ISSN 2277-3061

# Finding a Relative Testing approach for Service Oriented Architecture

[1]Bharat Choudhary, [2]Vineet Richhariy, [3]Shweta Shrivastava

[1]Department of Computer Sc.&Engg, Laxmi Narain College of Technology, Bhopal Madhya Pradesh India
[2]HOD Department of Computer Sc &Engg, Laxmi Narain College of Technology Bhopal, Madhya Pradesh,
[3] Asst. Prof,Department of Computer Sc. & Engg.Laxmi Narain College of Technology Bhopal, Madhya Pradesh India
[1]E-mail: hi.bharat2002@gmail.com, [2]E-mail:vineetrich100@gmail.com,
[3]E-mail: shwetashri.26@gmail.com

## Abstract

A SOA (Service Oriented Architecture) is an enterprise-scale IT architecture for linking resources on demand. In a SOA, resources are made available to participants in a value net, enterprise, and line of business. Service-oriented applications can be expensive to test because services are hosted remotely, are potentially shared among many users, and may have costs associated with their invocation. For interactive web services development SOA is a good approach. To find the desire outcomes from this architecture, we have to test this architecture than we can develop desire services from this architecture. There lots of works has done to test this architecture, like unit and integration testing are used to test this architecture but these methods cannot give hundred present testing capacity. In this paper we are concentrating to find a right approach to test the SOA. Regression testing is a better approach, rather than the previous methods.

**Keywords:** SOA; SOA Testing; Web Services; Unit and integration testing; regression testing.

Council for Innovative Research

Peer Review Research Publishing System

**Journal:** INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol 12, No.4

editor@cirworld.com

www.cirworld.com, member.cirworld.com

ISSN 2277-3061

## I. INTRODUCTION

Service-oriented computing promises greater flexibility and efficiency in application development by enabling applications to be composed using third-party services. The term service-oriented architecture refers to a style of building reliable distributed systems that deliver functionality as services, with the additional emphasis on loose coupling between interacting services. Initiatives for service-oriented architecture (SOA) and on demand business are being adopted at various corporations to meet the operating challenges of business in the 21st century. Currently, the primary focus is to apply SOA concepts incrementally to existing information technology (IT) systems to exploit short-term business benefits. SOA is a design pattern which is composed of loosely coupled, discoverable, reusable, inter-operable platform agnostic services in which each of these services follow a well-defined standard [8]. Enterprises who are in the early stages of their SOA initiatives or those contemplating starting a SOA initiative should also address the impacts of SOA as it pertains to testing. Success of any deployment in an enterprise depends on the quality assurance process undertaken. SOA testing is quite different from traditional testing. It requires its own type of test architecture and tester skills. SOA testing spans several levels taking into account both functional and non-functional aspects of the individual services and the inter-enterprise federations of systems. A comprehensive and maintainable testing methodology is critical for the successful functional testing of a service-oriented system.

Regression testing help identify changes between a selected product release and a previous release of the product called a baseline. A baseline is recorded snapshot of desirable product behaviour. Regression testing is also known as validation testing and provides a consistent, repeatable validation of each change to an application under development or being modified. Each time a defect is fixed, the potential exists to inadvertently introduce new errors, problems, and defects. An element of uncertainty is introduced about ability of the application to repeat everything that went right up to the point of failure.[2] It is important to understand that regression testing doesn't test that a specific defect has been fixed. Regression testing tests that the rest of the application up to the point or repair was not adversely affected by the fix.

## II. PRESENTATION OF RESEARCH

Mainly our research has been exploratory on the following points:

- What do you know about Regression Testing selection method?
- How the regression testing method applied in practice?
- How can we improve the current testing methods with the help of regression testing approaches?
- What is the best approach in regression testing method?

In order to address the first and fourth questions two different kinds of systematic literature reviews have been conducted: a systematic review on regression test selection and a systematic mapping on software product line testing [5]. To justify this approach we also give a comparison table of different regression testing approaches.

## III. PRINCIPAL OF SOA

There we are explaining some fundamental principles that a Service-oriented Architecture (SOA) should expose.

### A. Explicit Boundaries

Services are inextricably tied to messaging in that the only way into and out of a service are through messages. A service invocation should as a general pattern not rely on a shared context; instead service invocations should be modelled as stateless.

### B. Shared Contract and Schema, not Class

Starting from a service description (a contract), both a service consumer and a service provider should have everything they need to consume or provide the service. Following the principle of loose coupling, a service provider cannot rely on the consumer's ability to reuse any code that it provides in its own environment; after all, it might be using a different development or runtime environment.

