An Approach to Estimate Response Time of Composition in SOA Environment for Business Compliance

Chellammal Surianarayanan[1] *, and Ravishankar Palaniappan[2]

[1]Department of Computer Science
Bharathidasan University Constituent Arts & Science College
Navalurkuttapattu, Trichy 620009, India

[2]Cognizant, Chennai-600045, India

Email: chelsrsd@rediffmail.com, ravishankar.palaniappan@gmail.com

Abstract

Organizations adopting Service Oriented Computing (SOC) implement complex business processes through service composition where more than one services from different domains are composed in a particular pattern to implement a given process. One of the central needs of composition is that it should take place within the stipulated time as specified in the Service Level Agreement (SLA).The primary quality attribute used to measure performance of service composition is response time. In this paper, an approach is proposed to estimate the response time of composition using the response time of individual service invocation events towards checking compliance as per SLA. In case of any deviation, service providers will notify the clients and take corrective tasks to improve the quality of their service offers. A case study has been taken up with a composite shipping service which involves composition of services both internal and external to the organization. Different scenarios are discussed and results are presented.

Keywords: *Business compliance, Service Level Agreement, Service Composition, SOA environment.*

1. Introduction

Service Oriented Computing (SOC) promotes the development of business applications as reusable assets in the form of interoperable services with well defined interfaces. Web services[1] is the primary technology stack used to implement SOC and enables the implementation of complex business processes through service composition where more than one atomic services are combined in a specific pattern to realize the given process. In many real situations, service composition involves many services, also from different domains. For example, consider a user's query, 'Malaysia-Package-Plan' which involves the complete functionality of planning and booking facility for one week's package tour to Malaysia. The given user's query is represented as a workflow which is a combination of various abstract tasks from different organizations such as *Book_flight*, *Book_hotel* and *BooK_cab* as in Fig. 1.

Figure 1: Travel Plan *("Self Compiled")*

Each task is as an abstract function with one or more input and output parameters. Before composing a business process, for each task, a concrete service has to be identified based on functional characteristics. Service composition has to handle another aspect, the Quality of Service (QoS) characteristics given by clients which are equally important as functional characteristics. So, from the resulting set of functionally identified services, suitable services have to be chosen according to the QoS requirements of clients. QoS of services refer to non-functional characteristics of services such as execution time, latency, through, reliability, scalability, availability, etc[2]. The QoS attributes can either be deterministic (values are known before invocation, such as cost) or non-deterministic (values are computed based on the data collected during run time monitoring, such as latency)[3]. In service based applications, QoS plays a crucial role in deciding the overall success and accomplishment of composition as services invoked are distributed over network and they are invoked by many clients concurrently. During invocation of services, different issues such as network failure, system failure, service unavailability, human error [4] which may lead to the process not get completed as expected by the client. Of various QoS characteristics, performance of composition is found to be the most general requirement of almost all clients. The research work [3] categorizes different attributes, namely, execution time, latency, response time, round trip time, scalability, and throughput as performance related attributes. Service combinations having low round trip time, high throughput and high scalability will be chosen as optimized combination for implementing business processes. Every client wants his query to be fulfilled within some stipulated time and if services demanded by the clients are not returned to them within their stipulated time, it will create negative impact on the reputation of service provider. Service providers will lose their clients. Hence, estimating time of composition becomes an important aspect while maintaining and providing the agreed service levels to clients. Service Level Agreement(SLA) is a contract between service providers and consumers which defines the level of service quality that should provide and expect respectively[5].Meeting the time of composition as per SLA help organizations not only to stay focused on customer satisfaction but also to prepare effective resource planning of resources[6]. Further monitoring QoS has led to several studies as discussed in [7].

In this paper, it is proposed to analyze and estimate the time taken for service composition in terms of response time monitored for individual service invocation events. The following two aspects are of importance while estimating the performance of composition. At first, composition may take place either statically or dynamically [8] depending upon nature of how frequently the business requirements are changing. More important is that in the case of dynamic scenarios, the SLA monitoring should be aligned with the speed of changes in the applications environment [9]. Secondly, the services involved in composition may be internal to the organization or external to the organization. The proposed work analyzes and estimates the performance of composition considering these two aspects. A case study is presented as proof of concept to show how the proposed approach can be employed to estimate the response time of composition.

Rest of the paper is organized as follows. Section 2 describes the research work related to the theme of the paper. Section 3 describes the proposed approach. Section 4 presents a composite shipping service as a case study. Section 5 concludes the paper.

