A Self Analysing and Reliable SOA Model

Prachet Bhuyan[#], Asima Das[#], Durga Prasad Mohapatra[*]

[#]*School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India*

[*]*Department of Computer Science & Engineering, NIT, Rourkela, Odisha, India*

***Abstract*—Service Oriented Architecture (SOA) provides a new way of application development by using existing services. The required services are collected and loosely composed to meet the user's specification, where the architecture of SOA is dynamic that is, it can change dynamically at run time to meet the new requirements. This paper proposes an architecture which has the capability to analyze the developed services by comparing the user's requirements. Finding a method for testing the accuracy of a service is a challenge always. The architecture provides a unified way of self analysis which evaluates the perfectness of developed service, before its delivery, and to find out its accuracy. This calculated perfectness is stored in the service profile of that service. The database is maintained for each service which helps for searching a required service to meet user's specification and for composition of a new service. When this self analysis of the developed service does not satisfy to meet the user's requirement because of not satisfying any conditions, it generates the fault. Fault is handled to provide the required correctness for composition of the service by detecting the services responsible for those faults. The architecture provides both reliability and analysis capability by handling fault and by searching a service based on accuracy which provides reliable delivery of service.**

***Keywords*— SOA, Self Analysing, Reliable, Composition, Service**

I. INTRODUCTION

SOA supersedes over the traditional architecture due to its dynamic nature of service composition. Service is an encapsulated function that is different from other services in the environment [1]. It accepts requests and returns one or more responses by using any defined standard interfaces. A service can be requested by a user or by another service. In the current global systems in distributed network the services are distributed over various platforms and it needs to search the services. Composing loosely and dynamically services in an efficient way to get the goal mapping to user's requirements is a difficult task.

For composition, all required services may not be available in all the supporting platforms. The searching of services may be affected due to network failure or errors, which may also lead to unavailability of service. Sometimes searching a service in distributed platform becomes inefficient due to the complexity in the network. Without waiting for the availability of the non-available services, these services can be created to make the system efficient.

The rest of the paper is structured as follows: Section II reviews SOA and web technology to adopt SOA. Section III discusses the proposed architecture, its layers and their working. Section IV discusses the self analysis of developed services at the business execution and analysis layer. Section V deals with the conclusion of this paper.

II. SOA AND ITS ADOPTION BY WEB TECHNOLOGY

Architecture is the fundamental organization of a system embodied in its components, their relationship to each other and the environment and the principles governing its design and implementation [4].

SOA is an architecture where the main component is service, when a consumer requires a service on demand it is made available by a service provider. The service is searched in a service registry by means of a broker and the user which is the service consumer is connected to the service provider which finally delivers the service.

Software developers are grouped into three parties in SOA by means of their responsibilities. The main entities of SOA are service consumer (Application builders / service requesters), service brokers and service providers.

The service consumer is a client service, which requires a service. The service consumer (web service client) locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke the required services. The service consumer executes the service by sending a formatted request as specified according to the contract. They can access multiple services if need arises. Thus a target application is built through service discovery and composing the required services instead of designing and coding the required services.

The service provider delivers of the service. The service provider when creates a service, it publishes the service descriptions, like interface, access information, the service contracts in the registry to access the service by service consumer, other services etc. It can be a mainframe system, a component, or some other type of software system that executes the service request.

A service registry is a directory that contains the details of the available services found in the network. It accepts and stores contracts from service providers and responds to the service consumers accordingly.

Service brokers publish the available services to the public from service registry. Service brokers help to achieve this by making the contracts public. Depending on the business model, brokers can attempt to maximize look-up requests, number of listings or accuracy of the listings.

Web Technology in order to adopt SOA includes WSDL (web service description language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description Discovery and Integration); WSDL is an XML based interface definition language that defines service and service description. SOAP is a simple messaging protocol designed to exchange message on the Web. This is a XML based platform independent and application language independent protocol. UDDI is a specification which defines a way to register the Information for publishing and discovering the information about web services. UDDI depends on HTTP to transfer the data and XML to describe the information. The interaction and relationship among the Service Broker, Service Provider and the Service Requestor is shown in the web services architectural model in Figure 1.

