Prediction of Composite Service Execution Duration before Change in Service Composition

Leila Mollaey
Department of Computer Engineering Shabestar Branch Islamic Azad University
Shabestar, Iran

Mir Ali Seyyedi
Department of Computer Engineering South of Tehran Branch Islamic Azad University
Tehran, Iran

ABSTRACT
When user requirements alter in composite services, there is a need to change service composition or add and remove some services. Notwithstanding service level agreement, it is vital for service provider to assess quality properties. Computing execution duration process after any changes in service composition will be time consuming if it is completely performed ab initio. In this paper, a graph theory based approach is proposed, in which non-formal business process execution language is mapped into XBFG graph. then execution duration of primary composite service is computed and the results are stored. Using these results, execution duration of composite service can be computed after changing service composition in a less timely manner.

General Terms
Prediction of execution duration of composite service before change in service composition algorithm.

Keywords
changing in service composition, composite service, Extended BPEL Flow Graph, execution duration, critical path.

1. INTRODUCTION
Service level agreement is a contract between service provider and its customers, in which quality properties of the service and its credibility criteria are determined. Hence, assessment of service quality properties is important. Execution duration of composite service is one of the most important quality properties. After any changes in requirements, which are followed by changing the service composition, service provider needs predicate new execution duration of composite service.

So far, few studies have been conducted in the area of quality properties assessment and execution duration of composite service. In these studies, quality properties assessment is performed ab initio after any changes in service composition. For example, Reference [1] investigated timed constraints using timed petri net and also references [2]-[3] proposed some procedures for quality properties assessment based on runtime monitoring. Reference [4] presented a framework that predicts deviations from service level agreement by monitoring at run time and prevents them. Composite service performance was also predicted and evaluated in reference [5]-[6].

In this paper, an approach in which XBFG graph as the formal model of composite service is used, is presented. Taking into account critical path and execution duration of composite service before any change in its composition, execution duration is predicted after applying some changes in service composition. According to this approach, there is no need to perform all calculations ab initio and primary composite service computing information is used for calculating execution duration for new service composition and no repetitive calculations are performed to compute new execution duration.

2. Extended BPEL Flow Grap
Control flow model is used in change impact analysis and regression testing path selection on BPEL process in [7], Li et al.[8] exploited a test selection minimization algorithm based on [7]. BPEL is a semi-formal flow language with complex features such as concurrency and hierarchy[9]. BPEL Flow Graph (BFG) is a control flow model proposed by Y. Yuan et al. to describe BPEL process[10]. BPEL composite service is the combination of process and component services interacting with the process. In order to operate change impact analysis on composite service rather than the process solely, proposed a model called XBFG, that Bixin Li used this model for functional testing of composite service in [11] and [12] .

XBFG is defined as a triple < N,E,F >, which N denotes the node set and these nodes used in different forms for modeling activities and services, that including N = IN ∪ NN ∪ SN ∪ EN ∪ MN ∪ CN. E = CE ∪ ME, which denotes the edge set. Control Edge (CE) linking the BPEL activities that used in BFG to express control flows in BPEL and and add Message Edge (ME) linking IN and SN to denote message calling relationship between process and its component services. F is the field of XBFG element, element is the general designation of node and edge. XBFG nodes are classified into six types:

Interaction Node (IN), which is mapped from those basic activities with which component services interact, including <invoke>, <receive>, <reply> and <onMessage> in <pick>.

Normal Node (NN), which is mapped from other basic activities of BPEL, such as <assign>, <wait> and so on. Additionally, <onAlarm> activity in <pick> is also mapped to NN.

Service Node (SN), which is mapped from the partnerLinks defined in BPEL, a SN represents a component service the process interacts with.

Exclusive Node (EN), which is mapped from those structural activities providing conditional behavior, including <if>, <pick>, <while> and <repeatUntil>. EN is divided into Exclusive Decision Node (EDN) and Exclusive Merge Node (EMN).

Multiple Node (MN), which is mapped from the <link> with its <joinCondition> value equals to ”OR” or null. MN is divided into Multiple Branch Node (MBN) and Multiple Merge Node (MMN).

