Indirect Substitution Method in Combinable Services by Eliminating Incompatible Services

Forough Hematian Chahardah Cheriki
Department of Computer Engineering
Yazd Branch, Islamic Azad University
Yazd, Iran

Sima Emadi*
Department of Computer Engineering
Yazd Branch, Islamic Azad University
Yazd, Iran

***Abstract*—Service-oriented architecture is a style in information systems architecture with the aim of achieving loose coupling in communication between software components and services. Service, here means software implementation, is a well-defined business function that can be used and be called in various processes or software. An organization can choose and composite the Web services that fulfill its intended quality of service. As the number of available Web services increases, choosing the best services to composite is challenging and is the most important problem of service composition. In addition, due to the utilization of systems in dynamic environments, service characteristics and users' needs are constantly faced with changes which lead to deterioration of service, unavailability and quality loss of services. One of the ways to deal with this challenge is substitution of a Web service with another service, which is done at the runtime and dynamically. Substitution is both direct and indirect. Though there are many related works in the field of direct substitution, still no work is done for explaining substitution based on the indirect method, and works were conducted only on direct substitution. In this method, there are many problems such as the incompatibility of important services in composition. To solve the problems in this method and other challenges in this paper, considering a subset of inputs and outputs, qualitative parameters and service composition, simultaneous and dynamic service composition and use of the fitness function of genetic algorithm to compare the compositions are done. In addition, in substitution, a table which contains the best possible substitutes with dynamic updates through multi-threading techniques is provided. The results obtained by the analysis and evaluation of the proposed method, indicates the establishment of compatibility between the services, and finding the best possible substitute to reduce substitution time.**

Keywords*—component; indirect substitution; SLA; service composition; quality of service

I. INTRODUCTION

Through the discovery and development of Web services, an organization can choose and composite the Web services that satisfy its business needs and service quality. At the same as the number of available Web services increases, choosing the best services becomes more challenging for the given practice. Service quality plays an important role in the selection and composition of services. During the composition of services in a workflow or application to complete the process until the final result, services should be compatible in regard to the given the inputs, outputs and functions. The main problem of service composition is that the values of service quality may change from initial estimates during the implementation. The service may be unavailable or unreliable or may not offer other suitable solution. Thus, services should be evaluated dynamically to complete the program. Changes in the values of service quality may lead to failure of the expected adoption of the program for maintaining certain cases such as costs and response time. Two main issues of which adverse events require -the need for re-selection of service and services reprogramming include: extra time for selection process and lack of service composition compatibility with other service quality constraints [1]. In this paper, a method for composition of Web services that perform re-selection and prevent the deviation from the constraints of service quality, after reprogramming through the definition and evaluation of a potential substitution is proposed. In the proposed method re-selection and substitution are done only when it is needed. To help re-selection and substitution, services will be filtered based on their importance, because only the services that provide optimal solutions are considered in planning processes [1].

Substitution means the alternation of one component instead of another component; so that during the movement, the same component output be produced and meet the same needs with the replaced output component. Substitution means compatibility of Web service with the client requests and better setting than other competing Web services [1]. Substitution is both direct and indirect, till now there are many related works in the field of direct substitution. The main problem and error method in a direct substitution is that the compatibility between services in substitution process is ignored due to the lack of regulation of services in the substitution process. In addition, one cannot benefit the substitution services composition in this method and there is no possibility of automatic substitution. To solve this problem and to create compatibility between the services in substitution, indirect substitution method is used in regard to SLA violations. Every service includes its specific quality of service (QoS). In substitution process, one is trying to prevent the failure of qualitative constraints; However by maximizing the qualitative characteristics which should be augmented such as reliability and by minimizing the qualitative characteristics that should be reduced such as cost, maintaining the upper and lower limits to the maximum extent of assumed constraints and maintaining compatibility between available services, substitutability becomes possible. Finding compatible Web services to be replaced with a Web service after being selected is essential in each of the following events [2]:

- Web service should be eliminated during the runtime.

* Corresponding author

- Better service need to be available through a new Web service.
- A new version of the selected Web service should be available.

Substitution of Web services is possible only by fulfilling the following two conditions [2]:

- Web services should have the same functionality.
- They should be able to obtain their respective interfaces. The services must be compatible with each other for substitution.

The second section of this paper describes the concepts and terminology related to substitution and service composition. In the third section, the proposed method is described. In the fourth section, results of evaluation are discussed. Then in the fifth section a review of the earlier works is presented. Finally, the result of the research is provided in the sixth section.

II. Concepts and Definitions

A. direct and indirect substitutions

The direct substitutes are feasible because they can replace the failing service without further adjustment to the plan. Logically, it is very easy to compute the number of direct substitutes. [1].

