Journal of Computer and Communications, 2016, 4, 24-39
Published Online July 2016 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2016.49004

Scientific Research Publishing

Dynamic Service Discovery, Composition and Reconfiguration in a Model Mapping Business Process to Web Services

Zulqarnain Abdul Jabbar, Asia Samreen

Department of Computer Science, Bahria University, Karachi, Pakistan
Email: z_ul_qarnain@hotmail.com

Received 5 June 2016; accepted 24 July 2016; published 29 July 2016

Abstract

In any organization where SOA has been implemented, all of the web services are registered in UDDI and users' needs are served by using appropriate web services. So in this paper, we will try to discover a service from repository first that can provide the required output to the user. The process becomes difficult when a single service is not able to fulfill a user's need and we need a combination of services to answer complex needs of users. In our paper, we will suggest a simpler approach for dynamic service composition using a graph based methodology. This will be a design time service composition. This approach uses the functional and non-functional parameters of the services to select the most suitable services for composition as per user's need. This approach involves "service classification" on the basis of functional parameters, "service discovery" on the basis of user's need and then "service composition" using the selected services on the basis of non-functional parameters like response time, cost, security and availability. Another challenge in SOA implementation is that, once the composition has performed, some services may become faulty at runtime and may stop the entire process of serving a user's need. So, we will also describe a way of "dynamic service reconfiguration" in our approach that will enable us to identify and replace a faulty service that is violating the SLA or is not accessible anymore. This service reconfiguration is done without redoing or reconfiguring the entire composition. In the end, to simulate the proposed approach, we will represent a prototype application built on php 5.4 using My SQL database at backend.

Keywords

SOA, Web Services Composition, Web Services Classification, Web Services Discovery, Web Services Recomposition, Web Services Reconfiguration

How to cite this paper: Jabbar, Z.A. and Samreen, A. (2016) Dynamic Service Discovery, Compositionand Reconfiguration in a Model Mapping Business Process to Web Services. *Journal of Computer and Communications*, **4**, 24-39.
http://dx.doi.org/10.4236/jcc.2016.49004

1. Introduction

Section 1 gives the introduction of Service Oriented Architecture and the basic terminologies related to SOA. We are also going to discuss the service discovery, composition and reconfiguration along with the use of SLA in SOA in this section. In section 2, we will discuss the related work. In section 3, we will discuss the methodology used for services composition in our research and will show the services dependencies in form of Directed Acyclic Graph. In section 4, we will discuss the methodology used for services reconfiguration in our research. We will also discuss the role of SLA in service composition and service reconfiguration along with the related research work for services composition and reconfiguration. Section 5 presents a simulated system to demonstrate the methodology used in our research. Conclusion and future work will be presented in the next section.

1.1. Service Oriented Architecture

Service Oriented Architecture (SOA) is a services based software integration solution for most of the systems. It is a loosely-coupled architecture designed to meet the business needs of the organization.
Key features of service oriented architecture are to:

- Divide our code in reusable modules and encapsulate design decisions in modules.
- Combine modules with each other following loose coupling for different purposes.
- Easily respond to changes and make maintainability easier.
- Improve productivity, agility, and speed for both business and IT.
- Align IT services to business and deliver faster services.

Reusability, flexibility and maintainability are the main powers of service oriented architecture.
SOA can provide following benefits to business.

- **Efficiency**: Transform business processes from replicated processes into highly leveraged shared services that cost less to maintain.
- **Responsiveness**: Rapid adaptation and delivery of key business services are to meet market demands for increased service levels to customers, employees, and partners.
- **Adaptability**: More effectively rollout changes throughput the business with minimal complexity and effort, saving time and money.

Some web services standards discussed in different research are: WSDL, SOAP, UDDI, ESB, and BPEL.
WSDL: Web service description language is an XML file used to define what service is and how to interact with it.
SOAP: Simple Object Access Protocol provides a framework that manages the interaction between requester and responder through message passing.
UDDI: Universal Description, Discovery and Integration is used to register web services.
ESB: ESB serves as a middleware in SOA. Service provider publishes services on ESB and service users can invoke the services on the ESB according to their business requirements. ESB provides a set of infrastructure capabilities, implemented by middleware technology, which enable the integration of services in SOA.
BPEL: Business Process Execution Language is an XML based standard for defining business processes. It supports processes which exchange information by using web service interfaces.
SLA: Service Level Agreement is an agreement between consumer and service provider that is used to verify the QoS of services.

1.2. Service Classification

We can classify services based on any similarity found among them like similar functional parameters, similar non-functional parameters, similar complexity level, etc. For example, when some or all of the input and output parameters are common between two services, then these services can become a part of the same class on the basis of functional parameters. Another example is when two services can provide the same level of security or can complete process in similar response time or cost of the services is the same, then we can say that these services are in the same class based on non-functional parameter. Services can also be classified on the basis of complexity level, for example, if two or more services have the same complexity class like n (O) then we can say that these services are in the same class.

1.3. Service Dependency

When a service can fulfill any pre conditions of another service then we can say that the second service is dependent on the first service. But this dependency should not be cyclic; it means the first service should not be dependent on the second service in this case. Acyclic dependency among services leads us to the path of required service composition based on the provided input and required output of the user.

1.4. Directed Acyclic Graph

A directed acyclic graph helps us to map service dependencies in the form of graph. Services names can be represented as nodes and input and output parameters can be represented on edges of the graph. The direction of the graph represents which service is dependent on which another service exactly. There is no backward or feedback path in this graph so the graph will always be acyclic. Because the services are already classified on the basis of the similarities found among them, so the services of the same class cannot be dependent on each other thus graph complexity will be minimized and each service has to be represented only once in the form of a node using all possible edges for the input and output dependencies with other services except the services of the same class.

1.5. Service Discovery

Service Discovery means to find the desired service to fulfill the customer's need or to find a service for the appropriate service composition to serve the user's requirement. Service Discovery is performed to answer the following questions:

- Does the requested service exist?
- Is the service accessible or is service instance available to fulfill the request?

1.6. Service Level Agreement

A Service Level Agreement (SLA) is a contract between a service provider and its customers that document what services the provider will furnish. It is used to measure the performance of the service provider and to measure and maintain the QoS parameters like response time, cost, security, availability which are agreed by the provider for that particular service.

1.7. Service Composition

When a single service cannot serve a request for any user independently then we use a service composition. A service composition is a collection of existing services that is used to automate a particular task or business process. We cannot get benefits of SOA if we cannot compose service conveniently and appropriately. We can save the mostly used compositions for later use and to further reduce the response time for the end user.

1.8. Service Reconfiguration

Service Reconfiguration is the process of identifying a faulty service from a composite service and then replacing that service with another but similar type of service without reconfiguring the entire process. When "identification and replacement of a faulty service" is more feasible then "re-composition of the services", then we go with reconfiguration approach.

2. Related Work

Hajar Elmaghraoui *et al.* proposed a framework in [1] for the dynamic composition of web services. This framework is based on the model of semantic relationship between the web services and this relationship is presented using a Directed Graph approach. But they also reported some issues regarding graph complexity and size when there are a large number of services because they are not using service clustering.

Yang Bo *et al.* discussed the design time and deploy time service binding issues in [2] and proposed a loosely coupled and modular service oriented model that is realized by ESB technology. A real service partner is matched and bound at runtime in this model. Major drawback of this approach is runtime service binding is time

consuming. So they need some effective policies to save some time in time sensitive business processes.

Dmytro Pukhkaiev *et al.* discussed the dynamic composition of services in [3] and suggested a novel SLA-aware dynamic web services composition approach that takes required QoS parameters into consideration. A QoS based evaluation is performed to compare approaches of web services composition. This evaluations is performed without SLA first and then performed using SLA for different approaches. A new SLA-aware web service composition approach is presented and compared with older approaches based on QoS parameters. This new approach decreases service development and reconfiguration time. A tool can be implemented to test and verify web service development and reconfiguration in a real world scenario. Quantitative results can be measured to prove the efficiency of proposed approach.