Concurrent Node (CN), which is mapped from the <flow> activity and <link> with its <joinCondition> value equals to ”AND”. CN also has two forms, Concurrent Branch Node (CBN) and Concurrent Merge Node (CMN). The symbols of each kind of XBFG elements are shows in figure 1.

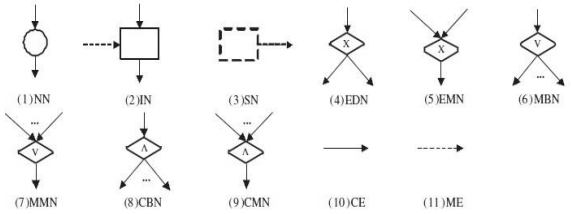

Fig 1: Symbols of XBFG elements

3. The Proposed Approach

Initially execution duration of primary composite service is computed and the results are stored. Then, execution duration of composite service after changing service composition is predicted by using stored results and without any need to perform duplicated operations.

3.1 Computing Execution Duration of Primary Composite Service

Total execution duration of primary composite service is determined by selecting the critical path(cp)[13] and designating its execution duration as total execution duration. The critical path is a path from initial state to the final state, which has the longest total execution duration. Initially, XBFG model of composite service is constructed and then paths of the model are determined. Execution duration for each path is computed by summing execution durations of the activities in the path. Finally, a path with the longest execution duration is selected as the critical path and its execution duration is designated as total execution duration composite service.

3.1.1 Calculation of Execution Duration for Basic Activities

In order to calculate execution duration for each path, it is needed to calculate execution duration for all activities in the paths. The total execution duration of one path is obtained by summing execution durations of all activities in that path. In this section and following section, rules for calculation of execution duration of each activity in BPEL process are given according to what proposed in [14]. Processing time of each activity includes the intrinsic processing time T_{intr} which refers to all internal operations of the WS-BPEL engine associated with the execution of this activity. The value of T_{intr} can be obtained by means of benchmarking.

<invoke> c: if the invocation is synchronous (request-reply), the communication time can be determined by (1)

$$T_{comm}[c] = d[n] + \frac{h_i[m] + h_o[m]}{j[n]} + b[m] \quad (1)$$

where *m* and *n* are the invoked operation and the node that provides the corresponding web service, respectively. d[n] is current network latency, hi[m] and ho[m] are average sizes of input and output (fault) messages for the web service operation m. j[n] is the maximum throughput of the network link between the integrator node and the node n and b[m] is the compute time of the operation m. These sizes can be determined on the basis of XML Schema definitions in WSDL documents. The concluding formula is (2):

$$T[c] = T_{intr} + T_{comm}[c] \quad (2)$$

<receive> c: the execution duration of this activity is obtained from summation of the T_{intr} and expecting waiting time of the received message.

<wait> c: the expecting execution duration of this activity is obtained from summation of the T_{intr} and waiting time required to reach next operation.

<reply> c: the execution duration of this activity is determined by (3):

$$T[c] = T_{intr} + d[n] + \frac{h_o[m]}{j[n]} \quad (3)$$

<assign>, <empty>, <validate> and <throw> c: the execution duration of these activities are only equal to T_{intr}.

Fault, compensation and termination handlers: ignored because exception, compensation and termination handling is not part of normal behavior of the scope.

<rethrow> : ignored because this activity can be called only from a fault handler.

<compensate Scope> and <compensate>: ignored because compensation can be initiated only from a fault handler or a compensation handler.

<terminate> and <exit> : the execution duration of these activities are only equal to T_{intr}.

3.1.2 Calculation of Execution Duration for Compound Activities

In this section, (recursive) aggregating formulas and algorithms for compound activities are given[14].

