International Journal of Computer Science and Informatics

Volume 4 | Issue 1 Article 9

July 2014

MODEL BASED REGRESSION TESTING APPROACH OF SERVICEORIENTED ARCHITECTURE(SOA) BASED APPLICATION: A CASE STUDY

PRACHET BHUYAN
School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India, pbhuyanfcs@kiit.ac.in

ABHISHEK KUMAR
School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India, abhikumar695@gmail.com

MODEL BASED REGRESSION TESTING APPROACH OF SERVICE-ORIENTED ARCHITECTURE(SOA) BASED APPLICATION: A CASE STUDY

PRACHET BHUYAN[1], ABHISHEK KUMAR[2]

[1,2]School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India
E-mail: pbhuyanfcs@kiit.ac.in, abhikumar695@gmail.com

Abstract- —Service-Oriented Architecture(SOA) is an approach for designing, deploying and managing services that represent reusable business functionality. SOA removes the gap between software and business. Reliability and fault-free implementation are major concern for SOA based applications. Traditional testing are no more beneficial for verifying and validating the quality of services in SOA systems. Regression testing is inevitable that is undertaken every time to provide confidence that modification do not introduce new bugs into previously validated code. In this paper we addressed the UML based regression testing method using UML use case diagram and UML activity diagram to generate a test case in the context of case study named ' Online Shopping System' .

Keywords- *SOA, Regression Testing, Business, UML.*

I. INTRODUCTION

Service-Oriented Architecture is loosely coupled, discoverable , reusable and inter-operable in which each of the service follow well defined standard. SOA promote organization agility and vendor diversity. A very common technology for SOA implementation is web-services, in which service interfaces are described using the WSDL (web-service description language) and XML (extensible markup language). Payload/message is transmitted using SOAP over HTTP. Web-services are different from the web applications. In web application there is a user interface whereas web service allow different machine to interact with each other. Web- service is a type of software that interact with other software to communicate or for transferring the data. Mostly, user interface is not necessary for a web service. It is just like a component of an application. Both web service and web application deliver contents using internet. But web application use it for specific client such as web browser whereas web service do not bother about whether the client is browser or mobile apps. Dynamic and Adaptive nature of SOA make it difficult to test service-centric application [5]. Validating and Verifying the operation of SOA is a complex and challenging problem [6] . SOA is an integration of several heterogeneous components and each component is built using different technology and incompatible interface. A service baseline effort of regression testing makes it easier to determine that defects have not been introduced as a result of evolution. Change in the baseline can serve as a triggers that indicate the need to perform regression testing . Some of the triggers are [3]:

a. Upgrade to service contracts or SLAs.
b. Changes to back-end system or data sources.
c. Changes in service's functional behavior or QoS.
d. Update rule used by testing tools.
e. Retirement of the services.

This Paper is structured as follow. In section 2 overview of SOA testing under which difference between web service testing and web application testing, different testing perspective, element to be tested in SOA and different testing level of SOA is addressed. Strategies for SOA regression testing is addressed in section 3. Section 4 focus on SOA testing challenges. Section 5 focus on problem with existing approaches of regression testing and in section 6 we use UML use case diagram and UML activity diagram to generate test cases in the context of a case study. In section 7 we compare our work with one of the existing case study given by Rajani Kanta Mohant etal [4]. Conclusion and future scope is addressed in Section 8.

II. OVERVIEW OF SOA TESTING

SOA based systems are form of software and so they should be tested. The testing of SOA based application is different from the testing of web applications. Web applications testing is mostly done by the software provider. The development team ensures objective and completeness of web applications.

In web application testing, test location is centralized with multi-phase testing. Web application support offline regression testing. There is static unit testing and integration testing. Whereas, SOA is dynamic in nature. so, verification is done among the service provider, client and independent service broker in a collaborative way.

SOA is composed of a set of web-service components that are distributed over the network. Hence testing location should be distributed, remote, multi-agent and multi-phase. Web-service support online regression testing where data collected dynamically and dynamic integration testing .