2. Related Work

Monitoring of QoS must be carefully planned and structured [10] as it ensures quality of services [11]. Different techniques for monitoring quality of services are classified and discussed in [12]. This paper reviews a QoS model which covers various dimensions of service quality and

proposes metrics to enhance QoS measurement on the service side [13]. Service monitoring involves evaluation of SLA-available QoS metrics based on measurable data such as response time, throughput, availability, etc [14]. Keller and Ludwig [15] presented a framework for monitoring SLAs in web services-based systems with dynamic business agreements. In [16] an approach is proposed to monitor, runtime errors, timeouts and violations of functional contracts of service compositions using assertions. An approach which adapts Business Process Execution Language (BPEL) processes to tolerate runtime faults such as overly loaded service invocations by redirecting the request to some other proxies where the slow services are replaced by alternate services [17]. In [18], an approach based on Parallel Performance Monitoring Service is proposed to monitor the run-time performance of dynamically composed media services. In [19] an approach is presented to assess (i) the efficiency of orchestration, (ii) the reliability of orchestration and (iii) the effect of service failure on orchestration. In [20], a proactive approach is proposed to take initiatives for replanning as soon as a deviation is detected. The research works such as [21] and [22] analyze QoS of business processes in terms of QoS of individual services statically at design time. In [23], a framework which detects SLA violation using statistical hypothesis testing is proposed. Further, the work focuses on repair policies, monitoring and process evolution. The proposed approach is different from [23]. At first it computes the response time of composition using the previously monitored values of individual service events. Further, a worst case observation is used to compute the QoS of composition. Then this practically computed value is compared with that of the one arrived using SLA specified value. If there is a deviation, it is notified and an appropriately alternate service which could yield better performance is chosen.

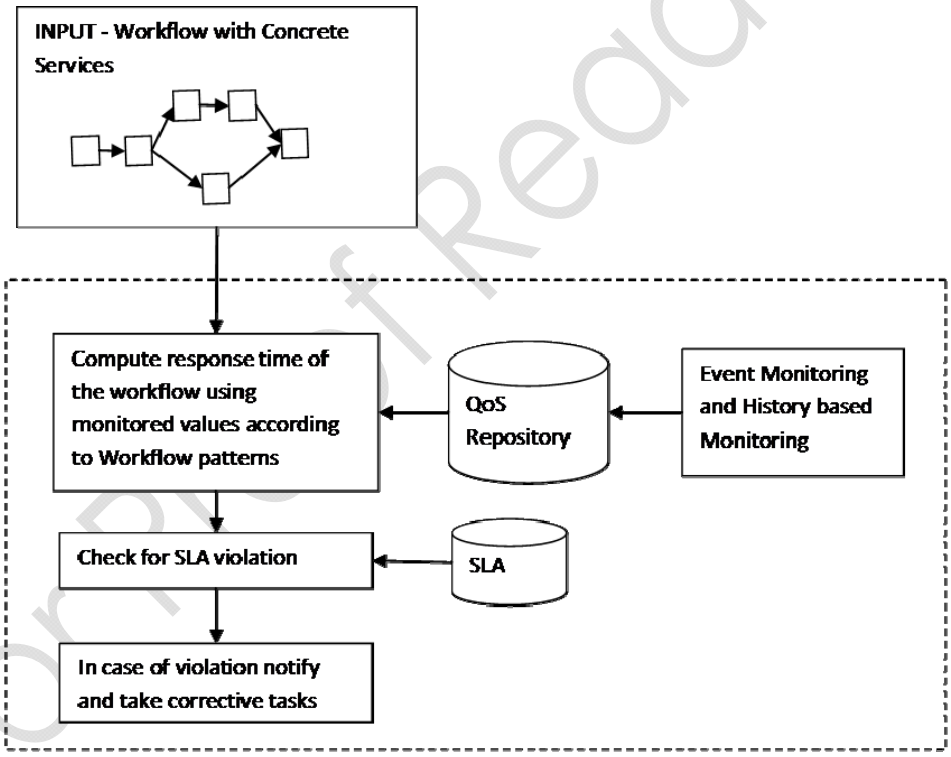

Figure 2: Block diagram of the Proposed Approach *("Self Compiled")*

3. Proposed Approach

Towards monitoring the performance of service composition, an approach is presented to estimate the response time of composition using the response time of individual services. The estimated value is compared with SLA against the values stated in SLA in order to check whether the estimated value will satisfy a client. In case the estimated response time deviate from that of SLA, service providers will notify the clients and take further action. Block diagram of the proposed approach is shown in Fig. 2.