<sequence> c: the execution duration of this activity with nested activities $c_1 \ldots c_k$, is obtained by formula (4):

$$T[c] = T_{intr} + \sum_{i=1}^{k} T[c_i] \quad (4)$$

<repeat Until> and <while> c: (execution duration of these activities) with expected loops count f ($f \geq 1$ for <repeat Until>, $f \geq 0$ for <while>) and expected duration of the body of the loop T_{body}, is obtained by formula (5):

$$T[c] = T_{intr} + f.T_{body} \quad (5)$$

<scope> c: execution duration of this compound activity with nested activity c_1 and optional event handlers c_2 . . . c_k, is obtained by formula (6):

$$T[c] = T_{intr} + \max(\{T[c_1 \dots T[c_k]\}) \quad (6)$$

In addition, the execution duration of activities that are as branching, do not be calculated separately and will be calculated execution duration of each activity's branch in dependent path.

3.2 Determining XBFG Graph Details Storage Sets

Some the sets are used for storing details of composite service before and after changing its composition. In this section, these sets are introduced. P and P´ sets are used for storing the paths obtained by XBFG model of primary composite service and composite service after change in its composition. N and N´ sets store the elements present in paths of P and P´. N_{add} set contains elements that are not in N but in N´ and N_{del} is the set of elements that exist in N and but not in N´. P_s is set of those paths that are subject to change and is obtained by union of P_{s1} and P_{s2}. P_{s1} is the set of paths that change due to deletion of the service and P_{s2} is the set of paths that change due to addition of services. P_{s1} and P_{s2} are obtained by algorithm 1 in reference [11]. Execution durations of paths are gathered into set T and execution duration of added and deleted elements are gathered into set τ.

3.3 Prediction of Composite Service Execution Duration Before Changing Service Composition

This process take places in 4 steps: in the first step, the changes are applied to XBFG model of primary composite service to obtain XBFG model for new composite service. Then in the second step, paths in new graph are determined. In the next step, execution duration regarding added and deleted elements are calculated and in the fourth step the critical path is selected and its execution duration is calculated using algorithm 1 which will be described in the following section. Finally execution duration of the critical path is designated as total execution duration.

3.3.1 Selecting the Critical Path After Changing Composite Service Composition

After changing composite service composition, it should be first determined what type is this change. Changes in service composition are classified into four types which include: adding or removing services in the critical path of primary composite service and adding or removing services in a path other than the critical path of primary composite service. In a case in which a service is added on the critical path, the new critical path overlaps the previous critical path and its execution duration will be equals to summation of the execution duration of the previous composite service and execution duration of the added element on the critical path. In the case when a service is deleted from previous critical path, all paths are compared with each other and the new critical path is selected. When adding services in a path other than critical path of composite service, the execution durations of all paths which changes due to addition of new services, i.e. paths in P_{s2} set, should be compared and the new critical path is selected. Finally when a service is deleted from a path other than the critical path, it can be stated that deletion of a service has no influence on the critical path and consequently its execution duration. Algorithm 1 describes the process of selection of the new critical path and calculation of its execution duration. In this algorithm, pathComparision algorithm which was described in [11] is used. map function takse the new path and converts it to its previous state.

```
input : N, N´, P, P´,T,τ,CP
output : CP´, T(CP´)
1 newCriticalPath(T, τ,N,N´,P,P´,CP):CP´,T(CP´);
2 CP´ = CP;
3 T(CP´) = T(CP);
4 pathComparision(P, P´, N, N´): Ps1, Ps2;
5 for each path of Ps2 : ps2 do
6   for each element of Nadd : nadd do
7     if nadd Є ps2 then
8       T(map(ps2)) = T(map(ps2)) + τ (nadd);
9       if map(ps2) = = CP then
10        T(CP´) = T(CP´) + τ (nadd);
11        CP´ = ps2;
12    end
13  end
14 end
15 end
16 for each path of Ps2 : ps2 do
17    if map(ps2) != CP && T(CP´) < T(map(ps2)) then
18          CP´ = ps2;
19          T(CP´) = T(map(ps2));
20      end
21 end
22 for each element of Ndel : ndel do
23  if ndel Є map(CP´) then
24    for each element of Ndel : ndel do
25      for each path of Ps1: ps1 do
26        if ndel Є ps1 then
27          T(map(ps1)) = T(map(ps1)) - τ (ndel);
28        end
29      end
```

```
30      end
31      for each path of  P´ : p´ do
32          P´ = P´- p´
33          for each path of  P´ : p″ do
34              if  T(p´) < T(p”) then
35                  CP´ = p″;
36                  T(CP´) = T(p″);
37              end
38          end
39      end
40      return  CP´, T(CP´);
41     end
42  end
43 return  CP´, T(CP´);
44 end
```

