	0.946	0.996	0.946	0.852
0.958	0.957	0.958	0.996	0.875
0.936	0.956	0.946	0.926	0.784

This pattern is also extensively used as business assessment. The routing conclusion is made dynamically allowing it to be postponed to the latest possible instant at runtime. For example, in student fee payment process, following routing branch depends on credit result task's result, if result is pass, then process goes to credit card producing task; if result is not pass, then process goes to request fail task; if result is pending, then process goes to next review task. This is a representative scenario which needs Conditional split pattern. After either branch is executed according to different credit result task's result, notify student task is needed in common. Here we need join pattern. Table 6 shows the sample conditional spit-join pattern and its aggregate reliability value.

Table 6 Conditional Workflow Patterns- Aggregate Reliability

X	Y	Z	W	Aggregate Reliability
0.956	0.956	0.926	0.937	0.851
0.946	0.937	0.936	0.957	0.848
0.956	0.946	0.996	0.946	0.865
0.958	0.957	0.958	0.996	0.913
0.936	0.956	0.946	0.926	0.827

Simple loop and dual loop occurs occasionally in the business process. Table 7 and Table 8 shown the sample result of simple loop and dual loop workflow patterns.

Table 7 Simple Loop Workflow Patterns- Aggregate Reliability

X	Aggregate Reliability
0.937	0.748
0.957	0.817
0.946	0.778
0.996	0.980
0.926	0.715

Table 8 Dual Loop Workflow Patterns- Aggregate Reliability

X	Y	Aggregate Reliability
0.956	0.937	0.675
0.946	0.957	0.686
0.956	0.946	0.692
0.958	0.996	0.810
0.936	0.926	0.611

After evaluating the five basic workflow patterns, we found that sequential workflow pattern is having the high reliability rate as compared to other workflow pattern. Among the basic workflow patterns, sequential workflow patterns is having high reliability rate. If number of services involved in the patterns increases, then reliability rate gradually decreases. Aggregate reliability of the workflow patterns inversely propositional to the number of services involved.

VI. CONCLUSION

The reliability estimation technique for composite web service is proposed. Basic workflow patterns are analyzed based on the aggregate reliability values. Workflow pattern with high degree of reliability is identified. Service reliability is based on node reliability and link reliability. In this paper, node reliability is considered. In future, the link reliability can also be considered for the estimation of aggregate reliability of the composed web services.

VII. ACKNOWLEDGMENT

This research work is being funded by the Department of Science and Technology (DST), Government of India.

VIII.REFERENCES

[1] D. A. Menascé, "QoS Issues in Web Services," 1EEE Internet Computing, 6 (6), 2002, pp. 72-75.

[2] J. Cardoso, J. Miller, A. Sheth, 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.

[3] R. Sumra, and D. Arulazi, "Quality of Service for Web Services – Demystification, Limitations, and Best Practices," Developer.com, 2003

[4] S. Ran, "A Model for Web Services Discovery with QoS", ACM S1Gecom Exchanges, 4 (1), spring 2003, pp 1-10.

[5] Zo, H., Supporting 1ntra- and 1nter-Organizational Business Processes with Web Services, Ph.D. Thesis, the University of Wisconsin-Milwaukee, 2006.

[6] Saaty, T. L., the Analytic Hierarchy Process, McGraw- Hill, New York, NY, 1980.

[7] Arsanjani, B. Hailpern, J. Martin, P. Tarr, "Web Services: Promises and Compromises", ACM Queue, 1 (1), March 2003, pp. 48-58.

[8] A.L. Goel, "Software Reliability Models: Assumptions, Limitation, and Applicability," 1EEE Transactions on Software Engineering, Vol. SE-11 (12), 1985, pp. 1411-1423.

[9] Musa, J. D., Software Reliability Engineering, McGraw- Hill, New York, NY, 1999

[10] L.-J. Jin, V. Machiraju, and A. Sahai, "Analysis on Service Level Agreement of Web Services", HP Labs Report HPL-2002-180, HP Laboratories, 2002.

[11] Xie, M., Software Reliability Modelling, World Scientific Publishing Co., Singapore, 1991.

[12] Abdelkarim Erradi, Piyush Maheswari, "A Broker-based Approach for Improving Web Services Reliability", Procedings of the IEEE International Conference on Web Services, 2005.

[13] Pat. P. W. Chan and Michael R. Lyu, "Dynamic Web Service Composition: A New Approach in Building Reliable Web Service", Proceedings of 22nd International Conference on Advanced Information Networking and Applications, 2008.

[14] Jianbin Huang, Heli Sun, "A Reliable Web Service Implementation Approach for Large Commercial Applications", IEEE Computer Society, pp. 82-86, 2008.

[15] Jiang Ma, Hao-peng Chen, "A Reliability Evaluation Framework on Composite Web Service" , Proceedings of IEEE International Symposium on Service-Oriented System Engineering, pp. 123-129, 2008.

[16] Duhang Zhong, Zhichang Qi, "A Petri Net Based Approach for Reliability Prediction of Web Services", Spinger-Verlag Berlin Heidelberg, pp. 116-125., 2006.

[17] J. Zhang, and L.-J. Zhang, "Criteria Analysis and Validation of the Reliability of Web Services-Oriented Systems", in Proceedings of the 1EEE International Conference on Web Services (1CWS'05), Orlando, Florida, July 2005, pp. 621-628.

[18] Hangjung Zo, Derek L. Nazareth, Hemant K. Jain, "Measuring Reliability of Applications Composed of Web Services", Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[19] Aalst, "Control-Flow Patterns",

http://www.workflowpatterns.com/patterns/control/, 2009

[20] San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee and Cheng-Hung Chen, "Dynamic Web Service Selection for Reliable Web Service Composition", IEEE Transactions on Services Computing, vol. 1, no. 2, 2008, pp. 104-110.

[21] Sandeep Chatterjee, James Webber, "Developing Enterprise Web Services: An Architect's Guide", Prentice Hall, 2003.

Short Bio Data for the Author

Dr. L. Arockiam, working as an Associate Professor in the Department of Computer Science, St.Joseph's College, Tiruchirappalli, Tamil Nadu, India. Having 23 years of experience in teaching and 14 years of experience in research. Published 85 research articles in the International / National Conferences and reputed Journals. Presented two research articles in the Software Measurement European Forum in Rome. Chaired many technical sessions and delivered invited talks in National and International Conferences. Authored a book on "Success through Soft Skills" and "Research in a nutshell". His research interests are: Software Measurement, Cognitive Aspects in Programming, Web Service, Mobile Networks and Data mining.

N.Sasikaladevi, doing research in St.Joseph's College, She presented papers in various national and international conferences. Published papers in various journals. Authored a book titled "Programming in C#.NET". Her research interests are: Web Services, QoS issues on Web Services and Workflow patterns.