In a direct substitution, the violated service can be replaced regardless of a subset of the inputs and outputs with exactly equal the input and output of a service, and without creating a service composition to achieve equal functionality.

Indirect substitutes are feasible service instances that can replace the failing service after adjustments with other service replacements are made to the resulting plan. [1].

B. Service quality traits

Quality of service is a subset of non-functional traits, which is different from various perspectives [2, 3, and 4].

In general, a non-functional trait divides into types of quantitative and qualitative traits [5]. A qualitative trait of the proposed method is defined here:

1) Response Time

The time it takes in which a service performs its task. This time is long as from the time of user's request to the time in which the answer is received, which is obtained in accordance with existing equations in Table 1 [5].

2) Service Reliability

If the service continue to work correctly and consistently in the specified period and under certain conditions. Despite the availability, reliability is defined in terms of time interval rather than time instant, and is obtained by the equations in Table 1.

3) Availability

The content and timeliness that service is immediately available for the operation and performance of its functions. Availability is related to system failure. The availability is obtained by existing equations in Table 1 [5, 6].

4) Cost

The cost for using Web services is obtained by the equations in table 1 [7].

TABLE I. Qualitative Parameters of Service and Their Calculations

Qualitative parameters	formula
Response time	Response time = Execution time+ Network time Response time= process time+ Transfer Time+ Latency Time
Reliability	Reliability=1- Probability of Failure
Availability	Availability=Uptime / (Up time +Down time) Availability =MTBF[1] / (MTBF+MTTR[2])
Cost	Total Cost = Service execution Cost+ (Network transportation/Transaction) Cost

5) SLA rules

It is a legal document format based on XML language, which consists of the three parties of the contract, guarantee terms and service terms; Figure (1) [8].

Fig. 1. SLA format [8]

Agreement contexts consist of general information such as parties and life cycle of the agreement. This information includes the address and profile of service producer, consumer, etc. The service terms consists of two parts: service reference and service properties. In service reference, an availability URL to the services is determined, and in service properties, information on quality parameters and indices is determined. Service terms are a key element of the SLA. The Guarantee terms consists of the quality objectives and financial agreements [8].

Main SLA requirements include [8]:

- SLA format should be a clear definition of a service, so that the consumer should understand the service functions
- Provide a level of service efficiency.
- Methods of monitoring service parameters and regulatory reporting format must be defined.
- Penalties when services are not met.

[1] Mean Time Between Failure
[2] Mean Time to Repair

C. static service compositions

Static service composition is created at design time and software system architecture. Adopted components will be selected, connected, and finally compiled and deployed. This case is suitable if serviced components rarely be changed or do not change in general.

D. Dynamic service composition

Service environment is a dynamic and very flexible environment. The new service will be available every day, and number of service providers will be growing. Ideally service processes must be able to accommodate to environmental changes and customer requirements with minimal user intervention.

E. Composite service

Composite service is created by composition of multiple services. Existing services in composite service might be implemented in different locations and in various fields. But they should interact with one another to achieve a goal. Service composition is referred to process of services development ranging from conventional services and compositing services.

III. Describing the Proposed Method

In [1], an approach is proposed in which due to the use of qualitative parameters in genetic algorithms and the techniques based on the total weight in the objective function of the algorithm, the potential diversion of restrictions during the program implementation has decreased, that leads to finding the best possible solution in accordance with qualitative traits of user's request. However, the method has limitations and the problems. This method is directly focused on replacing static service composition that limits the turnover in the composition and lack of focus on service composition which leads to service incompatibility. Thus, service incompatibility and composition limitations are among the most important problems in this approach. Another limitation of this method is lack of automatic service substitution, and as a result time consuming substitution process.

In this paper, indirect substitution is used to solve problems and mentioned limits as well as substitution optimization. In the proposed method substitution process dynamically improved by enjoying the [1] algorithm. The ability to build composite services, incompatibility problems of substitution and composition have been solved. In addition, in the proposed method, a solution for automatic substitution and reduction of process time is provided.

The proposed method consists of three main steps.

1) Preprocessing

- Selecting from the database.
- Receiving the requested qualitative parameters
- Receiving the requested weight of qualitative parameters
- Receiving the incoming and outgoing requests

2) Service composition

- Logical composition step
- Creating qualitative model
- Physical composition step
- services filtering
- Composition algorithm step
- Creating a service composition
- Creating the composition
- Finding the best composition
- Genetic Algorithm
- fitness function of the proposed model Reprogramming

3) Reprogramming

- Service Substitution
- Updating the substitution table

A. pre-processing phase

Pre-processing is all operations that must be performed before service composition, so that composition process be done according to the requests and qualitative parameters of user's requests along with maintaining the limits and SLA rules.