Algorithm 1 : the algorithm for new critical path selection and calculation of its execution duration

4. Evaluation Mechanism

In order to evaluate the proposed approach, calculating execution duration of loan approval composite service is considered after changing its composition under four types of changes using two approaches: 1) by the proposed approach and taking into account the information obtained from calculation of primary composite service execution duration and 2) performing all calculations for determination of new composite service execution duration regardless of execution duration of primary composite service. Loan approval process was described in [15]. XBFG graph for primary loan approval composite service is demonstrated in Figure 2. In order to calculate execution duration of primary composite service, initially paths are determined on XBFG graph and then execution duration of each path is calculated and finally execution durations of all paths are compared to select the critical path, i.e. the path with maximum execution duration. The execution duration of the critical path is considered as total execution duration.

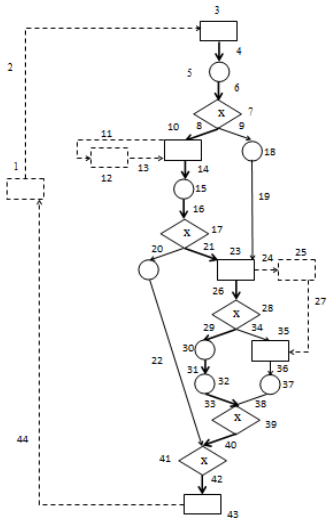

Fig 2 : XBFG Graph Constructed for Primary Loan Approval

When a service is added on the previous critical path, without making any comparison it can be concluded that critical path will not change. XBFG grph for loan approval process after adding service on last critical path direction is demonstrated in Figure 3.

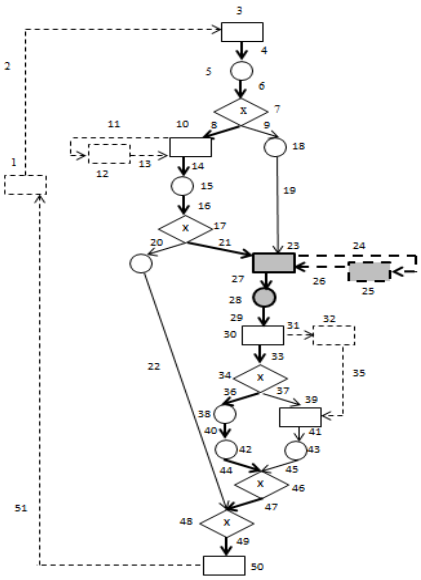

Fig 3: XBFG Graph Constructed for Loan Aapproval after Adding Service on last Critical Path Direction

In the case in which a service is removed from the previous critical path, only paths which change due to removal of the service are compared with each other and the critical path is selected. When a service is added on another path, all paths in P_{s2} are compared with each other and the new critical path is selected. XBFG grph for loan approval process after adding service on a path other than last critical path direction is demonstrated in Figure 4.

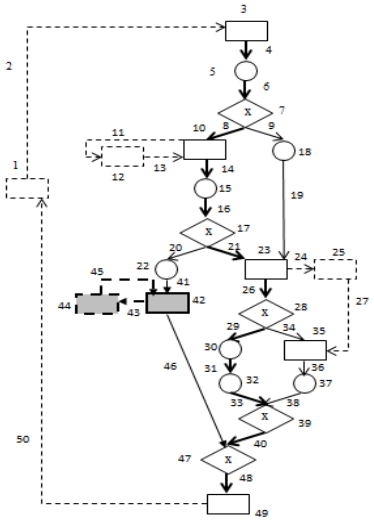

Fig 4: XBFG Graph Constructed for Loan Aapproval after Adding Service on a Path other than last Critical Path Direction

Finally if a service is removed from another path other than the critical path, the critical path remains unchanged. The numbers of required comparisons made for candidacy as the critical path using two approaches: 1) the proposed approach and 2) performing calculations ab initio are listed in table 1.