II. TESTING PERSPECTIVE

SOA based system must be tested from a service-by-service viewpoint. Testing of SOA as an aggregate of sub-system can be considered where interoperability between the different web services of the SOA is tested [2]. SOA impose different need and challenges to different stakeholder involved in the testing activities. For example stakeholder interested to make sure the originality of service behavior during its lifetime can be tested. Other testing perspectives are detailed below [6][7]:

a. Provider/Developer Perspective:- Services are build by service developer and he/she deliver both the interface and implementation of services. In order to release a high reliable services the service developer need to test the services. Test cases derived by the developer might not reflect real usage scenario. Service provider test the services that meet the service level agreement(SLA) agreed upon with the customer. Black box testing technique is used by the service provider [5][6][7].

b. Service Integrator's Perspective:- Here service testing is done during the design phase in order to explore the functional and non-functional assumptions. Service integrator utilizes existing services either to create composite services or to testing ensure that SOA based application meets the desired requirement [6].

d. Regression Testing- Regression Testing ensure that defects have not been introduced as a result of evolution. It check previously working web-services are still working correctly [2][5].

III. STRATEGIES FOR SOA REGRESSION TESTING

Rothermel and Harriod describe regression testing technique as follows [8]: Given a program P, a test set T used to test P and a modified version of P, P'. Find a way of making use of T, to gain sufficient confidence in the correctness of P'. Rothermel and Harriod outline the following approach of regression testing to solve this problem-

1. Identifies the changes that were made to P by creating a modified version P'.
2. Use the result of step 1 to select a set $T' \subseteq T$ a set of test set to execute on P', based on these modifications.
3. Use T' to test P'.
4. If necessary generate a new test set T'' a set of new tests for P'.
5. Use T'' to test software.

IV. SOA TESTING CHALLENGES

The main advantages of SOA based application are in terms of loose coupling and interoperability nature. But challenges arises when there are differences in the version of web-service standard, specification and differences in the protocol support. So, testing is the challenging task at this stage. Other various challenges in testing SOA are as follow [1][2][5]-

1. SOA is composed of web-service components dispersed over different hardware and operating system platform. This distributed nature of SOA must cover the different deployment configurations.
2. SOA is dynamic in that it implement adaptive behaviors such as adding new services, integrating new services and removing old services. Thus, performing effective regression testing is challenging task.
3. Combination of white box, black box and gray box testing is required for SOA. So, it is difficult for traditional testers.
4. SOA testing engineers need to understand all interfaces, the business need for service orientation and must have strong technical knowledge.
5. Unavailability of source code and structure of services make it difficult for white box testing.
6. SOA are heterogeneous in that there is incompatible interfaces, incompatible technologies, platform and programming languages. So, it requires multiple type of test engines.
7. Non-functional testing is difficult to determine a service workload parameters at service level agreement(SLA).
8. Web-services may involve many outside service providers who charge their service provided and it may be a high cost.
9. Creation of testing environment is challenge which is similar as deployment environment. Mirroring capabilities of deployed environment in a testing environment is very costly.

V. PROBLEM WITH EXISTING APPROACHES OF REGRESSION TESTING

Various existing regression testing approaches are as follows-

1. Code Based Regression Testing:- Code based regression testing can be applied effectively at unit level. But for larger or complex component it becomes difficult to manage all the information obtained from code. In some software systems many programming languages are used so more than one code based regression testing is necessary. This make the situation complex.
2. Specification Based Regression Testing:- In specification based regression testing there are many personal decision involved and it's make the selection criteria subjective. It is difficult to measure coverage and evaluate the completeness of the selected test suites.
3. Risk-Based Regression Testing:- Risk based regression testing focus on risk area of test suite. Though it does not guarantees the success of the test. Level of security is also less in risk based regression testing.

4. State Based Regression Testing :- State based regression testing used program dependence graph and control dependence graph. But, problem is that no edges and nodes be change as per the dependence

graph. Due to these problems the existing approach of model based regression testing is more familiar for SOA based application testing. In next section we use a model based testing approach for the generation of test cases in the context of our case study name "Online Shopping System".

VI. CASE STUDY: ONLINE SHOPPING SYSTEM

In this section we provide a case study in which we use UML use case and activity diagram to generate test cases. In use case diagram there is a flow of event which can help to generate test cases. Flow of event have two component first is 'Basic Flow of Event' which cover what normally happens when use case is performed and second is 'Alternate Flow of Event' cover the optional and unexceptional characteristics relative to normal behaviour.