- The first step: selection of database repository

In this step, according to user requests, demand-services are called from UDDI database. These services include all similar requested services or services that are similar in the input or output.

- The Second Step: Receiving the requested qualitative parameters

In this step, a value must also be considered for each qualitative parameters of availability, response time, reliability, cost and substitution. These parameters' input values range between zero to one, and is determined by the user's request.

- The third step: Receiving the requested weight of qualitative parameters

Considering the fact that in this study, composition operations are carried out on the basis of the users' requested weight of qualitative parameters, in this step the user enters the requested weight of each qualitative parameter.

- The fourth step: Receiving the incoming and outgoing requests

In this step, the user request his desired input and output with respect to the functionality of services.

In [1] service composition is performed statically and through the genetic algorithm. It involves different stages and steps. To improve and solve the concerning problems in the proposed model, the process is modified and composition is done dynamically.

B. Service composition phase

1) Logical composition step

- Creating qualitative model

2) Physical composition step

- Services filtering

3) Composition algorithm step

- Creating a service composition and compatibility
- Fitness function

1) The first step: Logical composition

The first step to obtain the optimal composition of Web services is creating a suitable model to describe qualitative characteristics. This model must be agreed by the client and service provider. The qualitative model can be circular, parallel, serial or probable. To calculate the qualitative model, based on the type of limits and qualitative model, the pattern of aggregation functions in Table (2) can be used. Sequential method is used in this proposed method.

TABLE II. AGGREGATION FUNCTIONS FOR CALCULATION OF SERVICE QUALITATIVE PARAMETERS [1]

Attribute	Dimension Type	Constraint Type	Aggregation function			
			Sequential Invocation	Probabilistic Invocation	Structured Cycles	Parallel Invocation
Cost	Decreasing	Upper	$\sum_{i=1}^{n} cost$	$\sum_{i=1}^{M} Pi * Cost(s)$	K*cost(s)	$\sum_{i=1}^{P} cost$
Response Time	Decreasing	Upper	$\sum_{i=1}^{n} RTim$	$\sum_{i=1}^{M} Pi * RTime(si)$	K*Time(s)	MAX(s0..sp)
Availability	Increasing	Lower	$\prod_{i=1}^{N} Ava$	$\prod_{i=1}^{M} Pi * Avail(si)$	Avail(s)k	$\prod_{i=1}^{P} Ava$
Reliability	Increasing	Lower	$\prod_{I=1}^{N} Rel($	$\prod_{i=1}^{M} Pi * Rel(si)$	Rel(s)k	$\prod_{I=1}^{P} Rel($

In this step of the proposed model, unlike [1], in order to improve the composition method, the values of input and output are received from the user in the pre-processing. Then, based on these values, only those services which include the requested input and output or a subset of users' request would be called. In fact, at this stage of the proposed model, filtering operation is performed on the service call on the basis of functionality. And in any composition, searchable input and output are specified by user which leads to the method's dynamic trait. The called services will be elected as a candidate. This stage which leads to selection of services is called logical composition. This set of services in form of a set of workflows as candidate services move to the next step which is the physical composition. For example, suppose the user requested service S with input and output of (A, B.C, D). As a result, services with a subset of the input and output such as S1 (A, B.J, K) and S2 (A, H.C, D) and S3 (A.F) etc. will be called.

2) The second step: physical composition

At this stage, workflows are filtered based on user's requested qualitative parameters. Then to perform service composition they will be entered to the composition algorithm. In the given example, called services will be filtered qualitative characteristics entered by the user. In this case, it is assumed that the user requests the values of qualitative parameters in Table (3).

TABLE III. EXAMPLE OF REQUESTING USER'S QUALITATIVE PARAMETERS

availability	0.28
reliability	0.28
Response time	0.91
cost	0.91
substitution	0.11

However, if called service S1 has qualitative characteristics less than requested characteristics. The service will not be considered as a candidate for this composition. In fact, called services are filtered based on qualitative parameters.

C. Filtering substitutable services

Unlike [1], in the proposed model, a filtering operation based on qualitative parameters and weights is done dynamically before calculating the quality of services on the basis of CIFs in Table 1. As a result, the quality of service is calculated only for services that include qualitative parameters and weights; therefore additional and unnecessary calculations will be avoided. This filter is rarely applied in linear form due to its complexity. The qualitative model for every workflow is calculated by existing equations in Table (1), and thus quality of service for each workflow can be obtained.