Table 1. The Number of Needed Comparisons for Candidacy as The Critical Path

Approach / Types of changes	The Proposed approach	Performing calculations ab initio
Adding a service on the last critical path	0	|P´-1|
Removing a service from last critical path	|P´-1|	|P´-1|
Adding a service on a path other than critical path	$\|P_{s2}\|$	|P´-1|
Removing a service from a path other than critical path	0	|P´-1|

5. Conclusion and Future Works

The critical path can be selected with fewer numbers of comparisons using the proposed approach and its execution duration be calculated. Therefore, it can be stated that using this approach is economical when user requirements will be subject to many changes in the future. In addition, other quality properties as well as execution duration are considered in service level agreement that assessment of them is important for service provider too. These quality properties such as: execution cost, reputation, reliability and availability. execution cost is the amount of money that a service requester has to pay for executing, reputation is a measure of service 's trustworthiness, reliability is the probability that a request is correctly responded within the maximum expected time frame and availability is the probability that the service is accessible. we will be proposed approaches to predict other quality properties of composite services before change in service composition in our future works.

6. REFERENCES

[1] Guilan, D., Rujuan, L., Chongchong, Zh., Changjun, Hu. 2008. Timing Constraints Specification and Verification for Web Service Compositions, IEEE Asia-Pacific Services Computing Conference.

[2] Moser, O., Rosenberg, F., Dustdar, S., 2008. Non-Intrusive Monitoring and Service Adaptation forWS-BPEL, International World Wide Web Conference, China, pp.21-25.

[3] Huang, T., Wu, G., Wei, J. 2009. Runtime Monitoring Composite Web Services Through Stateful Aspect Extension, JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, pp. 294-308.

[4] Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S. 2010. Monitoring, Prediction and Prevention of SLA Violations in Composite Services, IEEE International Conference on Web Services.

[5] Dong, Y., Xia, Y., Zhu, Q., Yang, R. 2009. A Stochastic Approach to Predicting Performance of Web Service Composition, JOURNAL OF COMPUTERS, VOL. 4, NO. 6.

[6] Dong, Y., Xia, Y., Sun, T., Zhu, Q. 2010. Modeling and Performance Evaluation of Service Choreography based on Stochastic Petri Net, JOURNAL OF COMPUTERS, VOL. 5, NO. 4.

[7] Liu, H., Li, Z., Zhu, J., Tan, H. 2007. Business Process Regression Testing, In: Proceedings of the 5th International Conference on Service Oriented Computing (ICSOC 2007), LNCS 4749, pp. 157-168.

[8] Li, J., Tan, F., Liu, H., Zhu, J., Mitsumori, N. 2008. Business-process-driven gray-box SOA testing, IBM System Journal, pp.457-472.

[9] Zheng, Y., Zhou, J., Krause, P. 2007. An Automatic Test Case Generation Framework for Web Services, JOURNAL OF SOFTWARE, VOL. 2, NO. 3.

[10] Yuan, Y., Li, Z. Sun, W. 2006. A Graph-search Based Approach to BPEL4WS Test Generation, In: Proceedings of the International Conference on Software Engineering Advances (ICSEA'06) , Tahiti , pp. 14-14.

[11] Bixin, Li., Dong, Q., Shunhui, J., Di, Wang. 2010. Automatic Test Case Selection and Generation for Regression Testing of Composite Service Based on Extensible BPEL Flow Graph, 26th IEEE International Conference on Software Maintenance in Timisoara, Romania.

[12] Wang, Di., Li, B., Cai, J. 2008. Regression Testing of Composite Service: An XBFG-based Approach, IEEE Congress on Services Part II.

[13] Zeng, Li., Benatallah, B., Dumas, M. Quality Driven Web Services Composition, 12th international conference on World Wide Web, Pages 411 - 421.

[14] Rud, D., Kunz, M., Schmietendorf, A., Dumke, R. 2007. Performance Analysis in WS-BPEL-Based Infrastructures.

[15] Cao, D., Felix, P., Castanet, R., Berrada, I. 2009. Testing Service Composition Using TGSE tool, 7th IEEE International Conference on Web Services, Los Angeles : United States, 2009.