Services in dynamic service composition may be from different service providers. A faulty service in this composition may violate end-to-end QoS of a service process. Kwei-Jay Lin *et al.* proposed an effective approach for service replacement of faulty service in [4] that can also replace some neighboring services to maintain end-to-end QoS constraints. A graph based approach is used to construct the service process model and an iterative search algorithm is used to construct the reconfiguration region for every faulty service based on the flexibility and ability of a service to deliver the original QoS in the region. This model helps to reconfigure fewer services rather than repairing a complete faulty service process.

Pavankumar Guluru *et al.* in [5] presented web service composition as a search problem in AND/OR service dependency graph. This is a multi input, single output web service composition. Two algorithms are proposed to find the web service composition to fulfill the service request. In first algorithm composition is performed by generating all the paths from input to output nodes. These paths then merged. Second algorithm focuses on the cost parameters and finds the minimal composition for the given request. Algorithms are evaluated experimentally and results are shown.

Saurabh Shrivastava and Ashish Sharma suggested an approach in [6] for faulty service reconfiguration. Faulty services are identified and replaced on the basis of QoS constraints. Using this approach, modification of whole composition is not necessary and the overheads of computations and service distractions in service delivery are reduced. They also discussed other composition strategies and analyzed the Feasibility of proposed methodology by initializing the Directed Acyclic Graph. But this approach is effective when there is a single fault is considered at a time. There is a need of an algorithm to handle multiple faults at an instant.

Philipp Leitner *et al.* developed a framework PREvent in [7] for optimized adaption of service composition based on service violation. This framework predicts the violation of SLA at runtime using the techniques of machine learning and heuristic optimization. But there is a need to extend this model to prevent violations of SLAs which are defined over a period of time instead of each composition instance individually.

Hui LIU *et al.* discussed in [8] that in multiple available compositions of different web services, there exists an ambiguity between different forms of SLA and getting an integrated service quality of several services is a complex task. They proposed an SLA based service composition model with Semantic Support to facilitate the composition of SLA and meet the customers' needs effectively. To provide a semantic support, a SLA management ontology model is built and then the framework composes the SLAs of different services. In last, it automatically searches the integrated SLAs to get the compositions to fulfill the customer's request. A case study is presented to show the effectiveness of the model to search the SLA satisfied services. But there is a need to further elaborate this ontology model in future up to the full life cycle management of SLA including SLA monitoring, assessment, termination and decomposition.

Karthikeyan and Kumar discussed in [9] that the major challenges in the web services is the Quality of Service. So, they introduce a framework of Quality based dynamic composition that monitors SLA as well as QoS parameters for composition of web services. An algorithm is also presented for monitoring and evaluation of proposed parameters. Experimental results are also discussed to prove how QoS parameters are monitored for composed web services. But they didn't focus the Corrective actions after a violation is detected.

Vallidevi Krishnamurty *et al.* discussed in [10] that in a web service composition, continuous monitoring of the response time of services is needed and there may be a need to reconfigure services without affecting the base code when any service probably expect to violate the contract. So, they proposed an AO4BPEL technique that can change the workflow of BPEL at runtime and bypass the faulty services. They created a SOA-based application where the throughput and the response time of the services were monitored using Apache Jmeter. The response time of an atomic as well as composite service are calculated. The monitored value is compared with the values specified in the SLA contract. If any of the monitored service is expected to violate the contract, then

that service is replaced with another service that has the same functionality using AO4BPEL. Thus, the BPEL workflow is modified at runtime. The identification of aspect service is done statically but it should be dynamic.

3. Service Composition Methodology

3.1. Service Composition Life Cycle

A web services based system is configured by using a simple life cycle of service composition discussed in [1]. Phases of the service composition life cycle are illustrated in **Figure 1**.

Service Request Phase defines the properties and goals of the desired service. These properties and goals are the inputs for the Service Discovery and Service Composition phases. Service Discovery Phase finds the service by analyzing the service request to fulfill user's requirements. If the desired service is not found then the service are automatically composed in Service Composition Phase. Service Execution Phase translates the service composition into an executable representation and deploys it to the service composition execution engine.

3.2. Service Composition at Runtime vs. Pro-Runtime (Design or Deploy Time)

Some of the problems discussed in [2] about the service selection pro-runtime are;

- Services had been selected at design time but after that selection, services offering better results have been deployed.
- Services had been selected at design time but have removed now and not accessible at runtime.
- Error occurs in service execution that interrupts the process and there is no recovery process now.

Service composition at runtime can solve these problems but it is a time consuming process that usually increase the throughput and response time to complete the user request.

We preferred the design time service composition in our research. We are using a Directed Graph Approach to compose services based on the dependencies exist among them and possible compositions will be saved in a Directed Graph Repository. The entire process is completed at design time and to overcome the shortcomings of the service binding at design time, we will use the approach to automatically update the Directed Graph Repository whenever a new service is published or an existing service is changed or removed.

Our approach is, find a service from Service Repository that fulfills the user requirement. If there is no any service that can fulfill all of the parts of the user requirement then find a composite service from Composite Service Repository where previously created composite service plans are stored and if not succeed then find a composite service from Directed Graph Repository.

3.3. Functional Parameters

A service can be defined by its inputs, pre-conditions, outputs and effects. Set of Inputs for a service describes what the service required to produce the required output? Set of outputs of a service describes what the service produce as an answer to a request? Set of pre-conditions for a service describes what are the expectations for a

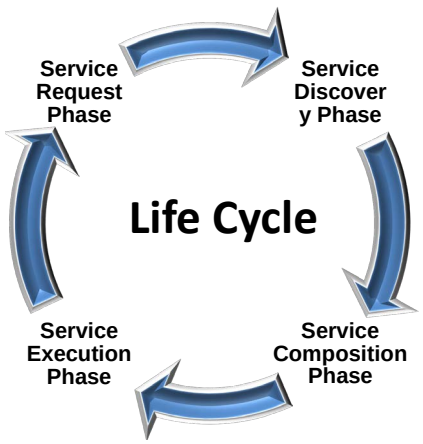


Figure 1. Service composition life cycle.

web service to provide required output? Set of effects of a service describes what is the result of a web service with respect to the inputs and conditions on it? In service composition process, we must know about the values of the functional parameters.

3.4. Directed Dependency Graph for Service Composition

There is more than one method to perform web services composition. For example, static service composition, dynamic service composition, manual service composition and automated service composition.

We have selected a graph based approach for service composition because we can easily map services composition into a graph structure [1]. Some of the advantages of using graph theory are; Nodes of a graph can represent services and edges can represent dependencies among services. Cost of the service usage can be expressed using weights on edges of graph. We can find optimal service composition by finding out the shortest path on graph to serve a user request. Graph structure also makes easier the calculation of response time and can easily point out any deadlock.

3.4.1. Classification of Services

Based on the functional parameters of the services defined in WSDL document of each service, we can classify them in service classes. All of the services in a class should provide the same functionality having same input/output types, but they differ on the basis of non-functional parameters thus providing different QoS value with same functionality [4]. This scenario is illustrated in **Figure 2**.

Class SC1 includes services S1and S2 both providing the same functionality to user request. S3, S4 and S5 are similar type of services thus enclosed in a class SC2. Class SC3 contains services S6 and S7. For the desired web service composition, we may select one service from each class having QoS values better than other services of the same class. Some ways of service composition are discussed and compared in [5] but we prefer graph based approach.