Here we chosen an 'Online Shopping System', a web-service to apply our method in which we could identify the test cases and address web-service regression testing. Figure1.1 is the use case diagram of "Online Shopping System". Here different use cases represent the different services which is provided by different service provider. For e.g.- 'choose payment option' use case represent the payment service which is provided by payment service provider and this service make the user to pay the product cost by choosing appropriate payment option. Similarly, 'Make purchase history enquiry' represent the service provided by shipping service provider and with the help of this service user can able to track their purchase product. So, here we see that different services are provided by the different service provider and these services are working in collaborate manner. Use case diagram helps us to generate various use case scenario. Some scenario interoperate multiple events.

We can generate various scenario of use case diagram of "Online Shopping System" with the help of 'Basic Flow of Event (B)' and ' Alternate Flow of Event (A)'. Possible scenarios generating from the use case diagram of online shopping system are as follow:-

Scenario 1: Home Page
Scenario 2: Successful Login/Sign in
Scenario 3: Unregistered User
Scenario 4: Invalid User Id/ Password
Scenario 5: Item Availability
Scenario 6: No Item Found
Scenario 7: Item Information
Scenario 8: Purchased Item Info
Scenario 9: Shipping Address Updating
Scenario 10: Getting Confirmation Number
Scenario 11: Purchase History Enquiry
Scenario 12: Cancel Order

Here 'Shipping Address Updating' is the scenario where multiple events interoperating. As shown in figure:-

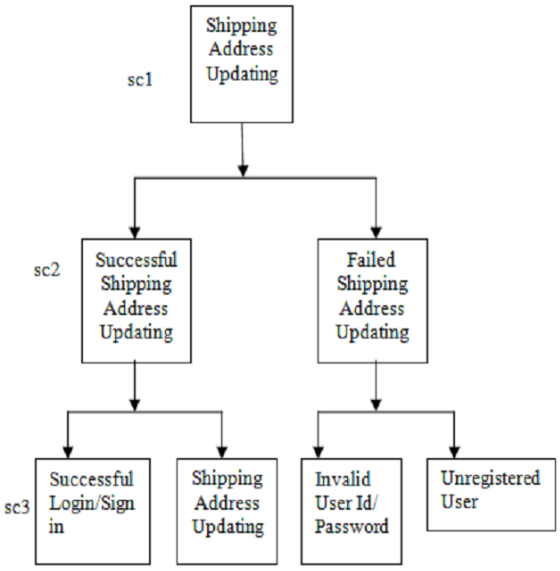

Figure 1. Multiple events interoperating scenario

Here sc1, sc2 and sc3 represent the scenario/ scenarios at level1, level2 and level3 respectively. Test case generation for this type of scenario is based on the combination of ' Basic Flow of Event (B)' and ' Alternate Flow of Event (A)'. For e.g.- Test case generation of ' Successful Login/Sign in' scenario is based on flow of event set { B1,B2,A1}. Similarly, test case generation of ' Invalid User Id/Password is based on flow of event set { B1, B2, A1, A2} and so on. Here, flow of events introduce what the system does when a user interact with the system.

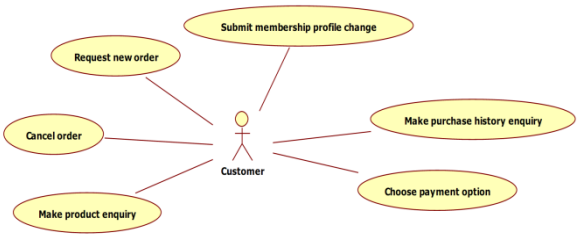

Figure1.1. use case diagram of online shopping System

1. ONLINE SHOPPING SYSTEM USE CASE DIAGRAM

Now we address a 'Basic Flow of Event(B)' of this use case which is detailed below.
B1. In the browser user enter a corresponding website address from which the user want to purchase an item/items. System show the home page of the website.
B2. User click on login link of that website and enter valid email address/user id and password. System confrm correct login and main page of the website it's displayed.
B3. User now search an item/items by typing item name in a search box or choose an item from a list of choice then system display availability information of the selected items.

B4. Now user select category of the item and get detail information about that item.

B5. User put the item in the shopping cart and shopping cart contents are displayed.

B6. Now user select a purchase option and enter the shipping address.

B7. User confrm shipping address and choose payment option (credit card, debit card, net banking/ cash on delivery).

B8. User conform payment option and provide final conformation to the system.

B9. System returns a confirmation number. Here B={ B1, B2, B3, B4, B5, B6, B7, B8, B9}. Now we address a 'Alternate Flow of Event' of the use case diagram of 'Online Shopping System'.

A1. System display a message 'Unregistered User' for the user who are not registered.