The approach consists of 3 steps, namely, (i) Archival of response time of individual service services from Event monitoring and History based monitoring (ii) Find out workflow pattern and compute response time according to the event monitoring and history based monitoring, and (iii)

detection of SLA violation. Step 1 is a regular monitoring activity and in this step, the response times of individual services are extracted and archived from data obtained during event monitoring and history based monitoring. In step 2, the response time of the workflow (which represents the given process) is computed using aggregation functions of response time of different execution patterns and monitored values. In step 3, the computed time is compared with the value specified in SLA for checking compliance. In case of deviation, notification and other corrective actions will be taken up.

3.1. Archival of response time of individual services

The computation of response time of workflow needs the availability of response time of individual services. The proposed approach considers that event monitoring and history based monitoring are being done during the implementation of business processes. In event monitoring, a monitoring component of run time environment monitors and records the events performed during the execution of a business process. History-based monitoring is an extension of event monitoring in which the previous events are archived in a repository which will be reasoned later. In the proposed approach, the response time of individual service events are extracted from event and history-based monitoring and archived in a QoS repository (Please refer to Fig. 2).

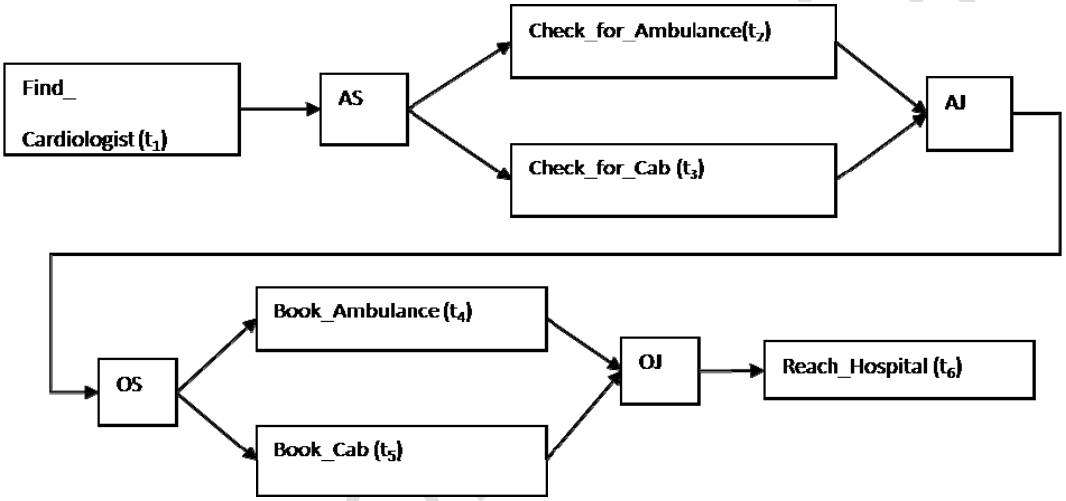

Figure 3: Combinational Workflow *("Self Compiled")*

3.2. Method for computing response time of business process

The proposed approach considers that a given business process is represented as workflow[24] and the complexity of a workflow varies from sequential as in Fig. 1 where the tasks of the workflow take place in sequence to a combinational one as in Fig. 3 where the tasks are invoked in different execution patterns.

Most common execution patterns include sequential pattern, AND pattern, OR pattern and Loop pattern. The workflow given in Fig. 3 contains six tasks, viz., 'Find_Cardiologist', 'Check_for_Ambulance', 'Check_for_Cab', 'Book_Ambulance' and 'Book_Cab' and 'Reach_Hospital'. The tasks are have IDs, t_1, t_2, t_3, t_4, t_5 and t_6. In Fig. 3, 'AS' and 'AJ' indicate AND Split and AND Join of an AND execution pattern respectively 'OS' and 'OJ' indicate OR Split and OR Join of an OR execution pattern respectively. As shown in Fig. 3, the task, t_1 is executed in sequence. After the execution of t_1, the tasks t_2 and t_3 are executed in AND fashion (i.e. in parallel). After the parallel execution and based on the outcome of AND pattern, t_4 or t_5 are executed in OR fashion (i.e. either t_4 or t_5). After the execution of OR pattern, t_6 is executed.

Once business process is represented as workflow, the response time of workflow (i.e. business process/composition) is computed using the aggregate functions for response time for different execution patterns. In this proposed model, response time for sequential, AND/OR, Loop patterns are computed using the aggregate functions given in one of our previous research work [25]. The computation of response time for different execution patterns are exemplified as different cases below.