After the pre-processing filter and reduction of candidate services we enter into algorithm phase. As mentioned in the previous section, unlike [1], composition operation is done dynamically based on the received input and output in the pre-processing phase as well as during the construction of composite service. After creating different service plans, the fitness function of genetic algorithm is used to compare and select the best composition with the highest fitness. For example, suppose you have a service consists of two inputs and outputs. In this method, based on the input and output received in preprocessing phase, three lists will be created and called services will be entered according to the inputs and output as shown in Table (4).

TABLE IV. AVAILABLE LISTS IN THE PROPOSED METHOD

Services in which their inputs are a subset of user's requested inputs	Input	LISTØ
Service in which their outputs are a subset of user's requested outputs	Out put	LIST1
Proper composition occurs through inputs and outputs and a subset	Creating service composition	LIST2

Various scenarios intended for creating List2 and service composition are as follows:

- The first scenario

According to equation (1), if the inputs of selected service in List2, are the subsets of the outputs in listØ service and if its outputs are a subset of inputs in list1 service, the condition of service composition, which is the accessibility to user's requested functionality, will be established and the compatibility will not be violated. Thus, the service in list2 will be considered for composition with services in listØ and list1 and it will be removed from list2. As a result composition process will be successfully performed, and the resulting service plan will be displayed in the output. The first scenario is given in example (1).

- The second scenario

As mentioned in Equation 2, if one of the conditions in the first scenario is violated; for example if inputs of the selected service are equal to ListØ outputs but its outputs is not a subset of List1, it results to incompatibility. Thus selected service in list2, becomes the basis among the services based on input and output to create service composition on the next survey. In other words, a service in which its outputs are a subset of the inputs in list2 service and its inputs are a subset of outputs in the mentioned service, will be sought for, and service composition will be formed. This cycles continues till the service composition condition is established, and service composition that includes a subset of the input and output according to user's request be made. Thus in the second scenario, creating a service composition with compatibility is done. For a better understanding, the proposed algorithm is expressed in example (1).

The user requested service S with the inputs of A, B and outputs of C, D. According to the proposed solution, three list is created. In the listØ S1 and in list1 service S2 are called. As the example shows, the two first rows, compatibility and service composition without creating a service composition is established like the first scenario. In the third row, S9 violated the terms of compatibility, therefore, among the set of candidate services, service S10 in which its inputs are a subset of the outputs in service S9 and its outputs are a subset of the inputs in service S8, are called. This cycles continues till compatibility condition is fulfilled. In this example, through the service composition of S9 and S10, composition problems are solved and compatibility can be achieved. Therefore, by implementation of S8, S10, S9, and S7 compatible composition is created through composite service.

Equation (1) of the first scenario:

$$\begin{cases} \text{if input list2 } \hat{c} \text{ output listØ} \\ \& \\ \text{if output list2 } \hat{c}\text{binput list1} \end{cases} \Rightarrow$$

Composite Services in list Ø &list2 & list 3 Equation (2) of the second scenario:

$$\begin{cases} \text{if input list2 } \hat{c} \text{ output listØ } \cap \text{output list 2 } \not\subset \text{input list1} \\ \& \\ \text{if input list 2 } \not\in \text{out put list Ø} \end{cases}$$

$\Rightarrow$ Research in data Service input & output ĉ list2 then composite Services listØ & list1&list2

TABLE V. SERVICE COMPOSITION ALGORITHM

	User's request	**List ∞**	**List $_1$**	**List $_2$**	**Output**
	S(A,B.C, D)	S_1(A,B.J, K)	S_2(A,H.C, D)	S_3(J,K.A,H)	S_1,S_3,S_2
First scenario	S(A,B.C, D)	S_4(B.C)	S_5(H.D)	S_6(B.C,H. T)	$S_4,S_6.S_5$
Second scenario	S(A,B.C, D)	S_7(B.F)	S_8(C.D)	S_9(F.K) S_{10}(K.C)	S_7,S_9,S_{10},S_8

This algorithm is carried out dynamically and constantly by using multi-threaded technique. After composition and creating different service plans, the fitness function of genetic algorithm is used to compare and select the best composition with the highest fitness. In addition, by entering the compositions into fitness function of the algorithm, SLA rules and violations will be investigated; as the qualitative parameters will be determined by entered weights by the user and calculation of qualitative model according to the table (2), and in case of violations, they will be outdated.

D. Fitness function

In the proposed model, after composition process, fitness function according to equation (3) is used to compare the compositions and presenting the best composition.