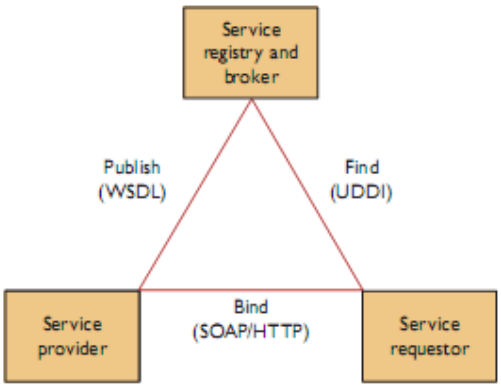


Figure 1. Web services architectural model [2]

III. PROPOSED ARCHITECTURE

In this section we propose a service oriented architecture. The proposed architecture of SOA emphasizes on, analysing the user's requirement, there by composing the services accordingly, execution analysis, for finding composition fault and finally how to get a perfect service in a reliable way. It includes six layers and four mechanisms dealing with those layers. Below we discuss all these six layers and four mechanisms in brief.

A. Layers of the proposed architecture

The proposed architecture is shown in the Figure 2. It consists of six horizontal layers and four vertical layers. Vertical layers handles different type of mechanisms included with six horizontal layers. Each layer is described in brief. Six horizontal layers are Data and Components Layer, Specification and Analysis Layer, Service Layer, Process Layer, Business Process Execution and Analysis Layer, and Presentation Layer.

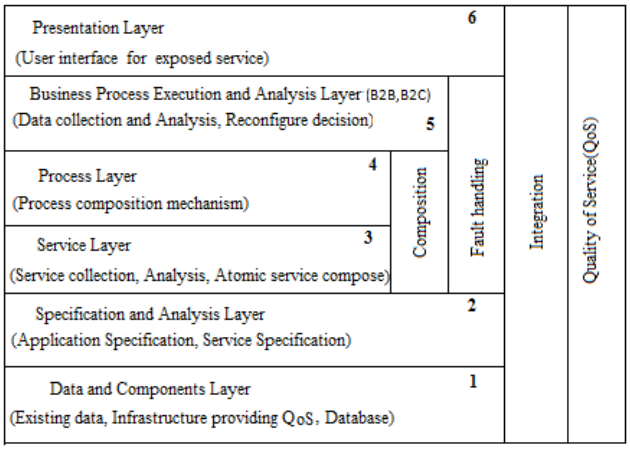


Figure 2. A Proposed Architecture of self Analysing and Reliable SOA

1) **Data and Components Layer**: Consists of older legacy systems, older object-oriented system implementations, and intelligence applications, Data repositories, old reusable data, components which are responsible for realizing functionality and maintaining the Quos (Quality of service) of the services, enterprise assets, and application servers.
2) **Specification and Analysis Layer**: This layer deals with the customer/user specification and analysis of the requirements. Based on application specification of user's, the usercondition_action database, service specification and workflow specification are built. So, this layer deals with two specifications that is service specification, workflow specification.
3) **Service Layer**: The services that will be exposed reside in this layer. The services can be discovered first and then invoked, or choreographed into a composite service. This layer contains mechanism to publish interfaces of components as service descriptions and exposes for use. If a service is not found, building of atomic services at same platform takes place.
4) **Process Layer**: This layer deals with composition of business services based on the mechanisms like choreography or orchestration. This layer integrates services together which are exposed in the service layer according to the workflow specification to build a single composite application.
5) **Business Process Execution and Analysis Layer**: This layer deals with deployment and execution analysis. The functionality of this layer is to analyse the developed service to find out the errors. This layer analyses the perfectness of the composed service, to find out the differences between the newly built service and the service as required by the user. When the self analysis process, satisfied, the service is delivered for deployment else finds the fault for further processing. The self analysis process is described in more detail in Section IV.
6) **Presentation Layer**: At the presentation layer, the developed services are exposed for delivery to the customer by means of various interfaces. It is not a mandatory layer in SOA. Now-a-days, this layer is needed, because there is an increasing convergence of standards, such as web services for Remote Portlets Version 2.0 and other technologies that seek to leverage web services at the application interface or presentation level [3].