A2. System sign not possible for wrong user id/password.

A3. When a user search an item then system display unavailability information if the item is not in the stock.

A4. User can buy more item and item add in the shopping cart.

A5. User can update their membership profile.

A6. User can be able to make purchase history enquiry.

A7. Order can be canceled. Here A={ A1, A2, A3, A4, A5, A6, A7}.

2. USE CASE SCENARIO

After getting the information about 'Basic Flow of Event' and 'Alternate Flow of Event' in the context of our case study now we see the different possible scenarios that can be used to generate test cases. Table1. shows the use case scenario.

Scenario 1	B1								
Scenario 2	B1	B2							
Scenario 3	B1	B2	A1						
Scenario 4	B1	B2	A1	A2					
Scenario 5	B1	B2	B3						
Scenario 6	B1	B2	B3	A3					
Scenario 7	B1	B2	B3	B4					
Scenario 8	B1	B2	B3	B4	B5	A4			
Scenario 9	B1	B2	B3	B4	B5	B6	A5		
Scenario 10	B1	B2	B3	B4	B5	B6	B7	B8	B9
Scenario 11	A6								
Scenario 12	A7								

Table1. use case scenario

3. TEST CASE IDENTIFICATION

After identification of all possible scenario now we have to identify the test cases. Each scenario contains at least one test case. Here we form a test case matrix.

Table2 shows the test case matrix of 'Online Shopping System' using use case scenario. Where

Test Case Id= TCI
Scenario= SNO
Valid Web Address= VWA
Valid User Id= VUI
Valid Password= VP
Item Selection= IS
Item Availability Info= IAI
Shipping Address= SA
Payment Mode= PM
Valid Credit Card Info.= VCI
Order Status= OS
Expected Result= ER
In Table 2:

* denotes valid test case for that scenario NT denotes invalid test case for that scenario. N/A denotes test case is not applicable.

TCI	SNO	VWA	VUI	VP	IS	IAI	SA	PM	VCI	OS	ER
T1	Home page	*	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Home Page Display
T2	Successful Sign In	*	*	*	N/A	N/A	N/A	N/A	N/A	N/A	User Login Page Display
T3	Unregistered User	*	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	System Display Message Like Unregistered User
T4	Invalid User Id/ Password	*	NT	NT	N/A	N/A	N/A	N/A	N/A	N/A	Display Error Message
T5	Item Availability	*	*	*	*	*	N/A	N/A	N/A	N/A	Display Availability Info of The Item
T6	No Item Found	*	*	*	NT	NT	N/A	N/A	N/A	N/A	Display Message Like Item Not Found
T7	Item Information	*	*	*	*	*	N/A	N/A	N/A	N/A	Detailed Information of The Item
T8	Purchase Item Info.	*	*	*	*	*	N/A	N/A	N/A	N/A	Display List of Item to Be Purchased
T9	Shipping Address Updating	*	*	*	*	*	*	N/A	N/A	N/A	Conformation of Shipping Address
T10	Getting Conformation Number	*	*	*	*	*	*	*	*	*	Unique Conformation number is provided
T11	Purchase History Enquiry	*	*	*	*	*	*	*	*	*	Display All Purchase Item With Detail Info.
T12	Cancel Order	*	*	*	*	N/A	N/A	N/A	N/A	*	Display Message Like Your Order Is Cancel

Table 2: Test Case Matrix For 'Online Shopping System' Using Use Case Scenario

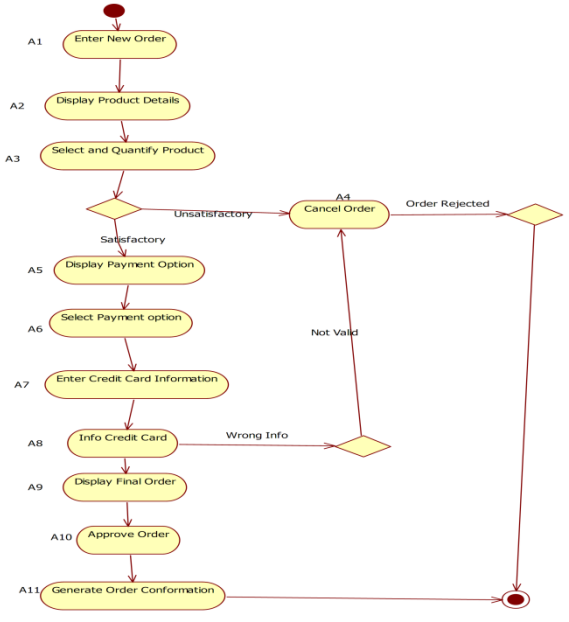

Figure2: Activity Diagram of Request new order use case

After identifying the test case matrix of use case scenario here we focus on individual use case and make an activity diagram for that use case to know the activity coverage area and test case that affect the activity related to our web-service name 'Online Shopping System'.