Case 1: Sequential Workflow

Consider a sequential workflow, W with n tasks. Let t_1, t_2, t_3,….t_n denote the ID of 1^{st}, 2^{nd}, 3^{rd},…n^{th}

task. Let $rt(t_i)$ denote the response time of t_i^{th} task. Let $rt(W)$ denote the response time of W and is computed using

$$rt(W) = \sum_{j=1}^{n} rt(t_j) \quad (1)$$

Case 2: AND/OR execution pattern

Consider an AND or OR execution pattern. An AND or OR pattern consist of more than one path of execution. In case of AND pattern, all the paths are executed in parallel and in case of OR pattern, depending on run time conditions one of the path will be executed. Let u denote an AND/OR pattern. Let u contain P_1, P_2, …, P_k paths.

Let rt(P_1), rt(P_2), …, rt(P_k) denote the response time of P_1, P_2, …,P_k. The response time of path is taken as the sum of response time of all tasks present in that path. Let $rt(u)$ denote the response time of AND/OR pattern and it is computed using

$$rt(u) = \max\{rt(P_i) \mid 1 \le i \le k\} \quad (2)$$

Table I: Response time of different tasks in the Example Workflow *("Self Compiled")*

Task ID	Response Time (in milli seconds)
t_1	50
t_2	60
t_3	40
t_4	100
t_5	200
t_6	300

Case 3: Loop pattern

Consider a loop, 'u' with z sequential tasks with 'm' number of iterations. Let $rt(u)$ denote the response time of loop pattern. The value of $rt(u)$ is computed using

$$rt(u) = m \times \sum_{j=1}^{z} rt(t_j) \quad (3)$$

From the above cases, it is clear that the aggregate functions express the response times of different patterns in terms of response times of individual tasks. For an example, consider the workflow given in Fig.3. Once the workflow is constructed for a given business process, concrete services for the individual tasks are discovered and identified according to functional and non-functional characteristics respectively. Let us assume that concrete services are selected for composition. Consider typical values of response times of these services as given Table I. Let rt(W) denote the response time of the given work flow.

The response time of the workflow given in Fig. 3 is computed using the response time for different execution patterns as

$$rt(W) = rt(t_1) + \max\{rt(t_2), rt(t_3)\} + \max\{rt(t_4), rt(t_5)\} + rt(t_6) \quad (4)$$
$$rt(W) = 610$$

3.3. Detection of SLA violation

.

The computed value of response time of the workflow is compared against values of SLA. SLA may be expressed in plain text or in standard format such as IBM'S Web Service Level Agreement (WSLA). Sample snippet of SLA in WSLA format is given in Fig.4 which specifies the value of *responsetime* attribute should be less than 5 units.

```
<ServiceLevelObjective>
<Obliged>providername</Obliged>
<Validity>
 <Start>2016-09-06</Start>
    <End>2017-09-06</End>
<Validity>
<Expression>
<Predicate xsi:type="wsla:less" >
<SLAparameter>ResponseTime</SLAparameter>
<Value> 5</Value>
....
</ServiceLevelObjective>
```

Figure 4: Sample Snippet of SLA *("Self Compiled")*

If the computed time violates the SLA, the client is notified and further corrective actions are taken up to improve the service offer.

4. Case Study

4.1. Case Description

To evaluate the proposed approach, the approach is employed to a realistic case, a composite shipping service. Let Organization X provides integrated shipping services to its customers. The organization has a web site through which its customers will interact. The case taken for study is a composite service. The organization implements the composite service by composing service from its partners via service composition. In order to realize the queries for shipping service, it is essential to combine different services, namely, pricing service, order service, pick & pack service, route/transport service and status notification service as shown in Fig. 5. Pricing service provides

pricing for shipping the given weight of goods between the given pair of points/zipcodes in the World. Clients or users can place the orders using Order service. Billing to the client and payment of client is performed using invoice and payment service. According the placed order, the goods are picked and packed up using pick & pack service. Proper routing and transportation of goods are provided by route/transport service. Once the shipping is started till it is transported to the said destination, status is notified using status notification service.