Four vertical layers that handle the mechanisms are as follows.

a) **Composition**: Composition is included in service layer and process layer. When an unavailability of a service is notified after searching service registries, the service layer initiates for atomic composition. The composition at process layer deals with mechanism for composition of the required services to develop the required composite service. It works based on a knowledge base. The knowledge base contains the service profiles and the condition_action database, for services which are already available. After composition, each service must have this condition action database. Each condition_action database of a service has the number of conditions that a service must meet to perform correctly. For each condition, there may or may not be one respective action to satisfy that condition. For calculation of perfectness, a value column is also there. The value 2 is taken for primarycondition and value 1 for secondary condition.
When a new service is composed by taking a number of services, a new condition_action database is created for

the composed service by adding all the condition_action database of the services taken part in composition for developing the new service.

The format of the condition action database for each service is given in Table 1. Here the value for a service is always equal to the primarycondition value of usercondition_action database.

TABLE 1
CONDITION_ACTION DATABASE FORMAT

primaryc ondition	Secondary condition	primary_ value	seconda ry_value	service

b) **Fault Handling**: While development and execution analysis continues, fault tolerance is achieved by handling composition fault. The composition fault for services are found out by analysing their executions at business process execution and analysis layer and these faults are corrected by the process recycle, where the recomposition takes place to handle the composition error. Error for any wrong data input at the time of execution can also be handled with rollback to the previous session by storing the previous session variables.

c) **Integration**: This layer enables the integration of services through the introduction of a reliable set of capabilities, such as intelligent routing, protocol mediation, and other transformation mechanisms, often described as the Enterprise Service Bus. Web Services Description Language (WSDL) specifies a binding, which implies a location where the service is provided. On the other hand, an ESB provides a location independent mechanism for integration [3].

d) **QoS**: Quality of Service layer provides the capabilities required to monitor, manage, and maintain QoS such as, performance, and availability. This is a background process through sense-and-respond mechanisms and tools that monitor the health of SOA applications and other relevant protocols and standards that implement quality of service for a SOA [3].

B. Working of the Architecture

In this section, the explanation of the working of the proposed architecture is given and it is shown in Figure 3. The processes of the proposed architecture get activated when a request from consumers arises. An application specification is built containing the user requirements as conditions. Conditions can be divided as primary condition and secondary condition. Primary conditions are those needs of the user, without which the service composed will never meet the goal of user. Secondary conditions are those requirements of the user without which also the service composed will be able to meet the goal of user. Based on the requirements a usercondition_action database is created having those conditions that should satisfy to meet the user's requirement and called as usercondition_action database.

The process controller is the controller of the whole process. User provides the application specification having primary and secondary conditions. Based on the application specification service specification and workflow specification is built. A usercondition_action database is created which stores the primary and secondary conditions specified in application specification and the required actions where needed. A required action may or may not be there to satisfy the condition against the same row. The format of the usercondition _action database similar to Table 1 is as follows.

TABLE 2
USERCONDITION_ACTION DATABASE FORMAT

Primary condition	Secondary condition	Primary value	Secondary value	Service

Service specification is handed over to the service provider and the services are collected accordingly from the service registry. Then the collected services are handed over to the process controller. The process controller then creates the workflow specification for the services and handover the collected services and the workflow specification to the dynamic composer.

The application monitoring is done to keep track of the processing, for data collection, while next processes continue. So, till the end of processing, if any error occurs, that can be obtained due to continuous process of data collection and handled.

Dynamic composition of the service now takes place, based on the workflow specification. If any atomic service is not found, it is developed and tested by analysing the execution.

At business process execution and analysis layer the developed service is tested to analyse the execution. Self analysis is done at this layer to find out the perfectness of the built service in comparison with the user's requirement. Dynamic reconfiguration takes place in this case where a new requirement arises or any service replacement has to take place due to some composition error which gives rise to recycle process.