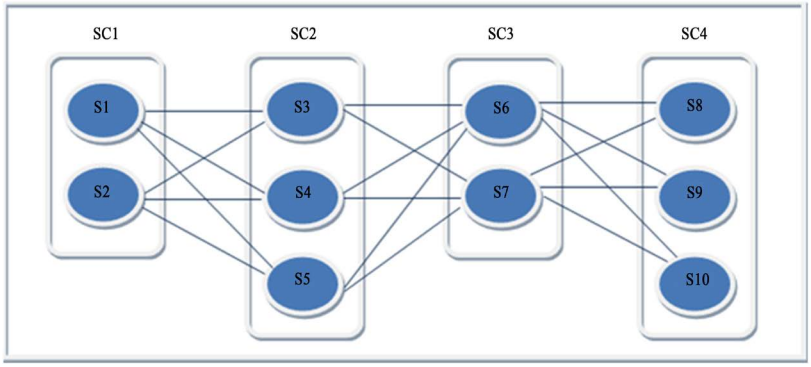


Figure 2. Classification of services.

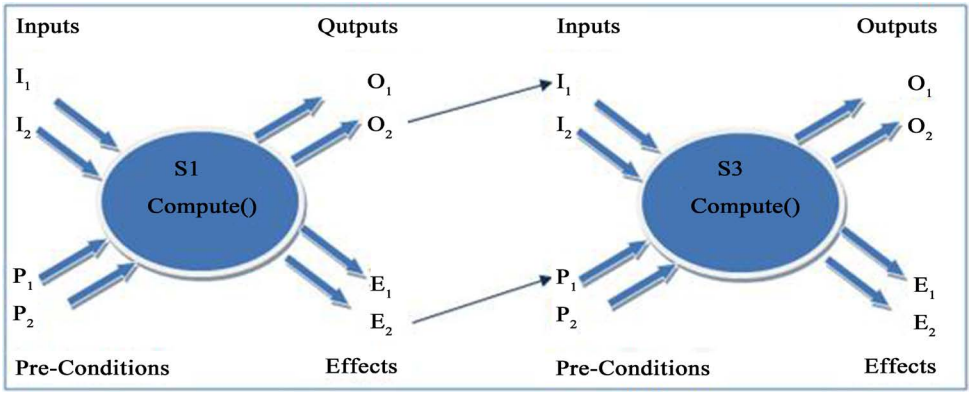


Figure 3. Services dependencies.

3.4.2. Service Dependencies

When an output parameter of a service in a class can fulfill completely or partially the input parameter requirement of a service in another class then there exist a dependency between two services. **Figure 3** shows an example of service dependency.

Service S1 takes I1 and I2 as input and computes the requested functionality if pre-conditions P1 and P2 are satisfied for I1 and I2 respectively. S1 generates the output as O1 and O2 with effects E1 and E2 respectively. Similarly I1 and I2 are inputs for service S3 with pre-conditions P1 and P2 and S3 generates output O1 and O2 with effects E1 and E2. Here one of the outputs of service S1 fulfills the input requirement of service S3 either partially or completely hence there exist a dependency between services S1 and S2 and both services can execute sequentially in any service composition.

3.4.3. Service Dependency Graph

We are constructing a directed acyclic graph to show the dependencies among services of different classes shown in **Figure 3**. Directed Acyclic Graph is shown in **Figure 4**.

Now, If the request is to find a service composition that takes {A, B} as input and gives {J, K} as output then possible service compositions with their aggregated values of QoS parameters are shown in **Table 1** where value of security and availability of each service is presented as L = Low, M = Medium or H = High. Individual values of QoS parameters for each service will be presented in **Table 2** later in this paper.

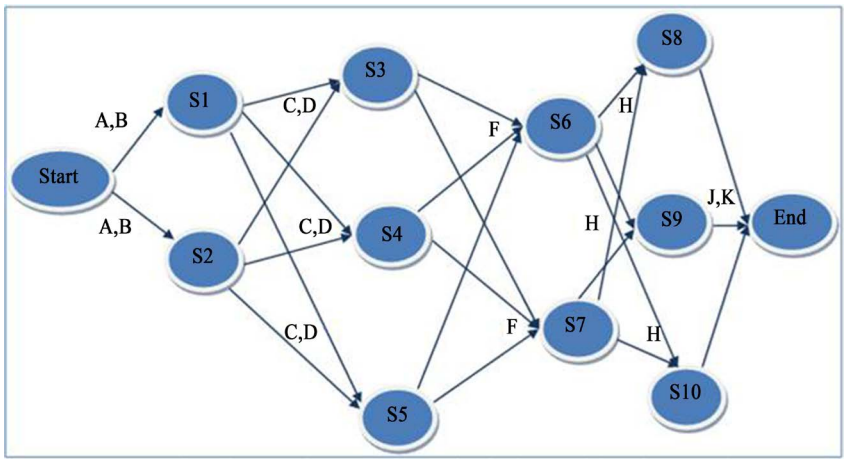


Figure 4. Services dependency graph.

Table 1. Service compositions with aggregated QoS parameters.

S No.	Service Composition	Total Response Time	Total Cost	Security	Availability
1	S1 –> S3 –> S6 –> S8	18	27	H, M, H, L	H, M, H, L
2	S1 –> S3 –> S6 –> S9	17	28	H, M, H, L	H, M, H, L
3	S1 –> S3 –> S6 –> S10	15	30	H, M, H, M	H, M, H, M
4	S1 –> S3 –> S7 –> S8	22	22	H, M, M, L	H, M, M, L
5	S1 –> S3 –> S7 –> S9	21	23	H, M, M, L	H, M, M, L
6	S1 –> S3 –> S7 –> S10	19	25	H, M, M, M	H, M, M, M
7	S1 –> S4 –> S6 –> S8	20	27	H, M, H, L	H, M, H, L
8	S1 –> S4 –> S6 –> S9	19	28	H, M, H, L	H, M, H, L
9	S1 –> S4 –> S6 –> S10	17	30	H, M, H, M	H, M, H, M
10	S1 –> S4 –> S7 –> S8	24	22	H, M, M, L	H, M, M, L

Table 2. QoS parameters.

Service	Input	Output	Response Time	Cost	Security	Availability
1	A, B	C, D	2	10	High	High
2	A, B	C, D	3	5	Med	Med
3	C, D	F	3	5	Med	Med
4	C, D	F	5	5	Med	Med
5	C, D	F	4	5	Med	Med
6	F	H, I	6	10	High	High
7	F	H	10	5	Med	Med
8	H	J, K	7	2	Low	Low
9	H	J, K	6	3	Low	Low
10	H	J, K	4	5	Med	Med

Based on this repository, one can select a service composition according to the required QoS values. When security and availability are the most critical requirements then the composition with services having low security should not be selected. When cost is the major issue and a user can compromise with security and availability then the composition having low cost should be selected. When a user requirement is time critical then a composition having low response time can be selected along with the acceptable values of cost, availability, and security.

3.5. QoS Based Analysis

We not only need accurate results from a service composition but also need the desirable QoS to maintain [6]. In most cases, high priority QoS parameters are Response Time, Cost, Security and Availability. So from the entire list of non-functional parameters, we have selected these four parameters:

- Response Time of a service in milliseconds.
- Cost of a service on a scale of 1 to 10.
- Security of a service on a scale of Low, Medium, High.
- Availability of a service on a scale of Low, Medium, High.

We will select services for service composition on the basis of these four parameters.

Table 2 shows the values of all four parameters for services S1 to S10 which are classified in classes SC1, SC2, SC3 and SC4 in **Figure 3**. Classification of the services was based on the functional parameters inputs, pre-conditions, outputs, effects. For simplicity we are considering only input and output parameters here.

3.5.1. Service Composition Based on Functional Parameters

Based on the functional and non functional parameters, service composition can be defined as:

$$SC = (I, O, F, NF)$$

where SC represents service composition, I is the set of input values, O is the set of output values, F represents the required functionality and NF is for the non-functional parameters.

3.5.2. Service Composition Based on Non-Functional Parameters

Based on the non functional parameters, service composition can be defined as:

$$NF = (P, RI, RB, AC, AV, C, S, CA, AR)$$

where P is the performance, RI is the reliability, RB is the Robustness, AC is the Accessibility, AV is the Availability, C is the Cost, S is the Scalability, CA is the Capacity, and AR is the Accuracy.