Given the importance of qualitative parameters in this method, a fitness function using the total weight which transfers multi objective problem to a single objective problem is used. As previously mentioned, the weights are selected based on the user's preferences and needs. Just as shown in equation (3), the composition from the previous steps is called wj; and w1, w2, w 3, w4 and w5 weights are provided by the user. In accordance with the calculation contract of qualitative model in table (1), the availability of all the services in the Wj composition are multiplied. Thus, the compositions' rate of availability will be obtained. Equation (4) shows the calculation of the qualitative parameters rating for each composition. Reliability and substitution composition are calculated in the same way. As Table 1 shows calculation contract for qualitative model, addition is used for rating the cost parameters and response time of composition. This means that cost of services composition are added together and the result is the rating of Wj composition cost. Similarly, the calculated response time will be entered to fitness function of the. In this function, as the reduction of cost parameters and response time are superior standard of composition, these parameters are placed in the denominator of fraction. Finally, the composition with best fitness will be selected and displayed in the output.

Equation (3) of fitness function [1]:

$$\text{Fitness} = w_{1*\text{Availability (wj)}} + w_{2*\text{Reliability (wj)}} + w_{3*\text{replacebility (wj)}}$$

$$W_4*\text{Cost (wj)} + w_{5*}\text{Response Time(wj)}$$

Equation (4) rating qualitative parameters of composition

$$\begin{cases} wsi \;\; i = 1{:}\,n \;\; \prod_{i=1}^{n} wsi \quad Availability \\ wsi \;\; i = 1{:}\,n \;\; \prod_{i=1}^{n} wsi \quad Relability \\ wsi \;\; i = 1{:}\,n \;\; \prod_{i=1}^{n} wsi \quad Replasebility \end{cases}$$

n= Number of Service

$$\begin{cases} wsi \quad i = 1{:}\,n \quad \sum_{i=1}^{n} wsi \quad Cost \\ wsi \;\; i = 1{:}\,n \quad \sum_{i=1}^{n} wsi \quad Respanse\ Time \end{cases}$$

E. Substitution phase

This phase involves the following operations.

- reprogramming
- Updating the substitution table

Substitution operation is performed in reprogramming step. If the service is faced with failure due to SLA violations, or if the user requests a service substitution, service substitution operation is performed. In [1], by any failure or service violation, reprogramming will be considered and if it is reasonable, reprogramming phase for substitution begins. Therefore, in case of the need for substitution, algorithm for each service for each violated service will be performed, which increases the substitution time.

First, three questions in relation to the reprogramming arises: when do we need to perform reprogramming? Where to begin reprogramming? How to perform reprogramming? To answer the first question, events which require reprogramming have been studied. Lack of service availability, breakdown in proper service response within a time period as well change of qualitative QoS before the implementation and due to the election of paths in substitution or changes in the number loops, the real qualitative values of the program are different from the estimated values.

In case of failure or services unavailability, the algorithm must be re-run to find service substitution, and seek to have service optimization. There are two reasons for this optimization. First, it can find better qualitative values. Second, it can produce a more acceptable substitution program [1].

In the proposed method to reduce the need of re-programming, the algorithm investigates the substitutable service before and after the implementation of the program. In this case, if the obtained service be replaced in the previous implementation, there would be no need for additional re-programming and extra process time. On the other hand, in case of a limited response time, if the time limit be diverted by the re-optimization, then the algorithm will use the best substitute in the previous implementation. Normally, when there is substitute, in most cases, substitution is suitable.

In the proposed model, according to the dynamically of service composition method, with any changes in qualitative parameters the best possible composition will be calculated during the composition. Therefore, in the first phase of re-programming, with the very changes in the qualitative parameter which is considered as SLA violations here, or a user's requests and failure, the program automatically runs the algorithm in the proposed model described in proposed composition model. The results can include various services and service plans along with the fitness number. Each are stored in a table. These results are dynamically updated in a specified interval. And if there is a need for reprograming and if the numeric is less that fitness value of Service plan in the table substitution will be done automatically with minimal time without running genetic algorithm and in regard to regulations of SLA.

The substitution process in the proposed model is done indirectly. Unlike [1] which is focused only on services with equal input and output and uses direct substitution, in this method, service composition in which its inputs are a subset of inputs and outputs of user's requests will be created. As a result, the possibility of substitution is available, leading to compatibility maintenance and enhancing the range of substitute services. In addition, in the event of service failures due to compatibility maintenance and if necessary, service plan can be replaced; while in [1] there is only the possibility of replacing the service.

The proposed method enjoys a higher speed compared to [1]. And composition time is significantly reduced. To achieve this all processes are running in the background. As mentioned in composition method phase, multi-threading techniques is used in the proposed algorithm. To implement service composition and construction of the composite service, a thread and to update the substitution table another parallel thread is considered.