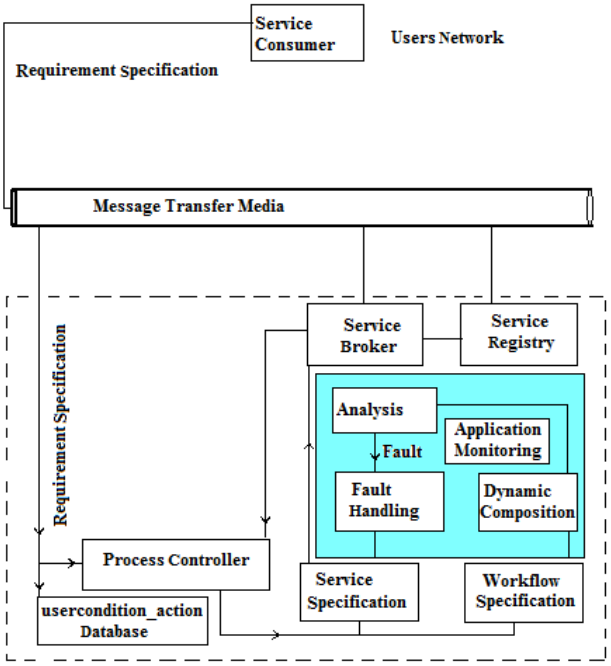


Figure 3. Workflow of the Proposed SOA Architecture

IV. SELF ANALYSIS OF DEVELOPED SERVICE

Self analysis of the developed service is carried out for finding out the perfectness or correctness of the service. The requirement is mapped based on the user's requirement. It provides a measure for accuracy. It also detects composition faults while analysing the services and handles them in order to provide reliability. New condition_action database is analysed by comparing with the usercondition_action database which was previously created, based on a user's requirement, at the time the requirement from the user arises. The usercondition_action database has columns such as primarycondition, secondarycondition, action, primary value and secondary value. The new condition_action database also has primaryconditions, which are the addition of all primaryconditions of the condition_action databases of those services that had taken part in composition of the new service. Similarly secondaryconditions , which are the addition of all secondaryconditions of the condition_action databases of all those services, taken part in composition of the new service. The service column indicates which service a particular condition belongs. The primary conditions must satisfy to meet the user's requirement, with comparison to the secondary conditions. We have taken the value for primary condition as 2, and then value for secondary conditions should be less than that primary condition value. So, we have taken the value for the secondary condition as 1.

As an example: if user requires the account details(account name, type, balance), customer details (name, socialsecurityno, telephone) along with the address details (street number, street name, postbox, city, state, country) of a customer as a service then here the secondary conditions will be zipcode and telephone. By excluding zip code accessing the particular customer with address having name, street number, street name, postbox, city, state, zip, and country is another need. So, here zip code can be taken as a secondary condition in the usercondition_action database. Similarly, we can exclude telephone number making telephone as secondary condition.

Let the service specification to build the service specify use of three services account, customer and address having their condition_action databases respectively as below. The service profile contains the accuracy of that service, where accuracy is found out by calculating the perfectness. Here perfectness is best for service account, better for service customer and the service address is a better service.

TABLE 3

CONDITION_ACTION DATABASE FOR SERVICE ACCOUNT

primary condition	**secondary condition**	**Primary _value**	**Secondary _value**	**service**
accountno	null	2	null	account
accounttype	null	2	null	account
amount	null	2	null	account

TABLE 4

CONDITION_ACTION DATABASE FOR SERVICE CUSTOMER

primary condition	**secondary condition**	**primary _value**	**secondary _value**	**service**
name		2		customer
socialsecurityno		2		customer
	telephone		1	customer

TABLE 5

CONDITION_ACTION DATABASE FOR SERVICE ADDRESS

primary condition	**secondary condition**	**primary _value**	**secondary _value**	**service**
streetno		2		address
streetname		2		address
postbox		2		address
city		2		address
state		2		address
	zip		1	address
country		2		address

The usercondition_action database for the service is created when a requirement arises. The usercondition_action database for the above example is shown in Table 6.

TABLE 6

USERCONDITION_ACTION DATABASE TO DEVELOP THE SERVICE

primary condition	secondary condition	primary _value	secondary _value	service
account no		2		account
account type		2		account
amount		2		account
name		2		customer
social securityno		2		customer
	telephone		1	customer
streetno		2		address
streetname		2		address
postbox		2		address
city		2		address
state		2		address
	zip		1	address
country		2		address

If any of the primarycondition of the composed service is not found in database of developed service then, the developed service is not acceptable, because it will not satisfy the primary requirement of the user. So, fault is detected and this fault can be corrected by recycle process. For the purpose of error handling the conditions responsible for errors should be kept for further processing while execution analysis continues.