4. Services Reconfiguration Methodology

4.1. Role of SLA in Service Composition and Service Reconfiguration

Service level agreement SLA is an agreement between consumer and service provider that is used to verify the QoS of services. In web services composition process, services can be selected on the basis of the QoS parameters defined in Service Level Agreement by service provider while publishing a service. Composed services can also be monitored on the basis of SLA to detect any violation of agreed parameters by a service [3]. **Figure 5** shows the role of SLA in service composition.

SLA represents the Service Level Objectives (SLOs) *i.e.* QoS that the service needs to fulfill and the monitoring is performed on the basis of agreed upon QoS in SLA. Philipp Leitner *et al.* presented a PREvent framework that predicts the SLA violation at runtime and suggests a adaptation action to improve the overall performance in a composition instance [7].

Hui LIU *et al.* presented a framework for the SLA based service composition where they suggest that service provider should also register the service quality information along with the functional details of a service in repository [8]. They also translate the SLAs of different services in a general format to remove ambiguity among SLAs of different services. This will make the monitoring thus detection of SLA violation easier.

Karthikeyan and Kumar presented a framework for the quality based dynamic composition [9]. They used SLA violation predictor and QoS Predictor in their framework to predict the violation against the customer.

Dmytro Pukhkaiev *et al.* proposed a SLA aware approach for web service composition [3]. This approach allows QoS parameters to consider. They suggested a generalized way to store QoS parameters to service providers.

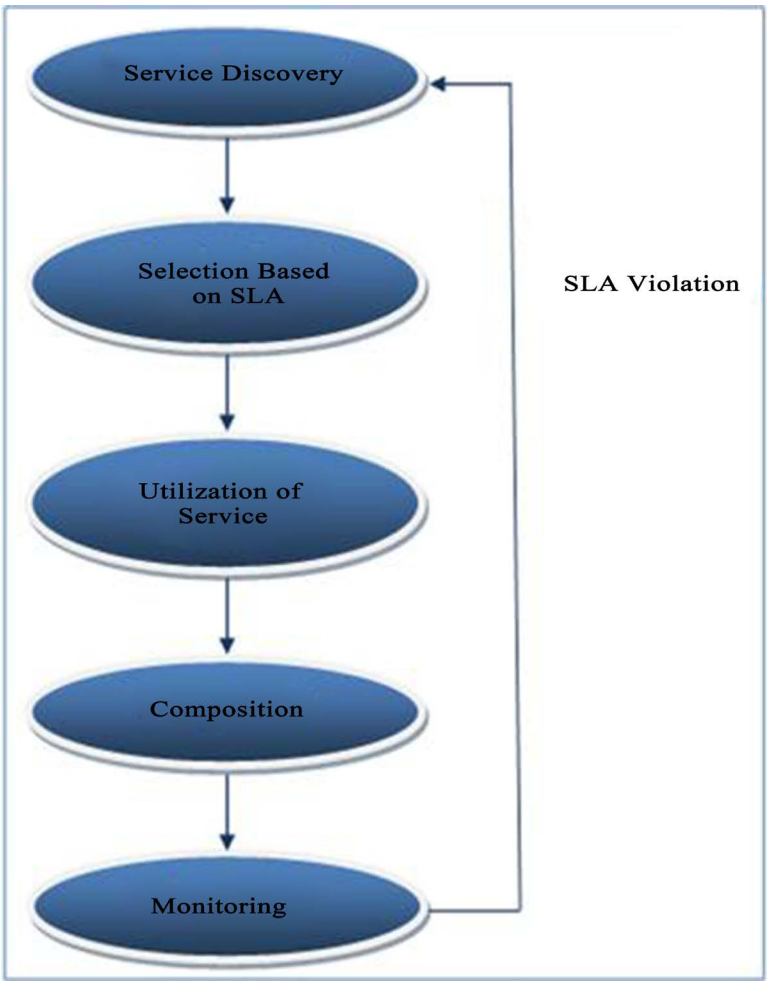


Figure 5. Role of SLA in service composition.

4.1. Finding and Replacing a Faulty Service

At runtime, services may violate pre-defined QoS levels due to many factors like host overload, network congestion etc. In this situation, we need to repair a service process to continue the business process function effectively [4].

Replacing the entire service process will not be efficient and will be time consuming and costly. Reconfiguration is the optimal service selection process that allows the system to work continuously and there is no need of the migration of system and services [6]. So, we need to identify the faulty service from the selected composite service and then replace it with the better service of the same class.

Saurabh *et al.* discussed two approaches of service reconfiguration: i) Finding an optional path of services from its predecessor services to the completion of service process; ii) Finding an alternate service path from the start to the end of the service process [6].

In our approach, we have first classified all of the services on the basis of functional dependencies and have created the directed dependency graph to know all of the possible compositions of the available service at design time. When any composition executed and a service violates the agreed QoS values then we suggest the alternative composition by replacing the faulty service with a service having same input and output dependencies. We can get familiar about the violation of agreed QoS values from the side of any service by using a service monitoring engine or the response getting executing a service composition. One of the service monitoring engines based on monitoring of SLA is discussed by Vallidevi *et al.* in [10].

Discussion of the service monitoring engine is not included in our scope. We are assume some scenarios like a service is violating the QoS values or service is not responding at all and are suggesting an approach to replace that faulty service with another service having better QoS values. In chapter 6, we will discuss a prototype system used to demonstrate the suggested approach of service composition and reconfiguration.

5. Simulation

We are assuming that we have a list of available services along with the details of functional and non-functional parameters of the services. We have created a web based system using Yii framework of php. MySQL database is in use at backend. We can insert the information of services in our system to find out the dependencies among services and to compose service as per user needs.

The web based application is not publicly accessible so a login screen will appear first to allow only authenticated users to access the system. **Figure 6** shows the login screen.

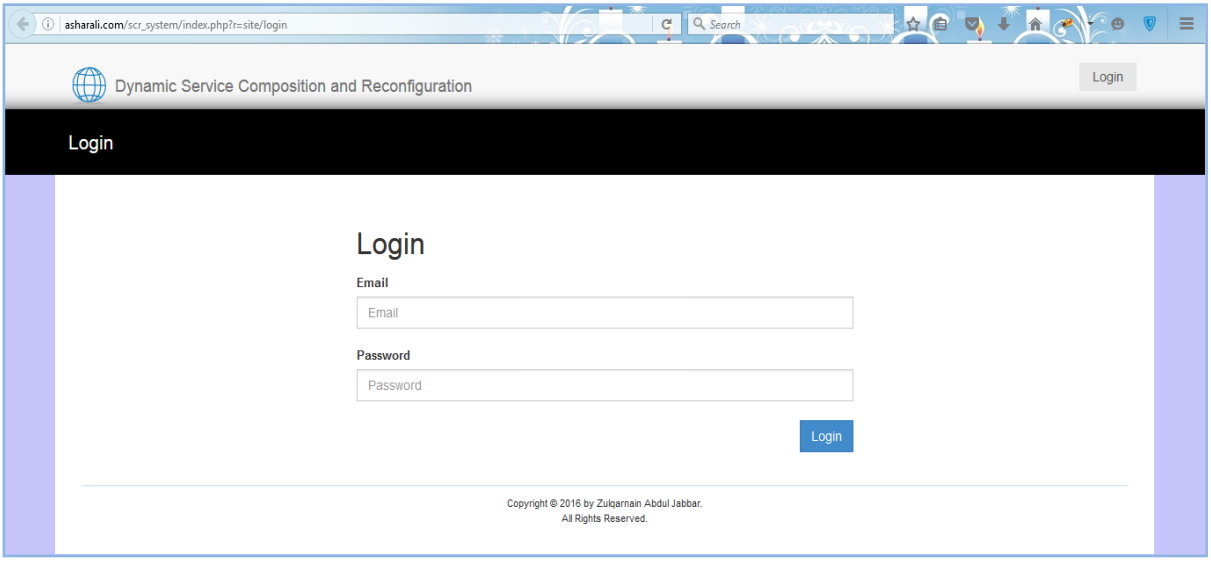


Figure 6. User authentication[1].