### C. Autonomous

Related to the explicit boundaries principle, a service is autonomous in that its only relation to the outside world at least from the SOA perspective is through its interface. In particular, it must be possible to change a service's runtime environment, e.g. from a lightweight prototype implementation to a full-blown, application server-based collection of collaborating components, without any effect on its consumers. Services can be changed and deployed, versioned and managed independently of each other. A service provider cannot rely on the ability of its consumers to quickly adapt to a new version of the service; some of them might not even be able, or willing, to adapt to a new version of a service interface at all.

### D. Wire formats, not Programming Language APIs

Services are exposed using a specific wire format that needs to be supported. This principle is strongly related to the first two principles, but introduces a new perspective: To ensure the utmost accessibility a service must be accessible from any platform that supports the exchange of messages adhering to the service interface as long as the interaction conforms to the policy defined for the service. For example, it is a useful test for conformance to this principle to consider whether it is

possible to consume or provide a specific service from a mainstream dynamic programming language. This consideration can serve as a litmus test to assess whether the following criteria are met:

- All message formats are described using an open standard, or a human readable description.
- The semantics and syntax for additional information necessary for successful communication, such as headers for purposes such as security or reliability, follow a public specification or standard.
- The semantics and syntax for additional information necessary for successful communication, such as headers for purposes such as security or reliability, follow a public specification or standard.
- At least one of the transport (or transfer) protocols used to interact with the service is a standard network protocol.

**E. Loosely coupled**

Most SOA proponents will agree that loose coupling is an important concept. Unfortunately, there are many different opinions about the characteristics that make a system "loosely coupled". There are multiple dimensions in which a system can be loosely or tightly coupled. Dimensions include:

- Time: When participants are loosely coupled in time, they don't have to be up and running at the same time to communicate.
- Location: If participants query for the address of participants they intend to communicate with, the location can change without having to re-program, reconfigure or even restart the communication partners.
- Type: In an analogy to the concept of static vs. dynamic and weak vs. strong typing in programming languages, a participant can either rely on all or only on parts of a document structure to perform its work.
- Cardinality: There may be a 1:1-relationship between service consumers and service providers, especially in cases where a request/response interaction takes place or an explicit message queue is used.
- Interface: Participants may require adherence to a service-specific interface or they may support a generic interface. If a generic interface is used, all participants consuming this generic interface can interact with all participants providing it.[7]

## IV. SOA LAYERS

A SOA generally has a highly symmetrical architecture on client side and server side, as it can (also) be found in many modern distributed object middleware systems.

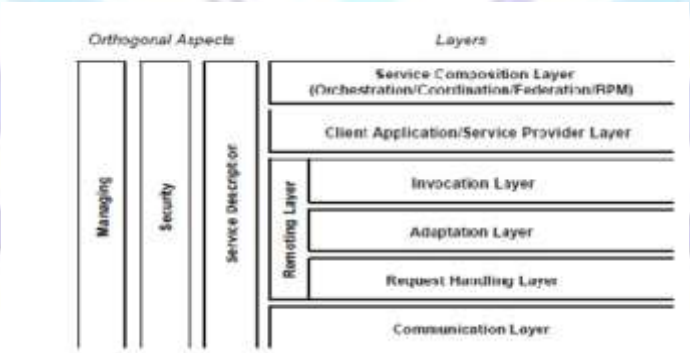


Fig1: Client and server SOA layers [10]

From the above diagram we can easily identified that SOA has following layers.

**A. Service composition:**

The top-level layer of a SOA deals with the composition of services and is optional. At this layer service orchestration, service coordination, service federation, or business process management (BPM) functionalities are implemented

**B. Client application/service provider:**

This layer consists of clients that perform invocations and the actual implementations of the services.

**C. Remoting:**

This layer implements the middleware functionalities of a SOA (for instance a Web services framework).[9] Usually, these details of the client side and the server side are hidden in BROKER architecture: a BROKER hides and mediates all

communication between the objects or components of a system. The remote layer consists itself of three layers: invocation, adaptation, request handling.

**D. Communication:**

The communication layer is responsible for defining the basic message flow and managing the operating system resources, such as connections, handles, or threads.

## V. WEB SERVICES

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or Web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML. It use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. The basic Web services platform is XML and HTTP. All the standard Web Services works using following components:

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

By the above discussion we can define the web services in single sentence. “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.” a web service is a basically a 'method' or 'function'. SOAP is the 'instructions' and 'data' to this method. It will outline data types, and possibly a bunch of data as well. It is an XML format.