Following is the Algorithm for Self Analysis

The algorithm for the analysis process of the developed service is as follows:

1. Begin analysis.
2. initialize integer variables i, j and counter to '0', primary_value=2, secondary_value=1, i=no_of_records in usercondition_action database, j= no_of_records in condition_action database
3. for m= 0 to i
4. pc = primarycondition of usercondition_action database.
5. for n = 0 to j
6. if pc = primarycondition column of new condition_action database, then exit for
7. else display message 'fault in service', so service is not acceptable because of error in condition 'pc'. Add 1 to counter. Store that respective related row

of usercondition_action database in the database called error_condition.

8. next
9. next
10. find out the number of primaryconditions and number of secondaryconditions from usercondition_action database; and number of primarycondition and number of secondaryconditions of new condition_action database respectively and store in variables no_of_ primary, no_of_secondary, no_of_ primary1, no_of_secondary1 respectively.
11. minperfectvalue = no_of_ primary * primary_value.
12. maxperfectvalue = (no_of_ primary*primary_value)+(no_of_secondary*second ary_value).
13. newperfectvalue = (no_of_ Primary1*primary_value) + (no_of_secondary1* secondary_value).
14. if (newperfectvalue = perfectvalue) then perfectness= "best". The developed service is a 'Best' service to accept.
15. if (newperfectvalue > minperfectvalue) then perfectness= "better". It is a 'Better' service and can be acceptable.
16. if (newperfectvalue = minperfectvalue) then perfectness= "good". The developed service is 'Good' and the service can be acceptable.
17. if counter >0 then recycle.
18. end.

If usercondition_action database has 100 primary conditions and 50 secondary conditions, then the need is to satisfy at least those 100 primary conditions excluding secondary conditions. Then the condition column of the new condition_action database of the composed service should have at least those 100 primary conditions to satisfy the requirement. The condition column may have primary conditions along with all those secondary conditions or may have few secondary conditions with all those primary conditions.

By taking value 2 for primaryconditions and 1 for secondaryconditions, if we get all the services to compose the required service with both primary as well as secondary conditions as primary condition in the available service then the maximum perfect value will be 250 which specifies the service as best. The minimum perfect value will be 200, where the need is to satisfy the primary conditions only which specify the service as good, where we can exclude all the secondary conditions. Any value within the maxperfectvalue and minperfectvalue indicates, all primary conditions along with some secondary conditions have been satisfied, which makes the service a better one.

Suppose, the service in the above example, is composed to analyse the perfectness. The condition_action database of the developed service will have the database given in Table 7 as follows by taking the above three available service's condition_action databases and adding it. The perfectness calculated will be better because the condition column have all the primary conditions of usercondition_action database and excluded the secondary condition.

TABLE 7
CONDITION_ACTION DATABASE FOR THE NEW DEVELOPED SERVICE

primary condition	secondary condition	primary _value	secondary _value	service
Accountno		2		account
accounttype		2		account
Amount		2		account
Name		2		customer
social securityno		2		customer
	telephone	1		customer
Streetno		2		address
Streetname		2		address
Postbox		2		address
City		2		address
State		2		address
Country		2		address

So, according to the algorithm Here, no_of_ primary=11, no_of_secondary=2, no_of_ primary1=11, no_of_secondary1 =1.

minperfectvalue= (no_of_ primary * primary_value) =22.

maxperfectvalue = (no_of_ primary* primary_value + no_of_secondary * secondary_value) =24.

Newperfectvalue=no_of_ primary1* primary_value + no_of_secondary1 * secondary_value=23.

then, (newperfectvalue > minperfectvalue), so the perfectness=better.

The recycle process is to handle faults detected at the time of execution analysis of the developed service .It deals mainly with the composition faults. It includes searching the conditions responsible for failure. It is possible by searching the error_condition database and verifying that particular condition. It leads to recomposition of the service. The recomposition takes place in the similar way as the composition takes place. So it is an error handling process which helps the recovery from wrong service composition fault. This is a forward error recovery process. The needed atomic service composed by taking the conditions present in error database and its condition_action database is created. This condition_action database is then added to the previously created new condition_action database to create the correct new condition_action database for the required service and again analysed to find out its perfectness.