In activity diagram nodes represent the various user actions, conditions and system outputs. Edges represent transition from node-to-node. In this activity diagram(Figure2) we give a unique node version say A1,A2 etc to each node i.e., each node having a unique node version. This unique node version help us to identify the changes made to the activity diagram. This changes happen due to changes made to the software requirement. Whenever the changes happens due to changes made in software requirement the corresponding node version of activity diagram be change as node version when A1 changes to node version A1.1. Changes occur in activity diagram by addition of a new node, deleting the existing node, modifying the existing node and shifting of an existing node.

Here, shifting means Deletion followed by addition of a node. So, it is easy to identify the change node based on node version. For this change node we check our test suit T and if the changes nodes is traced by the test cases present in the test suit T we add that test cases in the new test suit T'. If the node is not traced by the test cases present in the test suit T than we follow regression testing technique given by Rothermel and Harriod[8]. After identifying the test cases we generate activity path of each test cases. The main advantage of activity coverage path is that it gives a direct view to determines the activity coverage area of each test cases and based on this activity coverage area we can able to determine the priority of each test cases. With the help of activity coverage path we can also find out where the changes nodes are present in the activity coverage path. Based on it we can find out whether there is any dependency changing occur when the corresponding node changes. Here the changes node represent the unique node version and so it minimizes the complexity of activity path to identifying the changes node and their dependency in the activity path.

If AP is the original activity path and AP' is the updated activity path that is formed due to changes occur in the node version than for identifying the changes nodes in the AP' we do the following:-

1. For each activity node present in the original activity path AP identifying the corresponding activity node in the updated activity path AP'.
2. If the node is not found in the updated activity path AP' this means, that node is removed in the updated activity path and the corresponding node in place of removed node consider as changed node.

Based on this activity diagram now we consider following test cases:

T1: Item Selection
T2: Item Availability Information
T3: Payment Mode
T4: Credit Card Info.
T5: Order Status

Table 3 Shows the activity path of all test cases (T1 to T5). Here each column represent test cases and each rows represent activity. In this table:
* denotes valid activity for the related test case. NT denotes invalid activity for the related test case.

Activity	T1	T2	T3	T4	T5
A1	*	*	NT	NT	NT
A2	*	*	NT	NT	NT
A3	*	*	NT	NT	NT
A4	*	*	NT	NT	NT
A5	*	*	*	NT	NT
A6	*	*	*	NT	NT
A7	*	*	*	*	NT
A8	*	*	*	*	NT
A9	*	*	*	*	*
A10	*	*	*	*	*
A11	*	*	*	*	*

Table 3: Activity coverage for T1 to T5

Now we draw activity path for test cases. First we draw an activity path for test case T1 and T2. After that we draw an activity path for test case T3,T4 and T5 respectively.

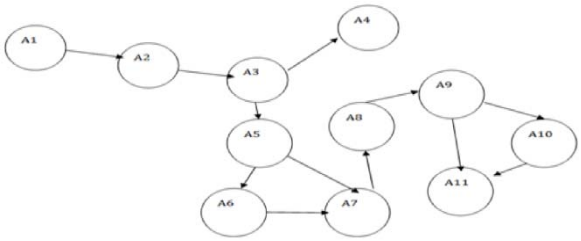

Figure 3: Activity path for test case T1 and T2

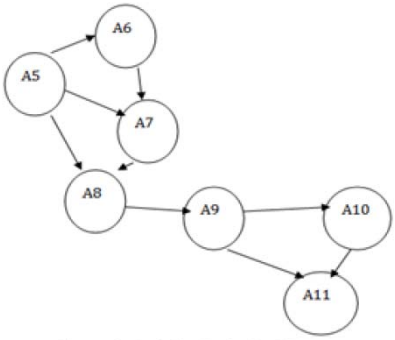

Figure 4: Activity Path for T3.