## VI. REGRESSION TESTING APPROACHS

Testing of SOA is a major task because only creating a better services is not sufficient for and system. Without testing we cannot say that our service is working properly, according to the desired requirement. There are such testing attempts for SOA, like unit testing and integration testing has already done but these are not sufficient to test the web services because some instant changes has performed in the web services and they cannot test each time.

Regression Testing in SOA is one of the major matters of concern because of the dynamic nature of the Service binding and the adaptability to accommodate the changed requirement. The actual configuration of the service is known only at run time, so It becomes very complex to verify whether the changes made in earlier version of the system are correct or not and it does not affect the functionality and performance of the existing system.

There are many approaches of the regression testing which are described as follows. Later on this paper we also compare these methods in a comparison table. The methods are as follows:

- Selective Regression Testing
- Risk Based Regression Testing
- State Based Regression Testing
- Code Based Regression Testing
- Model Based Regression Testing

Various approaches have addressed the need and methodologies of regression testing in traditional and Object- Oriented software development. SOA based system development caters to the agility of business strategies. In code based approach, regression testing is done for code changes. Also it can cater to the need for SOA-based regression testing when a service is upgraded with new versions. A combination of various UML diagrams can help us to get the dependency graph like component and state chart diagram. [6]

## VII. WHY THE REGRESSION TESTING

Regression test helps to identify changes between a selected product release and a previous release of the product-called a baseline. A baseline is recorded snapshot of desirable product behaviour [3]. This expected behaviour is then used to ensure that nothing has been broken in the system as a result of changes introduced in a program module. Establishing a regression testing framework is crucial for building reliable and stable software products. Regression testing is usually performed by running some, or all of the test cases created to test modification in previous versions of the software.

## VIII. SCENARIOS FOR FINDING FAULT

The process of the regression testing for Service is not only including the external interface’s testing, but also including testing the basic services that web Service relied on.

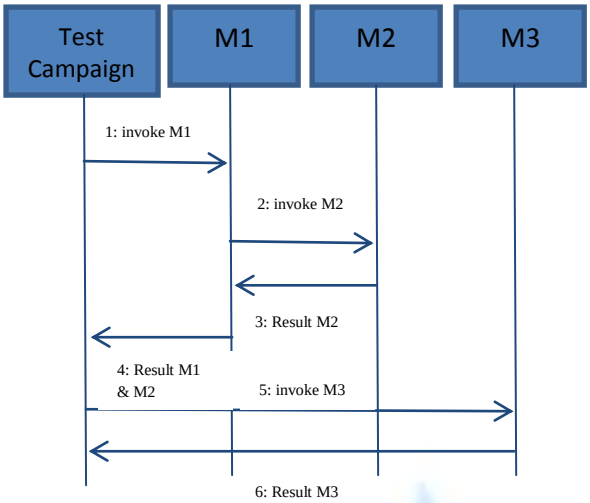


**Fig: Composite Service testing scenarios**

In the above diagram we show the testing scenarios of a web service. Now it is describing in following points.

1. M1 will call first.
2. After calling M1, will call M2 (also in implementation part m1 is call M2)
3. After invoking M2 it generate the self-result may be normal/abnormal.
4. After that M1& M2 both return the result normal/abnormal.
5. Now call the M3.
6. And it will return the result may be normal/abnormal

It's important to locating fault in a module. Many testing symbols are produced in the test script. These symbols record the testing information, including the test steps number and the interface name of a service. The symbol is relate to the normal information and exception information that be collected during execution.

$$(\mathrm{i})\ \exists i\ (\neg \mathrm{A}(\mathrm{i})) \rightarrow \neg CS$$

$$(\mathrm{ii})\ \forall i \mathrm{A}(\mathrm{i}) \rightarrow CS$$

Expression (1) is equivalence to expression (2), which is read as "A web service is no fault if all of basic service is correct under tested."

Means that if all the modules are correctly answered or produce the desire result then there is no errors/faults in that web service.

- R (i) is a Boolean value. This value obtain from the comparison of the expected value (EV) and the obtain result (OR).
- So EV(i) is the expected value of the implementation of the basic module. *i* represents the number of modules present in that particular service.
- OR(i) is the obtain value after testing the different module present in that particular WS.
- WS is the collection of different modules ($M_i$) WS = R(1) ∩ R(2) ∩ R(3) ……R(n)
- RE(i) is the error information of the implementation of the basic modules $M_i$.