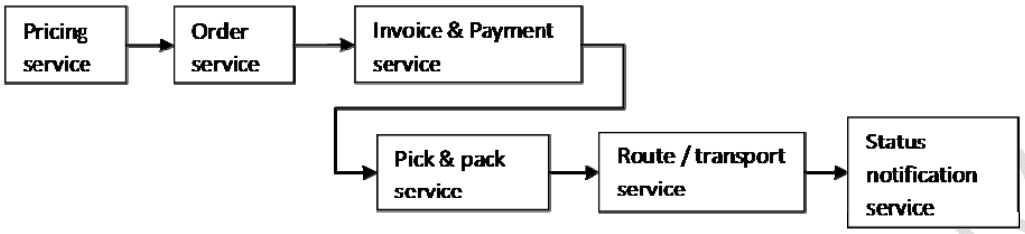

Figure 5: Integrated Shipping Service *("Self Compiled")*

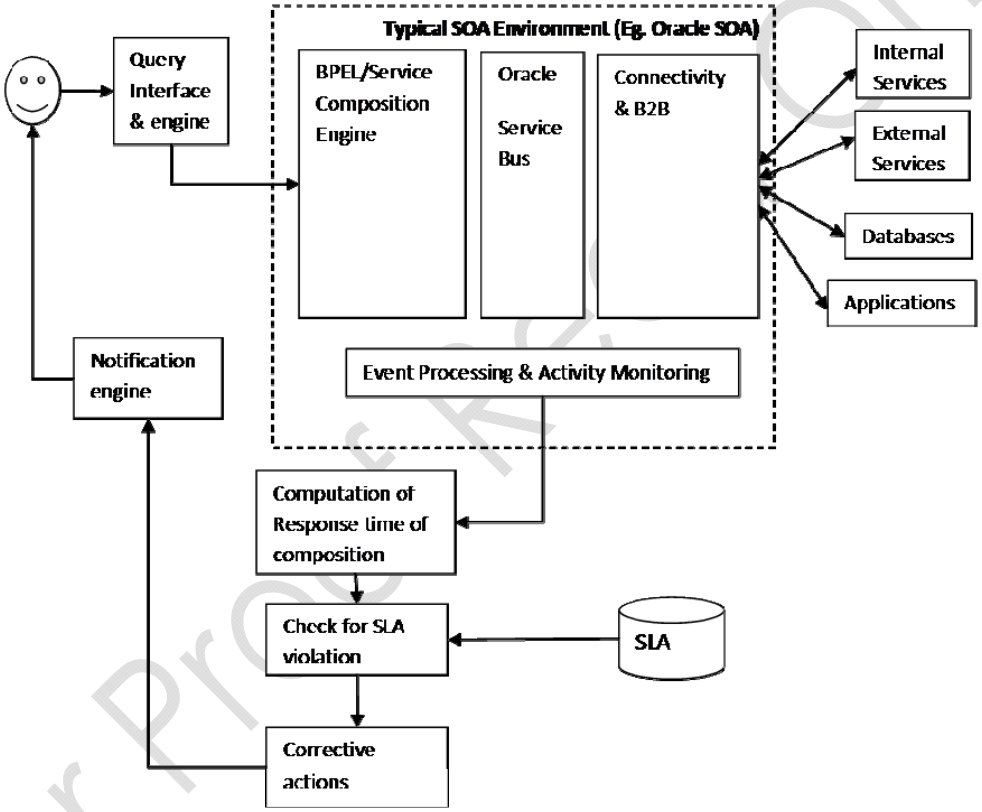

Figure 6: Implementation Model of the case under Study *("Self Compiled")*

Of the six services involved in shipping services, the services namely pricing service, order service, invoice & payment service and status notification service are internal to the organization whereas pick & place service and route/transport service are external to the organization. In implementing this composite service, the organization has to depend on other services providers for pick & pack service and route/transport service. The case can be realized as a sequential workflow.

4.2. Implementation Model for shipping service

Typically, business processes are implemented using SOA runtime environment such as Oracle SOA suite as in Fig. 6.

Service clients interact with concerned organization (organization A in this case) through its website which acts are query interface. The queries are handled by query engine and realized in oracle SOA runtime environment. Core components of any typical SOA Environment such as Oracle suite includes Service Bus for discovery, provisioning and integration of services, Business Process Execution Language (BPEL) Engine for composing discrete services into business process,

Event Processing for delivering actions in real-time and Business Activity Monitoring for monitoring business process and taking decisions. Also, the suite includes Business Rules, other connector architecture adapters for establishing connections to other packaged applications, legacy and mainframe applications, other applications, external services, technologies and protocols, files, databases, etc. Now while implementing the case under study, different scenarios can occur. The shipping service can be implemented statically or dynamically based on the nature of changes in business process.