When the service analysis successfully ends with an acceptable service then the service profile for that service maintains the calculated perfectness. The calculated value for perfectness is stored in a variable called perfectvalue. Perfectvalue provides the perfectness of the service which is the accuracy of the service. While registering the service, in service registry, accuracy about the service is also mentioned. This provides a reliable way to choose the existing service based on the criteria of accuracy as needed, by finding the data directly from the registry.

Case Study: The following is the case study by taking Bank as the consumer of the proposed SOA.

The main purpose of the bank is to handle customers' accounts to provide good services for its customers. For this purpose of the bank it needs to keep all the details of the account holders / bank customers. So, the job is to keep all the details of the customer's account and the details of

customers along with their address details. So, following are three cases to achieve this purpose of the bank.

Case 1: To have all the details of the customer's account.

Case 2: To have all the detail of the customers, for their unique identification.

Case 3: To have all the detail of the customers by which a contact through letter can be possible whenever needed.

To accomplish the required task, three atomic services account, customer and address are needed for Case 1, Case 2, and Case 3 respectively. So, the requirements of the bank will be fulfilled by a service which needs to be composed by taking these three services. With reference to the layers of the proposed architecture following functions are performed by the layers.

Specification and analysis layer: When the requirement comes first it is analysed in the 'specification and analysis layer' .The requirement specification are in the form of primary and secondary requirements and is called as primarycondition and secondary condition respectively. Primaryconditions are those requirements of the customer without fulfilling which the goal will never achieve. Secondarycoditions are those conditions those requirements of the customer without which also we can achive the goal. Based on the requirement the usercondition _action database is then created which will be similar to Table 6.

Data and component layer: Based on the requirement the usercondition_action database for the required service is searched in the service registry, which contains the list of existing services and their details. In this case following will be the usercondition_action database with the related existing service name filled against the respective row of the conditions. Here the atomic services account, customer and address are already present in the service registry. So the complete usercondition_action database will be similar to Table 7.

If any of the atomic services is not present in registry for example let the service account is not present then the usercondition_action database will be as in Figure 8. In this case the conditions against the row without having the service column are taken for atomic composition. Here the conditions are accountno, type and amount. This composition takes place in service layer and the atomic service account is created by taking all these 3 primary conditions.

TABLE 8
USERCONDITION_ACTION DATABASE WHEN SERVICE ACCOUNT IS NOT PRESENT IN SERVICE REGISTRY

primary condition	secondary condition	primary _value	secondary _value	service
account no		2		
account type		2		
amount		2		
name		2		customer
socialsecurityno		2		customer
	telephone		1	customer
streetno		2		address
streetname		2		address
postbox		2		address
city		2		address
state		2		address
	zip		1	address
country		2		address

Service layer: The atomic composition is performed in service layer by taking conditions of the vacant service column of usercondition_action database. Whenever a service is composed a condition_action database is created. Services account, customer and address are physically present in the service layer. All the existing services are exposed in this layer for the composition. The existing services at the time of composition have their condition_action databases. So, services account, customer and address have their condition_action databases as given in Table 3, Table 4, and Table 5 respectively.

Process layer: In the process layer the composite composition takes place by taking those services condition_action databases, which services are found against the service column of usercondition_action database. To compose the final service needed in order to fulfil the above requirement of the bank, that is condition_action databases of services account, customer and address taken and a new condition_action database is created by adding all the primary conditions and secondary conditions of these condition_action databases. Hence the new condition_action database composed for this case will be as given in Table 7.

Business Process Execution and Analysis Layer: After composition of the new condition_action database the new condition_action database is compared with the user condition_action database and the perfectness of the service is found out by taking the given algorithm for self analysis. The composition faults are also detected whenever a primary condition of usercondition_action database is not found in the new condition_action database and is stored in error database.

Implementation snapshots of self analysis as follows:

CONDITION_ACTION DATABASE

primarycondition	secondarycondition	primary_value	secondary_value	service
accountno		2	0	account
type		2	0	account
amount		2	0	account
customername		2	0	customer
socialsecurityno		2	0	customer
	telephone	0	1	customer
streetno		2	0	address
streetname		2	0	address
city		2	0	address
postbox		2	0	address
	zip	0	1	address
state		2	0	address
country		2	0	address