[1]This simulator is available online. URL: http://asharali.com/scr_system/index.php. Email: visitor@hotmail.com. PWD: 9A@1c$qrf.

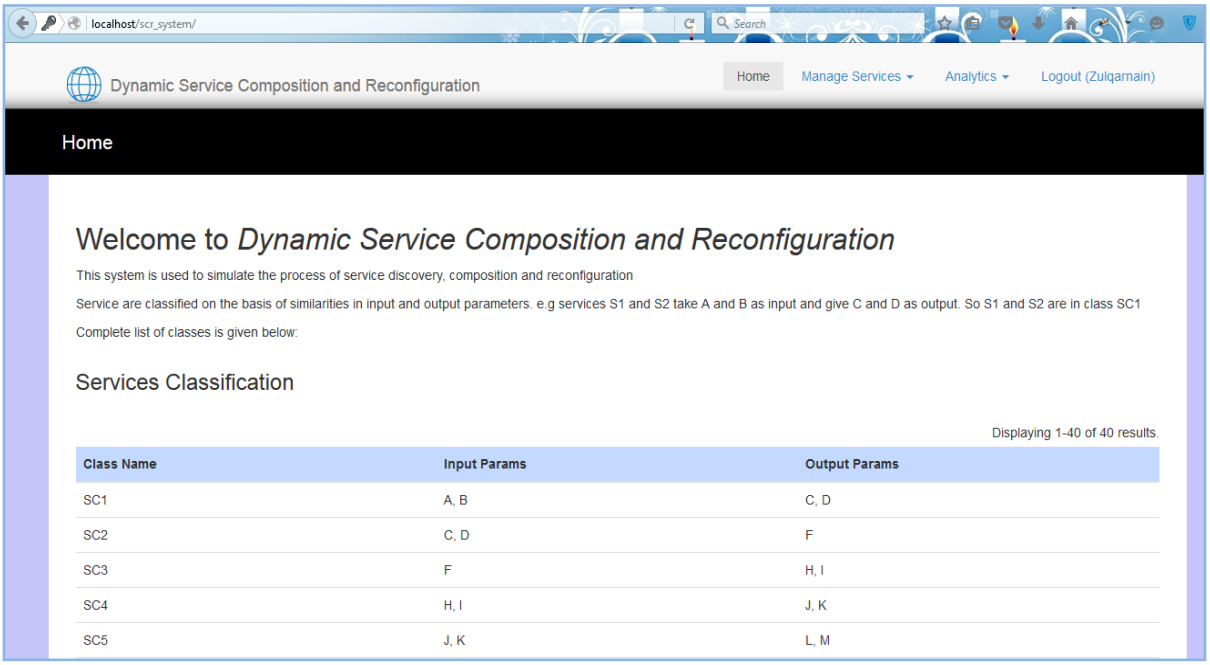


Figure 7. Home screen.

After a successful login, user can view the home screen to know the details about how to use the system. **Figure 7** shows the home screen. User can identify the functional and non-functional parameters of the services as defined in SLA of each service and can enter each service in the system to make it a part of repository. **Figure 8** shows the service info screen.

Service discovery screen lists all of the services which exist in repository along with the information of services about input, output, response time, cost, availability and security parameters. User can find a service by providing some parameter values as filter of the list. **Figure 9** shows the service discovery screen.

Service composition screen finds and list all of the atomic or composite services based on provided values of the input and output required by the user. All of the aggregated values of the QoS parameters are also listed along with the services map on this screen. **Figure 10** shows a service composition screen. We can select any available composition on this screen and click on "Run" button to check the effect of executing selected composition.

Service reconfiguration screen shows the effect of executing any service composition. **Figure 11(a)** shows the condition when a selected service composition executed successfully. **Figure 11(b)** shows the condition when the selected composition executed and any of the service is not responding or is not accessible. This screen also mentions which service is not responding and also suggests alternative compositions to fulfill the user request by replacing the faulty service with another service of the same class having similar type of input and output parameters. **Figure 11(c)** shows the condition when the selected composition executed and any of the service that is violating the agreed values of QoS parameters. This screen also mentions which service is violating the rules and also suggests alternative compositions to fulfill the user request by replacing the faulty service with another service of the same class having similar type of input and output parameters.

Graph in **Figure 12** is showing that how the number of dependencies is growing exponentially with number of services.

Graph in **Figure 13** is showing that the time to get dependencies grows exponentially as well with the number of services.

This graphical analysis clearly depicts that the methodology we are using will become complex with the increased number of services and we have to find a way to optimize the process. One possible way of this optimization can be "sort the list of compositions based on the number of times a composition is used and then show top 10 compositions to user".

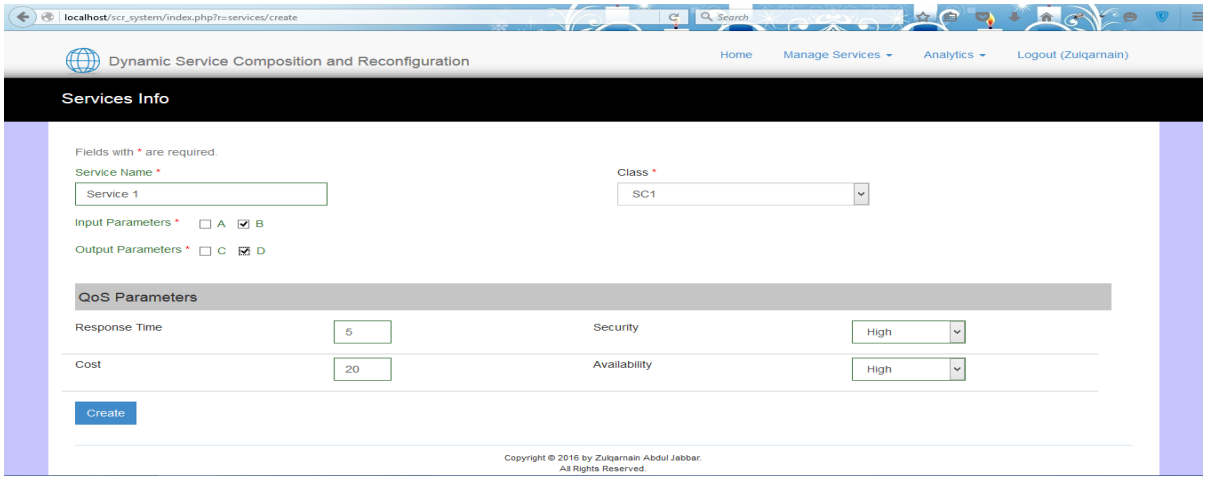


Figure 8. Service info screen.

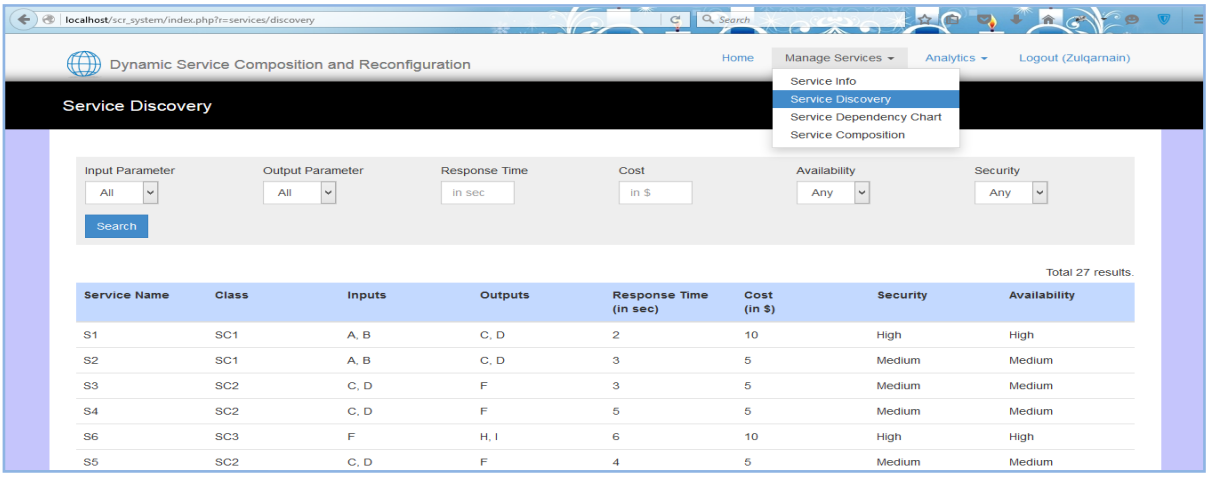


Figure 9. Service discovery screen.