Table II Typical Structure of SOA Instance Table *("Self Compiled")*

ECID	EY9XR00V100000000	EY9XR00V100000000	EY9XR00V100000000
ID	12755304	12755305	12755306
PARENT_ID			reference:12295575
CONVERSATION_ID			urn:57BC7530B4AC11E3BFEDCF4212BBD757
COMPOSITE_DN	ORDER/SendOrderMasterProvABCSImpl!1.1*soa_e6be5fbd-c83e-4943-8a39-1f8b124548e8	SHIPMENT/EBSInventoryItemMasterProvABCSImpl!1.1*soa_e6be5fbd-c83e-4943-8a39-1f8b124548e8	UTIL/XxintEventSoapCreateV1EsEbiz!1.3*soa_84125d42-d1b6-4226-b9ac-6baeb709f7c0
SOURCE_NAME	ebsordermasterprovabcsimpl_client_ep	ebsinventoryitemmasterprovabcsimpl_client_ep	CreateEvent_ep
SOURCE_TYPE	binding.ws	binding.ws	binding.local
SOURCE_ACTION_TYPE	operation	operation	operation
SOURCE_ACTION_NAME	process	process	execute
BATCH_ID			
BATCH_INDEX	0	0	0
BUSINESS_STATUS			
INDEX1			
INDEX2			
INDEX3			
INDEX4			
INDEX5			
INDEX6			
TITLE			
TAGS			
TEST_RUN_NAME			
TEST_RUN_ID			
TEST_SUITE			
TEST_CASE			
STATE			
LIVE_INSTANCES			
STATE_COUNT			
HAS_ASSOC			
VERSION			
PARTITION_DATE	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM
TENANT_ID	-1	-1	-1
CREATED_BY	XXX@YYY_SEIBEL	XXX@YYY_SEIBEL	XXX@YYY_SEIBEL
CREATED_TIME	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM
UPDATED_BY			
UPDATED_TIME	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM	26-MAR-14 06.03.25.044000000 AM

4.2.1 Implementing shipping service statically

In static composition, the concrete services which undergo composition are identified in design time itself. In the case of integrated shipping service, the concrete services which realize different tasks such as pricing, order, invoice & payment, pick&pack and route/transport and status notification are known at design time itself. Of the 6 services, 4 are internal to the organization whereas pick & pack service and route/transport service are provided by some other service providers. In static form, along with 4 internal services, organization A knows which service it is going to invoke to realize pick & pack and route/transport services at design time itself. In this case the Organization A has SLAs with other providers for these two tasks.

In regard to QoS attributes of services, there are two scenarios. In one scenario, the external services might be invoked previously by Organization A in which case, the history based monitoring yields, practically observed values for response time. In the other scenario, the external service may be invoked for the first time. As mentioned earlier, response time is the algebraic sum of latency and service execution time. Service execution time will be given by provider and available in SLA whereas the value of latency is not same for all client and it depends on the location of client. When a service is invoked for the first time, the latency involved in found out by sending test or mock request (not the actual invocation) to the concerned IP. Now, Organization A has to compute the response time of composition using the individual service invocation events which is monitored and archived as part of Business Activity Monitoring during the realization of business processes in SOA environment. The case under study is implemented in Oracle SOA environment. In oracle SOA, individual the service events are captured in SOA instance table. Structure of SOA instance table is given Table II.

There are 35 rows and 4 columns in Table II. As in Table II, each task is assigned with ID. The tasks which undergo composition have the same attribute 'ECID'. For example, from Table II, it is clear that the tasks having IDs 12755304, 12755305 and 12755306 share the same ECID, EY9XR00V100000000 which implies that these tasks are involved in implementing a same single business process through service composition. Further, the response time of an individual service is taken as the difference between the attributes 'CREATED_TIME' and 'UPDATED_TIME'. In practice, the response times of individual service invocation are collected from SOA instance table and archived in a separate database also. Further, as the case under study is realized as a sequential workflow, its response time is computed using (1).

4.2.2 Implementing shipping service dynamically

In dynamic form, the services which undergo composition are determined at run time. In the case under study, how dynamism has been brought into composition is illustrated as given below.

Let us consider that the organization A has say 3 vendors for pick & pack service, namely, Vendor-1, Vendor-2 and Vendor-3. It is considered that the services are selected for composition based on two non-functional attributes, namely cost (for the stated weight of goods and at stated location of client) and availability. The cost and availability values of pick & pack service from different vendors are typically taken as given Table III.