IV. The Results of Evaluation

Services data sets are produced by the program and to test the method, different numbers of services are used. Since the in proposed method, dynamic and indirect service composition and service substitution is done. Memory consumption is higher than static service composition method and direct substitution; which is considered normal. In addition, due to the dynamic composition, time taken to find the best composition in the algorithm increases, but this increase is seen only in the first survey. The proposed method will be evaluated from four aspects of composition time, re-programming time, memory usage and rate of failure.

A. service composition time

In this study, an increase in the number of services in the proposed algorithm due to the dynamic composition, composition time increase in providing a suitable initial composition; while in a direct composition algorithm, composition time decreases due to static composition. The results are shown in Table 6 and Fig. 2.

TABLE VI. THE RESULTS OF COMPOSITION TIME EVALUATION

Count	Static composition Time1	Dynamic composition Time2
5000	0. 76	0.007
50000	0.77	0.45
100000	0.78	1.54
150000	0.78	3.68
200000	0.79	6.96
250000	0.76	10.43
300000	0.8	15.49

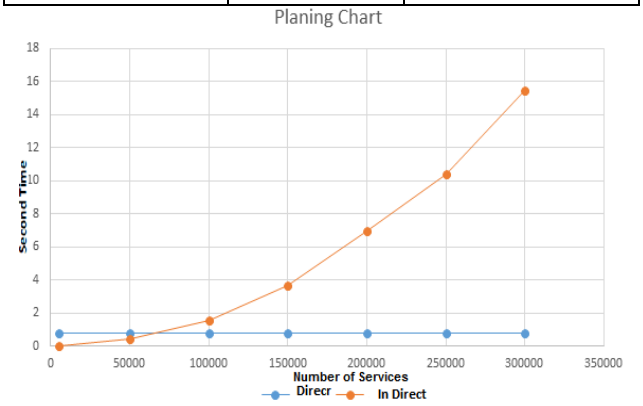

Fig. 2. The results of composition time evaluation

The results of composition time evaluation

B. memory usage

In the proposed model, the number of services is increased due to the dynamic approach. As a result, the amount of memory usage is increased compared to the direct substitution. The results of the evaluation in accordance with the memory usage are shown in Table 7 and Fig 3

TABLE VII. THE RESULTS OF MEMORY USAGE EVALUATION

Count	Static composition Memory1	Dynamic composition Memory 2
5000	34.6	34.4
50000	50.1	56.6
100000	66.8	79.2
150000	68.3	89.7
200000	78.2	108.6
250000	89.6	123.2
300000	101.1	141.9

Fig. 3. The results of Memory Usage evaluation

C. evaluation of reprogramming time (service substitution)

In the proposed model, due to use of updating technique, implementing the best solution in parallel in the background, time of reprogramming will be close to zero, and can be ignored. Also, due to using a linked list in the calculations instead of arrays, the time is reduced compared to direct substitution algorithm. The results are shown in Table 8 Fig. 4.

TABLE VIII. EVALUATION OF REPROGRAMMING TIME (SERVICE SUBSTITUTION)

Count	Direct Replacement Time1:	Indirect Replacement: Time2
5000	0.005	0.00021
50000	0.0051	0.00026
100000	0.0052	0.00042
150000	0.0061	0.00046
200000	0.0063	0.00051
250000	0.0072	0.00053
300000	0.0074	0.00059

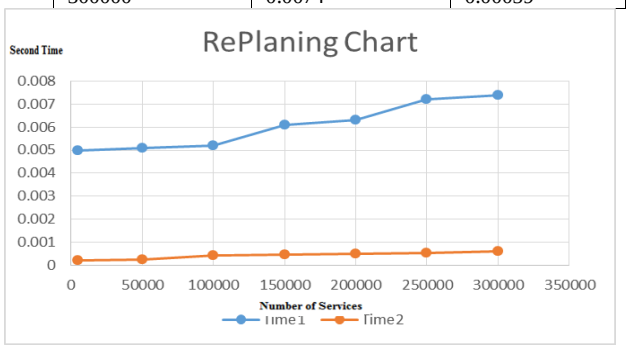

Fig. 4. Evaluation of reprogramming time (service substitution)

D. evaluation of failure rate

In the proposed method, failure rate is close to zero due to using updating technique in the table of the best solution. Since in re-programming, we always have the best answer in the table and there is no need for calculation. Although direct algorithm in [1], since computing must be performed in the reprogramming, to find best composition for the user, it is likely that there is no possibility of reprogramming and the algorithm may fail. In testing cases, the proposed algorithm had no failure. However, in cases where users demand high quality service and low cost and low response time, the

proposed algorithm is likely to fail. In case the user requests according to the table (8), the proposed algorithm is likely to fail after one or two times of reprogramming.