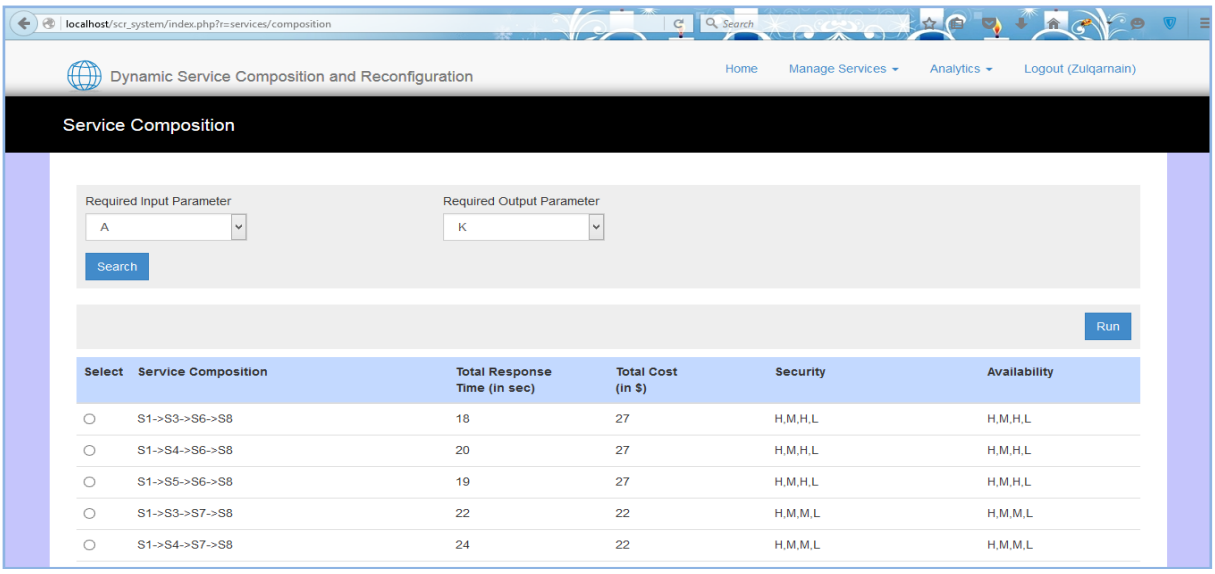
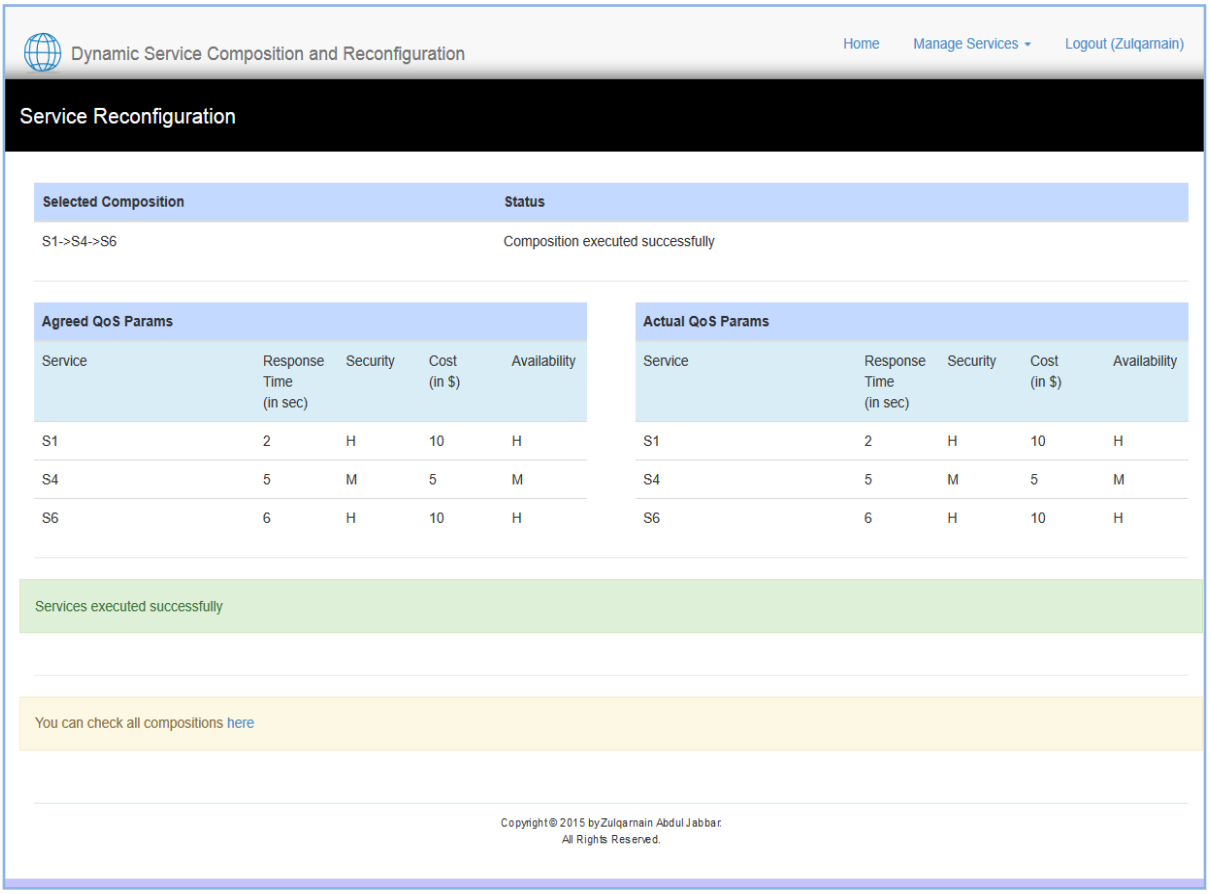
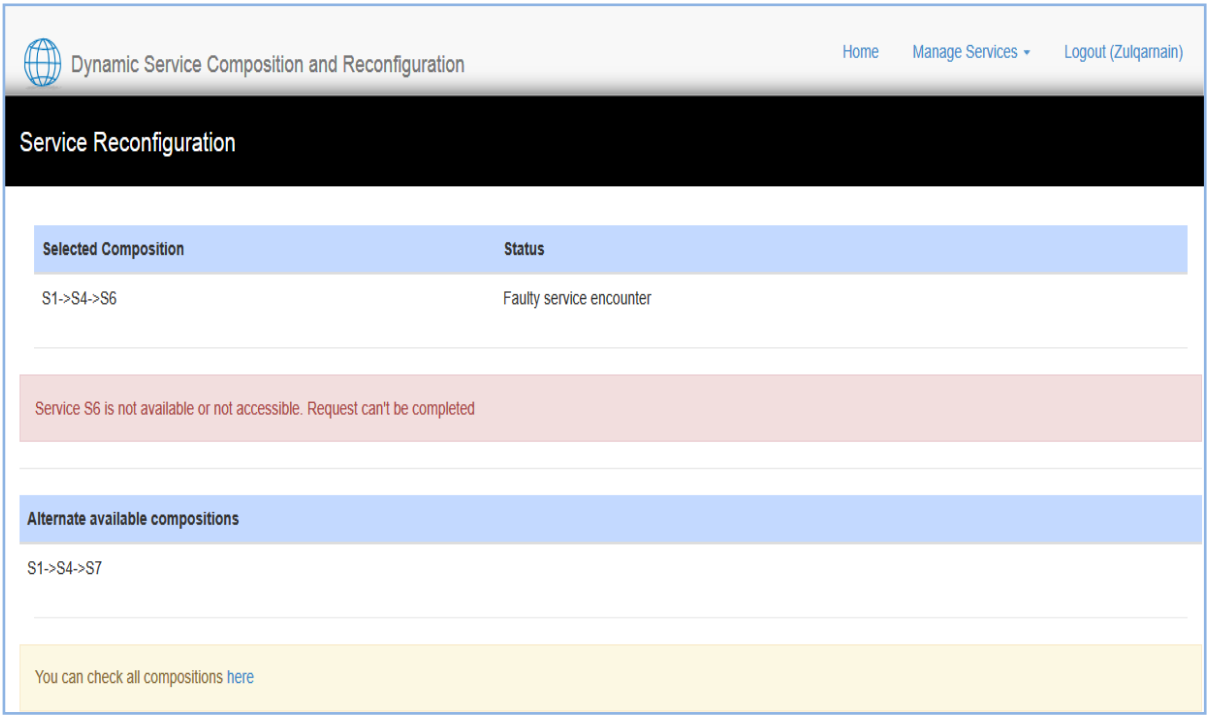