Table III: Typical values of Cost and availability for pick&pack service provided by different vendors
("Self Compiled")

Service	Vendor	Cost	Availability
Pick & Pack-1	Vendor-1	50$	90%
Pick & Pack-2	Vendor-2	60$	95%
Pick & Pack-3	Vendor-3	75$	100%

Table IV: Typical values of Cost of route/transport service provided by different vendors
("Self Compiled")

Service	Vendor	Cost	availability
Route/ transport-1	Vendor-A	500$	90%
Route/ transport-2	Vendor-B	550$	95%
Route/ transport-3	Vendor-C	580$	96%
Route/ transport-4	Vendor-D	600$	100%

Consider that service Pick&Pack-1 was initially chosen for composition based on client's cost constraint. But the service is found to be unavailable. Now the composition engine has to dynamically select other service say Pick&pack-2 for implementing pick&pack task.

Similarly, in the case of route/transport task, Organization A has tie-up with say 4 vendors, say, Vendor-A, Vendor-B, Vendor-C and Vendor-D who provide routing and transporting facility. Let us consider some typical values for cost of different vendors as in Table IV. Here also, Organization

would have statically bound to Vendor-A for route/transport. But in reality, with Vendor-A, vehicle for transporting the goods may not be available in which case, Organization has to deal with other vendors and accordingly the service provisioning will change.

In the above situations, the services are determined dynamically at run time and the values of response time are computed. In dynamic form also, a concerned service may be invoked either for the 1st time or for the nth time. If the service is invoked for the first time, its response time is computed as the sum of latency obtained with test/mock request with concerned IP and service execution time given in SLA. If the service is invoked for other than first time, an average value of response time from the history based monitoring is taken. The response time of composition is computed according to (1) and the computed values are checked against the SLA between Organization A and the client. If the estimated response time deviates from the specified limits in SLA, the client will be notified and further actions will be taken up.

5. Conclusion

In this paper, an approach is proposed to estimate the response time of composition using response time of individual service invocation events towards detecting compliance with SLA. In this approach, it is considered that a given business process is represented as workflow consisting of a combination of tasks. Once concrete services are identified based on functional and non-functional requirements, the response time of the workflow is computed by using standard aggregation functions and response time of individual service events which are archived from event based and history based monitoring. The estimated response time of composition is compared with the pre-agreed values of response time in order to check the compliance of composition with SLA. In case, if the estimated values are found to deviate from that of SLA, service providers will notify the service client and take further corrective tasks to improve the quality of their service offers. A specific case, composite shipping service has been taken up to illustrate the proposed approach. The case involves invoking services which are both internal and external to the organization. The case has been analysed for composition in both static and dynamic scenarios. Implementation model for the case study and discussions are presented. As a future work it is proposed to develop a framework for the estimation of response time of service composition considering various parameters like service priority, service downtimes and service risks.

References

[1] K.Gottschalk, S.Graham, H.Kreger, J.Snell, “Introduction to Web services architecture”, IBM Systems Journal, Vol. 41, No. 2, 2002.

[2] Yu, Hong Qing and Reiff-Marganiec, Stephan, “Non-functional Property based service selection: A survey and classification of approaches”, In: Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop co-located with The 6th IEEE European Conference on Web Services, 12 - 14 Nov 2008, Ireland, Dublin.

[3] Anton Michlmayr, Florian Rosenberg, Philipp Leitner, Schahram Dustdar, “Comprehensive QoS monitoring of web services and event-based SLA violation detection”, Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing, pp.1-6, 2009.

[4] Emmanuel Mulo, Uwe Zdun, Schahram Dustdar, “Monitoring Web Service Event Trails for Business Compliance”, Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA), vol., no., pp.1-8, 2009

[5] Mohd Hilmi Hasan, Jafreezal Jaafar, Mohd Fadzil Hassan, “Monitoring web services’ quality of service: a literature review”, Artificial Intelligence Review, Springer, vol. 42, issue 4, pp. 835-850, Dec. 2014.

[6] Allenotor D, Thulasiram RK, “A fuzzy grid-QoS framework for obtaining higher grid resources availability”, In: Proceedings of the 3rd International Conference on Advances in Grid and Pervasive Computing, pp. 128–139, 2008.

[7] Simmonds J, Gan Y, Chechik M, Nejati S, O’Farrell B, Litani E, Waterhouse J, “Runtime monitoring of web service conversations”, IEEE Trans Serv Comput 2(3): 223–244), 2009.

[8] Atif Alamri, Mohamad Eid and Abdulmotaleb El Saddik, “Classification of the state-of-the-art dynamic web services composition techniques”, International Journal of Web and Grid Services, Vol. 2, No.2, pp. 148-166, 2006.