TABLE IX. EVALUATION OF FAILURE RATE

availability	0.9
reliability	0.9
Response time	0.3
cost	0.3
substitution	0.11

However, due to the fact that in simulation, a service which becomes unavailable, will be outdated from the algorithm, but this rarely happens in real space. In the direct method [1], despite adopting substitution, the failure rate is high since the user's requested service was not along with so high qualitative weights users.

V. RELATED WORKS IN THE FIELD OF SERVICE SUBSTITUTION

In this section, different methods of Web services substitution are introduced.

Helal AL et al. used the concept of substitution for service composition. The way it works is as follows: by selecting a qualitative model of services and identifying the cumulative function and calculating qualitative model of each service, a set of service examples is considered as a candidate and by measuring the substitutability of each candidate service by using the filtering technique and nearest neighbor algorithm in this technique, candidates set is improved, and then by using genetic algorithm and its fitness function through total weight, services substitution and reprogramming will be done. The advantage is that one can make the best choice for service composition. In this study direct substitution is focused, i.e. the service will be replaced without any adjustments. This method is useless in case of incompatibility between the services. Thus, the defects of this study is lack of focus on indirect substitution and compatibility [1].

Yu et al. have used an approach based on graph theory. In general, the main idea behind this theory is to show a service as a node. Links indicate the relationship between the services. The costs are qualitative characteristics (cost and delay). The advantages of this method are optimum runtime and memory usage, respectively. One of the weakness in this methods is the lack of scalability [9].

Sheng et al. used a method based on backward theory for service composition. The main idea of this approach is that services are selected step by step. To choose a service at each step, the selection algorithm moves one step backward and checks the selected services to ensure the best service is selected. If the selected service is approved, it will be called. The advantages of this method is fault tolerance. Thus by deterioration of a service, another efficient service will be replaced. The disadvantages of this method is increased processing time [10].

Zhang et al. presented a heuristic algorithm based on taboo search for dynamic service substitution. They used the graph of candidate service; and by using simulation, they evaluated the efficacy and performance. Simulation results show that the proposed algorithm is very good in the substitution in large-scale space. The advantages of this method is it ensures service availability and uninterrupted process and its weaknesses is focusing only on substitution algorithm [11].

Wu et al. presented a cluster-based approach for service substitution. Concepts of logical service, real service, and the cluster service and the relationship between these services had been studied. The proposed method consists of two steps: Finding the expired real service dependent to the logical service and choosing a real service from the service cluster for substitution of violated service.

In this method, compatible services are put in a cluster and can be replaced by another. The advantage of this method is increased speed and reliability of service composition. However, if none of the cluster services are available, user's requests remain unanswered, which is as a drawback in the method [12].

Li et al. presented Web Service Composition based on QoS with Chaos Particle Swarm Optimization. In this study, based on desired qualitative parameters, services are selected, then selected services are entered into the algorithm and finally provides the best composition. Increased speed of service selection and service compositions is one of the advantages of this method. The drawback of this method is lack of attention to parallelism and inconsistent data [13].

Alrafai et al. provided an approach for using Skyline service for Web service composition based on QoS; in which integration of Web-based service composition were evaluated dynamically and without defect. The advantages of this method is division as Skyline calculations can be provided in parallel in groups without changing the final result. This is done by using Pad Skyline algorithmic framework for parallel processing of Skyline request in divided groups. The optimization technique within the group and multi-dimensional filtering for each group is performed. In particular, Skyline local points along with the request as the filtered points to help identifying services in the areas of poor quality on any site are sent through Skyline service. Another advantage of this method is reduction in the response time to user requests and increased speed of Web service composition. The method is affordable and effective for specified service composition. The drawback of this method is lack of investigation in limited and a structured environment. [14].

Lu et al. have provided Web Service Composition Based on Integrated Substitution and Adaptation. They showed that substitution and adaptation complementary are and believe that the integration of adaptation and substitution provides the design of highest flexibility and performance over time for substitution and running the service. In this study, web service composition is based on adaptation and using substitution. They studied substitution at two static and dynamic levels and proposed a dynamic substitution approach. The advantages of this research is service composition without passing through the adaptation and by considering the substitution and automatic service composition with increased workflow functionality for the composition, as well as more flexibility in the composition. The drawbacks may be a lack of focus on

timing constraints in the system and describing similarities based on non-functional parameters [15].