**A. TEST PROCESS**

Now we will define the test process of the regression testing method. It is finite state machine. The test process is a 5 – tuple TP = ($Q,\Sigma$, d, $q_0$, S ) . Where TP is the testing process.

1. Q = ($q_0$, $q_1$, $q_2$, $q_3$, $q_4$, $q_5$) it is finite set of state and the different state are define as follows.
    - $q_0$ is the initial state.
    - $q_1$ is a state the parsing script input data
    - $q_2$ is a state that parsing script needs to display the value and type
    - $q_3$ is a state that parsing the script to perform the web service.

- $q_4$ is a state that parsing the script to compare results of the implementation and expectations
- $q_5$ is the end of the state.

2. Σ = (obj, Echo, Invoke, Check, End)
    - Obj denotes input data
    - Echo denotes the value and type
    - Invoke denotes implementation services
    - Check denotes compare performance results and expectations
    - End denotes the process has finished
3. d: It is state transition function

    (d: Q × Σ →□ □ Q )
4. S: This is termination of the state set(final state set)

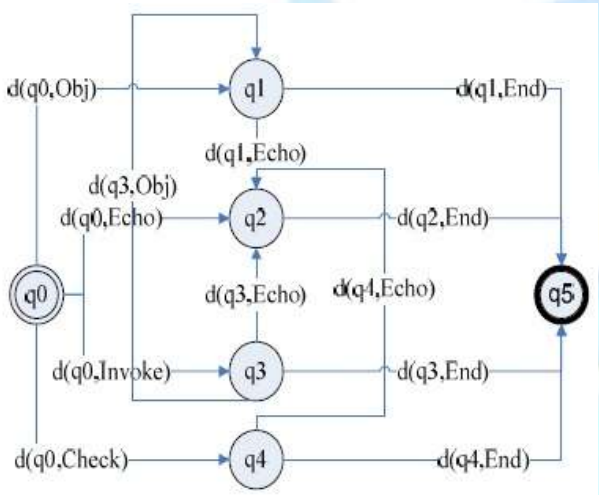


**Fig: Showing the testing process**

**B. TEST RESULT**

The test result is an XML document, which contains the test results returned by the implementation of test cases. An executive order may contain multiple test cases; each test case also contains a number of different test steps. Between the implementation of test cases are independent, (that is, a test case implementation of the failure would not affect the implementation of another test cases), but the testing step is to influence each other (in front of the impact test results of steps behind the test steps, in front of the test step will lead to the back of abnormal test steps cannot be implemented).

Test result is a 3-triple Test Result = (H, TCI, TSI)

- H refers to the information of the test results of the header, used to store a lot of information of the implementation;
- TCI refers to the information of the test cases;
- TSI refers to the information of the test steps.

TSI is a very important to the test results, It's not only records the information of each steps of the test steps, but also return the value Key tag identification for each test steps, through which you can locate the Composite Service's fault if it's exist.

ISSN 2277-3061

## IX. COMPARISON TABLE FOR DIFFERENT APPROACHES OF REGRESSION TESTING

**TABLE 1: COMPARISON OF REGRESSION TESTING APPROACHES**

| Approach of Regression Testing(RT) | Features of the Technique | Advantages | Disadvantages | References |
|---|---|---|---|---|
| Selective RT | • Algorithms construct control flow graphs for a procedure or program and its modified version and use these graphs to select tests that execute changed code from the original test suit. | 1. Economic, the cost of software maintenance dominates the overall cost of software.<br>2. Software quality.<br>3. Take, less than the reset time. | 1. Problem of selecting test from existing test suit.<br>2. Problem of determining where additional test may be required. | [11] |
| Risk based RT | • Prioritizes the tests of features and functions and based on the risk of their failure.<br>• Test technique such as boundary value analysis and state transition testing aim to find the area's most likely to be defective.<br>• Focus on test cases which test the risky area. | 1. It supports the scalability feature and it is also easy to implement. | 1. Ambiguity: ambiguous description can lead to incorrect or conflicting implementations.<br>2. In this approach level of safety is very less. | [14] |
| Code based RT | • In this approach regression is done for code changes part not consider the other part of the service. | 1. This technique is good for unit testing. | 1. It tests (traverse) only code modification parts.<br>2. It does not support scalability. | [14] |
| State based RT | • State based RT uses Program dependence graph and control dependence graph. | 1. In this approach DFS and BFS technique can easily be applied for traversing and path visibility. | 1. There, number of nodes and edges may be changed according to the dependence graph because of that complexity increases. | [1] |
| Model based RT | • WSDL file of the service is required and it is easy to implement.<br>• UML class and sequence diagram are used. | 1. This technique requires less effort.<br>2. It has better complexity management.<br>3. Testing activity can be started from the beginning phase of the software development. | 1. Identification of modification is challenging due to interdependency among models. | [12, 1] |

## X. OBSERVATION

From the above table we can easily see the merits and demerits of different approaches. Each has its own positive points and some negative points. Based on our observation we can conclude that the Model Based RT is best among others because in this approach we can give a better testing approach for each modification on the web services. So we are choosing the Model Based Regression Testing for our testing approach which is based on the finding a best testing approach for Service Oriented Architecture.