[9] Heiko Ludwig , Katerina Stamou, Mohamed Mohamed, Nagapramod Mandagere, Bryan Langston, Gabriel Alatorre, Hiroaki Nakamura, Obinna Anya, Alexander Keller, “rSLA: Monitoring SLAs in dynamic service environments” in ICSOC, LNCS 9435, pp. 139-153, 2015 edited by Alistair Barros, Daniela Grigori, Nanjangud C. Narendra, Hoa Khanh Dam, published by Springer-Verlag.

[10] Heward G, Müller I, Jun H, Schneider J-G, Versteeg S, “Assessing the performance

impact of service monitoring", In: Proceedings of the 21st Australian software engineering conference, IEEE Computer Society, pp 192–201, 2010.
[11] Modica GD, Tomarchio O, Vita L, "Dynamic SLAs management in service oriented environments", Journal of System Software 82(5): 759–771, 2009.
[12] Sara Zirak, Naser Nematbakhsh, Kamran Zamanifar, "An overview of methods for monitoring web services based on the quality of services", International Journal of Research in Engineering and Technology, vol. 3, issue 3, pp. 718-723, Mar-2014.
[13] Artaiam, N.; Senivongse, T., "Enhancing Service-Side QoS Monitoring for Web Services," in 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp.765-770, 6-8 Aug. 2008.
[14] M. Papazoglou. "Web Services: principles and tenchnology", Pearson Publications, 2008.
[15] A. Keller and H. Ludwig, "Defining and monitoring service level agreements for dynamic e-business," in USENIX Conference on System Administration (LISA'02). Philadelphia, USA: ACM Press, pp. 189–204, November 2002.
[16] Baresi, L., C. Ghezzi and S, Guinea, "Smart monitors for composed services", Proceedings of the 2nd International Conference on Service Oriented Computing, Nov. 15-19, ACM Press, New York, USA., pp: 193-202, 2004.
[17] Ezenwoye, O. and S.M. Sadjadi, "Proxy based approach to enhancing the autonomic behavior in composite services", Journal of Network., 3: pp. 42-53, 2008.
[18] G. Maria Kalavathy, P. Seethalakshmi, "Parallel Performance Monitoring Service for Dynamically Composed Media Web Services", Journal of Computer Science 5(7): 487-492, 2009.
[19] Tamrat Tewoldeberhan, Marijn Janssen, "Simulation-based experimentation for designing reliable and efficient Web service orchestrations in supply chains", Electronic Commerce Research and Applications, Vol. 8, pp. 82-96, 2008
[20] Nicolas Repp, Rainer Berbner, Oliver Heckmann, and Ralf Steinmetz, "A Cross-Layer Approach to Performance Monitoring of Web Services", Emerging Web Services technology part of Whitestein Series in Software Agent Technologies, edited by Cesare Pautasso, Christoph Bussler published by Birkhäuser Basel, 21-32, 2007
[21] H. Xiao, B. Chan, Y. Zou, J.W. Benayon, B. O'Farrell, E. Litani, and J. Hawkins, "A Framework for Verifying SLA Compliance in Composed Services", In the Proceedings of *ICWS*, pp. 457–464, 2008.
[22] L. Mei, W. K. Chan, and T. H. Tse, "An Adaptive Service Selection Approach to Service Composition" In Proceedings of *ICWS*, pp. 70–77, 2008
[23] Adina Mosincat, Walter Binder, "Automated Performance Maintenance of Service Compositions", 11th IEEE International Symposium on Web Systems Evolution, Edmonton, AB, pp. 131-140, 2009
[24] Daniela Barreiro Claro and Patrick Albers and Jin-Kao Hao, "Web Services Composition, Semantic Web Processes and their applications, pp.195-225, 2006
[25] Chellammal Surianarayanan, Gopinath Ganapathy, and Manikandan Sethunarayanan Ramasamy. "An Approach for Selecting Best Available Services Through a New Method of Decomposing QoS Constraints." Journal of Service Oriented Computing and Applications Service Oriented Computing and Applications, Springer, vol. 9, issue 2, pp.107-138, June 2015.

Author Biographies

Chellammal Surianarayanan has 10 years of R&D experience as Scientific Officer in Indira Gandhi Centre for Atomic Research, India. She completed Ph.D. in Semantic Services. She has been working as Assistant Professor since 2009.

Ravishankar Palaniappan completed Masters in Computer Applications. He has around 12 years of R&D experience in SOA environment. He worked as Principal Consultant, SOA in Schneider National and Technology Manager in Cognizant Services Chennai. Presently he is working as Data Scientist in CTS.