Kuang et al. are focused on the challenge of substitution through behavioral analysis of services. They achieved this by Security Operation Center approach and a formal definition of behavior in Web services. In this study, a formal definition for service composition by means of complex behavior π and calculus formulas as well as conception of behavioral substitution of services through simulations by using formulas and tools has been evaluated. Simulations showed that behavioral substitution of services can be improved based on behavioral analysis of services through using formulas and mathematical calculations. The drawbacks of this research is lack of focus on providing the tool for automated substitution and lack of compatibility with different conditions dynamically [16].

VI. CONCLUSION

In the proposed method, incompatibility problems in composition and substitution of services are figured out by considering qualitative parameters, requested inputs and outputs of user and SLA rules by techniques of creating linear list, fitness function table by total weight and update table of the best alternatives with parallel multi-threading approach. The algorithm efficiency in reprogramming and substitution of the service significantly increased. In this method, as a process is active in algorithm's background and substitution is done dynamically and at the same time it needs to run in parallel environments. In this model, due to dynamic composition and creating composite services based on a subset of the input and output as well as an updated table with the best alternatives, the memory usage increases to some extent compared to direct substitution and static composition. Therefore, studying and improving the memory usage may be a future research.

REFERENCES

[1] H. Al-Helal and R. Gamble, ''Introducing Replaceability into Web Service Composition'', IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 7, NO. 2, PP.198-209 ,2014.

[2] V. Andrikopoulos, S. Benbernou, and M. Papazoglou, "On the Evolution of Services", IEEE Transactions on Software Engineering, Vol.33, No. 3, PP. 609-628, 2012.

[3] R. Iordache and F. Moldoveanu," QoS-Aware Web Service Semantic Selection Based on Preferences", 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, vol. 69, pp. 1152–1161 ,2013.

[4] K. Saeedi, L. Zhao, P. R. Falcone Sampaio," Extending BPMN for Supporting CustomerFacing Service Quality Requirements", IEEE International Conference on Web Services (ICWS), PP. 616 - 623 ,2010

[5] DZG.Garcia, MBF.de Toledo,"Achieving autonomic web service integration:a quality of service policy based approach", Journal of International Transactions on Systems science and applications, vol.3, pp. 41-63, 2010.

[6] S. Bosse, M. Splieth,M. Turowski," Multi-Objective Optimization of IT Service Availability and Costs", Magdeburg Research and Competence Cluster for Very Large Business Applications, Faculty of Computer Science, Otto von Guericke University Magdeburg Germany,vol.147,pp.142-155,2016.

[7] A.Eleyan, L.Zhao, "Extending WSDL and UDDI with Quality Service Selection Criteria", In: Proceedings of the 3rd International Symposium on Web Services,pp.1-10, 2010.

[8] M. Alhamad, T. Dillon, E. Chang, "Conceptual SLA Framework for Cloud Computing", 4th IEEE International Conference on Digital Ecosystems and Technologies,pp. 606 – 610,2010.

[9] H.Q.Yu, S.Reiff-Marganiec,"Web Service Composition Methods:A Survey",Information Sciences,Vol280,PP.218-238,2014.

[10] Q.Z.Sheng,X.Qiao,A.V.Vasilakos,"Selection of QoS Support on Artificial Immune Network Classifier for Dynamic Web Service Composition",International Conference on Computational Intelligence and Security,PP.643-646,2014.

[11] C .Zhang ,H .Chen and J .Du , "A Tabu Search Approach for Dynamic Service Substitition in SOA Applications " , Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific ,INSPEC Accession , PP. 284 – 289,2011 .

[12] L .Wu , Y .Zhang and Z .Di, " A Service-cluster Based Approach to Service Substitution of Web Service Composition " , IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), , PP. 564 - 568 ,2012.

[13] W.Li,H.Yanxiang, "Web ServiceComposition based on Qos with Chaos Particle Swarm Optimization", 6th International Conference on wireless Communications Networking and Mobile Computing,PP.1-4,2010.

[14] J. Wu, L. Chen, T. Liang," Selecting Dynamic Skyline Services for QoS-based Service Composition", Applied Mathematics & Information ciences An International Journal.,vol. 8,PP. 2579-2588, 2014.

[15] L.Chen,R.Chow,"Web Service Composition Based On Integrated Substitution and Adaptation", IEEE International Conference on Information Reuse and Integration,pp. 34 – 39,2008.

[16] L.Kuang, Y.Xia, SH.Deng ,J.Wu,"Analyzing Behavioral Substitution Of Web Services Based On π-Calculus", IEEE International Conference on Web Services International College Wales Swansea, pp. 441 – 448,2010