Figure 4. NewCondition action Database

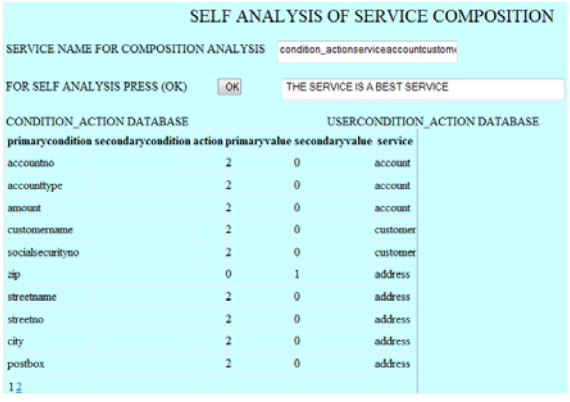
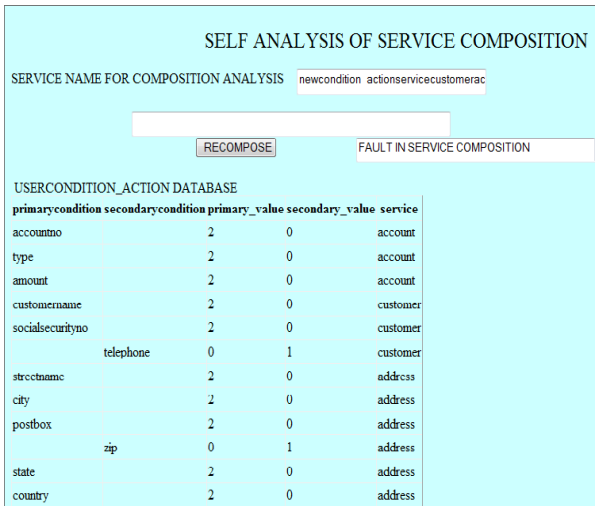


Figure 5. Self-analysis of above newcondition_action database with comparison to usercondition_action database

In Figure 5 the new composed service is a better service having all the primary conditions and secondary conditions are equal, in both newcondition_action database and usercondition_action database.


SELF ANALYSIS OF SERVICE COMPOSITION

SERVICE NAME FOR COMPOSITION ANALYSIS newcondition actionservicecustomerac

RECOMPOSE FAULT IN SERVICE COMPOSITION

USERCONDITION_ACTION DATABASE

primarycondition	secondarycondition	primary_value	secondary_value	service
accountno		2	0	account
type		2	0	account
amount		2	0	account
customername		2	0	customer
socialsecurityno		2	0	customer
	telephone	0	1	customer
streetname		2	0	address
city		2	0	address
postbox		2	0	address
	zip	0	1	address
state		2	0	address
country		2	0	address

Figure 6. Self-analysis of above newcondition_action database with comparison to usercondition_action database

In Figure 6 the new composed service have the fault in composition service having all the primary conditions that are not equal in both newcondition_action database and usercondition_action database as streetno is not present in newcondition_action database.

V. CONCLUSION

This work proposes a self analyzing and reliable SOA. The architecture has the facility to analyze the developed service by comparing with the actual user requirement to find out perfectness of the service, thereby it finds out, how much a service is accurate. This accuracy about the service is stored in service registry by which it provides a reliable way of choosing out the exact required service from the registry of existing services based on the criteria of service perfectness or accuracy. Self analysis performs automatically to find execution analysis. The Architecture has the capability to detect the composition fault at the time of analyzing the perfectness. A composition fault can be detected when unsatisfied with the perfectness and it has the facility to go for recycle process, to eliminate the related fault. This helps for better composition of the service that satisfies the user requirement. Hence on a concluding note it can be addressed that the proposed architecture provides reliability and efficient services to the users.

REFERENCES

[1] Hongqi Li, Zhuang Wu "Research on Distributed Architecture Based on SOA", International Conference on Communication Software and Networks, IEEE, 2009.

[2] M.N.Huhns and M.P.Singh "Service-oriented computing: key concepts and principles" IEEE internet computing, pp. 75-81, Jan-Feb. 2005.

[3] Ali Arsanjani, "Service-Oriented Modeling and Architecture: How to Identify, Specify, and Realize Services for Your SOA", whitepaper from IBM, Nov 2004, available at: http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/.

[4] http://blogs.technet.com/b/michael_platt/archive/2006/03/27. http://blogs.technet.com/b/michael_platt/archive/2006/03/27/423300.aspx