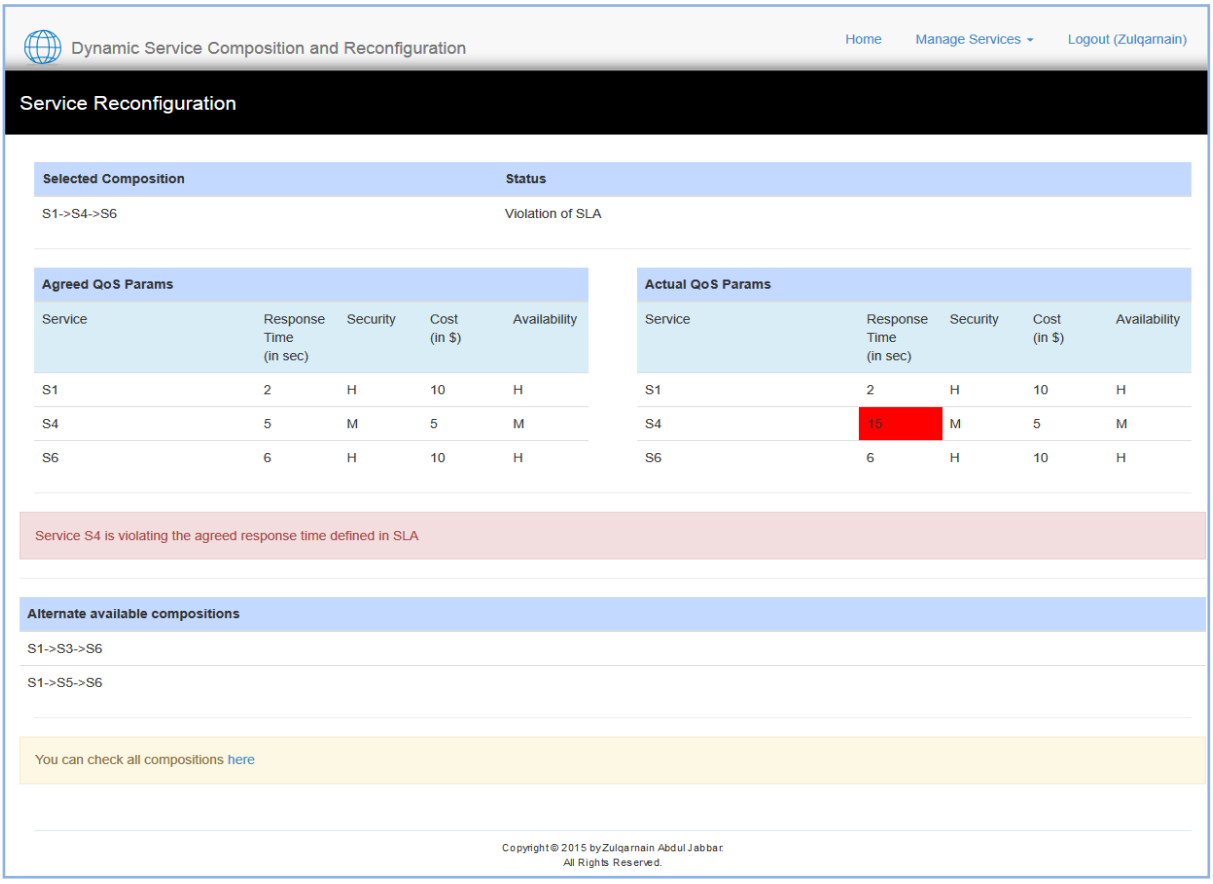
Figure 10. Service composition screen.


Dynamic Service Composition and Reconfiguration
Home
Manage Services
Logout (Zulqarnain)
Service Reconfiguration
Selected Composition
Status
S1->S4->S6
Composition executed successfully
Agreed QoS Params
Service
Response Time (in sec)
Security
Cost (in $)
Availability
S1 2 H 10 H
S4 5 M 5 M
S6 6 H 10 H
Actual QoS Params
Service
Response Time (in sec)
Security
Cost (in $)
Availability
S1 2 H 10 H
S4 5 M 5 M
S6 6 H 10 H
Services executed successfully
You can check all compositions here
Copyright © 2015 by Zulqarnain Abdul Jabbar.
All Rights Reserved.

(a)


Dynamic Service Composition and Reconfiguration
Home
Manage Services
Logout (Zulqarnain)
Service Reconfiguration
Selected Composition
Status
S1->S4->S6
Faulty service encounter
Service S6 is not available or not accessible. Request can't be completed
Alternate available compositions
S1->S4->S7
You can check all compositions here

(b)



(c)

Figure 11. (a) Service composition executed successfully. (b) Service not available. (c) SLA violation detected.

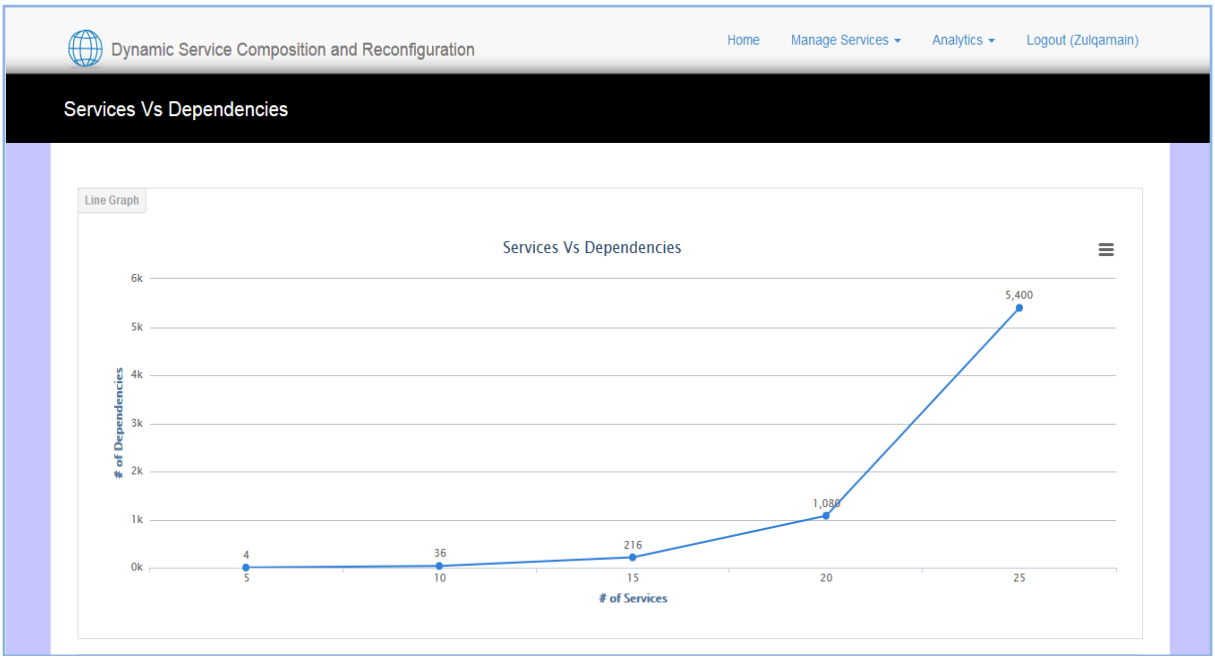


Figure 12. Services vs. dependencies.