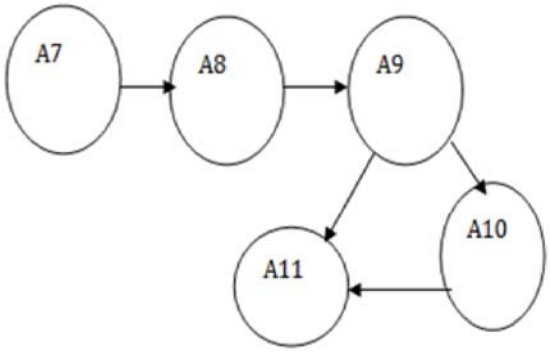

Figure 5: Activity Path for T4.

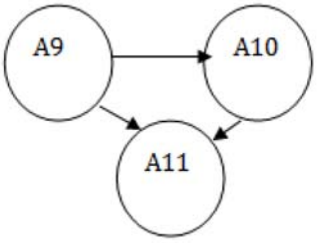

Figure 6: Activity Path for T5.

After getting the activity path of all test cases (T1 to T5) we found that T1 and T2 cover most activity path and by identifying the activity coverage path we can prioritized test cases as T1, T2, T3, T4 and T5.

VII. COMPARISON

We find paper ' UML based web-service regression testing using test cases: A Case Study' given by Rajani Kanta Mohanty et al[4] is more related to our work. Our paper also take model based approach for generation of test cases. In this paper we add activity diagram for individual use case to get more clear picture of test cases generation and activity coverage path of that test cases. Here we consider only one 'Use Case' for understanding of our approach because same concept apply on all other 'Use Cases'. After getting the activity path of all test cases our work prioritized the test cases based on activity coverage path of all test cases. Our paper also highlight the scenarios with multi events interoperating. Our paper introduces the advantages of activity path and provide a solution to identify the changes node in the activity path.

VIII. CONCLUSION AND FUTURE SCOPE

In this paper we focus on UML use case diagram and activity diagram to generate test cases and prioritize test cases in the context of our case study "Online Shopping System". After comparing various approaches and issues we found out that model based regression testing of SOA based application is the need of the hour. We have concentrated on generating test cases from use case diagram considering the flow of event. The test case prioritization is based on the activity path considering the UML activity diagram. In future we will focus on more effective method for selection of test cases that reduces redundancy in test case selection for regression testing.

REFERENCES

[1] Ebrahim Shamsoddin-Motlagh, "A survey of service-oriented architecture system testing", international journal of software engineering and application(IJSEA), vol.3, no-6, november-2012.

[2] Youssef Bassil, " Distributed, cross-platform and regression testing architecture for service-oriented architecture", Advance in computer science and its applications(ACSA) ISSN: 2166-2924, vol. 1 no. 1, March 2012.

[3] Ed Morris, William Anderson, Sriram Bala, David Camey, John Morley, Patrik Place, Soumya simanta, "Testing in service-oriented environment" Technical Report CMU/SEI-2010-TR-011 ESC-TR-2010-011.

[4] Rajani kanata Mohanty, Binod Kumar Pattanayak and Durga prasad Mohapatra, " UML based web service regression testing using test cases: A Case Study. ARPN journal of engineering and applied sciences, ISSN 1819-6608,vol.7,no.11, November2012

[5] Poonkavithai Kalamegam and Zayaraz Godandapani, " A survey on testing SOA built using web-services", International journel of software engineering and its applications, vol. 6, no. 4, October-2012.

[6] Rajani kanata Mohanty, Binod Kumar Pattanayak, Bhagabat Puthal, Durga prasad Mohapatra, " A road map of regression testing of service-oriented architecture(SOA) based applications", Journel of theoretical and applied information technology, volume 36 no. 1 15 February 2012.

[7] Massimiliano Di Penta, Marcello Bruno and Gerardo Canfora, "Web-service regression testing", RCOST-Research centre on software technology- University of Sannio Palazzo ex poste,via Traiano 82100 Benevento, Itly May-2007.

[8] Orest Pilskalns and Gunay Uyan, Anneliese Andrews," Regression testing UML design", 22nd IEEE international conference on software maintenance(ICSM-06)

[9] Gaurish Hattangadi, " A practitioners guide to modern SOA testing", infosys- whitepaper, July 2011.

[10] Thomas Erl," Service-Oriented Architecture Concept technology and Design" August-2005, ISBN:-0-13-185858-0.

◈◈◈