## XI. SCOPE OF THE SURVEY

To the best of our knowledge no survey work has been published on Regression Testing of SOA. Hence our focus in

this paper is to present a comparison of the various testing methodologies and advantages and disadvantages of these various approaches of regression testing. The survey will help to develop an automated testing approach which would be more relevant considering scenario and challenges posed by SOA based application.

There one more thing is that there are many testing tools which are based on the regression testing approaches like Soatest, TestMaker testing tool, SoapUI, E- Test, etc are available. By using these tools we can find our approach.

## XII. CONCLUSION

Regression Testing in Service-Oriented Architecture is one of the important testing activities which require a lot of effort to test the system because of the dynamic nature of the system. In this paper we made an effort to put the different issues involved with Regression Testing and the analysis among different approaches which will be helpful for the automation of the SOA based Regression Testing and by using the model base regression testing we give an approach to test the Service-Oriented Architecture applications.

## REFERENCES


[1] Ul-ann Farooq, Q., Iqbal, M. Z. Z., Malik, Z. I., and Riebisch, M. A model-based regression testing approach for evolving software systems with flexible tool support. Engineering of Computer-Based Systems, IEEE International Conference on the 0 (2010), 41-49.

[2] John E. Bentley, Wachovia Bank and Charlotte NC, “Software Testing Fundamentals—Concepts, Roles, and Terminology”, Paper 141-30.

[3] Soa regression testing. Code Project. www.codeproject.com.

[4] Endo, A. T., Linschulte, M., da Silva Simao, A., and do Rocio Senger de Souza, S. Event- and coverage-based testing of web services. Secure Software Integration and Reliability Improvement Companion, IEEE International Conference on 0 (2010), 62-69

[5] E. Engström and P. Runeson, ”Software product line testing – a systematic mapping study,”

[6] Bhuyan, Prachet; Kashyap, Chandra Prakash; Mohapatra, Durga Prasad, A survey of regression Testing on service oriented Architecture, International Journal of Computer Applications; Apr2012, Vol. 44, p22.

[7] SOA Principal, www. 10-soa-principles.html

[8] R. Heckel and M. Lohmann, “Towards contract-based testing of web services”, Theoretical Computer Science, vol. 116, (2005), pp. 145-156.

[9] M. Kircher and P. Jain. Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management. J. Wiley and Sons Ltd., 2004.

[10] Uwe Zdun, Carsten Hentrich, Wil M.P. van der Aalst, “A Survey of Patterns for Service-Oriented Architectures”, Internet Protocol Technology.

[11] Rothermel, G., and Harrold, M. J. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology 6, 2 (1997), 173-210.

[12] Naslavsky, L., Ziv, H., and Richardson, D. J. A model-based regression test selection technique. 2009 IEEE International Conference on Software Maintenance (2009), 515-518.

[13] E. Engström, P. Runeson and G. Wikstrand, “An empirical evaluation of regression testing based on fix-cache recommendations,” Proc.

Third IEEE International Conference on Software Testing, Verification and Validation (ICST 2010), in press.

[14] Chen, Y., and Probert, R. L. A Risk-based Regression Test Selection Strategy. In ISSRE 2003: Proceeding of the 14th IEEE International Symposium on Software Reliability Engineering (Nov. 2003), pp. 305-306.

[15] H. M. Sneed and S. Huang. WSDLTest – A tool for testing web services. In Proc. of the Intl. Symp. on Web Site Evolution, pages 14–21, Sept. 2006.

[16] E. Martin, S. Basu, and T. Xie. Automated testing and response analysis of web services. In Proc. of the Intl. Conf. on Web Services, pages 647–654, July 2007.

[17] W.-T. Tsai, X. Wei, Y. Chen, and R. Paul. A robust testing framework for verifying web services by completeness and consistency analysis. In Proc. of the Intl. Workshop on Service-Oriented Syst. Eng., pages 151–158, Oct. 2005.

[18] E. S. Motlagh, A survey of service oriented architecture systems testing, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012.