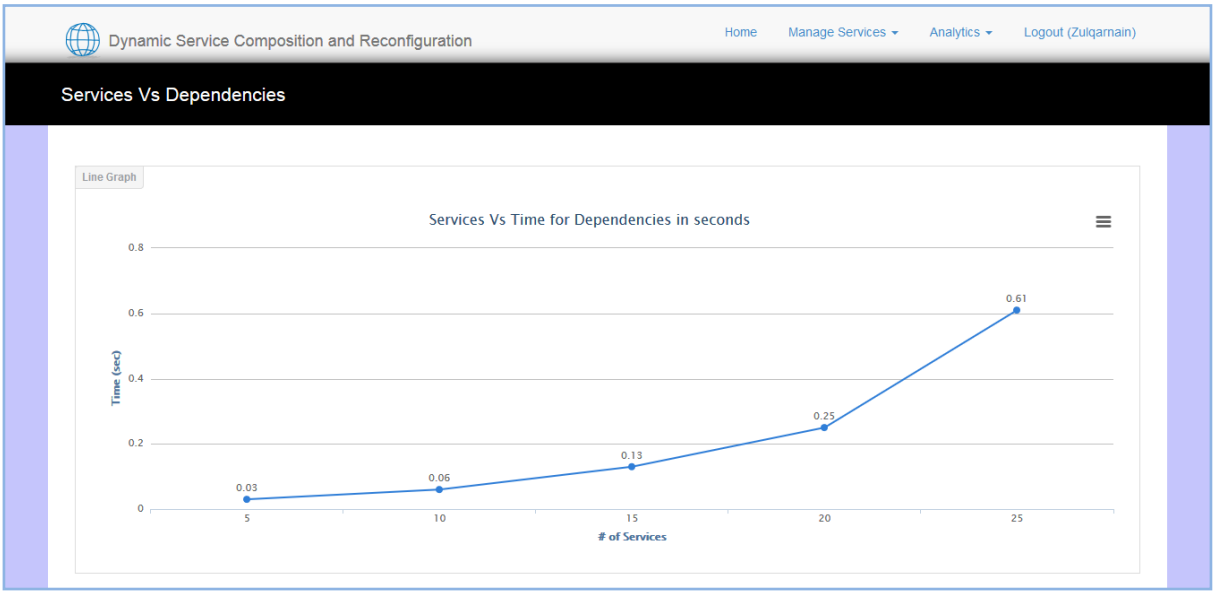


Figure 13. Services vs. time for dependencies in seconds.

6. Conclusions

In this paper, we proposed a graph based approach for dynamic service discovery, service composition and service reconfiguration. The first step is to classify the services based on input/output parameters of the services. The second step is to identify the service dependencies based on their input and output requirements. The third step is to identify the services. We called this step as service discovery phase. Service discovery also includes the description of the input/output parameters and QoS parameters of the identified services. We then list all of the possible combinations of the services based on the user's needs of input/output parameters as well as the QoS parameters like response time, security, cost and availability. A most suitable composition can then be selected from this list on the basis of required QoS parameters. One may prefer minimum response time and low cost and others may prefer high security and availability. A service that is a part of the selected composition may become faulty or may violate the SLA terms. To solve this problem, we have suggested a way in our approach to identify and replace the faulty service without reconfiguring the entire composition. We have also developed a prototype application to simulate the suggested approach.

In the future, we will work on the aggregated SLA of the composite application to make the composition and reconfiguration process more efficient.

Acknowledgements

I would like to pay special thanks to Allah (SWT) who gave me the opportunity to complete this paper. At last, I am also thankful to my family, friends, fellows, and colleagues who supported me in their best.

References

[1] Elmaghraoui, H., Benhlima, L. and Chiadmi, D. (2014) DynaComp: A Framework for Dynamic Composition of Semantic Web Services. 2014 *International Conference on Multimedia Computing and Systems* (*ICMCS*), Marrakech, 14-16 April 2014, 612-616. http://dx.doi.org/10.1109/ICMCS.2014.6911248

[2] Bo, Y., Hu, G.Y. and Jiang, J.S.. (2011) A Scalable and Dynamic Service Oriented Workflow Model. 2011 *International Conference on Intelligent Computation Technology and Automation* (*ICICTA*), Vol. 1, Shenzhen, 28-29 March 2011, 572-575. http://dx.doi.org/10.1109/ICICTA.2011.154

[3] Pukhkaiev, D., Kot, T., Globa, L. and Schill, A. (2013) A Novel SLA-Aware Approach for Web Service Composition. 2013 *IEEE EUROCON*, Zagreb, 1-4 July 2013, 327-334. http://dx.doi.org/10.1109/EUROCON.2013.6625004

[4] Lin, K.-J., Zhang, J., Zhai, Y. and Xu, B. (2010) The Design and Implementation of Service Process Reconfiguration with End-to-End QoS Constraints in SOA. *Service Oriented Computing and Applications*, **4**, 157-168.

http://dx.doi.org/10.1007/s11761-010-0063-6

[5] Guluru, P. and Niyogi, R. (2014) New Approaches for Service Composition Based on Graph Models. 2014 7*th International Conference on Contemporary Computing* (*IC*3), Noida, 7-9 August 2014, 507-512. http://dx.doi.org/10.1109/ic3.2014.6897225

[6] Shrivastava, S. and Sharma, A. (2013) An Approach for QoS Based Fault Reconfiguration in Service Oriented Architecture. 2013 *International Conference on Information Systems and Computer Networks* (*ISCON*), Mathura, 9-10 March 2013, 180-184. http://dx.doi.org/10.1109/ICISCON.2013.6524199

[7] Leitner, P., Dustda, S., Wetzstein, B. and Leymann, F. (2012) Cost-Based Prevention of Violations of Service Level Agreements in Composed Services Using Self-Adaptation. 2012 *Workshop on European Software Services and Systems Research—Results and Challenges* (*S-Cube*), Zurich, 5 June 2012, 34-35. http://dx.doi.org/10.1109/S-Cube.2012.6225506

[8] Liu, H., Bu, F.L. and Cai, H.M. (2012) SLA-Based Service Composition Model with Semantic Support. 2012 *IEEE Asia-Pacific Services Computing Conference* (*APSCC*), Guilin, 6-8 December 2012, 374-379. http://dx.doi.org/10.1109/APSCC.2012.54

[9] Karthikeyan, J. and Kumar, M.S. (2014) Monitoring QoS Parameters of Composed Web Services. 2014 *International Conference on Information Communication and Embedded Systems* (*ICICES*), Chennai, 27-28 February 2014, 1-7. http://dx.doi.org/10.1109/icices.2014.7033756

[10] Krishnamurty, V., Natarajan, R. and Babu, C. (2013) Monitoring and Reconfiguring the Services in Service Oriented System Using AOBPEL. 2013 *International Conference on Recent Trends in Information Technology* (*ICRTIT*), Chennai, 25-27 July 2013, 423-428. http://dx.doi.org/10.1109/icrtit.2013.6844241